IMPROVED DEEP CONVOLUTIONAL NEURAL NETWORKS (DCNN) APPROACHES FOR COMPUTER VISION AND BIO-MEDICAL IMAGING

Dissertation

Submitted to

The School of Engineering of the

UNIVERSITY OF DAYTON

In Partial Fulfillment of the Requirements for

The Degree of

Doctor of Philosophy in Engineering

By

Md Zahangir Alom

Dayton, Ohio

December 2018

IMPROVED DEEP CONVOLUTIONAL NEURAL NETWORKS (DCNN) APPROACHES

FOR COMPUTER VISION AND BIO-MEDICAL IMAGING

Name: Alom, Md Zahangir

APPROVED BY:

Tarek M. Taha, Ph.D.
Advisor Committee Chairman
Professor, Department of Electrical and Computer Engineering

Vijayan K. Asari, Ph.D.
Committee Member
Professor, Department of Electrical and Computer Engineering

Raul Ordonez, Ph.D.
Committee Member
Professor, Department of Electrical and Computer Engineering

Muhammad Usman, Ph.D.
Committee Member
Associate Professor, Department of Mathematics

Robert J. Wilkens, Ph.D., P.E.
Associate Dean for Research and Innovation
Professor
School of Engineering

Eddy M. Rojas, Ph.D., M.A., P.E.
Dean
School of Engineering

ABSTRACT

IMPROVED DEEP CONVOLUTIONAL NEURAL NETWORKS (DCNN) APPROACHES FOR COMPUTER VISION AND BIO-MEDICAL IMAGING

Name: Alom, Md Zahangir
University of Dayton

Advisor: Dr. Tarek M. Taha

Deep learning is showing tremendous success in variety of application domains and demonstrates state-of-the-art performance over traditional machine learning approaches in the fields of Computer Vision, Speech Recognition, Natural Language Processing (NLP), Bio-Medical imaging, Computational Pathology, and many more. This thesis presents several improved Deep Convolutional Neural Network (DCNN) models including the Inception Recurrent Convolutional Neural Network (IRCNN) and Inception Recurrent Residual Convolutional Neural Networks (IRRCNN), a Recurrent U-Net (RU-Net), a Recurrent Residual U-Net (R2U-Net) model, a R2U-Net regression model, and a Densely Connected Recurrent Network (DCRN). These models are evaluated for classification, segmentation, and detection tasks in computer vision, Bio-medical imaging, and computational pathology applications. There are four key contribution areas in this thesis.

The first contribution area is the introduction of two improved DCNN models for classification tasks: IRCNN and IRRCNN, which utilize the power of the Recurrent Convolutional Neural Network (RCNN), the Inception Network, and the Residual Network (ResNet). In addition, we

have evaluated the impact of recurrent convolutional layers on DenseNet which is called Densely Connected Recurrent Network (DCRN). The performance of the IRCNN, DCRN, and IRRCNN models was investigated with a set of experiments and computer vision tasks where we used several publicly available datasets including MNIST, CIFAR 10, CIFAR 100, SVHN, CU3D–100, and Tiny ImageNet-200. The experimental results show that IRCNN, DCRN, and IRRCNN provide superior performance compared to the equivalent DCNN based methods including equivalent RCNN, ResNet, Inception V3, DenseNet, and Inception Residual Network (Inception V-4) with the same number of network parameters for different computer vision tasks.

The second contribution area is the introduction of two different models including a Recurrent U-Net and Recurrent Residual U-Net models, which are named RU-Net and R2U-Net respectively. The proposed models utilize the power of U-Net, the Residual Network, the RCNN, and U-Net for image segmentation tasks. These proposed architectures have several advantages for segmentation tasks over the existing DL methods. First, a residual unit helps when training deep architectures. Second, feature accumulation with recurrent residual convolutional layers ensures better feature representation for segmentation tasks. Third, it allows us to design better U-Net architecture with the same number of network parameters with better performance for medical image segmentation. The proposed models are tested on three benchmark datasets for blood vessel segmentation in retina images, skin cancer segmentation, and lung lesion segmentation. The experimental results show superior performance on segmentation tasks compared to equivalent models including SegNet, U-Net and Residual U-Net (ResU-Net) in different Bio-medical segmentation tasks.

The third contribution area is the introduction of an R2U-Net based regression model which is named University of Dayton Network (UD-Net) and is used for end-to-end detection tasks in digital pathology. To generalize these advanced DCNN models, we have applied classification, segmentation, and detection tasks in Digital Pathology Image Analysis (DPIA) including: microscopic blood cell classification, Breast Cancer Classification (BCC), invasive ductal

carcinoma detection, and lymphoma classification, nuclei segmentation, epithelium segmentation, tubule segmentation, lymphocyte detection, and mitosis detection. The experiments have been conducted on different publicly available datasets and evaluated with different performance metrics. The results demonstrate superior performance compared to existing DCNN based methods.

The fourth contribution area is the introduction of an image reconstruction technique using Convolutional Sparse Coding (CSC) on IBM's TrueNorth Neuromorphic computing system and the results demonstrate promising sparse reconstructions for two different benchmarks: MNIST and CIFAR-10. In 2016, IBM's release of a deep learning framework for DCNNs called Energy Efficient Deep Neuromorphic Networks (EEDN). EEDN shows promise for delivering high accuracies across different benchmark while consuming very low power using IBM's TrueNorth chip. We have empirically evaluated the performance of different DCNN architectures implemented within the EEDN framework to discover the most efficient way to implement DCNN models for object classification tasks using the TrueNorth system. The results show that for datasets with large numbers of classes, wider networks perform better when compared to deep networks comprised of nearly the same core complexity on IBM's TrueNorth system. In addition, we have proposed an effective quantization approach for Recurrent Neural Networks (RNN): Long Short-Term Memory (SLTM), Gated Recurrent Unit (GRU), and Convolutional LSTM (ConvLSTM). Furthermore, an NP-hard optimization problem called Quadratic Unconstrained Binary Optimization (QUBO) has solved with vanilla RNN on IBM's Neuromorphic computing system.

My respective parents – Md Abu Bakker Akanda and Zahanara Parvin

(*I hope this achievement will fulfill the dream that you had for me*)

My respective parents in laws – Md Abdul Hakim and Laila Arjuman Banu

(*I hope this achievement will make them proud*)

My beloved wife – Mst Shamima Nasrin

(*I hope this achievement will be our ever-standing glory*)

My lovely daughter – Naziat Alam Zaara

(*I hope this achievement will inspire you on the way of your success)*

ACKNOWLEDGEMENTS

Firstly, I would like to express my deepest appreciation, respect and warmest affection to my advisor Professor Dr. Tarek M. Taha who has been supportive and inspiration to achieve this goal.

I would like to thanks to all the members in my Ph.D. advisory committee, Dr. Vijayan K. Asari, Dr. Raul Ordonez, and Dr. Muhammad Usman for their effort and valuable suggestions. My grateful appreciation to all my colleagues at High Performance Computing (HPC) Lab, faculties, staff, and student in the Department of Electrical and Computer Engineering at the University of Dayton from whom I have learned a lot.

Finally, I would like to express my deepest appreciation to my parents, wife, my sisters, and my daughter for their love and motivation throughout my study.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND NOTATIONS

DCNN	Deep Convolution Neural Network
YOLO	You Only Look Once
SSD	Single Short Multi-Box Detector
NCAB	National Cancer Advisory Board
AI	Artificial Intelligence
ANN	Artificial Neural Networks
DRL	Deep Reinforcement Learning
SNN	Spiking Neural Networks
RNN	Recurrent Neural Networks
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Units
GAN	Generative Adversarial Networks
AE	Auto Encoders
RBM	Restricted Boltzmann Machines
CT	Computer Tomography
CAD	Computer-Aided Diagnosis
SVM	Support Vector Machine
PCA	Principle Component Analysis
KPCA	Kernel PCA
RF	Random Forest
HER	Electronic Health Recoding

ELM	Extreme Learning Approach
SPELM	State Preserving Extreme Learning Machine
CSRN	Cellular Simultaneous Recurrent Network
IRCNN	Inception Recurrent Convolutional Neural Networks
DCRN	Densely Connected Recurrent Convolutional Network
EIN	Equivalent Inception Network
EIRN	Equivalent Inception-Residual Network
RCNN	Recurrent Convolutional Neural Networks
R-Net	Recurrent U-Net
R2U-Net	Recurrent Residual U-Net
FCN	Fully Connected Convolutional Neural Network
CSC	Convolutional Sparse Coding
QUBO	Quadratic Unconstrained Binary Optimization
SGD	Stochastics Gradient Descent
NiN	Network in Network
MLP	Multilayer Perceptron's
SLFNs	Single Hidden Layer Feedforward Neural Networks
RELM	Regularized Extreme Learning Machine
PHOG	Pyramid Histogram of Orientated Gradients (PHOG)
NLP	Natural Language Processing
OCR	Optical Character Recognition
RCL	Recurrent Convolutional Layers
ReLU	Rectified Linear Units
ELU	Exponential Linear Units
WBC	White Blood Cell
RBC	Red Blood Cell

SMMT	Self-Dual Multiscale Morphological Toggle
SIFT	Scale Invariant Feature Extraction
LBP	Local Binary Patterning
NNC	Nearest Neighbor Classifier
MTCNN	multi-task CNN
WSI	Whole Slide Images
DC	Dice Coefficient
H&E	Hematoxylin and Eosin
RRU	Recurrent Residual Unit
MSE	Mean Squared Error
AUC	Area Under Curve
SFM	Sparse Feature Maps
EEDN	Energy Efficient Deep Neuromorphic Network
BC	Binary Connect
TC	Ternary Connect
QC	Quaternary Connect

CHAPTER 1

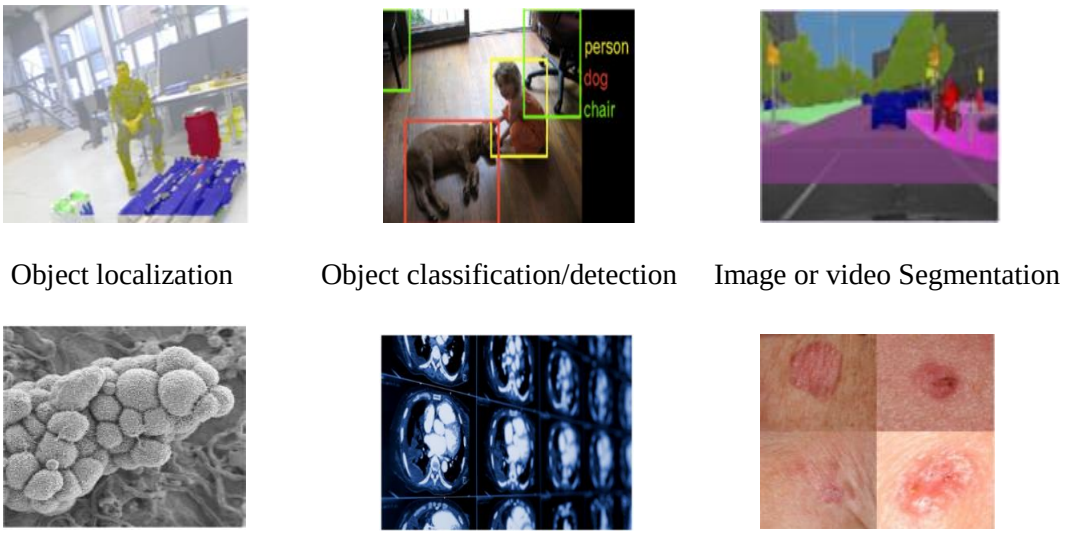
INTRODUCTION

1.1 Motivation

Nowadays, DL provides state-of-the-art performance for image classification, segmentation, detection and tracking, image or video captioning and much more. Since 2012, several DCNN models have been proposed such as AlexNet, VGG, Google Net, Residual Net, DenseNet, and CapsuleNet [1,2]. A DL based approach (CNN in particular) provides superior performance for classification, segmentation, and detection. In the case of classification task, the objective is to compute the class probability where the output is a vector of class confidence which is the same length of target vector. The classification models include AlexNet, Inception-v4, ResNet, DenseNet and so on. Typically, the DL based semantic segmentation approach is used for segmentation tasks, where the model computes the class probability of each pixel of an image. Some of the example models for semantic segmentation includes SegNet, RefineNet, DeepLab, U-Net, R2U-Net etc. For detection problems, the model computes the class probability of the target classes along with the location of the target with a DL based regression approach. For instates: You Only Look Once (YOLO), a single short multi-box detector (SSD), and DL based regression models. These DL based approaches provide superior performance against traditional methods for several reasons: first, recently developed activation functions (such as Rectified Linear Unit (ReLU), Exponential Linear Units (ELU)) resolve training problems for DL approaches. Second, dropout helps to regularize the networks that are very efficient for training a large DL model. Third, several efficient optimization techniques are available for training CNN models efficiently. However, in most cases, models are explored and evaluated for all different tasks on very large-scale datasets such as

ImageNet, and MSCOCO [1] and shows superior performance for image analysis, computer vision, and Bio-medical imaging tasks. In addition, DL based approaches are known as universal learning approaches, where a single model can be utilized efficiently in different modalities for computer vision and medical imaging (such as Pathology images, MRI, CT, and X-ray).

Due to the huge success of DCNNs in the field of computer vision, different variants of DL approach are applied in different modalities of medical imaging including segmentation, classification, detection, registration, and medical information processing. Meanwhile, these deep learning approaches have shown tremendous success in the different modalities of medical imaging [3,4]. For examples: Dermatologist-level performance for skin cancer detection [5], diabetic retinopathy, Neuroimaging for analysis Brain Tumor and Alzheimer diseases, lung cancer detection, breast cancer detection and classification [6], etc. Some example applications are shown in Figure 1.1.



Object localization | Object classification/detection | Image or video Segmentation

Medicine and biology | Brian Cancer Detection | Skin cancer recognition

Figure 1.1: Example images where DL is applied successfully and achieved state-of-the-art performance.

On the other hand, according to Signify research [12], it is expected that for the field of deep learning for medical imaging alone will be invested more than $300 million by 2021, which is more than the amount of entire analysis industry spent in 2016 [6]. For example IBM Watson (rule-based expert system for medical diagnosis and developing the radiology application of Dr. Watson) invested a billion of the dollar in the field of medical imaging and Google Deep Mind Health is another giant in this field. A big initiative from National Cancer Advisory Board (NCAB), Congress passed around $1.8 billion in funding for the Cancer Moonshot over 4 years till 2020 [13]. The goal of this project to accelerate access to next-generation immunotherapy cancer care by endorsing and collaborating between drug companies, biotech companies, insurers, and researchers. The acronym of "QUILT" stands for QUantitative, Integrative, Lifelong Trial is the first initiative which is consisted with the sequences of genomes of 100k cancer patients, assigning 20, 000 patients to next-generation immunotherapy by the year 2020. This trial includes patients with 20 types of cancers, including breast, lung, prostate, and pancreatic. The goal of this initiative is to provide a vaccine-based immunotherapy tailored for the unique tumor signature of a patient. In addition, Chan and Zuckerberg initiative has declared on investment around $2.5 million dollar to early career scientists for conducting research on neurodegenerative disease like Alzheimer's and Parkinson's in the next five years.

However, the main motivation is to contribute for this field so that this technology can be a part of the workflow of hospital for making better decision about patients. Thus, this system would able to help radiologist and doctors for making better decision which ultimately will ensure better treatment of the patients.

1.2 Deep Learning

Since the 1950s, a small subset of Artificial Intelligence (AI), often called Machine Learning (ML), has revolutionized several fields in the last few decades. Neural Networks (NN) are a subfield of

ML, and it was this subfield that spawned Deep Learning (DL). Since its inception DL has been creating ever larger disruptions, showing outstanding success in almost every application domain. Figure 1.2 shows, the taxonomy of AI. DL (using either deep architecture of learning or hierarchical learning approaches) is a class of ML developed largely from 2006 onward. Learning is a procedure consisting of estimating the model parameters so that the learned model (algorithm) can perform a specific task. For example, in Artificial Neural Networks (ANN), the parameters are the weight matrices $(w_{i,j}{}'s)$. DL on the other hand consists of several layers in between the input and output layer which allows for many stages of non-linear information processing units with hierarchical architectures to be present that are exploited for feature learning and pattern classification [1,2]. Learning methods based on representations of data can also be defined as representation learning [1]. Recent literature states that DL based representation learning involves a hierarchy of features or concepts, where the high-level concepts can be defined from the low-level ones and low-level concepts can be defined from high-level ones. In some articles DL has been described as a universal learning approach that is able to solve almost all kinds of problems in different application domains. In other words, DL is not task specific [1,2]. At present deep learning is being applied in almost all areas. As a result, this approach is often called a universal learning approach.

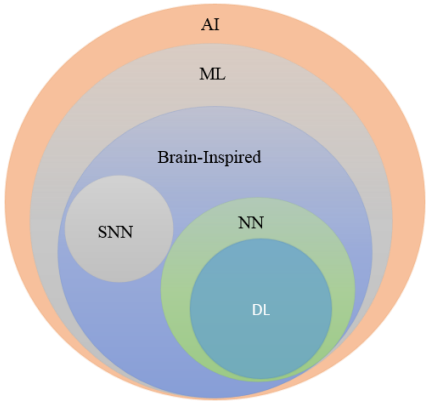


Figure 1.2: AI: Artificial Intelligence, ML, NN, DL, and Spiking Neural Networks (SNN) according to [27].

1.2.1 Types of DL approaches

Like machine learning, deep learning approaches can be categorized as follows: supervised, semi-supervised or partially supervised, and unsupervised which is shown in Figure 1.3. In addition, there is another category of learning called Reinforcement Learning (RL) or Deep RL (DRL) which are often discussed under the scope of semi supervised or sometimes under unsupervised learning approaches.

1.2.1.1 Supervised Learning

Supervised learning is a learning technique that uses labeled data. In the case of supervised DL approaches, the environment has a set of inputs and corresponding outputs$(x_t, y_t) \sim \rho$. For example, if for input x_t, the intelligent agent predicts $\hat{y}_t = f(x_t)$, the agent will receive a loss value $l(y_t, \hat{y}_t)$. The agent will then iteratively modify the network parameters for better approxima ation of the desired outputs. After successful training, the agent will be able to get the correct answers to questions from the environment. There are different supervised learning approaches for deep leaning including Deep Neural Networks (DNN), Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN) including Long Short-Term Memory (LSTM), and Gated Recurrent Units (GRU) [1].

1.2.1.2 Semi-supervised Learning

Semi-supervised learning is learning that occurs based on partially labeled datasets (often also called reinforcement learning). In some cases, DRL and Generative Adversarial Networks (GAN) are used as semi-supervised learning techniques. Additionally, RNN including LSTM and GRU are used for semi-supervised learning as well.

1.2.1.3 Unsupervised learning

Unsupervised learning systems are ones that can without the presence of data labels. In this case, the agent learns the internal representation or important features to discover unknown relationships or structure within the input data. Often clustering, dimensionality reduction, and generative techniques are considered as unsupervised learning approaches. There are several members of the deep learning family that are good at clustering and non-linear dimensionality reduction, including Auto-Encoders (AE), Restricted Boltzmann Machines (RBM), and the recently developed GAN. In addition, RNNs, such as LSTM and RL, is also used for unsupervised learning in many application domains [1].

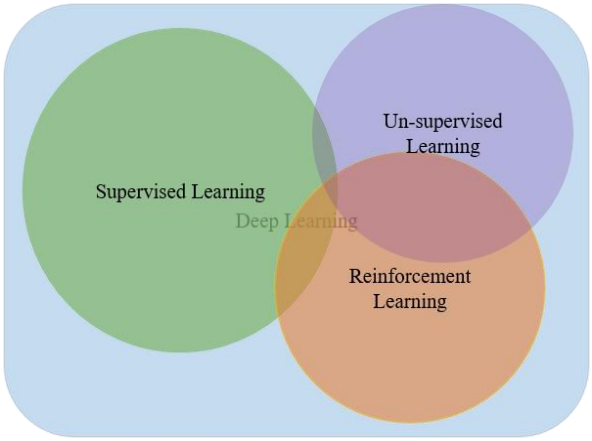


Figure 1.3: Category of Deep Learning approaches.

1.2.1.4 Deep Reinforcement Learning (DRL)

Deep Reinforcement Learning is a learning technique for use in unknown environments [14]. DRL began in 2013 with Google Deep Mind [15, 16]. From then on, several advanced methods have been proposed based on RL. Here is an example of RL: if environment samples inputs: $x_t \sim \rho$, agent predict: $\hat{y}_t = f(x_t)$, agent receive cost: $c_t \sim P(c_t|x_t, \hat{y}_t)$ where P is an unknown probability distribution, the environment asks an agent a question, and gives a noisy score as the answer. Sometimes this approach is called semi-supervised learning as well. There are many semi-supervised and un-supervised techniques that have been implemented based on this concept. In RL,

we do not have a straight forward loss function, thus making learning harder compared to traditional supervised approaches. The fundamental differences between RL and supervised learning are: first, you do not have full access to the function you are trying to optimize; you must query them through interaction, and second, you are interacting with a state-based environment: input x_t depends on previous actions. Depending upon the problem scope or space, you can decide which type of RL needs to be applied for solving a task. If the problem has a lot of parameters to be optimized, DRL is the best way to go. If the problem has fewer parameters for optimization, a derivation free RL approach is good. An example of this is annealing, cross entropy methods, and SPSA [14].

1.3 Why Deep Learning

1.3.1 Universal learning approach

This approach is sometimes called universal learning because it can be applied to almost in any application domain.

1.3.2 Robust

Deep learning approaches do not require the design of features ahead of time. Features are automatically learned that is optimal for the task at hand. As a result, the robustness to natural variations in the data is automatically learned.

1.3.3 Generalization

The same deep learning approach can be used in different applications or with different data types. This approach is often called transfer learning. In addition, this approach is helpful where the problem does not have sufficient available data. There are several papers have been published based on this concept.

1.3.4 Scalability

The deep learning approach is highly scalable. In a 2015 paper, Microsoft described a network known as ResNet [11,20]. This network contains 1202 layers and is often implemented on a supercomputing scale. In this paper, the authors explain details about how DL can deal with different criteria including volume, velocity, variety and veracity of the big data problem and have shown different advantages of DL approaches of dealing with big data problems [29,30]. Deep learning is a data-driven technique. Figure 1.4 clearly demonstrates that the performance of traditional ML approaches shows better performance for lesser amounts of input data.

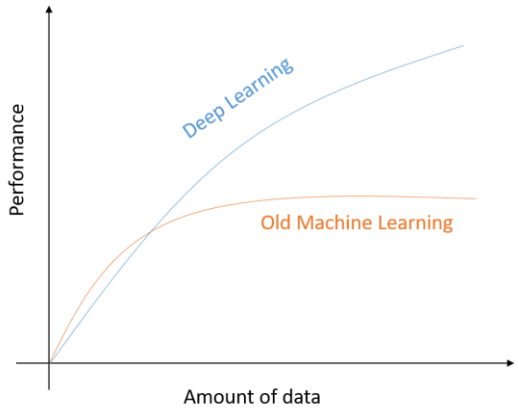


Figure 1.4: The performance of deep learning with respect to the number of data

1.4 Deep Learning in Computer Vision

There are some outstanding successes of DCNN models in the fields of computer vision as discussed below: One of the large-scale problems is named Large Scale Visual Recognition Challenge (LSVRC). DCNN based techniques show state-of-the-art accuracy on the ImageNet task [31]. Russakovsky et al. recently published a paper on the ImageNet dataset and the state-of-the-art accuracies achieved during the last few years [30]. The following graph shows the success story of deep learning techniques overtime on this challenge from 2012. ResNet-152 shows only 3.57% error, which is better than the human error for this task at 5% which is shown in Figure 1.5.

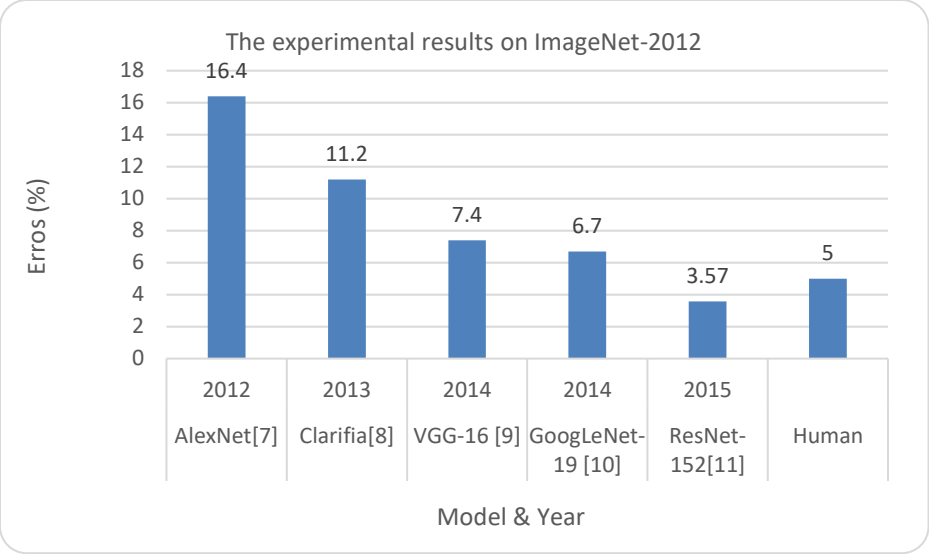


Figure 1.5: Accuracy of different DL models on ImageNet challenge dataset.

1.5 Deep Learning in Medical Imaging

There are many techniques of computer-aided detection system was developed and introduced in the clinical workflow in early 2010. One of the main tasks of medical imaging in radiology practice where the structural abnormalities are identified is classifying them into disease categories. The medical imaging comes from different imaging techniques such as Microscopic, Computer Tomography (CT), ultrasound, X-ray, and Magnetic Resonance Imaging (MRI). The goal of Computer-Aided Diagnosis (CAD) is to obtain a faster and better diagnosis to ensure better treatment of a large number of people at the same time. Additionally, efficient automatic processing without human involvement can help to reduce human errors, overall processing time and cost. Due to the slow process and tedious nature of manual segmentation approaches, there is a significant demand for computer algorithms that can be used for segmentation quickly and accurately without human interaction.

However, there are some limitations of medical image processing including data scarcity and class imbalance. Most of the time a large number of labels (often in the thousands) for training is not available for several reasons [2]. Labeling the sample requires domain expert for that field which

is expensive, and it requires a lot of effort and time. Sometimes, different data transformation or augmentation techniques (data whitening, rotation, translation, and scaling) are applied for increasing the number of labeled samples available [3, 4]. In addition, patch-based approaches are used for solving class imbalance problems. In this work, we have evaluated the proposed approaches on both patch-based and entire image-based approaches. However, to switch from the patch-based approach to the pixel-based approach that works with the entire image, we must be aware of the class imbalance problem. This patch-based approach has a chance of losing the global context of the entire image. Both patch and end-to-end image-based techniques are used for medical image classification and semantic segmentation tasks. In the case of semantic segmentation, the image backgrounds are assigned a label and the foreground regions are assigned a target class. Therefore, the class imbalance problem is resolved without any trouble.

Furthermore, in medical image processing, global localization and context modulation is very often applied for localization tasks. Each pixel is assigned a class label with a desired boundary that is related to the contour of the target lesion in identification tasks. To define these target lesion boundaries, we must emphasize the related pixels. Landmark detection in medical imaging [2] is one example of this. There were several traditional machine learning and image processing techniques available for medical image classification, segmentation, and detection tasks before the DL revolution, including amplitude Support Vector Machine (SVM), Principle Component Analysis (PCA), Random Forest (RF), segmentation based on histogram features, the region-based segmentation method, the graph-cut, and many more approaches. However, in this research, we have applied our proposed improved deep learning-based approaches for medical image classification, segmentation, and detection tasks.

A recent study shows the successes of DL in different modalities of medical imaging [3]. However, DL is not limited to medical imagining. The different DL based approaches have been used in different field including data acquisition and registration, classification, segmentation, automatic

labeling and captioning, computer-aided detection and diagnosis, reading assistants and automatic dictation, advanced Electronic Health Recording (HER) and precision imaging for personalized medicine. In this research, we have focused on bio-medical image classification, segmentation, and detection tasks. The following sections discuss each individual topic in detail, where DL approaches have been applied successfully and shown superior performance for medical imaging for classification, segmentation, and detection tasks.

1.5.1 Classification

The DL for computer vision utilizes a huge number of samples (for example the ImageNet dataset consists of 14 million samples and more than 20,000 classes) [31] and in most of the cases, transfer learning is used for training a model. People in the DL community uses the transfer learning technique for several reasons. Two main reasons are: first, use pre-trained weights for feature extraction, and second, use pre-trained weights to an existing network and fine-tune with a new dataset. For example several studies have been conducted where the trained weights are used form ImageNet dataset. Out of many of the studies, some of the researches have reported the best results for medical image classification with transfer learning [3,4]. Another outstanding work for skin cancer classification where they have achieved dermatologist-level performance for skin cancer recognition. In this work, the Google Inception-v3 model used with transfer learning [23].

DCNN approaches are massively applied for neuroimaging of Brain Tumor and Alzheimer disease segmentation and classification from MRI [34, 35]. 2D and 3D convolutional kernels were applied in different models for this purpose. DCNN based approaches achieved state-of-the-art performance compared against existing approaches [34, 35]. In addition, like Neuroimaging, there are different deep CNN techniques are applied for lung feature detection and classification and achieved superior performance compared to traditional machine learning approaches [36, 37]. Furthermore, deep learning techniques are applied for X-ray image analysis for chest radiographs [38]. Besides, DL approaches are also applied for medical histology and microscopic imaging for

pathological classification tasks. Recently, the MDNet was proposed and experimented on pathology bladder cancer images and its diagnostic reports (BCIDR) dataset and outperforms compared to baselines methods [39]. Locality sensitive DL is used for detection and classification of nuclei in routine color cancer histology images [40]. Breast cancer detection and classification from histology images is shown [6]. WBC classification approach with CNN first time applied by Mehdi in 2013. This approach achieved 85% classification accuracy for five classes which are better compared to these traditional machine learning approaches of SVM and Kernel PCA (KPCA) [41]. Furthermore, the variant models are applied to different modalities of medical imaging and achieved state-of-the-art performance in most of the cases [3].

1.5.2 Segmentation

There are several DL models that have proposed specifically for the medical image segmentation tasks considering the drawback of data insufficiency and class imbalance problems. One of the very first and most popular approaches for semantic medical image segmentation called "U-Net" 32]. However, meanwhile, different variants of U-Net models have been proposed, including a very simple variant of U-Net for CNN-based segmentation of Medical Imaging data [32]. In this model, two modifications are made to the original design of U-Net: first, a combination of multiple segmentation maps and forward feature maps are summed (element-wise) from one part of the network to the other. The feature maps are taken from different layers of encoding and decoding units and finally, summation (element-wise) is performed outside of the encoding and decoding units. The authors report improved performance during training with better convergence compared to U-Net, but no benefit was observed when using a summation of features during the testing phase [32]. However, this concept proved that feature summation impacts the performance of a network. The importance of skipped connections for biomedical image segmentation tasks have been empirically evaluated with U-Net and residual networks [42]. A deep contour-aware network called DCAN was proposed in 2016, which can extract multi-level contextual features using a hierarchical

architecture for accurate gland segmentation of histology images and shows very good performance for segmentation [43]. Furthermore, Nabla-Net: a deep convolutional architecture was proposed for segmentation in 2017 [44].

Other deep learning approaches have been proposed based on U-Net for 3D medical image segmentation tasks as well. The 3D-Unet architecture for volumetric segmentation learns from sparsely annotated volumetric images [45]. A powerful end-to-end 3D medical image segmentation system based on volumetric images called V-net has been proposed, which consists of an FCN with residual connections [46]. This paper also introduces a dice loss layer [46]. Furthermore, a 3D deeply supervised approach for automated segmentation of volumetric medical images was presented in [45]. High-Res3DNet was proposed using residual networks for 3D segmentation tasks in 2016 [47]. In 2017, a CNN based brain tumor segmentation approach was proposed using a 3D-CNN model with a fully connected Conditional Random Field (CRF) [48]. Pancreas segmentation was proposed in [50], and Voxresnet was proposed in 2016 where a deep voxewise residual network is used for brain segmentation. This architecture utilizes residual networks and summation of feature maps from different layers [50].

1.5.3 Computer-Aided Detection (CAD)

Medical imaging and data acquisition system have been developed a lot in the last few years. The current CAD system is developed to achieve two goals: detection and false positive reduction. The first object can be achieved based on the detection algorithm, DL approaches are showing tremendous performance on detection tasks (for example skin lesion detection [5]). On the other hand, traditional ML approaches such as SVM, PCA are applied to this purpose. Unfortunate, the traditional ML-based CAD does not perform well in clinical practice. However, recently DL based techniques are showing superior performance for false positive reduction tasks in different

modalities of medical imaging [3,4]. Recently, there are several CAD methods have proposed for breast cancer detection [6], lung cancer detection [3], and Alzheimer ’s disease (AD) [34,35].

1.6 Scientific Contributions

The novel scientific contributions of this dissertation proposal are summarized as follows:

- We have conducted a comprehensive survey on DL where we have discussed different learning approaches, different models, and recently developed advanced training techniques.
- We have proposed improved Extreme Learning Approach (ELM) which is named **State Preserving Extreme Learning Machine (SPELM)** and evaluated on different publicly available datasets and achieved better performance against ELM approach.
- The performance of DCNN approaches are evaluated for different types of convolutional kernels including random, Gabor filters, and filters which are created with the cellular simultaneous recurrent network (CSRN). This study shows that network initialization with Gabor filters shows better performance compared to other two approaches.
- The empirical evaluation of different DCNN models for Handwritten Bangla Character Recognition (HBCR) and achieved state-of-the-art testing performance for handwritten Bangla numeral, alphabets, and special characters recognition.
- We have proposed improved models: **Inception Recurrent Convolutional Neural Networks (IRCNN),** and **Densely Connected Recurrent Convolutional Network (DRCN)** and evaluated for classification tasks. In this study, we have discovered the impact of recurrent convolutional layers in popular DCNN architectures including inception network [22,23] and Dense-Net [27]. We have investigated IRCNN performance and the results show higher recognition accuracy when compared to most of the popular DCNN models including Equivalent Inception Networks (EIN), Equivalent Inception-Residual Networks (EIRN), and Recurrent Convolutional Neural Network (RCNN).

- The **Improved Recurrent Residual Convolutional Neural Networks (IRRCNN)** model which utilizes the power of the Recurrent Convolutional Neural Network (RCNN), the Inception network, and the Residual network. This approach improves the recognition accuracy of the Inception-residual network with the same number of network parameters. In addition, this proposed architecture generalizes the Inception Network (IN), the RCNN, and the Residual network with significantly improved training accuracy. The experiment has been conducted on several datasets where the IRRCNN provides better testing accuracy compared to the Inception Recurrent CNN (IRCNN), the Equivalent IN (EIN), and the equivalent Inception Residual Network (EIRN).
- We have proposed a Recurrent U-Net as well as a Recurrent Residual U-Net model, which are named **RU-Net** and **R2U-Net** models respectively. The proposed models utilize the power of U-Net, Residual Networks, and Recurrent Convolutional Neural Networks (RCNNs). The proposed models are tested on three different medical imaging datasets such as blood vessel segmentation in retina images, skin cancer segmentation, and lung lesion segmentation. The experimental results show superior performance on segmentation tasks compared to equivalent models including a fully connected convolutional neural network (FCN) called SegNet, U-Net, and the Residual U-Net (ResU-Net).
- We have developed a regression model with R2U-Net model which is named **UD-Net** for end-to-end detection task. The model is tested for nuclei and lymphocyte detection and achieved better performance with respect to the existing DL approaches.
- To generalize classification models, we have applied these modes in different modalities of medical imaging applications including blood cell classification, breast cancer classification, nuclei classification, Lymphoma classification, Invasive ductal carcinoma detection, and mitosis detection.

- The R2U-Net segmentation model is successfully applied in digital pathology including Nuclei segmentation, Epithelium segmentation, and Tubule segmentation and achieved state-of-the-art performance.
- We have also implemented and evaluated Convolutional Sparse Coding (CSC) on IBM's TrueNorth system for the very first time and evaluated for image reconstruction. The results demonstrate promising reconstruction with a very low power requirement.
- We have evaluated different energy efficient deep CNN models on TrueNorth Neuromorphic computing system. The experimental results show very promising classification accuracies against CPU implementation with very low power consumption on IBM's NS1e Neurosynaptic system. The results demonstrate that for datasets with large numbers of classes, wider networks perform better when compared to deep networks comprised of nearly the same core complexity on IBM's TrueNorth system.
- We propose an effective quantization approach for Recurrent Neural Networks (RNN) techniques including Long Short-Term Memory (LSTM), Gated Recurrent Units (GRU), and Convolutional Long Short-Term Memory (ConvLSTM). The experimental results are compared against the full precision versions of the LSTM, GRU, and ConvLSTM. They show promising results for both sentiment analysis and video frame prediction.
- For the first time, Quadratic Unconstrained Binary Optimization (QUBO) problem for the solution of graph problems on the IBM's Neurosynaptic TrueNorth System. We have experimented on different types of graph problems with different levels of complexities and achieved encouraging results on IBM's Neuromorphic TrueNorth chip. Along with the QUBO on quantum annealing, it is the important first step towards the solutions of QUBO on Neuromorphic computing systems.

1.7 Dissertation Outline

Chapter 1: This is a chapter on the overall introduction of this thesis which includes motivation, the background of this study, what is DL, reasons of using DL approaches, DL for computer vision and Bio-medical imaging.

Chapter 2: this chapter explains details of background study on neural networks (NN) and different training approaches: gradient descent, stochastics gradient descent (SGD), Back-propagation, Learning rates, weight decay. This chapter also discusses Extreme Learning Machine (ELM) and Convolutional Neural Networks (CNN). Popular CNN model including modern architecture such as All convolution (AllConv), Network in Network (NiN), Residual Network, DenseNet, FractalNet, and CapsuleNet are explained. Furthermore, we have discussed modern training techniques including dataset preparation, initialization approach, activation functions, pooling approaches, regularization techniques, and optimization method for DL.

Chapter 3: the improved Extreme Learning Machine (ELM) which is named state preserving Extreme Learning Machine (SPELM) is discussed in this chapter. This chapter also discusses the experimental results of SPELM on different publicly available datasets and comparison against ELM and Regularized ELM (RELM) approaches.

Chapter 4: This chapter discusses on the experimental evaluation of state-of-the-art DCCN models (All convolution (AllConv), Network in Network (NiN), Residual Network, DenseNet, FractalNet, and CapsuleNet) for Bangla Hand-Written Character Recognition including numeral, alphabets and special characters and achieved the state-of-the-art performance against existing machine learning and DL based methods.

Chapter 5: explains on IRCNN and DCRN models. This chapter also discusses on the experimental details of IRCNN and DCRN models on different datasets and discusses on the impact of recurrent convolutional layers on modern deep learning architectures.

Chapter 6: this chapter discusses on a new model which is called IRRCNN model. In addition, the experimental results on different datasets and explains on the trade-off between the split ratio during training versus the training and testing accuracy.

Chapter 7: the new R2U-Net model is explained in this chapter which is applied for Bio-medical image segmentation tasks. This chapter also includes the results on three different datasets including: retina blood vessel segmentation, skin cancer segmentation, and lung segmentation. Furthermore, a details comparison between R2U-Net, SegNet, U-Net, and ResU-Net are given in the end of this chapter.

Chapter 8: discusses how to apply IRRCNN model for Blood cell (both white and red blood cells) classification task. In addition, this chapter explains details on the experimental results and comparison against existing methods for blood cell classification.

Chapter 9: to generalize, the IRRCNN model is used and evaluated for Breast Cancer classification tasks which is tested on two different publicly available datasets and achieved state-of-the-art performance patient as well as image level analysis. This chapter discusses on methods in detail which experimental results.

Chapter 10: this chapter explains three different models including Densely Connected Recurrent Network (DCRN), and R2U-Net, and R2U-Net based regression which is named UD-Net. These models are evaluated for nuclei classification, segmentation, and detection tasks which are also explained in this chapter.

Chapter 11: this chapter explains on seven different use cases study for computational pathology where we have applied different models including IRRCNN, DRCN, R2U-Net, and UD-Net for classification, segmentation and detection tasks for digital pathology.

Chapter 12: the convolutional sparse coding (CSC) which is implemented on IBM's TrueNoeth neuromorphic computing system for the very first time, this chapter explains details how to map CSC technique on TrueNorth with experimental results.

Chapter 13: this chapter discusses on the energy efficient DCNN approaches which are implemented with Eedn deep learning framework for IBM's TrueNorth system. In addition, the experimental results on different publicly available datasets are also explained and compared against the CPU version of DCNN models in term of accuracy and power.

Chapter 14: different efficient quantization approaches (Binary connect, Ternary connecter and quaternary connect) for Recurrent Neural Networks (RNN) are explained in this chapter which is tested for sentiment analysis and video frame prediction with Convolutional Long Short-Term Memory (Conv LSTM).

Chapter 15: for the very first time, Quadratic Unconstrained Binary Optimization (QUBO) problem is implemented on IBM's TrueNorth Neuromorphic system. QUBO is applied for solving different graph problems with different constrains and difficulties on the IBM's Neurosynaptic is discussed in this chapter.

Chapter 16: Finally, the overall conclusions of this thesis is made in this chapter.

CHAPTER 2

BACKGROUND STUDY

2.1 The History of DNN

Below is a brief history of neural networks highlighting key events:

- 1943: McCulloch & Pitts show that neurons can be combined to construct a Turing machine (using ANDs, ORs, & NOTs) [53].
- 1958: Rosenblatt shows that perceptrons will converge if what they are trying to learn can be represented [54].
- 1969: Minsky & Papert show the limitations of perceptron's, killing research in neural networks for a decade [55].
- 1985: The backpropagation algorithm by Geoffrey Hinton et al [56] revitalizes the field.
- 1988: Neocognitron: a hierarchical neural network capable of visual pattern recognition [17].
- 1998: CNN with Backpropagation for document analysis by Yan LeCun [18].
- 2006: The Hinton lab solves the training problem for DNNs [57].
- 2012: AlexNet by Alex Krizhevesky in 2012 [7].

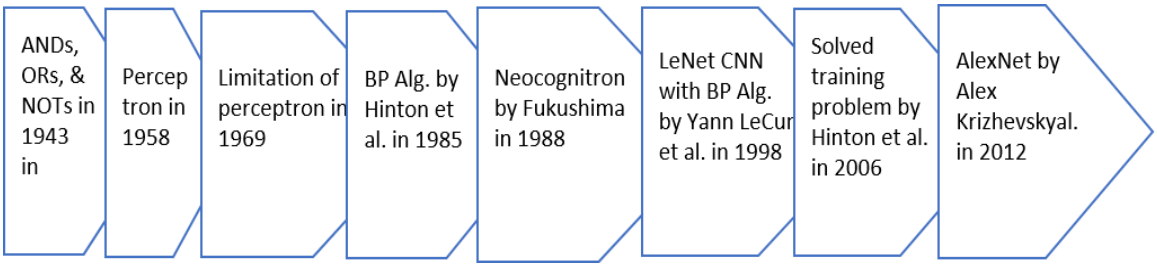


Figure 2.1: History of Neural Network to DL approaches.

Computational neurobiology has conducted significant research on constructing computational models of artificial neurons. Artificial neurons, which try to mimic the behavior of the human brain, are the fundamental component for building ANNs. The basic computational element (neuron) is called a node (or unit) which receives inputs from external sources and has some internal parameters (including weights and biases that are learned during training) which produce outputs. This unit is called a perception. The basic block diagram of a perceptron for NNs is shown in the following diagram.

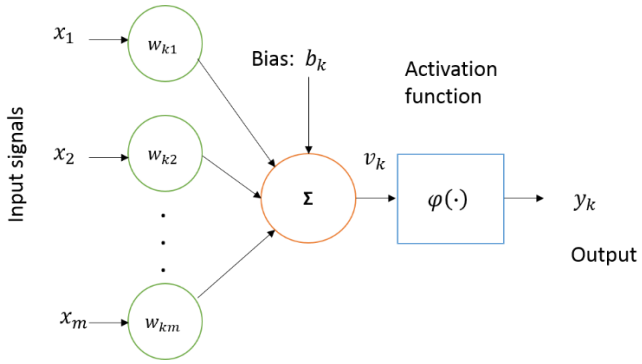


Figure 2.2: A basic model of a neuron.

Figure 2.1 shows the basic nonlinear model of a neuron, where $x_1, x_2, x_3, \cdots x_m$ are input signals; $w_{k1}, w_{k2}, w_{k3}, \cdots w_{km}$ are synaptic weights; v_k is the linear combination of input signals; $\varphi(\cdot)$ is the activation function (such as sigmoid), and y_k is the output. The bias b_k is added with a linear combiner of outputs v_k, which has the effect of applying an affine transformation, producing the outputs y_k. The neuron functionality can be represented mathematically as follows:

$$v_k = \sum_{j=1}^{m} w_{kj}\, x_j \tag{2.1}$$

$$y_k = \varphi(v_k + b_k) \tag{2.2}$$

ANNs or general NNs consist of Multilayer Perceptron's (MLP) which contain one or more hidden layers with multiple hidden units (neurons) in them. The NN model with MLP is shown in Figure 2.3.

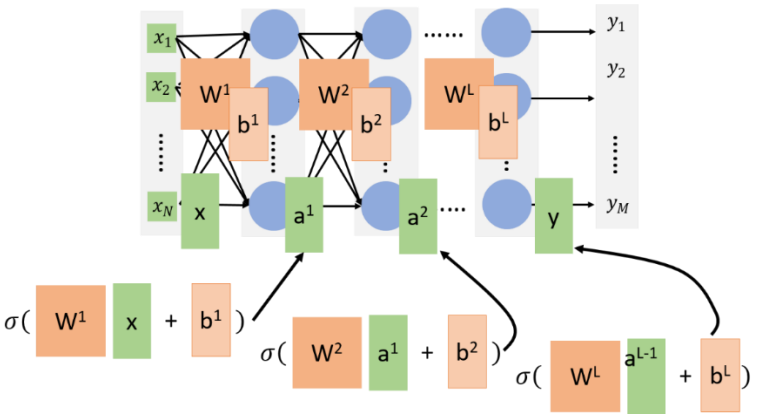


Figure 2.3: Neural network model with multiple layers of perceptron.

The multilayer perceptron can be expressed mathematically (which is a composite function) as follows:

$$y = f(x) = \varphi(w^L \cdots \varphi(w^2\varphi(w^1 x + b^1) + b^2) \cdots + b^L) \tag{2.3}$$

Algorithm 2.1: Gradient descent

Inputs: loss function ε, learning rate η, dataset X, y and the model $\mathcal{F}(\theta, x)$

Outputs: Optimum θ which minimizes ε

REPEAT until converging:

$$\tilde{y} = \mathcal{F}(\theta, x)$$

$$\theta = \theta - \eta \cdot \frac{1}{N}\sum_{i=1}^{N} \frac{\partial \varepsilon(y, \tilde{y})}{\partial \theta}$$

End

2.1.1 Gradient descent

The gradient descent approach is a first-order optimization algorithm which is used for finding the local minima of an objective function. This has been used for training ANNs in the last couple of decades successfully. Algorithm 2.1 explains the concept of gradient descent:

2.1.2 Stochastic Gradient Descent (SGD)

Since a long training time is the main drawback for the traditional gradient descent approach, the SGD approach is used for training Deep Neural Networks (DNN) [2]. Algorithm 2.2 explains SGD in detail.

Algorithm 2.2: Stochastic Gradient Descent (SGD)

Inputs: loss function ε, learning rate η, dataset $X, y,$ and the model $\mathcal{F}(\theta, x)$

Outputs: Optimum θ which minimizes ε

REPEAT until converging:

Shuffle X, y;

For each batch of x_i, y_i in X, y **do**

$\tilde{y}_i = \mathcal{F}(\theta, x_i)$;

$\theta = \theta - \eta \cdot \frac{1}{N} \sum_{i=1}^{N} \frac{\partial \varepsilon(y_i, \tilde{y}_i)}{\partial \theta}$

End

2.1.3 Back-propagation

DNN is trained with the popular Back-Propagation (BP) algorithm with SGD [1]. The pseudo code of the basic Back-propagation is given in Algorithm 2.3. In the case of MLPs, we can easily represent NN models using computation graphs which are directive acyclic graphs. For that representation of DL, we can use the chain-rule to efficiently calculate the gradient from the top to the bottom layers with BP as shown in Algorithm III for a single path network. For example:

$$y = f(x) = \varphi(w^L \cdots \varphi(w^2 \varphi(w^1 x + b^1) + b^2) \cdots + b^L) \tag{2.4}$$

This is a composite function for L layers of a network. In the case of $L = 2$, then the function can be written as

$$y = f(x) = f\big(g(x)\big) \tag{2.5}$$

According to the chain rule, the derivative of this function can be written as

$$\frac{\partial y}{\partial x} = \frac{\partial f(x)}{\partial x} = f'\big(g(x)\big) \,.\, g'(x) \tag{2.6}$$

2.1.4 Momentum

Momentum is a method which helps to accelerate the training process with the SGD approach. The main idea behind it is to use the moving average of the gradient instead of using only the current real value of the gradient. We can express this with the following equation mathematically:

$$v_t = \gamma\, v_{t-1} - \eta\, \nabla\, \mathcal{F}(\theta_{t-1}) \tag{2.7}$$

$$\theta_t = \theta_{t-1} + \, v_t \tag{2.8}$$

Algorithm 2.3: Back-propagation

Input: A network with l layers, the activation function σ_l , the outputs of the hidden layer $h_l = \sigma_l\big(W_l^T h_{l-1} + b_l\big)$ and the network output $\tilde{y} = h_l$

Compute the gradient: $\delta \leftarrow \frac{\partial \varepsilon(y_i, \tilde{y}_i)}{\partial y}$

For $i \leftarrow l$ to 0 **do**

Calculate gradient for the present layer:

$$\frac{\partial \varepsilon(y,\tilde{y})}{\partial W_l} = \frac{\partial \varepsilon(y,\tilde{y})}{\partial h_l}\frac{\partial h_l}{\partial W_l} = \delta \frac{\partial h_l}{\partial W_l}$$

$$\frac{\partial \varepsilon(y,\tilde{y})}{\partial b_l} = \frac{\partial \varepsilon(y,\tilde{y})}{\partial h_l}\frac{\partial h_l}{\partial b_l} = \delta \frac{\partial h_l}{\partial b_l}$$

Apply gradient descent using $\frac{\partial \varepsilon(y,\tilde{y})}{\partial W_l}$ and $\frac{\partial \varepsilon(y,\tilde{y})}{\partial b_l}$

The back-propagate gradient to the lower layer

$$\delta \leftarrow \frac{\partial \varepsilon(y,\tilde{y})}{\partial h_l}\frac{\partial h_l}{\partial h_{l-1}} = \delta \frac{\partial h_l}{\partial h_{l-1}}$$

End

Here γ is the momentum and η is the learning rate for the t^{th} round of training. Other popular approaches have been introduced during the last few years which are explained in the scope of optimization approaches. The main advantage of using momentum during training is to prevent the network from getting stuck in a local minimum. The values of momentum are $\gamma \in (0,1]$. It is noted that a higher momentum value overshoots its minimum, possibly making the network unstable. In general, γ is set to 0.5 until the initial learning stabilizes and is then increased to 0.9 or higher [1].

2.1.5 Learning rate (η)

The learning rate is an important component for training DNN (as explained in Algorithm 2.1 and 2.2). The learning rate is the step size considered during training which makes the training process faster. However, selecting the value of the learning rate is sensitive. For example: if you choose a larger value for η, the network may start diverging instead of converging. On the other hand, if you choose a smaller value for η, it will take more time for the network to converge. In addition, it may easily get stuck in local minima. The typical solution for this problem is to reduce the learning rate during training [1]. There are three common approaches used for reducing the learning rate during training: constant, factored, and exponential decay. First, we can define a constant ζ which is applied to reduce the learning rate manually with a defined step function. Second, the learning rate can be adjusted during training with the following equation:

$$\eta_t = \eta_0 \, \beta^{t/\epsilon} \tag{2.9}$$

Where η_t is the tth round learning rate, η_0 is the initial learning rate, and β is the decay factor with a value between the range of $(0,1)$. The step function format for exponential decay is:

$$\eta_t = \eta_0 \, \beta^{\lfloor t/\epsilon \rfloor} \tag{2.10}$$

The common practice is to use a learning rate decay of $\beta = 0.1$ to reduce the learning rate by a factor of 10 at each stage.

2.1.6 Weight decay

Weight decay is used for training deep learning models as an L2 regularization approach, which helps to prevent overfitting the network and model generalization. L2 regularization for $\mathcal{F}(\theta, x)$ can be define as:

$$\Omega = \|\theta\|^2 \tag{2.11}$$

$$\hat{\varepsilon}(\mathcal{F}(\theta, x), y) = \varepsilon(\mathcal{F}(\theta, x), y) + \frac{1}{2}\lambda\,\Omega \tag{2.12}$$

The gradient for the weight θ is:

$$\frac{\partial \frac{1}{2}\lambda\Omega}{\partial\theta} = \lambda \cdot \theta \tag{2.13}$$

General practice is to use the value $\lambda = 0.0004$. A smaller λ will accelerate training.

Other necessary components for efficient training including data preprocessing and augmentation, network initialization approaches, batch normalization, activation functions, regularization with dropout, and different optimization approaches.

In the last few decades, many efficient approaches have been proposed for better training of deep neural networks. Before 2006, attempts taken at training deep architectures failed: training a deep supervised feed-forward neural network tended to yield worse results (both in training and in test error) than shallow ones (with 1 or 2 hidden layers). Hinton's revolutionary work on DBNs spearheaded a change in this in 2006 [58]. Due to their composition, many layers of DNNs are more capable of representing highly varying nonlinear functions compared to shallow learning approaches [1,2, 58]. Moreover, DNNs are more efficient for learning because of the combination of feature extraction and classification layers. The following sections discuss in detail about different DL approaches with necessary components.

2.2 Extreme Learning Machine (ELM)

ELM typically applies random computational nodes in the hidden layer and increases learning speed by means of randomly generated weights and biases for hidden nodes rather than iteratively adjusting network parameters, which is commonly adopted by gradient-based methods. Different from traditional learning algorithms, ELM tends to reach not only the smallest training error but also the smallest norm of output weights [59,60].

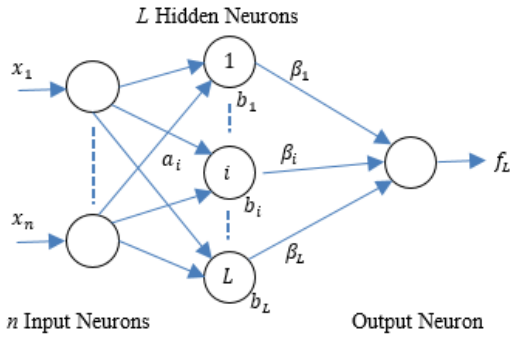


Figure 2.4: A typical architecture of the ELM.

A typical architecture of ELM is shown in Figure 2.3. The output function of ELM with L hidden nodes for generalized SLFNs is expressed as in [59]

$$f_L(x) = \sum_{i=1}^{L} \beta_i \, g_i(x) = \sum_{i=1}^{L} \beta_i G(a_i, b_i, x), x \in R^d, \beta_i \in R^m \qquad (2.14)$$

where $a_i = [a_{i1}, a_{i2}, \dots\dots, a_{in}]^T$ is the weight vector connecting the input nodes to the ith hidden node, b_i is the ith bias of the hidden node , g_i denotes the output function, i.e., activation function $G(a_i, b_i, x)$ of the ith hidden node, and $\beta_i = [\beta_{i1}, \beta_{i2}, \dots\dots, \beta_{im}]^T$ is the weight vector linking the ith hidden node to the output nodes. For N arbitrary distinct samples $(x_j, t_j) \in R^d \times R^m$ the SLFNs with L hidden nodes can approximate these N samples with zero error, meaning $\sum_{j=1}^{L} \|f_j - t_j\| = 0$. Hence, there exists (a_i, b_i) and β_i such that

$$\sum_{i=1}^{L} \beta_i G(a_i, b_i, x_j) = t_j. \; j = 1,2, \dots, N \qquad (2.15)$$

The above equations can be rewritten compactly as

$$H\beta = T \tag{2.16}$$

where,

$$H = \begin{bmatrix} h(x_1) \\ \vdots \\ h(x_n) \end{bmatrix} = \begin{bmatrix} G(a_1, b_1, x_1) & \dots & G(a_L, b_L, x_1,) \\ \vdots & \ddots & \vdots \\ G(a_1, b_1, x_N) & \dots & G(a_L, b_L, x_N) \end{bmatrix}_{N \times L} \tag{2.17}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L \times m}, \qquad T = \begin{bmatrix} t_1^T \\ \vdots \\ t_L^T \end{bmatrix}_{N \times m}$$

H is the hidden layer output matrix of the SLFN, and the ith column of H is the ith hidden node output with respect to inputs $x_1, x_2, \cdots\cdots, x_N$, while the jth row, i.e., $h(x_j)$, is the hidden layer feature mapping corresponding to the jth input x_j. As the hidden node parameters (a_i, b_i) can be randomly generated and remain unchanged, the only unknown parameters in ELM are the output weight vectors β_i between the hidden layer and the output layer, which can be simply resolved by ordinary least-square error analysis. Since ELM aims to minimize the training error $\|H\beta - T\|$ and the norm of weights$\|\beta\|$, the smallest norm least-squares solution of the above linear system is

$$\hat{\beta} = H^{\dagger}T, \tag{2.18}$$

where $H^{\dagger}$ is the Moore-Penrose generalized inverse of matrix H [1]. Hence, the prediction value matrix Y is expressed by

$$Y = H\hat{\beta} = HH^{\dagger}T. \tag{2.19}$$

The error matrix can be described as

$$e = \|Y - T\|^2 = \left\|HH^{\dagger}T - T\right\|^2. \tag{2.20}$$

To increase the stability and generalization ability of the traditional EML, Huang et al. introduced the equality constrained optimization-based ELM [62].

2.3 Convolutional Neural Networks

This network structure was first proposed by Fukushima in 1988 [17]. It was not widely used however due to limits of computation hardware for training the network. In the 1990s, LeCun et al. applied a gradient-based learning algorithm to CNNs and obtained successful results for the handwritten digit classification problem [18]. After that, researchers further improved CNNs and reported state-of-the-art results in many recognition tasks. CNNs have several advantages over DNNs, including being more similar to the human visual processing system, being highly optimized in the structure for processing 2D and 3D images, and being effective at learning and extracting abstractions of 2D features. The max pooling layer of CNNs is effective in absorbing shape variations. Moreover, composed of sparse connections with tied weights, CNNs have significantly fewer parameters than a fully connected network of similar size. Most of all, CNNs are trained with the gradient-based learning algorithm and suffer less from the diminishing gradient problem. Given that the gradient-based algorithm trains the whole network to minimize an error criterion directly, CNNs can produce highly optimized weights.

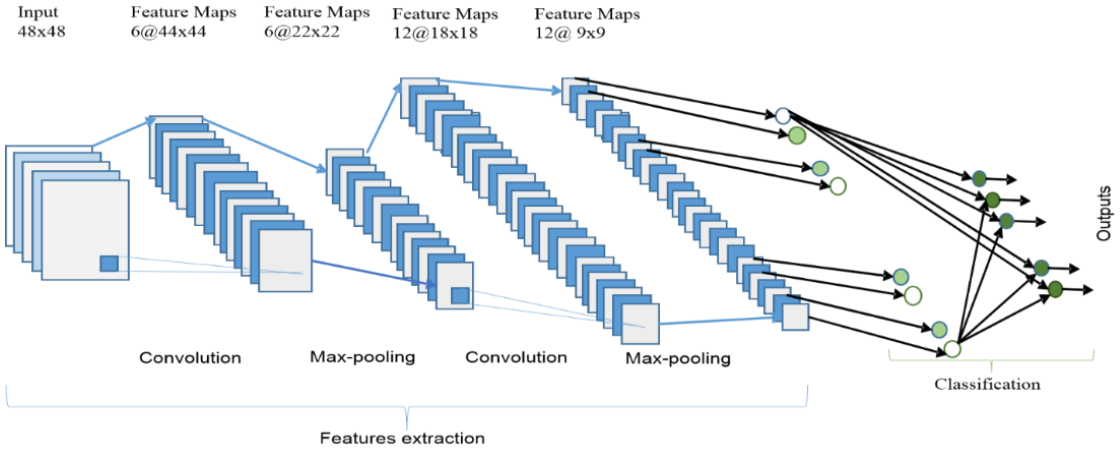


Figure 2.5: The overall architecture of the CNN includes with an input layer, multiple alternating convolution, and max-pooling layers, one fully-connected layer and classification layer.

Figure 2.5 shows the overall architecture of CNNs consists of two main parts: feature extractors and a classifier. In the feature extraction layers, each layer of the network receives the output from

its immediate previous layer as its input and passes its output as the input to the next layer. The CNN architecture consists of a combination of three types of layers: convolution, max-pooling, and classification. There are two types of layers in the low and middle-level of the network: convolutional layers and max-pooling layers. The even numbered layers are for convolutions and the odd-numbered layers are for max-pooling operations. The output nodes of the convolution and max-pooling layers are grouped into a 2D plane called feature mapping. Each plane of a layer is usually derived from the combination of one or more planes of previous layers. The nodes of a plane are connected to a small region of each connected planes of the previous layer. Each node of the convolution layer extracts the features from the input images by convolution operations on the input nodes. Higher-level features are derived from features propagated from lower level layers. As the features propagate to the highest layer or level, the dimensions of features are reduced depending on the size of the kernel for the convolutional and max-pooling operations respectively. However, the number of feature maps usually increased for representing better features of the input images for ensuring classification accuracy. The output of the last layer of the CNN is used as the input to a fully connected network which is called classification layer. Feed-forward neural networks have been used as the classification layer as they have better performance [1, 2]. In the classification layer, the desired number of features are selected as inputs with respect to the dimension of the weight matrix of the final neural network. However, the fully connected layers are expensive in terms of network or learning parameters. Nowadays, there are several new techniques including average pooling and global average pooling that is used as an alternative of fully-connected networks. The score of the respective class is calculated in the top classification layer using a soft-max layer. Based on the highest score, the classifier gives output for the corresponding classes. Mathematical details on different layers of CNNs are discussed in the following section.

Convolution Layer: In this layer, feature maps from previous layers are convolved with learnable kernels. The output of the kernels goes through a linear or non-linear activation function such as a(sigmoid, hyperbolic tangent, Softmax, rectified linear, and identity functions) to form the output feature maps. Each of the output feature maps can be combined with more than one input feature map. In general, we have that

$$x_j^l = f\left(\sum_{i \epsilon M_j} x_i^{l-1} * k_{ij}^l + b_j^l\right) \tag{2.21}$$

where x_j^l is the output of the current layer, x_i^{l-1} is the previous layer output, k_{ij}^l is the kernel for the present layer, and b_j^l are biases for the current layer. M_j represents a selection of input maps. For each output map, an additive bias b is given. However, the input maps will be convolved with distinct kernels to generate the corresponding output maps. The output maps finally go through a linear or non-linear activation function (such as sigmoid, hyperbolic tangent, Softmax, rectified linear, or identity functions).

Sub-sampling Layer: The subsampling layer performs the downsampled operation on the input maps. This is commonly known as the pooling layer. In this layer, the number of input and output feature maps does not change. For example, if there are N input maps, then there will be exactly N output maps. Due to the downsampling operation, the size of each dimension of the output maps will be reduced, depending on the size of the downsampling mask. For example: if a 2×2 downsampling kernel is used, then each output dimension will be half of the corresponding input dimension for all the images. This operation can be formulated as

$$x_j^l = \text{down}\left(x_j^{l-1}\right) \tag{2.22}$$

where $\text{down}(.)$ represents a sub-sampling function. Two types of operations are mostly performed in this layer: average pooling or max-pooling. In the case of the average pooling approach, the function usually sums up over N×N patches of the feature maps from the previous layer and selects the average value. On the other hand, in the case of max-pooling, the highest value is selected from

the N×N patches of the feature maps. Therefore, the output map dimensions are reduced by n times. In some special cases, each output map is multiplied with a scalar. Some alternative sub-sampling layers have been proposed, such as fractional max-pooling layer and sub-sampling with convolution.

Classification Layer: This is the fully connected layer which computes the score of each class from the extracted features from a convolutional layer in the preceding steps. The final layer feature maps are represented as vectors with scalar values which are passed to the fully connected layers. The fully connected feed-forward neural layers are used as a soft-max classification layer. There are no strict rules on the number of layers which are incorporated in the network model. However, in most cases, two to four layers have been observed in different architectures including LeNet [18], AlexNet [7], and VGG Net [9]. As the fully connected layers are expensive in terms of computation, alternative approaches have been proposed during the last few years. These include the global average pooling layer and the average pooling layer which help to reduce the number of parameters in the network significantly.

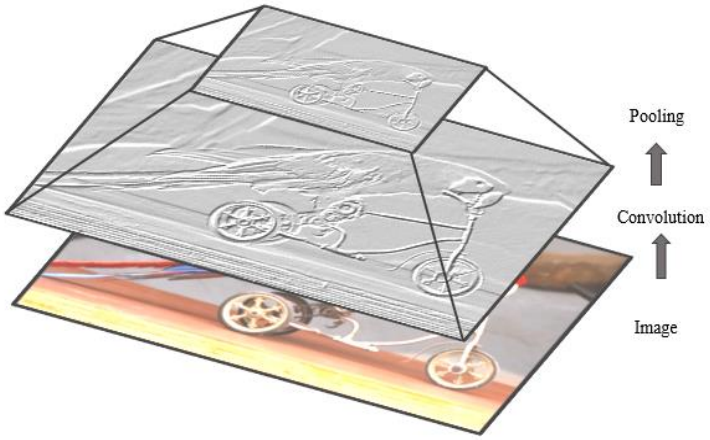


Figure 2.6: Example of convolution and pooling operation.

In the backward propagation through the CNNs, the fully connected layers update following the general approach of fully connected neural networks (FCNN). The filters of the convolutional layers are updated by performing the full convolutional operation on the feature maps between the

convolutional layer and its immediate previous layer. Figure 2.6 shows the basic operations in the convolution and sub-sampling of an input image.

2.3.1 Network parameters and required memory for CNN

The number of computational parameters is an important metric to measure the complexity of a deep learning model. The size of the output feature maps can be formulated as follows:

$$M = \frac{(N-F)}{S} + 1 \tag{2.23}$$

Where N refers to the dimensions of the input feature maps, F refers to the dimensions of the filters or the receptive field, M refers to the dimensions of output feature maps, and S stands for the stride length. Padding is typically applied during the convolution operations to ensure the input and output feature map have the same dimensions. The amount of padding depends on the size of the kernel. Equation 2.24 is used for determining the number of rows and columns for padding.

$$P = (F-1)/2 \tag{2.24}$$

Here P is the amount of padding and F refers to the dimension of the kernels. Several criteria are considered for comparing the models. However, in most of the cases, the number of network parameters and the total amount of memory are considered. The number of parameters ($Parm_l$) of l^{th} layer is calculated based on the following equation:

$$Parm_l = (F \times F \times FM_{l-1}) \times FM_l \tag{2.25}$$

If bias is added with the weights, then the above equation can be written as follows:

$$Parm_l = (F \times (F+1) \times FM_{l-1}) \times FM_l \tag{2.26}$$

Here the total number of parameters of l^{th} layer can be represented with P_l , FM_l is for the total number of output feature maps, and FM_{l-1} is the total number of input feature maps or channels. For example, let's assume the l^{th} layer has $FM_{l-1} = 32$ input features maps, $FM_l = 64$ output feature maps and the filter size is $F = 5$. In this case, the total number of parameters with a bias for this layer is $Parm_l = (5 \times 5 \times 33) \times 64 = 528{,}000$. Thus, the amount of memory (Mem_l) needs for the operations of the l^{th} layer can be expressed as

$$\text{Mem}_l = (N_l \times N_l \times FM_l) \quad (2.27)$$

2.4 Popular CNN Architectures

We will now examine several popular state-of-the-art CNN architectures. In general, most deep convolutional neural networks are made of a key set of basic layers, including the convolution layer, the sub-sampling layer, dense layers, and the soft-max layer. The architectures typically consist of stacks of several convolutional layers and max-pooling layers followed by a fully connected and SoftMax layers at the end. Some examples of such models are LeNet [18], AlexNet [7], VGG Net [9], NiN [19] and all convolutional (All Conv) [20]. Other alternatives and more efficient advanced architectures have been proposed including GoogLeNet with Inception units [22, 23], Residual Networks [11], DenseNet [27], and FractalNet [26]. The basic building components (convolution and pooling) are almost the same across these architectures. However, some topological differences are observed in the modern deep learning architectures. Of the many DCNN architectures, AlexNet [7], VGG [9], GoogLeNet [10, 64], Dense CNN [27] and FractalNet [16] are generally considered the most popular architectures because of their state-of-the-art performance on different benchmarks for object recognition tasks. Among all of these structures,

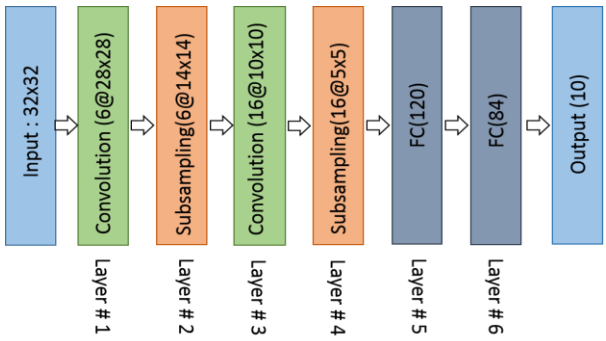


Figure 2.7: Architecture of LeNet model.

some of the architectures are designed especially for large-scale data analysis (such as GoogLeNet and ResNet), whereas the VGG network is considered a general architecture. Some of the architectures are dense in terms of connectivity, such as DenseNet [27]. Fractal Network is an alternative of ResNet.

2.4.1 LeNet

Although LeNet was proposed in the 1990s, limited computation capability and memory capacity made the algorithm difficult to implement until about 2010 [18]. LeCun, however, proposed CNNs with the back-propagation algorithm and experimented on handwritten digits dataset to achieve state-of-the-art accuracies. His architecture is well known as LeNet-5 [18]. The basic configuration of LeNet-5 is (see Figure 2.7): 2 convolutions (conv) layers, 2 sub-sampling layers, 2 fully connected layers, and an output layer with the Gaussian connection. The total number of weights and Multiply and Accumulates (MACs) are 431k and 2.3M respectively. As computational hardware started improving in capability, CNNs stated becoming popular as an effective learning approach in the computer vision and machine learning communities.

2.4.2 AlexNet

In 2012, Alex Krizhevesky and others proposed a deeper and wider CNN model compared to LeNet and won the most difficult ImageNet challenge for visual object recognition called the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012 [7]. AlexNet achieved state-of-the-art recognition accuracy against all the traditional machine learning and computer vision approaches. It was a significant breakthrough in the field of machine learning and computer vision for visual recognition and classification tasks and is the point in history where interest in deep learning increased rapidly.

The architecture of AlexNet is shown in Figure 2.8. The first convolutional layer performs convolution and max-pooling with Local Response Normalization (LRN) where 96 different receptive filters are used that are 11×11 in size. The max pooling operations are performed with 3×3 filters with a stride size of 2. The same operations are performed in the second layer with 5×5 filters. 3×3 filters are used in the third, fourth, and fifth convolutional layers with 384, 384, and 296 feature maps respectively. Two fully connected (FC) layers are used with dropout followed by a Softmax layer at the end. Two networks with similar structure and the same number of feature

maps are trained in parallel for this model. Two new concepts, Local Response Normalization (LRN) and dropout, are introduced in this network. LRN can be applied in two different ways: first applying on single channel or feature maps, where an N×N patch is selected from same feature map and normalized based on the neighborhood values. Second, LRN can be applied across the channels or feature maps (neighborhood along the third dimension but a single pixel or location).

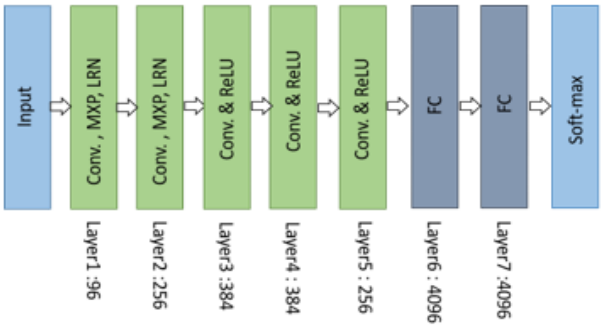


Figure 2.8: Architecture of AlexNet: Convolution, max-pooling, LRN and fully connected (FC) layer.

AlexNet has 3 convolution layers and 2 fully connected layers. When processing the ImageNet dataset, the total number of parameters for AlexNet can be calculated as follows for the first layer: input samples are 224×224×3, filters (kernels or masks) or a receptive field that has a size 11, the stride is 4, and the output of the first convolution layer is 55×55×96. According to the equations in section 3.1.4, we can calculate that this first layer has 290400 (55×55×96) neurons and 364 (11 ×11×3 = 363 + 1 bias) weights. The parameters for the first convolution layer are 290400×364 = 105,705,600. Table II shows the number of parameters for each layer in millions. The total number of weights and MACs for the whole network are 61M and 724M respectively.

2.4.3 ZFNet or Clarifai

In 2013, Matthew Zeiler and Rob Fergue won the 2013 ILSVRC with a CNN architecture which was an extension of AlexNet. The network was called ZFNet [8], after the authors' names. As CNNs are expensive computationally, an optimum use of parameters is needed from a model complexity point of view. The ZFNet architecture is an improvement of AlexNet, designed by

tweaking the network parameters of the latter. ZFNet uses 7x7 kernels instead of 11x11 kernels to significantly reduce the number of weights. This reduces the number of network parameters dramatically and improves overall recognition accuracy.

2.4.4 Network in Network (NiN)

This model is slightly different from the previous models where a couple of new concepts are introduced [19]. The first concept is to use multilayer perception convolution, where convolutions are performed with 1×1 filters that help to add more nonlinearity in the models. This helps to increase the depth of the network, which can then be regularized with dropout. This concept is used often in the bottleneck layer of a deep learning model. The second concept is to use Global Average Pooling (GAP) as an alternative of fully connected layers. This helps to reduce the number of network parameters significantly. GAP changes the network structure significantly. By applying GAP on a large feature map, we can generate a final low dimensional feature vector without reducing the dimension of the feature maps.

2.4.5 VGGNet

The Visual Geometry Group (VGG), was the runner-up of the 2014 ILSVRC [9]. The main contribution of this work is that it shows that the depth of a network is a critical component to achieve better recognition or classification accuracy in CNNs. The VGG architecture consists of two convolutional layers both of which use the ReLU activation function. Following the activation function is a single max pooling layer and several fully connected layers also using a ReLU activation function. The final layer of the model is a Softmax layer for classification. In VGG-E [9] the convolution filter size is changed to a 3x3 filter with a stride of 2. Three VGG-E [9] models, VGG-11, VGG-16, and VGG-19; were proposed the models had 11,16, and 19 layers respectively. Example of VGG architecture is shown in Figure 2.9.

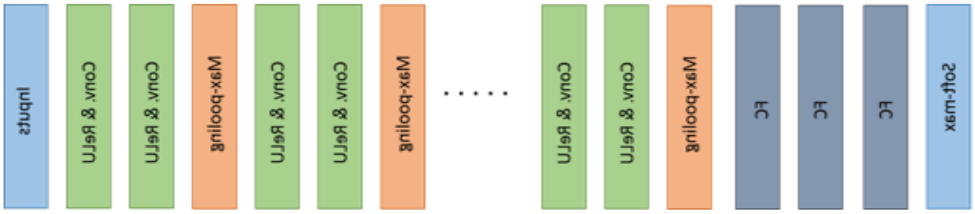


Figure 2.9: Basic building block of VGG network: Convolution (Conv) and FC for fully connected layers.

All versions of the VGG-E models ended the same with three fully connected layers. However, the number of convolution layers varied VGG-11 contained 8 convolution layers, VGG-16 had 13 convolution layers, and VGG-19 had 16 convolution layers. VGG-19, the most computational expensive model, contained 138Mweights and had 15.5M MACs.

2.4.6 GoogleNet

GoogLeNet, the winner of ILSVRC 2014[10], was a model proposed by Christian Szegedy of Google with the objective of reducing computation complexity compared to the traditional CNN. The proposed method was to incorporate "Inception Layers" that had variable receptive fields, which were created by different kernel sizes. These receptive fields created operations that captured sparse correlation patterns in the new feature map stack.

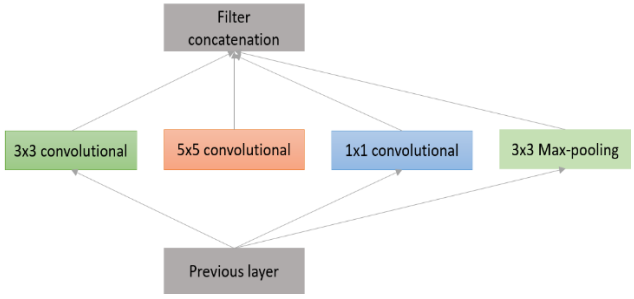


Figure 2.10: Inception layer: naive version.

The initial concept of the Inception layer can be seen in Figure 2.10. GoogLeNet improved the state of the art recognition accuracy using a stack of Inception layers seen in Figure 2.11. The difference between the naïve inception layer and final Inception Layer was the addition of 1x1

convolution kernels. These kernels allowed for dimensionality reduction before computationally expensive layers. GoogLeNet consisted of 22 layers in total, which was far greater than any network before it. However, the number of network parameters GoogLeNet used was much lower than its predecessor AlexNet or VGG. GoogLeNet had 7M network parameters when AlexNet had 60M and VGG-19 138M. The computations for GoogLeNet also were 1.53G MACs far lower than that of AlexNet or VGG.

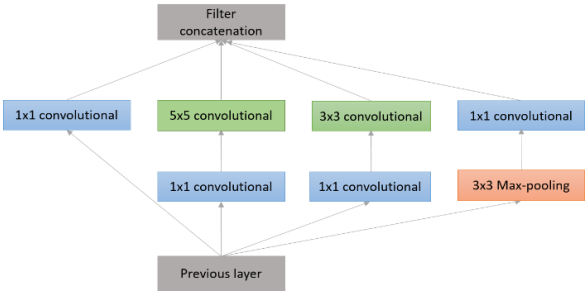


Figure 2.11: Inception layer with dimension reduction.

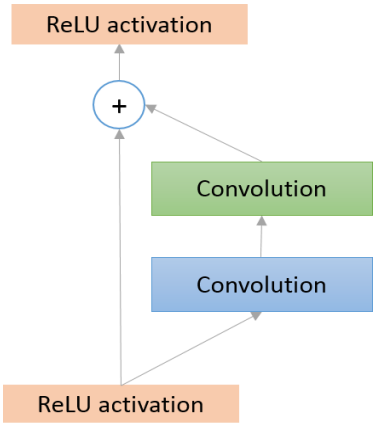


Figure 2.12: Basic diagram of Residual block.

2.4.7 Residual Network (ResNet)

The winner of ILSVRC 2015 was the Residual Network architecture, ResNet[11]. Resnet was developed by Kaiming He with the intent of designing ultra-deep networks that did not suffer from the vanishing gradient problem that predecessors had. ResNet is developed with many different

numbers of layers; 34, 50,101, 152, and even 1202. The popular ResNet50 contained 49 convolution layers and 1 fully connected layer at the end of the network. The total number of weights and MACs for the whole network are 25.5M and 3.9M respectively.

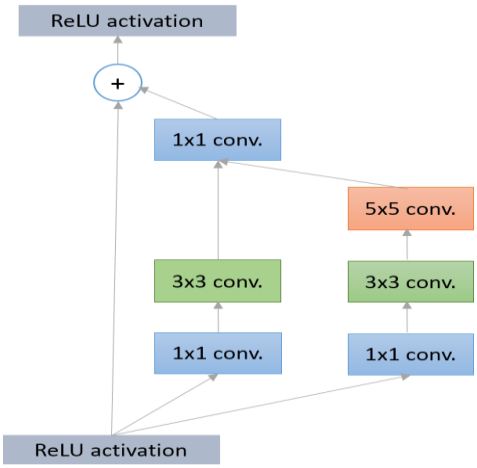


Figure 2.13: Basic block diagram for inception residual unit.

The basic block diagram of the ResNet architecture is shown in Figure 2.12. ResNet is a traditional feedforward network with a residual connection. The output of a residual layer can be defined based on the outputs of $(l-1)^{th}$ which comes from the previous layer defined as x_{l-1}. $\mathcal{F}(x_{l-1})$ is the output after performing various operations (e.g. convolution with different size of filters, Batch Normalization (BN) followed by an activation function such as a ReLU on x_{l-1}). The final output of the residual unit is x_l which can be defined with the following equation:

$$x_l = \mathcal{F}(x_{l-1}) + x_{l-1} \qquad (2.28)$$

The residual network consists of several basic residual blocks. However, the operations in the residual block can be varied depending on the different architecture of residual networks [11]. The wider version of the residual network was proposed by Zagoruvko el at. In 2016 [25]. Another improved residual network approach known as the aggregated residual transformation was proposed in 2016[25]. Recently, some other variants of residual models have been proposed based on the Residual Network architecture [20 and 25]. Furthermore, there are several advanced

architectures that have been proposed with the combination of Inception and Residual units. The basic conceptual diagram of Inception-Residual unit is shown in the following Figure 2.13.

Mathematically, this concept can be represented as

$$x_l = \mathcal{F}\left(x_{l-1}^{3\times3} \odot x_{l-1}^{5\times5} \right) + x_{l-1} \tag{2.29}$$

Where the symbol $\odot$ refers the concentration operations between two outputs from the 3×3 and 5×5 filters. After that, the convolution operation is performed with 1×1 filters. Finally, the outputs are added with the inputs of this block of x_{l-1} .The concept of Inception block with residual connections is introduced in the Inception-v4 architecture [24]. The improved version of the Inception-Residual network known as PolyNet was recently proposed [65].

2.4.8 Densely Connected Network (DenseNet)

DenseNet developed by Gao Huang and others in 2017[62], which consists of densely connected CNN layers, the outputs of each layer are connected with all successor layers in a dense block [62]. Therefore, it is formed with dense connectivity between the layers rewarding it the name "DenseNet". This concept is efficient for feature reuse, which dramatically reduces network parameters. DenseNet consists of several dense blocks and transition blocks, which are placed between two adjacent dense blocks. The conceptual diagram of a dense block is shown in Figure 2.14.

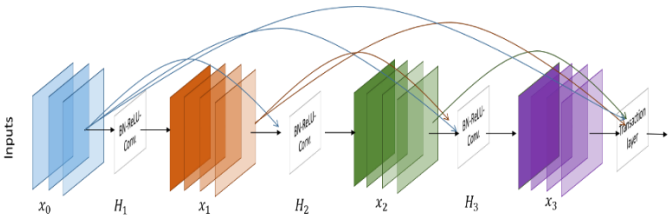


Figure 2.14: A 4-layer Dense block with a growth rate of $k = 3$.

Each layer takes all the preceding feature maps as input.

When deconstructing Figure 2.13, the l^{th} layer received all the feature maps from previous layers of $x_0, x_1, x_2 \cdots x_{l-1}$ as input:

$$x_l = H_l([x_0, x_1, x_2 \cdots x_{l-1}]) \tag{2.30}$$

Where $[x_0, x_1, x_2 \cdots x_{l-1}]$ are the concatenated features for layers $0, \cdots \cdots, l-1$ and $H_l(\cdot)$ is considered as a single tensor. It performs three different consecutive operations: Batch-Normalization (BN) [110], followed by a ReLU [58] and a 3×3 convolution operation.

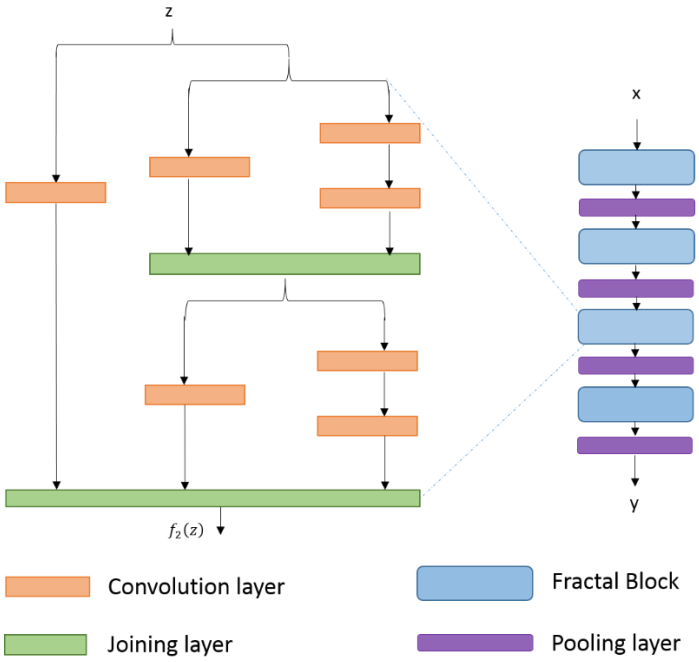


Figure 2.15: The detailed FractalNet module on the left and FractalNet on the right.

In the transaction block, 1×1 convolutional operations are performed with BN followed by a 2×2 average pooling layer. This new model shows state-of-the-art accuracy with a reasonable number of network parameters for object recognition tasks.

2.4.9 FractalNet

This architecture is an advanced and alternative architecture of ResNet model, which is efficient for designing large models with nominal depth, but shorter paths for the propagation of gradient during training [63]. This concept is based on drop-path which is another regularization approach

for making large networks. As a result, this concept helps to enforce speed versus accuracy tradeoffs. The basic block diagram of FractalNet is shown in Figure 2.15.

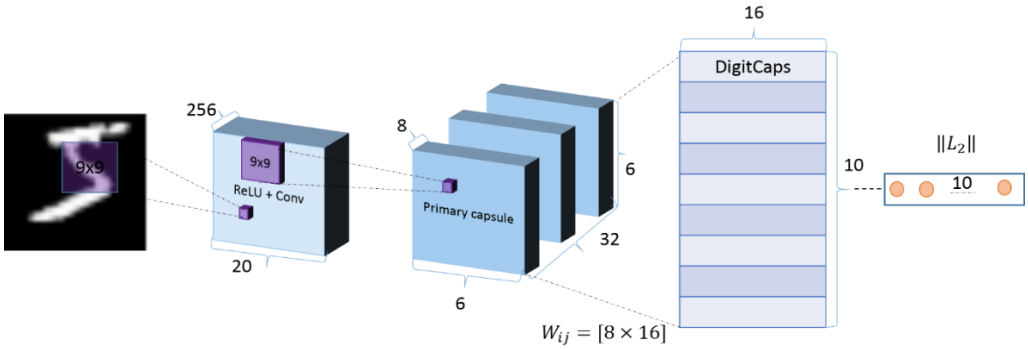


Figure 2.16: A CapsNet encoding unit with 3 layers. The instance of each class is represented with a vector of a capsule in DigitCaps layer that is used for calculating classification loss. The weights between the primary capsule layer and DigitCaps layer are represented with W_{ij}.

2.4.10 CapsuleNet

CNN's are an effective methodology for detecting features of an object and achieving good recognition performance compared to state of the art handcrafted feature detectors. There are limits to CNNs, which are that it does not take into account special relationships, perspective, size, and orientation, of features, For example: if you have a face image, it does not matter the placement of different components (nose, eye, mouth etc.) of the faces neurons of a CNN will wrongly active and recognition as face without considering special relationships (orientation, size). Now, imagine a neuron which contains the likelihood with properties of features (perspective, orientation, size etc.). This special type of neurons, capsules, can detect face efficiently with distinct information. The capsule network consists of several layers of capsule nodes. The first version of capsule network (CapsNet) consisted of three layers of capsule nodes in an encoding unit [66].

This architecture for MNIST (28×28) images, the 256 9×9 kernels are applied with a stride 1, so the output is $(28 - 9 + 1 = 20)$ with 256 feature maps. Then the outputs are fed to the primary capsule layer which is a modified convolution layer that generates an 8-D vector instead of a scalar. In the first convolutional layer, 9×9 kernels are applied with stride 2, the output dimension is

$((20-9)/2+1=6)$. The primary capsules are used 8×32 kernels which generates 32×8×6×6 (32 groups for 8 neurons with 6×6 size).

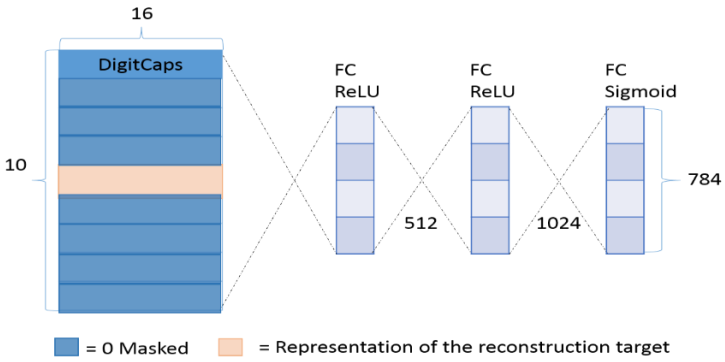


Figure 2.17: The decoding unit where a digit is reconstructed from caps layer representation. The Euclidean distance is used minimizing the error between the input sample and the reconstructed sample from the sigmoid layer. True labels are used for reconstruction target during training.

The entire encoding and decoding processes of CapsNet is shown in Figure 2.16 and Figure 2.17 respectively. We used max pooling layer in CNN often that can handle translation variance. Even if a feature moves if it is still under a max pooling window it can be detected. As the capsule contains the weighted sum of features from the previous layer, therefore this approach is capable of detecting overlapped features which is important for segmentation and detection tasks. In the traditional CNN, we have used a single cost function to evaluate the overall error which propagates backward during training. However, in this case, if the weight between two neurons is zero then the activation of a neuron is not propagated from that neuron. The signal is routed with respect to the feature parameters rather than a one size fits all cost function in iterative dynamic routing with the agreement. For details about this architecture, please see [66]. This new CNN architecture provides state-of-the-art accuracy for handwritten digits recognition on MNIST. However, from an application point of view, this architecture is more suitable for segmentation and detection tasks compare to classification tasks.

Table 2.1: The top-5% errors, network parameters and MACs for different deep CNN models.

Methods	**LeNet-5[18]**	**AlexNet [7]**	**OverFeat (fast)[8]**	**VGG-16[9]**	**GoogLeNet [10]**	**ResNet-50(v1)[11]**
Top-5 errors	n/a	16.4	14.2	7.4	6.7	5.3
Input size	28x28	227x227	231x231	224x224	224x224	224x224
Number of Conv Layers	2	5	5	16	21	50
Filter Size	5	3,5,11	3,7	3	1,3,5,7	1,3,7
Number of Feature Maps	1,6	3-256	3-1024	3-512	3-1024	3-1024
Stride	1	1,4	1,4	1	1,2	1,2
Number of Weights	26k	2.3M	16M	14.7M	6.0M	23.5M
Number of MACs	1.9M	666M	2.67G	15.3G	1.43G	3.86G
Number of FC layers	2	3	3	3	1	1
Number of Weights	406k	58.6M	130M	124M	1M	1M
Number of MACs	405k	58.6M	130M	124M	1M	1M
Total Weights	431k	61M	146M	138M	7M	25.5M
Total MACs	2.3M	724M	2.8G	15.5G	1.43G	3.9G

2.5 Comparison of Different Models

The comparison of recently proposed models based on error, network parameters, and a maximum number of connections are given in Table 2.1.

2.6 Advanced Training Techniques

What is missing in the previous section is the advanced training techniques or components which need to be considered carefully for efficient training of DL approaches. There are different

advanced techniques to apply to train a deep learning model better. The techniques including input pre-processing, a better initialization method, batch normalization, alternative convolutional approaches, advanced activation functions, alternative pooling techniques, network regularization approaches, and better optimization method for training. The following sections are discussed on individual advanced training techniques individually.

2.6.1 Preparing dataset

Presently different approaches have been applied before feeding the data to the network. The different operations to prepare a dataset are as follows; sample rescaling, mean subtraction, random cropping, flipping data with respect to the horizon or vertical axis, color jittering, PCA/ZCA whitening and many more.

2.6.2 Network initialization

The initialization of deep networks has a big impact on the overall recognition accuracy. Previously, most of the networks have been initialized with random weights. For complex tasks with high dimensionality data training, a DNN becomes difficult because weights should not be symmetrical due to the back-propagation process. Therefore, effective initialization techniques are important for training this type of DNN. However, there are many effective techniques that have been proposed during the last few years. In 1998, LeCun [67] and Y. Bengio in 2010 [68] proposed a simple but effective approach. In this method, the weights are scaled by the inverse of the square root of the number of input neurons of the layer, which can be stated $1/\sqrt{N_l}$, where N_l is the number of input neurons of l^{th} layer. The deep network initialization approach of Xavier has been proposed based on the symmetric activation function with respect to the hypothesis of linearity. This approach is known as “Xavier” initialization approach.

Recently in 2016, Dmytro M. et al. proposed Layer-sequential unit-invariance(LSUV), which is a data-driven initialization approach and provides good recognition accuracy on several benchmark datasets including ImageNet [70]. One of the popular initialization approaches has proposed by Kiming He in 2015 [69]. The distribution of the weights of l^{th} layer will be a normal distribution with mean zero and variance $\frac{2}{n_l}$ which can be expressed as follows.

$$w_l \sim \mathcal{N}\left(0, \frac{2}{n_l}\right) \tag{2.31}$$

In recent years, Convolutional Neural Networks (CNNs) have become very popular and have achieved great success in many computer vision tasks – particularly in object recognition. Partially inspired by neuroscience, CNNs share many properties with the visual system of the brain. However, the filters of convolutional layers play a vital role in the overall accuracy of CNN's. In this paper, the Cellular Simultaneous Recurrent Networks (CSRNs) is applied to generate initial filters of Convolutional Networks (CNs) for features extraction and Regularized Extreme Learning Machines (RELM) is used for classification. Furthermore, Deep Belief Networks (DBN), CNNs with random and Gabor filters are implemented to evaluate the overall performance against the proposed CSRN's filters based CNs with RELM. Experiments were conducted on three popular datasets for object recognition (such as the face, pedestrian, and car) to evaluate the performance of the proposed system. The experimental results show that in most of the cases, the proposed approach provides better performance on the extracted features using CSRN's filters with CNs compare to initialize with Gaussian random and DBN for object recognition.

2.6.3 Batch normalization

Batch normalization helps accelerate DL processes by reducing internal covariance by shifting input samples. What that means is the inputs are linearly transformed to have zero mean and unit variance. For whitened inputs, the network converges faster and shows better regularization during training, which has an impact on the overall accuracy. Since the data whitening is performed

outside of the network, there is no impact of whitening during training of the model. In the case of deep recurrent neural networks, the inputs of the n^{th} layer are the combination of $n\text{-}1^{th}$ layer, which is not raw feature inputs. As the training progresses the effect of normalization or whitening reduces respectively, which causes the vanishing gradient problem. This can slow down the entire training process and cause saturation. To better the training process during training batch normalization is then applied to the internal layers of the deep neural network. This approach ensures faster convergence in theory and during an experiment on benchmarks. In batch normalization, the features of a layer are independently normalized with mean zero and variance one [22]. The algorithm of Batch normalization is given in Algorithm 2.4.

Algorithm 2.4: Batch Normalization (BN)

Inputs: Values of x over a mini-batch: $\mathfrak{B} = \{x_{1,2,3\ldots\ldots,m}\}$

Outputs: $\{y_i = BN_{\gamma,\beta}(x_i)\}$

$\mu_{\mathfrak{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$ // mini-batch mean

$\sigma_{\mathfrak{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathfrak{B}})^2$ // mini-batch variance

$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathfrak{B}}}{\sqrt{\sigma_{\mathfrak{B}}^2 + \epsilon}}$ // normalize

$y_i = \gamma\hat{x}_i + \beta \equiv BN_{\gamma,\beta}(x_i)$ // Scaling and shifting

The parameters γ and β are used for the scale and shift factor for the normalization values, so normalization does not only depend on layer values. If you use normalization techniques, the following criterions are recommended to consider during implementation:

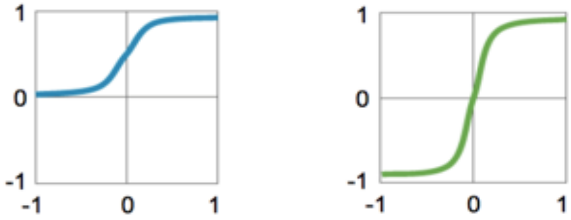
- Increase the learning rate
- Dropout (batch normalization does the same job)
- L_2 weight regularization
- Accelerating the learning rate decay
- Remove Local Response Normalization (LRN) (if you used it)
- Shuffle training sample more thoroughly
- Useless distortion of images in the training set

2.6.4 Alternative convolutional methods

Alternative and computationally efficient convolutional techniques that reduce the cost of multiplications by a factor of 2.5 have been proposed [71].

2.6.5 Activation function

The traditional Sigmoid and Tanh activation functions have been used for implementing neural network approaches in the past few decades. The graphical and mathematical representation is shown in Figure 2.18.



(a) (b)

Figure 2.18: Activation function: (a) sigmoid function and (b) Hyperbolic transient.

Sigmoid:

$$y = \frac{1}{1+e^{x}} \tag{2.32}$$

TanH:

$$y = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{2.33}$$

The popular activation function called Rectified Linear Unit (ReLU) proposed in 2010 solves the vanishing gradient problem for training deep learning approaches. The basic concept is simple to keep all the values above zero and sets all negative values to zero that is shown in Figure 2.19 [72]. The ReLU activation was first used in AlexNet, which was a breakthrough deep CNN proposed in 2012 by Hinton [7].

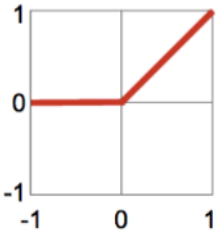
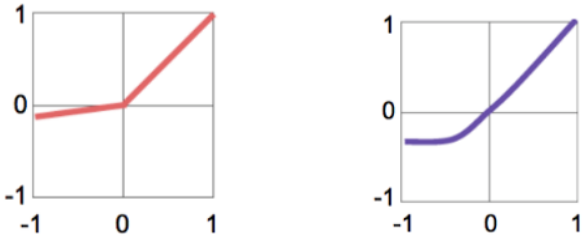


Figure 2.19: Pictorial representation of Rectified Linear Unit (ReLU).

Mathematically we can express ReLU as follows:

$$y = \max(0, x) \tag{2.34}$$

As the activation function plays a crucial role in learning the weights for deep architectures. Many researchers focus here because there is much that can be done in this area. Meanwhile, there are several improved versions of ReLU that have been proposed, which provide even better accuracy compared to the ReLU activation function.



(a) (b)

Figure 2.20: Diagram for (a) Leaky ReLU (b) Exponential Linear Unit (ELU).

An efficient improved version of ReLU activation function is called the parametric ReLU (PReLU) proposed by Kaiming He et al. in 2015. Figure 2.20 shows the pictorial representation of Leaky ReLU and ELU activation functions. This technique can automatically learn the parameters adaptively and improve the accuracy at negligible extra computing cost [73].

Leaky ReLU:

$$y = \max(ax, x) \tag{2.35}$$

Here a is a constant, the value is 0.1.

ELU:

$$y = \begin{cases} x, & x \geq 0 \\ a(e^x - 1), & x < 0 \end{cases} \tag{2.36}$$

The recent proposal of the Exponential Linear Unit activation function, which allowed for a faster and more accurate version of the DCNN structure [74]. Furthermore, tuning the negative part of activation function creates the leaky ReLU with Multiple Exponent Linear Unit (MELU) that are proposed recently [75]. S shape Rectified Linear Activation units are proposed in 2015 [76]. A survey on modern activation functions was conducted in 2015 [77].

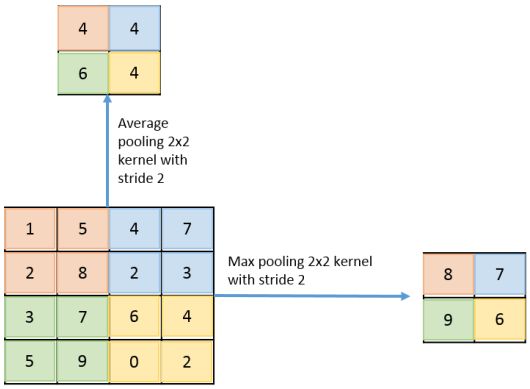


Figure 2.21: Average and max-pooling operations.

2.6.6 Sub-sampling layer or pooling layer

At present, two different techniques have been used for the implementation of deep networks in the sub-sampling or pooling layer: average and max-pooling. The concept of average pooling layer was used for the first time in LeNet [18] and AlexNet used Max-pooling layers instead of in 2012[7]. The conceptual diagram for max pooling and average pooling operation are shown in Figure 2.21. The concept of special pyramid pooling has been proposed by He et al. in 2014 which is shown in Figure 2.22 [78]. The multi-scale pyramid pooling was proposed in 2015 [79]. In 2015, Benjamin G. proposed a new architecture with Fractional max pooling, which provides state-of-the-art classification accuracy for CIFAR-10 and CIFAR-100 datasets. This structure generalizes the network by considering two important properties for a sub-sampling layer or pooling layer. First, the non-overlapped max-pooling layer limits the generalize of the deep structure of the

network, this paper proposed a network with 3x3 overlapped max-pooling with 2-stride instead of 2x2 as sub-sampling layer [80]. Another paper which has conducted research on different types of pooling approaches including mixed, gated, and tree as a generalization of pooling functions [81] which is shown in Figure 2.21.

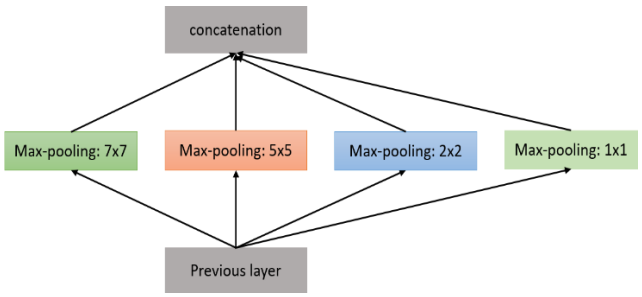


Figure 2.22: Spatial pyramid pooling operation.

2.6.7 Regularization approaches for DL

There are different regularization approaches that have been proposed in the past few years for deep CNN. The simplest but efficient approach called “dropout” was proposed by Hinton in 2012 [82]. In Dropout a randomly selected subset of activations is set to zero within a layer [83]. The dropout concept is shown in Figure 2.23.

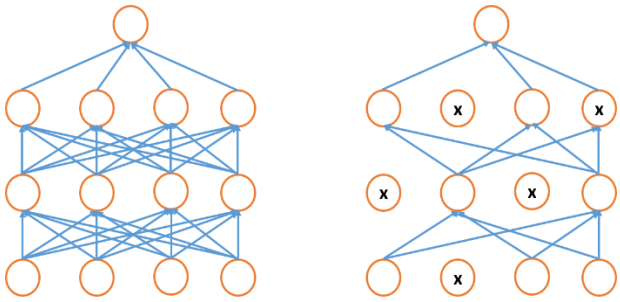


Figure 2.23: Pictorial representation of the concept Dropout.

Another regularization approach is called Drop Connect, in this case, instead of dropping the activation, the subset of weights within the network layers are set to zero. As a result, each layer

receives the randomly selected subset of units from the immediate previous layer [84]. Some other regularization approaches are proposed as well, details in [85].

2.6.8 Optimization methods for DL

There are different optimization methods such as SGD, Adagrad, AdaDelta, RMSprop, and Adam [86]. Some activation functions have been improved upon such as in the case of SGD where it has been proposed with an added variable momentum, which improved training and testing accuracy. In the case of Adagrad, the main contribution was to calculate adaptive learning rate during training. For this method, the summation of the magnitude of the gradient is considered to calculate the adaptive learning rate. In the case with a large number of epochs, the summation of the magnitude of the gradient becomes large. The result of this is the learning rate decreases radically, which causes the gradient to approach zero quickly. The main drawback to this approach is that it causes problems during training. Later, RMSprop was proposed considering only the magnitude of the gradient of the immediately previous iteration, which prevents the problems with Adagrad and provides better performance in some cases. The Adam optimization approach is proposed based on the momentum and the magnitude of the gradient for calculating adaptive learning rate similar RMSprop. Adam has improved overall accuracy and helps for efficient training with the better convergence of deep learning algorithms [87]. The improved version of the Adam optimization approach has been proposed recently, which is called EVE. EVE provides even better performance with fast and accurate convergence [88].

CHAPTER 3

SPELM FOR OBJECT RECOGNITION

Extreme Learning Machines (ELM) has been introduced as a new algorithm for training single hidden layer feedforward neural networks (SLFNs) instead of the classical gradient-based approaches. Based on the consistency property of data, which enforces similar samples to share similar properties, ELM is a biologically inspired learning algorithm that learns much faster with good generalization and performs well in classification tasks. However, the stochastic characteristics of hidden layer outputs from the random generation of the weight matrix in current ELMs lead to the possibility of unstable outputs in the learning and testing phases. This is detrimental to the overall performance when many repeated trials are conducted. To cope with this issue, we present a new ELM approach, named State Preserving Extreme Leaning Machine (SPELM). SPELM ensures the overall training and testing performance of the classical ELM while monotonically increases its accuracy by preserving state variables. For evaluation, experiments are performed on different benchmark datasets including applications in face recognition, pedestrian detection, and network intrusion detection for cybersecurity. Several popular feature extraction techniques, namely Gabor, Pyramid Histogram of Oriented Gradients (PHOG), and Local Binary Pattern (LBP) are also incorporated with SPELM. Experimental results show that our SPELM algorithm yields the best performance on tested data over ELM and RELM.

3.1 Introduction

Extreme Learning Machine (ELM) has attracted more and more attention of the community in the field of machine learning due to its higher regularization performance at a much faster speed [59-

60]. The basic principle of ELM can be described as when the input weight and bias are randomly allocated, the output weights are computed by the generalized inverse of the hidden layer outputs matrix(H). ELM can be viewed as a single hidden layer feedforward neural network (SLFN) with L hidden neurons that can learn L distinct samples with zero error. Even if the number of hidden neurons is less than the number of distinct samples, ELM still can assign random parameters to the hidden nodes and calculate the output weights using the pseudo-inverse of H giving only a small error $\epsilon > 0$. The hidden node parameters, i.e., input weights and biases or centers and impact factors, do not need to be tuned during training and may simply be assigned with random values [59-64]. Many studies have been conducted in the field of ELM from both theoretical and application aspects. Huang et al. introduced an incremental constructive method to universally approximate the parameters in ELM where the number of hidden neurons has been generated randomly to SLFNs one by one or group by a group [64]. ELM has several advantages, such as ease of use, faster learning speed, higher generalization performance, and is suitable for many nonlinear activation functions as well as kernel functions. It has also been shown that ELM yields much better generalization performance with much faster learning speed and less human interventions than other conventional methods.

From our points of view, there are two aspects that influence the robustness properties in ELM neural networks: 1) the computational robustness related to numerical stability, and 2) outliers robustness. The first aspect is generally ignored since many efforts emphasize the accuracy of applications [89]. Those computational problems occur when the hidden layer output matrix is ill-conditioned – typically caused by the random input weights and biased selection. This makes the linear system, used to train the output weights, result in a solution sensitive to data perturbation and become a poor estimation of the truth [89]. Additionally, it is known that the size of the output layer weight is more related for the generalization competency than the configuration of the neural

network, in terms of a number of neurons and format of activation function [64], [90]. Several studies [9-11] explore this issue.

The second aspect, related to outlier robustness, has been discovered in recent years in a few articles, using estimation methods that are known for being less sensitive to outliers than the Ordinary Least Squares (OLS). Studies such as Huynh and Wong [91] substitute the singular value decomposition method by the weighted least squares, which is similar to OLS but creates penalties corresponding to training patterns to weight their contribution to the final solution. Barros et al. [92] concentrate their efforts on robust classification problems with a proposal of an ELM that used Iteratively Reweighted Least Squares (IRLS), named ROB-ELM. Horata et al [93] address both aspects by applying three estimation methods: IRLS, the Multivariate Least-Trimmed Squares (MLTS) estimator and the One-Step Reweighted MLTS (RMLTS) modified by Extended Complete Orthogonal Decomposition (ECOD), which acts over the computational problem.

In this paper, we consider both aspects to achieve the improved performance of ELM. Based on the regularized extreme learning machine (RELM) [64], which on the concept of similar samples should share similar properties, we propose a State Preserving Extreme Learning Machine (SPELM). This is achieved by preserving and updating state variables that are instrumental to system accuracy. The experimental results demonstrate that the SPELM can achieve much better performance in comparison with conventional ELM and RELM. To evaluate the performance of the approach, we test the SPELM on three popular face recognition databases, namely Yale, CMU-AMP, and ORL.

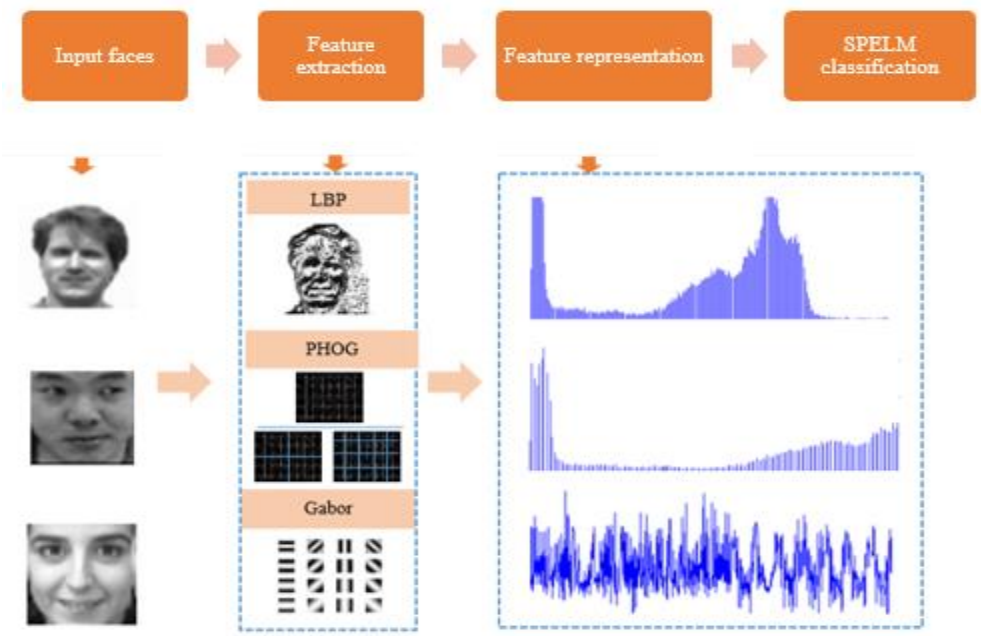


Figure 3.1: Proposed implementation scheme for face recognition.

To show the effectiveness of the SPELM, we further evaluate its performance by incorporating local appearance descriptors, such as Gabor wavelets [94], local binary patterns (LBP) [95], and pyramid histogram of orientated gradients (PHOG) features [96], into SPELM for face recognition. LBP feature is an efficient texture descriptor that extracts fine details of facial appearance and texture. In contrast, the Gabor feature captures facial shape and appearance information over a range of coarser scales [97]. The PHOG feature is computed by creating a pyramid histograms over the entire image and appending the histograms for each level of the pyramid into a single vector[96]. All these three features are rich in information content and computational efficiency. Thus, in this paper, we integrate these three feature extraction techniques with the SPELM for evaluation. Test results show that feature based SPELM yields a better face recognition accuracy. Figure 3.1 depicts the overall test scheme of the proposed algorithm. Our main contributions in this work are summarized as follows:

- A new approach of controlling state weights of RELM which leads to the proposed SPELM for a fixed number of hidden neurons generated automatically.
- Evaluation of the performance of the SPELM on face recognition by extracting facial features using three prominent feature extraction methods, namely Gabor, LBP, and PHOG.
- A comparison of the performance of SPELM with ELM and RELM.

3.2 Theoretical Analysis

In this section, we first review conventional and regularized ELM algorithms, and then introduce the proposed SPELM.

3.2.1 Extreme Learning Machine(ELM)

ELM typically applies random computational nodes in the hidden layer and increases learning speed by means of randomly generated weights and biases for hidden nodes rather than iteratively adjusting network parameters, which is commonly adopted by gradient-based methods. Different from traditional learning algorithms, ELM tends to reach not only the smallest training error but also the smallest norm of output weights [59,60].

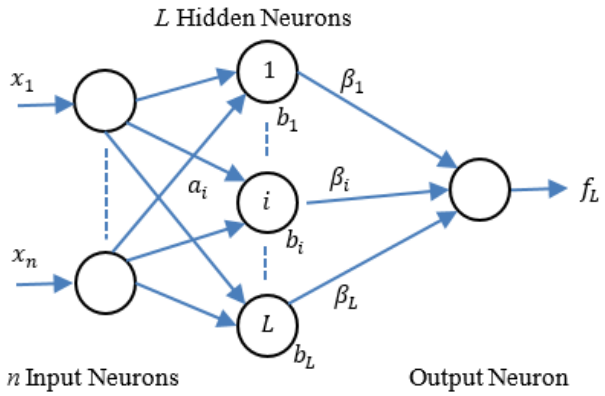


Figure 3.2: A typical architecture of the ELM.

A typical architecture of ELM is shown in Figure 3.2. The output function of ELM with L hidden nodes for generalized SLFNs is expressed as in [1]

$$f_L(x) = \sum_{i=1}^{L} \beta_i \, g_i(x) = \sum_{i=1}^{L} \beta_i G(a_i, b_i, x), x \in R^d, \beta_i \in R^m \tag{3.1}$$

where $a_i = [a_{i1}, a_{i2}, \dots\dots, a_{in}]^T$ is the weight vector connecting the input nodes to the ith hidden node, b_i is the ith bias of the hidden node, g_i denotes the output function, i.e., activation function $G(a_i, b_i, x)$ of the ith hidden node, and $\beta_i = [\beta_{i1}, \beta_{i2}, \dots\dots, \beta_{im}]^T$ is the weight vector linking the ith hidden node to the output nodes. For N arbitrary distinct samples $(x_j, t_j) \in R^d \times R^m$ the SLFNs with L hidden nodes can approximate these N samples with zero error, meaning $\sum_{j=1}^{L} \|f_j - t_j\| = 0$. Hence, there exists (a_i, b_i) and β_i such that

$$\sum_{i=1}^{L} \beta_i G(a_i, b_i, x_j) = t_j. \; j = 1,2, \dots, N \tag{3.2}$$

The above equations can be rewritten compactly as

$$H\beta = T \tag{3.3}$$

where,

$$H = \begin{bmatrix} h(x_1) \\ \vdots \\ h(x_n) \end{bmatrix} = \begin{bmatrix} G(a_1, b_1, x_1) & \dots & G(a_L, b_L, x_1,) \\ \vdots & \ddots & \vdots \\ G(a_1, b_1, x_N) & \dots & G(a_L, b_L, x_N) \end{bmatrix}_{N \times L} \tag{3.4}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L \times m}, \qquad T = \begin{bmatrix} t_1^T \\ \vdots \\ t_L^T \end{bmatrix}_{N \times m}. \tag{3.5}$$

H is the hidden layer output matrix of the SLFN, and the i^{th} column of H is the i^{th} hidden node output with respect to inputs $x_1, x_2, \cdots\cdots, x_N$, while the j^{th} row, i.e., $h(x_j)$, is the hidden layer feature mapping corresponding to the jth input x_j. As the hidden node parameters (a_i, b_i) can be randomly generated and remain unchanged, the only unknown parameters in ELM are the output weight vectors β_i between the hidden layer and the output layer, which can be simply resolved by

ordinary least-square error analysis. Since ELM aims to minimize the training error $\|H\beta - T\|$ and the norm of weights$\|\beta\|$, the smallest norm least-squares solution of the above linear system is

$$\hat{\beta} = H^{\dagger}T, \tag{3.6}$$

where $H^{\dagger}$ is the Moore-Penrose generalized inverse of matrix *H* [1]. Hence, the prediction value matrix Y is expressed by

$$Y = H\hat{\beta} = HH^{\dagger}T. \tag{3.7}$$

The error matrix can be described as

$$e = \|Y - T\|^2 = \left\|HH^{\dagger}T - T\right\|^2. \tag{3.8}$$

In order to increase the stability and generalization ability of the traditional EML, Huang et al. introduced the equality constrained optimization-based ELM [4]. According to the solution of the regularized ELM, the weight vector $\hat{\beta}$ can be represented as:

$$\hat{\beta} = \left(HH^{T} + \frac{I}{C}\right)^{-1} HT^{T}, \tag{3.9}$$

where C is a constant and I is an identity matrix. If $\lambda = 1/C$, (10.9) can be rewritten as

$$\hat{\beta} = \left(HH^{T} + \lambda I\right)^{-1} HT^{T}, \tag{3.10}$$

The solution of Eq. (10.10) can be obtained by solving the following optimization problem:

$$\min_{\beta} \left\|\beta^{T}H - T\right\|_2^2 + \lambda\|\beta\|_2^2, \tag{3.11}$$

where $\|\beta\|_2^2 = \sum_{j=1}^{K} \left\|\beta_j\right\|_2^2$ is a regularization factor and $\left\|\beta_j\right\|_2^2$ denotes the $\ell_2 -$ the norm of the vectorβ_j. Furthermore, λ indicates the regularization parameter to balance the influence of the error term and the model complexity. As a result, a simple learning method for SLFNs is called extreme learning machine that may be summarized as in Algorithm 3.1 [1].

Algorithm 3.1: Conventional Extreme Learning Machine

Inputs: Training set $\aleph$ where
$\aleph = \{(x_i, t_i) | x_i \in R^n, t_i \in R^m, i = 1, \ldots\ldots, N\}$,
Activation function g(x), and number of hidden nodes $\tilde{N}$;
Output:
Step 1: Input weight w_i and bias b_i are initialized randomly, $i = 1, \ldots\ldots\ldots \tilde{N}$,
Step 2: Hidden layer outputs matrix H is calculated.
Step 3: Output weight matrix β is computed as follows:
$\beta = H^{\dagger}T$,
where $T = [t_1, \ldots\ldots t_N]^T$.

3.2.2 State Preserving ELM (SPELM)

In ELM and RELM, there are three key steps to process: firstly, weight and bias are computed randomly in each learning step; secondly, the input sequences of testing samples are generated randomly for each iteration in case of batch learning; thirdly, the input samples are shuffled according to the output sequences of each iteration. In contrast, in the SPELM, training samples are randomly selected with corresponding labels and state variables such as weight, bias, test sample sequences, and test accuracy are preserved for each iteration. Then the highest accuracy with relevant parameters is stored until the following iteration to provide a better accuracy. The same procedure will be continued until the end of the iteration. The following section explains the details of the SPELM.

In SPELM, the state variables are the number of iterations $\mathcal{K}$, the state of the network $\mathcal{S}_i$ where $i = 1 \ldots\ldots \mathcal{K}$, the accuracy of the state represented by $\mathcal{T}_{\mathcal{S}_i}$, the number of hidden nodes $\mathcal{H}_n$ of state $\mathcal{S}_i$ where$(\mathcal{H}_n)_{\mathcal{S}_i} \in \mathbb{Z}^+$, and the activation function $G(w_{\mathcal{S}_i}, b_{\mathcal{S}_i}, x)$. The number of hidden nodes $(\mathcal{H}_n)_{\mathcal{S}_i}$for the state $\mathcal{S}_i$ is calculated based on the dimension of input features (d) represents as

$$(\mathcal{H}_n)_{\mathcal{S}_i} = \psi * d \tag{3.12}$$

where ψ is a constant. Empirically we set $\psi = 10$. The output function of SPELM with $(\mathcal{H}_n)_{\mathcal{S}_i}$ hidden nodes for generalized SLFNs is expressed as:

$$f_{(\mathcal{H}_n)_{\mathcal{S}_i}}(x) = \sum_{i=1}^{\mathcal{H}_n} \beta_{\mathcal{S}_i}\, g_i(x)$$

$$= \sum_{i=1}^{\mathcal{H}_n} \beta_{\mathcal{S}_i}\, G\left(w_{\mathcal{S}_i}, b_{\mathcal{S}_i}, x\right). \tag{3.13}$$

where $x \in R^d$, $\beta_{\mathcal{S}_i} \in R^m$, $w_{\mathcal{S}_i}$ is the weight vector connecting the input nodes to the ith hidden node, $b_{\mathcal{S}_i}$ is the ith bias , and g_i denotes the output function. Hence the activation function $G\left(w_{\mathcal{S}_i}, b_{\mathcal{S}_i}, x\right)$ is for the ith hidden node of input x in state $\mathcal{S}_i$. The weight matrix $w_{\mathcal{S}_i}$ and the bias $b_{\mathcal{S}_i}$ in the state of $\mathcal{S}_i$ are updated with respect to the present accuracy $\left(\mathcal{T}_{\mathcal{S}_i}\right)$ and the immediate previous accuracy $\left(\mathcal{T}_{\mathcal{S}_{i-1}}\right)$. These terms are defined by the $Eq.(14)$ and $Eq.(15)$, respectively.

$$w_{\mathcal{S}_i} = \begin{cases} w_{\mathcal{S}_i}, & \mathcal{T}_{\mathcal{S}_i} > \mathcal{T}_{\mathcal{S}_{i-1}} \\ w_{\mathcal{S}_{i-1}}, & otherwise \end{cases} \tag{3.14}$$

and

$$b_{\mathcal{L}_i} = \begin{cases} b_{\mathcal{S}_i}, & \mathcal{T}_{\mathcal{S}_i} > \mathcal{T}_{\mathcal{S}_{i-1}} \\ b_{\mathcal{S}_{i-1}}, & otherwise \end{cases} \tag{3.15}$$

For N arbitrary distinct samples $\left(x_j, t_j\right) \in R^d \times R^m$ SLFNs with $\mathcal{H}_n$ hidden nodes can approximate these N samples with zero error. Hence $\sum_{j=1}^{\mathcal{H}_n} \left\| f_{\mathcal{H}_j} - t_j \right\| = 0$, and there exists $\left(w_{\mathcal{S}_i}, b_{\mathcal{S}_i}\right)$ and $\beta_{\mathcal{S}_i}$ such that

$$\sum_{i=1}^{\mathcal{H}_n} \beta_{\mathcal{S}_i}\; G\left(w_{\mathcal{L}_i}, b_{\mathcal{L}_i}, x_j\right) = t_j;\; j = 1,2,\dots,N \tag{3.16}$$

Both equations above can be expressed as

$$\hat{\beta}_{\mathcal{S}_i} = \left(H_{\mathcal{S}_i} {H_{\mathcal{S}_i}}^T + \lambda I\right)^{-1} H_{\mathcal{S}_i} T^T, \tag{3.17}$$

where C is a constant and I is an identity matrix. If $\lambda = 1/C$, the solution of the $Eq.(17)$ can be obtained by solving the following optimization problem:

$$\min_{\beta} \left\| {\beta_{\mathcal{S}_i}}^T H_{\mathcal{S}_i} - T \right\|_2^2 + \lambda \left\| \beta_{\mathcal{S}_i} \right\|_2^2 \tag{3.18}$$

$\beta_{\mathcal{S}_i} = \left\| \beta_{\mathcal{S}_i} \right\|_2^2 = \sum_{j=1}^{K} \left\| \beta_{\mathcal{S}_j} \right\|_2^2$ is considered as the $\ell_2 - norm$ of the vector $\beta_{\mathcal{S}_j}$ mentioned in $Eq.(18)$ and λ is the regularization parameter. In order to update the state accuracy $\mathcal{T}_{\mathcal{S}_i}$ on test examples, the prediction value matrix $\mathrm{Y}_{\mathcal{S}_i}$ is expressed by

$$\mathrm{Y}_{\mathcal{S}_i} = \mathrm{H}_{\mathcal{S}_i} \hat{\beta}_{\mathcal{S}_i} \tag{3.19}$$

The error can be described as

$$\xi_{\mathcal{S}_i} = \left\| \mathrm{Y}_{\mathcal{S}_i} - \mathrm{T} \right\|^2 \tag{3.20}$$

Finally, the test accuracy of state $\mathcal{T}_{\mathcal{S}_i}$ updates based on $\xi_{\mathcal{S}_i}$ as follow

$$\mathcal{T}_{\mathcal{S}_i} = \left(1 - \xi_{\mathcal{S}_i}\right) * 100 \tag{3.21}$$

Algorithm 3.2: State Preserving ELM

Inputs: Training set $\aleph$, where
$\aleph = \{(\mathrm{x_i}, \mathrm{t_i}) | \mathrm{x_i} \in \mathrm{R^n}, \mathrm{t_i} \in \mathrm{R^m}, \mathrm{i} = 1, \ldots\ldots, \mathrm{N}\}$,
$\mathrm{g_i(x)}$, state $\mathcal{S}_i$ ($\mathrm{i} = 1 \ldots\ldots \mathcal{K}$) , $\mathcal{T}_{\mathcal{S}_i}$, $(\mathcal{H}_n)_{\mathcal{S}_i}$ generated according to the Eq. 3.12;
Output:
Step 1: while ($\mathrm{i} \leq \mathcal{K}$) { Start: if ($\mathcal{S}_i < 2$) {
Random initialization of input weight $w_{\mathcal{S}_i}$ and bias $b_{\mathcal{S}_i}$ for first state.}
else { if $\left(\mathcal{T}_{\mathcal{S}_{i-1}} \geq \mathcal{T}_{\mathcal{S}_{i-2}}\right)$
{Update weight and bias according to the Eq. (3.14) and Eq. (3.15) }
else { Random initialization of input weight $w_{\mathcal{S}_i}$ and bias $b_{\mathcal{S}_i}$ for current state.}}
end.
Step 2: Hidden layer outputs matrix $\mathrm{f}_{(\mathcal{H}_n)_{\mathcal{S}_i}}$ is calculated according to the Eq. (3.13)
Step3: Output weight matrix $\hat{\beta}_{\mathcal{S}_i}$ with $\ell_2 - \mathrm{norm}$ is computed according to Eq. (3.17)
Step 4: preserve all state variables
$\mathrm{i} = \mathrm{i} + 1$;
} end while

The implementation of the above mentioned SPELM algorithm can be expressed as in Algorithm 3.2:

3.3 Experimental Results

This section presents the experimental results of our proposed SPELM model for face recognition. The activation function of the hidden layer is set to a 'sigmoid' function and the number of hidden nodes is fixed to $10 \times \text{numDim}$ for all ELM, RELM, and SPELM. We evaluate the performance of SPELM on face recognition from two aspects: (1) compare the SPELM model with the conventional ELM and RELM; (2) compare their performance by incorporating feature extraction techniques for face recognition.

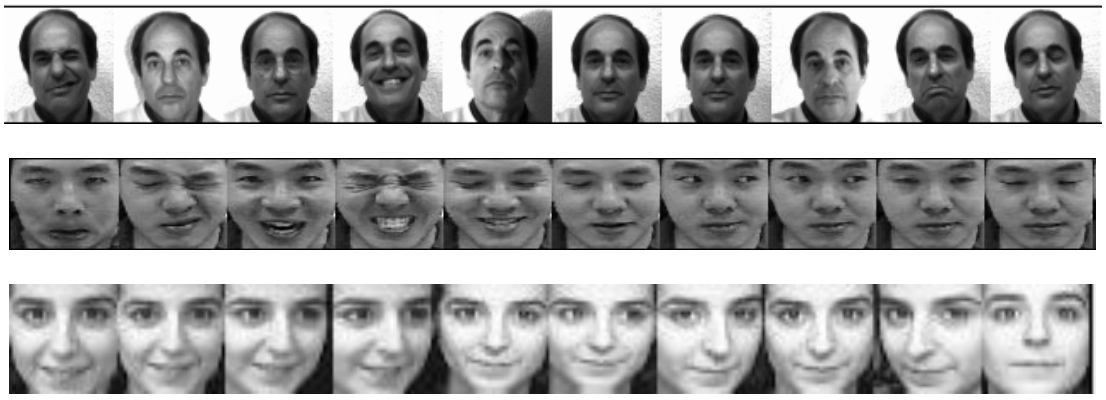


Figure 3.3: The three rows show ten image samples from the Yale, CMU-AMP databases and ORL, respectively.

3.3.1 Dataset

To evaluate the efficiency of the SPELM, we perform unconstrained face verification experiments on the Yale [98], CMU-AMP [99] and ORL [100] face recognition databases. The statistics of these datasets used in this experiment are summarized in Table 3.11. Figure 3.3 shows sample images from these three datasets, in which one subject is randomly selected from each database and each one has 10 samples. As seen in Figure 3.3, face images in these three databases contain various poses, illumination, and expressions.

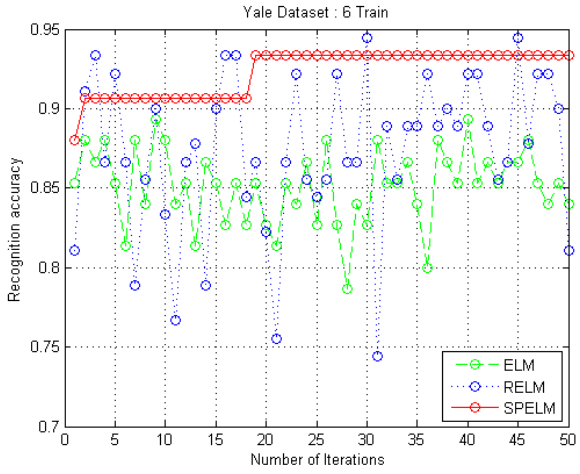
For each of the three databases, all face images are cropped and resized to 32×32 and represented as 1024-dimensional vector. Six training samples per subject are randomly chosen for training.

Table 3.1: Statistics of three face datasets used in the test.

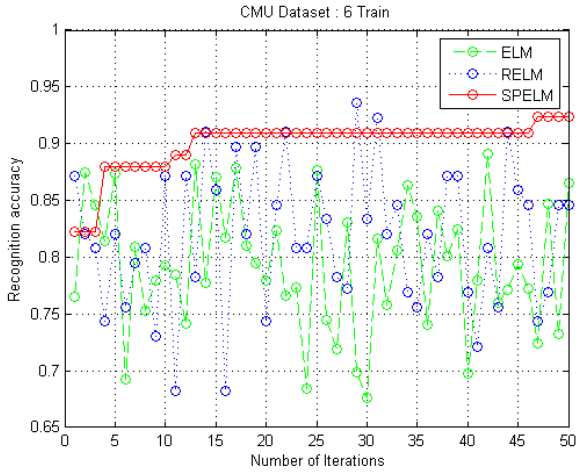
Database	samples	classes	sample/class
Yale	165	15	11
ORL	400	40	10
CMU-AMP	975	13	75

3.3.2 Results and comparison

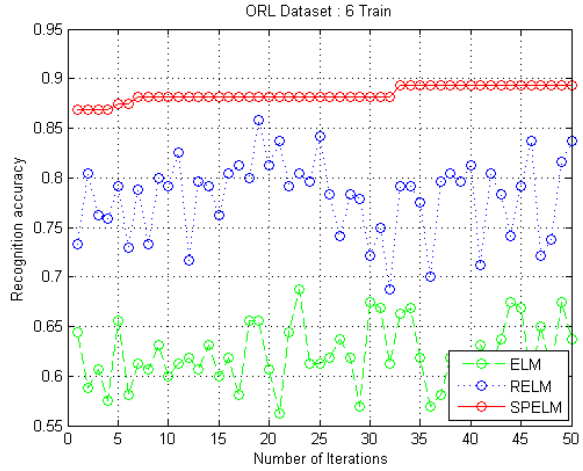
In this experiment, we compare the SPELM model with ELM and RELM. The algorithm procedure is repeated 50 times to produce a better estimation of recognition accuracy. Figure 3.4 illustrates the recognition results on the Yale, CMU-AMP, and ORL face databases without applying any feature extraction. From Figure 3.4, it is evident that the proposed SPELM model yields better performance on all three datasets. In each iteration stage, SPELM gives a better recognition rate than conventional ELM and RELM for the fixed number of hidden nodes generated automatically.



(a)



(b)



(c)

Figure 3.4: Training results on face recognition using ELM, RELM, and SPELM: (a) Yale, (b) CMU, and (c) ORL datasets.

3.3.3 Face recognition

Due to the state persevering properties of SPELM, the recognition accuracy is monotonically increasing during each iteration. In ELM and RELM the accuracy can decrease in any iteration as shown in Figure 3.4. This is because SPELM preserves the output weight variables and adaptively updates them when superior weights are obtained. Figure 3.4 shows that although the number of hidden neurons is fixed in each iteration, the overall performance of the ELM and RELM networks show scholastic behavior on the outputs. This is due to their random generation of weights and bias in each state. In contrast, SPELM yields monotonically increasing output accuracy with respect to

iterations. This adaptive learning property would significantly boost the learning characteristics of ELM for achieving a better classification accuracy.

Table 3.2: Face recognition accuracy (mean ± std.-dev. %).

Methods	Yale Database		
	LBP	PHOG	Gabor
ELM	77.47±4.06	99.33±0.51	97.47±1.22
RELM	82.23±1.12	99.53±0.65	98.07±0.81
SPELM	**85.45**±1.33	**100.00**±0.00	**99.80**±0.13
	CMU-AMP Database		
	LBP	PHOG	Gabor
ELM	93.66±0.68	99.70±0.18	100.00±0.00
RELM	96.08±0.24	99.55±0.16	100.00±0.00
SPELM	**96.96**±0.12	**99.81**±0.16	**100.00**±0.00
	ORL Database		
	LBP	PHOG	Gabor
ELM	58.50±2.39	90.06±1.03	97.50±0.35
RELM	76.63±1.85	91.19±0.95	97.56±0.35
SPELM	**79.47**±1.79	**92.45**±1.55	**97.97**±0.28

Feature embedding: To further demonstrate the efficiency of SPELM, we apply some popular feature extraction techniques, namely LBP, PHOG, and Gabor, on the raw inputs, and then perform the ELM based classification. In this experiment, the LBP feature vector is set to a length of 256. For PHOG we chose three pyramid levels with 9 bin histograms for each grid cell. In Gabor, 16 filters were used with a size of 8×8. Table 3.2 shows the face recognition accuracy of ELM, RELM, and SPELM using these three features separately. These results show that SPELM provides the best performance in all three face datasets, thus demonstrating its robustness. To better visualize the test results, Figures 3.5, 3.6, and 3.7 provide comparative histograms corresponding to Table 2 that show face recognition rate along with standard deviation on the Yale, CMU-AMP, and ORL face databases, respectively.

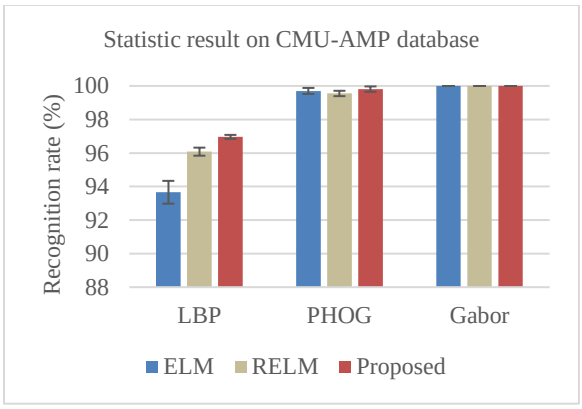


Figure 3.5: Testing result on Yale Dataset with respect to LBP, PHOG, and Gabor features.

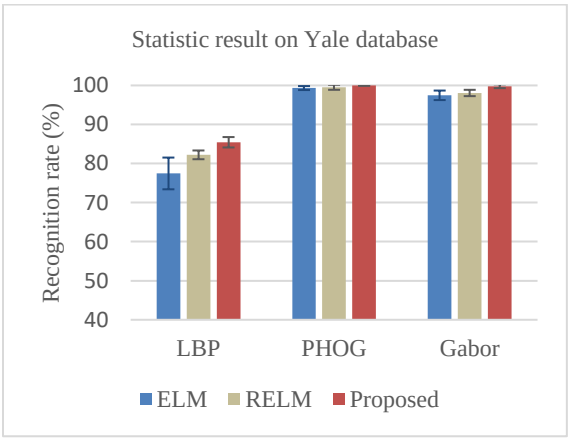


Figure 3.6: Testing result on CMU-AMP dataset with respect to LBP, PHOG, and Gabor features.

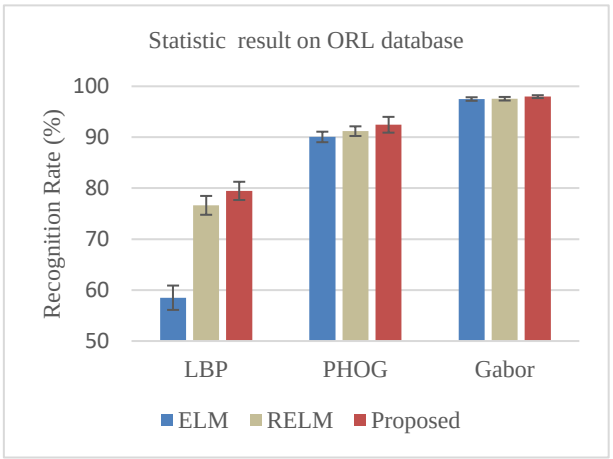


Figure 3.7: Testing result on ORL dataset with respect to LBP, PHOG, and Gabor features.

Time efficiency: The state preserving characteristics of the SPELM also contributes to computation speed. In ELM, the weights and bias are generated randomly, whereas the variables are only recomputed if a higher accuracy is found in SPELM. This saves a significant amount of memory and enhances the system processing speed. To experimentally show these merits, we used a desktop computer with a 1.7 GHz processor and 6GB of RAM to evaluate the processing time in MATLAB (R2014a). The evaluation is conducted on the three face databases using ELM, RELM, and SPELM. To avoid any bias, we repeat the evaluation is conducted on the three face databases using ELM, RELM, and SPELM. To avoid any bias, we repeat the experiments 50 times (iteration) and compute the average processing speed as shown in Table 3.3. From Table 3.3, it is clear that SPELM is fastest.

Table 3.3: A comparison of computation time (mean: sec./iteration).

Database	ELM	RELM	SPELM
Yale	0.406	0.385	**0.278**
ORL	0.230	0.155	**0.140**
CMU-AMP	0.244	0.165	**0.150**

3.3.4 Pedestrian detection

Multi-view INRIA pedestrian dataset [101] is used in this experiment. The image size in this database is 80×32. The proposed method is trained with 2000 multi-view positive and 3000 negative images. Some of the positive examples are shown in Figure 3.8. In SPELM, 2560 nodes are set for the input layer, 1000 neurons for the hidden layer, and 2 nodes for the output layer. A total of 30 trials were conducted for training and testing.

Figure 3.8: Example positive images in the multi-view pedestrian database.

Figure 3.9 shows pedestrian detection accuracy on INRIA dataset. It can be observed that the proposed PHOG+SPELM outperforms PHOG+ELM and PHOG+RELM in most of the trials and yields higher average recognition accuracy. From Figure 3.9, it can be seen that the accuracy of PHOG+SPELM rises as a number of trial increases. This is because of the inherent character of SPELM, in which it always preserves better weights and biases for achieving a better or equal recognition accuracy in each trial. The overall accuracy of PHOG+ELM, PHOG+RELM, and PHOG+SPELM are presented in Table 3.4 and the highest accuracy is indicated in boldface.

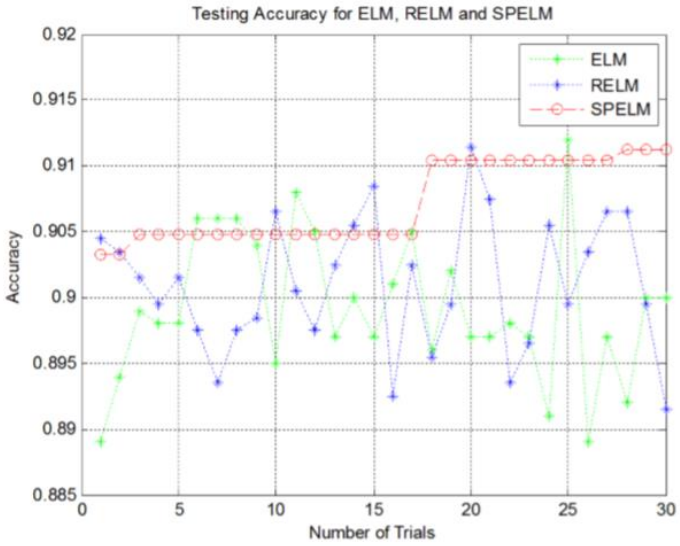


Figure 3.9: Recognition accuracy for pedestrian classification.

Table 3.4 Performance comparison and the best accuracy for each dataset is shown in boldface.

Methods	Pedestrian detection accuracy
PHOG+ELM	89.92
PHOG+RELM	90.10
PHOG+SPELM	**90.70**

3.3.5 Network intrusion detection

With advances in digital technology, security threats for computer networks have increased dramatically over the last decade, thus there is a need to develop more accurate systems for cybersecurity. In this paper, we explore the capabilities of SPELM on intrusion detection for cybersecurity using KDD Cup 1999 dataset [102]. In addition, the robustness of SPELM is evaluated using dimensionality reduction technique such as Principal Component Analysis (PCA) [103].

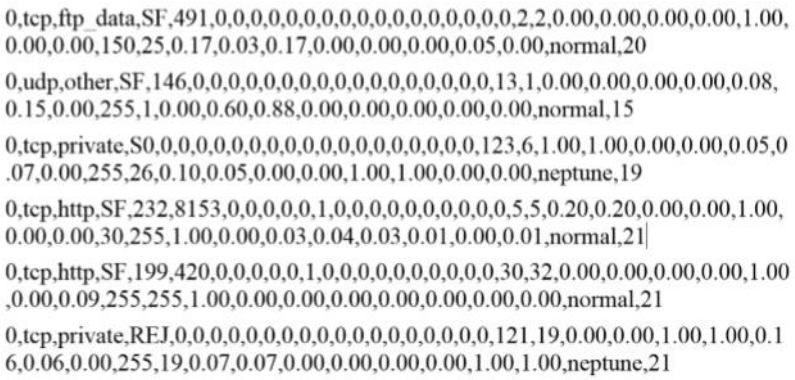


Figure 3.10: Sample data from the KDD Cup 1999 dataset.

Dataset description: The KDD Cup 1999 intrusion detection dataset is based on the DARPA (Defense Advanced Research Projects Agency) initiative to provide designers of intrusion detection systems (IDS) with a benchmark on which to evaluate different methodologies. Each KDD Cup 1999 connection record contains 41 features (e.g., protocol type, service, and flag) and is labeled as either normal or a specific type of attack as provided in Figure 3.10. The cyber-attacks can be summarized into five categories − Normal: Data with no attack. − Denial of Service (DoS):

Attacker tries to prevent legitimate users from using a service. − Probe: Attacker tries to gain information about the target host. − Remote to Local (R2L): Attacker does not have an account on the victim machine, hence tries to gain access. − User to Root (U2R): Attacker has local access to the victim machine and tries to gain superuser privileges.

Table 3.5: Categorization of network attacks.

Category	Example string	Class label
Normal	Normal network packet2	1
DoS	'Back', 'Land', 'Neptune', 'Pod', 'Smurf', 'Teardrop', 'Mailbomb', 'Processtable', 'Udpstorm', 'Apache2', 'Worm'	2
Probe	'Satan', 'IPsweep', 'Nmap', 'Portsweep', 'Mscan', 'Saint'	3
R2L	'Guess-password', 'Ftp-write', 'Imap','Phf', 'Multihop', 'Warezmaster', 'Warezclient', 'Xlock', 'Xsnoop', 'Snmpguess', 'Snmpgetattack', 'Httptunnel', 'Sendmail', 'Named'	4
U2R	'Buffer-overflow', 'Loadmodule', 'Rootkit', 'Perl', 'Sqlattack', 'Xterm', 'Ps'	5

In KDD Cup 1999, the training set contains a total of 5 categories of attack as given in Table 3.5. In this experiment, we use 25,000 numerical encoded and normalized data samples for training and testing. The components within a packet (network protocols, services, flag, etc.) are represented as strings. In order to use these with SPELM, we encode each string as a single numerical value. Table 6 represents the corresponding values of individual strings for protocol names as an instance.

Dataset preparation: The components within a packet (network protocols, services, flag, etc.) are represented as strings. In order to use these with SPELM, we encode each string as a single numerical value. Table 3.6 represents the corresponding values of individual strings for protocol names as an instance.

Table 3.6: Protocol names with corresponding values (TCP: Transmission Control Protocol; UDP: User Datagram Protocol; ICMP: Internet Control Message Protocol).

Protocol name	TCP	UDP	ICMP	Otherwise
Value	1	2	3	0

All the remaining feature strings are encoded in a similar manner and the input numerical values are then normalized by a max-min normalization method maps all values into the range of [0 1]. After this normalization, some input data features end up being encoded as all zeros. These feature vectors are discarded from the input data set, thus reducing the dimension of the input feature from 41 to 39.

Table 3.7: Testing accuracy comparison with ELM and RELM without feature dimensionality reduction.

# of training samples in %	ELM (%)	RELM (%)	Proposed SPELM (%)
20	97.41	97.78	97.96
30	97.75	97.87	98.10
40	97.84	97.99	98.15

Figures 3.11, 3.12, and 3.13 show the classification results on KDD Cup 1999 dataset using 39-dimensional feature vector, whereas Figures 3.14, 3.15 and 3.16 show the testing accuracy after reducing feature dimensions from 39 to 9 using PCA. From these results, it can be clearly seen that SPELM outperforms ELM and RELM in most of the trials, and yields higher average accuracy for both 39 and 9-dimensional feature vector as summarized in Tables 3.7 and 3.8, respectively.

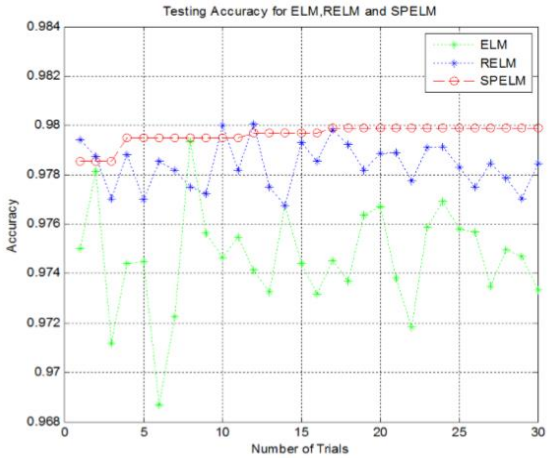


Figure 3.11: Recognition accuracy of 20% training dataset without feature dimensionality reduction.

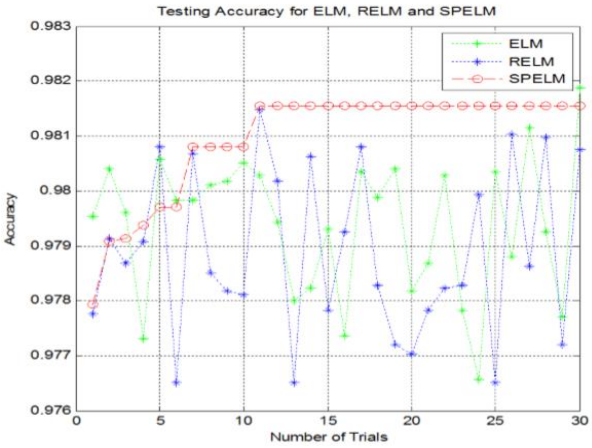


Figure 3.12: Recognition accuracy of 30% training dataset without feature dimensionality reduction.

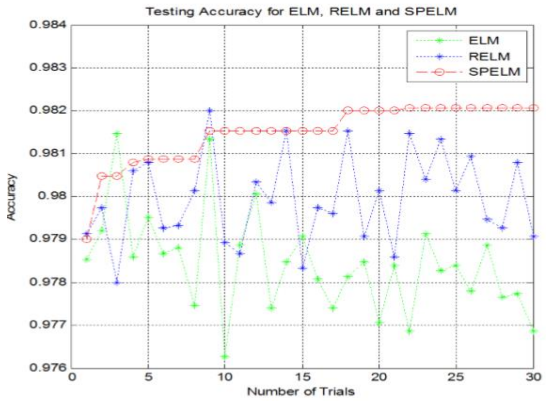


Figure 3.13: Recognition accuracy of 40% training dataset without feature dimensionality reduction.

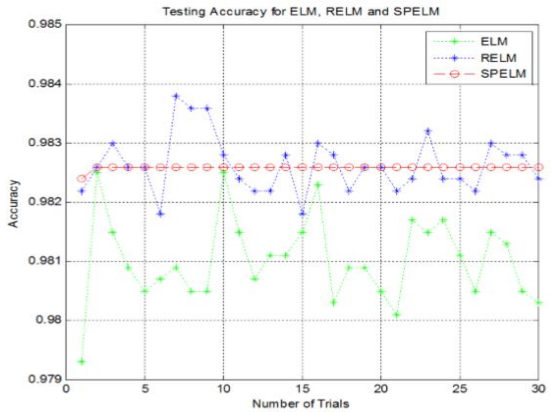


Figure 3.14: Recognition accuracy for 20% training set with 9 PCs.

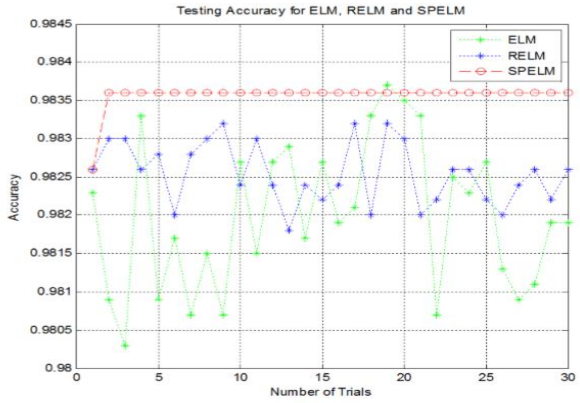


Figure 3.15: Recognition accuracy of 30% training set with 9 PCs.

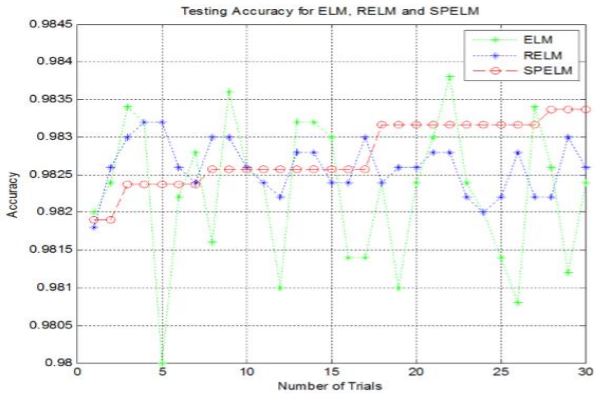


Figure 3.16: Recognition accuracy of 40% training set with 9 PCs.

Table 3.8: Accuracy comparison of SPELM with ELM and RELM using 9 PCs.

# of training samples in %	ELM (%)	RELM (%)	Proposed SPELM (%)
20	98.10	98.26	98.26
30	98.20	98.25	98.36
40	98.20	98.26	98.28

3.4 Conclusions

In this paper, we presented a new approach for computing state variables in ELM, termed as SPELM. SPELM provides monotonically increasing learning characteristics by preserving state variables in each training and testing trial. It is observed that SPELM not only improves the inherent

characteristics of the ELM based classification algorithms but also resolves the stochastic behavior of ELMs in training and testing phases. Experiments on face recognition, pedestrian detection, and network intrusion detection show the effectiveness and efficiency of SPELM.

CHAPTER 4

HRBCR USING DEEP CNNs

In spite of advances in object recognition technology, Handwritten Bangla Character Recognition (HBCR) remains largely unsolved due to the presence of many ambiguous handwritten characters and excessive cursive in Bangla handwritings. Even the best existing recognizers do not lead to satisfactory performance for practical applications related to Bangla character recognition and have much lower performance than those developed for English alpha-numeric characters. To improve the performance of HBCR, we herein present the application of the state-of-the-art Deep Convolutional Neural Networks (DCNN) including VGG Network, All Convolution Network (All-Conv Net), Network in Network (NiN), Residual Network, FractalNet, and DenseNet for HBCR. The deep learning approaches have the advantage of extracting and using feature information, improving the recognition of 2D shapes with a high degree of invariance to translation, scaling and other distortions. We systematically evaluated the performance of DCNN models on publicly available Bangla handwritten character dataset called CMATERdb and achieved the superior recognition accuracy when using DCNN models. This improvement would help in building an automatic HBCR system for practical applications.

4.1 Introduction

Automatic handwriting character recognition is of academic and commercial interest. Nowadays, Deep Learning techniques DCNN algorithms already excel in learning to recognize handwritten characters [126]. The main challenge in handwritten character classification is to deal with the enormous variety of handwriting styles by different writers in a different language. Furthermore, some complex handwriting scripts comprise different styles for writing words. Depending on

the language, characters are written isolated from each other in some cases, (*e.g.*, Thai, Laos, and Japanese). In some other cases, they are cursive and sometimes the characters relate to each other (*e.g.*, English, Bangladeshi and Arabic). This challenge is already recognized by many researchers in the field of Natural Language Processing (NLP) [104-106]. Handwritten character recognition is more challenging compared to the printed forms of the character. In addition, handwritten characters written by different writers are not identical but vary in different aspects such as size and shape. Numerous variations in writing styles of individual character make the recognition task challenging. The similarities in different character shapes, the overlaps, and the interconnections of the neighboring characters make further complicate the character recognition problem. The large variety of writing styles, writers, and the complex features of the handwritten characters are very challenging for accurately classifying the handwritten characters.

Bangla is one of the most spoken languages and ranked fifth in the world. It is also a significant language with a rich heritage; February 21st is announced as the International Mother Language Day by UNESCO to respect the language martyrs for the language in Bangladesh in the year 1952. This is the only language for which a lot of people sacrifices their life for establishing the Bangla is the first language of Bangladesh and the second most popular language in India. About 220 million people use Bangla as their speaking and writing purpose in their daily life. Therefore, automatic recognition of Bangla characters has a great significance. Different languages have different alphabets or scripts, and hence present different challenges for automatic character recognition respect to language. For instance, Bangla uses a Sanskrit based script which is fundamentally different from English or a Latin-based script. This accuracy of the results for character recognition algorithm may vary significantly depending on the script. Therefore, handwritten Bangla character recognition algorithms should be investigated with due importance [131,132].

In Bangla language, there are 10 digits and 50 characters in vowel and consonant where some contain additional sign up and/or below. Moreover, Bangla consists of many similar shaped characters. In some cases, a character differs from its similar one with a single dot or mark. Furthermore, Bangla language also contains with some special characters very often which is an equivalent representation of vowels which makes difficult to achieve a better performance with simple technique as well as hinders to the development of Bangla handwritten character recognition system. There are many applications of Bangla handwritten character recognition such as Bangla Optical Character Recognition (OCR), National ID number recognition system, automatic license plate recognition system for vehicle and parking lot management system, post office automation, online banking and many more. Some example images are shown in Figure 4.1. In this work, we investigate the handwritten character recognition on Bangla numerals, alphabets, and special characters using the state-of-the-art Deep Convolutional Neural Networks (DCNN). The contributions of this paper are summarized as follows:

- Comprehensive evaluation of the state-of-the-art DCNN models including VGG Net. [9], All-conv [20], NiN [19], ResNet [11], FractalNet [26], and DenseNet [17] on Bangla handwritten characters recognition.
- Extensive experiments on Bangla handwritten characters recognition including handwritten digits, alphabets, and special character recognition.
- The best recognition accuracy is achieved compared to many existing approaches on all experiments.

Figure 4.1: Application of Character recognition: national ID number recognition system on the left, postal code number recognition in middle and license plate recognition on the right.

4.2 Related Works

There are a few remarkable works are available for Bangla handwritten character recognition. Some literatures have been reported on Bangla characters recognition in past years [107–109], but there is only few research on handwritten Bangla numeral recognition who reach to the desired recognition accuracy. Pal *et al.* have conducted some exploring works for the issue of recognizing handwritten Bangla characters [110–112]. A survey has conducted on Bangla digits classification and recognition recently [117]. The proposed schemes are mainly based on extracted features from a concept called water reservoir. The reservoir is a concept that obtained by considering the accumulation of water poured from the top or from the bottom of the numerals. They deployed a system towards Indian postal automation. The accuracy of the handwritten Bangla and English numeral classifier is 94.13% and 93%, respectively. However, they did not mention recognition reliability and response time in their works, which are very important evaluation factors for a practical automatic letter sorting machine. Reliability indicates the relationship between the error rate and the recognition rate. Liu and Suen [113] have shown the benchmarked accuracy of recognition rate of handwritten Bangla digits on a standard dataset, namely the ISI database of handwritten Bangla numerals [114], which consists of 19392 training samples, 4000 test samples and 10 classes (i.e., 0 to 9). They have reported accuracy is 99.4% for numeral recognition. Such high accuracy has been attributed to the extracted features based on gradient direction and some advanced normalization techniques. Surinta *et al.* [115] proposed a system using a set of features such as the contour of the handwritten image computed using 8-directional codes, the distance calculated between hotspots and black pixels, and the intensity of pixel space of small blocks. Each of these features is used to a nonlinear SVM classifier separately, and the final decision has been taken based on majority voting. The dataset has been used in [115] is composed of 10920 examples, and this method achieves an accuracy of 96.8%. Xu *et al.* [116] used a hierarchical Bayesian network which directly takes raw images as the network inputs and classifies them using a bottom-up approach. Average recognition accuracy of 87.5% was achieved with a dataset consists of 2000

handwritten sample images. Sparse representation classifier is applied for Bangla digit recognition in [131] where 94% accuracy resulted for handwritten digit recognition. In [118], the handwritten Bangla basic and compound character recognition using MLP and SVM classifier has been proposed and they achieved around 79.73% and 80.9% of recognition rate, respectively. Handwritten Bangla numerals recognition using MLP is presented in [119] where the average recognition rate reached 96.67% using 65 hidden neurons. Das *et al.* [120] exploited genetic algorithm-based region sampling for local feature selection and achieved 97% accuracy on the handwritten Bangla numeral dataset named CMATERdb. The convolutional neural networks (CNN) based Bangla handwritten character recognition system has been introduced in [121], where the best recognition accuracy is reached at 85.36% on their own dataset for Bangla character recognition. Very recently, deep learning approaches including CNN, CNN with Gabor filters, and Deep Belief Network (DBN) have been applied to handwritten digits recognition [132]. This work has reported the improved recognition accuracy on handwritten Bangla digits recognition. These works lead to a field of deep learning for Bangla character recognition. However, in this paper, we have implemented a set of DCNN including VGG [9], All-conv [20], NiN [19], ResNet [11], FractalNet [26], and DenseNet [17] for Bangla handwritten characters (including digits, alphabets and special characters) recognition. We have achieved the state-of-the-art recognition accuracy in all the mentioned category of Bangla handwritten characters.

4.3 Deep Convolutional Neural Networks (DCNN)

In the last few years, deep leaning has proved the outstanding performance in the field of machine learning and pattern recognition. Deep Neural Networks (DNNs) model generally include Deep Belief Network (DBN) [72, 133], Stacked Auto-Encoder (SAE) [123], and CNN. Due to the composition of many layers, DNN methods are more capable of representing the highly varying nonlinear function compared to shallow learning approaches [1,124]. The low and middle level of

DNNs abstract the feature from the input image whereas the high level performs classification operation on extracted features. As a result, a uniform framework is formed by integrating with all necessary modules within a single network. Therefore, DNN models often lead to better accuracy compared with the training of each module independently. Among all deep learning approaches, CNN is one of the most popular models and has been providing the state-of-the-art recognition accuracy on object recognition [7], segmentation [174], human activity analysis [175], image super-resolution [176], object detection [1], scene understanding [178], tracking [179], and image captioning [180].

Table 4.1: Database statistics used in our experiment.

Dataset	**Number of training samples**	**Number of testing samples**	**Total samples**	**Number of classes**
Digit-10	4000	2000	6000	10
Alphabet-50	12,000	3,000	15,000	50
SpecialChar-13	2196	935	2231	13

4.4 Experimental Results and Discussion

The entire experiment has been conducted on Desktop computer with Intel® Core-I7 CPU @ 3.33 GHz, 56.00GB memory, and Keras with Theano on the backend on Linux environment. We evaluate the state-of-the-art DCNN models on three datasets for Bangla handwritten digits, alphabets, and special characters recognition. The statistics of the three datasets are summarized in Table 4.1. For convenience, we name the datasets as Digit-10, Alphabet-50, and SpecialChar-13, respectively. We have rescaled all the images to 32×32 pixels for this experiment.

4.4.1 Bangla handwritten digit dataset

We evaluate the performance of both DBN and CNN on a Bangla handwritten benchmark dataset called CMATERdb 3.1.1 [120]. The standard samples of the numeral with respective English numeral are shown in Figure 8.

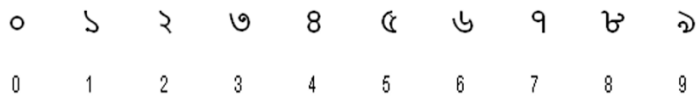


Figure 4.2: Bangla actual digits in the first row and the second row shows the corresponding English digits.

This dataset contains 6000 images of unconstrained handwritten isolated Bangla numerals. Each digit has 600 images that are rescaled to 32×32 pixels. Actual Bangla digits and equivalent Arabic numerals are shown in Figure 4.2. Some sample images of the database are shown in Figure 4.3. Visual inspection depicts that there is no visible noise. However, variability in writing style due to user dependency is quite high. The dataset was split into a training set and a test set for the evaluation of different DCNN models. The training set consists of 4000 images (400 randomly selected images of each digit). The rest of the 2000 images are used for testing [132].

Figure 4.3: Sample handwritten Bangla numeral images: 0-9 illustrate some randomly selected handwritten Bangla numeral images.

Figure 4.4 shows the training loss of all DCNN models for 250 epochs. It can be observed that FractalNet and DenseNet converge faster compared to other networks, and the worst convergence is obtained to be for the All-Conv network.

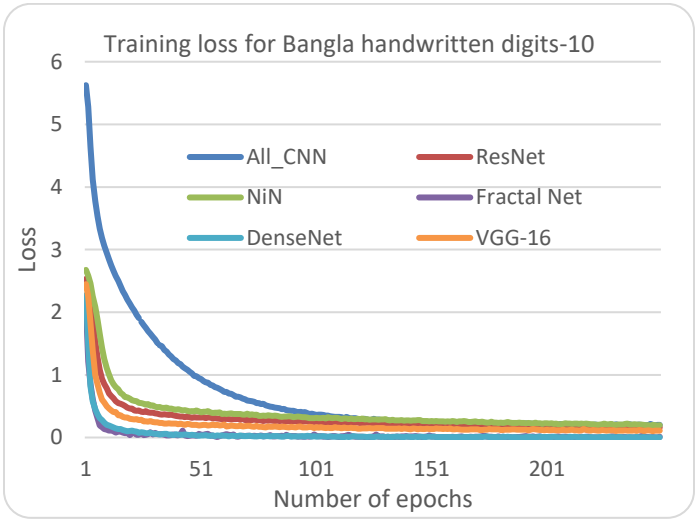


Figure 4.4: Training loss of different architecture for Bangla handwritten 10 digits.

The validation accuracy is shown in Figure 4.5 where DenseNet and FractalNet show better recognition accuracy among all DCNN models.

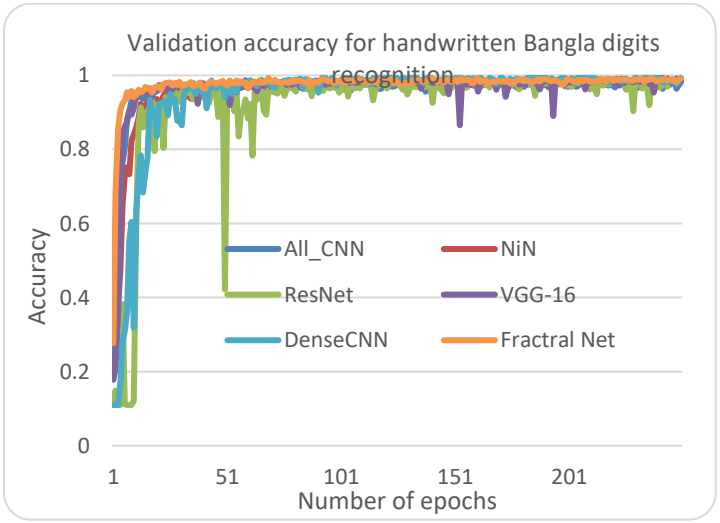


Figure 4.5: Validation accuracy of different architectures for Bangla handwritten 10 digits.

The testing accuracy of all the DCNN models for digit recognition is shown in Figure 4.6. From the result, it can be clearly seen that DenseNet provides the best recognition accuracy compared to other networks.

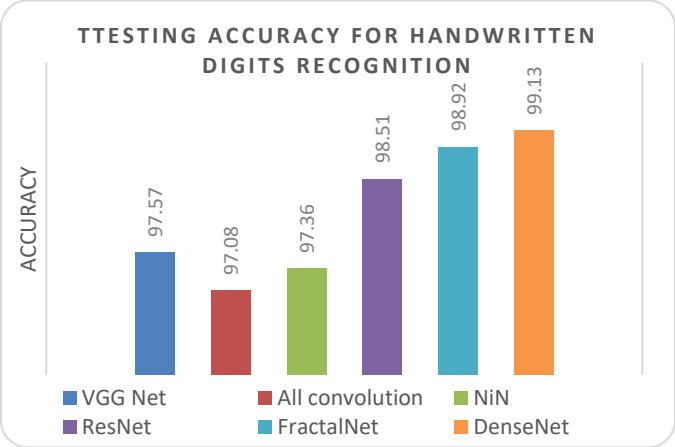
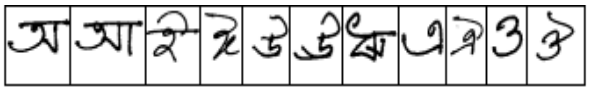


Figure 4.6: Testing accuracy for Bangla handwritten digits recognition.

(a)



(b)

Figure 4.7: Example images of handwritten characters: (a) Bangla consonants Characters and (b) vowels.

4.4.2 Bangla handwritten 50-alphabet

In our implementation, the basic fifty alphabets including 11 vowels and 39 consonants are considered. The samples of 39-consonant and 11-vowel characters are shown in Figure 4.7(a) and (b) respectively. This dataset contains 15,000 samples where 12,000 are used for training and the remaining 3000 samples are used for testing. The dataset contains samples with a different dimension, we rescale all input images to 32×32 pixels. The randomly selected samples from this database are shown in Figure 4.8.

Figure 4.8: Randomly selected handwritten characters of Bangla Alphabets from the dataset.

The training loss for different DCNN models is shown in Figure 4.9. It is cleared that the DenseNet shows the best convergence against other DCNN approaches. Same as the previous experiment the All Conv network shows the worst convergence behavior. In addition, an unexpected convergence behavior is observed in the case of NiN model. However, all DCCN models tend to converge after 200 epochs.

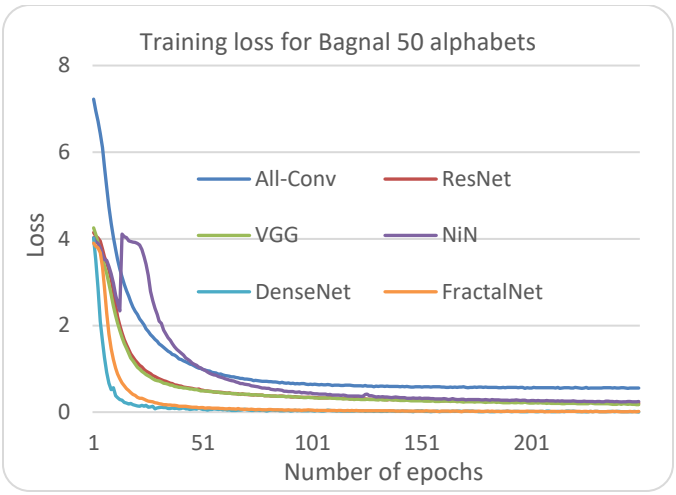


Figure 4.9: Training loss of different DCNN models for Bangla handwritten 50-alphabets.

The validation accuracy on Alphabet-50 is shown in Figure 4.10. DenseNet again shows superior validation accuracy compared to other DCNN approaches.

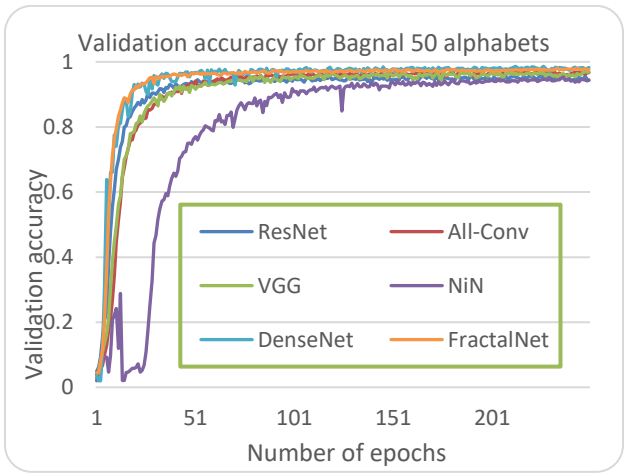


Figure 4.10: The validation accuracy of different architecture for Bangla handwritten 50-alphabet.

The following bar graph in Figure 4.11 shows the testing results on hand-written Alphabet-50. The DenseNet shows the best testing accuracy with a recognition rate of 98.31%. On the other hand, the All Conv Net provides around 94.31% testing accuracy which is the lowest testing accuracy among all the DCNN models.

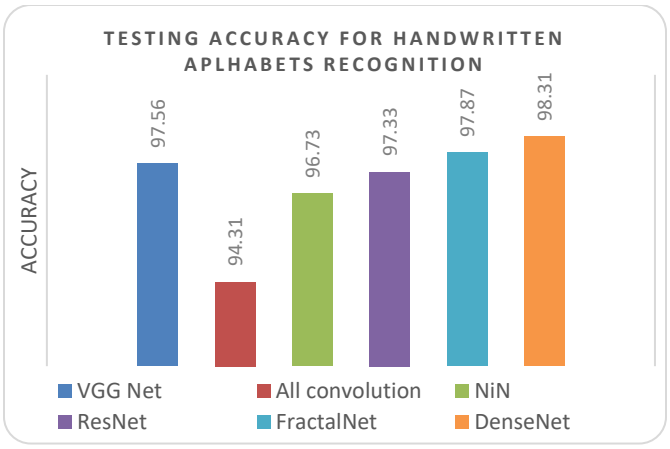


Figure 4.11: Testing accuracy for handwritten 50-alphabets recognition using different DCNN techniques

4.4.3 Bangla handwritten special characters

There are several special characters (SpecialChar-13) which are an equivalent representation of vowels that are combined with consonants for making meaningful words. In our evaluation, we used 11 special characters which are for 11 vowels and two additional special characters. Some samples of Bangla special characters are shown in Figure 4.12. It can be seen from the figure that the quality of the samples is not good, and different variants of the writing of the same symbols make this recognition task even difficult.

Figure 4.12: Randomly selected images of special character from the dataset.

The training loss and validation accuracy for SpecialChar-13 are shown in Figure 4.13 and Figure 4.14 respectively. From these figures, it can be said that the DenseNet provides better performance

with lower loss and with the highest validation accuracy among all DCNN models. Figure 4.15 shows the testing accuracy of DCNN model for the SpecialChar-13 dataset. It is observed from Figure 4.21 that DenseNet show the highest testing accuracy with the lowest training loss and it convergence very fast. On the other hand, VGG-19 network shows promising recognition accuracy as well.

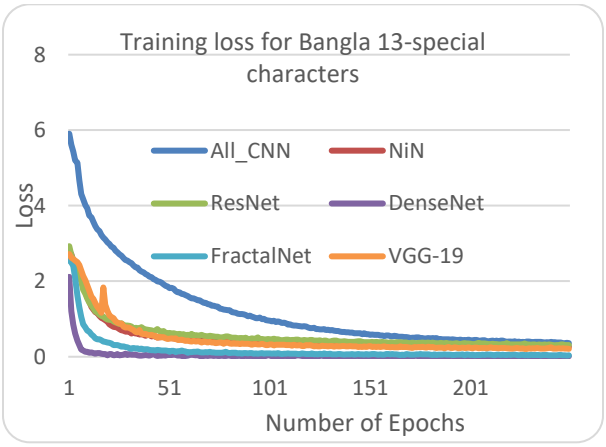


Figure 4.13: Training loss of different architecture for Bangla 13 special characters (SpecialChar-13).

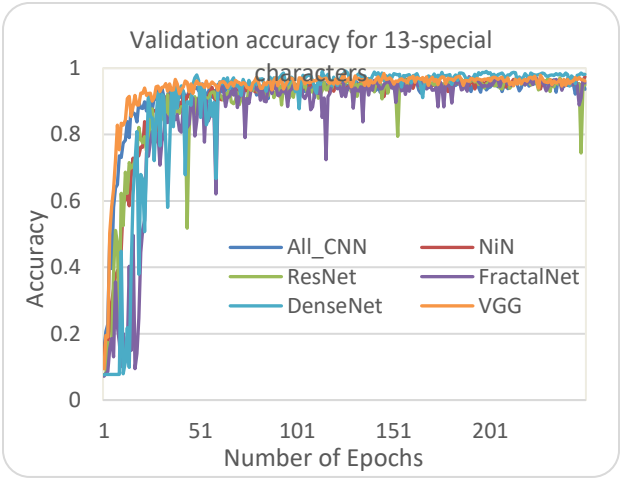


Figure 4.14: Validation accuracy of different architecture for Bangla 13 special characters (SpecialChar-13).

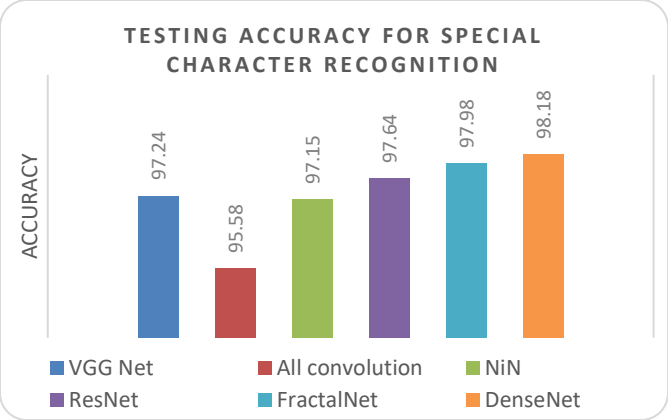


Figure 4.15: Testing the accuracy of different architecture for Bangla 13 special characters (SpecialChar-13).

Table 4.2: The testing accuracy of VGG-16 Network, All Conv. Network, NiN, ResNet, FractalNet, and DenseNet on Digit-10, Alphabet-50, and SpecialChar-13, and comparison against other existing methods.

Types	**Method name**	**Accuracy of Digit-10**	**Accuracy of Alphabet-50**	**Accuracy of SpecialChar-13**
Exiting approaches	Bhowmick et al. [127]	-	84.33 %	-
	Basu et al. [128]	-	80.58 %	-
	Bhattacharya et al. [129]	-	95.84 %	-
	BHCR-CNN [121]	-	85.96 %	-
	MLP (Basu et al. 2005) [119]	96.67 %	-	-
	MPCA + QTLR in 2012 [130]	98.55 %	-	-
	GA (Das et al. 2012B) [120]	97.00 %	-	-
	SRC (Khan et al. in 2014) [131]	94.00 %	-	-
	CNN+DBN [132]	98.78 %	-	-
DCNN	VGG Net [9]	97.57	97.56	96.15
	All convolution [20]	97.08	94.31	95.58
	Network in Network (NiN) [19]	97.36	96.73	97.24
	Residual Network (ResNet) [11]	98.51	97.33	97.64
	FractalNet [23]	98.92	97.87	97.98
	DenseNet [17]	**99.13**	**98.31**	**98.18**

4.5 Performance Evaluation

The testing performance is compared to several existing non-DCNN methods. The results are presented in Table 4.2. The experimental results show that the modern DCNN models including DenseNet [17], FractalNet [26], Residual Network [11] provide better testing accuracy against the

other deep learning approaches and the previously proposed classical methods. The DenseNet provides 99.13% testing accuracy for handwritten digits recognition which is the best accuracy till today. In the case of 50-alphabet recognition, DenseNet yields 98.31% recognition accuracy which is almost 2.5% better than the method in [129]. To best of our knowledge, this is so far, the highest accuracy for 50 handwritten Bangla alphabets recognition. In addition, on 13 special character recognition task, DCNNs show promising recognition accuracy, especially DenseNet achieves the best accuracy which is 98.18%.

4.6 Number of Parameters

For impartial comparison, we have trained and tested the networks with the optimized same number of parameters. Table 4.3 shows the number of parameters used for different networks for 50-alphabet recognition. The number of network parameters for digits and special characters recognition were the same except the number of neurons in the classification layer.

Table 4.3: Number of network parameters.

Models	**Number of parameters**
VGG-16 [9]	~ 8.43 M
All-Conv [20]	~ 2.26 M
NiN [19]	~ 2.81 M
ResNet [11]	~ 5.63 M
FractalNet [26]	~ 7.84 M
DenseNet [17]	~ 4.25 M

Table 4.4: Computational time (in Sec.) per epoch for different DCNNs models on Digit-10, Alphabet-50, and SpecialChar-13 datasets.

Models	**Digit-10**	**Alphabet-50**	**SpecialChar-13**
VGG	32	83	15
All Conv	7	23	4
NiN	9	27	5
ResNet	64	154	34
FractalNet	32	102	18
DenseNet	95	210	58

4.7 Computational Time for Training

We also calculate the computational cost for all methods, although the computation time depends on the complexity of the architecture. Table V represents the computational time per epoch (in second) during training of all the networks for Digit-10, Alphabet-50 and SpecialChar-13 recognition task. From Table 4.4, it can be said that the DenseNet takes the longest time during training due to its dense structure.

4.8 Conclusion

Despite the advances in character recognition technology, Handwritten Bangla Characters Recognition (HBCR) has remained largely unsolved due to the presence of many confusing characters and excessive cursive that lead to low recognition accuracy. On the other hand, deep learning has provided outstanding performance in many recognition tasks of natural language processing. In this research, we investigated handwritten Bangla characters (including digits, alphabets, and special characters) recognition approaches using different deep learning models including Visual Geometry Group (VGG) network, All convolution (All-conv), Network in Network (NiN), Residual Network (ResNet), FractalNet, and Densely Connected Network (DenseNet). The recognition accuracy of DCNN methods was also compared with the existing classical methods for HBCR. It is observed that the DenseNet provides the highest recognition accuracy in all the three experiments for digits, alphabets and special characters recognition. We have achieved a recognition rate of 99.13% for Bangla handwritten digits, 98.31% for handwritten Bangla alphabet and 98.18% special character recognition using DenseNet which is the best recognition accuracy so far. In the future, we would like to evaluate the performance of Inception Recurrent Convolutional Neural Network (IRCNN) for HBCR [194].

CHAPTER 5

IRCNN FOR OBJECT CLASSIFICATION

Deep convolutional neural network (DCNN) is an influential tool for solving various problems in the machine learning and computer vision. Recurrent connectivity is a very important component of the visual information processing within the human brain. The idea of recurrent connectivity is rarely applied within convolutional layers, the exceptions being a couple of DCNN architectures including Recurrent Convolutional Neural Network (RCNN) [27]. On the other hand, the Inception network architecture has become popular among the computer vision community [22,23]. In this paper, we have investigated the impact of a recurrent convolutional layer on modern architectures including Inception, Residual, and Inception-Residual Networks, where utilizes the power of an Inception and Residual network combined with recurrent convolutional layers. Although the inputs are static, the recurrent property plays a huge part in modeling the contextual information for object recognition tasks, and thus improves overall training and testing accuracy. In addition, this proposed architecture generalizes both Inception and RCNN models. We have empirically evaluated the recognition performance of the proposed IRCNN model using different benchmark datasets such as MNIST, CIFAR-10, CIFAR-100, and SVHN. Experimental results show similar or higher recognition accuracy when compared to most of the popular DCNNs including the RCNN. Furthermore, we have investigated IRCNN performance against equivalent Inception Networks and Inception-Residual Networks using the CIFAR-100 dataset. When using the augmented CIFAR-100 dataset, we achieved about 3.5%, 3.47% and 2.54% improvement in classification accuracy compared to the RCNN, equivalent Inception Network, and the Inception-Residual Network respectively. We have also conducted an experiment on TinyImageNet-200

dataset with IRCNN, EIN, EIRN, RCNN, DenseNet, and DenseNet with Recurrent Convolution Layer (RCL) which is called Densely Connected Recurrent, where the proposed model shows significantly better performance against baseline models.

5.1 Introduction

In recent years, deep learning using Convolutional Neural Networks (CNN) has shown enormous success in the field of machine learning and computer vision. CNNs provide state-of-the-art performance in various image recognition tasks including object recognition [9,10], object detection, tracking, and image captioning [1]. This technique has been applied massively in computer vision tasks such as video representation and classification [1]. In addition, deep learning approaches are successfully used in the field of medical imaging, medical information processing, and they have achieved (near) human-level performance with the Inception v3 architecture [23]. The Natural Language Processing (NLP) and Machine Translation (MT) have been used deep learning techniques and achieved great success in this domain [134, 135]. Furthermore, this technique has been used extensively in the field of speech recognition [136]. Moreover, deep learning technique has been applied for intelligent game development successfully [137, 138].

Presently, deep learning-based approaches (DCNN in particular [1]) perform very well in the domains of detection, classification, and scene understanding. The CNN is a very powerful technique used to learn high-level and multi-scale features, which helps to extract robust and discriminative features with global contextual information within a region of an input sample. However, most of the hierarchical feature learning models including CNN [1,10], Neocognitron [17], and HMAX [139] are proposed using a feed-forward architecture.

The visual cortex in the human brain consists of several visual processing units and processes information using feed-forward and feedback (recurrent connectivity) techniques [140]. The feed-forward technique implements concurrent probabilistic inference in the visual hierarchy, whereas the feedback technique serves to integrate contextual prior or extra-classical receptive field effects [140]. The model of the Convolutional Deep Belief Network (CDBN) adopted this strategy, using

feedback connections for propagation. This achieved very good accuracy for object classification tasks [141]. An alternative study shows that the interaction of the early visual (V1) with various higher-order visual processing units through recurrent connections are responsible for the re-integration of information analyzed by the higher visual areas. This study also shows V1 could be used to integrate and coordinate the computation of identity (WHAT) and object location (WHERE) in the visual scene with recurrent interaction [142]. Thus, the recurrent connectivity of synapses in the human brain plays a big role in context modeling for visual recognition tasks [143, 144]. The importance of context modulation for visual recognition tasks is demonstrated in different studies [52,143].

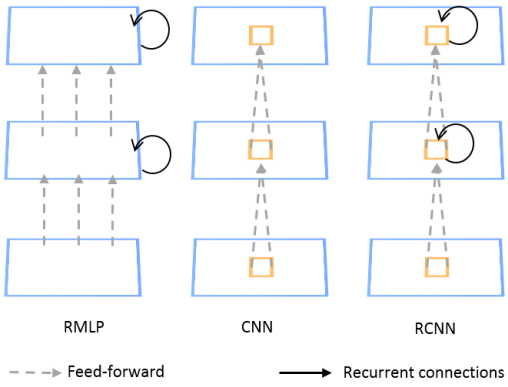


Figure 5.1: The architecture layout for the RMLP, the CNN, and the RCNN [28].

However, several models are proposed based on the concept of a recurrent layer in an artificial neural network. The architecture of the general Recurrent Multilayer Perceptron (RMLP) is used very often in the field of dynamic control [145]. The RMLP is simply the extension of a Multilayer Perceptron (MLP) with recurrent connectivity in the layers [146]. Even if the input is static, the object recognition task is a dynamic process because of the presence of recurrent or top-down connections. The concept of the RMLP has been extended to include convolutional layers, resulted in the development of the RCNN [52]. The diagrams of the RMLP (left), the CNN (middle) and the RCNN (right) are shown in Figure 5.1.

In this work, we have generalized both architectures of the Inception network [23] and the RCNN [52]. Where recurrent convolutional layers are incorporated within the inception block, the convolution operations are performed considering different time steps [141]. The proposed inception block is shown in Figure 5.2. The intention of the DCNN architecture of the Inception [23,24] and Residual networks [11,21] is to implement large-scale deep networks. As the model becomes larger and deeper, the computational parameters of the architecture are increased dramatically. Thus, the model becomes more complex to train and thus, more computationally expensive. In the scenario, the recurrent property ensures better training and testing accuracy with less or equal computational parameters.

Others research groups are trying to implement bigger and deeper DCNN architectures like Google Net [10], or a Residual network with 1001 layers [21] to achieve even better recognition accuracy. Alternatively, we are presenting an improved version of the DCNN model inspired by the information processing mechanism of the human visual cortex and the recently developed DCNN architectures such as Inception-v4 [24] and RCNN [52]. Therefore, we call this model the **Inception Recurrent Convolutional Neural Network (IRCNN)**. This model not only ensures better recognition accuracy with the same computational parameters against the state-of-the-art DCNN architectures but also helps to improve the overall training process of the deep learning approach. The contributions of this work are as follows:

- We combine two popular models of Inception and RCNN is proposed.
- Experimental evaluation of the proposed learning model's performance against different DCNN architectures on different benchmark datasets such as MNIST, CIFAR-10, CIFAR-100, and SVHN.
- An empirical investigation of the impact of the recurrent layer in the Inception Network.
- An empirical investigation of the impact of RCLs on Densely Connected Neural Networks namely DenseNet.

5.2 Related Work

5.2.1. Convolutional Neural Networks (CNNs)

The first self-organizing neural network model called the Neocognitron was proposed by Fukushima in 1982 [17]. Although, the deep learning revolution began in 1998 with LeNet [18]. From then on, several different architectures have been proposed that have shown massive success using many different benchmark datasets including MNIST, SVHN, CIFAR-10, CIFAR-100, ImageNet, and many more. Of the DCNN architectures, AlexNet [7], VGG [9], NiN [19], the All Convolutional Network [20], GoogLeNet [10], Inception-v4 [24], and Residual Networks [11,21] can be considered the most popular architectures due to their improved performance on different benchmarks for object classification. In 2012, Alex Krizhevesky et. al. proposed an improved deeper version of a CNN compared to LeNet [18], and won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012. This was a significant breakthrough in the field of machine learning and computer vision, as this was the first time a deep network outperformed the alternative approaches for visual recognition tasks [1]. GoogLeNet, or Inception-v1 [24], and the Residual network [11,21] won ILSVRC in 2014 and 2015 respectively.

Inception architecture has become very popular in the deep learning and computer vision community, and it has been refined in different ways. An Inception network with batch normalization [22] (Inception-v2) was proposed by Ioffe et al. The Inception network (Inception-v3) was proposed with factorization ideas in [35]. In most cases, the improvement of deep learning approaches has been due to the development of the following components: Initialization techniques of DCNNs [69,70], new deep network architectures [26,27], optimization of deep network structures (depending upon computational parameters) [24], deeper and wider deep networks [25], activation functions for deep learning approaches [77], and optimization methods for training DCNNs [87,88]. Some researchers have been focused on design alternatives that produce the same

level of recognition accuracy as state-of-the-art architectures (like Inception-V4 with Residual Net [24]) with fewer computational parameters [1]. In this work, we have emphasized the development of an alternative DCNN architecture called the IRCNN.

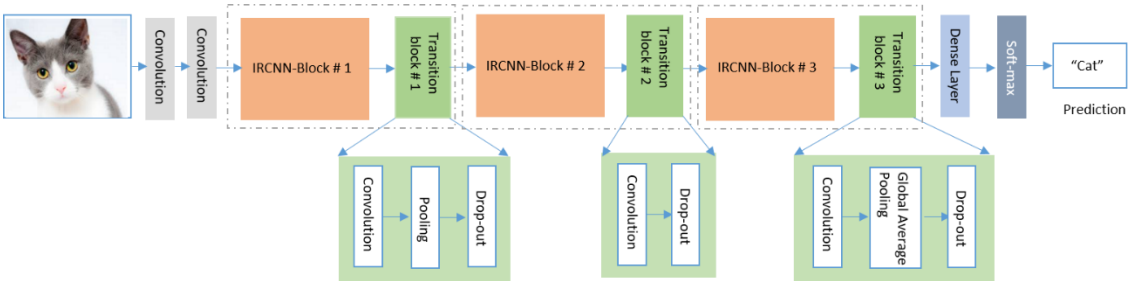


Figure 5.2: The overall operational flow diagram of the proposed Inception network with recurrent convolutional layers: consists of IRCNN-block, transition block and Softmax layer at the end.

5.2.2. Recurrent Neural Networks (RNNs)

Presently, most researches have been focused on improving recognition accuracy with new DCNN models. Very little research has been conducted on recurrent architectures within the layers of a CNN. Recurrent connectivity is inspired by the human visual cortex, and its importance is demonstrated for object recognition tasks using 100 different object categories containing naturally occurring variations such as location, rotation, size, and lighting [153]. In addition, in a real-world scenario, occlusion and low contrast have a big impact on visual recognition tasks. The recurrent network promotes robust object recognition in this particular situation [154,155].

As far as recurrent connectivity in DCNNs is concerned, the relationship between the Residual network (ResNet) [11,155], RNNs, and the visual cortex show that a shallow RNN with weight sharing among the layers is exactly equivalent to a very deep ResNet. The study shows that the RNNs provide better accuracy than ResNet while having an order of magnitude fewer parameters [21]. In 2015, Ming et al proposed the first RCNN structure tested using object recognition tasks. The architecture consists of several blocks of recurrent convolutional layers followed by a max-

pooling layer. In the second to last layer of the structure, global max-pooling is used followed by a soft-max layer at the end. In 2015, this architecture reported state-of-the-art accuracy for object classification on different benchmarks [52]. Another RCNN based approach was proposed for scene labeling for large input context modeling with limited capacity networks, and it achieved state-of-the-art performance on different scene understanding datasets [156]. The Long-term Recurrent Convolutional Network (LRCN) was proposed for visual recognition and description by Donahue et al. [157]. This architecture uses a combination of two popular techniques, CNN and LSTM. The features are extracted through the CNN, and LSTM is applied to identify how features vary with respect to time. This model shows outstanding performance for visual description [157].

5.2.3. CNN and RNN for object recognition

From the above discussion, it can be concluded that DCNNs with improved architectures show enormous achievement when performing visual recognition tasks. The following section demonstrates the theoretical details of proposed deep IRCNN learning architecture.

5.3. Inception Recurrent Convolutional Neural Networks (IRCNNs) Layer

The proposed architecture (IRCNN) is based on several recently developed deep learning architectures, including Inception Nets [23] and RCNNs [52]. It tries to reduce the number of computational parameters while providing better recognition accuracy. As shown in Figure 5.2, the IRCNN architecture consists of general convolution layers, IRCNN blocks, transition blocks, and a softmax layer at the end. One of the most novel features of this work is the introduction of recurrence into the Inception module, as shown in the IRCNN block in Figure 5.3. The key feature of Inception-v4 is that it concatenates the outputs of multiple differently sized convolutional kernels in the inception block [23]. Inception-v4 is a simplified version of Inception-v3, using lower rank filters and pooling layers. Inception-v4, however, combines Residual concepts with Inception networks to improve the overall accuracy over Inception-v3. The outputs of inception layers are

added with the inputs to the Inception-Residual module. In this work, we utilize the inception concepts from Inception-v4 [24].

5.3.1. IRCNN block

The IRCNN block performs recurrent convolution operations with different sized kernels (see Figure 5.3). In the recurrent structure, the inputs to the next time step are the sum of the convolutional outputs of the present time step and previous time steps. The same operations are repeated based on the number of time steps considered. As the input and output dimensions do not change, this is simply an accumulation of feature maps with respect to the time step considered. This helps to strengthen the extraction of the target features. As shown in Figure 5.3, one of the paths of inception block contains an average pooling operation is applied before the recurrent convolution layer. In this particular pooling layer, a 3×3 average pooling with stride 1×1 is applied by keeping the border size same, results in output samples with the same dimensions as the inputs. The overlapping average pooling technique helps in the regularization of the network [1].

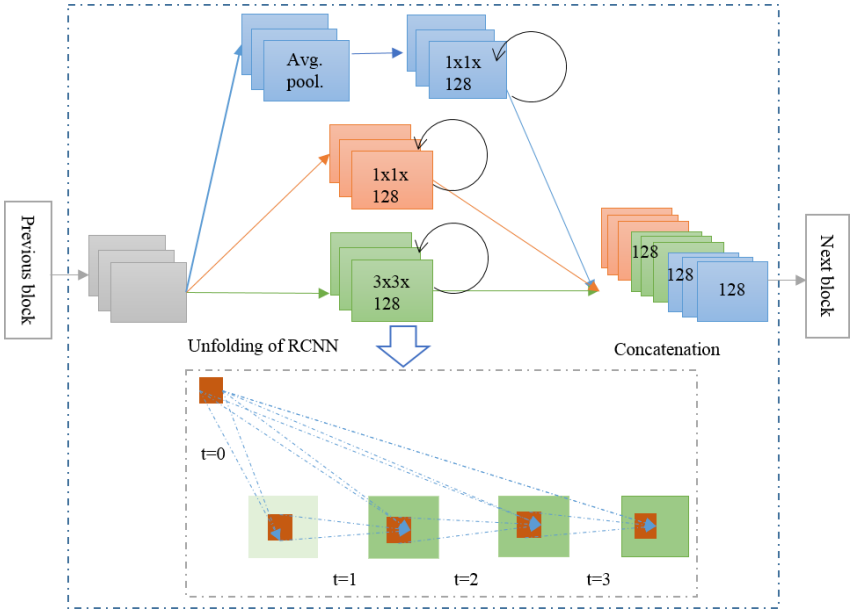


Figure 5.3: Inception-Recurrent Convolutional Neural Network (IRCNN) block with different convolutional layers respect to the different size of kernels

The operations of each Recurrent Convolution Layer (RCL) in the IRCNN block are similar to the operations in [28]. To describe these operations, consider a pixel collated at (i, j) of a particular patch from an input sample on the k^{th} feature map in the RCL. This is the output $y_{ijk}(t)$ at time step t. The output can be expressed as:

$$y_{ijk}(t) = \left(w_k^f\right)^T x_f^{(i,j)}(t) + (w_k^r)^T x_r^{(i,j)}(t-1) + b_k \tag{5.1}$$

Here $x_f^{(i,j)}(t)$ and $x_r^{(i,j)}(t-1)$ are the inputs for a standard convolutional layer and an RCL respectively. w_k^f and w_k^r are the weights for the standard convolutional layer and the RCL respectively, while b_k is the bias. The final output for the layer at time step t is:

$$z_{ijk}(t) = f(y_{ijk}(t)) = \max(0, y_{ijk}(t)) \tag{5.2}$$

where f is the standard Rectified Linear Unit (ReLU) activation function. The Local Response Normalization (LRN) function is applied to the outputs of each kernel in the IRCNN block [7]:

$$y = \text{norm}(z_{ijk}) \tag{5.3}$$

The outputs of the IRCNN block with respect to the different kernel sizes and average pooling operations are defined as $y_{1\times1}(x)$,$y_{3\times3}(x)$, and $y_{1\times1}^p(x)$. The final output (y_{out}) of the IRCNN-block can be described as:

$$y_{out} = y_{1\times1}(x) \oplus y_{3\times3}(x) \oplus y_{1\times1}^p(x) \tag{5.4}$$

where $\oplus$ represents the concatenation operation with respect to the channel axis of the output samples. In this implementation, we have used $t = 3$, that indicates the four-recurrent convolutional operations have been performed in each IRCNN-block (individual path) which is clearly shown in Figure 5.3. The outputs of the IRCNN-block become the inputs that are fed into the transition layer.

5.3.2. Transition block

In the transition block, three operations (Convolution, Pooling, and Dropout) are performed depending upon the placement of the block in the network. According to Figure 5.2, we have applied all of the operations in the very first transition block; whereas in the second transition block,

we have only used convolution with dropout operations. The third transition block consists of convolution, global-average pooling, and Dropout layers. The global-average pooling layer is used as an alternative to fully connected layers. There are several advantages of a global-average pooling layer. Firstly, it is very close in operation to convolution, hence enforcing correspondence between feature maps and categories. The feature maps can be easily interpreted as class confidence. Secondly, it does not need computational parameters, thus helping avoid overfitting of the network. The softmax layer is used at the end of the IRCNN architecture. Late use of the pooling layer is advantageous because it increases the number of non-linear hidden layers in the network. Therefore, we have applied only two special pooling layers in this architecture. The max-pooling layers perform operations with (3×3) patch with (2×2) stride over the input samples. Since the non-overlapping max-pooling operation has a negative impact on model regularization, we used overlapped max-pooling for regularizing the network. This is very important for training a deep network architecture [24]. Special pooling is carried out with the max-pooling layer in the middle of the network (not all transition-blocks have pooling layers). Eventually, a global average pooling layer is used at the very end before a softmax logistic regression layer.

5.3.3. Optimization of network parameters

To keep the number of computational parameters lower compared to other traditional DCNN approaches like AlexNet [7] and VGGNet[9], we have used only 1×1 and 3×3 convolutional filters in this implementation (inspired by the NiN [19] and Squeeze Net [148] models). There are significant benefits to using smaller sized kernels, which help incorporate more non-linearity in the network. For example: we can use a stack of two 3×3 respective fields (without placing any pooling layer in between) as a replacement for one 5×5; and a stack of three 3×3 respective fields instead of a 7×7 [24]. The benefit of adding a 1×1 filter is that it helps to increase the non-linearity of the decision function without having any impact on the convolution layer. Since the size of the input and output features do not change in the IRCNN blocks, it is just a linear projection on the same

dimension with non-linearity added using a ReLU. We have used a dropout of 0.5 after each convolutional layer in the IRCNN-block.

Finally, we have used a Softmax or a normalized exponential function [158] layer at the end of the architecture. For input sample x and weight vector W, with K distinct linear functions, the softmax operation can be defined for the i^{th}class as follows:

$$P(y = i|x) = \frac{e^{x^T w_i}}{\sum_{k=1}^{K} e^{x^T w_k}} \tag{5.5}$$

5.4 Densely Connected Recurrent Convolutional Network (DCRN)

According to the basic structure of Densely Connected Networks (DCN), the outputs from the prior layers are used as input for the subsequent layers. This architecture ensures the reuse the features inside the model, therefore it provides better performance on different computer vision tasks which in empirically investigated on different datasets in [27]. However, in this implementation, we have proposed an improved version of DCN which is named DCRCN in short University of Dayton Network (UD-Net) which is used for nuclei classification. The UD-Net is the building block of several Recurrent Connected Convolutional (DCRC) blocks and transition blocks. The pictorial representation of Densely Connected Recurrent Convolutional (DCRC) block is shown in Figure 5.4.

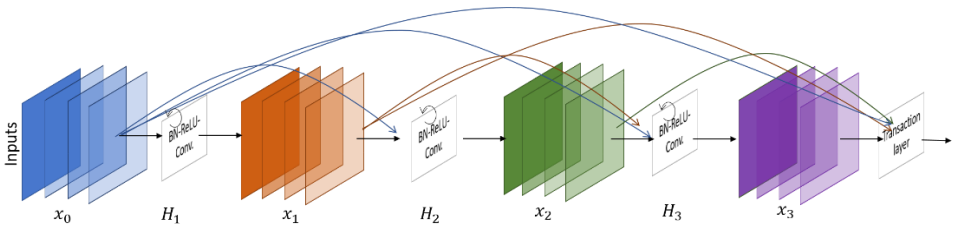


Figure 5.4: Densely Connected Recurrent Convolutional (DCRC) block.

According to the basic mathematical model of DenseNet which has explained in [52], the l^{th} layer receive all the feature maps $(x_0, x_1, x_2 \cdots x_{l-1})$ from the previous layers as input:

$$x_l = H_l([x_0, x_1, x_2 \cdots x_{l-1}]) \tag{5.6}$$

where $[x_0, x_1, x_2 \cdots x_{l-1}]$ are the concatenated features from $0, \cdots \cdots, l-1$ layers and $H_l(\cdot)$ is a single tensor. Let's consider the $H_l(\cdot)$ input sample from l^{th} DCRN block and contains $0, \cdots \cdots, F-1$ feature maps which are feed in the recurrent convolutional layers according to the method has proposed in [52]. This convolutional layer performs three consecutive operations which include Batch Normalization (BN), followed by ReLU and a 3×3 convolution (conv). Let's consider a center pixel of a patch located at (i, j) in an input sample on the k^{th} feature of $H_{(l,k)}(\cdot)$. Additionally, let's assume the output of the network is $H_{lk}(t)$ at the time step t. The output can be expressed as follows:

$$H_{lk}(t) = \left(w^{f}_{(l,k)}\right)^{T} * H^{f(i,j)}_{(l,k)}(t) + \left(w^{r}_{(l,k)}\right)^{T} * H^{r(i,j)}_{(l,k)}(t-1) + b_{(l,k)} \tag{5.7}$$

Here $H^{f(i,j)}_{(l,k)}(t)$ and $H^{r(i,j)}_{(l,k)}(t-1)$ are the inputs to the standard convolution layers and the l^{th} recurrent convolution layers respectively. The $w^{f}_{(l,k)}$ and $w^{r}_{(l,k)}$ values are the weights of the standard convolutional layer and the recurrent convolutional layers of the k^{th} feature map respectively, and $b_{(l,k)}$ is the bias. The recurrent convolution operations are performing with respect to t [149]. The pictorial represents of convolutional operation for $t = 2$ is shown in Figure 5.5.

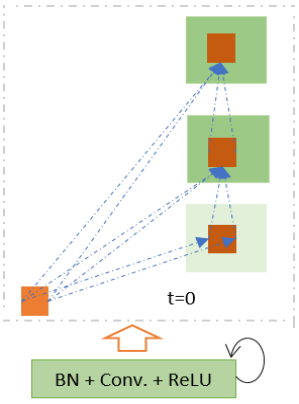


Figure 5.5: Unfolded recurrent convolutional units for t = 2.

In the transition block, 1×1 convolutional operations are performed with BN followed by 2×2 average pooling layer. The DenseNet model consists of several dense blocks with feedforward convolutional layers and transition blocks whereas the DCRN uses the same number of dense blocks with recurrent convolutional units and transition blocks. For both models, we have used 4 blocks, 3 layers per block, and the growth rate is 5.

5.5 Experiments

We have evaluated the proposed IRCNN method with a set of experiments on different benchmark datasets: MNIST [159], Cifar-10[160.], Cifar-100[160], SVHN [161], and TinyImageNet-200[173] and compared against different models. The entire experiment has been conducted on Linux environment with Keras [171] and Theano [172] in the Backend running on the single GPU machine with NVIDIA GEFORCE GTX-980 Ti.

5.5.1 Training methodology

In the first experiment, we have trained the proposed IRCNN technique using the stochastic gradient descent (SGD) technique with the default initialization technique for deep networks found in Keras [171]. We set the Nesterov momentum to 0.9 [162] and decay to 9.99e-07 9.99×10^{-7}. Second, we experimented with our proposed approach with the Layer-sequential unit-variance (LSUV) technique, which is a simple method for the initialization of weights in a deep neural network [70]. We have also used a very recently proposed an improved version of the optimization function based on "Adam" that is called EVE [151]. The following parameters are used for the EVE optimization function: the value of the learning rate (λ) is 1e-4, decay (γ) is 1e-4, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\beta_3 = 0.999$b, κ=0.1, K=10, and $\in = 1e - 08$. The β_1, $\beta_2 \in [0,1)$ are exponential decay rates for moment estimation in Adam. The $\beta_3 \in [0,1)$ is an exponential decay rate for computing relative changes. The κ, and K values are lower and upper thresholds for relative change

and $\in$ is a fuzzy factor [151]. It is noted that each convolutional layer in the IRCNN-block is the $l2-\text{norm}$ for weight regularization with 0.002. In both experiments, we have used ReLU activation functions. We have generalized the network with dropout (0.5). The only horizontal flipping technique is applied in data augmentation. We train the models for 350 epochs with a 128 batch size for CIFAR-10 and 100. During the training of MNIST and SVHN, we use 200 epochs with a mini-batch size of 128. For the impartial comparison, we have trained and tested against equivalent Inception networks and Inception residual networks. By equivalent, we mean having the same number of layers and computational parameters. We describe these networks as the Equivalent Inception Network (EIN) and the Equivalent Inception Residual Network (EIRN).

5.5.2 Results

5.5.2.1 MNIST

One of the most popular dataset for handwritten digits from 0-9 [159], the dataset contains 28x28 pixels grayscale images with 60,000 training and 10,000 testing examples. For this experiment, we trained our proposed model with two IRCNN-block of convolution with ReLU activation function. The model has been trained with 60,000 samples and 10,000 samples for used for validation of the model. Eventually, the trained network was tested with 10,000 testing examples. We obtained a test error of 0.32% with the IRCNN and SGD and achieved around 0.29% error for the IRCNN when initialization with LSUV [70] and the EVE [151] optimization function. This provided the best accuracy compared to the RCNN, as well as the other state-of-the-art networks. The summary of the classification accuracies is given in Table 5.1. No data augmentation techniques have been applied in this experiment on MNIST. On the contrary, global contract normalization and ZCA whitening are applied in the experiments using most of the mentioned models.

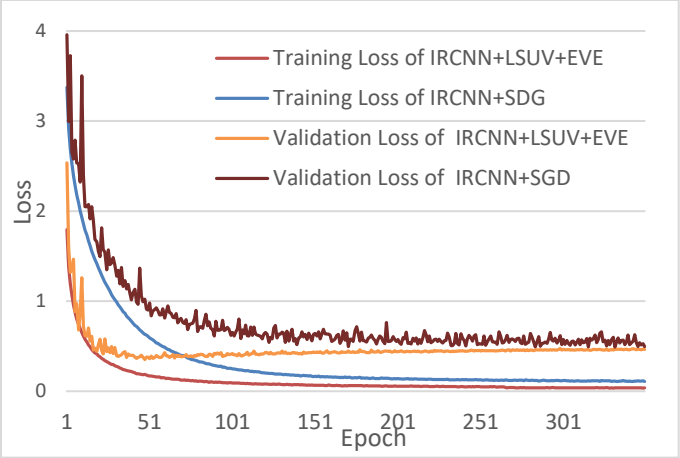


Figure 5.6: Training and validation loss for IRCNN with SGD and LSUV+EVE on CIFAR-10.

5.5.2.2 CIFAR-10

CIFAR-10 is an object classification benchmark [160] consisting of 32x32 color images that represent 10 classes. It is split into 50,000 samples for training and 10,000 samples for testing. The experiment was conducted with and without data augmentation. The entire experiment was conducted using models similar to the one shown in Figure 5.2. Using the proposed approach, we achieved around 8.41% error without data augmentation and 7.37% error with data augmentation using the SGD technique. These results are better than most of the DCNN models summarized in Table 5.1.

Better performance is observed when using the IRCNN with LSUV [70] initialization approach and EVE [151] as the optimization technique. The results show about 8.17% and 7.11% error without and with data augmentation respectively. When comparing these results to those of the different models in Table 5.1, it can be observed that our proposed approach provides better accuracy compared to various advanced and hybrid models. The training and validation loss when using CIFAR-10 and the proposed model is shown in Figure 5.6. Figure 5.7 shows the training and validation accuracy of the IRCNN with SGD and LSUV+EVE.

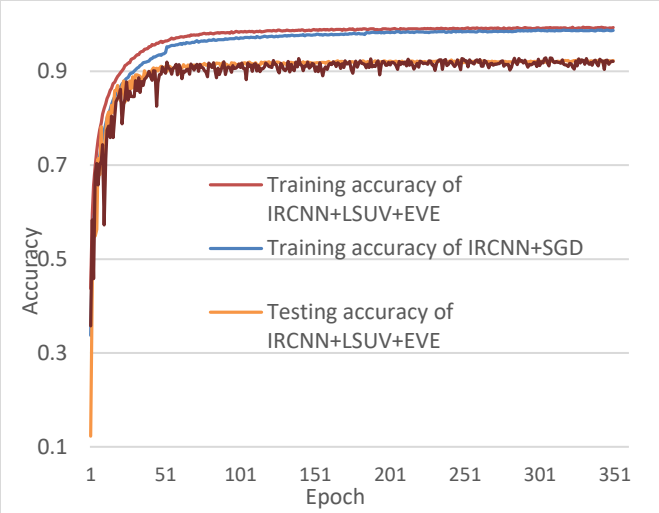


Figure 5.7: Training and validation accuracy for IRCNN with SGD and LSUV+EVE on CIFAR-10.

Table 5.1: Testing errors (%) of IRCNN on MNIST, CIFAR-10(C-10), CIFAR-100(C-100), and SVHN. Here "+" indicates standard data augmentation using random horizontal flipping. IRCNN achieves lower testing in most of the cases indicates with bold.

Methods	**MNIST**	**C10**	**C10$^+$**	**C100**	**C100$^+$**	**SVHN$^+$**
Maxout [163]	0.45	11.6	9.38	-	38.57	2.47
NiN [19]	0.47	10.41	8.81	35.68	-	2.35
DSN [166]	0.39	9.69	7.97	-	34.57	1.93
Prob maxout [168]	-	9.39	-	-	38.14	2.39
ALL-CNN [20]	-	9.08	7.25	-	33.71	-
Highway Network [167]	-	-	7.72	-	32.24	-
RCNN [52]	0.31	8.69	**7.09**	-	31.75	1.77
dasNet[165]	-	-	9.22	-	33.78	-
FitNet [169]	-	-	8.39	-	35.04	-
DropConnect (5 Nets) [164]	1.12	-	9.41	-	-	1.94
CNN+Tree [170]	-	-	-	-	36.85	-
IRCNN +SDG	0.32	8.41	7.37	34.13	31.22	1.89
IRCNN + LSUV + EVE	**0.29**	**8.17**	7.11	**30.87**	**28.24**	**1.74**

5.5.2.3 CIFAR-100

An alternative dataset developed by the same research group (K and Hinton, 2009) [160] when used in this experiment. The dataset contains 60,000 (50,000 for training and for 10,000 testing) color 32x32 images, and it has 100 classes. We used the SGD and LSUV [70] initialization

approach with the EVE optimization technique [151] in this experiment. The experimental results are shown in Table 1. In both cases, the proposed technique shows state-of-the-art accuracy compared with different DCNN models. IRCNN+SGD shows about 34.13% testing error without data augmentation and 31.22% classification error with data augmentation. In addition, this model achieved about 30.87% and only 28.24% error in the second experiment that uses the LSUV initialization approach and EVE. This is the highest accuracy achieved in any of the deep learning models summarized in Table 5.1. For augmented datasets, we have achieved 71.76% recognition accuracy with LSUV+EVE, which is about a 3.5% improvement compared to RCNN [28].

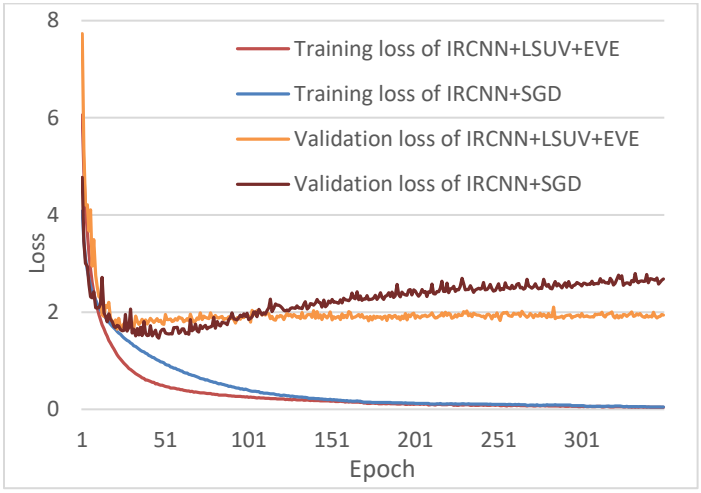


Figure 5.8: Training and validation loss for IRCNN with SGD and LSUV+EVE on CIFAR-100.

Figure 5.8 shows the training and validation loss of the IRCNN for both experiments using the CIFAR-100 dataset with data augmentation (with and without initialization and optimization). It is clearly shown that the proposed model has a lower error in both experiments, showing the effectiveness of the proposed IRCNN learning model. The training and testing accuracy of the IRCNN with LSUV and EVE are shown in Figure 5.9.

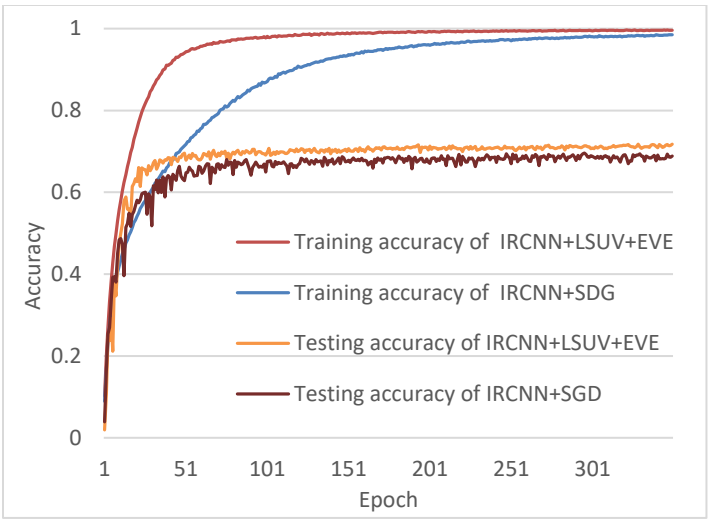


Figure 5.9**:** Training and validation accuracy for IRCNN with SGD and LSUV+EVE on CIFAR-100.

5.5.2.4. Street View House Numbers (SVHN)

SVHN (Netzer et al. 2011) is one of the most challenging datasets for street view house number recognition [161]. This dataset contains color images representing house numbers from Google Street View. Two versions of this dataset are available. In this experiment, we have considered the second version, which consists of 32×32 color examples. There are 73,257 samples in the training set and 26,032 samples in the testing set. In addition, this dataset has 531,131 extra samples that are used for training purposes. As single input samples of this dataset contain multiple digits, the main goal is to classify the central digit. Due to the huge variety of color and brightness, this dataset is much more difficult to classify compared to the MNIST dataset. In this case, we have experimented with the same model as was used for CIFAR-10 and CIFAR-100. We used the same preprocessing steps that were used in the RCNN in [52]. The experimental results show better recognition accuracy in both cases, as shown in Table 5.1. We have obtained about 1.89% testing error with the IRCNN+SGD and 1.73% error with the IRCNN+LSUV+EVE respectively. It is noted that Local Contract Normalization (LCN) is applied during experiments such as MaxOut

[163], NiN [19], DSN [166], and Drop Connect [164]. The drop-connection results are based on the average performance of five networks [163].

5.5.3. Impact of recurrent layers

The proposed architecture also performs well when compared to traditional architectures. One LSUV with a traditional DCNN architecture is called FitNet4, and it only achieved 70.04% classification accuracy with data augmentation using mirroring and random shifts for CIFAR-100 [70]. On the other hand, we have only applied random horizontal flipping for data augmentation in this implementation and achieved about 1.72% better recognition accuracy against FitNet4 [168].

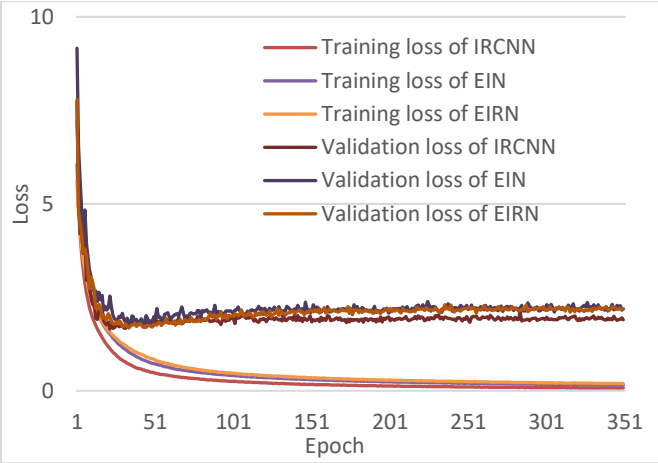


Figure 5.10. Training and validation loss for IRCNN, EIN, and EIRN on CIFAR-100.

For an impartial comparison with the EIN and EIRN models, we have implemented the Inception network with the same number of layers and parameters as in the transition and Inception-block. Instead of using recurrent connectivity in the convolutional layers, we used sequential convolutional layers for the same time-step with the same kernels. During the implementation of EIRN, we only added a residual connection in the Inception-Residual block, where the inputs of the Inception-Residual block are accumulated with the outputs of that particular block.

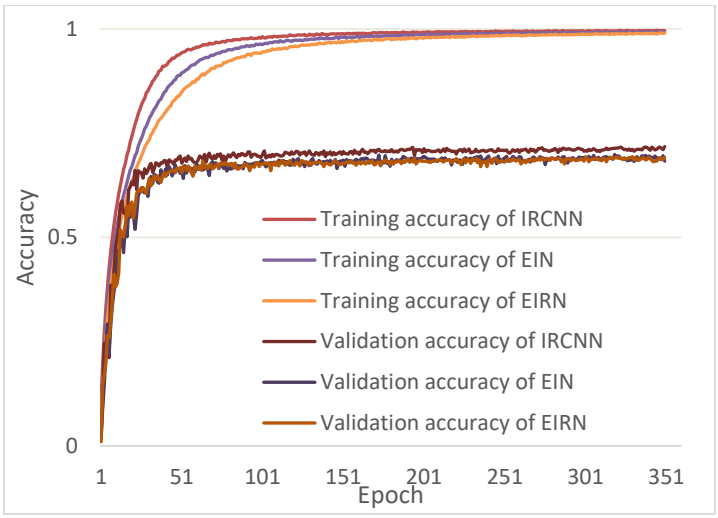


Figure 5.11: Training and validation accuracy for IRCNN, EIN, and EIRN on CIFAR-100.

In this case, all of the experiments have been conducted using the augmented CIFAR-100 dataset [160]. The model loss and accuracy for both training and validation phases are shown in Figures 5.10 and 5.11 respectively. From both figures, it can be observed that the proposed model shows a lower loss and the highest recognition accuracy compared with EIN and EIRN, proving the effectiveness of the proposed models. It also demonstrates the advantage of recurrent layers in Inception networks. The testing accuracy of IRCNN, EIN, and EIRN are shown in Figure 5.12. It can be summarized that our proposed IRCNN shows around 3.47% and 2.54% better testing accuracy compared to EIN and EIRN respectively.

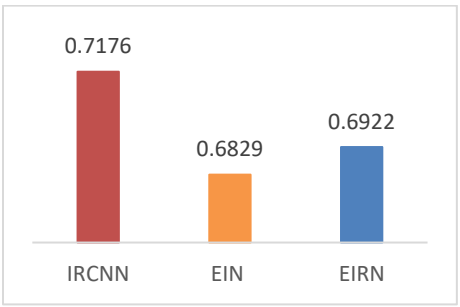


Figure 5.12: Testing accuracy of IRCNN model against EIN and EIRN on an augmented dataset of CIFAR-100.

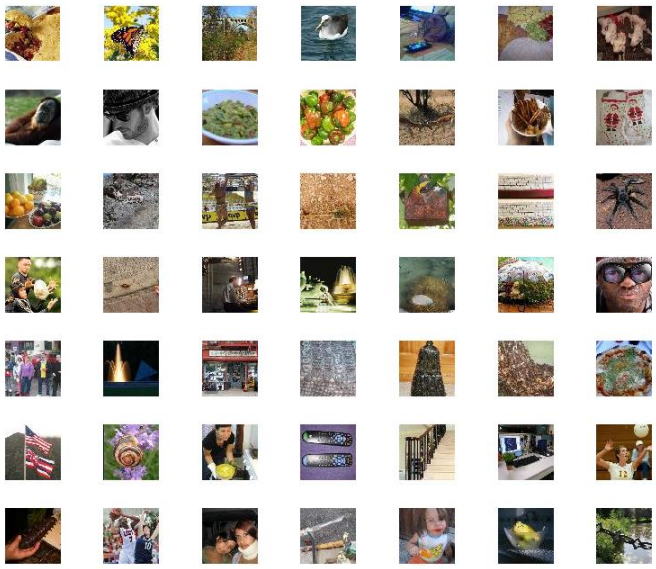


Figure 5.13: Some example images from TinyImageNet-200dataset.

5.5.4 TinyImageNet-200 dataset

In this experiment, we have evaluated the proposed technique on the TinyImageNet-200 dataset. This dataset contains 100,000 samples for training, 10,000 samples for validation, and 10,000 samples for testing https://www.kaggle.com/c/tiny imagenet (2017)[173]. These images are sourced from 200 different classes of objects. The key different between the main ImageNet dataset and TinyImageNet is that the images are down sampled from 224×224 to 64×64. The main impact of down sampling is a loss of detail. Therefore, down sampling, the images might lead the ambiguity problem which may have an effect on overall model accuracy. The original ImageNet image sizeis482×418pixels, where the average object scale is 17%. The size of the images in this experiment is 64 × 64, which makes the TinyImageNet problem even harder. Some of the example images are shown in Figure 5.13. We have experimented with IRCNN, EIN, EIRN and RCNN models with almost the same number of parameters shown in Table 5.2. The SGD with a starting learning rate of 0.001, a batch size 64. The training and validation accuracy are shown in Figures 5.14 and 5.15.

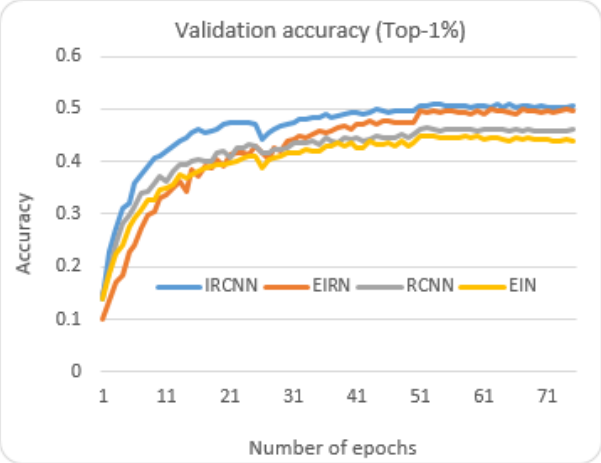


Figure 5.14: The validation accuracy for IRCNN, EIN, EIRN, and RCNN on the Tiny-ImageNet-200 dataset.

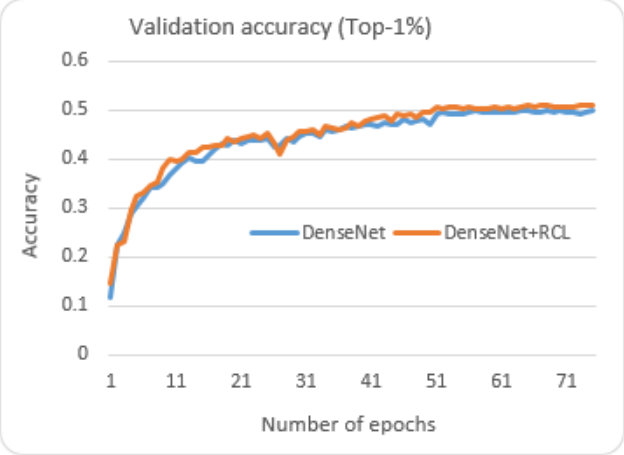


Figure 5.15: The validation accuracy of DenseNet and DenseNet with a Recurrent Convolutional Layer(RCL).

5.5.5 Evaluation

From the above empirical evaluations, it can be concluded that the proposed IRCNN technique provides better recognition accuracy compared to different deep learning models in most cases, demonstrating the precision of the proposed deep learning model. This model also shows better recognition performance with the same number of computational parameters (~3.12M) when compared to the EIN and EIRN models. Furthermore, if we observed the figures for model loss and accuracy, it can be clearly seen that the proposed model demonstrates less loss with better

recognition accuracy. We have also empirically evaluated the rate of convergence in our proposed IRCNN algorithm compared with traditional EIN and EIRN models. The proposed model converged earlier with much lower model loss compared to EIN and EIRN. We have experimented with IRCNN, EIN, EIRN and RCNN models with almost the same number of parameters shown in Table 5.16. The SGD with a starting learning rate of 0.001, a batch size 64, and a total number of 75 epochs were used. In this case, we used the transfer learning approach where weights have been stored after every 25 epochs and then reused as initial weights for the next 25 epochs. The learning rate is decreased by the factor of 10 and weight decay is decreased with respect to the number of epochs of 25 for single time evaluation. The impact of transfer learning is clearly observed during the validation accuracy of IRCNN, EIRN, EIN, and RCNN which is shown in Figure 5.12. Figure 5.13 shows the validation accuracy for DenseNet and DCRN.

In the testing phase, we have evaluated the proposed approaches for Top-1% and Top-5% testing accuracy. Table 5.2 shows the testing accuracy for all the models including RCNN and DenseNet. According to Table 2, the IRCNN provides better performance compared to EIN, EIRN, and RCNN with almost the same number of parameters for object recognition task on the TinyImageNet-200 dataset. We have also conducted experiments with DenseNet [27] and DenseNet with RCL on the TinyImageNet-200 dataset. The experimental results show that DenseNet with RCLs provides about 0.38% improvement on Top-1% accuracy compared to DenseNet with only 1M network parameters. The experimental results show DenseNet with RCLs provides higher testing accuracy in both Top-1% and Top-5% compared against DenseNet model Huang et al. (2016).

Table 5.2: Top-1% and Top-5% testing accuracy on TinyImageNet-200 dataset.

Model	Parameters	Top-1%	Top-5%
EIN	~ 9.3 M	45.27	67.73
RCNN	~ 9.5 M	47.36	69.17
EIRN	~ 9.3 M	51.14	71.41
IRCNN	~ 9.3 M	**51.92**	**72.04**
DenseNet	~ 1.0 M	50.85	70.22
DenseNet-RCL	~ 1.0 M	51.23	70.57

5.5.6 Computational time

The computational cost (in seconds) per epoch of IRCNN, RCNN, EIN, EIRN, DenseNet, and DenseNet with RCLs (DCRN) models for different benchmark datasets are provided in Table 5.3. From this table, it can be seen that as the model becomes bigger, it takes more time per epoch. However, DenseNet takes significantly higher time per epoch compared to other models even though the number of network parameters of this model is less.

5.5.7 Introspection

In this implementation, we only augmented data by applying random horizontal flipping techniques, whereas other models published results when using additional data augmentation techniques such as transition, central crop, and ZCA. The proposed model will provide better recognition accuracy when using datasets with additional augmentation. Due to hardware constraints, we were not able to experiment on massive scale implementations of the IRCNN. This architecture will probably provide even better classification accuracy with large networks on the same datasets. A large-scale implementation of the proposed IRCNN model with advanced components such as LSUV, EVE, and an Exponential Linear Unit (ELU) [150] will likely provide further improved recognition on the CIFAR-10 and CIFAR-100 datasets.

Table 5.3: Computational cost of proposed IRCNN model, EIN, and EIRN in second.

Model	Dataset	Computational time/epoch (in sec.)
IRCNN	MNIST	112
	CIFAR-10	418
	CIFAR-100	422
	SVHN	610
EIN	CIFAR-100	425
EIRN	CIFAR-100	426
IRCNN/RCNN/EIN/EIRN	TinyImageNet-200	~672
DenseNet/ DRCN	TinyImageNet-200	~2780

5.6 Conclusion and Future Works

In this paper, we have proposed a new architecture: Inception Recurrent Convolutional Neural Network (IRCNN) for object classification where we have utilized the power of recurrent techniques for context modulation along with the architecture of inception networks. The experimental results show the promising recognition accuracy compared with different state-of-the-art Deep Convolutional Neural Network (DCNN) models on different benchmark datasets such as MNIST, CIFAR-10, CIFAR-100, and SVHN. However, when the proposed IRCNN architecture is initialized with the LSUV initialization technique and the optimization function of EVE, it achieved an object recognition accuracy of 71.76% on the CIFAR-100 dataset. This is about a 3.5% improvement with respect to an RCNN [52]. In addition, this architecture accelerates the training procedure, which is a concerning issue right now for training large-scale deep learning approach. Furthermore, we empirically investigated our model and determined that it outperforms both the baseline Inception Network and the Inception-Residual Network models. Furthermore, this observation is also true in case of DenseNet and DenseNet Connected Recurrent Network (DCRN) which is experimentally investigated on the TinyImageNet-200 dataset. Moreover, this proposed architecture accelerates the training procedure with faster convergence, which is a big concerning issue right now for training large-scale deep learning approach.

In the future, we would like to improve this model and experiment with large-scale implementation using the teacher-student paradigm (Net2Net) on the ImageNet dataset [31]. In addition, the proposed IRCNN will be tested with advanced activation functions such as ELU [150]. Furthermore, from our observation, this new architecture would be able to model context in input videos, which is another future direction for this work.

CHAPTER 6

IRRCNN FOR OBJECT RECOGNITION

Machine learning and computer vision have driven many of the greatest advances in the modeling of Deep Convolutional Neural Networks (DCNNs). Nowadays, most of the research has been focused on improving recognition accuracy with better DCNN models and learning approaches. The recurrent convolutional approach is not applied very much, other than in a few DCNN architectures. On the other hand, Inception-v4 and Residual networks have promptly become popular among computer the vision community. In this paper, we introduce a new DCNN model called the Inception Recurrent Residual Convolutional Neural Network (IRRCNN), which utilizes the power of the Recurrent Convolutional Neural Network (RCNN), the Inception network, and the Residual network. This approach improves the recognition accuracy of the Inception-residual network with the same number of network parameters. In addition, this proposed architecture generalizes the Inception network, the RCNN, and the Residual network with significantly improved training accuracy. We have empirically evaluated the performance of the IRRCNN model on different benchmarks including CIFAR-10, CIFAR-100, TinyImageNet-200, and CU3D-100. The experimental results show higher recognition accuracy against most of the popular DCNN models including the RCNN. We have also investigated the performance of the IRRCNN approach against the Equivalent Inception Network (EIN) and the Equivalent Inception Residual Network (EIRN) counterpart on the CIFAR-100 dataset. We report around 4.53%, 4.49% and 3.56% improvement in classification accuracy compared with the RCNN, EIN, and EIRN on the CIFAR-100 dataset respectively. Furthermore, the experiment has been conducted on the TinyImageNet-

200 and CU3D-100 datasets where the IRRCNN provides better testing accuracy compared to the Inception Recurrent CNN (IRCNN), the EIN, and the EIRN.

6.1 Introduction

Recently, deep learning using Convolutional Neural Networks (CNN) has shown great success in the field of machine learning and computer vision. The CNN provide state-of-the-art accuracy in various image recognition tasks including object recognition [10], segmentation [174], human activity analysis [175], image super-resolution [176], object detection and tracking [177,179], image captioning [180], and scene understanding [178]. Additionally, this approach has been applied passively in video processing tasks including video classification [181], video representation, and classification of human activity [182]. Deep learning is applied in sentiment analysis which is used for online movie recommendation systems, in addition to other applications [181]. Deep learning approaches are used in the field of machine translation and natural language understanding, and they achieve state-of-the accuracy in this application domain [183]. Furthermore, this technique has been used extensively in the field of speech recognition [184]. Moreover, the deep learning technique is not limited to signal, natural language, image, and video processing tasks; it has been successfully applied in the field of game development [185].

Machine intelligence provides improved performance in many different fields including calculation, chess, memory, and pattern matching, whereas human intelligence still shows better performance in the fields of object recognition and scene understanding tasks. In recent years, deep learning techniques (DCNNs in particular) have been providing outstanding performance for most of the tasks in computer vision. The DCNN is a hierarchical feature learning approach with the multi-level and multi-scale abstraction of features, which aids in the learning of global contextual information from the input samples. However, there is still a gap that must be closed before human-level intelligence can be achieved when performing visual recognition tasks. To reach human-level

performance during recognition tasks, a lot of research is dedicated to understanding the actual process of recognition, as well as understanding the tasks of the visual cortex in the human brain. Studies show that the human brain processes visual information using operations that are similar to convolution or filtering, activation, pooling, and normalization with recurrent connectivity in the visual cortex [186]. The recurrent connectivity of synapses in the human brain plays a big role in context modeling in visual recognition tasks [186, 187].

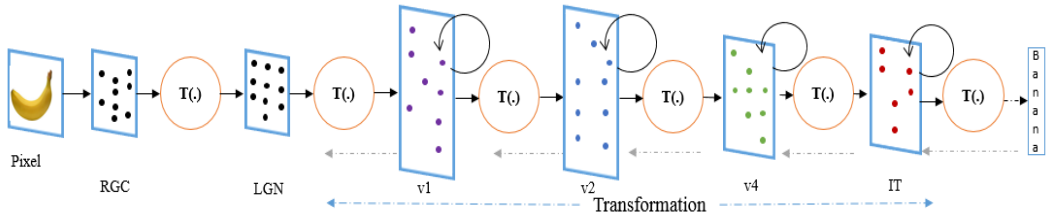


Figure 6.1: Visual information processing pipeline of the human brain, where v1 through v4 represent the visual cortex areas. The visual context areas of v1 through v4 process information using recurrent techniques.

If we observe the structure of recently developed DCNN models, most functionalities are included to design better architectures which are successfully applied to segmentation, detection, and recognition tasks. However, the concept of Recurrent Convolution Layers (RCLs) is included in very few DCNN models, the most prominent being the Recurrent Convolutional Neural Network (RCNN) [52], a CNN with LSTM for object classification [189], and the Inception RCNN [190]. On the other hand, Inception [23,24], and Residual [11,21] architectures are commonly used for solving computer vision tasks. The common practice in the most recently developed Inception and Residual networks is to implement larger and deeper networks to archive better performance. As the model becomes larger and deeper, the parameters of the network are increased dramatically. As a result, the model becomes more complex to train and thus, more computationally expensive. Therefore, it is very important to design an architecture which provides better performance using reasonably fewer numbers of network parameters.

While others are trying to implement bigger and deeper DCNN architectures like GoogLeNet [10], or a Residual Network with 1001 layers [21] to achieve high recognition accuracy on different benchmark datasets. We are presenting an improved version of the DCNN model inspired by the recently developed promising DCNN architectures like Inception-v4 [24], Residual [1], and the RCNN [52]. The proposed model not only ensures better recognition accuracy with the same number of network parameters against other DCNN architectures but also helps to improve the overall training accuracy. The contributions of this work are as follows:

- A new deep learning model named the Inception Recurrent Residual Convolutional Neural Network (IRRCNN) is proposed.
- Empirical evaluation of the performance of the proposed model against different DCNN models on different benchmark datasets such as CIFAR-10, CIFAR-100, TinyImageNet-200, and CU3D-100.
- An empirical investigation of the impact of the RCLs of the IRRCNN against that of the equivalent Inception and Inception-Residual models on the CIFAR-100 and TinyImageNet-200 datasets.
- Large-scale implementation and comparison against Inception-v3 on the CU3D-100 object recognition dataset.

6.2 Related Work

Most of the breakthroughs in the field of computer vision (as well as the ImageNet challenges) have driven the development of the different DCNN architectures in recent years. The deep learning revolution began in 1998 with [18]. From then on, several different architectures have been proposed that have shown great success using many different benchmark datasets including MNIST, SVHN, CIFAR-10, CIFAR-100, ImageNet, and many more. Of the DCNN architectures, AlexNet [7], VGG [9], NiN [19], the All Convolutional Network [20], GoogLeNet [10], Inception-

v4 [24], and the Residual Network [11,21] can be considered the most popular deep learning architectures due to their outstanding performance on different benchmarks for object classification tasks. In most cases, researchers experiment with different models such as NIN, the All Convolutional Network, VGG, GoogLeNet, Inception, and Residual networks, and then select the best model for their application based on the performance. Nevertheless, new models, hybrid models, and optimized versions of existing models have been proposed to achieve better accuracy with fewer network parameters in the last few years.

The concept of Inception was introduced with GoogLeNet [10], and it won the most difficult ImageNet challenge for visual object recognition called the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2014 with remarkably few parameters. The main contribution of this network is to reduce the network parameters drastically when compared to the traditional CNN used in AlexNet. This model introduced a new technique called an inception layer. This approach is not only computationally convenient when compared to the traditional approach, but it also provided the best recognition accuracy in ILSVRC 2014. In terms of network parameters and memory, GoogLeNet needs only 4M whereas AlexNet needs around 60M [7]. An improved version of the Inception network was proposed by Szegedy al et. in 2015, where they scaled up the Inception model utilizing more computation with factorized convolution and aggressive regularization [25]. This model shows a significant improvement in recognition accuracy on ILSVRC 2012.

In 2015, Kiming He al et. proposed a new DCNN architecture called the Residual Network [11] and won the most difficult ILSVRC in 2015. This deep learning technique achieves state-of-the-art recognition accuracy on different benchmarks including ImageNet and CIFAR, as well as on object detection and segmentation tasks on PASCAL VOC and MSCOCO. This architecture is applied to different application domains including machine translation, speech synthesis, speech recognition, and audio classification. Residual networks provide the possibility of building deep network architectures with thousands of layers resulting in significantly improved recognition accuracy

[191]. However, improving just a fraction of a percentage in recognition accuracy requires almost doubling the number of layers in the networks. As a result, the number of model parameters and complexity increases. Therefore, training with very deep networks becomes very difficult due to diminishing feature reuse, which makes the networks very slow to train. Research has been conducted focusing on designing alternative models that produce the same level of recognition accuracy like SqueezeNet, which requires significantly fewer model parameters [148]. To overcome the problem of training complexity in residual networks, wide residual networks (WRN) have been proposed [25], where the width (the number of feature maps) of the networks is increased instead of the depth (number of layers). In 2016, the aggregated residual network was also proposed which is a slight variant of the basic residual network structure [191].

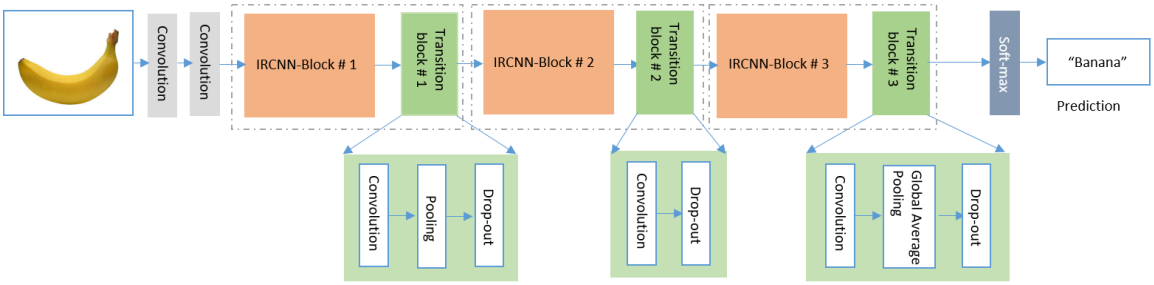


Figure 6.2: The overall layer flow diagram of proposed IRRCNN) consisting of the IRRCNN-Block, the IRRCNN-Transition block, and the Softmax layer at the end.

Most existing research has been concentrated on improving recognition accuracy with different DCNN models. Out of the many modes, very few studies are using RCLs in their models. However, the recurrent approach is very important for context modeling in sequential images and videos. The RCNN structure was proposed for object recognition tasks by Ming et al in 2015 [52]. This deep learning model contains several blocks of RCLs followed by a max-pooling layer. The global max-pooling layer is placed before the classification layer with Softmax at the end. This model provided state-of-the-art accuracy for object classification at that time [52]. In 2014, the Long-term Recurrent Convolutional Network (LRCN) was proposed for visual recognition and description by

Donahue et al. [188]. This architecture contains two popular techniques, the CNN and LSTM. The CNN technique is used for feature extraction, and LSTM is applied to observe how features vary with respect to time. This model shows outstanding performance for visual description [188]. Moreover, some research is being conducted that emphasizes bridging the gap between machine and human intelligence, where the proposed networks utilize recurrent concepts using residual network models [38].

Inception and Residual architectures are very prevalent in the computer vision community. The success of both architectures in the last few years has produced a new path of research that focuses on the discovery of even better models with better performance. Incorporating the new functionalities of RCLs into these state-of-the-art models improves overall recognition accuracy while utilizing the same number of the network parameters. This will have a significant impact on both computer vision and machine learning communities. In this paper, we have proposed an improved DCNN architecture based on Inception [23], Residual networks [11] and the RCNN architecture [52]. Therefore, we call this model the Inception Recurrent Residual Convolutional Neural Network (IRRCNN).

6.3 IRRCNN Architecture

The main objective of this model is to improve recognition performance using the same number or fewer computational parameters when compared to alternative equivalent deep learning approaches. In this model, the inception-residual units utilized are based on Inception-v4 [24]. The Inception-v4 network created by Szeged al et. in 2015 is a deep learning model that concatenates the outputs of the convolution operations with different sized convolution kernels in the inception block [24]. Inception-v4 is a simplified structure of Inception-v3 containing more inception modules using lower rank filters. Furthermore, Inception-v4 includes a residual concept in the inception network called the Inception-v4 Residual Network, which improves the overall accuracy of recognition tasks. In the Inception-Residual network, the outputs of the inception units are added

to the inputs of the respective units. The overall structure of the proposed IRRCNN model is shown in Figure 6.2. From the figure, it can be clearly seen, that the overall model consists of several convolution layers, IRRCNN blocks, transition blocks, and a Softmax at the output layer.

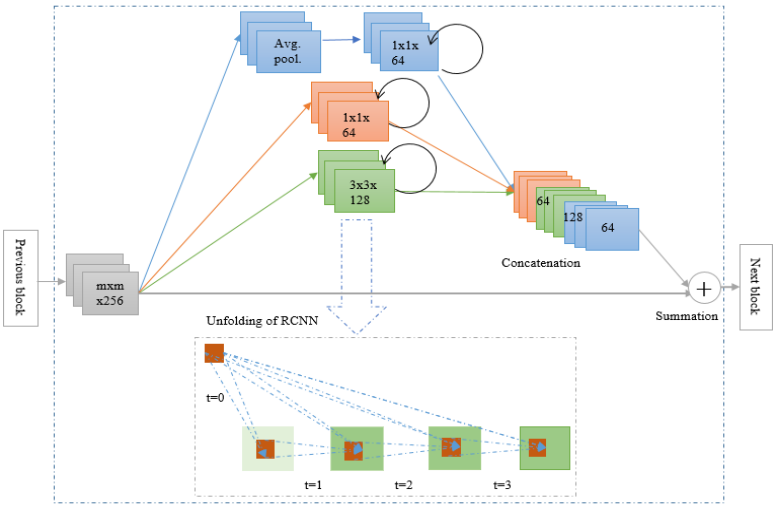


Figure 6.3: The Inception Recurrent Residual Convolutional Neural Network (IRRCNN) block consisting of the inception unit at the top which contains recurrent convolutional layers that are merged by concatenation, and the residual units (summation of the input features with the outputs of the inception unit can be seen at the end of the block).

The most significant part of this proposed architecture is the IRRCNN block that includes RCLs, inception units, and residual units (shown in detail in Figure 6.3). The inputs are fed into the input layer, then passed through inception units where RCLs are applied, and finally, the outputs of the inception units are added to the inputs of the IRRCNN-block. The recurrent convolution operations perform with respect to the different sized kernels in the inception unit. Due to the recurrent structure within the convolution layer, the outputs at the present time step are added with the outputs of the previous time step. The outputs at the present time step are then used as inputs for the next time step. The same operations are performed with respect to the time steps that are considered. For example, here k=2 means that 3 RCLs are included in IRRCNN-block. In the IRRCNN-block, the input and output dimensions do not change, this is simply an accumulation of feature maps with respect to the time steps. As a result, the healthier features ensure that better recognition accuracy is achieved with the same number of network parameters.

The operations of the RCL are performed with respect to the discrete time steps that are expressed according to the RCNN [52]. Let's consider the x_l input sample in the l^{the} a layer of the IRRCNN-block and a pixel located at (i, j) in an input sample on the k^{th} feature map in the RCL. Additionally, let's assume the output of the network $O_{ijk}^{l}(t)$ is at the time step t. The output can be expressed as follows:

$$O_{ijk}^{l}(t) = \left(w_k^f\right)^T * x_l^{f(i,j)}(t) + (w_k^r)^T * x_l^{r(i,j)}(t-1) + b_k \tag{6.1}$$

Here $x_l^{f(i,j)}(t)$and $x_l^{r(i,j)}(t-1)$ are the inputs for the standard convolution layers and for the l^{th} RCL respectively. The w_k^f and w_k^r values are the weights for the standard convolutional layer and the RCL of the k^{th} feature map respectively, and b_k is the bias.

$$y = f(O_{ijk}^{l}(t)) = \max(0, O_{ijk}^{l}(t)) \tag{6.2}$$

Here f is the standard Rectified Linear Unit (ReLU) activation function. We have also explored the performance of this model with the Exponential Linear Unit (ELU) activation function in the following experiments. The outputs y of the inception units for the different size kernels and average pooling layer are defined as $y_{1x1}(x)$, $y_{3x3}(x)$, and $y_{1x1}^{p}(x)$ respectively. The final outputs of Inception Recurrent Convolutional Neural Networks (IRCNN) unit are defined as $\mathcal{F}(x_l, w_l)$ which can be expressed as

$$\mathcal{F}(x_l, w_l) = y_{1x1}(x) \odot y(x) \odot y_{1x1}^{p}(x) \tag{6.3}$$

Here $\odot$ represents the concatenation operation with respect to the channel or feature map axis. The outputs of the IRCNN-unit are then added with the inputs of the IRRCNN-block. The residual operation of the IRRCNN-block can be expressed by the following equation.

$$x_{l+1} = x_l + \mathcal{F}(x_l, w_l) \tag{6.4}$$

Where x_{l+1} refers to the inputs for the immediate next transition block, x_l represents the input samples of the IRRCNN-block, w_l represents the kernel weights of the l^{th} IRRCNN-block, and $\mathcal{F}(x_l, w_l)$ represents the outputs from of l^{th} layer of the IRCNN-unit. However, the number of feature maps and the dimensions of the feature maps for the residual units are the same as in the

IRRCNN-block shown in Figure 6.3. Batch normalization is applied to the outputs of the IRRCNN-block [22]. Eventually, the outputs of this IRRCNN-block are fed to the inputs of the immediate next transition block.

In the **transition block**, different operations are performed including convolution, pooling, and dropout, depending upon the placement of the transition block in the network. We did not include inception units in the transition block on the small-scale implementation for CIFAR-10 and CIFAR-100. However, we have applied inception units to the transition block during the experiment using the TinyImageNet-200 dataset and for the large-scale model which is the equivalent model of Inception-v3 [23]. The down-sampling operations are performed in the transition block where we perform max-pooling operations with a 3×3 patch and a 2×2 stride. The non-overlapping max-pooling operation has a negative impact on model regularization, therefore we used overlapped max-pooling for regularizing the network which is very important when training a deep network architecture [84]. Late use of a pooling layer helps to increase the non-linearity of the features in the network, as this results in higher dimensional feature maps being passed through the convolution layers in the network. We have applied two special pooling layers in the model with three IRRCNN-blocks and a transition-block for the experiments that use the CIFAR-10 or CIFAR-100 dataset.

We used only 1×1 and 3×3 convolution filters in this implementation, as inspired by the NiN [19] and Squeeze Net [148] models. This also helps to keep the number of network parameters at a minimum. The benefit of adding a 1×1 filter is that it helps to increase the non-linearity of the decision function without having any impact on the convolution layer. Since the size of the input and output features does not change in the IRRCNN blocks, it is just a linear projection on the same dimension and non-linearity is added to the RELU and ELU activation functions. We used a 0.5 dropout after each convolution layer in the transition block. Finally, we used a Softmax or normalized exponential function layer at the end of the architecture. For input sample x, weight

vector W, and K distinct linear functions, the Softmax operation can be defined for the i^{th}class as follows:

$$P(y = i|x) = \frac{e^{x^T w_i}}{\sum_{k=1}^{K} e^{x^T w_k}} \quad (6.5)$$

This proposed IRRCNN model has been investigated through a set of experiments on different benchmark datasets and compared across different models.

6.4 Experiments

The proposed IRRCNN model has been evaluated using four different benchmark datasets: CIFAR-10[160], CIFAR-100[160], TinyImageNet-200 [173], and CU3D–100 [192,193]. The dataset statistics are provided in Table 1. We used different validation and testing samples for the TinyImageNet-200 dataset. The entire experiment was conducted on a Linux environment with Keras [171] and Theano [173] at the backend running on a single GPU machine with an NVIDIA GTX-980Ti.

Table 6.1: Statistics for the datasets studied in these experiments.

Dataset	**Training Samples**	**Validation/Testing Samples**	**Total Samples**
CIFAR-10	50,000	10,000/10,000 (same)	60,000
CIFAR-00	50,000	10,000/10,000 (same)	60,000
TinyImageNet-200	100,000	10,000/10,000 (different)	120,000
CU3D-100	12,717	1,413/4,710 (different)	18,840

6.4.1 Experiments on CIFAR-10 and 100 datasets

In this experiment, we used two convolution layers at the beginning of the architecture, three IRRCNN blocks followed by three transition blocks, and one global average pooling and Softmax layer at the end. First, we evaluated the IRRCNN model using the stochastic gradient descent (SGD) technique with the Keras 2.0 [171] default initialization technique. We used momentum

equal to 0.9 [162] and decay equal to 9.99e-07 in this experiment. Second, we evaluated the same model with the Layer-sequential unit-variance (LSUV) initialization method [40] and the latest improved version of the optimization function called EVE [44]. The hyperparameters for the EVE optimization function are as follows: the value of learning rate (λ) is 1e-4, the decay (γ) is 1e-4, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\beta_3 = 0.999$b, κ=0.1, K=10, and $\in = 1e-08$. The values β_1, $\beta_2 \in [0,1)$ are exponential decay rates for moment estimation in Adam. The $\beta_3 \in [0,1)$ is exponential decay rate for computing relative changes. The IRRCNN-block uses the $l2-$ norm for a weight regularization of 0.002. We used the ReLU activation function in the first experiment, and the ELU activation is used in the second experiment. In both experiments, we trained the networks for 350 epochs with a batch size of 128 for CIFAR-10 and 100.

CIFAR -10: The CIFAR-10 dataset is a benchmark dataset for object classification [160]. The dataset consists of 32×32 color images split into 50,000 samples for training, and the remaining 10,000 samples are used for testing (classification into one of 10 classes). The experiment was conducted with and without data augmentation. When using data augmentation, we applied only random horizontal flipping. Using this proposed approach, we have achieved around 8.41% testing error without data augmentation and 7.37% testing error with augmented data (only horizontal flipping) using SDG techniques.

The proposed model shows better recognition against most of the DCNN models displayed in Table 6.2. Furthermore, improved performance is observed in the IRCNN that used LSUV [70] initialization and the EVE [151] optimization function. The results show a testing error of around 8.17% and 7.11% without and with data augmentation respectively. It is also observed that the IRRCNN shows better performance when compared to the equivalent IRCNN model [194].

Figure 6.4: Example images from the CIFAR-10 dataset.

Table 6.2: Testing error (%) of the IRRCNN on CIFAR-10 object classification dataset without and with data augmentation. For unbiased comparison, we have listed the accuracy stated in recent studies using a similar experimental setting.

Methods	# Parameters	Error (%) without data augmentation	Error (%) with data augmentation
Maxout [163]	>6M	11.6	9.38
Network in Network (NiN) [19]	~1M	10.4	8.81
Deeply Supervised Network DSN [166]	~1M	9.69	7.97
ALL-CNN [20]		9.08	7.25
Highway Network [167]		-	7.72
RCNN [52]		8.69	7.09
dasNet[165]		-	9.22
FitNet [169]	~2.5M	-	8.39
Residual Net [11]			7.51
IRCNN +SDG+ReLU	3.5 M	8.41	7.37
IRCNN + LSUV + EVE +ReLU	3.5 M	8.17	7.11
IRRCNN+SGD+ReLU	3.5 M	8.14	7.23
IRRCNN + LSUV + EVE+ReLU	3.5 M	8.11	**7.06**

CIFAR-100: Another similar benchmark for object classification was developed in 2009 [160]. The dataset contains 50,000 samples for training and 10,000 samples for validation and testing. Each sample is a 32×32×3 image, and the dataset has 100 classes. The proposed IRRCNN model

was studied with and without data augmentation. During the experiment with augmented data, the SGD and LSUV [70] initialization approaches and the EVE optimization function were used [151]. In both cases, the proposed technique shows better recognition accuracy compared with different DCNN models including the IRCNN [194]. The validation accuracy of the IRRCNN model for both experiments on CIFAR-100 with data augmentation is shown in Figure 6.5. The proposed IRRCNN model shows better performance in both experiments when compared to the IRCNN [194], EIN, and EIRN models. The experimental results when using CIFAR-100 are shown in Table 6.3. The IRRCNN model provides better testing accuracy compared to many recently developed methods. We have achieved 72.78% recognition accuracy with LSUV+EVE which is around a 4.49% improvement compared to one of the baseline RCNN methods with almost the same number of parameters (~3.5M) [52].

Table 6.3: Testing error (%) of the IRRCNN on the CIFAR-100 object classification dataset without and with data augmentation (DA). For unbiased comparison, we have listed the accuracy provided by recent studies in a similar experimental setting. Here C100 refers without data augmentation and C100+ refers to data augmentation.

Methods	# of parameters	Error (%) without DA.	Error (%) with DA.
CNN+Tree based priors [170]	-		36.85
Maxout 163]	>5M		38.57
Prob maxout [168]	>5M		38.14
NIN [19]	1.98M	35.68	35.68
DSN [166]	1.98M		34.57
RCNN-160 [52]	1.87M		31.75
dasNet[165]	-		33.78
ALL Conv [20]	-		33.71
HighwayNet[167]	-		32.24
FitNet 169]	-		35.04
IRCNN+SGD+ReLU	~3.5M	34.13	31.22
IRCNN+LSUV+EVE+ReLU	~3.5M	30.87	28.24
IRRCNN+SGD+ELU	~3.5M	33.07	29.21
IRRCNN + LSUV + EVE+ELU	~3.5M	29.67	27.10

6.4.2 Impact of recurrent convolution layers

A question may arise here: is there any advantage of the IRRCNN model against the EIRN and EIN architectures? The EIN and EIRN models are implemented with a similar architecture with the same number of network parameters (~3.5 M). We used sequential convolution layers with the same time-step with the same size kernels instead of using RCLs for implementing the EIN and EIRN models. In addition, in the case of EIRN, we incorporated the residual concept with an Inception-block like Inception-v4 [24]. Furthermore, we have investigated the performance of the IRRCNN model against the RCNN with the same number of parameters on the TinyImageNet-200 dataset.

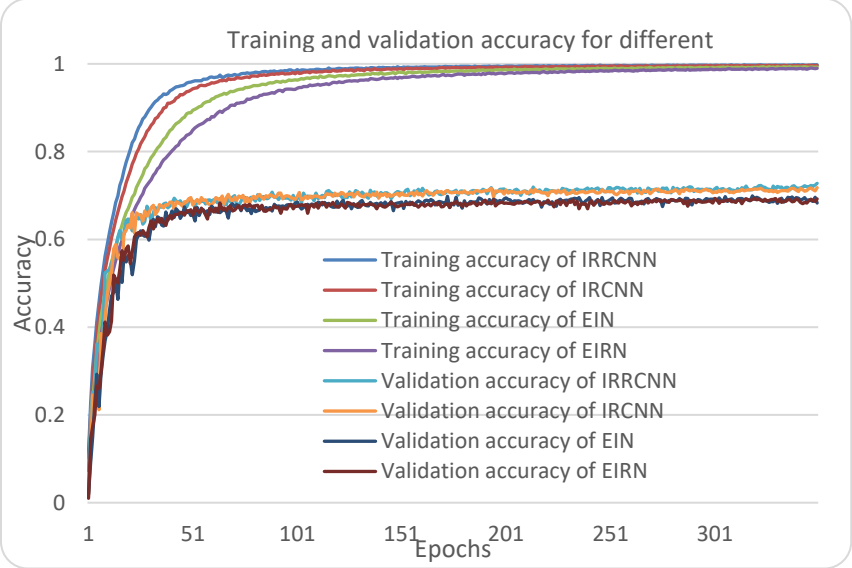


Figure 6.5: Training and validation accuracy for IRRCNN, IRCNN, BIN, and BIRN on CIFAR-100. The vertical and horizontal axis represents accuracy and epochs respectively. Our proposed model shows the best recognition accuracy in all cases.

A possible second question may arise: Is the IRRCNN model providing better performance due to the use of advanced deep learning techniques? It is noted that LSUV initialization approach applied to the DCNN architecture called FitNet4 achieved 70.04% classification accuracy on augmented data with mirroring and random shifts for CIFAR-100 [160]. In contrast, we only

applied random horizontal flipping for data augmentation and achieved around 1.76% better recognition accuracy against FitNet4 [169]. The model accuracy for both training and validation are shown in Figure 6.5. From the figures, it is clearly observed that this proposed model shows a lower loss and highest recognition accuracy compared to EIN and EIRN, which proves the necessity of the proposed models. The testing accuracy of IRRCNN, IRCNN, EIN, and EIRN are shown in Figure 6.6. It can be summarized that the proposed IRRCNN provides around 1.02%, 4.49%, and 3.56% improved testing accuracy compared to IRCNN, EIN, and EIRN respectively.

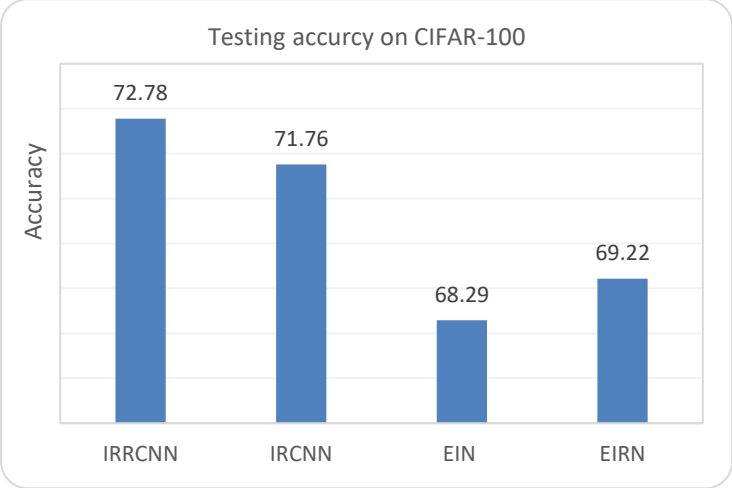


Figure 6.6: Testing accuracy of the proposed IRRCNN model against IRCNN, EIN, and EIRN on the augmented CIFAR-100 dataset.

6.4.3 Experiment on TinyImageNet-200

We also evaluated the proposed approach on the TinyImageNet-200 dataset [173]. This dataset contains 100,000 samples for training, 10,000 samples for validation, and 10,000 samples for testing. These images are sourced from 200 different classes of objects. The main difference between the main ImageNet dataset and Tiny ImageNet is the images are downsampled from 224x224 to 64x64. There are some negative impacts of down-sampling, like loss of detail. Therefore, downsampling the images lead to ambiguity, which makes this problem even harder and this effects overall model accuracy.

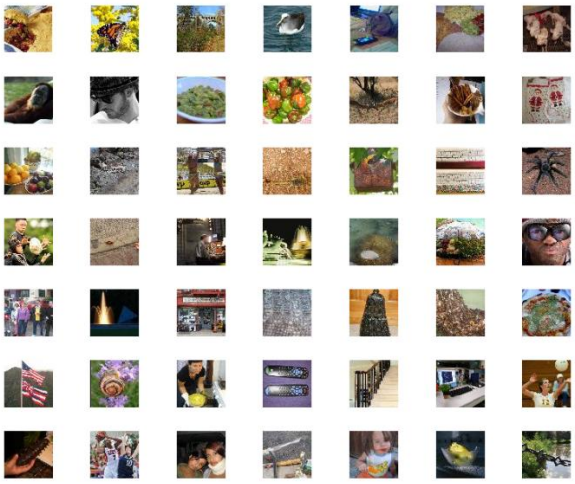


Figure 6.7: Sample images from the TinyImageNet-200 dataset.

For this experiment, we used the IRRCNN model with two general convolution layers with a 3×3 kernel at the beginning of the network followed by a sub-sampling layer with 3×3 convolution using a stride of 2×2. After that, four IRRCNN blocks are used followed by four transition blocks. Finally, a global average pooling layer is used followed by a Softmax layer.

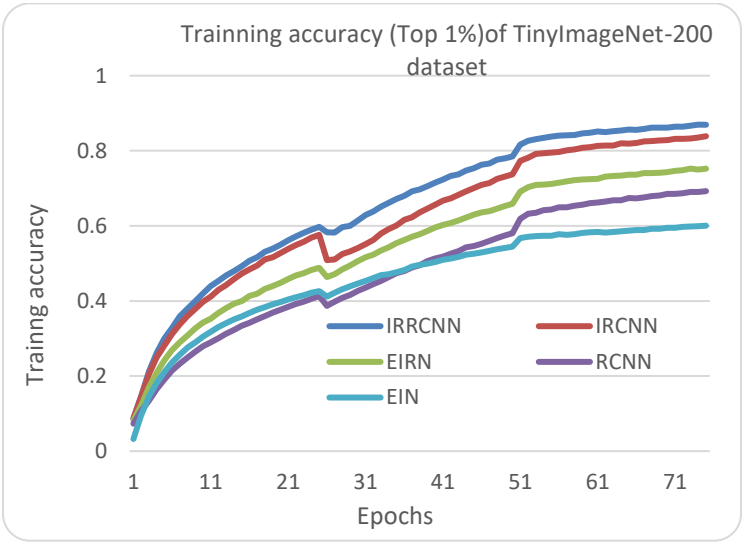


Figure 6.8: Training accuracy during training for TinyImageNet-200 dataset.

We have experimented with the IRRCNN, IRCNN, equivalent RCNN, EIN, and EIRN using the TinyImageNet-200 dataset. The training accuracy of this experiment is shown in Figure 6.8. The

proposed IRRCNN model provides better recognition accuracy during training compared to equivalent models including IRCNN, EIN, and EIRN with almost the same number of network parameters (~15M). Generally, DCNN takes a lot of time and power when training a reasonably large model. The Inception-Residual networks with RCLs significantly reduce training time with faster convergence and better recognition accuracy. The validation accuracy for all of these models is shown in Figure 6.9.

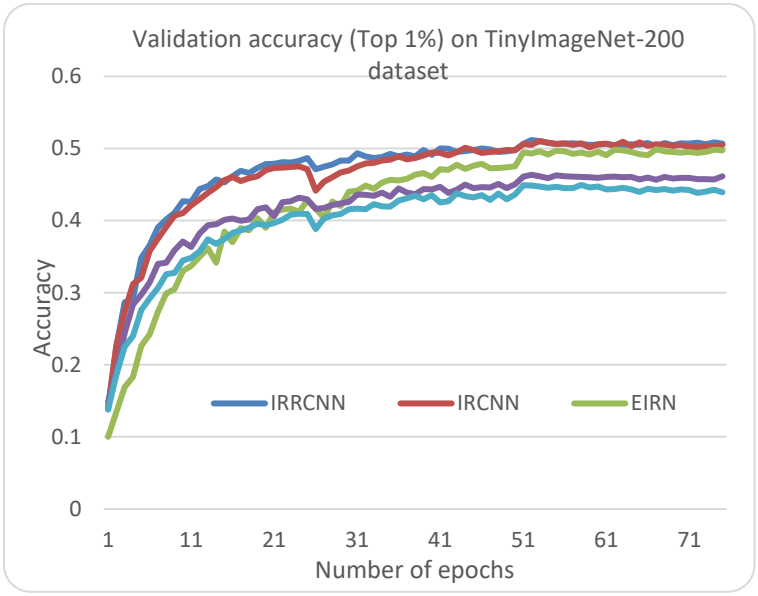


Figure 6.9: Validation accuracy on the Tiny-ImageNet dataset.

We have evaluated our proposed approach for both Top-1% and Top-5% testing accuracy as shown in Figure 6.10. From the bar graph, the impact of recurrent connectivity is clearly observed, and we have achieved 52.23% top-1% testing accuracy whereas the EIRN and EIN show 51.14% and 45.63% top-1% testing accuracy. The same behavior is observed for Top-5% accuracy as well. The IRRCNN provides better testing accuracy when compared against all other models in both cases which absolutely displays the robustness of the proposed deep learning architecture.

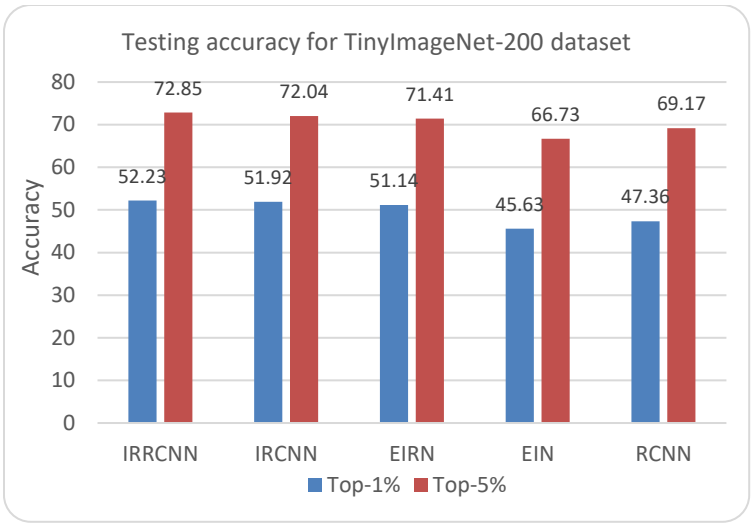


Figure 6.10: Top-1% and Top-5% testing accuracy on TinyImageNet-200 dataset.

6.4.4 Inception-v3, WRN versus equivalent IRRCNN model

We evaluated the IRRCNN model with large-scale implementation against the Inception-v3 and WRN. The IRRCNN model is implemented a with a similar structure to the Incpeiton-v3 for impartial comparison. We used the default implementation of Keras version 2.0 and we just incorporated the RCLs, where $(k = 2)$, which means 2 RCLs are used in the Inception units and a residual layer is added at the end of the block. We trained the network with the SGD method with momentum. The concept of transfer learning is used where training was performed for 100 epochs in total. After successfully completing the initial training process for 50 epochs with a learning rate of 0.001, the learned weights were used as initial weights for the next 50 epochs for fine-tuning of the network with a learning rate of 0.0001.

CU3D-100 dataset: Another very high-quality visual object recognition dataset with well-controlled images (e.g., object invariance, features complexity) is CU3D-100, which is suitable for the evaluation of new deep learning algorithms. This dataset contains 18,840 color images in total that have a dimension of 64×64×3 and 20 samples per exemplary. The following figure shows some example images from the CU3D-100 dataset.

Figure 6.11: Example images of the CU3D-100 dataset.

The images in this dataset are three-dimensional views of real-world objects normalized for different positions, orientations, and scales.

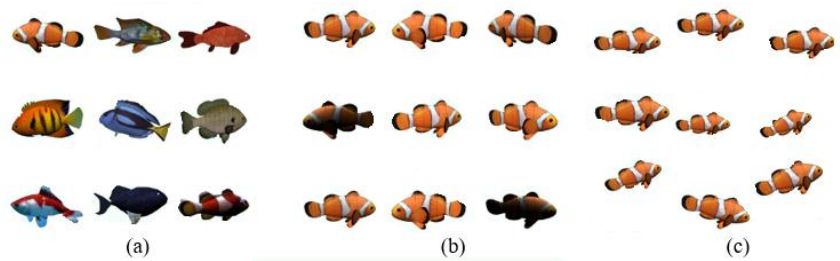


Figure 6.12: Sample images displaying (a) nine examples from the fish category, (b) nine depth, tilt, and lighting variations of the fish category, and (c) nine affine transformation images for a single view.

The rendered images have a 40^0 depth rotation about the y-axis (plus a horizontal flip), a 20^0 tilt rotation about the x-axis, and an 80^0 overhead lighting rotation. We used 75% percent of the images for training and the remaining 25% images for testing, which were selected randomly from the

whole dataset. The example images in the fish category with different lighting condition and affine transformations are shown in Figure 6.12.

Experimental results on CU3D: We have conducted two different experiments wherein the first case, the models are trained from scratch and transfer learning approach with the trained weights of the ImageNet dataset is used for the second experiment on the CU3D-100 dataset. We have considered models of IRRCNN which is equivalent of Inception-v3 which contains ~19.74M and ~21.25M network parameters respectively. The WRN model consists of deep and wide factors n = 6 and k = 6 respectively and contains ~ 31.25M network parameters [25]. The entire dataset was first divided into two sets where 75% of the (14,130) samples are used for training and validation, and the remaining 25% (4,710) of the samples are used for testing. For the first experiment, using 14,130 samples, 10% of the samples are used for validation during training. The training and validation accuracies for 25 epochs are shown in Figure 6.13. Figure 6.13 shows that the IRRCNN model exhibits lower error during training and validation when compared to the Inception-v3 and WRN models.

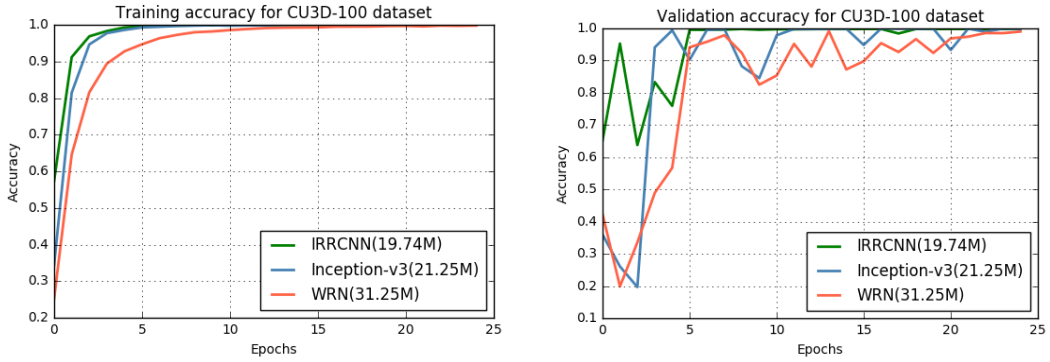


Figure 6.13: Training and validation accuracy with respect to epoch for the CU3D-100 dataset.

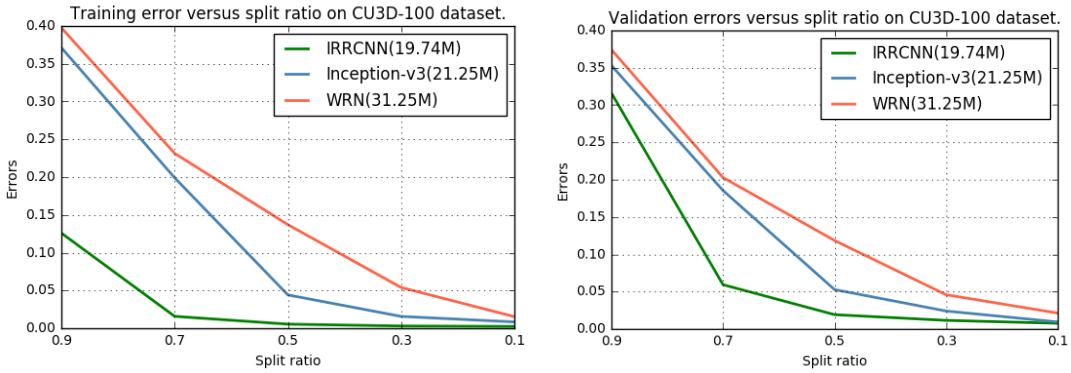
In the testing phase of the first experiment, we have achieved 99.81%, 99.13%, and 98.51% testing accuracy with IRRCNN, Inception-v3, and WRN respectively. The IRRCNN model shows 0.68% and 1.30% higher testing accuracy against Inception-v3 and WRN respectively. A recently published paper with sparse parameterization back-propagation in a network with recurrent layers

reported about 94.6% testing accuracy on the CU3D-100 dataset [193], which is around 5.24% less testing accuracy compared to this proposed IRRCNN model. In the second experiment, the pre-trained ImageNet weights are used as initial weights for IRRCNN and Inception-v3 models where we have trained only a few of the layers at the top of the models. The trained weights were taken from GitHub [195]. The proposed model gives about 98.84% testing accuracy whereas Inception-v3 model gives 92.16% testing accuracy on the CU3D-100 dataset. The IRRCNN model shows around 6.68% better testing accuracy compared to the similar Inception-v3 model and clearly demonstrates the impact of RCL and residual layers in the models. This experiment also proves the robustness of the IRRCNN model when dealing with scale invariance, position and rotation invariance, and different lighting condition input samples.

6.4.5 Trade-off between split ratio and accuracy

To further investigate the performance of the proposed IRRCNN model, the trade-o between the split ratio versus performance is investigated against Inception-v3[23] and WRN[25]. During this experiment, we used different split ratios including [0.9, 0.7, 0.5, 0.3, and 0.1]. The number of training and validation samples are taken according to the split ratio where the number of training samples is increased, and the number of validation samples is decreased in the trials respectively. For example, a split ratio of 0.9 refers to only 10% of the samples (1423) being used for training and the remaining 90% of the samples (12815) are used for validation, while a split ratio of 0.7 means 30% of the samples are used for training and the remaining 70% of the samples are used for validation and so on. However, it can be also observed from Figure 6.13 that the models converged after 22 epochs. Therefore, in each trial, we considered 25 epochs and the error here is the average training and validation error for the last five epochs. Figure 6.14 shows the training and validation errors with respect to split ratios. This figure shows that the proposed IRRCNN model shows less training and validation errors for five different trials in both cases. These results clearly demonstrate

that the IRRCNN is more capable at extracting, representing, and learning features during the training phase which ultimately helps to ensure better testing performance. In each trial, we tested the models with the remaining 25% of the samples, and the testing errors are shown in Figure 6.15. From this figure, it is clear that IRRCNN shows the lowest error for almost all trails compared to Inception-v3 [23] and WRN [25].



(a) (b)

Figure 6.14: The training and validation errors versus split ratio for five different trials on the CU3D-100 dataset.

It can be also observed from the which is shown in Figure 6.14 that the models are converged after 22 epochs. Therefore, in each trial, we have considered 25 epochs and the errors here is the average training and validation errors of the last five epochs. The figures 15(a) and (b) show the training and validation errors respect to split ratios. These figures show that the proposed IRRCNN model shows less training and validation errors for five different trials in both cases. These results clearly demonstrate that the IRRCNN is more capable at extracting, representing, and learning features during the training phase which ultimately helps to ensure better testing performance. In each trial, we have tested the models with the remaining 25% of the samples and the testing errors are shown in Figure 6.15. From this figure, it is clearly seen that R2U-Net shows the lowest error for almost all trails compared to Inception-v3 [23] and WRN [25].

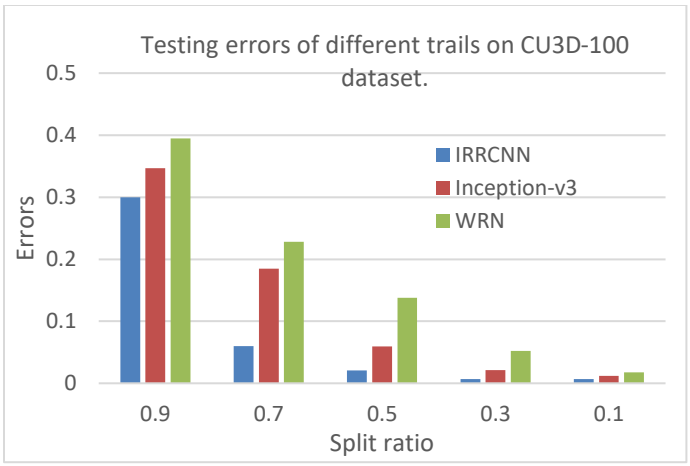


Figure 6.15: Testing accuracy for different trials on the CU3D-100 dataset.

Computational time: the computational time of the proposed models with other equivalent models for different datasets are shown in Table 6.4.

Table 6.4 Computational time per epoch for different models using different datasets.

Models	Dataset	time/epoch (in sec.)
IRRCNN/IRCNN /EIN/EIRN	CIFAR-10	~422
IRRCNN/IRCNN /EIN/EIRN	CIFAR-100	~425
IRRCNN/IRCNN /EIN/EIRN	TinyImageNet-200	~765
IRRCNN/ Inception-v3	CU3D-100	~958

6.4.6 Introspection

From our investigation, we have observed that the proposed IRRCNN model converges faster when compared to the RCNN, EIR, EIRN, and IRCNN models which are clearly evaluated using a set of experiments. The proposed techniques provide promising recognition accuracy during the testing phase with the same number of network parameters compared with other models. In this implementation, we have augmented input samples by applying only random horizontal flipping. From our observation, the proposed model will provide even better recognition accuracy with more augmentations including transition, central crop, and ZCA.

6.5 Conclusion

In this paper, we have proposed the Inception Recurrent Residual Convolutional Neural Network (in short IRRCNN) for object recognition where we have utilized the power of recurrent convolution neural layers for context modulation based on the Inception and Residual Network architectures. The experimental results show promising recognition accuracy compared with different Deep Convolutional Neural Network (DCNN) models on different benchmarks including CIFAR-10, CIFAR-100, TinyImageNet-200, and CU3D-100. However, the proposed model has been evaluated with different advanced training approaches including SGD, initialization with Layer-sequential unit-variance (LSUV), and the recently proposed optimization methods of EVE. The IRRCNN model with LSUV and EVE achieved a promising object recognition accuracy of 72.78% on the CIFAR-100 dataset which is about a 4.53% improvement when compared to the Recurrent Convolutional Neural Network (RCNN) [19]. In addition, our model provides about 4.49% and 3.56% improvement in recognition accuracy when compared with Equivalent Inception Networks (EIN) and Equivalent Inception-Residual Networks (EIRN) on the CIFAR-100 dataset. Furthermore, we have achieved better recognition accuracy with IRRCNN when compared to EIRN, EIN, RCNN, and IRCNN on the TinyImageNet-200 dataset. Moreover, the large-scale implementation of the Inception-v3 network with recurrent convolutional layers (RCL) provides around 6.5% better recognition accuracy against the Inception-v3 model on the CU3D-100 dataset. Based on all experimental evaluations, it is clearly observed that the proposed architecture accelerates the training process, which is a big issue right now for training large-scale deep learning approach. In the future, we would like to improve this model and explore segmentation and detection tasks.

CHAPTER 7

R2U-Net FOR MEDICAL IMAGE SEGMENTATION

Deep learning (DL) based semantic segmentation methods have been providing state-of-the-art performance in the last few years. More specifically, these techniques have been successfully applied to medical image classification, segmentation, and detection tasks. One deep learning technique, U-Net, has become one of the most popular for these applications. In this paper, we propose a Recurrent U-Net as well as a Recurrent Residual U-Net model, which are named RU-Net and R2U-Net respectively. The proposed models utilize the power of U-Net, Residual Networks, and Recurrent Convolutional Neural Networks (RCNNs). There are several advantages of these proposed architectures for segmentation tasks. First, a residual unit helps when training deep architectures. Second, feature accumulation with recurrent residual convolutional layers ensures better feature representation for segmentation tasks. Third, it allows us to design better U-Net architectures with the same number of network parameters with better performance for medical image segmentation. The proposed models are tested on three benchmark datasets such as blood vessel segmentation in retinal images, skin cancer segmentation, and lung lesion segmentation. The experimental results show superior performance on segmentation tasks compared to equivalent models including a variant of a fully connected convolutional neural network (FCN) called SegNet, U-Net, and the residual U-Net (ResU-Net).

7.1 Introduction

Nowadays DL provides state-of-the-art performance for image classification [10], segmentation [174], detection and tracking [179], and captioning [180]. Since 2012, several Deep Convolutional

Neural Network (DCNN) models have been proposed such as AlexNet [7], VGG [9], GoogleNet [10], Residual Net [11], DenseNet [27], and CapsuleNet [66]. A DL based approach (CNN in particular) provides state-of-the-art performance for classification and segmentation tasks for several reasons [1]: first, activation functions resolve training problems in DL approaches. Second, the concept of dropout helps to regularize the networks. Third, several efficient optimization techniques are available for training CNN models [10]. However, in most cases, models are explored and evaluated using classification tasks on very large-scale datasets like ImageNet [10], where the outputs of the classification tasks are a single label or probability values. Alternatively, small models with architectural variants are used for semantic image segmentation tasks. For example, an FCN also provides state-of-the-art results for image segmentation tasks in computer vision [174]. Another variant of FCN was also proposed which is called SegNet [174].

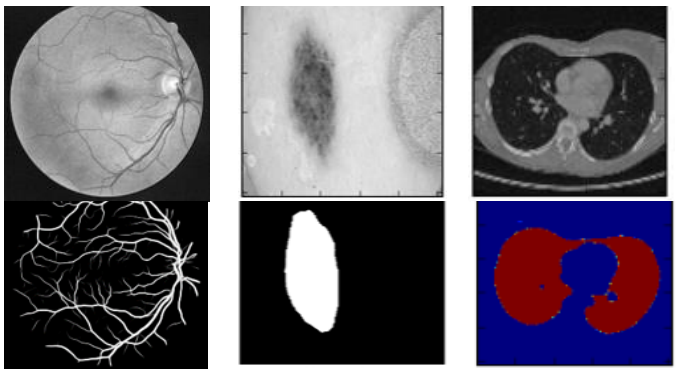


Figure 7.1: Medical image segmentation examples displaying retina blood vessel segmentation on the left, skin cancer lesion segmentation in the middle, and lung segmentation on the right.

Due to the great success of DCNNs in the field of computer vision, different variants of this approach are applied in different modalities of medical imaging including segmentation, classification, detection, registration, and medical information processing. The medical imaging comes from different imaging techniques such as Computer Tomography (CT), ultrasound, X-ray, and Magnetic Resonance Imaging (MRI). The goal of Computer-Aided Diagnosis (CAD) is to obtain a faster and better diagnosis to ensure better treatment of a large number of people at the

same time. Additionally, efficient automatic processing reduces human error and also reduces overall time and cost. Due to the slow process and tedious nature of manual segmentation approaches, there is a significant demand for computer algorithms that can perform segmentation quickly and accurately without human interaction. However, there are some limitations of medical image segmentation including data scarcity and class imbalance. Most of the time a large number of labels (for example, thousands) for training is not available for several reasons [3]. Labeling the dataset requires an expert in this field which is expensive, and it requires a lot of effort and time. Sometimes, different data transformation or augmentation techniques (data whitening, rotation, translation, and scaling) are applied for increasing the number of labeled samples available [199, 200, and 201]. In addition, patch-based approaches are used for solving class imbalance problems. In this work, we have evaluated the proposed approaches on both patch-based and entire image-based approaches. However, to switch from the patch-based approach to the pixel-based approach that works with the entire image, we must be aware of the class imbalance problem. In the case of semantic segmentation, the image backgrounds are assigned a label and the foreground or target regions are assigned with different classes. Therefore, the class imbalance problem is resolved without any trouble. Two advanced techniques including cross-entropy loss and dice similarity were introduced for efficient training of classification and segmentation tasks in [200, 201].

Furthermore, in medical image processing, global localization and context modulation is very often applied for localization tasks. Each pixel is assigned a class label with the desired boundary that is related to the contour of the target lesion in identification tasks. To define these target lesion boundaries, we must emphasize the related pixels. Landmark detection in medical imaging [202, 203] is one example of this. There were several traditional machine learning and image processing techniques available for medical image segmentation tasks before the DL revolution, including amplitude segmentation based on histogram features [204], the region-based segmentation method [205], and the graph-cut approach [206]. However, semantic segmentation approaches that utilize DL have become very popular in recent years in the field of medical image segmentation, lesion

detection, and localization [207]. In addition, DL based approaches are known as universal learning approaches, where a single model can be utilized efficiently in different modalities of medical imaging such as MRI, CT, and X-ray.

According to a recent survey, DL approaches are applied to almost all modalities of medical imaging [3, 4]. Furthermore, a large number of papers has been published on segmentation tasks in different modalities of medical imaging [3, 4]. A DCNN based brain tumor segmentation and detection method were proposed in [208]. From an architectural point of view, the CNN model for classification tasks requires an encoding unit and provides class probability as an output. In classification tasks, we have performed convolution operations with activation functions followed by subsampling layers, and this reduces the dimensionality of the feature maps. As the input samples traverse through the layers of the network, the number of feature maps increases but the dimensionality of the feature maps decreases. This is shown in the first part of the model (in green) in Figure 7.2. Since, the number of feature maps increases in the deeper layers, the number of network parameters also increases. Eventually, the softmax operations are applied at the end of the network to compute the probability of the target classes.

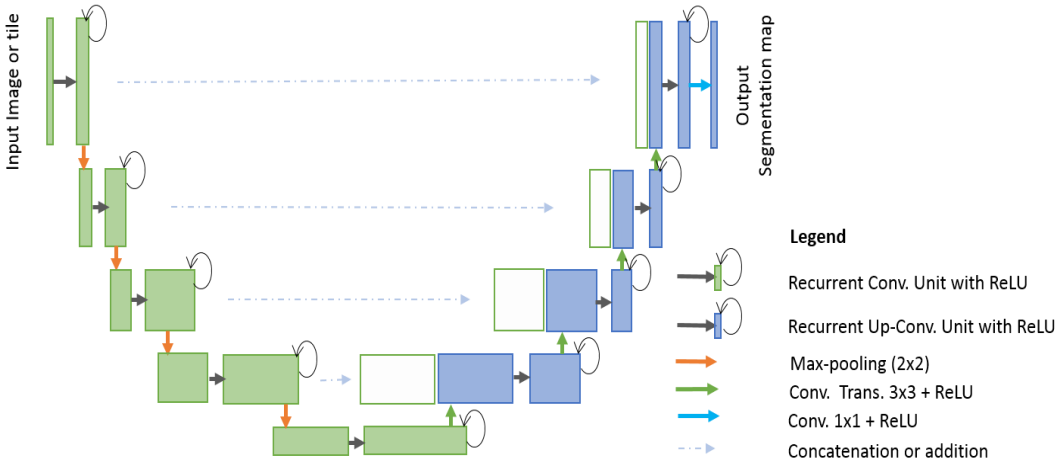


Figure 7.2: The RU-Net architecture with convolutional encoding and decoding units using recurrent convolutional layers (RCL), which is based on a U-Net architecture. The residual units are used with the RCL and R2U-Net architectures.

As opposed to classification tasks, the architecture of segmentation tasks requires both convolutional encoding and decoding units. The encoding unit is used to encode input images into a larger number of maps with lower dimensionality. The decoding unit is used to perform up-convolution (transpose convolution, or what is occasionally called de-convolution) operations to produce segmentation maps with the same dimensionality as the original input image. Therefore, the architecture for segmentation tasks generally requires almost double the number of network parameters when compared to the architecture of the classification tasks. Thus, it is important to design efficient DCNN architectures for segmentation tasks which can ensure better performance with fewer numbers of network parameters.

This research demonstrates two modified and improved segmentation models, one using recurrent convolution networks, and another using recurrent residual convolutional networks. To accomplish our goals, the proposed models are evaluated on different modalities of medical imaging as shown in Figure 7.1. The contributions of this work can be summarized as follows:

- Two new models called RU-Net and R2U-Net are introduced for medical image segmentation.
- Experiments are conducted on three different modalities of medical imaging including retinal blood vessel segmentation, skin cancer segmentation, and lung segmentation.
- Performance evaluation of the proposed models is conducted for the patch-based method for retina blood vessel segmentation tasks and the end-to-end image-based approach for skin lesion and lung segmentation tasks.
- Comparison against recently proposed state-of-the-art methods shows superior performance against equivalent models with the same number of network parameters.
- Empirical evaluation of the robustness of the proposed R2U-Net model against SegNet [197] and U-Net [32] based on the trade-off between the number of training samples and performance during the training, validation, and testing phases.

7.2 Related Work

Semantic segmentation is an active research area where DCNNs are used to classify each pixel in the image individually, which is fueled by different challenging datasets in the fields of computer vision and medical imaging [3, 4]. Before the deep learning revolution, the traditional machine learning approach mostly relied on hand engineered features that were used for classifying pixels independently. In the last few years, a lot of models have been proposed that have proved that deeper networks are better for recognition and segmentation tasks [1]. However, training very deep models are difficult due to the vanishing gradient problem, which is resolved by implementing modern activation functions such as Rectified Linear Units (ReLU) or Exponential Linear Units (ELU) [1,150]. Another solution to this problem was proposed by He et al., a deep residual model that overcomes the problem of utilizing identity mapping to facilitate the training process [11].

In addition, CNN based segmentation methods based on the FCN provide superior performance for natural image segmentation [33]. The performance of FCN has improved with recurrent neural networks (RNN), which are fine-tuned on very large datasets [209]. Semantic image segmentation with DeepLab is currently one of the state-of-the-art methods [210]. SegNet consists of two parts, one part is the encoding network which is a 13-layer VGG16 network [9], and the corresponding decoding network uses pixel-wise classification layers. The main contribution of [197] is the way in which the decoder up-samples its lower resolution input feature maps. Later, an improved version of SegNet, which is called Bayesian SegNet was proposed in 2015 [211]. Most of these architectures are explored using computer vision applications. However, there are some deep learning models that have been proposed specifically for the medical image segmentation, as they consider data insufficiency and class imbalance problems.

One of the first and most popular approaches for semantic medical image segmentation is called U-Net [32]. According to the U-Net architecture, the network consists of two main parts: the convolutional encoding and decoding units. The basic convolution operations are performed followed by ReLU activation in both parts of the network. For down sampling in the encoding unit,

2×2 max-pooling operations are performed. In the decoding phase, the convolution transpose (representing up-convolution, or de-convolution) operations are performed to up-sample the feature maps. The very first version of U-Net used to crop and copy feature maps from the encoding unit to the decoding unit. The U-Net model provides several advantages for segmentation tasks: first, this model allows for the use of global location and context at the same time. Second, it works with very few training samples and provides better performance for segmentation tasks [32]. Third, an end-to-end pipeline processes the entire image in the forward pass and directly produces segmentation maps. This ensures that U-Net preserves the full context of the input images, which is a major advantage when compared to patch-based segmentation approaches [32].

However, U-Net is not only limited to applications in the domain of medical imaging but nowadays this model is also applied for computer vision tasks as well [32, 212]. Meanwhile, different variants of U-Net models have been proposed, including a very simple variant of U-Net for CNN-based segmentation of Medical Imaging data [213]. In this model, two modifications are made to the original design of U-Net: first, a combination of multiple segmentation maps and forward feature maps are summed (element-wise) from one part of the network to the other. The feature maps are taken from different layers of the encoding and decoding units, and finally, summation (element-wise) is performed outside of the encoding and decoding units. The authors report promising performance improvement during training with better convergence compared to U-Net, but no benefit was observed when using a summation of features during the testing phase [213]. However, this concept proved that feature summation impacts the performance of a network. The importance of skipped connections for biomedical image segmentation tasks has been empirically evaluated with U-Net and residual networks [214]. The Deep Contour-Aware Network (DCAN) was proposed in 2016, which can extract multi-level contextual features using a hierarchical architecture for accurate gland segmentation of histology images, and it shows very good performance for segmentation [43]. Furthermore, Nabla-Net, a deep dig-like convolutional architecture was proposed for segmentation in 2017 [44].

Other deep learning approaches have been proposed based on U-Net for 3D medical image segmentation tasks as well. The 3D U-Net architecture for volumetric segmentation learns from sparsely annotated volumetric images [45]. A powerful end-to-end 3D medical image segmentation system based on volumetric images called V-Net has been proposed, which consists of an FCN with residual connections [51]. This paper also introduces a dice loss layer [51]. Furthermore, a 3D deeply supervised approach for automated segmentation of volumetric medical images was presented in [37]. HighRes3DNet was proposed using residual networks for 3D segmentation tasks in 2016 [38]. In 2017, a CNN based brain tumor segmentation approach was proposed using a 3D-CNN model with a fully connected CRF [48]. Pancreas segmentation was proposed in [50], and VoxResNet was proposed in 2016 where a deep voxel wise residual network is used for brain segmentation. This architecture utilizes residual networks and summation of feature maps from different layers [41].

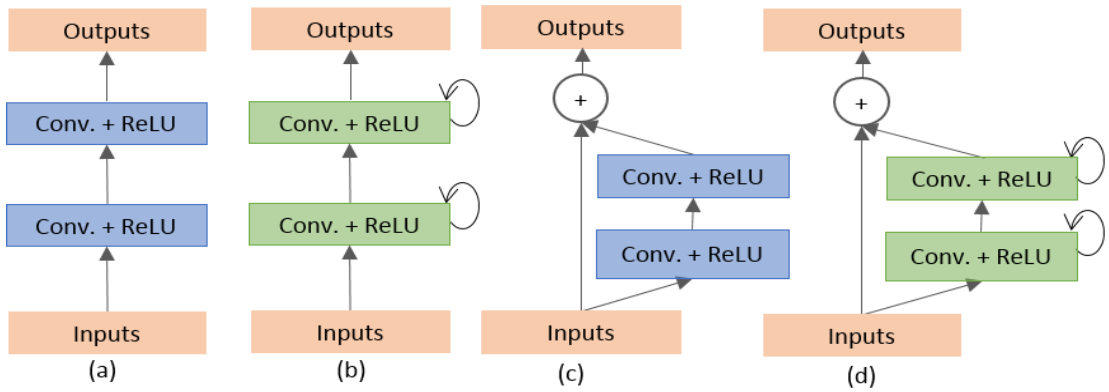


Figure 7.3: Different variants of the convolutional and recurrent convolutional units including (a) the forward convolutional unit, (b) the recurrent convolutional block (c) the residual convolutional unit, and (d) the Recurrent Residual Convolutional Unit (RRCU).

Alternatively, we have proposed two models for semantic segmentation based on the architecture of U-Net in this paper. The proposed Recurrent Convolutional Neural Network (RCNN) model based on U-Net is named RU-Net, which is shown in Figure 7.2. Additionally, we have proposed

a residual RCNN based U-Net model which is called R2U-Net. The following section provides the architectural details of both models.

7.3 RU-Net and R2U-Net Architectures

Inspired by the deep residual model [11], the RCNN [52], and the U-Net [32] model, we propose two models for segmentation tasks which are named RU-Net and R2U-Net. These two approaches utilize the strengths of all three recently developed deep learning models. The RCNN and its variants have already shown superior performance on object recognition tasks using different benchmarks [52,194]. The recurrent residual convolutional operations can be demonstrated mathematically according to the improved-residual networks in [196]. The operations of the Recurrent Convolutional Layers (RCL) are performed with respect to the discrete time steps that are expressed according to the RCNN [52]. Let's consider the x_l input sample in the l^{th} layer of the residual RCNN (RRCNN) block and a center pixel of a patch located at (i, j) in an input sample on the k^{th} feature map in the RCL. Additionally, let's assume the output of the network $O_{ijk}^{l}(t)$ is at the time step t. The output can be expressed as follows:

$$O_{ijk}^{l}(t) = \left(w_k^f\right)^T * x_l^{f(i,j)}(t) + (w_k^r)^T * x_l^{r(i,j)}(t-1) + b_k \tag{7.1}$$

Here $x_l^{f(i,j)}(t)$ and $x_l^{r(i,j)}(t-1)$ are the inputs to the standard convolution layers and the l^{th} RCL respectively. The w_k^f and w_k^r values are the weights of the standard convolutional layer and the RCL of the k^{th} feature map respectively, and b_k is the bias. The outputs of the RCL are fed to the standard ReLU activation function f and are expressed:

$$\mathcal{F}(x_l, w_l) = f(O_{ijk}^{l}(t)) = \max(0, O_{ijk}^{l}(t)) \tag{7.2}$$

$\mathcal{F}(x_l, w_l)$ represents the outputs from of l^{th} layer of the RCNN unit. The output of $\mathcal{F}(x_l, w_l)$ is used for down sampling and up sampling layers in the convolutional encoding and decoding units of the RU-Net model respectively. In the case of R2U-Net, the final outputs of the RCNN unit are passed

through the residual unit that is shown in Figure 7.3 (d). Let's consider that the output of the RRCNN-block to be x_{l+1}, and it can be calculated as follows:

$$x_{l+1} = x_l + \mathcal{F}(x_l, w_l) \tag{7.3}$$

Here, x_l represents the input samples of the RRCNN-block. The x_{l+1} sample is the input for the immediate succeeding subsampling or up sampling layers in the encoding and decoding convolutional units of the R2U-Net model. However, the number of feature maps and the dimensions of the feature maps for the residual units are the same as in the RRCNN-block shown in Figure 7.3 (d).

The proposed deep learning models are the building blocks of the stacked convolutional units shown in Figures 7.3 (b) and (d). There are four different architectures evaluated in this work. First, the U-Net with forward convolution layers and feature concatenation is applied as an alternative to the crop and copy method found in the primary version of U-Net [32]. The basic convolutional unit of this model is shown in Figure 3 (a). Second, the U-Net model with forward convolutional layers with residual connectivity is used, which is often called a residual U-net (or a ResU-Net) and is shown in Figure 3 (c) [198]. The third architecture is the U-Net model with forward recurrent convolutional layers as shown in Figure 3 (b), which is named RU-Net. Finally, the last architecture is the U-Net model with recurrent convolution layers with residual connectivity as shown in Figure 7.3 (d), which is named R2U-Net. The pictorial representation of the unfolded RCL layers with respect to the time step is shown in Fig 4. Here t = 2 (0 ~ 2), refers to the recurrent convolutional operation that includes one single convolution layer followed by two sub-sequential recurrent convolutional layers. In this implementation, we have applied concatenation to the feature maps from the encoding unit to the decoding unit for the RU-Net and R2U-Net models.

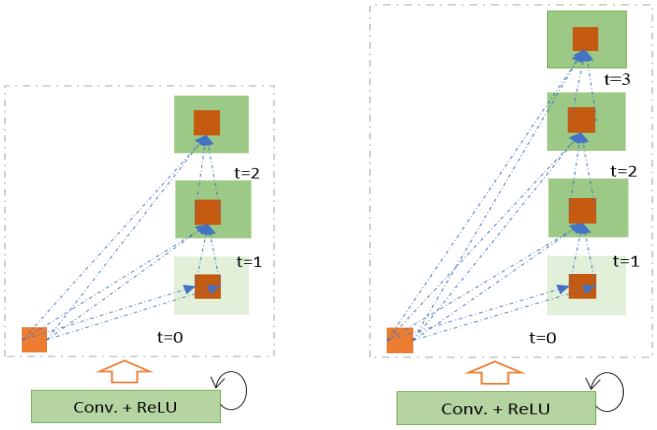


Figure 7.4: Unfolded recurrent convolutional units for t = 2 (left) and t = 3 (right).

The differences between the proposed models with respect to the U-Net model are three-fold. This architecture consists of convolutional encoding and decoding units that are the same as those used in the U-Net model. However, the RCLs (and RCLs with residual units) are used instead of regular forward convolutional layers in both the encoding and decoding units. The residual unit with RCLs helps to develop a more efficient deeper model. Second, the efficient feature accumulation method is included in the RCL units of both proposed models. The effectiveness of feature accumulation from one part of the network to the other is shown in the CNN-based segmentation approach for medical imaging. In this model, the element-wise feature summation is performed outside of the U-Net model [32]. The U-Net model only shows the benefit during the training process in the form of better convergence. However, our proposed models show benefits for both training and testing phases due to the feature accumulation inside the model. The feature accumulation with respect to different time steps ensures better and stronger feature representation. Thus, it helps extract very low-level features which are essential for segmentation tasks for different modalities of medical imaging (such as blood vessel segmentation). Third, we have removed the cropping and copying unit from the basic U-Net model and use only concatenation operations. Therefore, with all the above-mentioned changes, the proposed models are much better compared to equivalent SegNet,

U-Net, and ResU-Net models, which ensures better performance with the same or fewer number of network parameters.

There are several advantages of using the proposed architectures when compared to U-Net. The first is the efficiency in terms of the number of network parameters. The proposed RU-Net and R2U-Net architectures are designed to have the same number of network parameters when compared to U-Net and ResU-Net, and the RU-Net and R2U-Net models show better performance on segmentation tasks. The recurrent and residual operations do not increase the number of network parameters. However, they do have a significant impact on training and testing performance which is shown through an empirical evaluation with a set of experiments in the following sections [196]. This approach is also generalizable, as it can easily be applied to deep learning models based on SegNet [197], 3D-UNet [45], and V-Net [51] with improved performance for segmentation tasks.

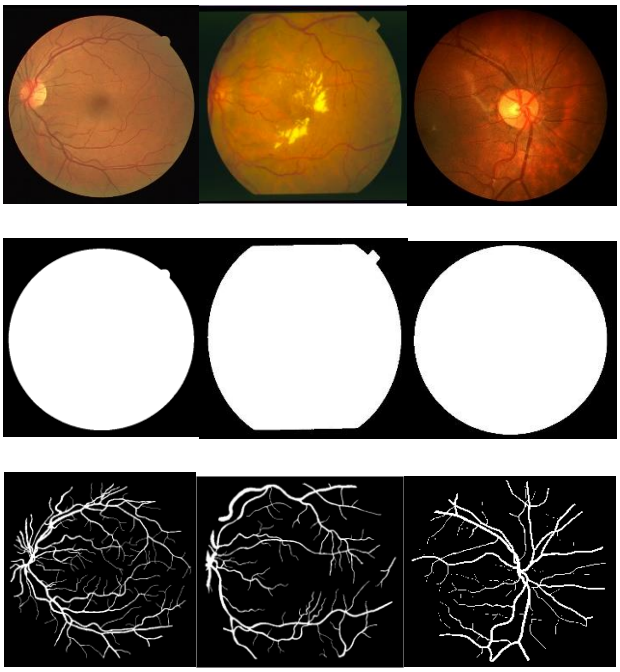


Figure 7.5: Example images from training datasets where the image in the left column was taken from the DRIVE dataset, the middle column was taken from the STARE dataset, and right column was taken from the CHASE-DB1 dataset. The first row shows the original images, the second row shows the fields of view (FOV), and the third row shows the target outputs.

Network architectures: We have conducted experiments using several different models including SegNet [197], U-Net [32], ResU-Net [198], RU-Net, and R2U-Net. These models are evaluated

with different numbers of convolutional layers in the convolutional blocks, and the numbers of layers are determined with respect to time step t. The network architectures along with the corresponding numbers of feature maps in different convolutional blocks are shown in Table 7.1. From the table, it can be clearly seen in rows 2 and 4 that the numbers of feature maps in the

Table 7.1: Architectural details, numbers of feature maps in the convolutional blocks, and the number of network parameters for Retina Blood Vessel Segmentation (RBVS), Skin Lesion Segmentation (SLS), and Lung Segmentation (LS).

Dataset	***t***	**Network architectures**	**Number of parameters (millions)**
RBVS+LS	2	1→ 16(3)→32(3)→64(3)→128(3)→64 (3)→32(3)→16(3)→1	0.841
LS	3	1→16(4)→32(4)→64(4)→128(4)→64(4) →32(4)→16(4)→1	1.037
SLS +RBVS	2	1→32(3)→64(3)→128(3)→256(3)→512(3)→256(3)→128 (3)→64(3)→32-(3)→1	13.34

convolutional blocks remain the same, however, as a convolutional layer is added in the convolutional block when t = 3, the number of network parameters increases. Feature fusion is performed with an element-wise addition operation in different residual, recurrent, and recurrent residual units. In the encoding unit of the network, each convolutional block consists of two or three RCLs, where 3×3 convolutional kernels are applied, proceeded by ReLU activation layers, followed by a batch normalization layer. For downsampling, a 2×2 max-pooling layer followed by a 1×1 convolutional layer is used between the convolutional blocks. In the decoding unit, each block consists of a convolutional transpose layer followed by two convolutional layers and a concatenation layer. The concatenation operations are performed between the features in the encoding and decoding units in the network. The features are then mapped to a single output feature map where 1×1 convolutional kernels are used with a sigmoid activation function. Finally, the segmentation region is generated with a threshold (T) which is empirically set to 0.5 in our experiment.

The architecture shown in the fourth row is used for retina blood vessel segmentation on the DRIVE dataset, as well as skin cancer segmentation. We have also implemented the SegNet model [197] with a similar architecture and a similar number of feature maps for impartial comparison in the cases of skin cancer lesions and lung segmentation. The architecture we used can be written as 1→32(3)→64(3) →128(3)→256(3)→512(3)→256(3)→128(3)→64(3)→32(3)→1 in the SegNet model for skin cancer lesion segmentation, where each convolutional block contains three convolutional layers and a batch normalization layer which requires a total of 14.94M network parameters. For lung segmentation, the architecture can be written as 1→32(3)→64(3)→128(3)→256(3)→128(3)→64(3)→32(3)→1 for the SegNet model (three convolutional layers and a batch normalization layer are used in each block) which requires a total of 1.7M network parameters.

7.4 Experimental Setup and Results

To demonstrate the performance of the RU-Net and R2U-Net models, we have tested them on three different medical imaging datasets. These include blood vessel segmentation from retina images (DRIVE, STARE, and CHASE_DB1 shown in Figure 7.5), skin cancer lesion segmentation, and lung segmentation from 2D images. For this implementation, the Keras and TensorFlow frameworks are used on a single GPU machine with 56G of RAM and an NVIDIA GEFORCE GTX-980 Ti with 6GB of memory.

7.4.1 Database summary

Blood Vessel Segmentation: we have experimented on three different popular datasets for retina blood vessel segmentation including DRIVE [215], STARE [216], and CHASE_DB1 [217]. The DRIVE dataset consists of 40 color retina images in total, of which 20 samples are used for training and the remaining 20 samples are used for testing. The size of each original image is 565×584 pixels [215]. To develop a square dataset, the images were cropped to only contain the data from columns 9 through 574, which then makes each image 565×565 pixels. In this implementation, we

considered 190,000 randomly selected patches from 20 of the images in the DRIVE dataset, where 171,000 patches were used for training, and the remaining 19,000 patches were used for validation. The size of each patch is 48×48 for all three datasets shown in Figure 7.6. The second dataset, STARE, contains 20 color images, and each image has a size of 700×605 pixels [216, 218]. Due to the small number of samples in the STARE dataset, two approaches are often applied for training and testing when using this dataset. First, training is sometimes performed with randomly selected samples from all 20 images [219].

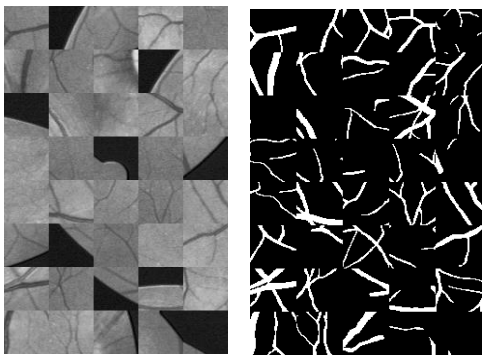


Figure 7.6: Example patches are shown in the left image, and the corresponding outputs of the patches are shown in the right image.

Another approach is the "leave-one-out" method, where in each trial one image is selected for testing, and training is conducted on the remaining 19 samples [217,220]. Therefore, there is no overlap between the training and testing samples. In this implementation, we used the "leave-one-out" approach for the STARE dataset. The CHASE_DB1 dataset contains 28 color retina images and the size of each image is 999×960 pixels [217]. The images in this dataset were collected from both left and right eyes of 14 school children. The dataset is divided into two sets where samples are selected randomly. A 20-sample set is used for training and the remaining 8 samples are used for testing.

As the dimensionality of the input data in the STARE and CHASE_DB1 datasets larger than that of the DRIVE dataset, we considered 250,000 patches in total from 20 images for both STARE and CHASE_DB1 datasets. In this case, 225,000 patches are used for training and the remaining 25,000

patches are used for validation. Since the binary FOV (which is shown in the second row in Figure 7.5) is not available for the STARE and CHASE_DB1 datasets, we generated FOV masks using a similar technique to the one described in [220]. One advantage of the patch-based approach is that the patches give the network access to local information about the pixels, which has an impact on overall prediction. Furthermore, it ensures that the classes of the input data are balanced. The input patches are randomly sampled over an entire image, which also includes the outside region of the FOV.

Skin Cancer Segmentation: this dataset is taken from the Kaggle competition on skin lesion segmentation that occurred in 2016 [221]. This dataset contains 900 images along with associated ground truth samples for training. Another set of 379 images is provided for testing. The original size of each sample was 700×900, which was rescaled to 128×128 for this implementation. The training samples include the original images, as well as corresponding target binary images containing cancer or non-cancer lesions. The target pixels are set to a value of either 255 or 0, denoting pixels inside or outside the target lesion respectively.

Lung Segmentation: Lung Nodule Analysis (LUNA)-16 competition at the Kaggle Data Science Bowl in 2017 was held to find lung lesions in 2D and 3D CT images. This dataset consists of 267 2D samples in total, each containing a sample photograph, and label image displaying correct lung segmentation [222]. For this study, 80% of the images were used for training, and the remaining 20% were used for testing. The original image size was 512×512, however, we resized the images to 256×256 pixels in this implementation.

7.4.2 Evaluation metrics

For quantitative analysis of the experimental results, several performance metrics were considered, including accuracy (AC), sensitivity (SE), specificity (SP), F1-score, Dice coefficient (DC), and Jaccard Index (JA). To do this we also use the variables True Positive (TP), True Negative (TN),

False Positive (FP), and False Negative (FN). The overall accuracy is calculated using Eq. (7.4), and sensitivity and specificity are calculated using Eq. (7.5).

$$AC = \frac{TP+TN}{TP+TN+FP+FN} \tag{7.4}$$

$$SE = \frac{TP}{TP+FN} \qquad SP = \frac{TN}{TN+FP} \tag{7.5}$$

Furthermore, DC and JA are calculated using the following Eq. (6).

$$DC = \frac{2.TP}{2.TP+FN+FP} \qquad JA = \frac{TP}{TP+FN+FP} \tag{7.6}$$

In addition, we have also conducted an experiment to determine the Dice Index (DI) loss function according to [223], and the Jaccard similarity score (JS) is represented using Eq. (7.7) as in [224]. Here GT refers to the ground truth and SR refers to the segmentation result.

$$DI(GT, SR) = 2\,\frac{|GT \cap SR|}{|GT|+|SR|} \qquad JS(GT, SR) = = \frac{|GT \cap SR|}{|GT \cup SR|} \tag{7.7}$$

The F1-Score is calculated according to the following equations:

$$F1 - score = 2 \times \frac{precision \times recall}{precision+recall} \tag{7.8}$$

where the precision and recall are expressed as:

$$precision = \frac{TP}{TP+FP} \qquad recall = \frac{TP}{TP+FN} \tag{7.9}$$

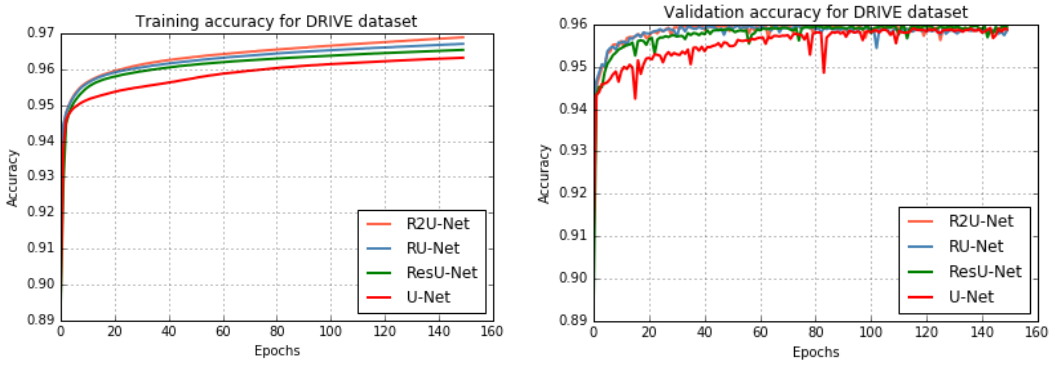


Figure 7.7: Training and validation accuracy of the proposed RU-Net and R2U-Net models compared to the ResU-Net and U-Net models for 150 epochs. Training is on the left and validation is on the right.

The area under the curve (AUC) and the receiver operating characteristics (ROC) curve are common evaluation measures for medical image segmentation tasks. In this experiment, we utilized both analytical methods to evaluate the performance of the proposed approaches and compared our results to existing state-of-the-art techniques.

Table 7.2: Experimental results of proposed approaches for retina blood vessel segmentation and comparison against other traditional and deep learning-based approaches.

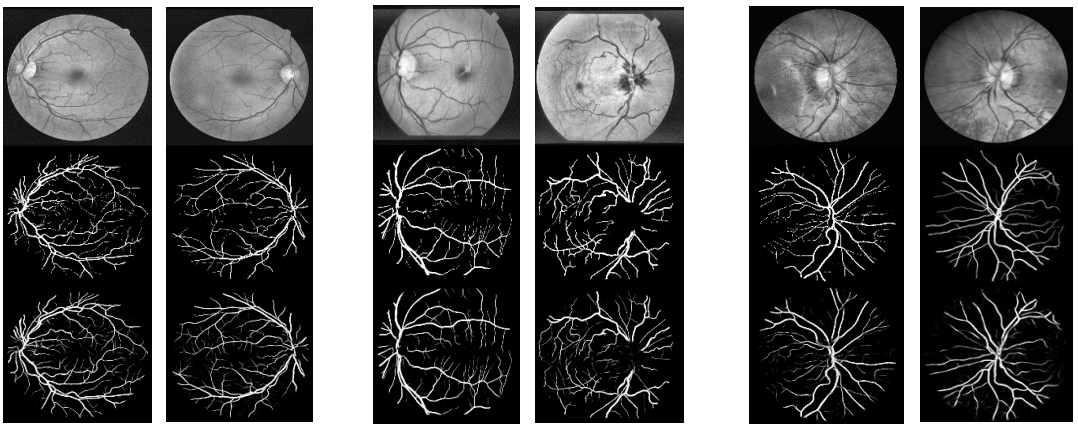
Dataset	**Methods**	**Year**	**SE**	**SP**	**AC**	**AUC**
DRIVE	Chen [219]	2014	o.7252	0.9798	0.9474	0.9648
	Azzopardi [227]	2015	0.7655	0.9704	0.9442	0.9614
	Roychowdhury[228]	2016	0.7250	0.9830	0.9520	0.9620
	Liskowsk [229]	2016	0.7763	0.9768	0.9495	0.9720
	Li [226]	2016	0.7569	0.9816	0.9527	0.9738
	Zhao [225]	2018	0.7740	0.9790	0.9580	0.9750
	U-Net (1.07M)	2018	0.7537	0.9820	0.9531	0.9755
	ResU-Net(1.07M)	2018	0.7726	0.9820	0.9553	0.9779
	RU-Net(1.07M)	**2018**	0.7751	0.9816	0.9556	0.9782
	R2U-Net(1.07M)	**2018**	0.7792	0.9813	0.9556	0.9784
	R2U-Net (13.34M)	**2018**	0.7661	0.9807	**0.9613**	0.9793
STARE	Marin et al. [230]	2011	0.6940	0.9770	0.9520	0.9820
	Fraz [231]	2012	0.7548	0.9763	0.9534	0.9768
	Roychowdhury[228]	2016	0.7720	0.9730	0.9510	0.9690
	Liskowsk [229]	2016	0.7867	0.9754	0.9566	0.9785
	Li [226]	2016	0.7726	0.9844	0.9628	0.9879
	Zhao [225]	2018	0.7880	0.9760	0.9570	0.9590
	U-Net (1.07M)	2018	0.8270	0.9842	0.9690	0.9898
	ResU-Net(1.07M)	2018	0.8203	0.9856	0.9700	0.9904
	RU-Net(1.07M)	**2018**	0.8108	0.9871	0.9706	0.9909
	R2U-Net(1.07M)	**2018**	0.8298	0.9862	**0.9712**	0.9914
CHASE_DB1	Fraz [231]	2012	0.7224	0.9711	0.9469	0.9712
	Fraz [232]	2014	-	-	0.9524	0.9760
	Azzopardi [227]	2015	0.7655	0.9704	0.9442	0.9614
	Roychowdhury[228]	2016	0.7201	0.9824	0.9530	0.9532
	Li [227]	2016	0.7507	0.9793	0.9581	0.9793
	U-Net (1.07M)	2018	0.8288	0.9701	0.9578	0.9772
	ResU-Net(1.07M)	2018	0.7726	0.9820	0.9553	0.9779
	RU-Net(1.07M)	**2018**	0.7459	0.9836	0.9622	0.9803
	R2U-Net(1.07M)	**2018**	0.7756	0.9820	**0.9634**	**0.9815**

7.4.3 Experimental results

7.4.3.1 Retina blood vessel segmentation

Due to the data scarcity of retina blood vessel segmentation datasets, the patch-based approach is used during the training and testing phases. We used a random initialization method, and a Stochastic Gradient Descent (SGD) optimization approach with categorical cross entropy loss, a batch size 32, and 150 epochs in this implementation. Results on DRIVE dataset: Figure 7.7 shows

the training and validation accuracy when using the DRIVE dataset. The proposed R2U-Net and RU-Net models provide better performance during both the training and validation phase when compared to the U-Net and ResU-Net models. Quantitative results are achieved with the four different models using the DRIVE dataset, and the results are shown in Table 7.2. The overall accuracy and AUC are considered when comparing the performance of the proposed methods in most cases. The results we have achieved with the proposed models with 0.841M network parameters (Table 7.1, second row) are higher than those obtained when using the state-of-the-art approaches in most cases. However, to compare against the most recently proposed method [225], a deeper R2U-Net is evaluated with 13.34M network parameters (Table 7.1, fourth row) and that shows the highest accuracy (0.9613) and a better AUC of 0.979. Most importantly, we can observe that the proposed RU-Net and R2U-Net models provide better performance in terms of accuracy and AUC compared to the U-Net and RU-Net models.



(a) DRIVE (b) STARE (c) CHASE-DB1

Figure 7.8: Experimental outputs for three different datasets for retina blood vessel segmentation using R2UNet. The first row shows input images in grayscale, the second row shows the ground truth, and the third row shows the experimental outputs. The images correspond to the (a) DRIVE, (b) STARE, and (c) CHASE_DB1 datasets.

The precise segmentation results achieved with the proposed R2U-Net model are shown in Figure 7.8 (a). STARE dataset: The quantitative results when using the STARE dataset, along with a comparison to existing methods, are shown in Table 2. In 2016, a cross-modality learning approach

was proposed by Li et.al. [226] and has reported an accuracy of approximately 0.9628 for STARE dataset, which was previously the highest recorded result. Recently in 2018, Zhao et. al. proposed method with a Weighted Symmetry Filter (WSF) and showed an accuracy of 0.9570 [225]. In this work, we used the "leave-one-out" method and report the average results of five different trials. We have achieved an accuracy of 0.9712 with the R2U-Net model for the STARE dataset, which is 0.0084 and 0.0142 better than the results obtained when using the methods proposed by Li and Zhao respectively. In addition, the RU-Net and R2U-Net models outperform the U-Net and ResU-Net models in this experiment. The qualitative results of R2U-Net when using the STARE dataset is shown in Figure 7.8 (b).

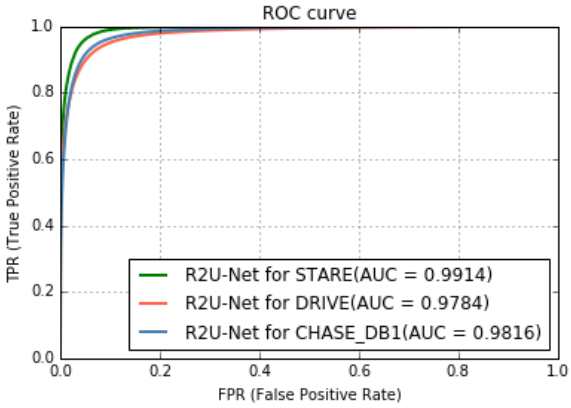


Figure 7.9: AUC for retina blood vessel segmentation for the best performance achieved with R2U-Net on three different datasets.

CHASE_DB1 dataset: For quantitative analysis, the results are given in Table I. From the table, it can be seen that the RU-Net and R2U-Net models provide better performance against the U-Net and ResU-Net models when applying the CHASE-DB1 dataset. In addition, the proposed methods are compared against the recently proposed approaches for blood vessel segmentation using the CHASE_DB1 dataset. In 2016, Li et. al. [226] proposed an approach with cross-modality learning and achieved an accuracy 0.9581. However, we have achieved an accuracy of approximately 0.9634 with the R2U-Net model, which is about a 0.0053 improvement compared to the result in [226]. The precise segmentation results with the proposed R2U-Net model on the CHASE_DB1

dataset are shown in Figure 7.8 (c). The ROC for the highest AUCs for the R2U-Net (with 1.07M network parameters) model on each of the three-retina blood vessel segmentation datasets is shown in Figure 7.9.

7.4.3.2 Skin Cancer Lesion Segmentation

In this implementation, this dataset is preprocessed with mean subtraction and was normalized according to the standard deviation. We used the ADAM optimization technique with a learning rate of 2×10^{-4} and binary cross entropy loss. In addition, we also calculated MSE during the training and validation phase. In this case, 10% of the samples are used for validation during training with a batch size of 32 and 150 epochs. The training accuracy of the proposed R2U-Net and RU-Net models was compared with that of the ResU-Net and U-Net models for an end-to-end image-based segmentation approach. The training and the validation accuracy for all four models are shown in Figure 7.10. In both cases, the proposed RU-Net and R2U-Net models show better performance when compared with the equivalent U-Net and ResU-Net models. This clearly demonstrates the robustness of the learning phase of the proposed models for end-to-end image-based segmentation tasks.

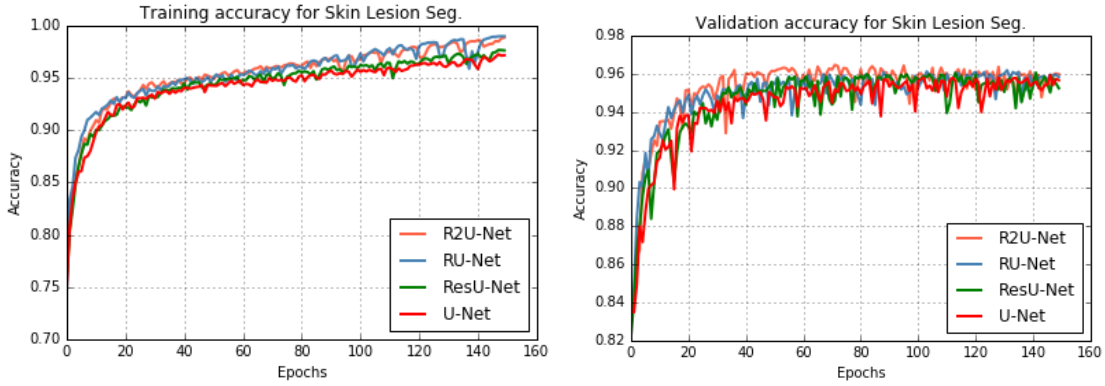


Figure 7.10: Training and validation accuracy of R2U-Net, RU-Net, ResU-Net, and U-Net for skin lesion segmentation. Training accuracy is one the left and validation accuracy is on the right.

The quantitative results of this experiment were compared against existing methods as shown in Table 7.3. We have evaluated the proposed RU-Net and R2U-Net models with respect to the time step t = 2 in the RCL unit. The time step value t = 2 means that the RCL unit consists of one forward convolution followed by two RCLs.

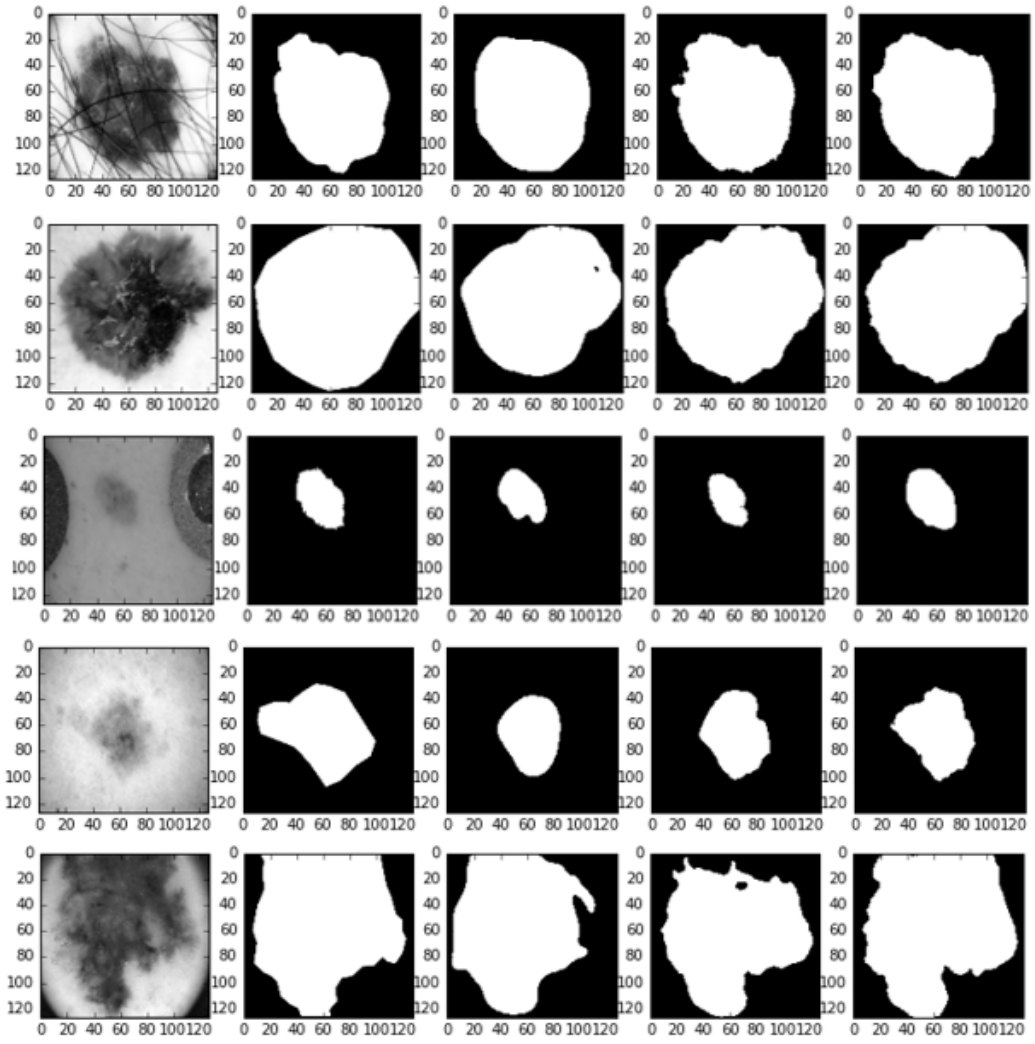
Table 7.3: Experimental results of the proposed approaches for skin cancer lesion segmentation and comparison against other traditional and deep learning-based approaches.

Methods	**Year**	**SE**	**SP**	**AC**	**AUC**	**DC**	**JA**
ISIC-2016 [221]	2016	0.910	0.965	**0.953**			08430
Conv. classifier VGG-16 [234]	2017	0.533	-	0.613	0.6420	-	-
Conv. classifier Inception-v3[234]	2017	0.760	-	0.693	0.7390	-	-
VGG-16 [233]	2017	0.796	0.945	0.903	-	0.794	0.707
GoogleNet [233]	2017	0.901	0.916	0.916		0.848	0.776
FCRN-38 [233]	2017	0.882	0.932	0.929		0.856	0.785
FCRN-50 [233]	2017	0.911	0.957	**0.949**	-	0.897	0.829
FCRN-101[233]	2017	0.903	0.903	0.937	-	0.872	0.803
SegNet [197]	2018	0.9395	0.9222	0.9263	0.9308	0.9502	0.9052
U-Net (16.67M)	2018	0.9457	0.9307	0.9343	0.9324	0.9554	0.9148
ResU-Net (16.67M)	2018	0.9287	0.9479	0.9432	0.9378	0.9608	0.9245
RecU-Net (16.67M)	2018	0.9477	0.9443	0.9458	0.9383	0.9624	0.9273
R2U-Net (16.67M)	2018	**0.9224**	0.9545	**0.9472**	**0.9430**	**0.9627**	**0.9278**

Note: the results of VGG-16 and GoogleNet are taken from [233].

We compared the proposed approaches against recently published results using performance metrics including sensitivity, specificity, accuracy, AUC, and DC. The proposed R2U-Net model provides a testing accuracy 0.9472 with a higher AUC, which is 0.9430. Furthermore, the JA and DC are calculated for all models, and the R2U-Net model provides 0.9278 for JA, and 0.9627 for the DC for skin lesion segmentation. Although we are in the third position in terms of accuracy compared to ISIC-2016 [221] (highest) and FCRN-50 [233] (second highest), the proposed R2U-Net models show better performance in term of the DC and JA. These results were achieved with

an R2U-Net model with 34 layers that contains approximately 13.34M network parameters. The architectural detail is shown in Table 1. However, the accuracy of the proposed RU-Net and R2U-Net models is still higher when compared to the FCRN-38 networks [233]. In addition, the work presented in [221,233] was evaluated with the VGG-16 and Inception-V3 models for skin cancer lesion segmentation.



Inputs Ground truth SegNet U-Net R2U-Net

Figure 7.11. Illustration of qualitative assessment of the proposed R2U-Net for the skin cancer segmentation task. The first column shows the input sample, the second column shows the ground truth, the third column shows the outputs from the SegNet [197] model, the fourth column shows the outputs from the U-Net [32] model, and the fifth column shows the results of the proposed R2U-Net model.

These models contain approximately 138M and 23M network parameters respectively. Furthermore, the RU-Net and R2U-Net models show higher accuracy and AUC compared to the VGG-16 [233] and GoolgeNet models [233]. In most cases, the RU-Net and R2U-Net models show better performance against equivalent SegNet [197], U-Net [32], and ResU-Net [198] models for skin lesion segmentation.

Table 7.4: The experimental results of the proposed RU-Net and R2U-Net approaches for lung segmentation and comparison against the SegNet, U-Net, ResU-Net models for $t = 2$ and $t = 3$.

Methods	**Year**	**SE**	**SP**	**JSC**	**F1-Score**	**AC**	**AUC**	**DI**
SegNet (1.02M) [197]	2018	0.9766	0.9791	0.9784	0.9575	0.9784	0.9778	0.9652
SegNet (1.752M) [197]	2018	0.9757	0.9931	0.9887	0.9777	0.9887	0.9844	0.9754
U-Net (t=2)	2018	0.8645	0.9929	0.9635	0.9156	0.9635	0.9287	0.9780
ResU-Net(t=2)	2018	0.9781	0.9975	0.9781	0.9522	0.9781	0.9568	0.9792
RU-Net (t=2)	2018	0.9747	0.9962	0.9911	0.9811	0.9911	0.9855	0.9831
R2U-Net (t=2)	2018	0.9861	0.9940	0.9922	0.9830	0.9922	0.9901	0.9857
U-Net (t=3)	2018	0.9816	0.9945	0.9916	0.9822	0.9916	0.9881	0.9801
ResU-Net(t=3)	2018	0.9838	0.9951	0.9926	0.9833	0.9926	0.9895	0.9825
RU-Net (t=3)	2018	0.9875	0.9959	0.9942	0.9872	0.9942	0.9918	0.9863
R2U-Net (t=3)	2018	0.9912	0.9952	0.9943	0.9879	**0.9944**	**0.9933**	**0.9880**

Some qualitative outputs of the SegNet, U-Net, and R2U-Net models for skin cancer lesion segmentation are shown for visual comparison in Figure 7.11. In most cases, the target lesions are segmented accurately with a similar shape in ground truth. However, if we closely observe the outputs in the first, second, and fourth rows of images in Figure 7.11, it can be clearly distinguished that the proposed R2U-Net model provides a very similar output shape to the ground truth when compared to the outputs of the SegNet and U-Net models. If we observe the third row of images in Figure 7.11, it can be clearly seen that the input image contains three lesions. One is a target lesion, and the other brighter lesions are not targets. The R2U-Net model segments the desired part of the image more accurately when compared to the SegNet and U-Net models. Finally, the fifth row

clearly demonstrates that the R2U-Net model provides a very similar shape to the ground truth, which is a much better representation than those obtained from the SegNet and U-Net models. Thus, it can be stated that the R2U-Net model is more capable and robust for skin cancer lesion segmentation.

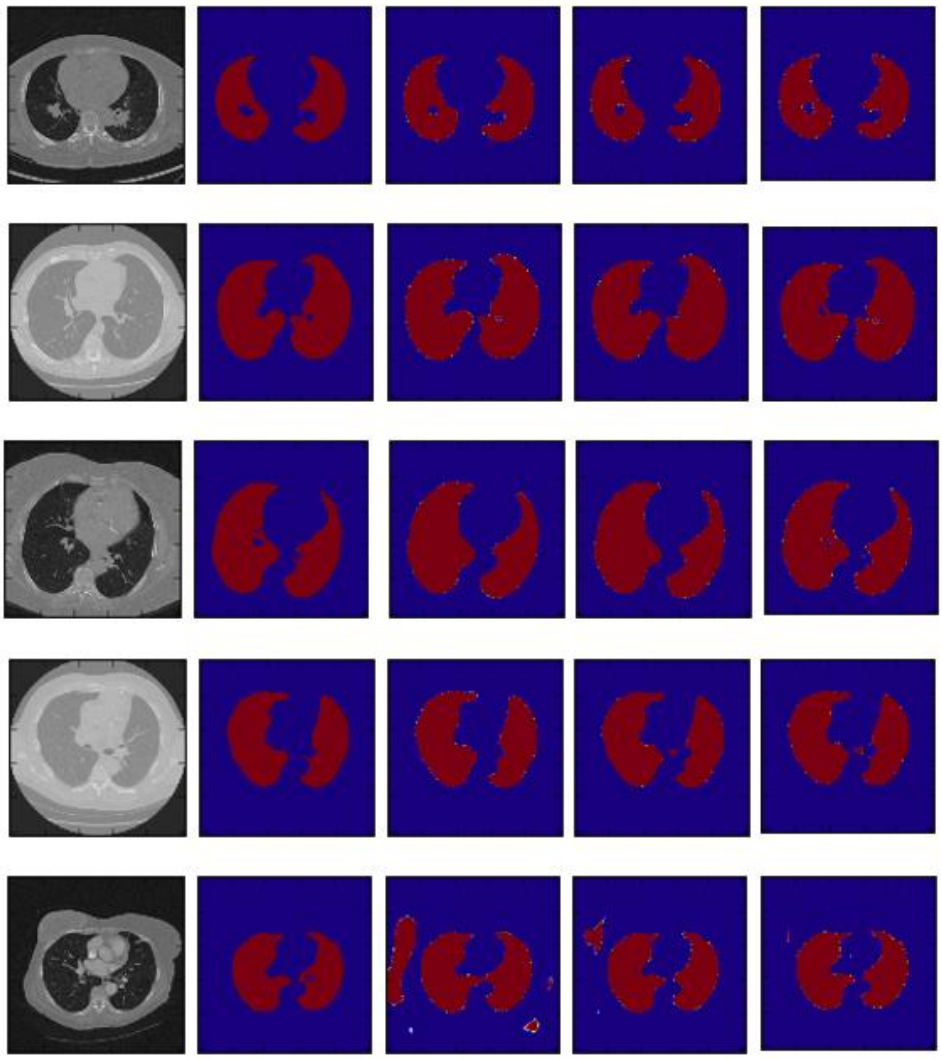


Figure 7.12: The experimental results for lung segmentation where the first column shows the inputs, the second column shows the ground truth, the third column shows the outputs of SegNet [10], the fourth column for the outputs of U-Net [12], and a fifth column for the outputs of R2U-Net.

7.4.3.3 Lung Segmentation

Lung segmentation is very important for analyzing lung-related diseases, and it can be applied to lung cancer segmentation and lung pattern classification for identifying other problems. In this experiment, the ADAM optimizer is used with a learning rate of 2×10^{-4}. We used DI loss function according to Eq. (7.7). In this case, 10% of the samples were used for validation with a batch size of 16 for 150 epochs. Table 7.4 shows the summary of how well the proposed models performed against the equivalent SegNet[197], U-Net, and ResU-Net models[198]. In terms of accuracy, the proposed R2U-Net model showed 0.26 and 0.55 percent better testing accuracy compared to the equivalent SegNet [197] and U-Net[32] models respectively. In addition, the R2U-Net model provided 0.18 percent better accuracy against the ResU-Net model with the same number of network parameters. Qualitative results are shown in Figure 7.12, where the first column shows the input samples, the second column represents ground truth, and the third, fourth, and fifth columns show the outputs of the SegNet [197], U-Net [32], and R2U-Net models respectively. It can be visualized that the R2U-Net shows better segmentation results with internal details that are very similar to those displayed in the ground data. If we observe the input, ground truth, and output of the different approaches in the first and second rows, the outputs of the proposed approaches show better segmentation with more accurate internal details. In the third row, the R2U-Net model clearly defines the inside hole in the left lung, whereas the SegNet [197] and U-Net [32] models do not capture this detail. The last row of images in Figure 7.12 shows that the SegNet [197] and U-Net models provide outputs that incorrectly capture parts of the image that are outside of the lesion. On contrary, the R2U-Net model provides a much more accurate segmentation result. Many models struggle to define the class boundary properly during segmentation tasks [234]. The outputs in Figure 7.12 are provided as heat maps which show the sharpness of the segmentation borders. These outputs show that ground truth tends to have a sharper boundary when compared to the model outputs. The ROC with AUCs is shown in Figure 7.13. The highest AUC is achieved is that of the proposed R2U-Net model.

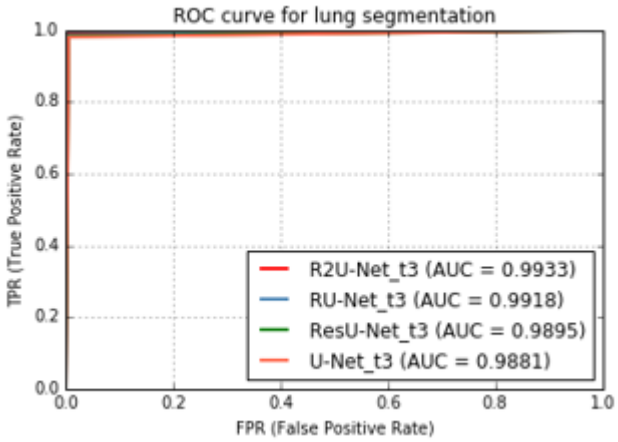
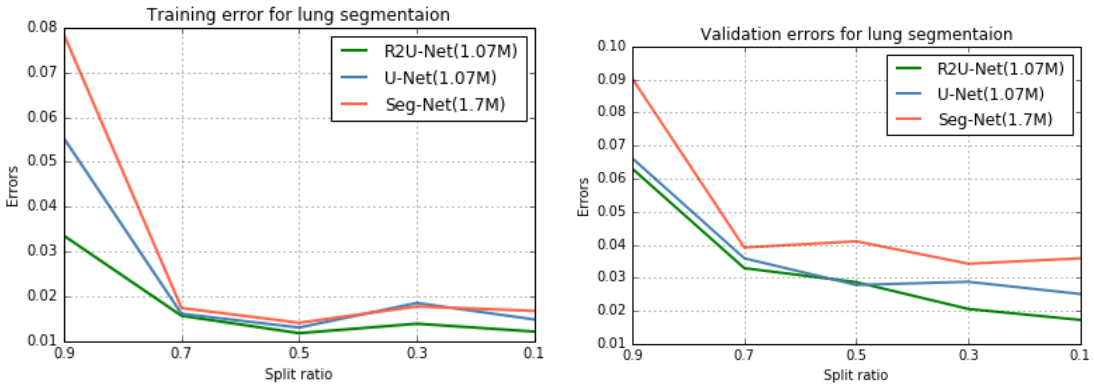


Figure 7.13: ROC curve for lung segmentation for four different models where t = 3.

In this implementation, we evaluated both of the proposed models for patch-based modeling of retina blood vessel segmentation, as well as end-to-end image-based methods for skin and lung lesion segmentation. In both cases, the proposed models outperform existing state-of-the-art methods including SegNet [197], U-Net [13], ResU-Net [198] and FCRN-38 [233] in terms of AUC and accuracy on all three datasets. Thus, the quantitative and qualitative results clearly demonstrate the effectiveness of the proposed approach for segmentation tasks.



(a) (b)

Figure 7.14: The performance of three different models (SegNet, U-Net, and R2U-Net) for different numbers of training and validation samples where displays (a) the training DI coefficient errors (1-DI) and (b) displays validation DI coefficient errors for five different trials.

7.4.4 Analysis

The trade-off between a number of training samples and accuracy: To further investigate the performance of the proposed R2U-Net model, the trade-off between the number of training samples versus performance is investigated for the lung segmentation dataset. We considered the U-Net and R2U-Net models with t = 3, and these models contained 1.07M network parameters. In the case of SegNet [197], we considered a similar architecture that was proposed in [191] with 1.7M network parameters. At the beginning of the experiment, the entire dataset was divided into two sets where 80% of the samples were used for training and validation, and the remaining 20% of the samples were used for testing during each trail. During this experiment, we used different split ratios of [0.9, 0.7, 0.5, 0.3, and 0.1] where the number of training samples was increased, and the number of validation samples was decreased for each successive trail. For example, a split ratio of 0.9 means that only 10 percent of the samples are used for training and the remaining 90% of the samples are used for validation. Likewise, a split ratio of 0.7 means that only 30% of the samples are used for training and the remaining 70% of the samples are used for validation. Figures 7.14 (a) and (b) show the training and validation DI coefficient errors (1-DI) with respect to the number of training and validation samples. In each trial, we considered 150 epochs, and the errors presented are the average training and validation errors of the last twenty epochs.

These figures show that the proposed R2U-Net model shows the lowest training and validation error for all of the tested split ratios, except for result where the split ratio is equal to 0.5 for the validation case. In this case, the error for the R2U-Net model is only slightly greater than that of the U-Net model. These results clearly demonstrate that the R2U-Net model is a more capable tool when used to extract, represent, and learn features during the training phase, which ultimately helps to ensure better performance. In each trial, we have tested the models with the remaining 20% of the samples and the testing errors are shown in Figure 7.15. The R2U-Net model shows the lowest error for almost all trials relative to the error obtained from the SegNet [197] and U-Net [32] models.

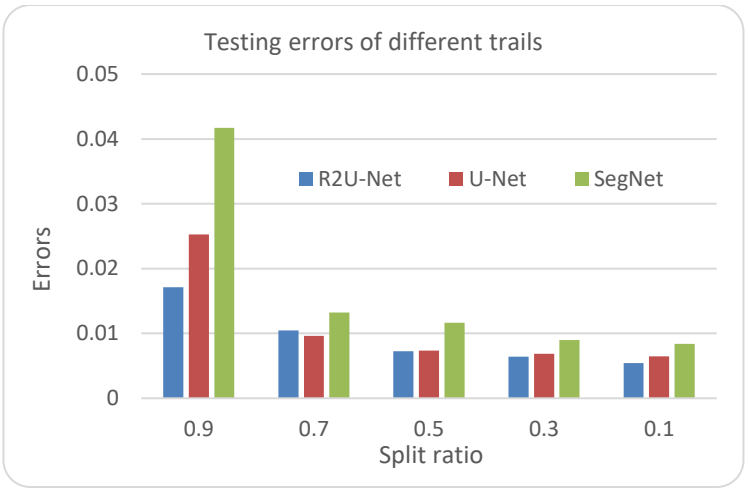


Figure 7.15: Testing errors of the R2U-Net, SegNet, and U-Net models for different split ratios for the lung segmentation application.

Network parameters versus accuracy: In our experiments, the U-Net, ResU-Net, RU-Net, and R2U-Net models were utilized with the following architecture: 1→16→32→64→128→64→32→16→1 for retina blood vessel segmentation and lung segmentation. In the case of the retina blood vessel segmentation, we used a time step of t = 2. This same architecture was tested for lung lesion segmentation for both t = 2 and t = 3. Even though the number of network parameters slightly increased with respect to the time step in the recurrent convolution layer, improved performance was still observed, as seen in the last rows of Table 7.4.

Table 7.5: Computational time for processing an entire image during the testing phase.

Dataset		**Time (seconds / sample)**
Blood vessel segmentation	DRIVE	6.42
	STARE	8.66
	CHASE_DB1	2.84
Skin cancer segmentation		0.32
Lung segmentation		1.15

Furthermore, we implemented an equivalent SegNet [197] model which required 1.73M and 14.94M network parameters respectively. For skin cancer lesion and lung segmentation, the

proposed models show better performance against SegNet [197] when using both 1.07M and 13.34M network parameters, which is around 0.7M and 2.66M less when compared to SegNet [197]. Thus, it can be stated that our model provides better performance with the same or fewer number of network parameters compared to the SegNet, U-Net and ResU-Net model. Thus, our proposed model possesses significant advantages in terms of memory and processing time.

7.4.5 Computational time

The computational time to segment per sample in the testing phase is shown in Table 7.5 for all three datasets. The processing times during the testing phase for the STARE, CHASE_DB, and DRIVE datasets were 6.42, 8.66, and 2.84 seconds per sample respectively. According to [56], it took around 90 seconds on average to segment an entire image (which is equivalent to a few thousand image patches). Alternatively, the proposed R2U-Net approach takes around 6 seconds per sample, which is an acceptable rate in a clinical use scenario. In addition, when executing skin cancer segmentation and lung segmentation entire images could be segments in 0.32 and 1.145 seconds respectively.

7.5 Conclusion and Future Works

In this paper, we proposed an extension of the U-Net architecture using Recurrent Convolutional Neural Networks and Recurrent Residual Convolutional Neural Networks. The proposed models are called “RU-Net” and “R2U-Net” respectively. These models were evaluated using three different applications in the field of medical imaging including retina blood vessel segmentation, skin cancer lesion segmentation, and lung segmentation. The experimental results demonstrate that the proposed RU-Net and R2U-Net models show better performance in most of the cases for segmentation tasks with the same number of network parameters when compared to existing methods including the SegNet, U-Net, and residual U-Net (or ResU-Net) models on all three datasets. The quantitative and qualitative results, as well as a trade-off between the number of

training samples versus performance, show that the proposed RU-Net and R2U-Net models are more capable of learning during training, which ultimately shows better testing performance. In the future, we would like to explore the same architecture with a novel feature fusion strategy in the encoding and decoding units.

CHAPTER 8

BLOOD CELL CLASSIFICATION WITH IRRCNN

Deep Learning (DL) approaches have been explored in different modalities of biomedical image analysis, and they provide superior performance against alternative machine learning approaches. In this paper, we have evaluated the performance of a deep learning model called the Inception Recurrent Residual Convolutional Neural Network (IRRCNN) for White Blood Cell (WBC) and Red Blood Cell (RBC) classification. We have tested the performance of the IRRCNN approach on two publicly available blood cell datasets for both RBC and WBC classification obtained from the Yale School of medicine and CellaVision respectively. The experimental results show almost 100% recognition accuracy for the WBC dataset and 99.94% testing accuracy for RBC classification. This is approximately a 1.4% and 2.35% improvement when compared to existing deep learning-based approaches.

8.1 Introduction

Pathological image analysis for blood cell recognition has a significant impact on the diagnosis process. The examination results of the Complete Blood Count (CBC) have a wide range of applications in hematic pathologies. The CBC contains the total number of leukocytes (WBC) and erythrocytes (RBC) in the blood [235,236]. The shape, elasticity, and adhesion properties of RBCs are changed due to the lack of oxygen when HbS molecules polymerize inside the RBCs. The diseases are determined based on the number of infected blood cells. For example, some kinds of cancer and leukemia diseases can be diagnosed based on the results of classification and count of red and white blood cells. The WBCs are responsible for fighting against any kind of infection, and

an elevated WBC count may be caused by problems including inflammation and stress. The normal blood cell contains five types of WBCs according to the following distribution: neutrophils (40~78%), lymphocytes (25~33%), monocytes (2~8%), eosinophils (1~4%), and basophil granulocytes (0~2%). The normal blood cell also contains plasma cells (0.2~2.8%) [237]. Another type of illness known as Sickle Cell Disease (SCD), also called sickle cell anemia, is an inherited disease due to an RBC disorder associated with abnormal hemoglobin S (HbS) [238,239]. A lot of pathological information can be derived from the size and shape of an RBC. Some diseases are determined based on the ratio between the number of RBCs and the total number of blood cells in a sample. SCD patients are prone to several complicated problems including stroke, AIDS, renal tumor, cancer, and organ damage over time. A recent study shows that around 3.2 million people had SCD as of 2013. Additionally, 43 million possess the sickle cell trait, which resulted in approximately 176,000 deaths in 2013 (mostly of African origin) [240].

For CBC analysis, microscopic imaging is commonly used in medical laboratories. The examination of microscopic blood imaging provides detailed diagnosis information, which helps to determine patient health status. Pathologists are experts who provide definitive disease diagnoses to guide patient treatments. Primarily, the pathologists review a set of microscopy images to determine a problem. However, pathologists experience limitations including lack of standardization and diagnostic errors. Furthermore, the significant workload is placed upon the pathologists, as they must evaluate thousands of cells or hundreds of images per day.

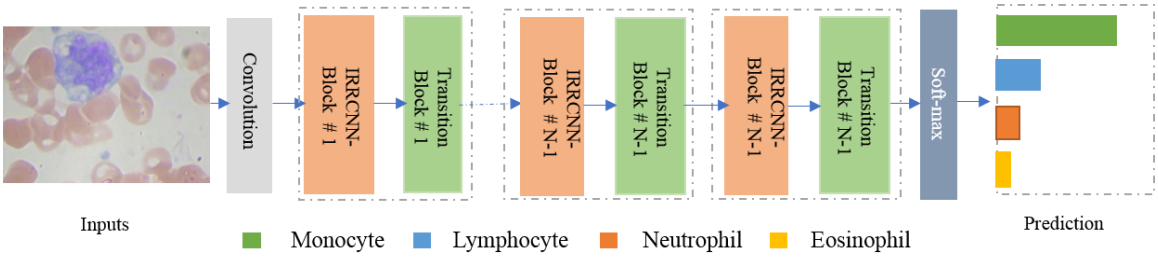


Figure 8.1: IRRCNN model and transition units for WBC classification with four classes.

A computer-based system could aid in providing a much more efficient image evaluation process, as the Region of Interest (ROI) within image sets could be quickly identified. This would significantly speed up a pathologist's practice. Many studies in digital pathology currently show that traditional image processing and machine learning techniques are applied for blood cell identification. These procedures often involve the following steps: (1) image segmentation, (2) feature extraction (shape, color, size, etc.) and some basic statistics including mean and variance of the distance between the nucleus and the cell walls, (3) traditional machine learning classification such as SVM, PCA, neural network (NN), or random forest (RF). For example, an automatic blood cell counter with flow cytometry is proposed using a blood flow detector [240]. This technique has been used in medical laboratories for the last few years. In some cases, very expensive chemicals are used to classify a few classes of blood cells [240]. However, the recent development of Deep Learning (DL) [1] and advanced medical imaging techniques could provide effective tools to monitor the WBC and RBC of a patient, which could help ensure better treatment and life [235]. In this paper, we have explored an advanced deep learning approach known as the Inception Recurrent Residual Convolutional Neural Network (IRRCNN) model for WBC and RBC classification. The complete IRRCNN model for WBC classification is shown in Figure 8.1. The contributions of this work can be summarized as follows:

- The IRRCNN approach is applied to the classification of WBCs and RBCs.
- The experiments are conducted on two different datasets containing WBCs and RBCs to classify four and ten types of blood cells respectively, where most of the previous methods are evaluated based on a single of blood cell (white or red) classification problem.
- This work is compared against recently proposed state-of-the-art methods and shows superior performance against equivalent models with the same or fewer number of network parameters.

8.2 Related Works

Several research projects have been conducted on the automated analysis of different modalities of medical imaging, including blood cell segmentation, classification, and detection. A lot of progress has been made in this field due to the advancement of artificial intelligence (deep learning), image analysis, and pattern recognition techniques for blood cell analysis [235]. Before the deep learning revolution, traditional image processing and machine learning approaches were most commonly explored to accomplish this goal. These approaches generally deal with features; thus better feature extraction methods are required before data is sent to a classifier [241,242]. A semi-supervised approach was proposed in 2013 where a Self-Dual Multiscale Morphological Toggle (SMMT) was used during pre-processing along with watershed transformation and level sets for segmentation. A K-Nearest Neighbor (KNN) based technique was proposed for clustering five types of WBCs by considering shape features and reported 78% performance accuracy. The classification rate of the domain expert was approximately 85% [243].

A neural network-based WBC classification approach was proposed in 2014, where the color characteristics of the pixels of the nucleus in a HIS color space were utilized. This approach was evaluated on 450 samples and achieved around 99.11% classification performance [244]. However, the first WBC classification approach that used Convolutional Neural Networks (CNN) was proposed by Mehdi in 2013. This approach achieved an 85% classification accuracy when using five classes, which provided a classification improvement for this task when compared to traditional machine learning approaches such as SVM and Kernel PCA (KPCA) [245]. Other than WBC classification, work has also been presented on RBC classification. A Hep-2 cell image classification using a CNN was proposed in 2014. To improve the performance of the Hep-2 cell classification problem, the deep CNN technique was applied, and it achieved significantly better classification accuracy. In this work, a dataset was evaluated based on four different patients and considered different lighting conditions [246,247]. RBC classification was also proposed by the

Yale School of Medicine, and they released a standard dataset along with a technical report where they stated around 97% testing accuracy with VGG like deep learning models [248].

In 2017, a deep convolutional neural network (DCNN) based approach was proposed for RBC classification of sickle cell anemia, which is named the RBC-dCNN. The entire system was implemented with following steps: (1) the preprocessing step where the RBC region is extracted, (2) the RBC patch normalization, and (3) the RBC-dCNN is applied for classification of the RBC. This approach was evaluated on over 7,000 samples collected from 8 different patients using a 5-fold cross-validation method for both oxygenated and deoxygenated RBCs. They used around 5,500 samples for training and around 1,500 samples for evaluation. This model has been explored with different learning rates and different numbers of iterations. They reported around 91.01% and 89.28% average training accuracy and testing accuracy for 5 classes respectively. They stated approximately 89.69% and 87.50% average training and validation accuracy respectively on experiments with 8 classes [249].

Recently, another work was published on microscopic blood smear segmentation and classification with a CNN for feature extraction. The Extreme Learning Machine (ELM) was applied to features for classification. These techniques were evaluated on the publicly available ALL-IDB dataset for blood cell segmentation and classification. This approach provided around 98.12% and 98.16% testing accuracy for RBCs and WBCs respectively [250]. In our implementation in this work, we used the IRRCNN model for both WBC and RBC classification.

8.3 IRRCNN Model

Several advanced deep learning approaches have been proposed, and they have shown state-of-the-art performance in different modalities of computer vision and medical imaging applications in the last few years [1]. The Inception Recurrent Residual Convolutional Neural Network (IRRCNN) [52] is one of many, which is an improved hybrid DCNN architecture based on inception [24], residual networks [11], and the RCNN architecture [52,196]. The main advantage of this model is

that it provides better recognition performance using the same or fewer number of network parameters when compared to alternative equivalent deep learning approaches including inception [24], the RCNN [52], and the residual network [11]. In this model, inception-residual units are utilized with respect to the Inception-v4 model [24]. The IRRCNN has been compared against equivalent inception-residual networks and shows better performance [196]. The IRRCNN model uses stacked Inception Recurrent Residual Units (IRRU) and transition units. The entire model is shown in Figure 8.1, which shows that the model consists of several convolution layers, IRRUs, transition blocks, and a softmax at the output layer.

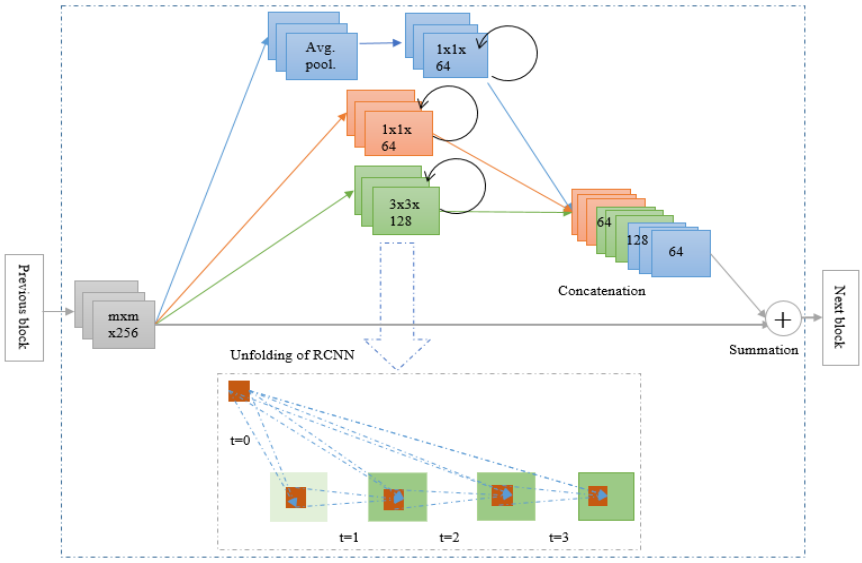


Figure 8.2: Inception Recurrent Residual Units (IRRUs) consisting of the inception unit at the top, which contains recurrent convolutional layers that are merged by concatenation, and the residual units. The summation of the input features with the outputs of the inception unit can be seen at the end of the block.

The most important unit in the IRRCNN architecture is the IRRU which includes Recurrent Convolutional Layers (RCLs), inception units, and residual units. The IRRU is shown in Figure 8.2. The recurrent convolution operations are performed with respect to the different sized kernels in the inception unit. In recurrent layers, the outputs at the present time step are then used as inputs for the next time step. The operations are performed with respect to the time steps that are considered. For example, where t = 2 (0~2) means that one feed forward convolution and 2 RCLs

are included in IRRU. The operation of the RCLs with respect to different time steps (t = 2 (0~2) and t = 3 (0~3)) is shown in Figure 8.3. Due to the residual connectivity in the IRRU, the input and output dimensions do not change, as an accumulation of feature maps is performed with respect to the time steps. Thus, better feature representation is ensured, which provides a significant improvement in recognition accuracy while using the same number of network parameters. Mathematical details about the IRRCNN model are discussed in [196,251]. The number of feature maps and the dimensions of the feature maps for the residual units is the same as in the IRRCNN unit shown in Figure 8.3. Batch normalization is applied to the outputs of the IRRU [21]. Eventually, the outputs of this IRRU are fed to the inputs of the immediate next transition unit.

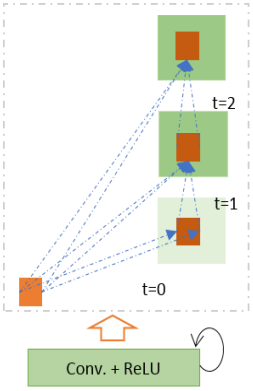


Figure 8.3: Unfolded recurrent convolutional layer for t = 2.

In the transition unit, different operations including convolution, pooling, and dropout are performed depending on the placement of the transition unit in the model. The inception units are included in the transition unit. The down-sampling operations are performed in the transition units where we perform max-pooling operations with a 3×3 patch and a 2×2 stride. The non-overlapping max-pooling operation has a negative impact on model regularization, therefore we used overlapped max-pooling for regularizing the network, which is very important when training a deep network architecture [148]. The late use of a pooling layer helps to increase the non-linearity of the features in the network, as this results in higher dimensional feature maps being passed through

the convolution layers in the network. We have applied two special pooling layers in the model with three IRRCNN units and a transition unit for this implementation.

We used 1×1 and 3×3 convolution filters in this implementation, as inspired by the NiN [19] and Squeeze Net [148] models. This helps to keep the number of network parameters at a minimum. We used a 0.5 dropout after each convolution layer in the transition block. Finally, we used a softmax or normalized exponential function layer at the end of the architecture. For input sample x, weight vector W, and K distinct linear functions, the softmax operation can be defined for the i^{th}class as in Eq. (8.1).

$$P(y = i|x) = \frac{e^{x^T w_i}}{\sum_{k=1}^{K} e^{x^T w_k}} \tag{8.1}$$

This proposed IRRCNN model has been investigated through a set of experiments on different benchmark datasets for blood cell classification and compared against the existing DL based models.

8.4 Results and Discussion

To demonstrate the performance of the IRRCNN models, we have tested them on two different microscopic imaging datasets for WBC and RBC classification. In most cases, research has been conducted on datasets that are not publicly available. The details of the datasets used in this work are given in the following paragraphs. For this implementation, the Keras and TensorFlow frameworks were used on a single GPU machine with 56G of RAM and an NVIDIA GEFORCE GTX-980 Ti.

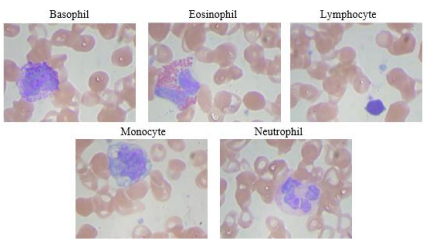


Figure 8.4: Example classes from the WBC dataset.

8.4.1 Database

8.4.1.1 WBC dataset

This dataset contains 352 colored images of WBCs [252]. Each image is of size 640 x 480. A random selection of example images is shown in Figure 8.4. The database statistics and distributions of the samples are given in Table 8.1. In this implementation, we have downsampled the images to 128×128 pixels. Since the number of samples for the basophil case is very low (there are only 3 samples), we did not consider this blood cell class in this implementation. Therefore, the IRRCNN is used to classify four classes (Monocyte, Eosinophil, Neutrophil, and Lymphocyte) in this implementation.

Table 8.1: Statistics of the WBC dataset.

WBC class	**Type**	**Number of samples**
Monocyte	Mononuclear	21
Lymphocyte	Mononuclear	33
Neutrophil	Polynuclear	207
Eosinophil	Polynuclear	88
Basophil	Mononuclear	3
Total		352

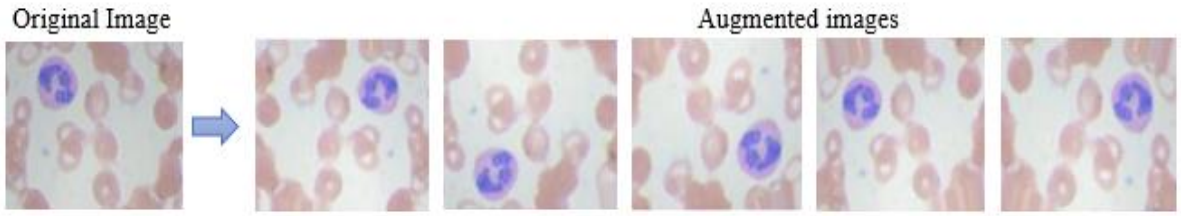


Figure 8.5: Original image and augmented samples.

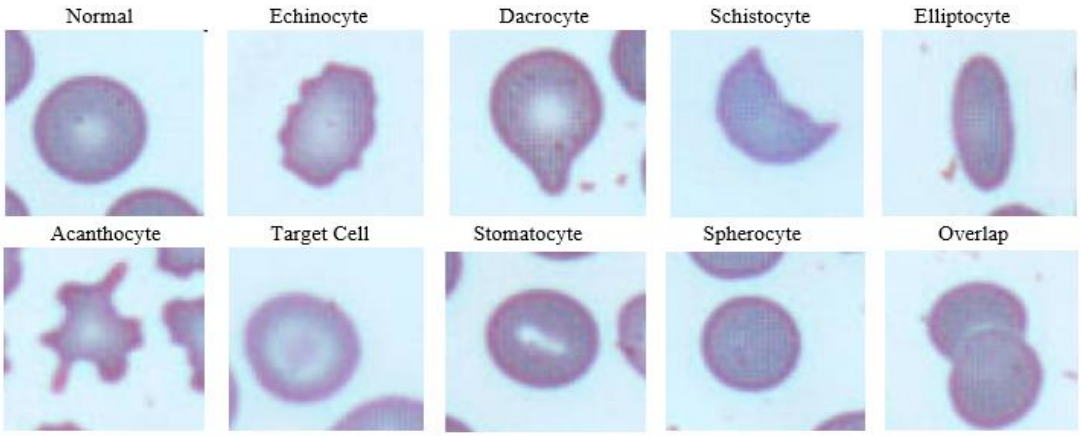


Figure 8.6: Randomly selected samples from the RBC dataset.

Given that the cells do not have an expected orientation per their nature, we have performed different augmentation approaches including flipping, rotation, and shearing to dramatically increase the number of samples in the training set. An example original image and several augmentations of that image are shown in Figure 8.5. Using this technique, we increased the number of training samples from 352 to 9,957. We also applied the data augmentation technique to ensure that we have the same number of samples for each of the four blood cell classes (around 2,500 samples each). We used approximately 80% of the samples for training and the remaining 20% of the samples for testing. From the training set, about 10% of the samples are used for validation during the training phase.

8.4.1.2 RBC dataset

Table 8.2: Experimental results of the IRRCNN approaches for the WBC classification, and comparison against other existing approaches.

Methods	**Year**	**PARAMS.**	**SE**	**SP**	**JSC**	**F1-score**	**AC (%)**	**AUC**
AlexNet like model [252]	2016	0.987M					98.6 %	
IRRCNN (*t*=2)	2018	0.648M	0.9989	0.9996	0.9994	0.9989	99.94%	0.999
IRRCNN (*t*=2)	2018	2.250M	1.00	1.00	1.00	1.00	100 %	1.00

The RBC dataset consists of 3,737 colored images with ten different classes. In this work, 2,989 samples were used for training, and the remaining 784 samples were used for testing [253]. During training, we used 10% of the samples from the training data for validation purposes. A random selection of example images is shown in Figure 8.6.

Table 8.3: Experimental results of the IRRCNN approaches for RBC classification, and comparison against other existing approaches.

Methods	**Year**	**PARAMS**	**SE**	**SP**	**JSC**	**F1-score**	**AC**	**AUC**
VGG like Net [253]	2017	-	-	-	-	-	97.59%	-
IRRCNN (*t*=2)	2018	**0.57M**	0.9478	0.9949	0.9902	0.9510	99.02%	0.9970
IRRCNN (*t*=2)	2018	**2.17M**	0.9933	0.9992	0.9986	0.9933	99.86%	0.9999
IRRCNN (*t*=3)	2018	**9.15M**	0.9973	0.9997	0.9994	0.9973	99.94%	0.9999

8.4.2 Evaluation metrics

Several performance metrics are considered. Accuracy (AC), sensitivity (SE), specificity (SP), F1-score, and Jaccard similarity (JS) are used for quantitative analysis of the experimental results in accordance with work in [222]. The variables True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) are calculated as well. The overall accuracy is calculated using Eq. (8.2), and sensitivity is calculated using Eq. (8.3). Furthermore, specificity is calculated using Eq. (8.4).

$$\mathrm{AC} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}} \tag{8.2}$$

$$\mathrm{SE} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \tag{8.3}$$

$$\mathrm{SP} = \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}} \tag{8.4}$$

The Area Under the Curve (AUC) and the Receiver Operating Characteristics (ROC) curve are common evaluation metrics for medical image classification and segmentation tasks. We utilized

both of these analytical methods to evaluate the performance of the IRRCNN model for blood cell classification considering the mentioned criteria against existing state-of-the-art techniques.

8.4.3 Results

8.4.3.1 WBC classification results

Table 8.2 shows the experimental results for WBC classification. We performed testing with two different types of IRRCNN models with different numbers of network parameters, where t = 2 for the recurrent convolutional layers used. For evaluating the performance of the proposed approach, we calculated sensitivity, specificity, accuracy, and Area Under the Curve (AUC). The experimental results are compared against existing deep learning-based approaches, and our proposed approach shows approximately 1.4% superior performance for WBC classification compared to recently proposed CNN based approaches.

8.4.3.2 RBC classification results

Figure 8.7 shows the training accuracy with different IRRCNN models for RBC classification for ten classes. We experimented with the IRRCNN models using different numbers of network parameters with t = 2 and t = 3. From Figure 8.7, it can be observed that the IRRCNN model with 9.17 million parameters shows the best performance during training, and the IRRCNN with 2.1 million parameters shows nearly the same level of performance. However, we did not observe a significant difference in accuracy during the testing phase. The testing accuracy is shown in Table 8.3. The best testing accuracy was achieved with the IRRCNN model where t = 3, which was approximately 99.94%. This is about a 2.35% better testing accuracy when compared to the VGG like a model in [27]. The precision and recall curve for RBC classification is shown in Figure 8.8.

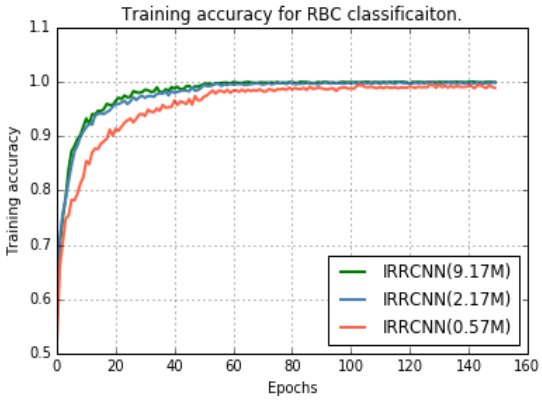


Figure 8.7: Training accuracy of IRRCNN for different numbers of network parameters.

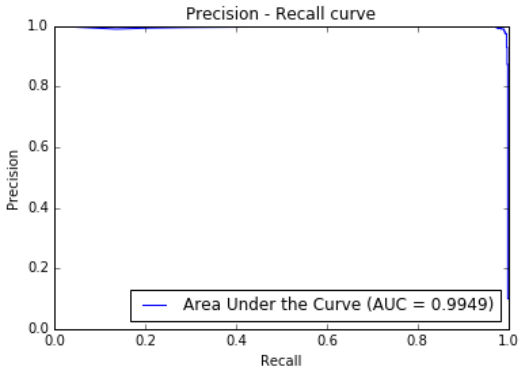


Figure 8.8: Precision and recall curve for RBC classification with IRRCNN.

8.4.4 Evaluation

In this experiment, we used IRRCNNs to achieve high accuracy with only 0.648 million and 0.57 million network parameters for WBC and RBC classification respectively. The testing time per sample is given in Table 8.4. Existing works [26,27] used the AlexNet and VGG Net networks for WBC and RBC classification, and these show significantly lower testing accuracy compared to our proposed model.

Table 8.4. Computational time per testing sample.

Dataset	Time (seconds/sample)
WBC classification	0.023
RBC classification	0.047

Furthermore, the AlexNet and VGG Net models require a much larger number of network parameters, which also makes them computationally expensive without any performance improvement for blood cell classification.

8.5 Conclusion and Future Works

In this paper, we applied an advanced IRRCNN model to blood cell classification. The model has evaluated on publicly available datasets for white and red blood cell classification. The experimental results show a promising classification accuracy during the testing phase, which was 100% and 99.94% for white blood cells and red blood cells respectively. This is approximately a 1.4% and 2.35% better testing accuracy compared to existing deep learning-based approaches. We have achieved the state-of-the-art accuracy for blood cell classification in this implementation. In the future, we would like to experiment on more difficult datasets for blood cell classification.

CHAPTER 9

IRRCNN FOR BREAST CANCER RECOGNITION

The Deep Convolutional Neural Network (DCNN) is one of the most powerful and successful deep learning approaches. DCNNs have already provided superior performance in different modalities of medical imaging including breast cancer classification, segmentation, and detection. Breast cancer is one of the most common and dangerous cancers impacting women worldwide. In this paper, we have proposed a method for breast cancer classification with the Inception Recurrent Residual Convolutional Neural Network (IRRCNN) model. The IRRCNN is a powerful DCNN model that combines the strength of the Inception Network (Inception-v4), the Residual Network (ResNet), and the Recurrent Convolutional Neural Network (RCNN). The IRRCNN shows superior performance against equivalent Inception Networks, Residual Networks, and RCNNs for object recognition tasks. In this paper, the IRRCNN approach is applied for breast cancer classification on two publicly available datasets including BreakHis and Breast Cancer Classification Challenge 2015. The experimental results are compared against the existing machine learning and deep learning-based approaches with respect to image-based, patch-based, image-level, and patient-level classification. The IRRCNN model provides superior classification performance in terms of sensitivity, Area Under the Curve (AUC), the ROC curve, and global accuracy compared to existing approaches for both datasets.

9.1 Introduction

Nowadays, cancer is one of the leading causes of morbidity and mortality around the world. Approximately 14.5 million people have died due to cancer, and it is estimated that this number

will be above 28 million by 2030. According to a study by the American Cancer Society (ACS), in the USA the estimated deaths due to breast cancer account for approximately 14% of all cancer deaths (a total of 41,000 in 2017) which is in the second-leading cause of cancer death in women after lung and bronchus cancer. Additionally, breast cancer accounts for 30% of all newly discovered cancer cases. Breast cancer is the most frequently diagnosed cancer in women in the USA. A biopsy followed by microscopic image analysis is common when diagnosing breast cancer [254]. A breast tissue biopsy allows the pathologist to histologically access the microscopic level structures and components of the breast tissue. These histological images allow the pathologist to distinguish between the normal tissue, non-malignant (benign) tissue, and malignant lesions. The resulting information is then used to perform a prognostic evaluation [255].

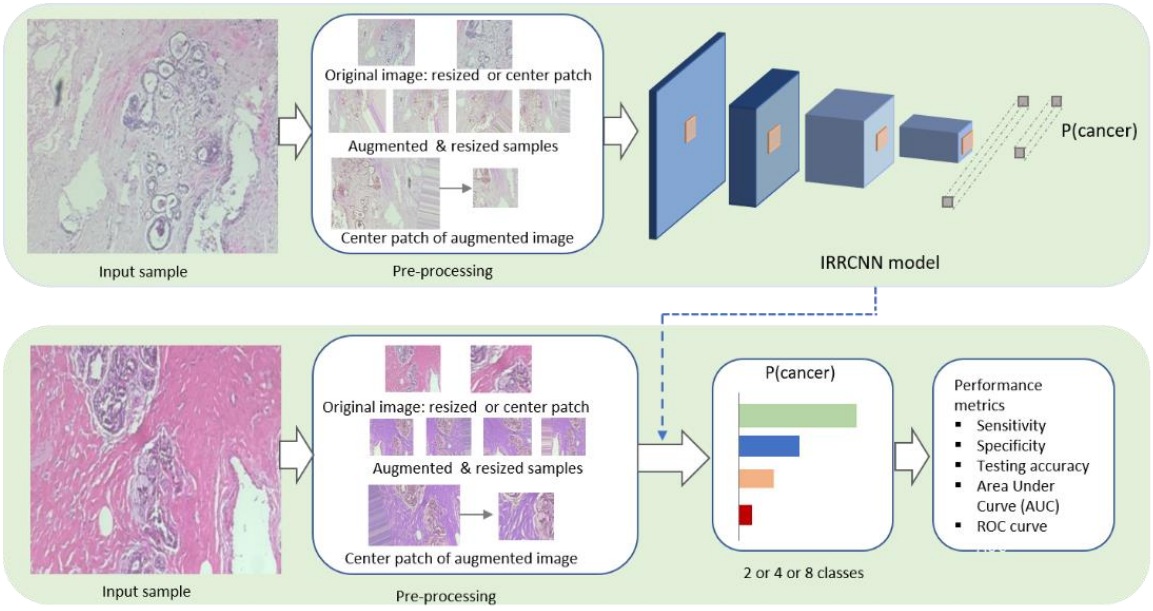


Figure 9.1: Implementation diagram for breast cancer recognition using the IRRCNN model. The upper part of this figure shows the steps that are used for training the system, and the lower part of this figure displays the testing phase where the trained model is used. These results are evaluated with several different performance metrics.

Benign lesions refer to changes in the normal tissue of breast parenchyma and are not related to the progression of malignancy. There are two different carcinoma tissue types including in-situ and

invasive. The in-situ tissue type refers to tissue contained inside the mammary ductal-lobular. On the other hand, the invasive carcinoma cells spread beyond the mammary ductal-lobular structure. The tissue samples that are collected during biopsy are commonly stained with Hematoxylin and Eosin (H&E) prior to the visual analysis performed by the specialist. During the diagnosis process, the affected region is determined from whole-slide tissue scans [256]. In addition, the pathologist analyzes microscopic images of the tissue samples from the biopsy with different magnification factors. Nowadays, to produce the correct diagnosis, the pathologist considers different characteristics within the images including patterns, textures, and different morphological properties [257]. Analyzing images with different magnification factors requires panning, zooming, focusing, and scanning of each image in its entirety. This process is very time consuming and tiresome, as a result, this manual process sometimes leads to inaccurate diagnosis for breast cancer identification. Due to the advancement of digital imaging techniques in the last decade, different computer vision and machine learning techniques have been applied for analyzing the pathological images at a microscopic resolution [257,258]. These approaches could help to automate some of the tasks related to the pathological workflow in the diagnosis system. However, an efficient and robust image processing algorithm is necessary for use in clinical practices. Unfortunately, traditional approaches are unable to fulfill the expectation. As a result, we are still far from the practical application of automatic breast cancer detection based on histological images [258].

However, recent developments in Deep Learning (DL) have already shown vast potential with state-of-the-art performance on different recognition tasks in the field of computer vision and image processing, speech recognition, and natural language understanding [1]. These approaches have been applied in different modalities of medical imaging including pathological imaging with superior performance in classification, segmentation, and detection [259]. In some cases, the DL based systems have become part of the workflow for clinical practices with pathologists and doctors. Some examples include a dermatologist-level performance for skin cancer detection, diabetic retinopathy, neuroimaging for analysis of brain tumors and Alzheimer disease, lung cancer

detection, and breast cancer detection, segmentation, and classification [259]. Although these approaches have shown tremendous success in medical imaging, they require a very large amount of label data, which is still not available in this domain of applications for several reasons. Most significantly, it requires a lot of expertise to annotate a dataset which is very expensive. In this paper, we propose a DL based approach for breast cancer recognition system using the IRRCNN model which is evaluated using the BreaKHis and Breast Cancer Classification Challenge 2015 datasets. The contributions of this paper are summarized as follows:

- Successful magnification factor invariant binary and multi-class breast cancer classification using the IRRCNN model.
- Experiments have been conducted on recently released publicly available datasets for breast cancer histopathology (such as the BreaKHis dataset) where we evaluated the image and patient-level data with different magnifying factors (including 40×, 100×, 200×, and 400×).
- The image-based and patch-based evaluation was performed for both the BreaKHis and Breast Cancer Classification Challenge 2015 datasets
- The experimental results are compared against recently proposed deep learning and machine learning approaches, and our proposed model provides superior performance when compared to the existing algorithms for breast cancer classification.

9.2 Related Works

Significant effort has been put forth for breast cancer (BC) recognition from histological images in the last decade, where most efforts are made to classify the two fundamental types of breast cancer (benign and malignant) using Computer Aided Diagnosis (CAD). Before the deep learning revolution, machine learning approaches including the Support Vector Machine (SVM), Principle Component Analysis (PCA), and Random Forest (RF) were used to study data whose features were extracted with Scale Invariant Feature Extraction (SIFT), Local Binary Patterning (LBP), Local

Phase Quantization (LPQ), the Gray-Level Co-occurrence Matrix (GLCM), Threshold Adjacency Statistics (TAS), and Parameter Free TAS (PFTAS). In 2016, one of the very popular databases for BC classification problem was released, and one research group reported approximately 85.1% accuracy utilizing SVM and PFTAS features for patient-level analysis [260], which was the highest recognition accuracy at the time. Another work was published in 2013 where different algorithms (including K-means, fuzzy C-means, competitive learning neural networks, and Gaussian mixture models) were used for nuclei classification on a dataset with 500 samples from 50 patients. The accuracies were reported for binary classes (benign versus malignant). This work produced accuracies ranging from 96 – 100 percent [261].

A machine learning system for breast cancer recognition based on Neural Networks (NN) and SVM was published in 2013 that reported 94% recognition accuracy on a dataset that consisting of 92 samples [262]. Another method was proposed based on cascading with a rejection option that was tested on a dataset with 361 samples from the Israel Institute of Technology, and it reported around 97% classification accuracy [263]. For the most part, research in this area has been conducted using a very small number of samples from primarily private datasets. Recently a survey was published on histological image analysis for breast cancer detection and classification that clearly describes the qualities and limitations of different publicly available annotated datasets [264]. An effective framework has been proposed with color texture features and multiple classifiers utilizing a voting technique that reported approximately 87.53% average recognition rate for patient-level BC classification. In this implementation, the SVM, the Decision Tree (DT), a Nearest Neighbor Classifier (NNC), Discriminant Analysis (DA), and Ensemble classifiers were used. Before 2017, this system achieved the best recognition accuracy of all machine learning based approaches [265]. Furthermore, many works have already been published that discuss breast cancer recognition using DL approaches, where CNN variants are applied for classification. A few of these experiments are conducted with the BreaKHis dataset. In 2016, a magnification independent breast cancer classification was proposed based on a CNN where different sized convolution kernels (7×7, 5×5,

and 3×3) were used. They performed patient level classification of breast cancer with CNN and multi-task CNN (MTCNN) models and reported an 83.25% recognition rate [266]. In the same year, another work was published based on a model similar to AlexNet with different fusion techniques (including sum, product, and max) for image and patient level classification of breast cancer. This paper reports 90% and 85.6% average recognition accuracy with the max fusion method for images and patient level classification respectively [267]. Another deep learning-based method was published in 2017. In this work, a pretrained CNN was used to extract the feature vectors, and eventually, the feature vectors were used as the input to a classifier. This method was called DeCAF and achieved a recognition accuracy of 86.3% and 84.2% at the patient level and image level respectively [268].

The CNN model was used for the classification of H&E stained breast biopsy images from another challenging dataset in 2017 [269]. The images were classified according to four different classes: normal tissue, benign lesion, in-situ carcinoma, and invasive carcinoma. Images were also classified in terms of binary classes, carcinoma (which includes normal and benign tissue) and non-carcinoma (which includes the in-situ and invasive carcinoma classes) are considered. Work in [269] provides results for both image-based and patch-based evaluation. The CNN based approach achieved approximately 77.8% recognition accuracy when performing the four-class experiment, and 83.3% recognition accuracy for the binary class experiment when tested with the BC Classification Challenge 2015 dataset. Recently, multi-classification of breast cancer from histopathological images was presented using a structured deep learning model called CSDCNN. This new DL architecture shows superior performance when compared to different machine learning and deep learning-based approaches on the BreaKHis dataset. This model shows state-of-the-art performance for both image-level and patient-level classification. An average of 93.2% accuracy for patient-level breast cancer classification has been reported [270,271]. In 2017, different SMV based techniques were applied for breast cancer recognition, an accuracy of 94.97% for data with a 40× magnification factor was achieved using an Adaptive Sparse SVM (ASSVM)

[272]. However, our work presents an application of a new deep learning model called the Inception Recurrent Residual Convolutional Neural Network (IRRCNN) for BC classification on both the BreaKHis and 2015 Breast Cancer Classification Challenge datasets.

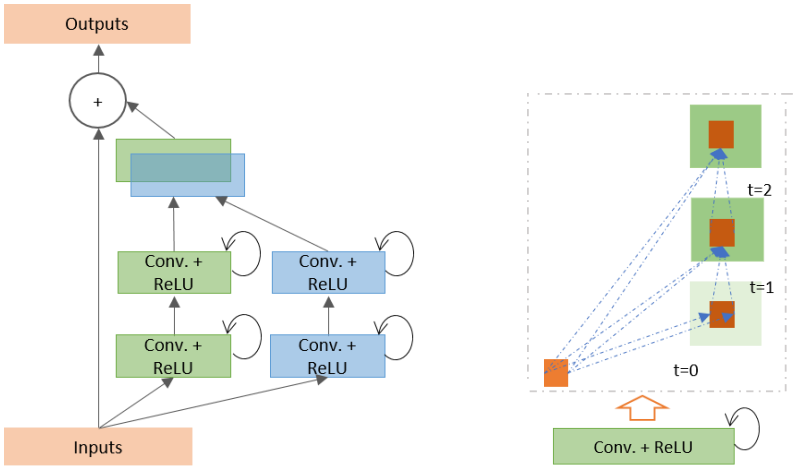


Figure 9.2: Diagrams displaying the Inception Recurrent Residual Unit (IRRU) consisting of the inception unit and recurrent convolutional layers that are merged by concatenation, and the residual units (summation of the input features with the outputs of the inception unit can be seen just before the output block).

9.3 IRRCNN Model for Breast Cancer Recognition

DL approaches show tremendous success in cases where sufficient labeled data is available, and several advanced deep learning approaches have been proposed that have shown state-of-the-art performance in different modalities of computer vision and medical imaging in the last few years [1,259]. The Inception Recurrent Residual Convolutional Neural Network (IRRCNN) [196] is one out of many which is an improved hybrid DCNN architecture based on inception [24], residual networks [11], and the RCNN architecture [52]. The main advantage of this model is that it provides better recognition performance using the same number or fewer network parameters when compared to alternative equivalent deep learning approaches including inception, the RCNN, and the residual network. In this model, the inception-residual units are utilized with respect to the Inception-v4 model [24]. The IRRCNN has been compared against equivalent inception-residual networks, and it shows better performances [196]. The IRRCNN model is comprised of stacks that

include both inception recurrent residual units (IRRU) and transition units. The entire model is shown in Figure 9.1. The overall model consists of several convolution layers, IRRUs, transition blocks, and a softmax at the output layer. A pictorial view of the IRRU is shown in Figure 9.2.

The most important unit in the IRRCNN architecture is the IRRU, which includes Recurrent Convolutional Layers (RCLs), inception units, and a residual layer. The inputs are fed into the input layer, then passed through inception units where RCLs are applied, and finally, the outputs of the inception units are added to the inputs of the IRRU. The recurrent convolution operations are performed with respect to the different sized kernels in the inception unit. Due to the recurrent structure within the convolution layer, the outputs at the present time step are added to the outputs of the previous time step. The outputs at the present time step are then used as inputs for the next time step. The same operations are performed with respect to the time steps that are considered. For example, where $t = 2$ *(0~2)* means that one feed forward convolution along with 2 RCLs are included in IRRU. The operation of the RCLs with respect to the different time steps ($t = 2$ *(0~2)* and $t = 3$ *(0~3)*) is shown in Fig. 2. Due to the residual connectivity in the IRRU, the input and output dimensions do not change. The IRRU simply performs an accumulation of feature maps with respect to the time steps. Thus, better feature representation is ensured and this system achieves superior recognition accuracy with the same number of network parameters.

The operations of the RCL are performed with respect to discrete time steps that are expressed according to the IRRCNN in [196]. Let's consider the x_l input sample in the l^{th} layer of the IRRCNN block, and the unit (i, j) from an input sample in the k^{th} feature map in the RCL. Additionally, let's assume the output of the network $O_{ijk}^{l}(t)$ is at time step t. Given this information, the output can be expressed as in equation (9.1).

$$O_{ijk}^{l}(t) = \left(w_k^f\right)^T * x_l^{f(i,j)}(t) + (w_k^r)^T * x_l^{r(i,j)}(t-1) + b_k \tag{9.1}$$

Here $x_l^{f(i,j)}(t)$ and $x_l^{r(i,j)}(t-1)$ are the inputs for the standard convolution layers and for the l^{th} RCL respectively. The w_k^f and w_k^r values are the weights for the standard convolutional layer and the RCL of the k^{th} feature map respectively, and b_k is the bias.

$$y = f(O_{ijk}^l(t)) = \max(0, O_{ijk}^l(t)) \tag{9.2}$$

In (9.2), f is the standard Rectified Linear Unit (ReLU) activation function. We have also explored the performance of this model with the Exponential Linear Unit (ELU) activation function in the following experiments[150]. The outputs y of the inception units for the different size kernels and average pooling layer are defined as $y_{1x1}(x)$, $y_{3x3}(x)$, and $y_{1x1}^p(x)$ respectively. The final outputs of Inception Recurrent Convolutional Neural Network (IRCNN) unit are defined as $\mathcal{F}(x_l, w_l)$ which can be expressed as in equation (9.3).

$$\mathcal{F}(x_l, w_l) = y_{1x1}(x) \odot y(x) \odot y_{1x1}^p(x) \tag{9.3}$$

Here $\odot$ represents the concatenation operation with respect to the channel or feature map axis. The outputs of the IRCNN unit are then added with the inputs of the IRRCNN block. The residual operation of the IRRCNN block can be expressed as in equation (9.4).

$$x_{l+1} = x_l + \mathcal{F}(x_l, w_l) \tag{9.4}$$

In equation (9.4), x_{l+1} refers to the inputs for the immediate next transition block, x_l represents the input samples of the IRRCNN block, w_l represents the kernel weights of the l^{th} IRRCNN block, and $\mathcal{F}(x_l, w_l)$ represents the outputs from of l^{th} layer of the IRCNN unit. However, the number of feature maps and the dimensions of the feature maps for the residual units are the same as in the IRRCNN unit shown in Figure 9.2. Batch normalization is applied to the outputs of the IRRU [196]. Eventually, the outputs of this IRRU are fed to the inputs of the immediate next transition unit.

In the **transition unit,** different operations including convolution, pooling, and dropout are performed depending upon the placement of the transition unit in the model. The inception units are included in the transition unit. The down-sampling operations are performed in the transition units, where we perform max-pooling operations with a 3×3 patch and a 2×2 stride. The non-overlapping max-pooling operation has a negative impact on model regularization, therefore we

used overlapped max-pooling for regularizing the network which is very important when training a deep network architecture [1]. The late use of a pooling layer helps to increase the non-linearity of the features in the network, as this results in higher dimensional feature maps being passed through the convolution layers in the network. We used two special pooling layers in the model with three IRRCNN units and one transition unit for this implementation.

We used only 1×1 and 3×3 convolution filters in this implementation, as inspired by the NiN [19] and Squeeze Net [147] models. This also helps to keep the number of network parameters at a minimum. The benefit of adding a 1×1 filter is that it helps to increase the non-linearity of the decision function without having any impact on the convolution layer. Since the size of the input and output features does not change in the IRRCNN units, the result is just a linear projection on the same dimension, and non-linearity is added to the RELU and ELU activation functions. We used a 0.5 dropout after each convolution layer in the transition block. Finally, we used a softmax or normalized exponential function layer at the end of the architecture. For an input sample *x*, a weight vector *W*, and *K* distinct linear functions, the softmax operation can be defined for the i^{th}class as in equation (9.5).

$$P(y = i|x) = \frac{e^{x^T w_i}}{\sum_{k=1}^{K} e^{x^T w_k}} \tag{9.5}$$

The proposed IRRCNN model has been investigated through a set of experiments on different benchmark datasets, and the results have been compared across several different models.

The IRRCNN model is evaluated with different numbers of convolutional layers in the convolution blocks, and the number of layers is determined with respect to time step *t*. In these implementations, $t = 2$ refers to an RCL block that contains one forward convolution followed by two RCLs [196]. For both breast cancer recognition datasets, we used a model with two convolutional layers at the beginning, four IRCNN blocks followed by transition blocks, a fully connected layer, and a softmax layer at the end of the model. For this model, we considered 32 and 64 feature maps for the first three convolutional layers, and we used 128, 256, 512, and 1024 feature maps in the first, second,

third, and fourth IRRCNN blocks respectively. Batch Normalization (BN) is used in each IRCNN block. This model contains approximately 9.3 million network parameters.

9.4 Experimental Results and Discussion

9.4.1 Experimental setup

To demonstrate the performance of the IRRCNN models, we have tested them on two different BC datasets: the BreakHis dataset, and the Breast Cancer Classification Challenge 2015 dataset for both binary and multi-class BC classification. The following paragraph discusses both datasets in detail. For this implementation, the Keras [171], and Tensor Flow [274] frameworks were used on a single GPU machine with 56G of RAM and an NVIDIA GEFORCE GTX-980 Ti. We considered different criterion for pathological image analysis in this implementation. In most cases, the dimensions of the Whole Slide Images (WSI) are larger than typical digital images. In addition, the pathological images are acquired with different magnification factors. In some cases, the image size is larger than 2000 × 2000 pixels. However, in this case, the images are typically fed to the model as several patches. There are two common processes used for patch selection, one of which is a random crop method where the patches are cropped from a random location within an input sample. The alternative is to use sequential and non-overlapping patches. We considered both methods in this implementation.

Table 9.1: Statistics for the main and subclass samples and number of patients for the BreaKHis dataset.

Classes	**Subclasses**	**Number of Patients**	**Magnification factors**				**Total**
			40×	**100×**	**200×**	**400×**	
Benign	A	4	114	113	111	106	444
	F	10	253	260	264	237	1014
	TA	3	109	121	108	115	453
	PT	7	149	150	140	130	569
Malignant	DC	38	864	903	896	788	3451
	LC	5	156	170	163	137	626
	MC	9	205	222	196	169	792
	PC	6	145	142	135	138	560
Total		82	1995	2081	2013	1820	7909

9.4.2 Datasets

BreakHis: The BreaKHis dataset is publicly available and is commonly used to study the breast cancer classification problem. This dataset contains 7909 samples each falling within two main classes: benign or malignant. The benign subset contains 2440 samples and the malignant subset contains 5429 samples. The samples are collected from 82 patients with different magnification factors including 40×, 100×, 200×, 400×. Some of the example images with a 400× magnification factor are shown in Figure 9.3. Each class has four subclasses, the four types of benign cancer are Adenosis (A), Fibroadenoma (F), Tubular Adenoma (TA), and Phyllodes Tumor (PT). The four subclasses of malignant cancer are Ductal Carcinoma (DC), Lobular Carcinoma (LC), Mucinous Carcinoma (MC), and Papillary Carcinoma (PC). The statistics for this dataset are given in Table 9.1. In this experiment, we used 70% of the samples for training and 30% of the samples for testing, per the work in [264,268]. To generalize the classification task to perform successfully when testing new patients, we ensure that the patients selected for training are not used during testing. Per the experimental design in [264], we reported the average accuracy after successfully completing five trials.

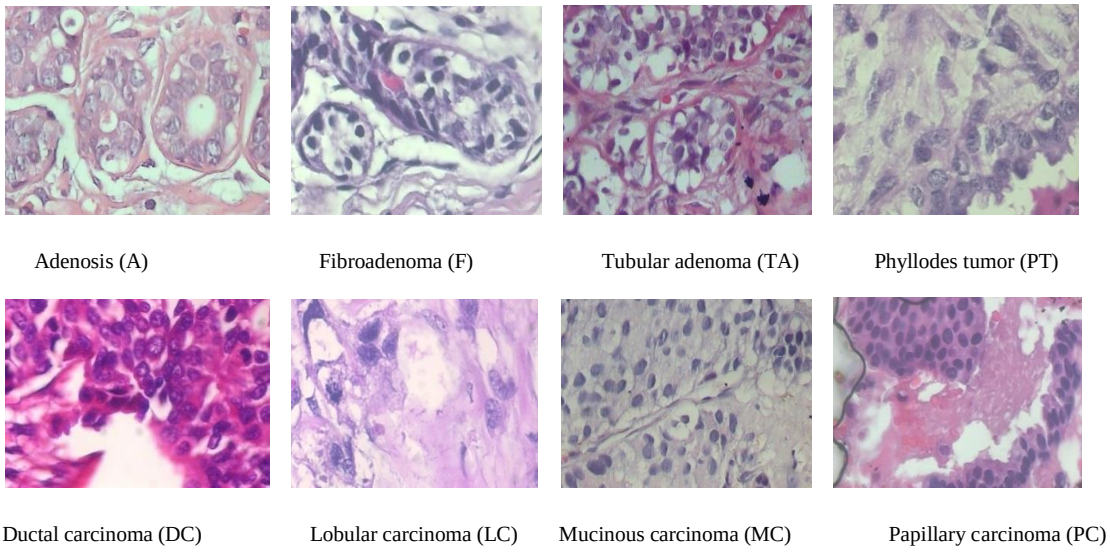


Figure 9.3: The first row shows the four types of benign tumors, and the second row shows the malignant tumors. The magnification factor of these images is 400×.

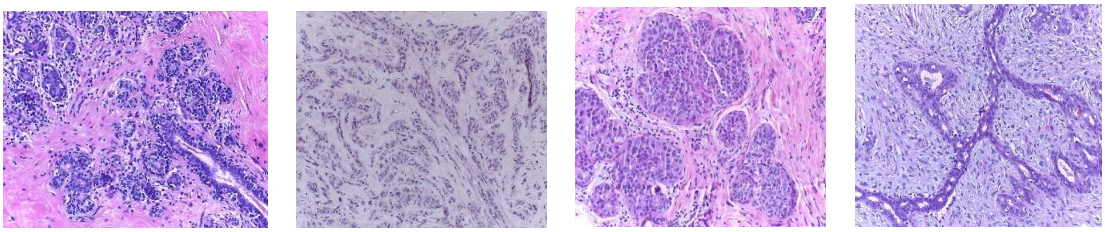
For data augmentation, we generated 21 samples from each single input sample with different augmentation techniques including rotation, flipping, shearing, and translation. Therefore, the total number of samples was increased by 21 times. For example, the total number of images available at a 40× magnification is now 41,895. We generated 43701, 42273, and 38220 patches from the original samples for the 100×, 200×, and 400× magnification factors respectively. Thus, a total number of augmented samples for all magnification factors is 166,068. We evaluated the image-level and patient-level performance for both binary and multi-class breast cancer recognition.

Table 9.2: Statistics for the BC classification challenge dataset.

Methods	**Non-Carcinoma**		**Carcinoma**		**Total**
	Normal	**Benign**	**In situ**	**Invasive**	
Image-wise	55	69	63	62	249
Augmented samples	1155	1449	1323	1302	5229
Random patches	9716	12057	11059	10875	43,707

Breast Cancer Classification Challenge 2015: This dataset consists of very high resolution (2040×1536) digital pathology images, which are annotated H&E stained images for breast cancer classification released in 2015 [269]. This dataset contains a total of 249 samples, from which 229 samples are separated for training, and the remaining samples are considered for testing, per the work in [269]. The images were labeled by two pathologists and the overall context has been considered without specifying the area of interest. Each image is assigned one of the following four categories: (a) normal tissue (b) benign (c) in-situ, and (d) invasive carcinoma. Sample images displaying the four different types of BC are shown in Figure 9.4. Each class has about 60 samples, which resolves the class imbalance problem for classification tasks. In this implementation, the model is evaluated for binary and multi-class BC classification. In case of the binary classification problem, the normal tissue and benign subsets are considered class one, and the in-situ and invasive carcinoma subsets are considered to be a part of class two. According to a visual analysis of the dataset, it is observed that the nuclei radius ranges from 3 to 11 pixels (or 1.26μm to 4.62 μm).

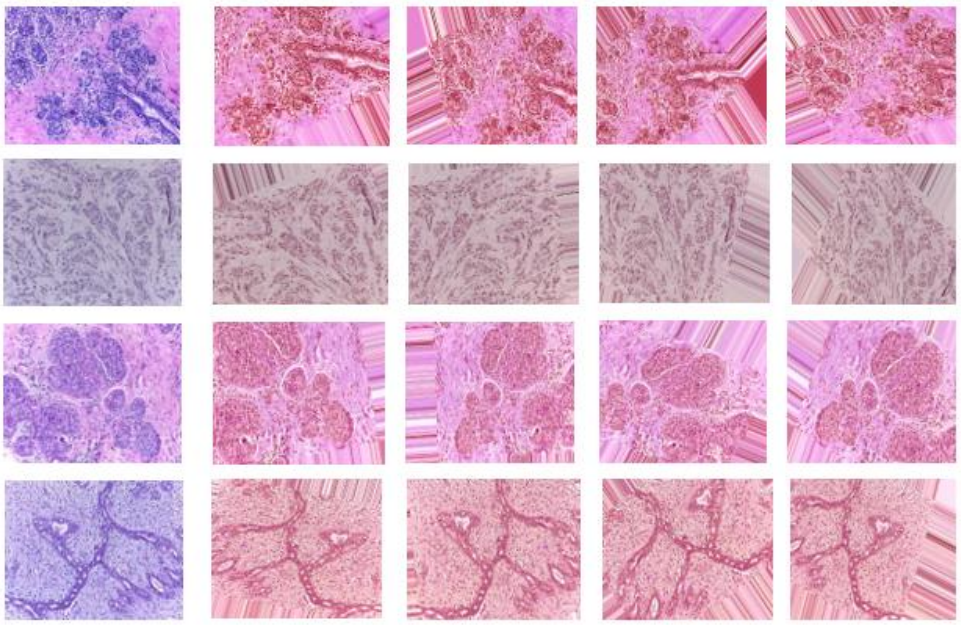
Therefore, patches with size 128×128 pixels are able to cover enough of the tissue structure (in accordance with the experiment conducted in [269]).



Normal Benign In-situ Carcinoma Invasive Carcinoma

Figure 9.4: Sample images of four types of breast cancer (normal, benign, in situ carcinoma, and invasive carcinoma) from the 2015 BC Classification Challenge dataset.

We have conducted experiments using both image-wise and patch-wise evaluation. For image-wise classification, we used three different approaches: first, we resized the input samples to 128×128 pixels which significantly degrades the information contained in the samples. Second, different data augmentation techniques were applied to the resized images where 20 different augmented samples were generated for each sample. Third, 200 random patches were cropped to create a patch database for training and testing the model. A Winner Take All (WTA) method was used to produce the results where the final class was determined based on the class where the maximum number of patches were nominated. The labels of the patches are considered to have same class label as the original images. On the other hand, using a patch-wise approach: first, 128×128 pixel center patches were cropped from an input sample. Second, the augmentation techniques were applied to the center patches and 20 augmented samples per patch were generated. Third, we evaluated the model with 200 randomly selected patches with a size of 128×128 pixels from a single image. The statistics for the image-wise and patch-wise approach are given in Table 9.2.



Input samples Augmented samples

Figure 9.5: Four example images with corresponding augmented images. The actual images are shown on the left, and four augmented samples (of the 20 created for each image) are shown on the right.

9.4.3 Data augmentation

In each dataset, we applied different data augmentation techniques including sequential rotation by 40 degrees, width shift with a factor of 0.2, height shift with a factor of 0.2, shear with a factor of 0.2, zooming with a range 0.2, horizontal flipping, and vertical flipping. Figure 9.5 shows some example images along with different augmented samples for the four different data classes. From Figure 9.5, it can be observed that noise has been added to some of the parts of the images. Therefore, we have also evaluated our method using only the center patch of the augmented samples. The downsampled and center patches are shown for two different input samples in Figure 9.6.

Training Methodology**:** In the first experiment, we trained with the IRRCNN architecture using the stochastic gradient descent (SGD) optimization function. We set the momentum to 0.9 and decay is calculated based on the initial learning rate and number epochs of the respective trial. We have

experimented for three trials where 50 epochs are used in each trail. After 50 epochs, the learning rate is decreased by the factor of 10.

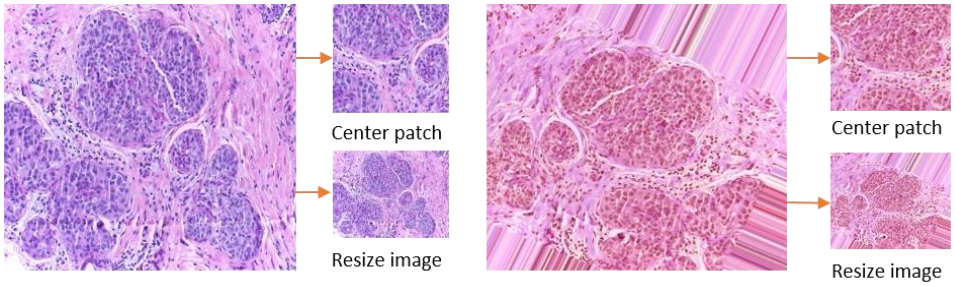


Figure 9.6: Center patch and resized images from an original sample (left) and from an augmented sample (right).

9.4.4 Results and discussion

In this work, we introduced automated breast cancer classification for both the binary and multi-class problems on two different datasets. In the case of the multi-class BC classification problem, four and eight classes were considered in this implementation. We achieved state-of-the-art testing accuracy for both datasets.

Results for BreakHis: According to the work in [264,270], we considered two criteria on which to evaluate the performance of the IRRCNN model. We considered (1) image-level and (2) patient-level performance for multi-class classification for eight types of breast cancer that fall within the two main types (either benign or malignant). In addition, we have also evaluated the performance of a binary class system for benign and malignant types. For image-level classification, we did not consider images with respect to the patient. For this experiment, the images are organized into eight classes, and the images contain a magnification factor of either 40×, 100×, 200×, or 400×. Performance is measured by different evaluation metrics in this case. Two different performance criteria are considered to evaluate the performance of the IRRCNN deep learning approach as in [270]. First, we considered a patient-level performance analysis where the total number of patients is defined as N_{np}, the number of BC images of associated patient (P) is defined as N_{ncp}. The

number of correctly classified images for a patient is denoted N_{ntp}. Equation (9.6) defines the patient score (P_s).

$$P_s = \frac{N_{ncp}}{N_{ntp}} \tag{9.6}$$

The global patient recognition rate (P_{rt}) is defined in equation (9.7).

$$P_{rt} = \frac{\sum P_s}{N_{np}} \tag{9.7}$$

We also calculated the performance of the IRRCNN approach for image-level classification. We define the total number of samples available for testing as N_T. The correctly classified number of histopathological samples is defined as N_{CCT}. The image-level recognition rate (I_{rt}) is expressed in equation (9.8).

$$I_{rt} = \frac{N_{CCT}}{N_T} \tag{9.8}$$

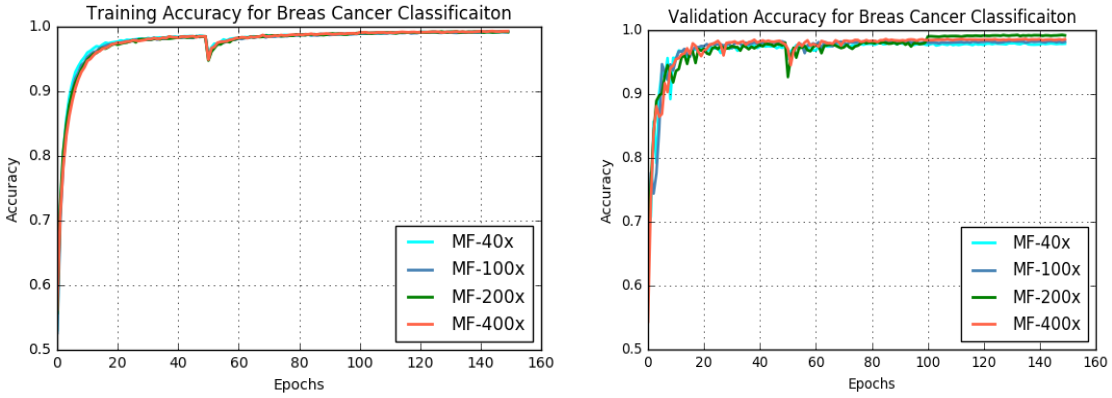


Figure 9.7. Training and validation accuracy for BC classification with 8 classes for the IRRCNN model at different magnification factors.

The training and validation accuracy of the IRRCNN model for breast cancer classification is shown in Figure 9.7. From this figure, it can be observed that the magnification factors of the samples have an impact on training and testing accuracy. We achieved the best training accuracy with a magnification factor of 100×, and the training accuracy achieved for data with a magnification factor of 200× is a very close second.

Table 9.3: Breast cancer classification results for multi-class (8 classes) on the BreakHis dataset.

	Methods	Year	Classification Rate (100R) at Magnification Factor			
			40×	100×	200×	400×
Image Level	CNN+patches [267]	2016	85.6 ± 4.8	83.5 ± 3.9	83.1 ± 1.9	80.8 ± 3.0
	LeNet + Aug [270]	2017	40.1 ± 7.1	37.5 ± 6.7	40.1 ± 3.4	38.2 ± 5.9
	AlexNet + Aug [270]	2017	70.1 ± 7.4	75.8 ± 5.4	73.6 ± 4.8	84.6 ± 1.8
	CSDCNN + Aug [270]	2017	92.8 ± 2.1	93.9 ± 1.9	93.7 ± 2.2	92.9± 2.7
	IRRCNN +w/o Aug.	2018	95.69 ± 1.18	95.37 ±1.29	95.61± 1.37	95.15 ± 1.24
	IRRCNN + w Aug.	2018	97.09 ±1.06	97.57 ±0.89	97.29 ±1.09	97.22 ±1.22
Patient Level	LeNet + Aug [270]	2017	48.2 ± 4.5	47.6 ± 7.5	45.5 ± 3.2	45.2 ± 8.2
	AlexNet + Aug [270]	2017	74.6 ± 7.1	73.8 ± 4.5	76.4 ± 7.4	79.2 ± 7.6
	CSDCNN + Aug [270]	2017	94.1 ± 2.1	93.2 ± 1.4	94.7 ± 3.6	93.5± 2.7
	IRRCNN +w/o Aug.	2018	95.81 ± 1.81	94.44 ± 1.3	95.61 ± 2.9	94.32± 2.1
	IRRCNN + Aug.	2018	96.76 ± 1.11	96.84±1.13	96.67±1.27	96.27±0.87

Table 9.4: Breast cancer classification results for binary classification (benign vs. malignant tumor) using the BreaKHis dataset.

	Method	Year	Classification Rate at Magnification Factor			
			40×	100×	200×	400×
Image Level	CNN +fusion [267]	2016	85.6 ± 4.8	83.5 ± 3.9	83.6 ± 1.9	80.8 ± 3.0
	AlexNet + Aug [270]	2017	85.6 ± 4.8	83.5± 3.9	83.1 ± 1.9	80.8 ± 3.0
	ASSVM [270]		94.97	93.62	94.54	94.42
	CSDCNN + Aug [270]	2017	95.80± 3.1	96.9 ± 1.9	96.7 ± 2.0	94.90 ± 2.8
	IRRCNN	**2018**	**97.16 ± 1.37**	**96.84 ±1.34**	**96.61± 1.31**	**95.78 ± 1.44**
	IRRCNN + Aug	**2018**	**97.95± 1.07**	**97.57± 1.05**	**97.32± 1.22**	**97.36± 1.02**
Patient Level	CNN +fusion	2016	90.0 ± 6.7	88.4 ± 4.8	84.6 ± 4.2	86.10 ± 6.2
	Bayramoglu et al. [266]	2016	83.08 ± 2.08	83.17 ± 3.51	84.63 ± 2.72	82.10 ± 4.42
	Multi-classifier. [265]	2017	87.2 ± 3.74	88.22 ± 3.23	88.89 ± 2.51	85.82 ± 3.81
	CSDCNN + Aug [270]	2017	92.8 ± 2.1	93.9 ± 1.9	93.7 ± 2.2	92.90 ± 2.7
	IRRCNN +wo aug.	2018	96.69 ±1.18	96.37 ±1.29	96.27±1 .57	96.15 ±1.61
	IRRCNN + w. Aug.	2018	**97.60± 1.17**	**97.65± 1.20**	**97.56± 1.07**	**97.62± 1.13**

The testing accuracy for multi-class and binary BC classification is shown in Table 9.3 and Table 9.4 respectively. In both cases, our IRRCNN based approaches show superior performance compared to existing DL based methods. In [267], the performance is analyzed with different fusion techniques including sum, product, and max. Thus, we compared against the highest accuracy reported in [267]. Our proposed method shows better performance in all cases.

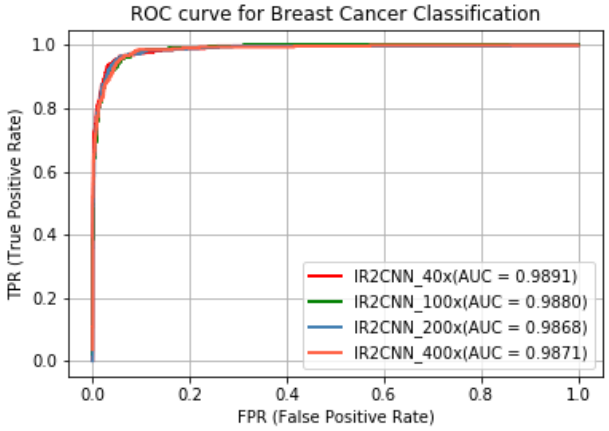
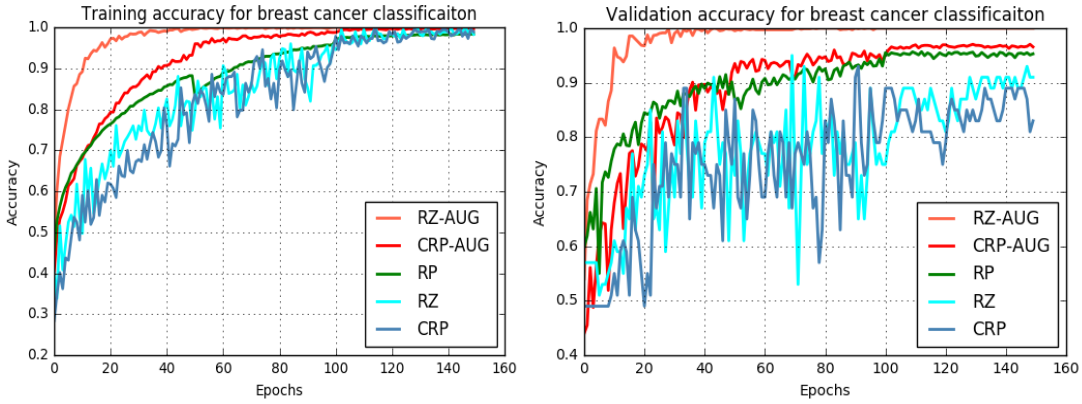


Figure 9.8: ROC curve with AUC for different magnification factors for eight class BC classification.



(a) (b)

Figure 9.9: Training and validation accuracy for the multi-class case using the 2015 BC Classification Challenge dataset. Sample sets are either resized and augmented (RZ+AUG), center patch cropped and augmented (CRP+AUG), random patches (RP), sample resized (RZ), or center patch cropped (CRP).

Results for Breast Cancer Classification Challenge 2015: For the 2015 BC Classification Challenge dataset, the training and validation accuracy for different methods are shown in Figs. 9 (a) and (b) respectively. The experimental results when using resized and augmented samples show the highest training and validation accuracy according to Figure 9.9.

Patch-wise classification results: The experimental results for different patch-based methods are shown in Tables 9.5 and 9.6. From the tables, for both binary and multi-class cases, the experiments with augmented center patches show the highest testing accuracy which is 97.51% and 97.11%

respectively. Similar performance is observed with random patches, but the experiments with single center patches show the lowest accuracies which are 88.7% and 88.12% for the binary and multi-class cases respectively.

Table 9.5: Performance for center patches (CRP), augmented CRP, and random patches (RP) for the binary class case.

CNN model	Methods	Year	Sensitivity	Specificity	Accuracy	AUC
CNN[17]			-	-	0.776	-
CNN+SVM[17]			-	-	0.769	-
IRRCNN	CRP	2018	0.8732	0.8812	0.887	0.9239
IRRCNN	CRP + Aug.	2018	0.9452	0.9829	0.9751	0.9925
IRRCNN	RP	2018	0.9307	0.9797	0.9676	0.9882

Table 9.6: Performance for center patches (CRP), augmented CRP, and random patches (RP) for the multi-class case.

CNN model	Criterions	Year	Sensitivity	Specificity	Accuracy	AUC
CNN[17]			-	-	0.667	-
CNN+SVM[17]			0.810	-	0.650	-
IRRCNN	CRP	2018	0.868	0.8733	0.8812	0.9169
IRRCNN	CRP + Aug.	2018	0.9371	0.9809	0.9711	0.9905
IRRCNN	RP	2018	0.9290	0.9752	0.9634	0.9824

Image-wise Classification results: The experimental results for the binary and multi-class cases are given in Tables 9.7 and 9.8 respectively. We achieved 100% testing accuracy for the experiment with random patches using the WTA method. In addition, the experiments with augmented resized samples show 99.05% and 98.59% testing accuracy for the binary and multi-class cases respectively. The lowest testing accuracy was observed for single resized images.

Table 9.7: Performance for image-wise breast cancer classification for the binary case.

CNN model	Criterion	Year	Sensitivity	Specificity	Accuracy	AUC
CNN[17]		2017	-	-	0.806	-
CNN+SVM[17]		2017	-	-	0.833	-
IRRCNN	RZ samples	2018	0.878	0.926	0.884	0.912
IRRCNN	RZ + Aug.	2018	0.9831	0.9912	0.9905	0.9932
IRRCNN	RP + WTA	2018	1.00	1.00	1.00	1.00

Table 9.8: Performance for image-wise breast cancer classification for the multi-class case.

CNN model	Criterion	Year	Sensitivity	Specificity	Accuracy	AUC
CNN[17]		2017	-	-	0.778	-
CNN+SVM[17]		2017	-	-	0.778	-
IRRCNN	RZ samples	2018	0.889	0.916	0.9204	0.917
IRRCNN	RZ + Aug.	2018	0.9771	0.9889	0.9859	0.9905
IRRCNN	RP + WTA	2018	1.00	1.00	1.00	1.00

9.4.5 Analysis and comparison against state-of-the-Art

Performance Analysis for the BreakHis Dataset: Most previous studies have reported classification results for benign and malignant cases [260,267,270]. However, some studies have shown results for the multi-class problem for breast cancer classification [267,270]. These experiments have been conducted for both binary and multi-class problems on samples with magnification factors of 40×, 100×, 200×, and 400×. Based on the BreakHis dataset, different feature-based approaches including PFTAS, GLCM, QDA, SVM, 1-NN, and RP were applied, and an accuracy of approximately 85% for patient-level analysis was reported [260]. In addition, AlexNet was used for binary breast cancer recognition at different magnification factors, and the highest recognition accuracy achieved was 95.6±4.8% for image level analysis and 90.0±6.7% for patient-level analysis [15]. Furthermore, the highest accuracies reported for classifying benign and malignant BC were 96.9±1.9% for the image level and 97.1±2.8% for the patient level [267]. For multi-class breast cancer classification, the best testing accuracies achieved were 93.9±1.9% and 94.7±3.9% for image level and patient level analysis respectively [267].

Alternatively, in this work, we achieved 97.95±1.07% and 97.65±1.20% testing accuracy for benign and malignant BC classification for image and patient level analysis. Therefore, we have achieved a 1.05% and 0.55% improvement in average performance against the highest accuracies reported for image and patient level analysis in [270]. Furthermore, our proposed IRRCNN model produced testing accuracies of 97.57±0.89% and 96.84±1.13% for multi-class BC classification at the image level and patient level respectively. These results are a 3.67% and 2.14% improvement of average recognition accuracy compared to the latest reported performance [270].

Performance Analysis for the 2015 BC Classification Challenge Dataset: In 2014, Crus-Roa et al. proposed a CNN based method for classification with a patch-based input, and they reported a sensitivity of 79.6% [273]. The highest accuracy that was reported in 2017 for four different types of breast cancers in the same dataset and the experiments were conducted for both binary and multi-class breast cancer classification problems. As the data dimensionality is high (2040×1536 pixels), both image-level and patch-level analyses have been conducted for binary and multi-class breast cancer classification. A CNN approach was used, and the best results were reported for image-level classification which was 77.8% and 83.3% testing accuracy for four and two classes respectively [269]. On the contrary, we have conducted an experiment based on the IRRCNN model considering different criteria including resizing, cropping, random patches, and different data augmentation techniques. For resized and augmented samples, we achieved 99.05% and 98.59% testing accuracy for binary and multi-class breast cancer recognition respectively. In addition, we achieved 100% testing performance for the experiment where the classification model is applied to random patches, followed by a winner take all method to produce the final results. Therefore, our method shows significant improvement in the state-of-the-art for both binary and multi-class breast cancer recognition on the 2015 BC Classification Challenge dataset. The computation times for these experiments are given in Table 9.9**.**

Table 9.9: Computational time per sample for the BC classification experiments.

Dataset	Method	Total Time (s)	Number of Samples	Time per Sample (s)
BreakHis	Image Based	72.06	8732	0.08
BCC dataset 2015	Image Based	45.72	50	0.9144
	Patch Based	75.97	8742	0.008

9.5 Conclusion

In this paper, we proposed binary and multi-class breast cancer recognition methods using the Inception Recurrent Residual Convolutional Neural Network (IRRCNN) model. The experiments

were conducted using the IRRCNN model on two different benchmark datasets including BreakHis, and the 2015 Breast Cancer Classification Challenge, and performance was evaluated using different performance metrics. The performance of the proposed method was evaluated via image level, patient level, image-based, and patch-based analysis. We considered different criteria such as the magnification factor, resized sample inputs, augmented patches and samples, and patch-based classification in this implementation. The proposed approach shows approximately 3.67% and 2.14% improvement of average recognition accuracy on the BreakHis dataset against all results published in scientific reports in 2016. In addition, this method shows 99.05% and 98.59% testing accuracy for binary and multi-class breast cancer recognition on the 2015 Breast Cancer Classification Challenge dataset, which is significantly higher than that of any other CNN based approach for image-based and patch-based recognition performance respectively. We have also evaluated the performance of the proposed method with random patches and Winner Take All (WTA) approaches for image-based recognition and achieved 100% testing accuracy. Thus, the experimental results show state-of-the-art testing accuracy for breast cancer recognition compared against existing methods for both datasets.

CHAPTER 10

NUCLEI CLASSIFICATION, SEGMENTATION, AND DETECTION

Due to cellular heterogeneity, cell nuclei classification, segmentation, and detection from pathological images are challenging tasks. In the last few years, Deep Convolutional Neural Networks (DCNN) approaches have been shown state-of-the-art (SOTA) performance on histopathological imaging in different studies. In this work, we have proposed different advanced DCNN models and evaluated for nuclei classification, segmentation, and detection. First, the Densely Connected Recurrent Convolutional Network (DCRN) model is used for nuclei classification. Second, Recurrent Residual U-Net (R2U-Net) is applied for nuclei segmentation. Third, the R2U-Net regression model which is named UD-Net is used for nuclei detection from pathological images. The experiments are conducted with different datasets including Routine Colon Cancer(RCC) classification and detection dataset, and Nuclei Segmentation Challenge 2018 dataset. The experimental results show that the proposed DCNN models provide superior performance compared to the existing approaches for nuclei classification, segmentation, and detection tasks. The results are evaluated with different performance metrics including precision, recall, Dice Coefficient (DC), Means Squared Errors (MSE), F1-score, and overall accuracy. We have achieved around 3.4% and 4.5% better F-1 score for nuclei classification and detection tasks compared to recently published DCNN based method. In addition, R2U-Net shows around 92.15% testing accuracy in term of DC. These improved methods will help for pathological practices for better quantitative analysis of nuclei in Whole Slide Images(WSI) which ultimately will help for better understanding of different types of cancer in clinical workflow.

10.1 Introduction

People all around the work is suffering from different diseases including cancer, heart diseases, chronically diseases, Brain-related diseases including Alzheimer's, and diabetes. Recently study shows that one of the very famous company named "Pfizer" which has been conducting research for developing a drug for Alzheimer and Parkinson's diseases going to stop working on new drugs to fight Alzheimer's disease and Parkinson's disease due to hugely expensive and longtime. On the other hand, according to the recent study shows that on an average to discover a new drug it takes around 12 years to come to the market [275].

Medical imaging speed up the assessment process of almost every disease from lung cancer to heart disease. The automatic nucleus classification, segmentation, and detection algorithm can help to unlock the cure faster from the critical disease like cancer to the common cold. To identify the cell's nuclei is the starting point to analysis about 30 trillion cells contain a nucleus full of DNA of the human body. Identifying the cell accurately can help the researcher to observe how to react the cell with respect to the different treatments. As a result, researchers can understand the underlying biological process of cell-level analysis at clinical workflow. This solution can help to ensure the better treatment of patients and can accelerate the treatment for the patient and the drug discovery process. Therefore, the computational pathology and microscopy images play a big role in decision making for disease diagnosis, since these images able to provide a wide range of information for computer-aided diagnosis (CAD), which enables quantitative and qualitative analysis of these images with a very high throughput rate. Nowadays, the computational pathology becomes very popular in the field of medical imaging research which can greatly benefit pathologist and patient, therefore this field significantly get attention from both research community and the community from clinical practice [276,277]

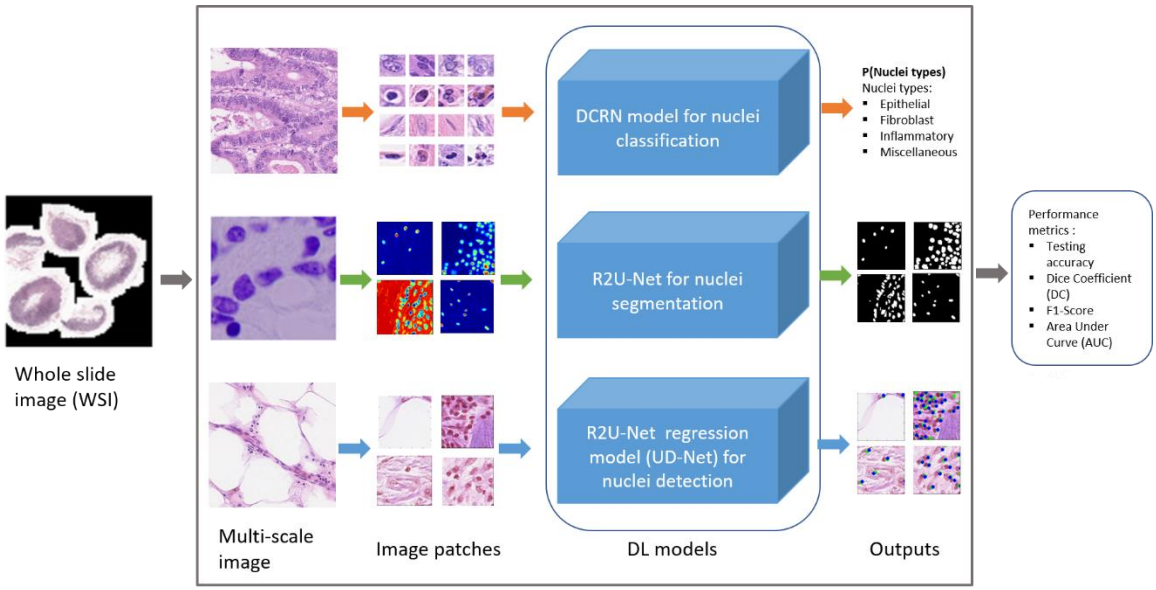


Figure 10.1: Overall proposed architecture: the microscopy WSI are acquired with different magnification factors, the patches are extracted from the multi-scale as required. Different DL models are applied for nuclei classification, segmentation, and detection. Eventually, the performance is evaluated with different performance metrics.

This computation approaches able to provides faster and efficient image analysis compare to the manual system for the researchers and clinician scientists which can release from difficult and repeated routine efforts [278]. Since the computational pathology and microscopic imaging is very challenging for manual image analysis, it might lead to large inter-observer variations [279]. On the other hand, CAD reduces the bias significantly and provide a characterization of diseases accurately [280]. Additionally, computational pathology gives a reproducible and rigorous measurement of pathological image features, can be used for clinical follow-up and helps to study personalized medicine and treatment which would significantly benefit patients.

As a prerequisite of clinical practice of CAD is nuclei classification, segmentation, and detection are considered as basically annotated image analysis method with DCNN. These techniques provide different quantitative analysis including cellular morphology, such as size, shape, color, texture, and other imagenomics. However, it is a very difficult task to achieve robust and accurate performance for mentioned three different challenging tasks for pathological imaging due to several

reasons. First, the pathologically and microscopy images contain background clutter with the noise, artifacts (image are blur sometimes), low signal to noise ratio (SNR), and poor depth resolution, which is usually happened during image acquisition with devices. Second, it contains low contrast between the foreground and the background of the images. Third, the difficult issue is the variant of the size, shape, and intercellular intensity of nuclei or cell. Fourth, it can be observed very often that the nuclei or cells are partially overlapped with one another. Meanwhile, there are several methods have been proposed to tackle these issues with automatic nuclei classification, segmentation, and detection for pathological imaging.

In the last few years, there are several surveys have been conducted on different methods and summarized the CAD technologies in the field of biomedical imaging including computational pathology [281]. These reviews briefly discuss technique related to pre-processing, nuclei classification, segmentation, and detection, and post-processing. One of the recently published paper discusses several techniques related to data acquisition and ground truth generation, image analysis, recognition, detection, segmentation, and statistics in terms of survival analysis in [282]. Another review has conducted on different approaches related to feature extraction, predictive modeling, and visualization from WSI in [283]. A survey conducted on nuclei detection, segmentation, and classification on hematoxylin and eosin (H&E) and immunohistochemistry (IHC) stained histopathology images. Along with traditional image processing and computer vision-based approaches, there are several surveys have been on deep learning-based approaches for pathological image analysis. Due to available annotated data and huge computing power, the Deep Learning(DL) approached Convolutional Neural Network (CNN) providing state-of-the-art-accuracy on different computer vision problems [1]. First, in a classification task, the target is to identify the class probability from the input samples. From example: for binary breast cancer recognition problem the system defines class whether the input sample is in the category of benign or malignant. Second, most of the cases the semantics segmentation techniques are used for deep learning-based methods which describe the process of associating each pixel of an image with a

class label. Another objective of this task is to define the proper contour of an object in the input image. Third, the DCNN based detection tasks, the objective is to identify the central or object rectangular coordinate of a certain object. Define the bounding box of an object is also the goal of this task. For example: in this implementation, identify the center pixel coordinate of nuclei from the input image. A recent study shows that the DL methods show a huge success in different modalities of medical imaging domain including mammographic mass classification, segmentation of lesions from neuroimaging, leak detection in airway tree, diabetic retinopathy, prostate segmentation, lung nodule detection, Breast cancer detection, x-ray imaging [3]. However, in this work, we have used applied three different improved DCNN models for nuclei classification, segmentation, and detection. The overall implementation diagram is shown in Figure 10.1. The contribution of this paper is summarized as follows:

- We have proposed an improved model named Densely Connected Recurrent Convolutional Network (DCRN) for nuclei classification.
- To generalize the R2U-Net model (UD-Net), this model is applied here for nuclei segmentation task in this implementation.
- The R2U-Net regression model is proposed for end-to-end nuclei detection from pathological images.
- The experiments have been conducted on three different publicly available datasets for nuclei classification, segmentation, and detection.
- The results show superior performance compared to existing machine learning and recently developed DL based approaches for nuclei classification, segmentation, and detection tasks.

10.2 Related Works

Automatic nuclei classification, segmentation, and detection is a prerequisite for various quantitative and qualitative analysis in computational pathology. The morphological features

compute for different disease including routine colon cancer, breast cancer, drug development and many more. In the last few years, there are different DCNN approaches has been proposed and successfully applied on medical image analysis problems and shows superior performance on different benchmarks dataset for classification, segmentation, and detection task [1]. There are several types of research ongoing in the field of digital pathology and trying to improve performance due to the complex nature of images. However, we have considered the nuclei classification, segmentation, and detection have been treated as a separate problem. The related works on traditional machine learning and deep learning-based nuclei classification, segmentation, and detection is as follow.

Classification: Nuclei classification can be used to various histopathology related applications. In the past, features including shape, texture, and size of nuclei are considered for nuclear pleomorphism grading in breast cancer images [284]. Malon et al. have applied CNN for classifying the mitotic and non-mitotic cells using color, shape and texture information in [285]. Cancerous nuclei are classified lymphocyte or stromal based on morphological features in H&E stained for breast cancer images and the method requires an accurate segmentation of tissue from the input samples which is explained in [286]. Another method with AdaBoost classifier where intensity, morphological and texture feature are used and the main focused of that work was on nuclei segmentation with classification approach [287]. However, recent studies have shown the deep learning-based approaches produce promising classification accuracy on large-scale pathological images. In 2014, Wang et al. used hand-crafted featured and applied cascaded ensemble CNN for detecting mitotic cells and achieved promising improvement result for nuclei classification task in [288]. Another deep learning based approach is proposed for cell classification and compared against a method with a bag of features and canonical representations in [289]. In 2017, the histopathological image classification approach is proposed and applied support vector machine (SVM), AdaBoost, and DCNN. The experiment is conducted on four different H&E stained image datasets namely prostate, breast, renal clear cell, and renal papillary cell cancer dataset. The results

demonstrate that Color-Encoder deep network achieves the best performance out of nine individual methods and they achieved around 91.2% testing accuracy in term of F1-score as highest testing accuracy [290].

Segmentation: a novel contour based "minimum-model" cell detection and segmentation approaches are proposed in 2012. That method uses minimal a priori information and detects contours independent of their shape and achieved promising segmentation results in [291]. Nuclei membrane segmentation with the CNN model is proposed from microscopic images in [292]. Ronneberger et al. proposed a CNN based approach called U-Net for general medical image segmentation in [32]. In addition, the U-Net has been applied range of segmentation problem including Nuclei segmentation. A learning-based framework for robust and automatics nucleus segmentation with shape preservation in pathological image, the CNN model generates the probability hit maps, on which an iterative region merging technique is applied for shape identification. In addition, a Nobel segmentation approach was exploited to separated individual nuclei combining a robust selection-based shares shape model and a local repulsive deformable model which have tested in several scenarios for pathological image segmentation and shows state-of-the-art performance against existing approaches till 2016 [293]. A very simple CNN model-based nuclei segmentation approach is proposed in 2017 which are named CNN2, and CNN3 models with respect to the number of output classes. For the two-class model, the network is used to classify pixel for inside and outside of the nuclei region respectively. On the other hand, the model for three classes, they were used for classifying pixels belong to inside, an output side, and the boundary of the nuclei regions in [294]. In 2017, Ho, D.J. et al. have proposed a fully 3D nuclei segmentation method using 3D CNN [295]. In 2018, another very promising deep learning-based one-step contour aware nuclei segmentation approach is proposed where a fully convolutional neural network is applied to segment the nuclei with their boundaries simultaneously [296].

A 3D Convolutional Network is used for joining cell Nuclei detection and simultaneously segmentation in microscopic images. The model is tested on two different datasets and achieved

state-of-the-art accuracy in detection and segmentation tasks [297]. However, for general medical image segmentation, an improved version of U-Net deep learning model has proposed in 2018 where recurrent residual modules are incorporated in U-Net instead of feedforward convolutional layers. The model was evaluated on different modalities of medical imaging including retina blood vessel segmentation, skin cancer segmentation, and lung segmentation. The experimental results are compared against U-Net, and SegNet and show superior testing performance [251]. To generalized of R2U-Net model, we have used R2U-Net model for end-to-end nuclei segmentation in this implementation. Along with that, the nuclei classification and detection methods are included as an extension of the primary results were published in 2018 [298].

Detection: Nowadays, there are two different methods are mainly used for nuclei detection. First, detection-based counting, which requires a prior detection or segmentation that discussed in [299]. On the other hand, density estimation-based method is used for nuclei detection without using segmentation which is explained in [300]. A framework with supervised max-pooling CNN is trained to detection cell pixels region which is preselected with Support Vector Machine (SVM). The method has shown outperformance compare to hand-crafted features-based approaches in [301]. For nuclei detection, a stacked sparse autoencoder based approach is used for non-nuclei and nuclei region detection with unsupervised fusion where a Softmax classifier is used in [302]. A CNN based regression model is used for nuclei detection and counting where a fully convolutional neural regression network model is used and able to density map for an input image of arbitrary size. They have used a patch-based method instead of end-to-end image method which is explained in [303]. However, we have proposed R2U-Net based regression model for end-to-end nuclei detection in this implementation.

10.3 Proposed Deep CNN Models

10.3.1 Densely Connected Recurrent Convolutional Network (DCRN)

According to the basic structure of Densely Connected Networks (DCN), the outputs from the prior layers are used as input for the subsequent layers. This architecture ensures the reuse the features inside the model, therefore it provides better performance on different computer vision tasks which in empirically investigated on different datasets in [27]. However, in this implementation, we have proposed an improved version of DCN which is named DCRN in short which is used for nuclei classification. The UD-Net is the building block of several Recurrent Connected Convolutional (DCRC) blocks and transition blocks. The pictorial representation of Densely Connected Recurrent Convolutional (DCRC) block is shown in Figure 10.2.

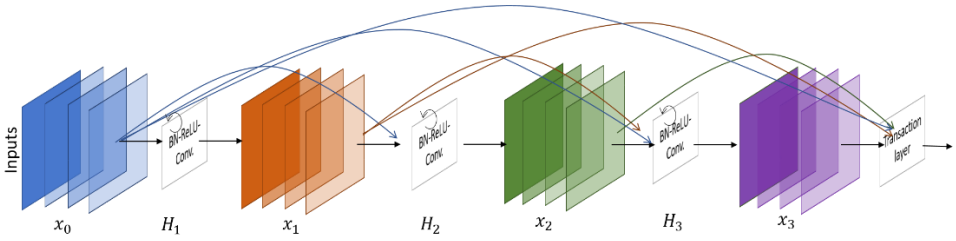


Figure 10.2: Densely Connected Recurrent Convolutional (DCRC) block.

According to the basic mathematical model of DenseNet which has explained in [34], the l^{th} layer receive all the feature maps $(x_0, x_1, x_2 \cdots x_{l-1})$ from the previous layers as input:

$$x_l = H_l([x_0, x_1, x_2 \cdots x_{l-1}]) \tag{10.1}$$

where $[x_0, x_1, x_2 \cdots x_{l-1}]$ are the concatenated features from $0, \cdots \cdots, l-1$ layers and $H_l(\cdot)$ is a single tensor. Let's consider the $H_l(\cdot)$ input sample from l^{th} DCRN block and contains $0, \cdots \cdots, F-1$ feature maps which are feed in the recurrent convolutional layers according to the method has proposed in [194,196]. This convolutional layer performs three consecutive operations which include Batch Normalization (BN), followed by ReLU and a 3×3 convolution (conv). Let's

consider a center pixel of a patch located at (i, j) in an input sample on the k^{th} feature of $H_{(l,k)}(\cdot)$. Additionally, let's assume the output of the network is $H_{lk}(t)$ for l^{th} layer and k^{th} feature maps at the time step t. The output can be expressed as follows:

$$H_{lk}(t) = \left(w^{f}_{(l,k)}\right)^{T} * H^{f(i,j)}_{(l,k)}(t) + \left(w^{r}_{(l,k)}\right)^{T} * H^{r(i,j)}_{(l,k)}(t-1) + b_{(l,k)} \quad (10.2)$$

Here $H^{f(i,j)}_{(l,k)}(t)$ and $H^{r(i,j)}_{(l,k)}(t-1)$ are the inputs to the standard convolution layers and the l^{th} recurrent convolution layers respectively. The $w^{f}_{(l,k)}$ and $w^{r}_{(l,k)}$ values are the weights of the standard convolutional layer and the recurrent convolutional layers of l^{th} layer and k^{th} feature map respectively, and $b_{(l,k)}$ is the bias. The recurrent convolution operations are performed with respect to t [52]. The pictorial representation of convolutional operation for $t = 2$ is shown in Figure 10.3.

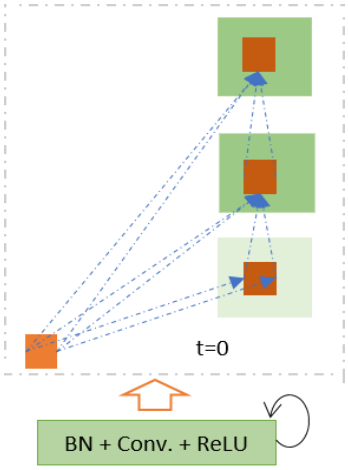


Figure 10.3: Unfolded recurrent convolutional units for t = 2.

In the transition block, 1×1 convolutional operations are performed with BN followed by 2×2 average pooling layer. The DenseNet model consists of several dense blocks with feedforward convolutional layers and transition blocks whereas the DCRN uses the same number of DCRC units and transition blocks. For both models, we have used 4 blocks, 3 layers per block, and the growth rate is 5 in this implementation and the model details are given in Table 10.1.

10.3.2 R2U-Net

we have applied the R2U-Net model for nuclei segmentation from microscopic images in [298]. The R2U-Net has been constructed with U-Net [32], Recurrent Convolutional Neural Networks (RCNN) [52], and Residual Network (Reset) [11]. The entire R2U-Net model is provided in Figure 104. This model consists of two main units which are encoding unit (shown in green) and a decoding unit (shown in blue). In both units, the recurrent residual convolutional operations are performed in the convolutional blocks.

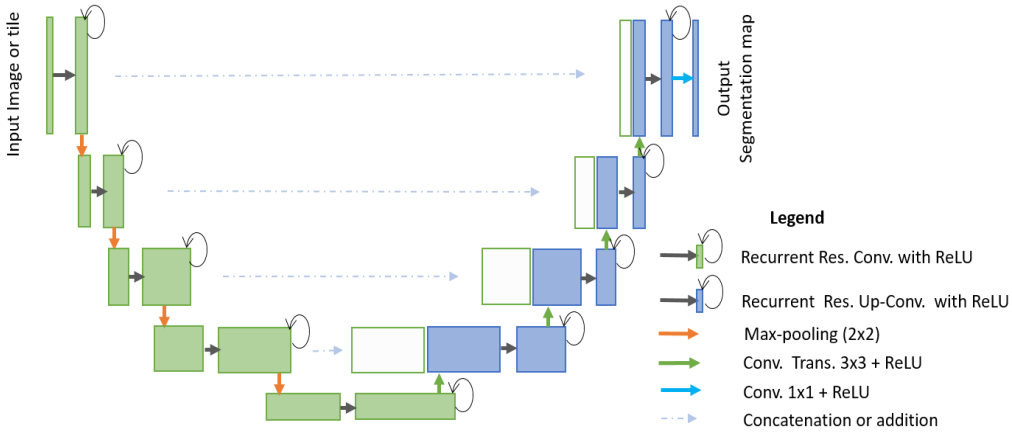


Figure 10.4: The end-to-end Nuclei segmentation method with R2U-Net model: green part refers the encoding unit, and blue part stands for decoding units. The features are concatenated from encoding units to the decoding units.

The conceptual diagram of the recurrent residual unit is shown in Figure 10.5. The recurrent operation is performed with respect to different time steps, which is shown in Figure 10.3. For the recurrent convolutional unit, t = 2, which means one forward convolution layer and two recurrent layers are used in this convolutional unit. The feature maps from the encoding unit are concatenated with the feature maps from decoding units. The softmax layer is used at the end of the model to calculate the pixel label class probability. For further details about R2U-Net, please see [251]. The model details and number of feature maps for this implementation are shown in Table 10.1.

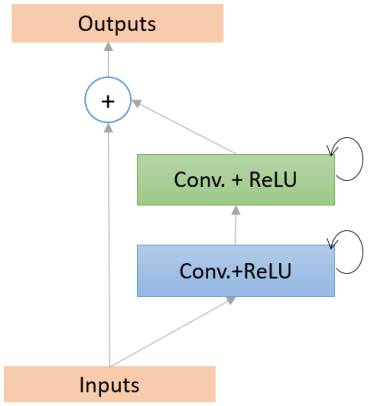


Figure 10.5: The recurrent residual unit (RRU) for R2U-Net.

10.3.3 Regression model with R2U-Net

In general, for cell detection and counting problem, the gourd truth is created with a single pixel annotation where the individual dot represents a cell. For example: the dataset, we have used in this implementation contains around at least five to five hundred nuclei with center pixel of the cell in input samples. For training with a regression model, each dot is represented with a Gaussian density. In case of the regression model, we have applied R2U-Net model to estimate the Gaussian densities from the input samples instead of computing the class or pixel level probability which is considered for DL based classification and segmentation model respectively. This model is named the University of Dayton Network shortly "UD-Net". For each input sample, a density surface D(x) is generated with superposition of these Gaussian. The objective is to regress this density surface for the corresponding input cell image I(x). The goal is achieved with R2U-Net model with the mean squared errors loss between the output heat maps and the target Gaussian density surface which is the ultimate loss function for the regression problem. However, in the inference phase, for the given input cell image I(x), the model R2U-Net computes the density heat maps D(x). The details on R2U-Net is described in the previous section, the network architecture and the number of network parameters are shown in Table 10.1.

Table 10.1: The model configuration and a number of network parameters utilize in this implementation.

Model	Tasks	t	Network architectures	Network parameters (million)
DenseNet	Classification	-	Blocks #4, layers#3, and growth rate # 5	0.582
DRCN	Classification	2	Blocks #4, layers#3, and growth rate # 5	0.582
R2U-Net	Segmentation	2	1→ 16→32→64→28→64→ 32→16→1	0.845
UD-Net	Detection	3	1→16→32→64→128→64 → 32→16→1	1.038

Network architectures: We have used similar architecture for DenseNet and DRCN models only difference between these two network models are the recurrent connectivity in the convolutional unit. In the case of DenseNet model, the two feedforward convolutional layers are used whereas for DRCN, we have two recurrent convolutional layers. For segmentation, we have used R2U-Net model with 0.84M the network parameters which is implemented for *t=2*. However, we have applied the DRCN model with *t=3* which increases the number of network parameters (1.038M).

10.4 Experiments and Results

To demonstrate the performance of the DCRCN, R2U-Net, and R2U-Net based regression (UD-Net) models, we have tested them for the nuclei classification, segmentation, and detection tasks. The dataset for classification and detection tasks are taken from [304] and segmentation dataset is downloaded from the 2018 Data Science Bowl Grand Challenge [310]. A discussion of these datasets is provided in the following sections. For this implementation, the Keras [171] and TensorFlow [274] frameworks were used on a single GPU machine with 56G of RAM and an NVIDIA GEFORCE GTX-980 Ti.

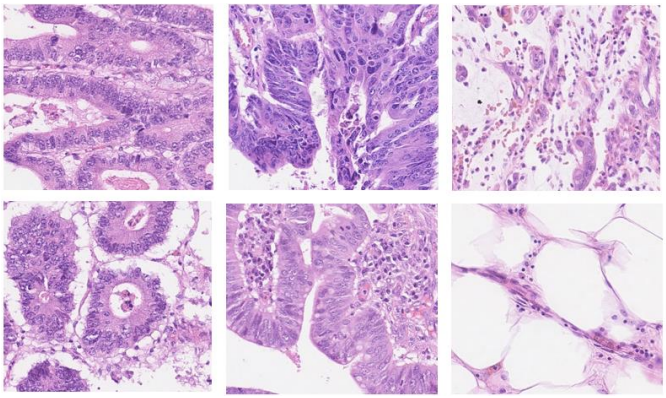


Figure 10.6: Example images from the dataset for Nuclei classification

10.4.1 Dataset for nuclei classification

This dataset contains 200 annotated samples for classification and detection tasks, where the total number of 100 samples are utilized for classification task and remaining 100 samples are used for detection task respectively. The actual sample size is 500x500 pixels. Some of the randomly selected database samples for nuclei classification are shown in Figure 6. In this implementation, we have selected 200 random patches from each sample, the selected patches size is 32x32 pixels. This dataset has four different classes of routine colon cancer including Epithelial, Fibroblast, Inflammatory, and miscellaneous. For the classification task, we have had total 20,000 samples [304]. The example patches with respective classes are given in Figure 7.

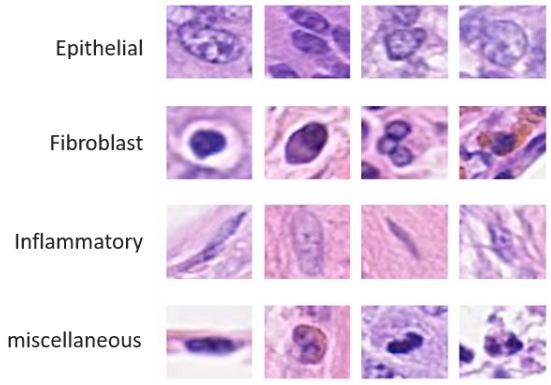


Figure 10.7: Image patches for four different routine colon cancer.

10.4.2 Dataset for nuclei segmentation

In 2018, Data Science Bowl has launched a competition with a mission to create an effective algorithm for automatic nucleus detection and segmentation. This work utilizes the dataset from 2018 Data Science Bowl grand challenges [310], which contains 735 images in total. The size of the database sample is 256×256 pixels. From the total image, 650 images and corresponding with pixel-level annotation masks are considered for training and remaining 65 samples are unlabeled to testing. From the training set, 80% of samples are used for training and the remaining 20% are used for validation during training. The number of training and validation samples are 536 and 134 respectively. The randomly selected some samples from the training set are shown in Figure 10.8. Figure 10.9 shows the normalized training samples in the first rows and corresponding label in the second row.

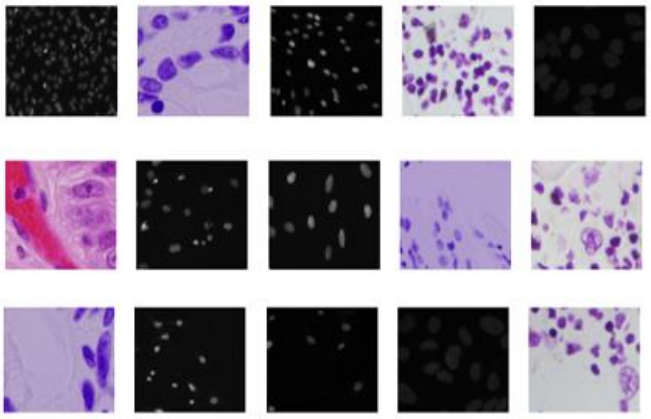


Figure 10.8: Randomly selected images from the dataset for segmentation.

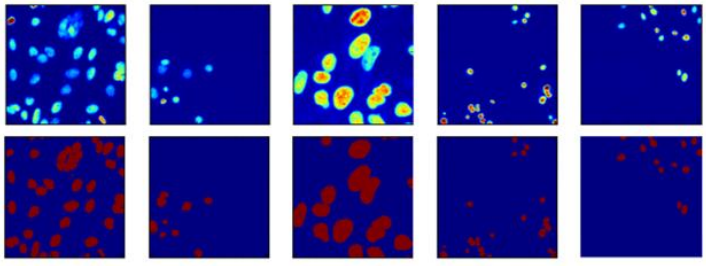


Figure 10.9: Example images with labels: first normalized samples and the second row shows the label of the corresponding images.

10.4.3 Database for nuclei detection

The database contains 100 samples and 100 masks with single pixel annotation [304]. The original size of the database sample is 500×500. For detection task, the nuclei cells are usually annotated with a single dot which is the center pixel of nuclei. For better understanding, some of the randomly selected samples and corresponding single pixel annotated masks are shown in Figure 10.10.

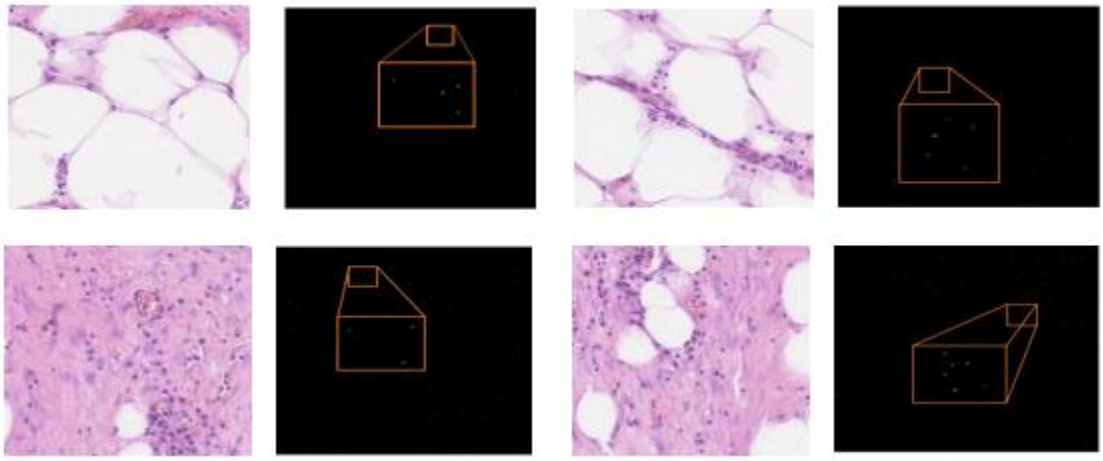


Figure 10.10: Example image with corresponding single pixel annotated masks for nuclei detection.

In this implementation, we have selected the non-overlapping patches (96×96 pixels) from original input samples and corresponding masks. We have extracted total 4,392 non-overlapping patches. From these patches, around 80% patches are used for training and the remaining 20% of samples are used for testing.

10.4.4 Training methods

To train models for the classification task, we have applied DenseNet and DCRN with the same architecture and network parameters. In both cases, we have used stochastic gradient descent(SGD) optimization method with a learning rate of 0.001, weight decay 1×10^{-4}, momentum 0.9 and cross entropy loss. The entire model is trained for 100 epochs with batch size 32. For the segmentation task, we have applied the Dice Coefficient (DC) and Mean Squared Error (MSE) loss function in

this implementation. The DC is expressed in equation (10.3), where GT refers to the ground truth and SR refers to the segmentation result.

$$\mathrm{DC} = 2\,\frac{|\mathrm{GT} \cap \mathrm{SR}|}{|\mathrm{GT}| + |\mathrm{SR}|} \tag{10.3}$$

Another metric used to evaluate the performance of the segmentation algorithm is the MSE as defined in equation (2). In this case, Y represents desired outputs and $\widehat{\mathrm{Y}}$ represents the predicted outputs. For an input sample with height *h* and width *w* and $n=h\times w$.

$$\mathrm{MSE} = \frac{1}{\mathrm{n}} \sum_{\mathrm{i}=1}^{\mathrm{n}} \left(\mathrm{Y_i} - \widehat{\mathrm{Y}}_\mathrm{i}\right)^2 \tag{10.4}$$

We considered 250 epochs and used Adam optimizer with a learning rate of 2×10^{-4} and the batch size is 16. Finally, for detection task with UD-Net regression model, we have used Adam optimizer with learning rate 2×e-4 and means squared errors (MSE). The model is trained with 500 epochs and batch size 64.

10.4.5 Results and discussion

10.4.5.1 Nuclei classification

We have tested models with a completely separated dataset from training dataset and achieved 90.86% and 91.14% testing accuracy with DenseNet and DCRN models respectively. The experimental results and comparison against the existing approaches are shown in Table 10.2. This table demonstrates that DRCN shows superior performance compared to existing methods.

Table 10.2: Nuclei classification accuracy and compared against other methods.

Methods	Average F1-score	AUC	Accuracy
CRImage [305]	0.488	0.684	-
Super-pixel descriptor [305]	0.687	0.853	-
SoftMax CNN + SSPP [304]	0.748	0.893	-
SoftMax CNN + NEP [304]	0.784	0.917	-
DenseNet [27]	0.8121	0.958	0.9086
Proposed (DRCN)	0.8180	0.9615	0.9114

The area under Receiver Operating Characteristic (ROC) curve for both DenseNet and DRCN is shown in Figure 10.11. From this figure, it can be clearly seen that DRCN provides show higher Area Under Curve (AUC) with similar network architecture and the same number of network parameters which clearly demonstrates the robustness of our proposed model over DenseNet for nuclei classification.

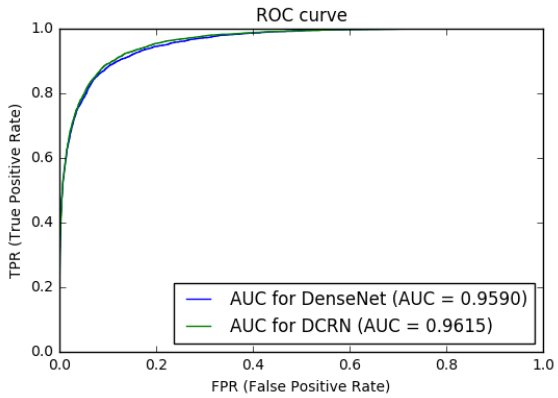


Figure 10.11: Area under ROC curve for DenseNet and DCRN models.

10.4.5.2 Nuclei segmentation

In this implementation, we used a simple R2U-Net model where only 0.84 million network parameters were utilized. The network architecture along with a number of feature parameters is shown in Table 10.1.

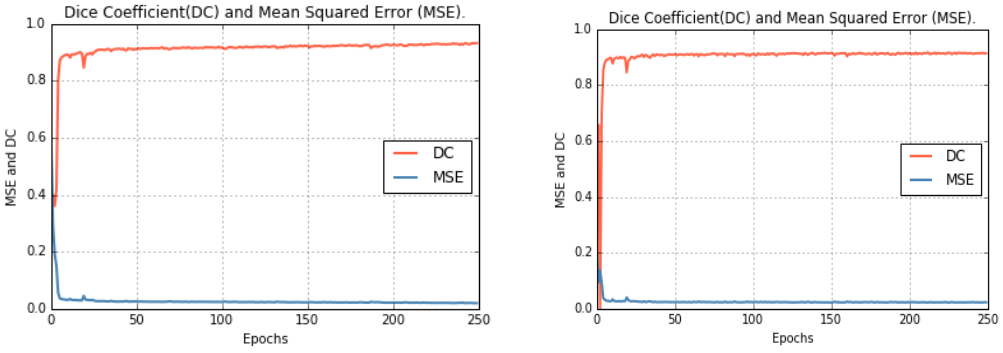


Figure 10.12: Training and validation accuracy in term of the Dice Coefficient (DC) on the left and Mean Squared Error (MSE) on the right for 250 epochs.

We considered the DC and MSE for observing training progress and performance in the training and testing phases. Figure 10.12(a) shows the DC and MSE with respect to the number of epochs during training. The validation DC and MSE are shown in Figure 10.12(b). From these figures, it can be observed that the model converged after 100 epochs, the training and evaluation continued until 250 epochs were completed to ensure optimum convergence. From this experiment, we achieved approximately 92.15% testing accuracy for nuclei segmentation with the R2U-Net model.

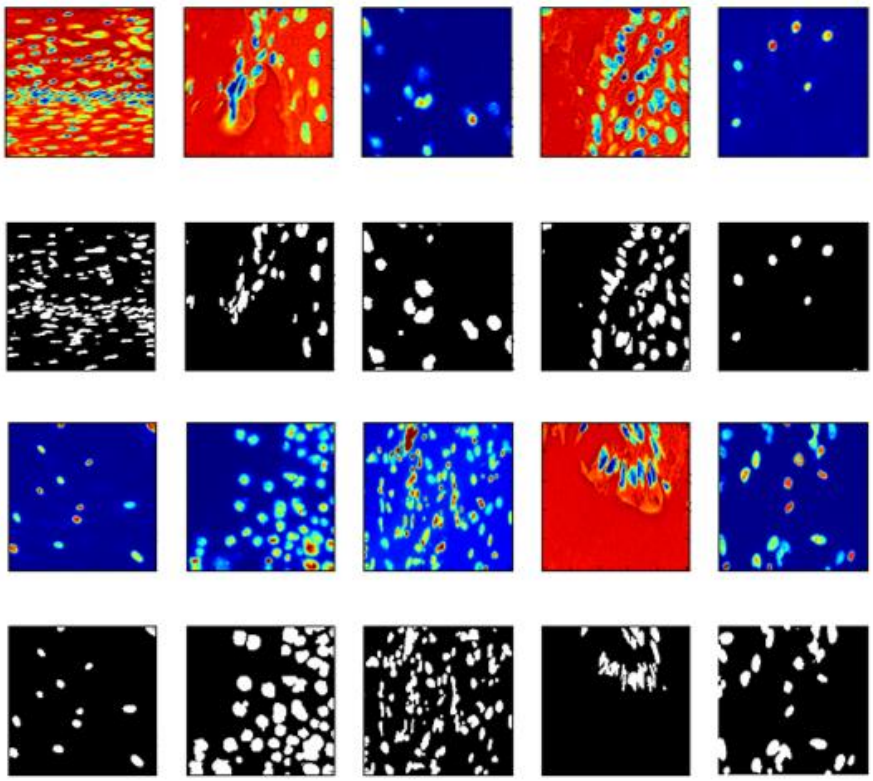


Figure 10.13: Qualitative experimental outputs in the testing phase with R2U-Net: first and third rows show the testing inputs and second and fourth rows show outputs samples.

Qualitative results: Figure 10.13 shows some example output samples when using the R2U-Net model for nuclei segmentation where the first and third rows show the input samples. Likewise, the second and fourth rows represent the corresponding network outputs. Based on these results, our proposed segmentation model provides promising segmentation outputs during the testing phase. In addition, if we observe the input samples in the first and second columns of the first row, there

is a strong separation between nuclei and similar objects of less interest. Similar behavior is displayed in the input sample in the first row of the fourth column. The input shown in the third row of the fourth column contains a complex background. However, in all cases, the model shows almost similar segmentation outputs with respect to the desired outputs. This result clearly demonstrates the robustness of R2U-Net for nuclei segmentation from pathological images.

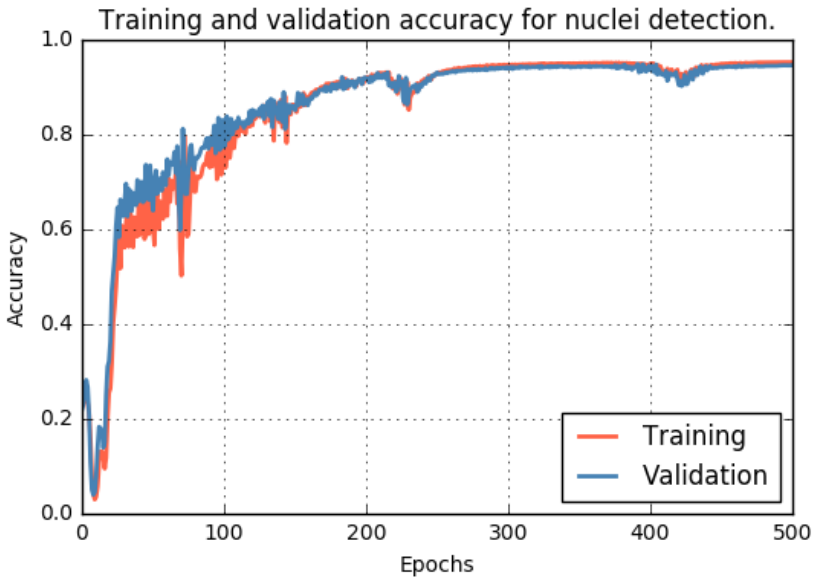


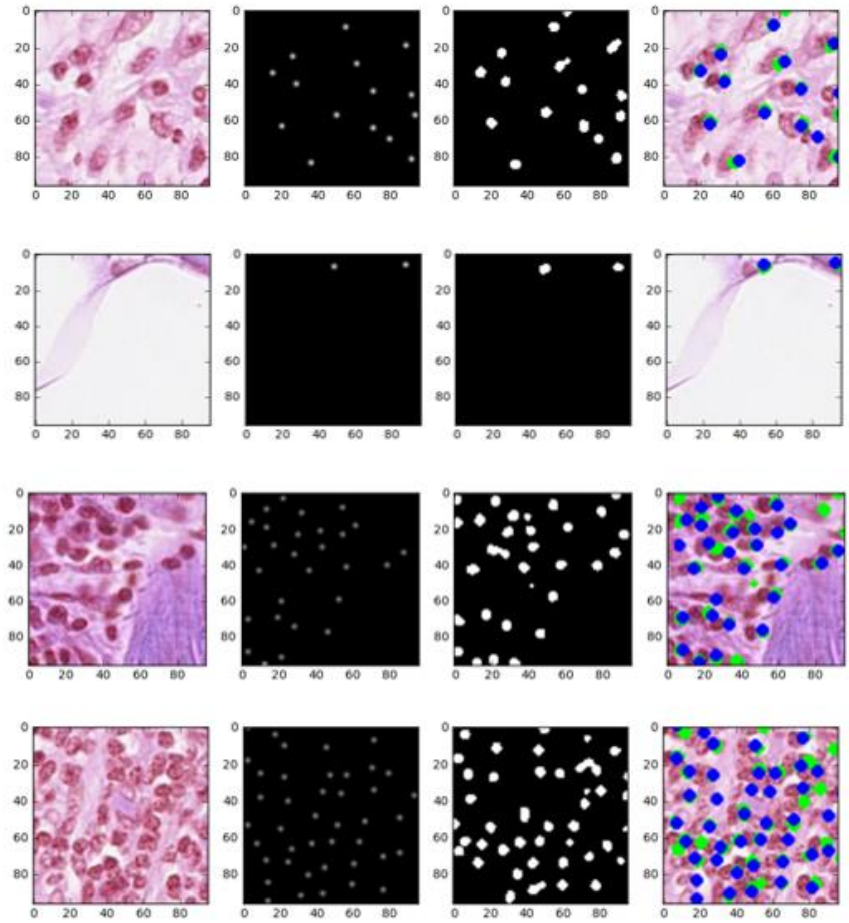
Figure 10.14: Training and validation accuracy of R2U-Net based regression model for nuclei detection.

Table 10.3: Nuclei detection accuracy and compared against other methods.

Methods	Precision	Recall	Average F1-score
CRImage [305]	0.657	0.461	0.542
CNN[305]	0.783	0.804	0.793
SSAE [308]	0.617	0.644	0.630
LIPSyM [309]	0.725	0.517	0.604
SC-CNN [304] (M=1)	0.758	0.827	0.791
SC-CNN [304] (M=2)	0.781	0.823	0.802
Proposed (UD-Net)	0.821	0.842	0.832

10.4.5.3 Nuclei detection

In this experiment, we have considered a patch-based approach, the experimental results are shown in Table 10.3. The recently published paper has reported 0.802 F1-score as the highest testing accuracy for nuclei detection in 2016 whereas our proposed model shows 0.8318 for nuclei detection task which is around 2.98% higher performance over the recently published SC-CNN model in [39].



Inputs ground truth model output final outputs

Figure 10.15: Nuclei detection outputs with inputs, ground truth, network outputs, and final outputs with a blue and green dot. The blue dot represents the center pixel of ground truth and green dot shows center pixels of the network outputs.

The quantitative results for nuclei detection are shown in Figure 10.15. The result shows promising detection accuracy for input patches. The first column shows the input patches, the second column shows the inputs label masks, third rows represents the model outputs, and the fourth column shows the final outputs with blue and green dots. Here the blue dot indicates the ground truth and the green dot represents the model outputs. The quantitative results clearly demonstrate that the UD-Net able to detect the nuclei accurately.

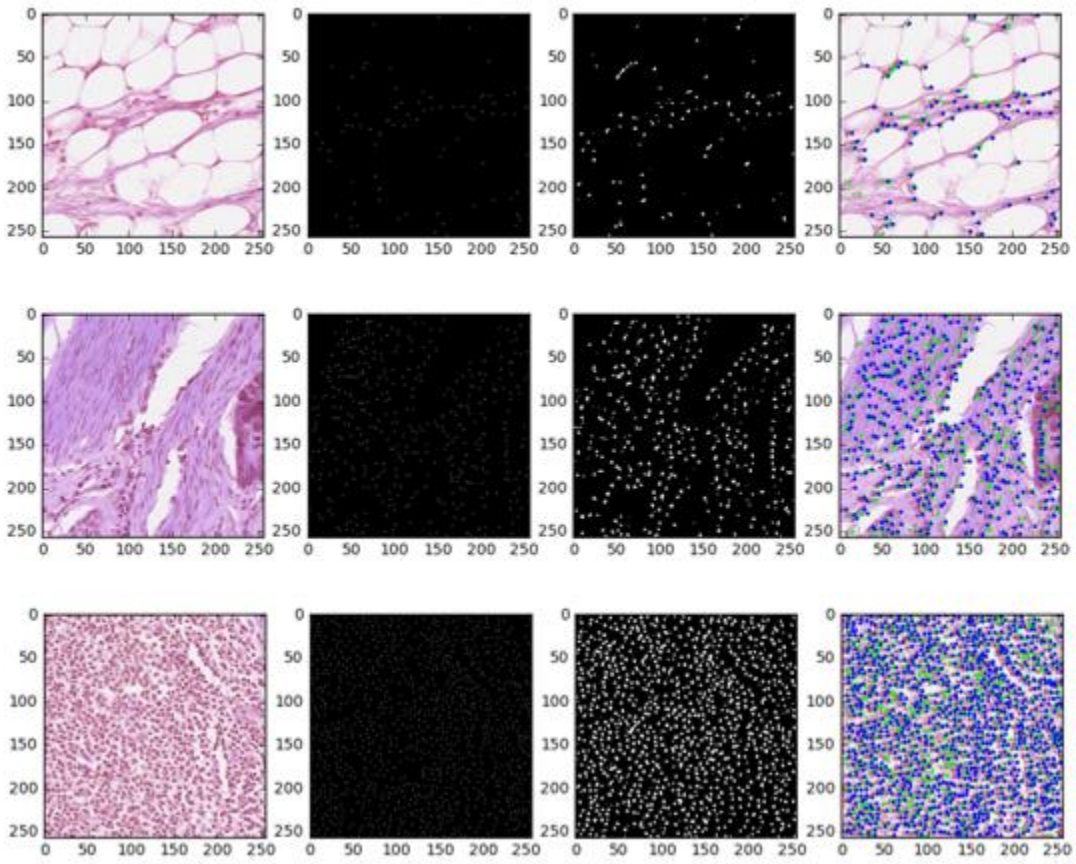


Figure 10.16: Nuclei detection outputs for entire samples which are generated from output patches.

Entire Images based outputs: after generating the patch-based outputs, we have merged all the patches together to generate the entire input image and corresponding outputs. The Figure 10.16, shows the output for entire input image where the first column shows the inputs, the second column is for ground truth Gaussian density surface, the third column represents the model outputs and the

fourth column shows the final output puts with a blue and green dot for each nucleus. The blue dot represents the ground truth and the green dot represents the model's outputs.

10.4.6 Analysis

We have solved three important tasks for computational pathology: nuclei classification, segmentation, and detection. In classification, we have applied DenseNet and improved version of DenseNet is named DCRN. The DenseNet provides 0.8121 and 0.958 performance in term of F1-score and AUC whereas the proposed DCRN provides around 0.8180 and 0.9615 for F1-score and AUC. The DCRN provides around 3.4% and 4.45% better performance for F1-score and AUC against recently published results in [304]. In addition, our proposed method shows 91.17% testing accuracy which around 0.3% better compared to DenseNet. Second, we have used very simple R2U-Net where only 0.84 million (M) network parameters for nuclei segmentation, we have achieved 92.15% testing accuracy on the publicly available dataset. Third, R2U-Net based regression model is used for Nuclei detection task and achieved 82.17%, 84.23% and 83.2% for precision, recall, and F1-score respectively. Overall, our models provide superior performance for three tasks. The testing time per sample for classification, segmentation, and detection tasks are shown in Table 10.4.

Table 10.4: Computational testing time of DCRN, R2U-Net, and UD-Net models in second.

Model	Tasks	Computational time/epoch (in sec.)
DCRN	Classification	0.0017
R2U-Net	Segmentation	0.40239
UD-Net	Detection	3.19906

10.5 Conclusion

In this study, we have proposed three different models including Densely Connected Recurrent Convolutional Networks (DCRN), Recurrent Residual U-Net (R2U-Net), and R2U-Net based regression (UD-Net) model for nuclei classification, segmentation, and detection tasks

respectively. These models have evaluated on publicly available three different datasets. We have achieved 91.14% testing accuracy with DCRN for nuclei classification task and achieved 3.4% and 4.45% higher than the average F1-score and AUC compare to recently published DL based method a softmax Convolutional Neural Networks(CNN) and neighboring ensemble predictor (NEP) which is called softmax CNN+NEP. The proposed R2U-Net model is applied for Nuclei segmentation task and shows 92.15% testing accuracy. For detection task, we have achieved 83.2% testing accuracy in term of F1-scope with R2U-Net regression model shows around 3% better F1-score compared to the existing methods. In the future, we would like to explore and evaluate these models on other datasets in the field of computational pathology.

CHAPTER 11

USE CASES FOR COMPUTATIONAL PATHOLOGY

11.1 Introduction

Deep Learning (DL) approaches have been providing state-of-the-art performance in different modalities in the field of Bio-medical imagining including Digital Pathology Image Analysis (DPIA). Out of many different DL approaches, Deep Convolutional Neural Network (DCNN) technique provides superior performance for classification, segmentation, and detection tasks in DPIA. Most of the objectives of DPIA problems are somehow solvable with classification, segmentation, and detection tasks. In addition, sometimes pre- and post-processing steps are required for solving some specific type of problems. In the last few years, different advanced DCNN models including Inception residual recurrent CNN (IRRCNN), Densely Connected Recurrent Convolution Network (DCRCN), and Recurrent Residual U-Net (R2U-Net) have been proposed which provide state-of-the-art performance for different computer vision and Bio-medical image analysis problems against existing DCNN models. However, these advanced DCNN models have not been explored massively for solving different problems related to DPIA.

11.2 Methods

In this study, we have applied IRRCNN, DCRCN, and R2U-Net advanced DCNN techniques for solving different DPIA problems that are evaluated on publicly available benchmark datasets which related to seven unique tasks in computational pathology. These tasks include: Invasive ductal

carcinoma (IDC) detection, Lymphoma classification, Epithelium segmentation, Tubule segmentation, Nuclei segmentation, Lymphocyte detection, and Mitosis detection.

11.3 Results

The experimental results are evaluated considering different performance metrics including: precision, recall, accuracy, F1-score, Area under Receiver Operating Characteristics (ROC) curve, dice coefficient (DC), and Intersection over Union (IoU) [222].

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+FP+TN+FN}} \tag{11.1}$$

$$\text{F1} - \text{score} = \frac{2\text{TP}}{2\text{TP+FP+FN}} \tag{11.2}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \tag{11.3}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \tag{11.4}$$

The results demonstrate superior performance for classification, segmentation, and detection tasks compared to existing DCNN based approaches.

11.3.1 Lymphoma classification

Even the expert pathologist sometimes phases difficulties to differential sub-type of H&E. To ensure better and consistent diagnosis of different diseases sub-type of H&E classification is very significant in the field of digital pathology.

In this implementation, there are three different Lymphoma subtypes are considered to classify from pathological images including: Chronic lymphocytic leukemia (CLL), Follicular lymphoma (FL), and Mantle cell lymphoma (MCL). The following Figure 11.1 shows three different type of cancer cell

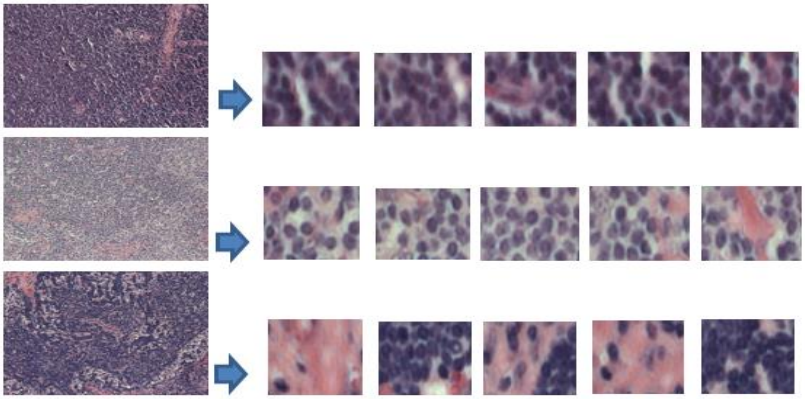
(a) (b) (c)

Figure 11.1: Three different cancer cells: (a) CLL (b) FL and (c) MCL.

Lymphoma classification dataset: The original size of the image is 1338×1040 pixels, the images are downsampled to 1344×1024 to crop the non-overlapping and sequential patches of size: 64x64 non-overlapping. The actual database sample and patches from the original images are shown in Figure 11.2. The statistics of original dataset and the number of samples after extracting non-overlapping patches are shown in Table 11.1.

Table 11.1: The statistics for the dataset of lymphoma classification.

Method	CALL	FL	MCL	Total samples
Entire image	113	139	122	374
Patches (64x64)	20,250	24,750	22,500	67,500



Inputs non-overlapping patches from original samples

Figure 11.2: The original samples in the left for CLL, FL, and MCL in the first second and third rows respectively. The right side shows the patches from the original images.

In this implementation, we have evaluated the performance of IRRCNN model with two different approaches: entire image-based approach, patch-based approach [196]. In image-based approach, the original samples are resized to 256×256. During training of the IRRCNN model, we have considered 8 and 32 samples per batch for image-based and patch-based method respectively. The Stochastics Gradient Descent (SGD) optimization method is used with an initial learning rate of 0.01. We have trained the model for only 40 epochs where after 20 epochs the learning rate is decreased with the factor of 10. The training and validation accuracy for lymphoma classification is shown in Figure 11.3.

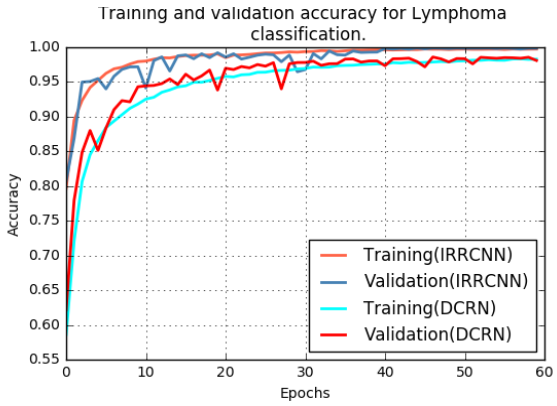


Figure 11.3: The training and validation accuracy for lymphoma classification with IRRCNN and DCRN.

Results for Lymphoma classification: After successfully training the model, the testing accuracy is computed with a testing dataset which is totally different samples from training samples. We have achieved around 92.12% and 99.8% testing accuracy for the entire image based and patch-based method which is shown in Table 11.2. From this evaluation, it can conclude that as the number of samples increases, the performance of the DL approach increases significantly. The highest accuracy is achieved in the patch-based method which is around 3.22% better performance compared to existing deep learning-based approach for Lymphoma classification in [311]

Table 11.2: Testing accuracy for Lymphoma classification for image and patch-based methods.

Methods	F1-score	ROC curve	Accuracy
DCNN in [311]	-	-	0.9659
DCRN [27]	0.9810	0.996	0.9873
IRRCNN(patches)	0.9969	0.999	0.9979

The area under ROC curve for image and patch-based method is shown in Figure 11.4.

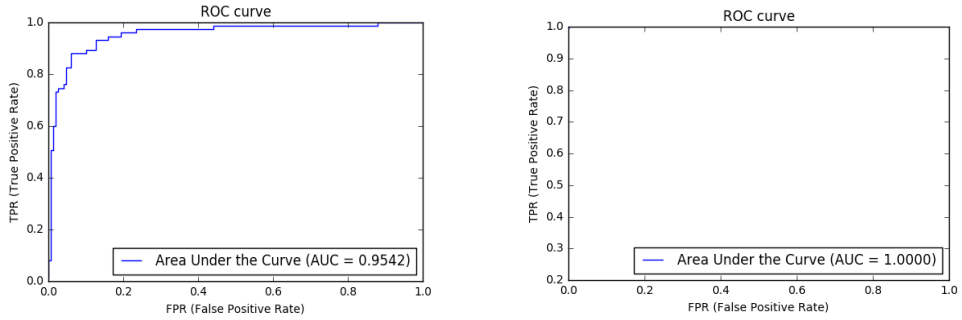


Figure 11.4: Area under ROC curve: left image is for an image-based method and the right image is for the patch-based method.

The confusion matrix with respect to the input patches of three different classes of Lymphoma is given in Table 11.3. A total number of testing patches is 13,500.

Table 11.3: Confusion matrix for lymphoma classification with respect to the number of patches

	CALL	FL	MCL
CALL	4051	3	2
FL	8	4889	0
MCL	14	14	4519

11.3.2 Invasive ductal carcinoma (IDC) detection

One of the very common types of breast cancer is IDC and most of the times pathologist focus on regions to identify IDC cancer. A common preprocessing step called automatic aggressiveness grading method is used to define the exact region of IDC from whole slides images.

Samples without IDC tissue Samples with IDC tissue

Figure 11.5: Examples samples from the database: left images show tissue without IDC and right images show tissue with IDC.

Method: we have used IRRCNN model with four IRRCNN and transition blocks in this implementation [196].

Experiment and discussion: the database is used from the recently published paper in [311]. The samples are in the database down sampled their original images x40 by the factor of 16:1 for an apparent magnification of x2.5. Since the data samples size is 50×50 pixels and pre-processed which is used in [311]. Therefore, we have considered entire images where the input samples are resized to 48×48 pixels. A total number of samples in the database are 275,223 where 196,455 samples are for the first one and remaining 78,768 samples for second class two. To resolve the class imbalance problem, we have randomly selected 78,000 from each class. The example samples are shown in Figure 11.6.

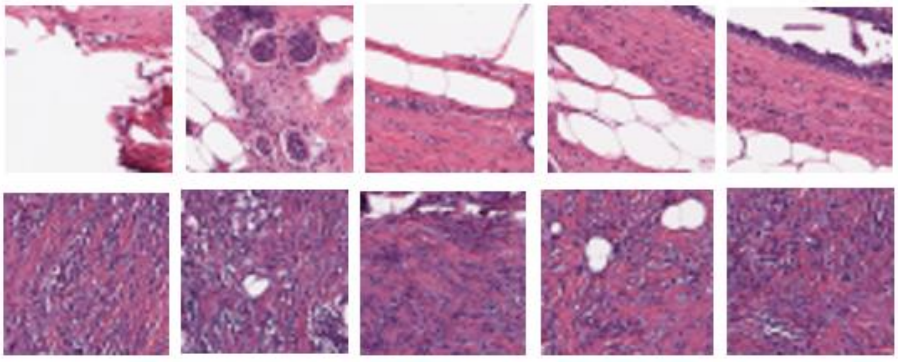


Figure 11.6: Randomly selected samples from the database. The images for the first class show in the first row and the second class is shown in the second row.

Experimental results**:** Stochastics Gradient Descent (SGD) is used with learning rate started at 0.01. The training is performed for 60 epochs, after 20 epochs learning rate decreases with the factor of 10. The training and validation accuracy for IDC classification is shown in Figure 11.7.

Figure 11.7: The training and validation accuracy for invasive ductal carcinoma classification.

IDC classification testing accuracy: we have evaluated the performance on testing database contains 31,508 samples. We have achieved around 89.06% F1-score and 89.07% testing accuracy. The testing results are shown in Table 11.4.

Table 11.4: Testing accuracy for invasive ductal carcinoma classification

Methods	F1-score	ROC curve	Accuracy
Original paper (AlexNet)	0.718	-	0.8423
AlexNet [311]	0.7648	-	0.8468
Patches based	0.8907	0.9573	0.8907

From this table, it can be clearly observed that around 4.39% better accuracy compared to existing latest published deep learning-based approach for Invasive ductal carcinoma (IDC) detection. The previous method provides results for 32×32 pixels where the resizing, center cropping, and center cropping with a different rotation. However, we did not apply any data augmentation techniques in

this implementation. In the testing phase, the testing result shows around 0.9573 as Area under the ROC curve which is shown in Figure 11.8. The total testing time for 31508 samples is 109.585 seconds. Therefore, testing time per sample is 0.0035 sec.

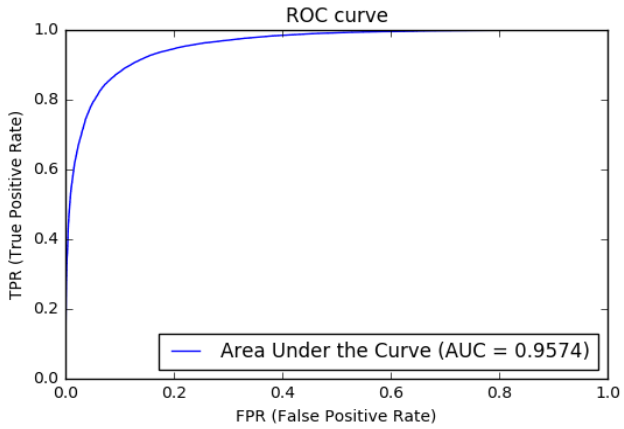


Figure 11.8: Area under ROC curve for invasive ductal classification.

The confusion matrix for testing samples is as follows:

	Class # 1	Class # 2
Class # 1	13,818	1849
Class # 2	3,445	14,245

11.3.3 Nuclei segmentation

Nuclei segmentation is a very import problem in the field of digital pathology for several reasons. First, nuclei morphology is a key component in most cancer grading shame, Second, and efficient nuclei segmentation techniques can significantly reduce the human effort for cell level analysis. Therefore, it can drastically reduce the cell analysis cost. However, there are several challenges to segment the nuclei region: first, finding accurate bounding box. Second, segment the overlapping nuclei. In this implementation, we have used R2U-Net (1→32→64→128→256→128→64→32→1) with 4M network parameters [251]

Database: In this implementation, we have a database from ISMI-2017 which is published in 2017. A total number of samples are 100 images with 100 annotated masks. The size of the sample is

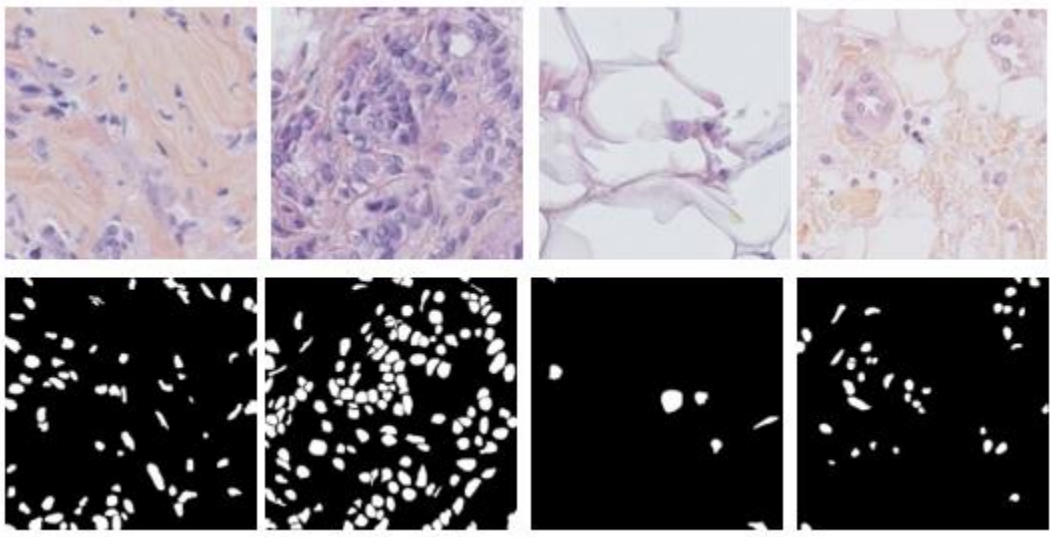


Figure 11.9: Randomly selected samples from nuclei segmentation dataset from ISMI-2017.

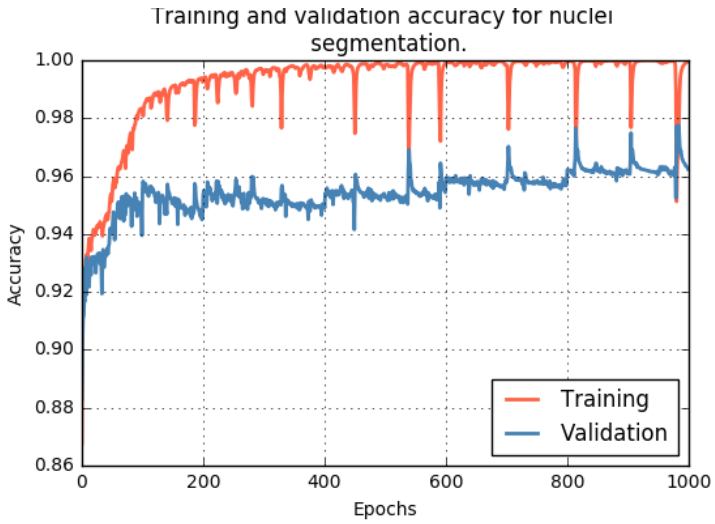


Figure 11.10: Training and validation accuracy for nuclei segmentation.

512x512. These samples are connected from 11 patients where each patient has a different number of the sample varies from 3 to 8 images. Some of the example images are shown in Figure 11.9. During training, one patient out method is used where randomly one patient is selected for testing and training is conducted on remaining ten patients. We have run ten times for as described in

[312]. We have applied Adam optimizer with learning rate 2xe-4 and cross entropy loss, batch size 2 and number of epochs 1000. We have trained the entire model with 1000 epochs and transfer learning approaches is used after 200 epochs. The training and validation accuracy for nuclei segmentation is shown in Figure 11.10. From the figure, it can be observed that the model shows very high accuracy for training, however, we have achieved around 98% accuracy during the validation phase.

However, in the testing phase, the proposed method shows around 97.70% testing accuracy on a testing dataset which is 20% of the total samples. The experimental results are shown in Table 11.5. From this table, it can be seen that we have observed around 3.31% better performance compared to existing deep learning-based approach for nuclei Segmentation on the same dataset.

Table 11.5: One patient output method-based testing accuracy for nuclei segmentation.

Methods	Recall	Precision	F1-score	Accuracy	ROC curve
PangNet[312]	0.655	o.814	0.676	0.924	-
DeconvNet[312]	0.773	0.864	0.805	0.954	-
Fully Convolutional Net (FCN)[312]	0.752	0.823	0.763	0.944	-
Ensemble[312]	0.900	0.741	0.802	0.944	-
Proposed approach (500 epochs)	0.8673	0.8654	0.8586	0.9643	0.9183
Proposed approach (1000 epochs)	0.9184	0.9097	0.9077	0.977	0.9464

Qualitative results for nuclei segmentation: The quantitative results for nuclei segmentation is shown in Figure 11.11. The first column shows the inputs images, the second column shows the ground truth, the third columns show the model outputs, and the fourth column shows the only nuclei on the inputs sample.

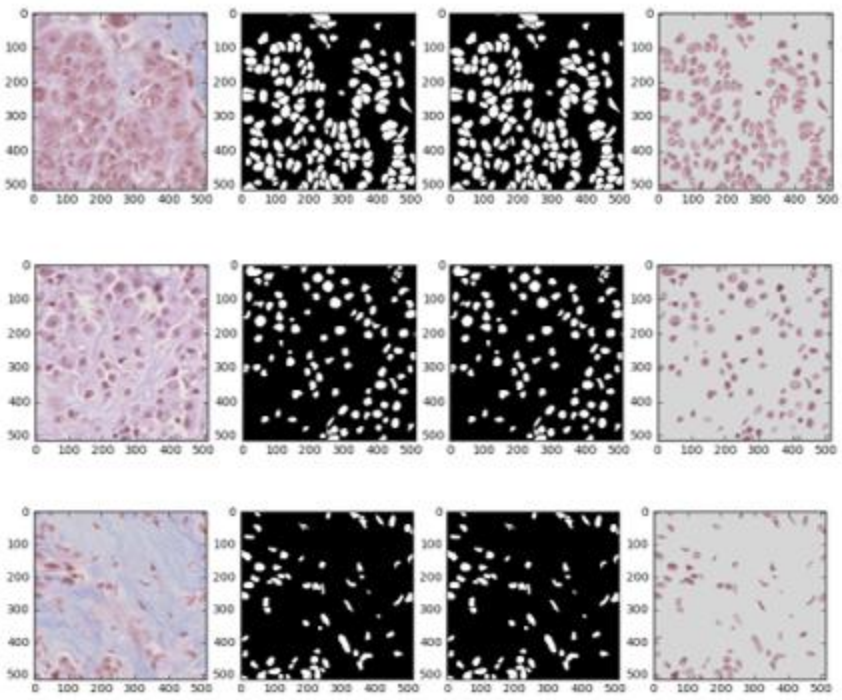


Figure 11.11: The experimental outputs for nuclei segmentation.

The R2U-Net is applied for nuclei segmentation from whole slide images (WSI) which is evaluated on ISMI-2017 dataset published in 2017. One patient out based approach is used for analyzing the accuracy and we have achieved 97.7% testing accuracy for nuclei segmentation which is around 3.31% better testing accuracy compared to the recently proposed DL based approach. Qualitative results demonstrate very accurate segmentation compared to ground truth

11.3.4. Epithelium segmentation

Most cases, the regions of cancer are manifested in the epithelium area, therefore epithelium and stoma regions are very important for identifying the cancer cancerous cells. It is very hard to predict the overall survival and outcome of breast cancer patients based on the histological pattern within the stroma region. Epithelium segmentation can help to identify the cancerous region where most of the cancer cells manifest. Epithelium segmentation samples:

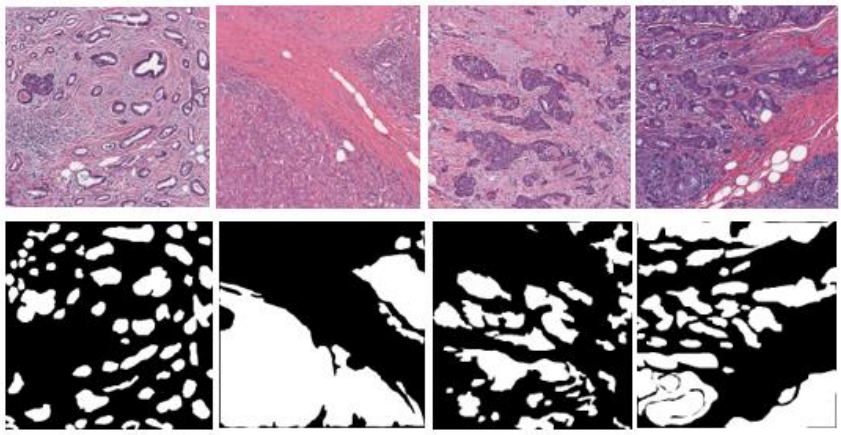
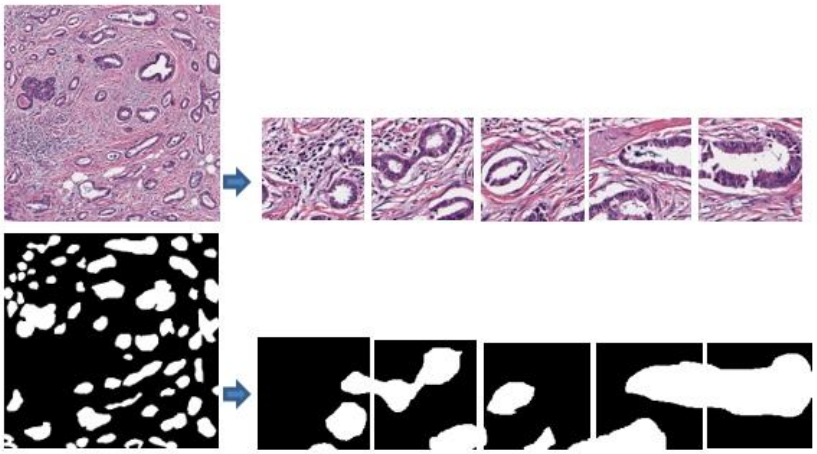


Figure 11.12: Example database samples from Epithelium segmentation. The first row shows the input samples and the second row shows the corresponding binary masks for input samples.

Method: we have used R2U-Net model which is consisted of encoding and decoding units. The total number of network parameters: 1.107 million [251].



(a) (b)

Figure 11.13: Database samples for epithelium segmentation: (a) input sample and ground truth of the corresponding samples are shown in the left side, (b) Extracted non-overlapping patches for input mages and output masks are shown on the right side.

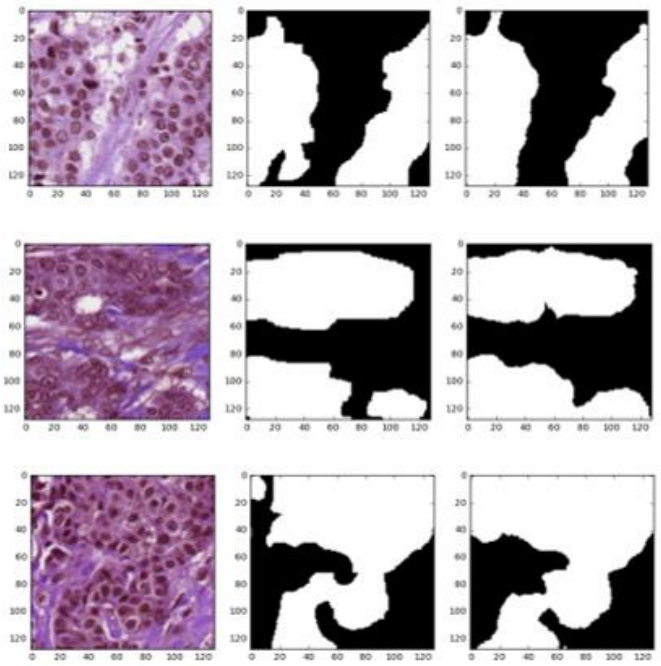
Database: This dataset was taken from the paper published in 2017 [311]. For epithelium segmentation, we had only 42 images in total. The size of the sample is 1000 x 1000 pixels. In this implementation, we have cropped non-overlapping patches with the size of 128x128. Therefore, the total number of patches: 11,780. Testing accuracies are shown in Table 11.6. We have used

80% (9,424) patches are used for training and the remaining 20% (2,356) are used for testing. The Adam optimizer is used with learning rate 2×e-4 and cross entropy loss. The experiment has been conducted with batch size 16 and number of epochs 150.

Table 11.6: Testing accuracy for epithelium segmentation.

Methods	F1-score	The area under ROC curve	Accuracy
AlexNet[311]	0.84	-	-
Patches based	0.9050	0.9202	0.9254

Experimental results: we have evaluated performance with different testing matrices. However, we have achieved around 92.54% testing accuracy and 90.5% for F1-score. Our method shows around 6.5% better performance compared to existing deep learning-based approach for Epithelium Segmentation.



Inputs Ground Truth Outputs

Figure 11.14: The experimental outputs for Epithelium segmentation.

Qualitative analysis: The quantitative results for Epithelium segmentation is shown in Figure 11.14. The first column shows inputs samples, the second column shows the ground truth, and the third column shows the model outputs. The area under ROC is shown in Figure 11.15 and we have achieved 92.02% area under the ROC curve for epithelium segmentation.

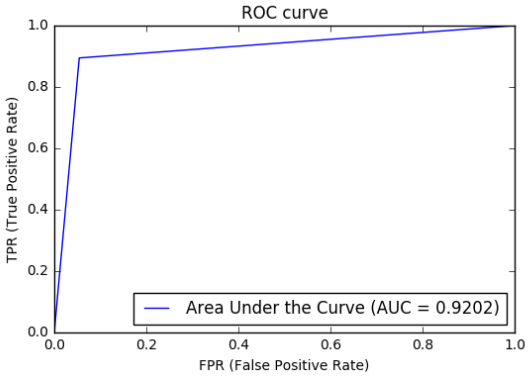


Figure 11.15: The area under the ROC curve for Epithelium segmentation.

The R2U-Net is applied for Epithelium segmentation from whole slide images (WSI). The experiment has been done with Epithelium segmentation dataset and achieved 90.50% and 92.54% for F1-score and accuracy respectively. The qualitative results demonstrate very accurate segmentation compared to ground truth

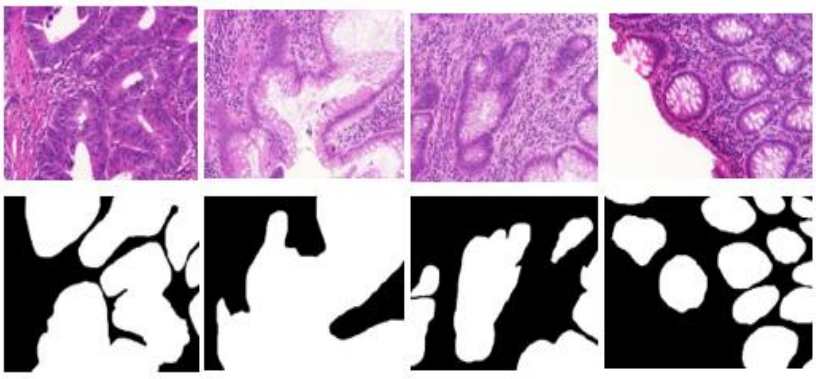
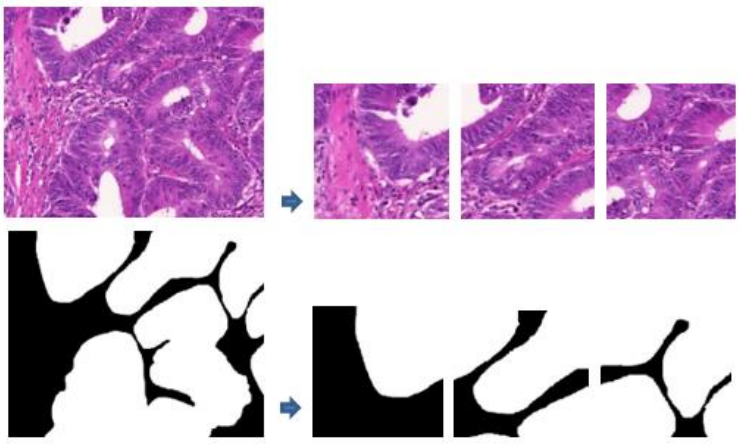


Figure 11.16: Database samples for Tubule segmentation.

11.3.5 Tubule segmentation

The aggressiveness of cancer can be determined based on the morphology of tubule from the pathological images. The tubule region becomes massively disorganized in the later stage of cancer. Network architecture: we have used R2U-Net model which is the end-to-end model consisted of encoding and decoding units [251]. The total number network parameters: 1.107 million.

Dataset: the database is taken from [311]. Total number samples in database: 42 and size of samples 775 x 522 pixels. As the number of samples is too low to train a deep learning approach. Some of the example samples are shown in Figure 11.16. Therefore, we have considered 256 x 256 non-overlapping patches and a total number of patches: 970. From these samples, 402 patches for benign and remaining 568 patches for malignant. Some of the example patches from an input sample are shown in Figure 11.17. We have used 80% patches are used for training and the remaining 20% are used for testing.



(a) (b)

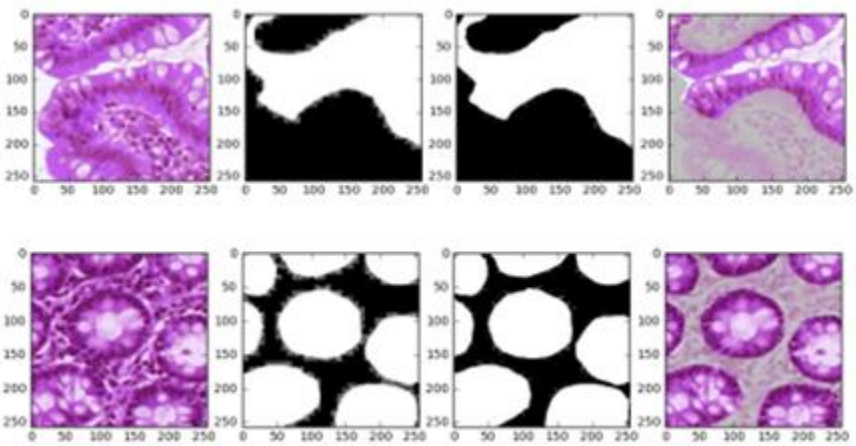
Figure 11.17: Database samples for tubule segmentation: (a) input sample and ground truth of the corresponding samples are shown in the left side, (b) Extracted non-overlapping patches for input mages and output mask are shown on the right side.

Experimental results: we have applied Adam optimizer with learning rate 2e-4 and cross entropy loss. Batch size 16 and number of epochs 500 are used during training for tubule segmentation.

Table 11.7: Testing results for tubule segmentation.

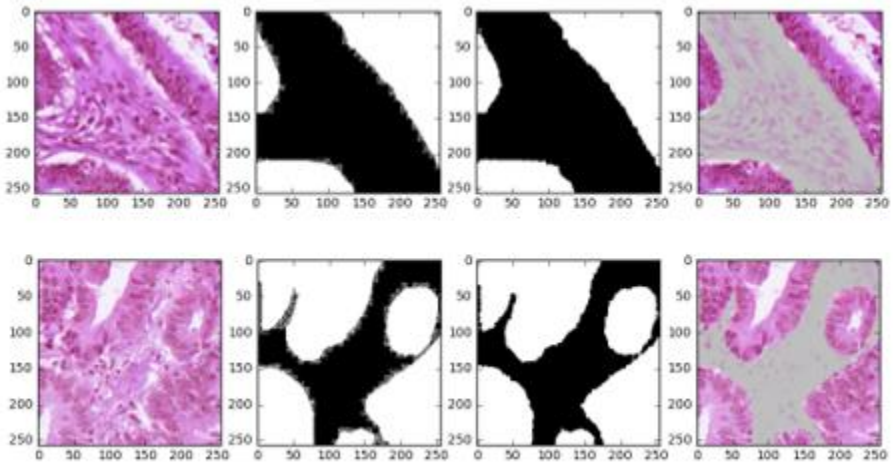
Methods	F1-score	ROC curve	Accuracy
AlexNet[311]	0.8365	-	-
Random polygons model [313]	0.8450	-	-
Domain knowledge-based approach[314]	0.8600	-	-
Proposed approach	0.9013	0.9045	0.9031

Testing accuracy: The testing accuracies and the comparison against the existing approaches are shown in Table 11.7. We have achieved around 90.31% testing accuracy and 90.13% for F1-score which is around 4.13% better performance compared to existing deep learning-based approach for Epithelium Segmentation. The quantitative results for benign are shown in Figure 11.18 and model outputs for malignant are shown in Figure 11.19.



Inputs Ground truth Outputs transparent outputs

Figure 11.18: The quantitive results for tubule segmentation. the first column shows the inputs samples, second columns show the label masks, the third column shows the model outputs and finally, the fourth column shows the only tubule part from benign images.



Inputs Ground Truth Outputs transparent outputs

Figure 11.19: The quantitive results for tubule segmentation. the first column shows the inputs samples, second columns show the label masks, the third column shows the model outputs and finally, the fourth column shows the only tubule part from benign images.

Analysis: we have achieved 90.45% area under the ROC curve which is shown in Figure 11.20.

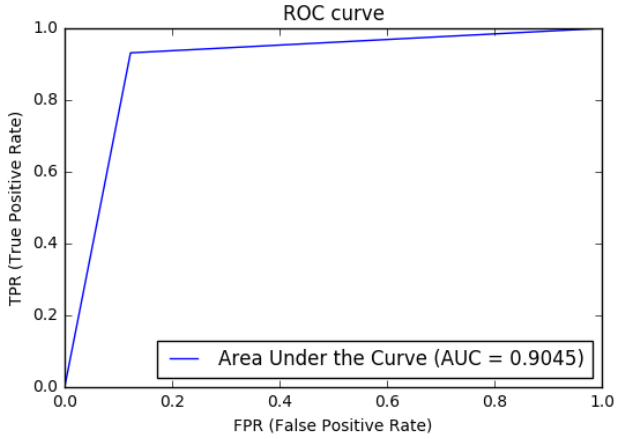


Figure 11.20: ROC curve for Tubule segmentation.

The R2U-Net is applied for tubule segmentation from whole slide images (WSI). The performance of R2U-Net is analysis on a publicly available dataset for tubule segmentation. We have achieved 90.13% and 90.31% for F1-score and accuracy respectively. Qualitative results demonstrate very accurate segmentation compared to ground truth.

11.3.6 Lymphocyte detection

A lymphocyte is a very important part of our immune system and a subtype of white blood cell (WBC). This type of cell is used to determine different types of cancer such as breast, and ovarian. The main challenges and applications of lymphocyte detection: first, In general lymphocytes looks like a blue tint from the absorption of hematoxylin. Second, the appearance and other morphology are very similar in hue to nuclei. The applications of Lymphocyte detection are to identify cancer patient to place on immunotherapy and many more.

Lymphocyte detection dataset: The database image for Lymphocyte detection is shown in Figure 11.21. The top row shows the input images and the bottom row shows the label mask with single pixel annotation for each Lymphocyte which is indicated with green dot pixel. The boundary of the mask is indicated with a black border in Figure 11.21.

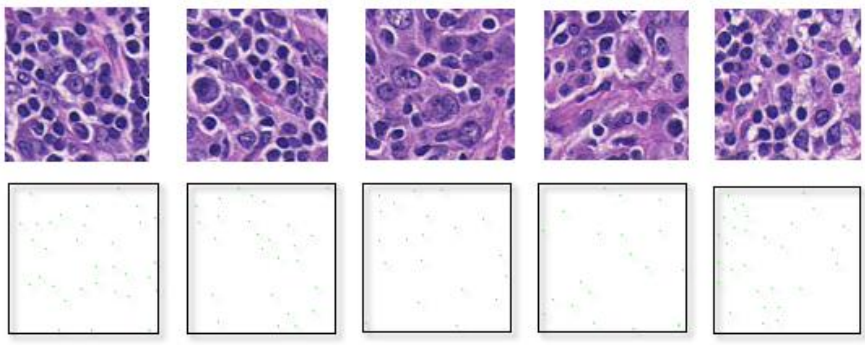


Figure 11.21: The first row shows the inputs samples and the second row shows the label masks with single pixel annotation.

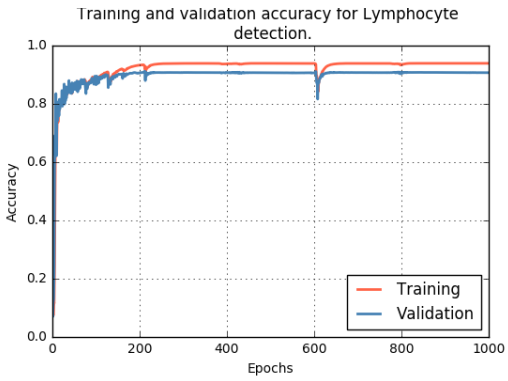


Figure 11.22: Training and validation accuracy for lymphocyte detection.

Experimental results: This dataset is taken from [311]. The total number of samples is 100 with 100 center pixel annotated masks. The size of the image is 100×100. We have used 90% patches are used for training and the remaining 10% are used for testing. We have applied Adam optimizer with learning rate 2×e-4 and cross entropy loss. In this implementation, we have used batch size 32 and number of epochs 1000. Training and validation accuracies are shown in Figure 11.22.

Table 11.8: Testing results for lymphocyte detection.

Methods	F1-score	ROC curve	Accuracy
AlexNet[311]	0.9010	-	-
Proposed approach	0.9092	0.9045	0.9031

Testing accuracy and comparison with others existing approaches are shown in Table 11.8.

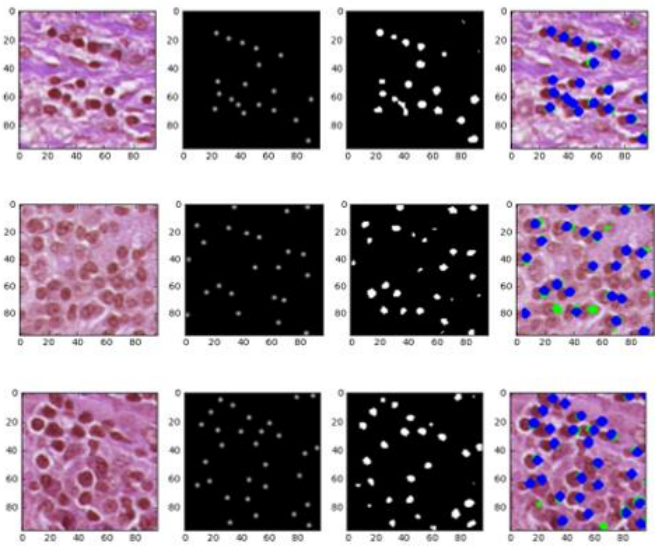


Figure 11.23: Lymphocyte detection outputs: the first column represents inputs samples, the second column shows the ground truth, the third column shows the models outputs, and the fourth column shows the final outputs where the blue dots are for ground truth and green dots for model outputs.

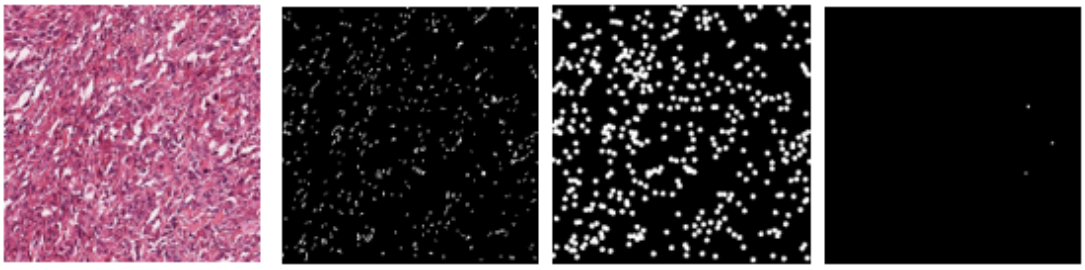
We have achieved around 90.23% testing accuracy and achieved around 0.82% better performance compared to existing deep learning-based approach for lymphocyte detection [311]. The qualitative results for lymphocyte detection with UD-Net are shown in Figure 11.23. The R2U-Net is applied

for tubule segmentation from whole slide images (WSI) for Lymphocyte detection. We have achieved 90.92% accuracy for lymphocyte detection. Qualitative results demonstrate very accurate segmentation compared to ground truth

11.3.7 Mitosis detection

The cell growth rate can be determined with the counting of mitotic events from the pathological images which are an important aspect to determine the aggressiveness of breast cancer diagnosis. Presently, the manual counting process is applied in pathological practice that is very extremely difficult and time-consuming. Therefore, automatic mitosis detection approach has an efficient application in pathological practice.

Database: in this study, we have used a publicly available dataset from [311]. The total number of images 302 which have collect from 12 patients. The actual size of the input sample is 2000x2000 pixels One of the example images from the dataset is shown in Figure 11.24.



Input Actual Mask Dilated Mask Mask with mitosis

Figure 11.24: Sample for mitosis detection: very left image show input and very right sample shows the mask with target mitosis.

As the number of mitosis cell are very less, we have applied different augmentation approaches with {0,45,90,135,180,215,270}. In this study, we have extracted 32x32 patches for the input images and a total number of patches are 728,073. Out of all, the patches, we have randomly selected 100,000 patches where 80,000 patches are used for training and remaining 20,000 patches are used for testing. The example patches are shown in Figure 11.25.

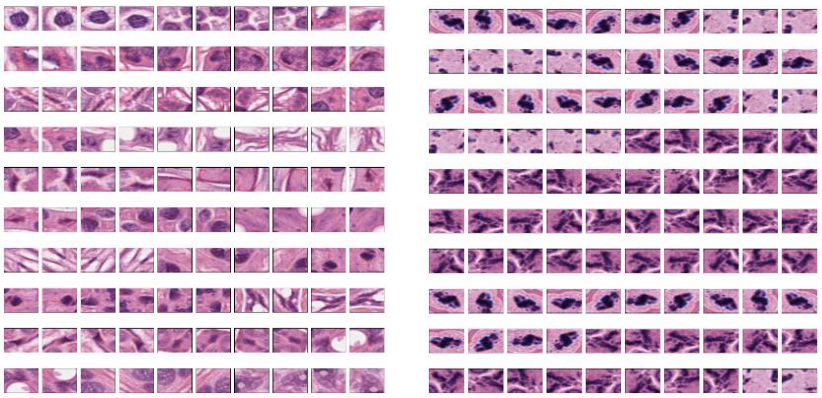


Figure 11.25: Non-mitosis on the left and mitosis cells on the right.

Training approach: the SGD optimization approach is used with the initial learning rate of 0.01. We have used 30 epochs, after every 10 epochs we have decreased the learning rate with a factor of 10. The training and validation outputs are shown in Figure 11.26.

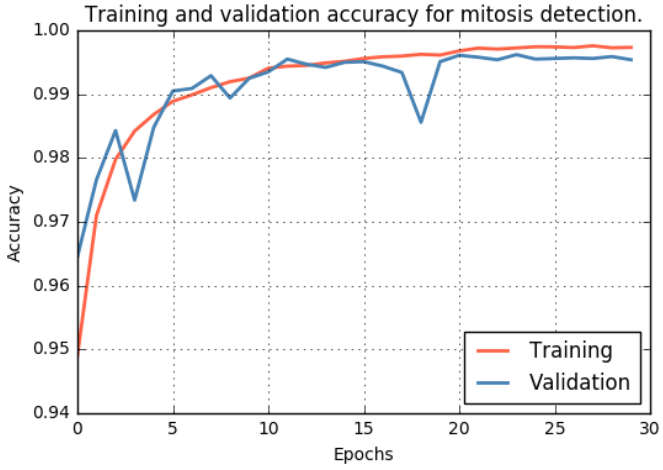


Figure 11.26: Training and validation accuracy for mitosis detection.

In the testing phase, we have achieved 97.32% testing accuracy for mitosis detection. In addition, the experimental results show 99.68% area under ROC curve. It takes 138.94 seconds for 20,000 samples.

Table 11.9: Testing results for mitosis detection.

Methods	F1-score	ROC curve	Accuracy
AlexNet[311]	0.5410	-	-
IRRCNN (image-level)	0.9746	0.9968	0.9732
IRRCNN (patient-level)	0.5929	0.6113	0.595

11.4 Conclusions

We evaluated different advanced DCNN approaches including IRRCNN, DCRCN, and R2U-Net for solving classification, segmentation and detection problems for digital pathology image analysis. For classification task, we have achieved 99.8% and 89.07% testing accuracy which is 3.22% and 4.39% better accuracy for lymphoma and invasive ductal carcinoma (IDC) detection. For segmentation tasks, the experimental results show the 3.31%, 6.5%, 4.13% superior performance for nuclei, epithelium, and tubule segmentation compared to existing Deep Learning (DL) based approaches. For lymphocyte detection, we have achieved 0.82% better testing accuracy and 97.32% and around 60% testing accuracy for mitosis detection for image-level and patient-level respectively, which is significantly higher compared to existing methods. The experimental results demonstrate the robustness and efficiency of our proposed DCNN methods for different use cases of computational pathology.

CHAPTER 12

CONVOLUTIONAL SPARSE CODING ON TRUENORTH

Image features can be learned and subsequently used for reconstruction and classification tasks in the fields of machine learning and computer vision. In this work, we propose image reconstruction using Convolutional Sparse Coding (CSC) on IBM's TrueNorth Neuromorphic computing system. CSC explicitly models local interactions through the convolution operations. Convolutional kernels define a dictionary and Sparse Feature Maps (SFMs) that are generated through a training process. The images are reconstructed with convolutional operations on SFMs and respective kernels. In this paper, we report on experimental results demonstrating promising sparse reconstructions on the IBM Neuromorphic TrueNorth hardware for two different benchmarks: MNIST and CIFAR-10. It is noted that this is the first ever important step towards convolutional sparse coding on neuromorphic hardware.

12.1 Introduction

We are living in a world consumed by instrumentation that continuously draws data from different kinds of sensors. Nowadays big data is a challenging issue, and we need high-performance information processing systems to solve this big data problem. However, a typical HPC environment (such as a supercomputing center, or data processing cluster) requires huge amounts of power. Traditional CPUs with multiple processing cores and GPU based computing systems provide good performance, but they also consume a significant amount of power for performing computations. Therefore, different types of energy efficient and faster computing systems have been developed in the last few years such as field programmable gate arrays (FPGA) [315, 316]

and the IBM neurosynaptic TrueNorth chip [328-334]. These specialized computing systems have some constraints including the number of inputs, memory capacity, and programmability. Mapping Big Data processing algorithms to these specialized computing systems is one of the challenging tasks in using these novel computing systems.

On the other hand, data processing algorithms are always undergoing improvements, and deep learning algorithms have become one of the most prevalent techniques for extracting of complex high-level features for object classification and recognition. The deep learning algorithm, Convolutional Neural Network (CNN) [10], is a layered, or hierarchical data representation and learning approach [317-318]. As the amount of data and data sources are increasing dramatically, deep learning has been playing a key role by providing solutions for Big Data analytics, data representations, and restoration. Therefore, it becomes very important to implement these algorithms for different applications with very power efficient systems such as the IBM TrueNorth processor.

However, in the big data arena, object classification is a very difficult task in the field of machine learning and computer vision. The extraction of meaningful features from input images is an important phase for any object recognition task. Feature extraction is a difficult task because of the high dimensionality of images, different orientations, scales, occlusions, and clutter within the scenes. A sparse representation of images is an important feature extraction technique in machine learning and computer vision, with applications in de-noising, up-sampling, compression, and object detection. Sparse coding is a class of unsupervised methods for learning sets of over-complete bases to represent data efficiently as shown in Figure 12.1 [319, 320]. Sparse coding is a kind of neural code where each item is encoded by strong activations of a relatively small set of neurons. The sparse features of an image can be extracted from the whole image or patches of the input image [321] or using convolutional approaches [322]. It has recently been considered to have important applications in a wide area of computer vision and image processing problems such as low and high-level feature, and low-level image reconstruction [323, 324].

The computational complexity is the main drawback of convolutional models. It is not only challenging to get an appropriate solution within a particular amount of time, but it is also difficult to find local minima. CSC is a non-convex problem which provides little to no guarantees of global convergence. However, in this paper, CSC is implemented for feature extraction for object reconstruction on general purpose computing systems and IBM's specialized TrueNorth Neurosynaptic Cognitive system (TN). This is the very first step towards solving CSC approaches in the TrueNorth neurosynaptic system.

The main objective of convolutional sparse coding can be modeled as reconstructing of image X using convolution with a set of sparse coefficients called SFMs as F and Dictionary as D (see eq. 12.1).

$$E(D, F) = \text{argmin}_{D,F} \|X - D * F\|_2^2 + \beta \|F\|_1 \qquad (12.1)$$

β is a constant, that is set to 0.01 for CPU implementations and 0 in case of TN implementations.

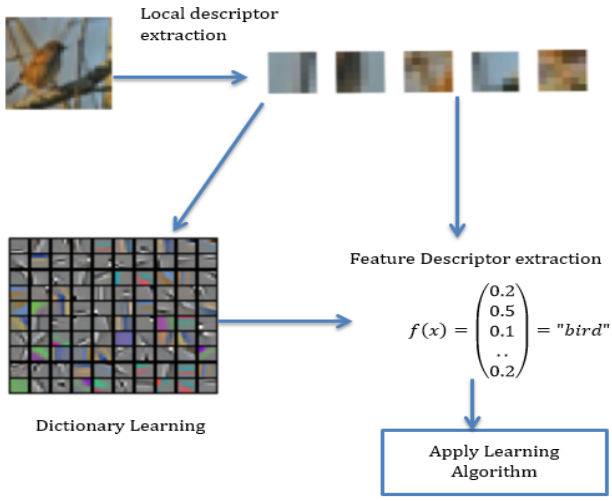


Figure 12.1: Pictorial representation of sparse feature learning.

The contributions of this paper are as follows:

- To the best of our knowledge, this is the first ever implementation of image reconstruction using convolutional sparse coding on the TrueNorth system.
- We experimented on two very popular benchmarks: MNIST and CIFAR-10.

Algorithm 12.1: CSC Algorithm [12-14]

Input: Training Samples $X = \{x_i\}_{i=1}^{N}, K, \alpha$

Output: Initialization filters bank for convolution $D = \{d_k\}_{k=1}^{K}$ with $D \sim N(1,0), Z \leftarrow 0$

Repeat:

for i=1 to N

Normalize individual kernel in D until energy fixing D, and compute sparse feature maps F^i solving according to the following equation:

$$F^i = \text{argmin}_{x_k^i \in Z^i} \left\| x_i - \sum_{k=1}^{K} d_k * f_k^i \right\|_F^2 + \beta \sum_{k=1}^{K} \left\| f_k^i \right\|_1$$

Fixing F and update D

$D \leftarrow D - \mu \nabla_D \zeta(D, F)$

end for

Until: convergence (maximum iterations reached or objective function ≤threshold)

12.2 Convolutional Sparse Coding

In this experiment, we used a direct convolutional approach instead of convolution in the Fourier domain. Let $X = \{x_i\}_{i=1}^{N}$ be a set of N training sample images, each with the dimension of 16×16. $D = \{d_k\}_{k=1}^{K}$ is the dictionary with Kconvolutional filters, where each dictionary atom d_k has a dimension of 3×3. $F = \{F^i\}_{i=1}^{N}$ are the sparse feature maps, where the subset of $F^i = \{f_k^i\}_{k=1}^{K}$ represents the K feature maps for the corresponding input x_i. The main goal of convolutional sparse coding is to decompose the input training samples x_i as a sum of a set of sparse feature maps (SFMs) $f_k^i \in F^i$ convolved with the respective kernels d_k from the dictionary or filters bank D. The objective function can be represented in detail as shown in Eq. (12.2). The pseudo code of the algorithm is given in Algorithm 12.1.

$$\min_{D,F} \mathcal{L} = \sum_{i=1}^{N} \left\{ \left\| x_i - \sum_{k=1}^{K} d_k * f_k^i \right\|^2 + \beta \sum_{k=1}^{K} \left\| f_k^i \right\|_1 \right\} \quad (12.2)$$

12.3 Neuro-Synaptic Cognitive Chips

The traditional von Neumann computing system with a GPU and a multicore processor consumes an abundance of power and area. As systems continue to become larger, the power requirement of these systems has been increasing drastically. To combat this trend, IBM developed and released the TrueNorth Neurosynaptic cognitive architecture as shown in Figure 12.2 in 2015. This is an alternative computing system for implementing machine learning, deep learning, and computer vision algorithms such as CSC with very low power and high energy efficiency [328-334]. The basic characteristics of IBM's cognitive chip are:

- It is based on a non-von Neumann architecture.
- It has 4096 cores per chip, each core consists 256 output neurons, each having 256 axons. A 256×256 crossbar of configurable synapses is in each core.
- Each chip contains 1 million programmable neurons and 256 million synapses.

Recently, IBM has released two systems: the NSe1 single chip system and the NS16e, a 16 chip system. Figure 12.2 shows the multi-chip and single-chip systems with internal details.

The overall TrueNorth architecture is parallel and easily scalable. The internal operation and communication between axons, neurons and other units are performed in spiking form. In every millisecond, each neuron's and each synapse's state are updated. This time interval is called one tick. IBM's neurosynaptic system is very energy-efficient. It is a high throughput neuro-synapse chip that is capable to run between 1200 and 2600 frames/s using only 25 and 275 mW respectively (effectively >6,000 frames / s per W) [331].

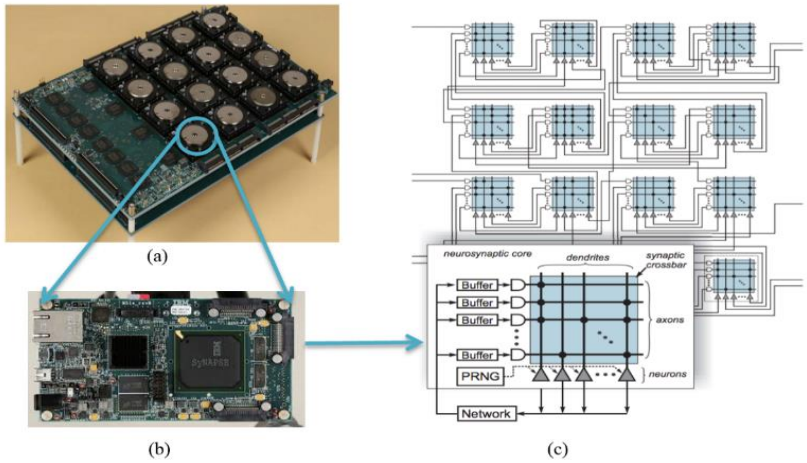


Figure 12.2: IBM's Neurosynaptic Cognitive TrueNorth Chips: (a) TrueNorth multi-chip system, (b) a single chip, and (c) a zoomed-in internal structure of a core.

Each individual axon is assigned 1 of 4 axon types, which is used as an index into a lookup table of s-values, unique to each neuron that provides a 9 signed bits integer synaptic strength to the corresponding synapse.

During each 1ms tick, all neuron membrane potentials are processed and all event routing inside the chip is completed asynchronously. The spikes which are generated by neurons can be sent to any single axon on the chip. Each neuron is represented with over 20 individual programmable features, such as synaptic weight, crossbar weight, threshold, leak, and reset. The structure of TrueNorth is very efficient because of the following reasons: (i) formation of neurons clusters which are created from inputs of similar pools of axons; (ii) spiking events only, which are sparse with respect to time and the communications among the cores performed through a long-distant communication network; (iii) the active power of this architecture is proportional to the firing activity.

12.4 Data Encoding on TrueNorth Chip

The human brain works in the spiking form and represents non-binary information as binary spikes [332]. Inspired by the representation of data in the human brain, there is four type of neural coding

schemes defined to represent different types of information in the TrueNorth system. The neural coding schemes are binary code (B), rate code (R), population code (P), and time-to-spike (T) code. In this study, the rate coding scheme has been used for encoding data in the reconstruction phase. In rate coding, the temporary temporal window has been used so that it utilizes the number of spikes occurring within a specific number of time steps.

Neurons: there are different types of neuron models have been used in the TrueNorth system. The Leaky Integrate-and-Fire (LIF) neurons are used in this study. The following five basic operations describe the LIF neuron model: 1. synaptic integration, 2. leak integration, 3. threshold, 4. spike firing, and 5. reset. In general cases, the LIF neuron model can be described by the following equations

Synaptic integration:

$$V_j(t) = V_j(t-1) + \sum_{i=0}^{N-1} x_i(t)\, s_i \tag{12.3}$$

Leak integration:

$$V_j(t) = V_j(t) - \lambda_j \tag{12.4}$$

Threshold, fire, and reset

$$\text{If} \quad V_j(t) \geq \alpha_j \tag{12.5}$$

Spike

$$V_j(t) = R_j$$

End-if

The parameter $V_j(t)$ in the above equations stands for the sum of membrane potential of the j^{th} neuron in the t^{th} timestep, and $V_j(t-1)$ is the sum membrane potential of the previous timestep. $x_i(t)$, and s_i are the synaptic input as the sum of spike input in the current time step and the signed synaptic weights respectively. λ_j is the leak value that is subtracted in every time-step from membrane potential. Then the membrane potentials are compared with the threshold voltage α_j. If

the membrane potential is greater than or equal to the threshold voltage, the neuron fires a spike and resets the membrane potential.

Crossbar Weights and Synaptic weight: The crossbar weights $w_{i,j} \in \{0,1\}$ of the neurosynaptic core are 0 or 1 (representing active or inactive states) and are represented using a single bit per weight. In addition, each active synapse can have one of the four values as its synaptic weight $s_j^{G_i}$ depending on the axon type. The weight of the core $w_{i,j}$ has been generated from filters which are encoded and has values between -3 to 3. There are four types of axons which are determined with the values of $\{G_i \in 0, 1, 2, 3\}$ and each neuron has four possible sign weights based on the four axon types G_i . In this work, the default value of {8 4 2 1} are used as synaptic weights $s_j^{G_i}$.

Challenges of implementing algorithms on IBM's neurosynaptic system: There are several constraints for implementing an algorithm on the TrueNorth system: first of all, traditional neural networks and convolutional neural network algorithms are implemented using artificial neurons, whereas the IBM TrueNorth architecture uses versatile spiking neurons. Therefore, it is very important as well as challenging to determine a better representation of data in spiking format to use in the TrueNorth system. The spiking neuron model was chosen to balance the dual objectives of capability (from a computational perspective) and cost (from an implementation perspective). The neurons' capability should be sufficient to support useful and interesting cognitive algorithms [333], while the cost should be no more than necessary in terms of power, area, and speed. This cognitive architecture allows the possibility of reducing the computation cost in three ways: first, the use of parallel computations to reduce overhead and increase speed. Second, the power supply to the circuits can be turned off while they are quiescent, which reduces total power consumption. Third, the neurons can be implemented in an event-driven fashion to reduce power consumption as well.

Secondly, the TrueNorth system consists of a parallel architecture inspired by the human brain. If someone wants to implement a traditional serial algorithm in the parallel TrueNorth architecture,

the serial algorithm needs to be appropriate for the parallel TrueNorth system, which will produce better accuracy with high speed and low energy. It is already a challenging problem to design high-performance large-scale deep learning algorithms or applications on TrueNorth neuromorphic hardware. Additionally, there are other challenges such as network connectivity, and weight and bias quantization, which requires architectural and design strategies for physical realization. Most Neuromorphic systems have several constraints on the types of networks they can implement such as limited connectivity between neurons, and a limitation on synaptic weights available.

12.5 Implementation Details

We have implemented CSC for reconstruction approach both on general purpose CPUs and the special purpose TrueNorth computing system. The following section describes implementation details on both platforms.

12.5.1 CPU implementation

The workflow diagram of CPU implementation is shown in Figure 12.3. In the CPU implementation, the input images have been converted to binary format by applying canny edge detection and thresholding approaches on grayscale images of the CIFAR-10 and MNIST datasets respectively. Then the CSC algorithm is applied in the training to generate the dictionary (D) and SFMs. The size of the dictionary is 3×3 and the respective feature maps are is 16×16 which is same as the size of the input images. Nine feature maps are generated for each individual input sample. According to a different study on convolutional sparse coding approaches, it has been shown that the small filter sizes provide better reconstruction from sparse features [319, 320]. Here each individual dictionary element has been named as a dictionary atom and is 3×3 pixels in size. The generated SFMs are represented as nine channel grayscale images $(0\sim255)$. The number of feature maps has been selected empirically – it was observed that nine channels representation gave better reconstruction results. Each dictionary atom is represented with quantized sign values between -3 to 3. The corresponding values of the nine dictionary atoms for CIFAR-10 are shown as an example

in Figure 12.12. Finally, the generated feature maps and respective dictionary atoms are used for reconstructing the original images using convolutional approaches on CPUs.

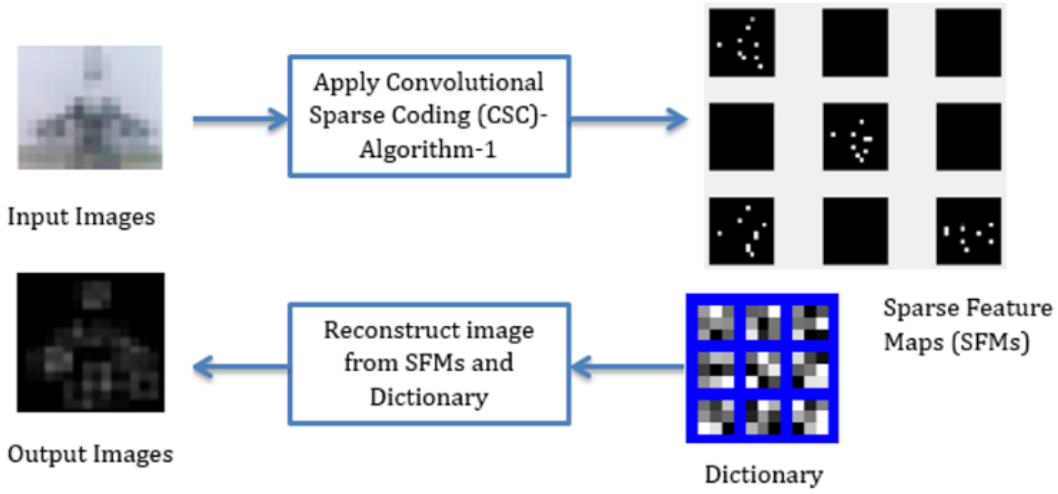


Figure 12.3: Overall implementation diagram of CSC on CPU.

12.5.2 TrueNorth Implementation

In the TN implementation, feature maps are first encoded to grayscale images. Then the gray-scale images are represented as spike files using a rate coding technique where a 20 time-tick window is considered. A maximum of 16 ticks is considered within the window-size. Each image is represented in vector form in column major order. Then each element of the vector is multiplied with the maximum number of ticks (16 in this study). The detailed step-by-step approach for the image to spike representation is shown in Algorithm 12.2. Here ImgVec refers to the vector representation of the input sparse feature matrix, vec_{size} the size of the vector ImgVec, the window size of the spike file is win_{Ticks}, and the maximum number of spikes within the window is represented by max_{Ticks}. The generated spikes are stored into a Spike file. The crossbar weights are generated from the values of atoms of the dictionary using k-bits binary quantization. The quantization of input matrices is generated with the following several steps. Suppose A is an n×m non-quantized input weight matrix and S is an n×k array of synaptic weights. Here $n \leq 64$ represents the number of axons which are repeated 4 times with respect to the type of axons from

0 to 3. m indicates the number of neurons in the crossbar, with $m \leq 256$ and $k \leq 4$ type of axons. Each column of the input matrix of A is quantized to k-bits.

Algorithm 12.2: Image to Spikes using rate coding

Inputs : $\text{ImgVec}, \text{vec}_{\text{size}}: \text{N}, \text{win}_{\text{Ticks}} = 20, \text{max}_{\text{Ticks}} = 16$

Outputs : $\text{Spikes} \in (0,1)$

Initialize : $\text{Spikes} = \text{zeros}(\text{vec}_{\text{size}}, \text{win}_{\text{Ticks}})$

Repeat: $\text{k} = 1: \text{N}$

$$\text{num}_{\text{spikes}} = \lceil (\text{max}_{\text{Ticks}} * \text{ImgVec(k)})/255 \rceil$$

$$\text{rate}_{\text{Tick}} = \text{win}_{\text{Ticks}} / \text{num}_{\text{spikes}}$$

$$\text{the}_{\text{spikes}} = \lfloor \text{max}(1, \lfloor \text{rand} * \text{rate}_{\text{Tick}}/2 \rfloor \rfloor : \text{rate}_{\text{Tick}} : \text{win\_Ticks}$$

$$\text{Spikes}(\text{k}, \text{the}_{\text{spikes}}) = 1$$

End

Consider a column vector Vof size n and row vector S size of k. S multiplied by lookup table B generates n×k binary values of matrix Q where row number i is the binary representation (quantization) of the i^{th} a column of V with respective synaptic weight S of $\{8\ 4\ 2\ 1\}$. These synaptic values are considered as the most significant values for generating the crossbar weights. Moreover, a binary lookup table B is used, where each of the rows contains binary representation values from 0 to 15. For example: binary lookup table for 4 will be a 16×4 matrix in size and the elements are $\{0\ 0\ 0\ 0; 0\ 0\ 0\ 1; \ldots \ldots; 1\ 1\ 1\ 1\}$. Finally, the quantized matrix Q is reshaped to map onto a 256×256 crossbar. Details for the quantization approach are shown in Algorithm 12.3.

Algorithm 12.3: Quantize to k-bit

Inputs : $V_{64x1}, S_{1x4}, B_{16x4} \in (0,1)$

Outputs : $Q_{64x4} \in (0,1)$

Initialize : $n = \text{length}(V) \text{and } k = \text{length}(S)$

$$B = (0\ 0\ 0\ 0; 0\ 0\ 0\ 1; \ldots\ldots\ldots; 1\ 1\ 1\ 1)$$

Body:

$$Q_{values} = S * B'$$

$$QV_{mat} = \text{repeated\_matrix}\ (Q_{values}, n, 1)$$

$$V_{mat} = \text{repeated\_matrix}\ (V, 1, 2\text{^}k)$$

$$[err, i] = \min\left(\text{sqrt}\left(\text{abs}(V_{mat} - QV_{mat})\right)\right)$$

$$Q(1{:}\,n, i) = B(i, :)$$

End

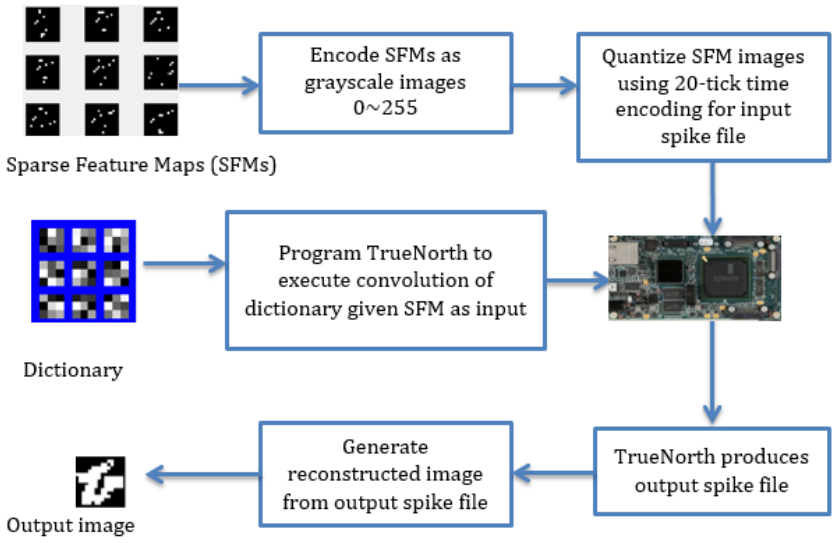


Figure 12.4: Diagram for image reconstruction on IBM's Neurosynaptic system.

In this implementation, the default synaptic weight values are {8 4 2 1}, the threshold is 7, and a leak value of 0 is considered. IBM's cognitive system produces spike files as outputs. The final output images are generated from TrueNorth output spike files. The overall TrueNorth implementation diagram is shown in Figure 12.4. A pictorial representation of the utilization of the cores for individual feature maps on the TrueNorth system is shown in Figure 12.5. During this implementation, the individual SFMs have been divided into four equal and non-overlapping patches with 8×8 in size which is indexed from (1,1) to (2,2) in Figure 12.5.

The convolution operations are performed on the individual patches with single TrueNorth convolutional cores (CC). The convolutional operations are performed for extracting the horizontal and vertical edges. The CC provides four outputs, two for horizontal and two for vertical edges. We have used two adders to sum up the horizontal and vertical pixels respectively.

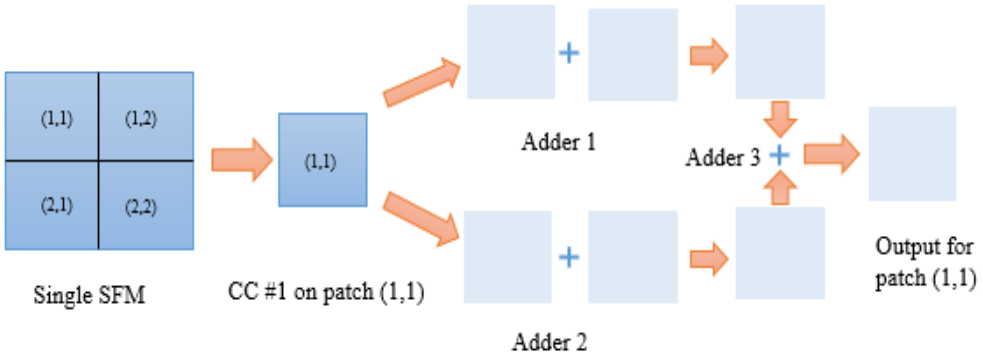


Figure 12.5: Cores utilization on TrueNorth for individual feature map.

According to Figure 12.5, the outputs for horizontal edges are accumulated with adder 1 and adder 2 is used to sum up the outputs of the vertical edges. Adder 3 performs summation operation on the two adder outputs and produce output for the individual path. Finally, the outputs of the patch are converted to matrix form and merged with respect to the patch location to produce the outputs for the individual feature maps. The same operations have been performed for nine feature maps. As we have nine SFMs in total for an input, therefore 9×4=36 cores are used for performing convolution operations.

Table 12.1: Number of cores are used for reconstruction individual image.

Function	**Number of Cores**
Addition	4×3×9 = 108
Convolution for (16×16×9) SFMs	4×9 = 36
Total	144

The total number of cores for addition is 3 which is used for individual (8×8) patches of each input sample. Therefore, for input samples of size 16×16, the total numbers of adders is 4×3×9=108. Hence, there are 144 cognitive cores are used to perform convolution and addition operations to reconstruct one output.

12.5.3 Programming in neurosynaptic system

This entire model is written in MATLAB, using the integrated programming environment called corelet programming for IBM's Neurosynaptic system. The corelet programming environment is entirely new for MATLAB to design and develop complex cognitive algorithms and applications for TrueNorth systems. This framework represents TrueNorth Neurosynaptic cores that encapsulate all details except external inputs and outputs. This is an object-oriented programming framework which is used for creating, composing, and decomposing corelets of the TrueNorth chip [334]. There are two implementation platforms, the first is a simulation platform called the Neurosynaptic Simulator for Corelet System (NSCS) and the other is an actual TrueNorth chip. The runtime environment (platform) can be selected in MATLAB by changing a mode parameter to "TN" or "NSCS". The exact same program can be run on actual a TrueNorth chip or the simulator depending upon flags "TN" or "NSCS" respectively. The outputs of both environments are identical.

12.6 Results and Discussion

In our experiments, we applied CSC our approach for sparse feature generation on CPUs and reconstruction of input images on both CPU and IBM's Neurosynaptic system. The dataset details

are given in the following database subsections. The entire experiments have been conducted on Lawrence Livermore National Laboratory (LLNL) surface cluster and TN clusters. The entire project is implemented using MATLAB (R2015a). In this study, we used Binarized input samples from both datasets. Thresholding and canny edge detection have been applied to MNIST and CIFAR-10 for binarization respectively. The inputs samples of both databases are resized to 16×16. It is noted that we have tested this proposed approach both in "NSCS" and "TN" platforms. However, we have shown all of the experimental results here generated on an actual TrueNorth chip.

Figure 12.6: Example samples from the MNIST database.

12.6.1 Database

Two very popular benchmarks for digit and object classification are MNIST and CIFAR-10. These are used for conducting the experiments.

MNIST: MNIST is one of the benchmark image classification databases [160]. This dataset consists of 60000 training samples and 10000 test samples that are 28×28 gray-scale image representing digits ranging from 0 to 9. We did not apply any data-augmentation during this experiment except for resizing input samples of datasets. The sample images are shown in Figure 12.6.

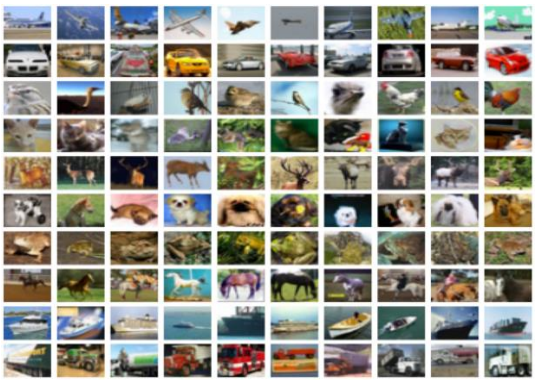
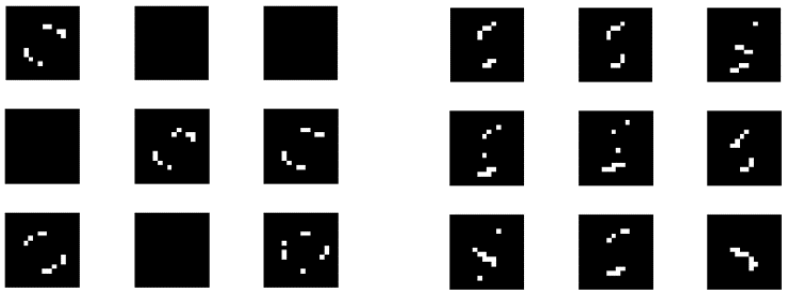


Figure 12.7: Example samples from the CIFAR-10 database.

CIFAR-10: Another image classification benchmark is the CIFAR-10 database. This database consists of 50,000 training samples and a test set of 10,000 32×32 color images that represent airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks [160]. We have used 20,000 training samples in this experiment. Some of the examples samples of the respective classes are given in Figure 12.7.

12.6.2 Experimental results

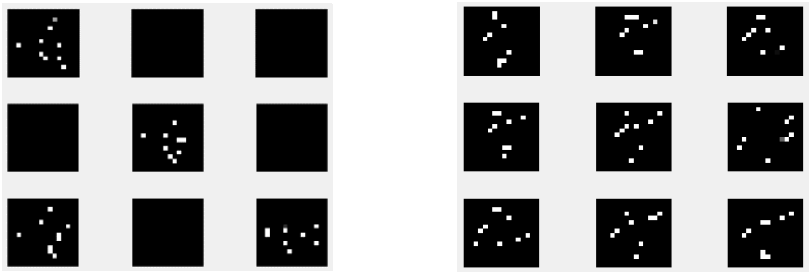
The generated SFMs and Dictionary for both datasets are generated through training. Overall training processing has been done using the algorithms stated in Section 12.2.



(a) (b)

Figure 12.8: Example nine SFMs for inputs from MNIST: (a) digit zero (b) digit five.

The SFMs which are generated in this experiment for some of the input samples of MNIST and CIFAR-10 are shown in Figures 12.8 and 12.9 respectively.



(a) (b)

Figure 12.9: Example nine SFMs for inputs from CIFAR-10 dataset: (a) Airplane (b) head of a deer.

It can be observed that the generated feature maps are very sparse. As an example, Figure 12.8(a) and (b) show the sparse representation of digits zero (0), and five (5) for the MNIST database respectively. Figures 12.9 (a) and (b) show the sparse representation for airplanes and head of the deer from the CIFAR-10 dataset respectively. We generated the dictionary for both MNIST and CIFAR-10 datasets. The size of the dictionary atom is 3×3 and 9 atoms for the respective feature maps. The dictionary images for MNIST and CIFAR-10 are shown in Figures 12.10 and 12.11 respectively.

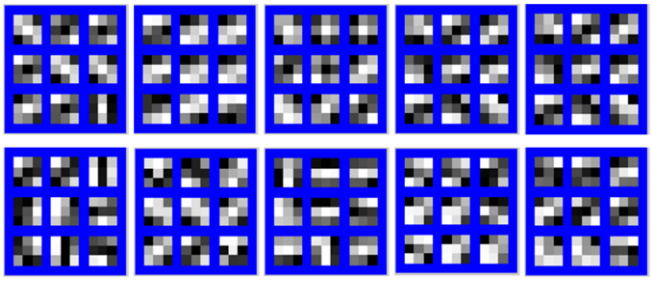


Figure 12.10: Learning Dictionary for MNIST database.

In the TrueNorth implementation, the values of atoms are encoded between values of -3 to 3. The example encoded values of atoms of the generated dictionary for CIFAR-10 are shown in Figure 12:12.

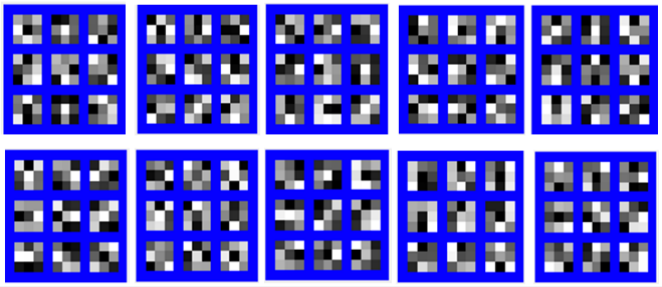


Figure 12.11: Learning Dictionary for CIFAR-10 database.

Eventually, the images are reconstructed from SFMs and the respective atoms from the dictionary in the reconstruction phase on traditional CPUs and specialized TrueNorth chip. The final experimental results are discussed in the following sections in detail.

-2 -2 -2 -1 -1 -3 -1 0 -2	2 2 1 1 0 3 1 2 2	2 0 2 2 1 2 2 3 2
(a)	(b)	(c)
2 2 1 1 1 3 2 1 2	-2 -2 -1 -3 -1 0 -1 -2 -2	1 2 3 2 1 1 2 1 1
(d)	(e)	(f)
-1 -1 -2 -1 -1 -3 -2 -2 -1	-1 -2 -2 -3 -1 0 -2 -2 -2	-2 -1 -2 -2 -2 -1 -1 -2 -3
(g)	(h)	(i)

Figure 12.12: Quantized values of 9 dictionary atoms from (a) to (i) for CIFAR-10 dataset.

12.6.2.1 Results on MNIST dataset

The reconstructed results for MNIST on CPU and TN system are shown in Figure 12.13. The first row shows the input sample of zero to nine (0-9) from the MNIST database. The second row shows the binary representations of the respective input samples after applying the thresholding approach from the MNIST database.

Figure 12.13: Experimental outputs of MNIST dataset.

The third row of the figure shows the reconstructed digits on a Matlab on a CPU. The fourth row shows the outputs on IBM's Neurosynaptic TrueNorth system. Although the input samples are very challenging in terms of rotation, scale, and variation of writing style in the MNIST dataset, the TrueNorth outputs show promising results for reconstruction.

12.16.2.2 Results on the CIFAR-10 dataset

CIFAR-10 is another very challenging dataset. The input samples for individual classes are shown in the very first row of Figure 12.14.

Figure 12.14: Experimental outputs of the CIFAR-10 dataset.

The binarized input images are shown in the second row of the same figure. The third row shows the Matlab based CPU reconstruction using CSC approaches. Finally, the fourth row shows the results on IBM's TrueNorth system. Object representation is more complex compared to digits due to them having more different edge orientations. However, the proposed Neuromorphic sparse

reconstruction approach shows good reconstruction results for different objects in the CIFAR-10 dataset. The objects are clearly recognizable with respect to input samples (in the very first row) for the respective classes in the last row of Figure 12.14.

12.7 Power Consumption

In this section, the power consumption of the TrueNorth system is compared with the traditional computing system. Traditional computing systems, such as CPUs and GPUs easily consume 100W or more power, whereas an entire TrueNorth system consumes only up to 100mW to operate the 4096 cores on it. Here we only used 144 cores for image reconstruction as shown in Table 12.1 in section V. It is noted that about 50% of the power is passive power in TrueNorth system. This is means that the overall power requirement for 144 cores is about $0.5 * 100\text{mW} + (144/4096) * 50\text{ mW} = 51.75\text{ mW}$. Even if there is an overhead to use of 144 cores, there is still an order of magnitude difference in terms of power consumption between the Neuromorphic computing system and the traditional computing system. By considering the extremely low power budgets, such low power systems can be used for a low-power application such as mobile device, sensor application, and robotics. Additionally, it could be used for energy efficient supercomputing applications.

12.8 Conclusion and Future Works

This work represents a very important first step towards convolutional networks based sparse coding techniques on the extremely low power TrueNorth Neurosynaptic computing system. The key contribution of this work is to implement the convolutional sparse coding (CSC) approach onto the Neuromorphic architecture and evaluate the performance of reconstruction on two popular benchmarks such as MNIST and CIFAR-10. We achieved promising reconstruction on the TrueNorth implementation for both datasets. As future work, we would like to implement and investigate recognition accuracy with the generated sparse features.

CHAPTER 13

DEEP VERSUS WIDE DCNN ON TRUENORTH

In the last decade, special purpose computing systems, such as Neuromorphic computing, have become very popular in the field of computer vision and machine learning for classification tasks. In 2015, IBM's released the TrueNorth Neuromorphic system, kick-starting a new era of Neuromorphic computing. Alternatively, Deep Learning approaches such as Deep Convolutional Neural Networks (DCNN) show almost human-level accuracies for detection and classification tasks. IBM's 2016 release of a deep learning framework for DCNNs, called Energy Efficient Deep Neuromorphic Networks (Eedn). Eedn shows promise for delivering high accuracies across a number of different benchmarks, while consuming very low power, using IBM's TrueNorth chip. However, there are many things that remained undiscovered using the Eedn framework for classification tasks on a Neuromorphic system. In this paper, we have empirically evaluated the performance of different DCNN architectures implemented within the Eedn framework. The goal of this work was to discover the most efficient way to implement DCNN models for object classification tasks using the TrueNorth system. We performed our experiments using benchmark data sets such as MNIST, COIL-20, and COIL-100. The experimental results show very promising classification accuracies with very low power consumption on IBM's NS1e Neurosynaptic system. The results show that for datasets with large numbers of classes, wider networks perform better when compared to deep networks comprised of nearly the same core complexity on IBM's TrueNorth system.

13.1 Introduction

We are living in a world consumed by instrumentation that continuously draws data from many kinds of sensors. Nowadays big data is a challenging issue, and we need high-performance information processing systems to solve this big data problem. However, a typical high-performance computing (HPC) environment (such as a supercomputing center, or data processing cluster) requires huge amounts of power. Traditional CPUs with multiple processing cores, and larger implementation of Deep Learning (DL) model on Graphics Processing Units (GPU) based computing systems provide state-of-the-art performance, but it consumes a significant amount of power for performing computations. Therefore, different energy efficient and faster computing systems have been developed in the last few years such as field programmable gate arrays (FPGA) [315, 316] and the IBM's Neurosynaptic TrueNorth chip [329-334]. These specialized computing systems have some constraints as well. The constraints are on the number of inputs, memory capacity, and programmability. Mapping big data processing algorithms to these specialized computing systems is one of the most challenging tasks. Among the many available architectures, IBM's TrueNorth (TN) system is one of the first Neuromorphic Chips, which is very efficient in term of power consumption, and with very high throughput [329, 331]. In addition, the MATLAB based corelet programming language was developed, providing a highly scalable objected oriented programming structure [334,335]. Currently, there are many different applications implemented on IBM's TrueNorth system which have shown promising performance, including object recognition [332], cybersecurity [336], optimization approaches on the TrueNorth system [337], convolutional sparse coding [338], and many more [339].

Data processing algorithms are always undergoing improvements, and deep learning algorithms have become one of the most prevalent techniques for extracting complex high-level features for object classification and recognition. Deep Learning algorithms, Convolutional Neural Networks (CNN) in particular, use a layered, or hierarchical data representation and learning approach [7, 317]. Furthermore, researchers using modified CNNs have reported improved results for object

recognition on different benchmarks including MNIST, CIFAR 10 or 100, Caltech 101 or 256, ImageNet, and many more [7, 11]; as well as improved object detection [340] tasks. Accordingly, DCNN approaches have become very popular and widely used in machine learning and computer vision tasks; the main drawback, however, is the increased computational complexity of convolutional network models. In most of the implementations, GPUs are used for training the big networks, which, in most of the cases are utilizing wider and deeper networks for training with higher precision (more than or equal 32 bits) on different benchmarks [10]. As the network size increases, the computational parameters also increase dramatically. It follows that the increased computational costs resulting in significantly greater power consumption due to the use of power-hungry GPUs [341].

As the amount of data and data sources are increasing dramatically, deep learning has been playing a key role by providing solutions for Big Data analytics, data representations, and restoration. In 2015, IBM released the TrueNorth chip, a very low power Neuromorphic processor made up of a massively parallel architecture. TrueNorth is ideally suited to address the big data problem with a significantly lower power profile than conventional systems. Following the trend of deep learning development, IBM released the Energy Efficient Deep Neuromorphic Network (Eedn) framework for implementing CNN approaches on the TrueNorth system [335]. Accordingly, it becomes very important to implement and evaluate different Deep Learning models for different applications on the very power efficient IBM TrueNorth system. In this implementation, we have implemented different DCNN architectures utilizing the Eedn framework. The contributions of this paper are summarized as follows:

- Implemented different energy efficient DCNN models with the Eedn framework.
- Experimented on three popular benchmarks to evaluate different architectures of DCNN including MNIST, COIL-20, and COIL-100.
- Experimented with different deeper and wider deep convolutional networks and discovered the impact of deepness and wideness of networks on the TrueNorth system.

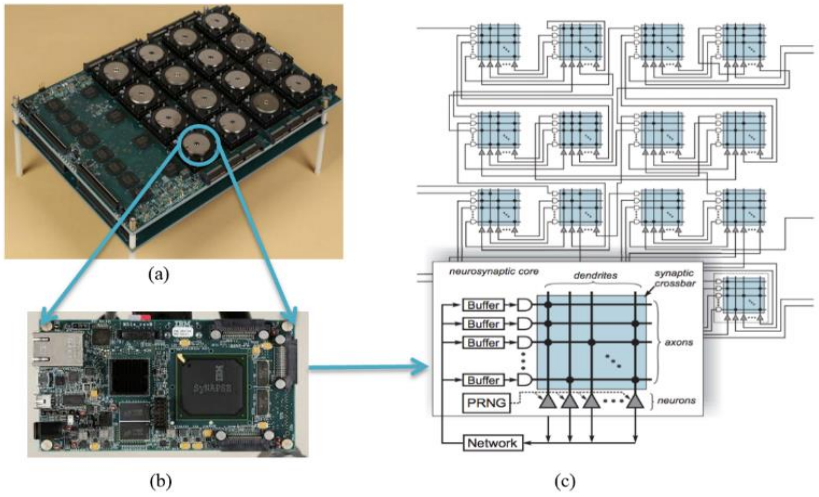


Figure 13.1: IBM's Neurosynaptic Cognitive TrueNorth Chips: (a) TrueNorth multi-chip system, (b) a single chip, and (c) a zoomed-in internal structure of a core.

13.2 Related Works

In the deep learning research community, most of the researcher uses the basic structure of Convolutional Neural Networks (CNNs) with alternative convolution and max-pooling layer followed by a small number of fully connected layers [342]. The piecewise-linear or non-linear activation functions are used within each of these layers. The dropout technique has been used for regularizing the overall network [83]. In addition, drop connection is also used for regularizing the network. However, in general, to evaluate those DCNN architectures, the general-purpose computing system such as CPU, GPGPU, and the multicore system is used [341]. The optimization of DCNN models is proposed with respect to structural and computational optimization. As structural optimization is of concern, several research studies have been conducted in the community to improve the overall accuracy of the DCNN model with lesser numbers of computational parameters, which significantly decreases computational time and power consumption. Some papers have been published on structural optimization of DCNN techniques, which is called SqueezeNet [148]. In most of the cases, these power efficient and faster models are proposed based on low precision implementations of a DCNN [343]. In 2015, Y. Bengio et al., show that a deep network can achieve very high precision training networks using only binary

weight values [0 1] [344]. Recently, the ternary weight-based CNN is proposed [345]. The IBM Eedn framework is implemented based on the concept of ternary connected networks [332]. Another controversy is wide versus deep convolutional networks. There are many papers that have been published with full precision implementation on this topic and there is still some research ongoing.

A recently published paper entitled "Do deep learning need to be deep", clearly stated the impact of network structure on overall recognition accuracy. It concluded that the deeper network (which incorporates more layers for better feature embedding) provides better accuracy compared to the wider network (increase number of neurons with a larger number of feature maps in a layer) [24]. In addition, some research was conducted to evaluate the impact of network structure (deeper and wider network) on accuracy with the same number of parameters. This implementation also summarizes that the deeper network performs better compared to wider networks [346]. Another study shows that shallow networks are unable to reach the same levels of accuracy against deep networks with the same number of network parameters. Eventually, they demonstrate that deeper networks provide better performance compared to shallow networks [347].

The question now becomes, is this true in case of the DCNN with the ternary connect method on TrueNorth system? This answer has not yet been determined for IBM's TrueNorth system using a deep learning training methodology with binary weights (0,1), which is well suited to map deep learning onto the TrueNorth system. In 2016, IBM released the Eedn deep learning framework for implementing deep learning on the TrueNorth system, which opened a new opportunity to implement energy efficient deep learning approaches on Neuromorphic hardware [332]. This deep learning framework is very power efficient and provides promising accuracies for image classification tasks. Unlike the implementation of deep learning on CPU, GPGPU, and multicore systems, it is necessary to evaluate the impact of network structures on recognition accuracy for IBM's Neuromorphic TrueNorth system. We have empirically evaluated the performance of different DCNN architectures, tested on different data sets which will help determine the efficient

design of DCCN models for use on the TrueNorth system; this can lead to the development of additional energy efficient models with better recognition performance.

13.3 Neuro-synaptic Cognitive System

The traditional von Neumann computing system with a GPU and a multicore processor consumes an abundance of power and area. As systems continue to become larger, the power requirement of these systems has been increasing drastically. To combat this trend, IBM developed and released the TrueNorth Neurosynaptic cognitive architecture as shown in Figure 13.1 in 2015. This is an alternative computing system for implementing machine learning, deep learning, and computer vision algorithms with very low power and high energy efficiency [329,331]. The basic characteristics of IBM's cognitive chip are: first, it is based on a non-von Neumann architecture. Second, it has 4096 cores per chip, each core consists 256 output neurons, each having 256 axons. A 256×256 crossbar of configurable synapses is in each core. Third, each chip contains 1 million programmable neurons and 256 million synapses.

Figure 13.1 shows the single chip and multi-chip systems with internal architectural details. The overall TrueNorth architecture is parallel and easily scalable. The internal operation and communication between axons, neurons and other units are performed in spiking form. It is a high throughput Neurosynaptic chip that is capable to run between 1200 and 2600 frames per second using only 25 and 275 mW respectively (effectively greater than 6,000 frames per second per watt) [26]. Each individual axon is assigned 1 of 4 axon types that provides a nine signed bits integer synaptic strength to the corresponding synapse. All event routing inside the chip is completed asynchronously. Each neuron is represented with over 20 individual programmable features, such as synaptic weight, crossbar weight, threshold, leak, and reset. The structure of TrueNorth is very efficient because of the following reasons: first, the formation of neurons clusters which are created from inputs of similar pools of axons. Second, spiking events only, which are sparse with respect to time and the communications among the cores performed through a long-distant communication network. Third, the active power of this architecture is proportional to the firing activity.

13.4 Data Encoding on TrueNorth Chip

The human brain works in the spiking form and represents non-binary information as binary spikes [329] There is four type of neural coding schemes defined to represent different types of information in the TrueNorth system. The neural coding schemes are binary code (B), rate code (R), population code (P), and time-to-spike (T) code. In general, for data encoding rate coding scheme is used.

13.4.1 Neurons

Different types of neuron models can be modeled in the TrueNorth system. For purposes of this study, the Leaky Integrate-and-Fire (LIF) neuron model will be examined. The following five basic operations describe the LIF neuron model: synaptic integration, leak integration, threshold, spike firing, and reset. In general cases, the LIF neuron model can be described by the following equations:

Synaptic integration:

$$V_j(t) = V_j(t-1) + \sum_{i=0}^{N-1} x_i(t)\, s_i \tag{13.1}$$

Leak integration:

$$V_j(t) = V_j(t) - \lambda_j \tag{13.2}$$

Threshold, fire, and reset

$$\text{If} \quad V_j(t) \geq \alpha_j \tag{13.3}$$

$$V_j(t) = 1$$

Else

$$V_j(t) = 0$$

End-if

The parameter $V_j(t)$ in the above equations stands for the sum of membrane potential of the j^{th} neuron in the t^{th} timestep, and $V_j(t-1)$ is the sum membrane potential of the previous timestep. $x_i(t)$, and s_i are the synaptic input as the sum of spike input in the current time step and the signed

synaptic weights respectively. λ_j is the leak value that is subtracted in every time-step from membrane potential. Then the membrane potentials are compared with the threshold voltage α_j. If the membrane potential is greater than or equal to the threshold voltage, the neuron fires a spike and resets the membrane potential.

13.4.2 Crossbar weights and synaptic weight

The crossbar weights $w_{i,j} \in \{0,1\}$ of the neurosynaptic core are 0 or 1 (representing active or inactive states) and are represented using a single bit per weight. Moreover, each active synapse can have one of the four values as its synaptic weight $s_j^{G_i}$ depending on the axon type. There are four types of axons which are determined with the values of $\{0\ 1\ 2\ 3\}$. In this work, the default value of S_j $\{8\ 4\ 2\ 1\}$ are used as synaptic weights [329].

13.5 Implementation with Energy Efficient Deep Networks (EEDN)

IBM's Eedn is a complete deep learning framework for the TrueNorth Neuromorphic system that is used for training, mapping the network onto the Neuromorphic chip, and testing of a DCNN model. This framework consists of different types of layers to construct deep convolutional architectures. The layers are: an input layer, pre-processing layer, convolution layer, network in the network layer, pooling layer, and drop out layer. During the training, the convolution layer performs basic convolutional operation respect to filter with input features of this layer. For example: if the input dimension is 28x28 and the filter size is 3x3 with 12 feature maps then the output size of this layer will be 26x26x12. The pooling or sub-sampling operation is performed using convolution with stride size 2. The dropout operations are applied with fractional value and we have used 0.5 in this implementation. In the training phase, the deep neural network is trained using some steps on the GPU which is given in Algorithm 13.1 [332].

Algorithm 13.1: Training steps for DCNN on TrueNorth

Step 1. Training performs iteratively

Step 2. The network's response is computed through the forward pass of the network

Step 3. The network errors are calculated with network outputs and desired outputs

Step 4. The gradient errors are computed at each synapse in the backward pass

Step 5. Update weight along with gradient respect to the errors

After successfully completing the training process, the network is mapped onto the IBM's TrueNorth system. In this case, the grouping approach is used in the convolution layers. Let's considered the following components such as mask size or kernel size (K), number of features map (F), and the number of groups (G). However, during the implementation of the network onto the TrueNorth system, the following conditions need to be satisfied: first, the number of inputs must be less than or equal to 128.

$$\mathrm{K} \times \mathrm{K} \times \frac{\mathrm{F}}{\mathrm{G}} \leq 128 \tag{13.4}$$

Second, the number of group of i^{th} the layer must divisible with the number of feature maps of $\mathrm{i}-1^{\mathrm{th}}$ layer.

$$\mathrm{G}_{\mathrm{i}}^{\mathrm{n}} \ \% \ \mathrm{F}_{\mathrm{i}-1}^{\mathrm{n}} = 0 \tag{13.5}$$

In Eq. 7, $\mathrm{G}_{\mathrm{i}}^{\mathrm{n}}$ is the number of group of i^{th} layer and $\mathrm{F}_{\mathrm{i}-1}^{\mathrm{n}}$ is a number of feature maps of $\mathrm{i}-1^{\mathrm{th}}$ layer. Third, the total number of cores (TNC) of the architecture must be less or equal to 4096 cores.

$$\mathrm{TNC} \leq 4096 \tag{13.6}$$

13.6 DCNN on TrueNorth

In this paper, we have empirically evaluated the performances of different architecture of DCNN on different benchmarks. The DCNN architecture consists of different components including:

preprocessing layer (P), Convolution layer (C), sub-sampling layer (S), and Network in Network (NiN) layers. We have tested different networks; however, the deeper and wide network structure are shown in Figure 13.2 and Figure 13.3 respectively. Here one example network architecture is provided for the deep and wide network which contains 4064 and 4096 cores respectively. Due to the different hyperparameter of Eedn including a number of feature maps and a number of groups, it is hard to implement a network model with the same number of cores. For example, in the case of the first implementation of the deep model, the number of the splitter is 384 which is used in the second convolution layer in C2. On the other hand, the total number of the splitter is 384+1024 = 1408 which is used in C2 and C4 respectively.

	Layer Size		Patch		TN Cores	Patch-in-Image
Lyr	Row × Col × Ftr	(Grp)	Str	Row Col Ffr	Base+Splt	Str Row × Col
I	32 × 32 × 3					
P1	32 × 32 × 12	(1)	1	[3 × 3 × 3]	0 +0	1 [3 × 3]
C2	16 × 16 × 252	(2)	2	[4 × 4 × 6]	512 +384	2 [6 × 6]
C3	16 × 16 × 256	(2)	1	[1 × 1 × 126]	512 +0	2 [6 × 6]
C4	8 × 8 × 256	(8)	2	[2 × 2 × 32]	512 +0	4 [8 × 8]
C5	8 × 8 × 512	(32)	1	[3 × 3 × 8]	512 +0	4 [16 × 16]
C6	8 × 8 × 512	(4)	1	[1 × 1 × 128]	256 +0	4 [16 × 16]
C7	8 × 8 × 512	(4)	1	[1 × 1 × 128]	256 +0	4 [16 × 16]
C8	8 × 8 × 512	(4)	1	[1 × 1 × 128]	256 +0	4 [16 × 16]
C9	4 × 4 × 512	(16)	2	[2 × 2 × 32]	256 +0	8 [20 × 20]
C10	4 × 4 × 1024	(64)	1	[3 × 3 × 8]	256 +0	8 [36 × 36]
C11	4 × 4 × 1024	(8)	1	[1 × 1 × 128]	128 +0	8 [36 × 36]
C12	2 × 2 × 1024	(32)	2	[2 × 2 × 32]	128 +0	16 [44 × 44]
C13	2 × 2 × 1024	(8)	1	[1 × 1 × 128]	32 +0	16 [44 × 44]
C14	2 × 2 × 1024	(8)	1	[1 × 1 × 128]	32 +0	16 [44 × 44]
C15	2 × 2 × 2040	(8)	1	[1 × 1 × 128]	32 +0	16 [44 × 44]

Figure 13.2: Deeper network architecture with 15 layers.

	Layer Size		Patch		TN Cores	Patch-in-Image
Lyr	Row × Col × Ftr	(Grp)	Str	Row Col Ffr	Base+Splt	Str Row × Col
I	32 × 32 × 3					
P1	32 × 32 × 12	(1)	1	[3 × 3 × 3]	0 +0	1 [3 × 3]
C2	16 × 16 × 252	(2)	2	[4 × 4 × 6]	512 +384	2 [6 × 6]
C3	16 × 16 × 512	(2)	1	[1 × 1 × 126]	512 +0	2 [6 × 6]
C4	8 × 8 × 512	(32)	2	[2 × 2 × 16]	1024+1024	4 [8 × 8]
C5	4 × 4 × 1024	(32)	2	[2 × 2 × 16]	256 +0	8 [12 × 12]
C6	4 × 4 × 1024	(8)	1	[1 × 1 × 128]	128 +0	8 [12 × 12]
C7	2 × 2 × 2048	(32)	2	[2 × 2 × 32]	128 +0	16 [20 × 20]
C8	2 × 2 × 2048	(32)	1	[1 × 1 × 64]	64 +0	16 [20 × 20]
C9	2 × 2 × 4096	(64)	1	[1 × 1 × 32]	64 +0	16 [20 × 20]

Figure 13.3**:** Wider network model with 9 layers.

13.7 Experimental Results and Discussion

The entire experiments have been conducted on a desktop computer with an Intel ® Core ™ 2 Duo CPU E86 @ 3.33 GHz processor and 12GB of RAM to evaluate the processing time in MATLAB (R2015a). The datasets details are given in the following database section. The model is implemented in MATLAB, using the integrated programming environment called corelet programming for IBM's Neurosynaptic system. There are two platforms to evaluate, first simulation platform called Neurosynaptic Simulator for Corelet System (NSCS) and another is in actual hardware. We have tested this experiment on both platforms. The running environment (platform) can be selected by changing the model parameters of "TN" and "NSCS". The exact same program can be run on actual TrueNorth chip or simulator depending upon flag "TN" or "NSCS" respectively. It is noted that the outputs of both environments are almost identical. However, in this implementation, we have evaluated the performance on a single chip TrueNorth system.

13.7.1 Database

Three popular benchmarks for digit and object recognition such as MNIST, COIL-20, and COIL-100 datasets are used in this implementation.

Figure 13.4: Example samples from the MNIST database.

MNIST: MNIST is one of the benchmark image classification database [348]. This dataset consists of 60000 training samples and 10000 test samples 28x28 gray-scale image representing digits

ranging from 0 to 9. We did not apply any data-augmentation except resizing input sample of dataset during this experiment. The samples images are given in below Figure 13.4.

Figure 13.5: Example images from the COIL-20 dataset.

COIL-20 dataset: there is two version of the database available for Columbia Object Image Library (COIL)-20, the first version of this dataset with background and another version is without background. In this implementation, we have used the training and testing samples with the background. This database contains 1440 observations (20 objects with 72 poses each) in total, where 1100 samples are used for training the network and remaining 300 samples are used for testing [349]. The example images are shown in the following Figure 13.5.

COIL-100: COIL-100 dataset is extended dataset of COIL-20. This dataset contains color images for 100 classes of object. This dataset contains 7000 Color images where 5000 samples are used for training and remaining 2000 samples are used for testing in this implementation. The turntable was rotated through 360 degrees to vary object pose with respect to a fixed color camera. Images of the objects were taken at pose intervals of 5 degrees. This corresponds to 72 poses per object [350]. Due to the input size contained TrueNorth system, we have resized the input sample to 32x32 pixels. The examples images of the dataset are shown in Figure 13.6.

Figure 13.6: Example image of COIL-100 dataset.

13.7.2 Results

We have evaluated the performance of different architectures which consist of different numbers of layers and cores on TrueNorth system. The performance of the network varies with respect to the number of cores. We have tested on MNIST, COIL-20, and COIL-100 datasets. We have also investigated the variation of recognition accuracy respect structure of network with the same number of cores on IBM's TrueNorth.

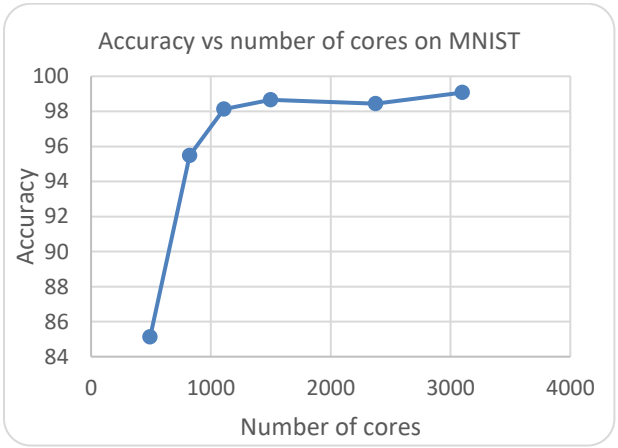


Figure 13.7: Testing accuracy versus a number of cores on MNIST dataset.

MNIST:

To evaluate the performance on MNIST dataset, we have taken the default implementation network for MNIST dataset in the Eedn framework and we have a varied number of features maps and groups and testing with different architectures. Figure 13.7 shows the accuracy with respect to the number of cores with different architecture. The figure clearly shows that the performance increase with respect to the number of cores used with bigger networks. Figure 13.8 shows that testing accuracy for MNIST dataset with a network consisted with more cores with a bigger structure and we have achieved around 99.07 percent accuracy with the deeper network. In addition, we have also tested with the wide version of the same network with an almost the same number of cores. However, from Figure 13.8, it can be clearly concluded that the deeper network provides better testing accuracy compare against the wider network.

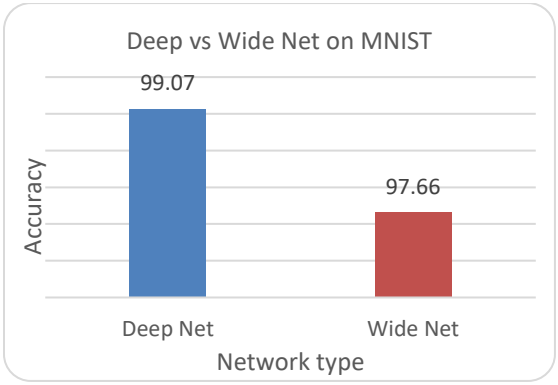


Figure 13.8: Comparison of testing accuracy of deep versus wide network on MNIST dataset.
COIL-20.

The following figures show the training loss and accuracy of this implementation for 3000 iterations. From Figure 9, it is clearly shown that DCNN model on TrueNorth system provides promising recognition accuracy with only 3000 iterations on the COIL-20 dataset. After round 1000 iteration, we have achieved almost 100% training accuracy on this dataset.

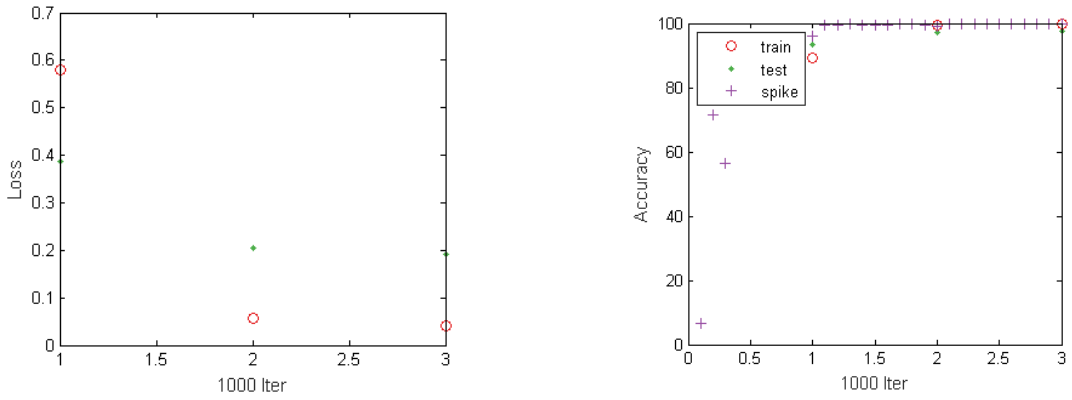


Figure 13.9: Training loss for COIL-20 only for 3000 iterations.

We have also investigated different models on TrueNorth system with different models with a different number of cores from 480 to 4094 cores shown in Figure 10. According to Figure 13.10, we have observed the highest accuracy with only around 1400 cores. The very close accuracy is observed with 4064 cores on TrueNorth for COIL-20 dataset. It is noted that for all different networks, we have conducted an experiment with only 3000 iterations. In both experiments of COIL-20 and COIL-100, we have used batch size 50 and learning rate 0.1 and 0.01.

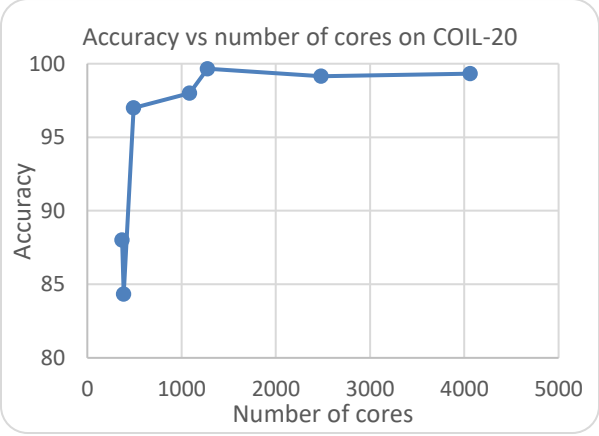


Figure 13.10: Network accuracy respect to the number of cores on the COIL-20 dataset.

However, we have implemented two version of the network with 4064 and 4096 cores respectively; one is deeper (increase number of layers within the network) and another one is wider (increase number of neurons in the network). Figure 13.11 shows the recognition accuracy for COIL-20,

where deep networks use 4064 cores and wide version of network utilizes 4096 cores on single chip implementation which are shown in Figure 13.2 and Figure 13.3 respectively. Figure13.11 shows the testing recognition accuracy for COIL-20, the wider version networks provide around 99.36% recognition accuracy whereas the deeper of the network shows 99.23% accuracy. According to Figure 13.11, it is clearly concluded that the wider network provides better recognition accuracy compare to the deeper network with an almost the same number of cores.

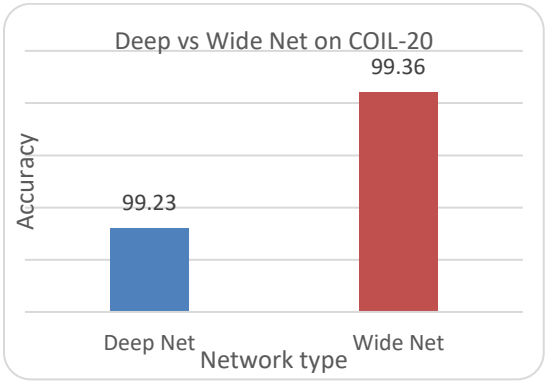
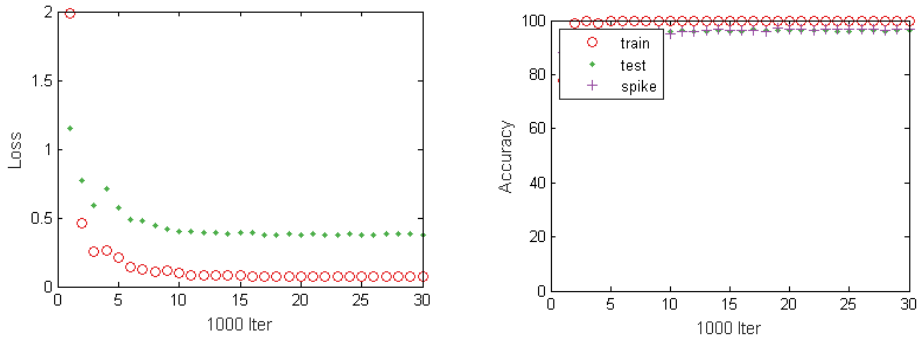


Figure 13.11: Testing with respect to deep versus wide network on COIL-20

COIL-100

The training loss and training accuracy are shown in Figure 13.12. Figure 13.12(a) shows the loss for training. According to the figure, it can be said that the convergence of the network during training is fast.



(a) (b)

Figure 13.12: Training loss and accuracy for COIL-100 dataset: (a) Training loss and (b) Training and testing accuracy.

Figure 13.12(a) shows the training loss for 30,000 iterations and Figure 13.12(b) shows the training and testing accuracy with red and green color respectively. The weights updating status during training is shown in Figure 13.13.

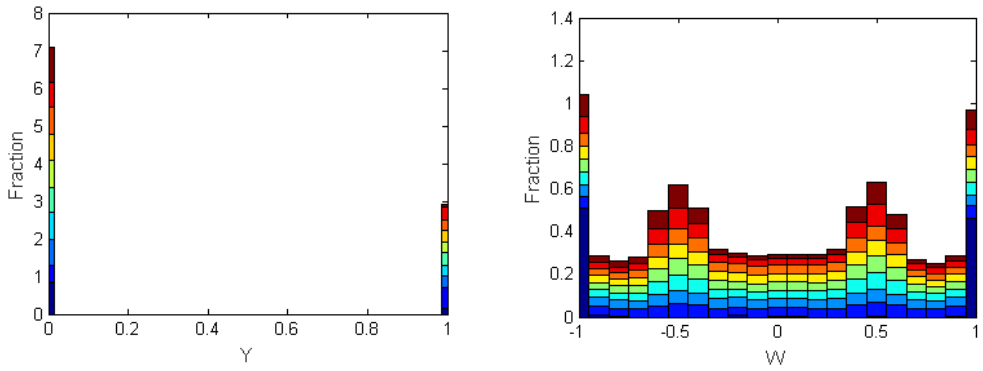


Figure 13.13: Weight update status during training.

We have also conducted the experiment with different wide and deep networks on COIL-100 for 30000 iterations. As with COIL-20, we have investigated different network with 520, 840, 2200, 4064, and 4094 cores. The experiment results are shown in Figure 13.14, and it is clearly shown that the bigger network with more cores performs better compared to the smaller network. We have achieved the best accuracy with the biggest network 4096 cores.

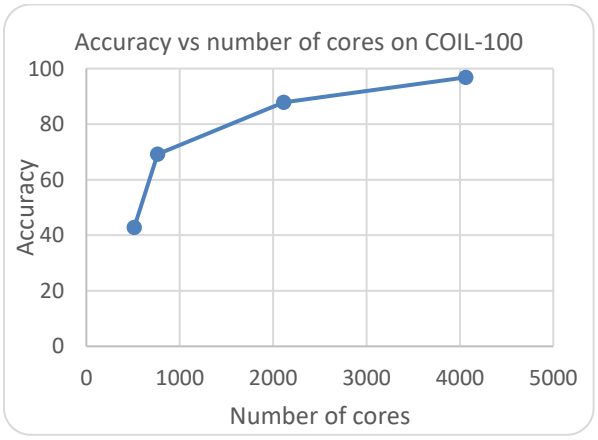


Figure 13.14: Testing accuracy versus a number of cores on COIL-100 dataset.

The experimental results for deeper versus wider networks are shown in Figure 13.15. The deeper network contained 15 layers and 4064 cores and wider version of the network contained 8 layers

with 4096 cores. The architectures of the deeper and wider networks are shown in Figure 2 and Figure 3 respectively. The wider network shows better results compare to deeper network in this case. The result shows around 96.8% testing accuracy on both simulator and TrueNorth chip.

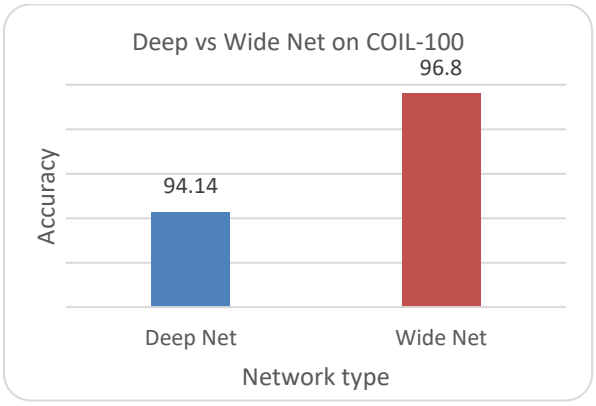


Figure 13.15: Deeper versus wider network on COIL-100.

13.7.3 Evaluation

When desiring to utilize more cores in IBM's TrueNorth Neuromorphic system, it is difficult to implement a network that utilizes the maximum number of core resources. We were limited to using a single chip TrueNorth system that contains 4096 cores. However, the architectures we have explored and evaluated in this experiment are limited to 4096 cores. The DCNN architecture on TrueNorth, the layers at the beginning of the network requires more cores whereas the posterior layer needs fewer cores for mapping onto the chip. For mapping the network onto the TrueNorth system, the splitter cores are used. It is observed from the network architecture that the wider network requires more splitter cores (1408) compare to the number of splitter cores (384) of the deeper network. From the experimental results, it is clearly observed that the recognition accuracy varies with respect to the network architecture and the number of cores on the TrueNorth system. As the number of classes increases, the wider network performs better compare against deeper network with an almost the same number of cores which are evaluated with a set of experiment.

In addition, we have implement DCNNS model with Keras and TensorFlow on the back end on a single GPU machine. This model consisted of seven layers including Softmax layer and contains around 0.5 million network parameters and training with ADAM optimizer with learning rate 0.001. We have achieved 100% accuracy for both COIL-20 and COIL-100 dataset. On the other hand, we have achieved 99.36% testing accuracy for COIL-20 and around 97% testing accuracy for COIL-100 on IBM's TrueNoth system. Although we have received 0.64% and around 3% less testing accuracy, the DCNN models on TrueNorth has a significant advantage in term of power.

13.8 Power Consumption

The power consumption of the TrueNorth system is compared with a traditional computing system in this section. Traditional computing systems, such as CPUs and GPUs easily consume 100W or more power, whereas an entire TrueNorth system consumes only up to 100mW to operate the 4096 cores on it. Here, we have used almost all cores for object recognition task with different network architectures. In addition, it is noted that about 50% of the power is passive power in TrueNorth system. The overall power requirement for 4096 cores is 100 mW. However, for 4064 cores of the deeper network, $0.5 * 100\text{mW} + (4064/4096) * 50\ \text{mW} = 99.64\ \text{mW}$ is required. We have achieved almost the same level of recognition accuracy which is achieved by CPU and GPU system for all datasets. However, implementation of the TrueNorth system requires significantly lower power with respect to the traditional computing system.

13.9 Conclusion

This work represents a very important step towards the evaluation of the impact of the architecture of DCNN and number of cores on recognition accuracy in a TrueNorth neuromorphic computing system. We have empirically evaluated the recognition accuracy of different DCNN models on three popular benchmarks including MNIST, COIL-20, and COIL 100 on IBM's TrueNorth system. We have achieved about 99.07%, 99.36% and 96.8% as the highest testing accuracy on MNIST, COIL-20, and COIL-100 respectively. The experimental result shows the wider version

of the network outperforms the deeper version of the network with the same number of cores on the TrueNorth system. We have achieved the highest accuracy with the wider network for COIL-20 and COIL-100 datasets with almost the same number of cores compared to the deeper network. In the future, we would like to conduct this experiment with more complex datasets on a multi-chip TrueNorth system.

CHAPTER 14

EFFECTIVE QUANTIZATION APPROACHES FOR RNN

Deep learning, Recurrent Neural Networks (RNN) have shown superior accuracy in a large variety of tasks including machine translation, language understanding, and movie frames generation. However, these deep learning approaches are very expensive in terms of computation. In most cases, Graphics Processing Units (GPUs) are in use for large-scale implementations. Meanwhile, energy efficient RNN approaches are proposed for deploying solutions on special purpose hardware including Field Programming Gate Arrays (FPGAs) and mobile platforms. In this paper, we propose an effective quantization approach for Recurrent Neural Networks (RNN) techniques including Long Short-Term Memory (LSTM), Gated Recurrent Units (GRU), and Convolutional Long Short-Term Memory (ConvLSTM). We have implemented different quantization methods including Binary Connect {-1, 1}, Ternary Connect {-1, 0, 1}, and Quaternary Connect {-1, -0.5, 0.5, 1}. These proposed approaches are evaluated on different datasets for sentiment analysis on IMDB and video frame predictions on the moving MNIST dataset. The experimental results are compared against the full precision versions of the LSTM, GRU, and ConvLSTM. They show promising results for both sentiment analysis and video frame prediction.

14.1 Introduction

Deep Neural Networks have been successfully applied and have achieved superior recognition accuracies in different application domains such as computer vision, speech processing, natural language processing (NLP), and medical imaging [2,3]. Several variants of deep learning approaches have been trained and tested with deeper and wider networks for achieving

classification accuracies which are similar to, or sometimes beyond, human level recognition accuracies. Typically, when the size of a neural network increases, it becomes more powerful and provides better classification accuracies. This comes at the significantly increasing costs of storage consumption, memory bandwidth, and computational cost. In most of the cases, the training is being executed on GPUs for dealing with big data volumes. This is very expensive in terms of power. In addition, deep learning approaches are expensive in terms of the number of networks parameters. This requires large storage and runtime memory for use. On the other hand, these types of massive scale implementations with large numbers of network parameters are not suitable for low power implementation, such as unmanned aerial vehicles (UAV), medical devices, a low memory system such as mobile devices, and Field Programmable Gate Arrays (FPGA).

Several research efforts are on-going to develop better networks with lower computation costs and fewer network parameters for low-power and low-memory systems without dropping classification accuracy. There are two main ways to design very efficient deep network structures: the first approach is by optimizing the internal operational cost with efficient network architectures. The second approach is to design networks with low precision operations for hardware efficient networks. As far as the network structure is concerned, the number of parameters can be reduced dramatically by using low dimensional convolutional filters in the convolutional layer as this also helps to add more non-linearity to networks [148,352]. One intuition is that larger activation maps (due to delayed down-sampling) can lead to higher classification accuracies [148]. This intuition has been investigated by K. He and H. Sun by applying delayed down-sampling into four different architectures of CNNs. It was observed that in each case, delayed down-sampling led to higher classification accuracies [353].

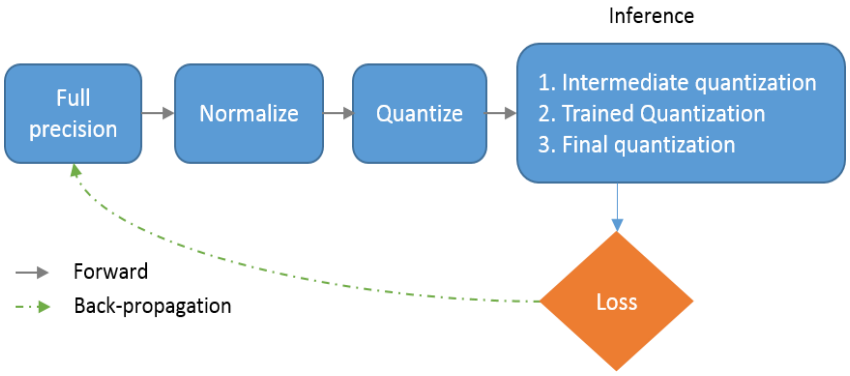


Figure 14.1: Quantization approach of deep neural networks [345].

Computation cost and memory can be saved significantly with lower precision multiplications and fewer multiplications through drop connection [354, 355]. These papers introduced Binary Connect Neural Networks (BNN) and Ternary Connect Neural Networks (TNN). Generally, multiplication of a real-valued weight by a real-valued activation (in the forward propagations) and gradient calculation (in the backward propagations) are the main operations of deep neural networks. Binary connect or BNN is a technique that eliminates the multiplication operations by converting the weights used in the forward propagation to be binary, i.e. constrained to only two values (0 and 1 or -1 and 1). As a result, the multiplication operations can be performed with simple additions (and subtractions), making the training process faster. There are two ways to represent real values to its corresponding binary values: deterministic and stochastic. In the deterministic technique, a straightforward thresholding technique is applied to the weights. In the stochastic approach, a matrix is converted to binary based on probabilities where the "hard sigmoid" function is used because it is computationally inexpensive. Experimental results show significantly better recognition performance on different benchmarks, including ImageNet [356, 344]. A flow diagram of the quantization approach is shown in Figure 14. 1, based on the recently published paper [345]. There are several advantages of BNNs: first, it is observed that binary multiplications on GPUs are almost seven times faster than traditional matrix multiplications on GPU. Second, in the forward pass, BNNs drastically reduce memory size and accesses and replace most arithmetic operations

with bit-wise operations, which leads to great increases of power efficiency. Third, binarized kernels can be used in CNNs, which can reduce the complexity of dedicated hardware by 60%. Forth, it is also observed that memory accesses typically consume more energy compare to arithmetic operations and that memory access costs increase with memory size. BNNs are beneficial with respect to both aspects.

Other techniques have also been proposed in the last few years [343,357,358]. Another power efficient and hardware friendly network structure has been proposed for CNNs with XNOR operations. In XNOR based CNN implementations, both the filters and inputs to the convolution layer are binary. This results in about 58x faster convolutional operations and 32x memory savings. In the same paper, Binary Weight Networks (BWN) have been proposed, which enable around 32x memory savings, allowing implementation of state-of-the-art networks on CPUs for real-time operations instead of GPUs. This model was tested on the ImageNet dataset and provided only 2.9% less classification accuracy than the full-precision AlexNet (in the top-1% measure). This network required less power and computation time. It accelerated the training process of deep neural networks dramatically for specialized hardware implementations [359]. The Energy Efficient Deep Neural Network (EEDN) architecture was first proposed for neuromorphic systems in 2016. In addition, they released a deep learning framework called EEDN, which provides accuracies that are very close to the state-of-the-art for almost all the popular benchmarks except the ImageNet dataset [328,331,368].

Some papers have been published recently which are based on quantization approaches proposed for RNNs [360, 361, 362]. However, in this paper, we have proposed effective quantization methods for RNN and empirically evaluated the performance of different datasets. The contribution of this work can be summarized as follows:

- Proposed effective quantization with binary connect, ternary connect, and quaternary connect approaches for RNNs.
- Evaluated on three different recurrent methods including LSTM, GRU, and ConvLSTM.

- Performance evaluation of LSTM and GRU for sentiment analysis on Amazon IMDB dataset.
- To our knowledge, the first time toward the evaluation of quantized ConvLSTM for video frame generation with the moving MNIST dataset.

14.2 Recurrent Neural Networks (RNN)

Human thoughts have persistence; Humans don't throw everything away and start their thinking from scratch every second. When you are reading a novel, you are understanding each word or sentence based on the understanding of previous words or sentences. The traditional neural network approaches including DNN and CNN cannot deal with this type of problems. The standard Neural Networks and CNNs are incapable of this due to the following reasons. First, these approaches only handle a fixed-size vector as input (e.g., an image or video frame) and produce a fixed-size vector as output (e.g., probabilities of different classes). Second, those models operate with a fixed number of computational steps (e.g. the number of layers in the model). The RNNs are unique as they allow operation over a sequence of vectors over time. A very basic RNN model, where the outputs from the hidden layers are used as inputs together with the inputs of hidden layers [363] is

$$h_t = \sigma_h(w_h x_t + u_h h_{t-1} + b_h) \tag{14.1}$$

$$y_t = \sigma_y(w_y h_t + b_y) \tag{14.2}$$

where x_t is the input vector, h_t is the hidden layer vector, y_t is the output vector, w and u are weight matrices, and b is the bias vector. A loop allows information to be passed from one step of the network to the next. An RNN can be thought of as multiple copies of the same network, each network passing a message to a successor. The diagram below shows what happens if we unroll the loop of an RNN model.

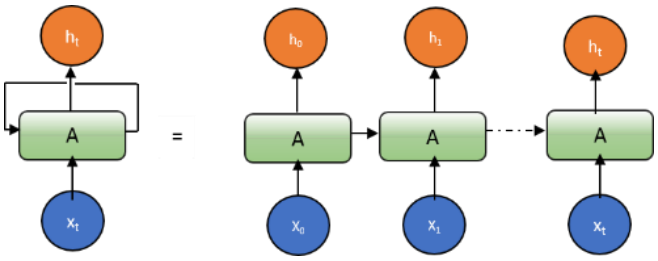


Figure 14.2: An unrolled RNNs.

The main problem is vanishing gradient problem to learn RNN approach depending upon the length of input sequences. For the very first time, this problem is solved by Hochreiter el at. in 1992 [364]. However, there are several solutions that have been proposed for solving the vanishing gradient problem of RNN approaches in recent decades. Two possible effective solutions of this problem are: first, clip the gradient (scale the gradient if its norm is too big) and second, better RNN models.

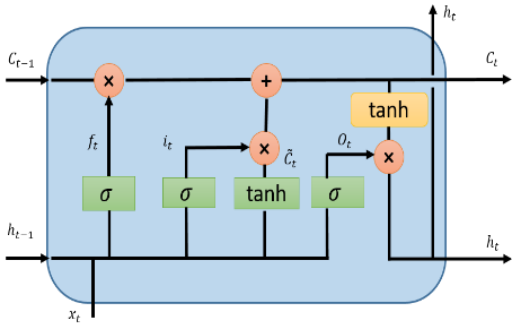


Figure 14.3: Diagram for Long Short Term Memory (LSTM).

14.2.1 Long Short Term Memory (LSTM)

The LSTM is introduced by Hochreiter and Schmidhuber in 1997 [365]. Further, an improved model of LSTM is introduced by Felix A. el at. in 2000 [371]. From then on, there are different variants of models that have been proposed based on this model. This improved version of RNN approaches allow larger sequences in the input, the output, or in the most general case, both and applying vastly for text mining, language understanding efficiently. The key idea of LSTMs is the cell state, the horizontal line running through the top of the Figure 14.3. LSTM removes or adds

information to the cell state called gates: an input gate(i_t), forget gate (f_t), and output gate(O_t) can be defined as:

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \quad (14.4)$$

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \quad (14.5)$$

$$\tilde{C}_t = \tanh(W_C.[h_{C-1}, x_t] + b_C) \quad (14.6)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (14.7)$$

$$O_t = \sigma(W_O.[h_{t-1}, x_t] + b_O) \quad (14.8)$$

the

$$h_t = O_t * \tanh(C_t) \quad (14.9)$$

The LSTM model is very popular for temporal information processing. Most of the paper includes the LSTM model with some variant, which is very minor.

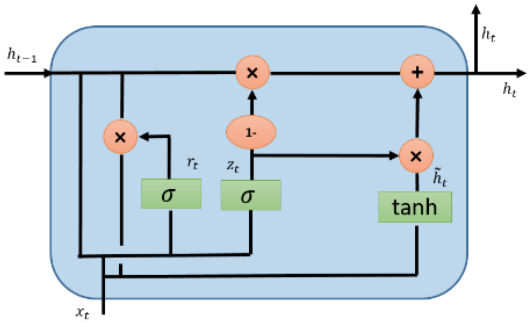


Figure 14.4: Diagram for Gated Recurrent Unit (GRU).

14.2.2 Gated Recurrent Unit (GRU)

The of the model of GRU also from LSTM with a slightly more variation by Cho, et al. in 2014, which is now very popular in the community working with recurrent networks. The main reason for the popularity is lower computation cost and simplicity of the model, which is shown in Figure 14.4. GRU is a significantly lighter version of RNN approach than standard LSTM in term of topology, computation cost, and complexity [366]. This technique is combined with the forgetting and input gates into a single "update gate" and merges the cell state and hidden state and makes

some other changes. The simpler model of GRU has been growing increasingly popular. Mathematically GRU can be expressed with the following equations:

$$z_t = \sigma(W_z.[h_{t-1}, x_t]) \tag{14.10}$$

$$r_t = \sigma(W_r.[h_{t-1}, x_t]) \tag{14.11}$$

$$\tilde{h}_t = \tanh(W.[r_t * h_{t-1}, x_t]) \tag{14.12}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \tag{14.13}$$

According to different empirical studies, there is no clear evidence of the winner. However, GRU requires fewer network parameters, which makes the model faster. On the other hand, LSTM provides better performance, if you have enough data and computational power [367].

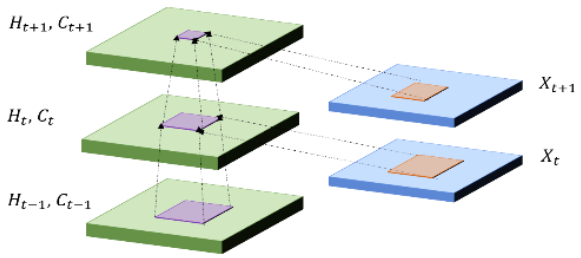


Figure 14.5: Pictorial diagram for ConvLSTM unit [357].

In this work, we have evaluated both the quantized version of LSTM and GRU for sentiment analysis in this implementation.

14.2.3 Convolutional LSTM (ConvLSTM)

The problem with fully connected (FC) LSTM in short FC-LSTM model is handling spatiotemporal data and its usage of full connection in input-to-state and state-to-state transactions, where no spatial information has been encoded. In ConvLSTM model, the internal gates of ConvLSTM are a 3D tensor, where the last two dimensions are spatial dimensions (rows and columns). The ConvLSTM determines the future states of a certain cell in the grid with respect to inputs and the

past states of its local neighbors which can be achieved using convolution operation in the state-to-state or inputs-to-states transition shown in Figure 14.5.

Figure 14.6: ConvLSTM layer with batch-normalization and 3D convolution.

ConvLSTM provides very good performance for temporal data analysis with video dataset [357]. Mathematically, the ConvLSTM is expressed as follows, where * represents the convolution operation and ∘ denotes for Hadamard product:

$$i_t = \sigma(w_{xi} . \mathcal{X}_t + w_{hi} * \mathcal{H}_{t-1} + w_{hi} \circ \mathcal{C}_{t-1} + b_i) \quad (14.14)$$

$$f_t = \sigma(w_{xf} . \mathcal{X}_t + w_{hf} * \mathcal{H}_{t-1} + w_{hf} \circ \mathcal{C}_{t-1} + b_f) \quad (14.15)$$

$$\widetilde{C_t} = \tanh(w_{xc} . \mathcal{X}_t + w_{hc} * \mathcal{H}_{t-1} + b_C) \quad (14.16)$$

$$C_t = f_t \circ C_{t-1} + i_t * \widetilde{C_t} \quad (14.17)$$

$$o_t = \sigma(w_{xo} . \mathcal{X}_t + w_{ho} * \mathcal{H}_{t-1} + w_{ho} \circ \mathcal{C}_t + b_o \quad (14.18)$$

$$h_t = o_t \circ \tanh(C_t) \quad (14.19)$$

In this implementation, we have used a very basic ConvLSTM structure where a single ConvLSTM layer, one batch-norm layer, and one 3D reconstruction layer are used. The basic diagram is shown in Figure 14.6.

14.3 Proposed Quantization Approaches

To quantize of the weights of a neural network, the quantization techniques are applied in the forward propagation, which reduces the operations compared to full precision and reduces memory requirement significantly. After calculating the loss of the model, weight gradients are updated with respect to the full precision weight values. The flow diagram according to the Ternary connect neural networks [345] is shown in Figure 14.1. According to the ternary connect quantization

method, the value of $\pm\Delta$ is optimized by minimizing the expectation of l_2 the distance between full precision and ternary weights. The maximum absolute value of the weights is used as a reference threshold to the layers and maintain a constant factor t for all the layers, which represents with $\Delta_l = t \times \max(|\tilde{w}|)$. They maintain a constant sparsity r for all layers throughout training and this hyperparameter r helps to obtain ternary weight networks with various sparsity's. The $t = 0.05$ is used in the experiment of CIFAR 10 and ImageNet dataset [345].

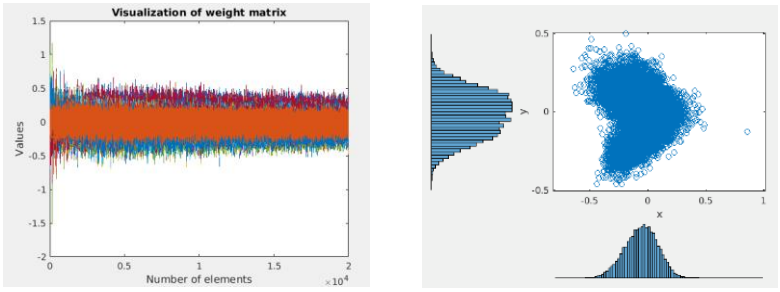


Figure 14.7: Visualization of weights on the left and weight distribution is shown on the right.

In addition, another very close work for ternary connect networks, the weights (W) are uniformly or normally distributed in $[-a, a]$ and Δ lies in $[0, a]$. In the case of uniform distribution: the approximated value is $\frac{1}{3}a$, which is equal to $\frac{2}{3}E(|W|)$. For a normal distribution, $N(0, \sigma^2)$, the approximated Δ^* is $\frac{1}{3}\sigma$, which is equal to $0.75 * E(|W|)$ is used. Finally, this paper proposed a rule of thumb that $\Delta^* = 0.75 * E(|W|) \approx \frac{0.7}{n}\sum_{i=1}^{n}|W_i|$, which is a strictly optimized threshold [345]. Furthermore, according to [360], the weights follow the characteristics of normal distribution and therefore they assume W has a symmetric distribution around zero. They scaled the mean absolute weights with a factor of 0.25 and evaluated for different bits for weights and activation. A straight-through estimator is used for this implementation [360].

Like others, we have determined the threshold values with basic statistics (mean and standard deviation) of weights in a layer. However, if we observe Figure 14.7, it shows that the weight

distribution is normal with mean (μ) and the standard deviation (σ). In addition, we observe that most of the weights values fall very close to zero. For binary connect neural networks, the thresholding is done with respect to zero on normalized weights of a layer. The equation is as follows:

$$\begin{matrix} \text{if } w_0 \geq 0 & 1 \\ \text{otherwise} & -1 \end{matrix} \tag{14.20}$$

Figure 14.8 shows the outputs distribution of weights after applying Eq. 14.20. It demonstrates clearly that the weights in a layer are uniformly distributed with respect to zero. Thus, we do not need to worry about the distribution of quantized weights for BC.

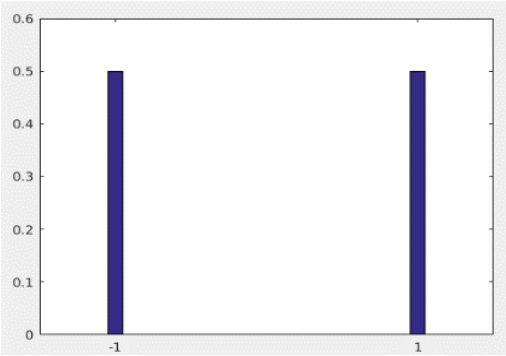


Figure 14.8: Distribution of weights for binary connect.

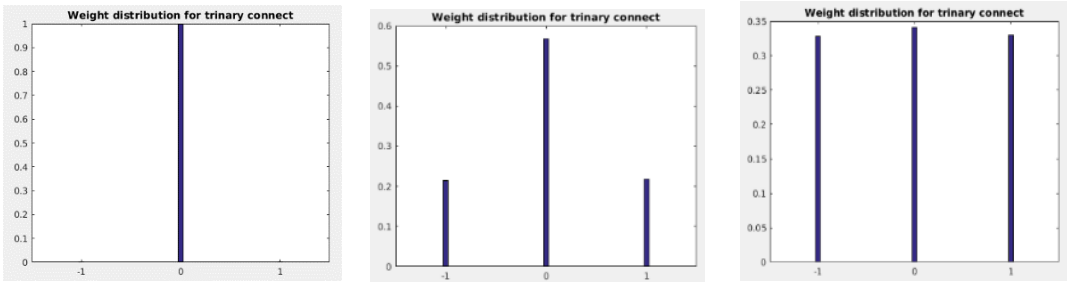
However, the ternary connects neural network contains the values of {-1, 0, 1}. In ternary connect networks, we have calculated the mean (μ_w) standard deviation (σ_w) of the weight of a layer. To achieve the approximate normal distribution of the quantize weights, the following equation is applied:

$$\begin{matrix} \text{if } w \leq -(\mu_w + \sigma_w) & -1 \\ -(\mu_w + \sigma_w) < w \leq (\mu_w + \sigma_w) & 0 \\ w > (\mu_w + \sigma_w) & 1 \end{matrix} \tag{14.21}$$

After applying the quantization with Eq. 14.22, the resulting quantized weights show approximated normal distribution which is stated in Fig 9(b). However, if we apply Eq. 14.22 then we achieve uniform distribution for the quantized weights.

$$\begin{cases} \text{if } w \leq -\left(\mu_w + \frac{\sigma_w}{2}\right) & -1 \\ -\left(\mu_w + \frac{\sigma_w}{2}\right) < w \leq \left(\mu_w + \frac{\sigma_w}{2}\right) & 0 \\ w > \left(\mu_w + \frac{\sigma_w}{2}\right) & 1 \end{cases} \tag{14.22}$$

The following figure shows normal and uniform distribution graphically in Figure 14.9(b) and Figure 14.9(c) respectively.



(a) (b) (c)

Figure 14.9: Weight distribution for ternary: (a) {-0.5, 0.5} (b) Weight distribution for Eq. 14.21, and (c) Weight distribution for Eq. 14.22 as threshold.

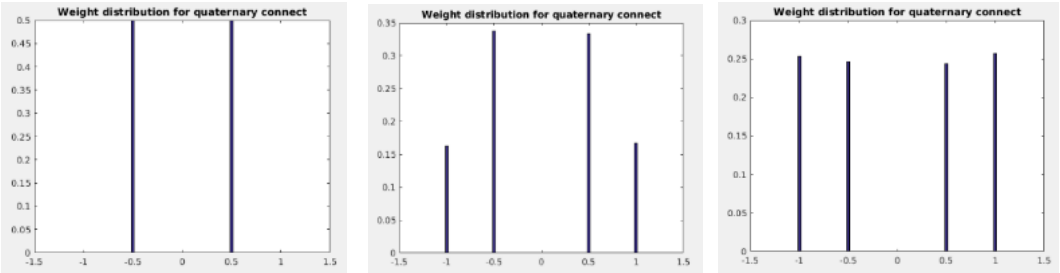
From Figure 14.9 (a), if we use {-0.5, 0.5} as threshold like other approaches, then the resulting weight distribution is unary where almost all weights get a single value of zero. However, we have applied thresholding according to the Eq. 14.21 and 14.22. This proposed approach ensures proper weight distributions shown in Figure 14.9(b) and Figure 14.9(c) respectively. In the QC approach, the {-1, -0.5, 0.5, 1} values are considered for weight representation. The threshold values are determined based on Eq. 14.23 and Eq. 14.24, which produces a normal and uniform distribution of quantized weights.

$$\begin{cases} \text{if } w \leq -\left(\mu_w + \frac{\sigma_w}{4}\right) & -1 \\ -\left(\mu_w + \frac{\sigma}{4}\right) < w \leq 0 & -0.5 \\ 0 < w \leq \left(\mu_w + \frac{\sigma_w}{4}\right) & 0.5 \\ w > \left(\mu_w + \frac{\sigma}{4}\right) & 1 \end{cases} \tag{14.23}$$

To implement the uniform distribution of weights after quantization

$$\begin{cases} \text{if } w \leq -\left(\mu_w + \frac{\sigma_w}{6}\right) & -1 \\ -\left(\mu_w + \frac{\sigma_w}{6}\right) < w \leq 0 & -0.5 \\ 0 < w \leq \left(\mu_w + \frac{\sigma_w}{6}\right) & 0.5 \\ w > \left(\mu_w + \frac{\sigma_w}{6}\right) & 1 \end{cases} \tag{14.24}$$

The visualization of weights distribution after applying different thresholding are shown in Figure 14.10. From the figure, it can be clearly observed that if we apply {-0.5, 0, 0.5} as threshold then instead of quaternary connected, it works like binary connect to the network with values of {-0.5, 0.5}, which is shown in Figure 14.10 (a). However, the proposed approach shows proper normal and uniform distribution of quantized weights which is shown in Figure10 (b) and Figure 14.10 (c) respectively.



(a) (b) (c)

Figure 14.10: Weight distribution for quaternary: (a) {- 0.5, 0, 0.5} (b) after applying Eq. 14.23 and (c) Outputs for Eq.14. 24 as threshold.

14.4 Results and Discussion

The entire experiment has been conducted in the Surface cluster of the Supercomputing Center at the Lawrence Livermore National Laboratory (LLNL) and is implemented with Keras and TensorFlow. We have evaluated our proposed quantization techniques for sentiment analysis on the IMDB dataset [369] and movie frame prediction task on the moving MNIST dataset [370]. Before going to the main experiment, we have experimented on a very simple summation problem for selecting appropriate weight distributions. We have evaluated the full precision (FP) and three

approaches with quantization including Binary Connect (BC), Ternary Connect (TC), and Quaternary Connect (QC).

```
[3, 10]] [13]
[' 3+10'] ['13']
[[11, 3, 10, 1, 0]] [[1, 3]]
[[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]]
[[[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]]]
```

Figure 14.11: Inputs, an encoding approach for summation problem.

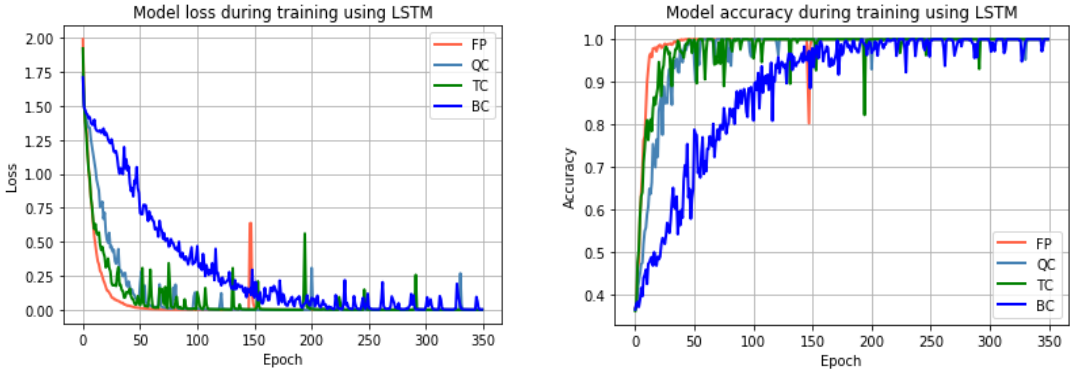


Figure 14.12: Model loss on the left and accuracy on the right for LSTM

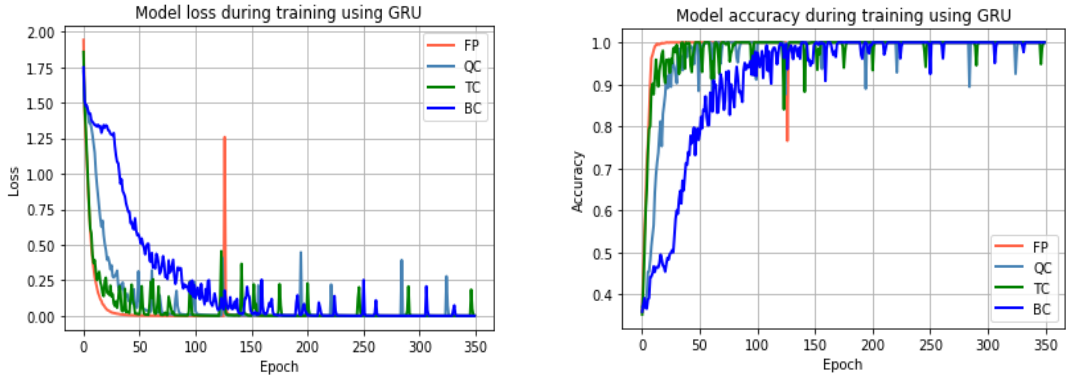


Figure 14.13: Model loss and accuracy for GRU are shown on the left and right side respectively.

The inputs set contains 12 characters including {‘0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '+', ' '}. We have encoded each character with a binary value which is shown with orange color in Figure 14.11. In the testing phase, after getting the encoding outputs shown in blue, we have decoded values for producing the desired outputs. Inputs in the first two rows and third row show the encoding position. For example, before 3 there is a space. Encoding position number is 11. Orange color represents the encoded vectors of inputs. The blue color shows the encoded outputs which are equivalent to 13. We have experimented for normal distribution (ND) and uniform distribution (ED) of weights after quantization for LSTM and GRU. Total 1000 samples per epoch are considered and this experiment is run for 350 epochs shown in Figure 14.11.

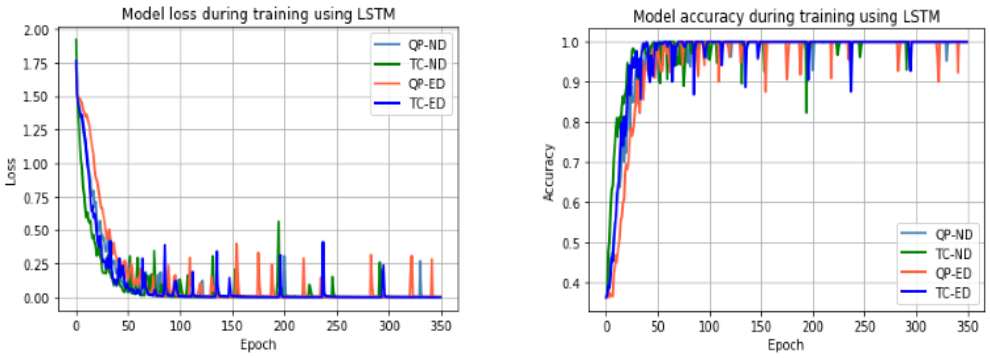
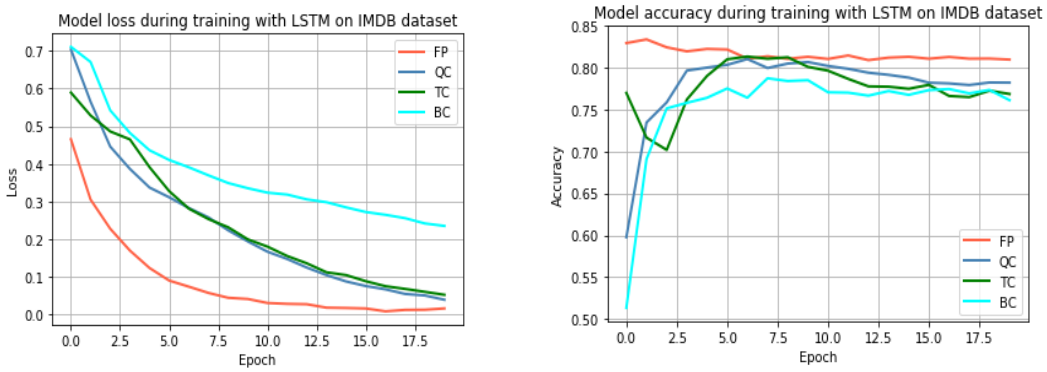


Figure 14.14: Loss for ND and ED using LSTM on the left and accuracy on the right.

Figure 14.12 and 13 show the training loss and accuracy for LSTM and GRU for the summation problem respectively. From Figure 14.12, it can be observed that the LSTM and GRU with full precision show better performance than other quantization approaches. The same behavior is observed for accuracy as well. It is also noticed that the TC and QC version of LSTM and GRU provides promising training accuracy compared to the of LSTM and GRU with full precision. Figure 14.14 shows the loss and accuracy of LSTM for normal and uniform distribution respectively. From the figure, it can be clearly observed that the approximate normal distribution performs better than the uniform distribution. Thus, the approximate normal distribution is used for TC and QC for the following experiments. The results are compared against the performance of

LSTM, GRU, and ConvLSTM with full precision (32 bits) for all datasets. In this experiment, we have used ADAM optimizer and binary cross entropy loss.



(a) (b)

Figure 14.15: Model loss and accuracy during training for LSTM. (a) Loss and (b) Accuracy.

14.4.1 Sentiment analysis

The experiment is conducted on the sequence to sequence problems for addition and IMDB sentiment analysis dataset. Here we report preliminary results that demonstrate the effectiveness of the proposed quantization methods on learning of recurrent models of LSTM and GRU. To accomplish this, the IMDB sentiment analysis dataset is used with max-feature numbers 20000, max number of words 80, and batch size 64. In both LSTM and GRU architectures, we have considered hidden units 128, and a number of epochs 20.

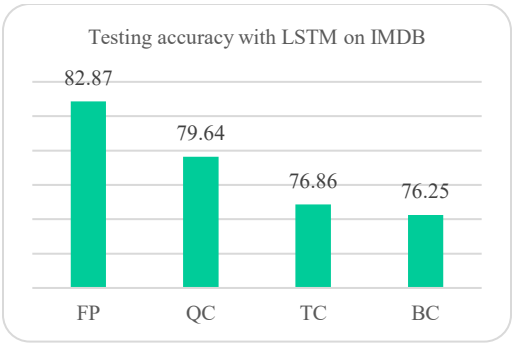
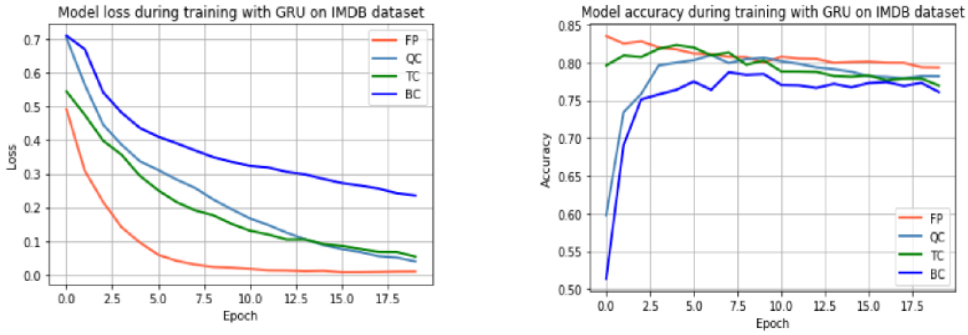


Figure 14.16: Testing accuracy in percentage using LSTM.

Results with LSTM: the training loss and validation accuracy with LSTM is shown in Fig 15. Fig 15 (a) shows that the LSTM with full precision converges much faster with the lowest loss compared to BC, TC, and QC. However, validation result shows good accuracy for sentiment analysis. In both cases, TC and QC provide better performance compared to BC. Figure 14.16 shows the testing accuracy on IMDB dataset. The experimental result shows testing accuracies of 82.87%, 79.64%, 76.86%, and 76.25% for FP, QC, TC, and BC respectively. We have achieved around 2.00% less on testing accuracy with QC and around 4% less accuracy compared against TC. There is, however, a significant advantage in terms of computational time and energy. In addition, this type of compressed version of recurrent approaches is suitable for embedded and mobile applications.



(a) (b)

Figure 14.17: Model loss and accuracy during training for GRU : (a) Loss and (b) Accuracy.

Results with GRU: training loss and validation accuracy for GRU are shown in Figure 14.17 (a) and (b) respectively. In this experiment, GRU with quantization of BC, TC, and QC gives very good testing accuracy with respect to the full precision GRU. Testing accuracy is shown in Figure 14.18.

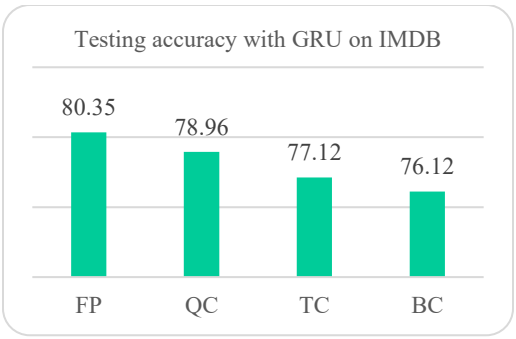


Figure 14.18: Testing accuracy in percentage using GRU.

However, LSTM provides overall better performance in most of the cases against GRU for sentiment analysis tasks.

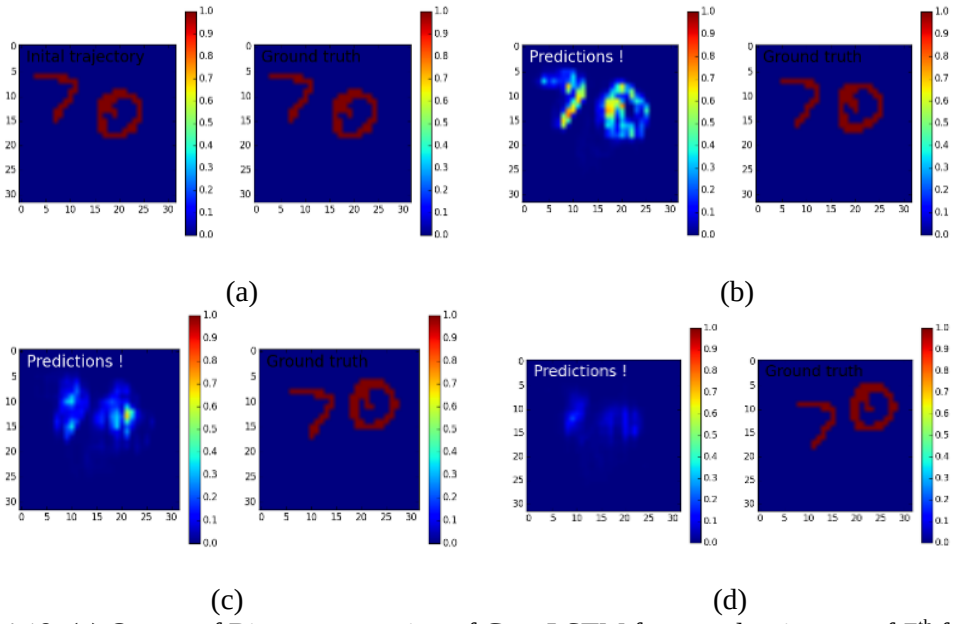


Figure 14.19: (a) Output of Binary connection of ConvLSTM for actual trajectory of 7th frame on left and ground truth on the right, (b) Predicted frame on the left and ground truth on the right for 8th number frame, (c) Predicted frame on the left and ground truth on the right for 9th number frame and (d) Prediction result for 10th frame.

14.4.2 Movie frames prediction

We have tested the performance of quantized ConvLSTM for object states prediction from the input video frames. We have implemented ConvLSTM with different quantization methods including BC, TC, QC, and ConvLSTM with full precision, which is tested on the moving MNIST dataset.

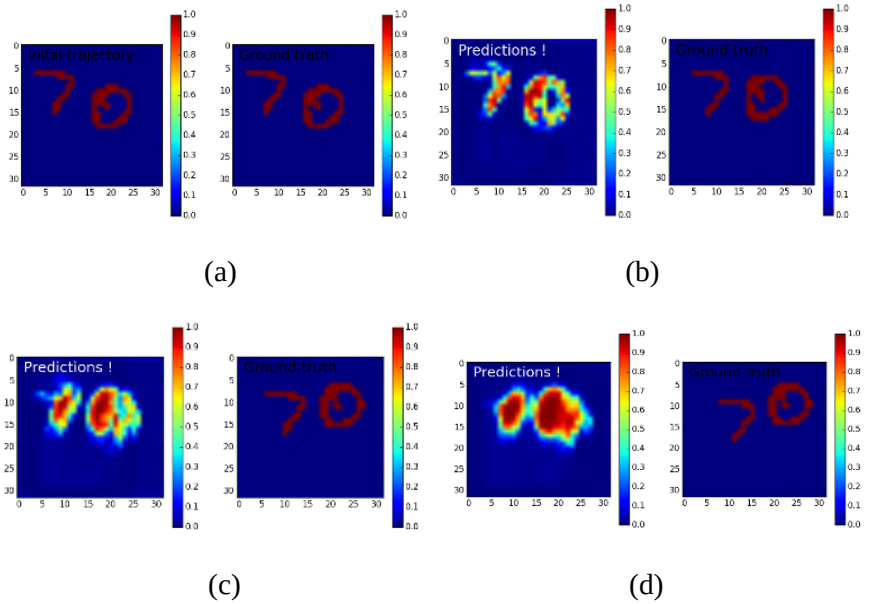


Figure 14.20: (a) Output of ternary connection of ConvLSTM for actual trajectory of 7th frame on left and ground truth on the right, (b) Predicted frame on the left and ground truth on the right for 8th number frame, (c) Predicted frame on the left and ground truth on the right for 9th number frame and (d) Prediction result for 10th frame.

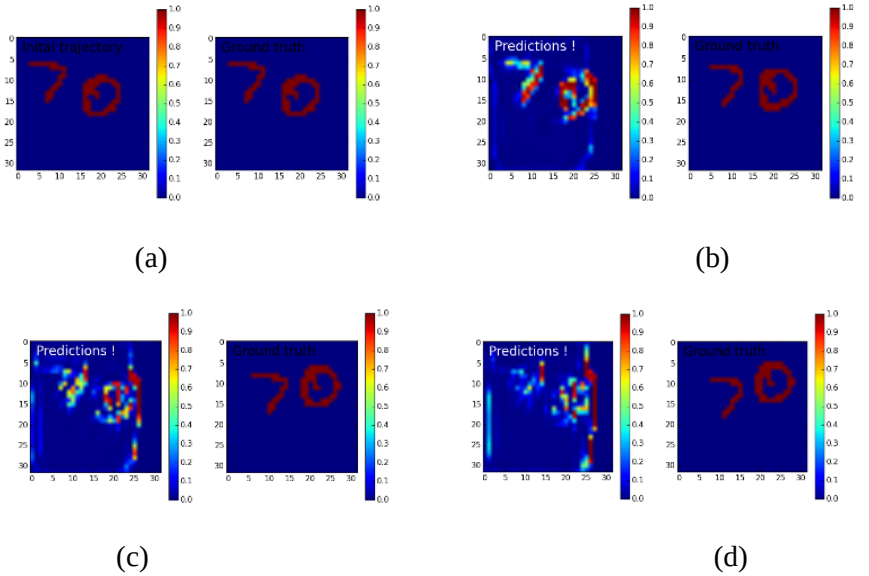


Figure 14. 21: (a) Output of quaternary connection of ConvLSTM for actual trajectory of 7th frame on left and ground truth on the right, (b) Predicted frame on the left and ground truth on the right for 8th number frame, (c) Predicted frame on the left and ground truth on the right for 9th number frame and (d) Prediction result for 10th frame.

There are 15 frames in a total of the input moving MNIST dataset where seven frames are used for training. After training successfully, we have tried to generate posterior frames from frame number 8. The experiment illustrates promising results for video frame prediction on the moving MNIST dataset. In this implementation, we have applied 50 epochs for training. The following figure shows the predicted frames with BC ConvLSTM. Figure 14.19 (a) shows the initial trajectory and ground truth which is 7th frame. The prediction and ground truth of 8th, 9th, and 10th frames are shown in Figure 14.19 (b), (c), and (d) respectively. The results for TC and QC are shown in Figure 14.20 and 14.21 respectively which demonstrates the qualitative performance of ConvLSTM. The outputs of ConvLSTM with full precision are shown in Figure 14.22. The experimental result shows good reconstruction compare to BC, TC, and QC. If we observe Figure 14. 22(d) then the reconstruction of 10th frame is much better than others. For analysis the performance of ConvLSTM on moving MNIST experiment, we have calculated the MSE between input frames and predicted frames. In the following equation, I is the input frame and K is the predicted frame:

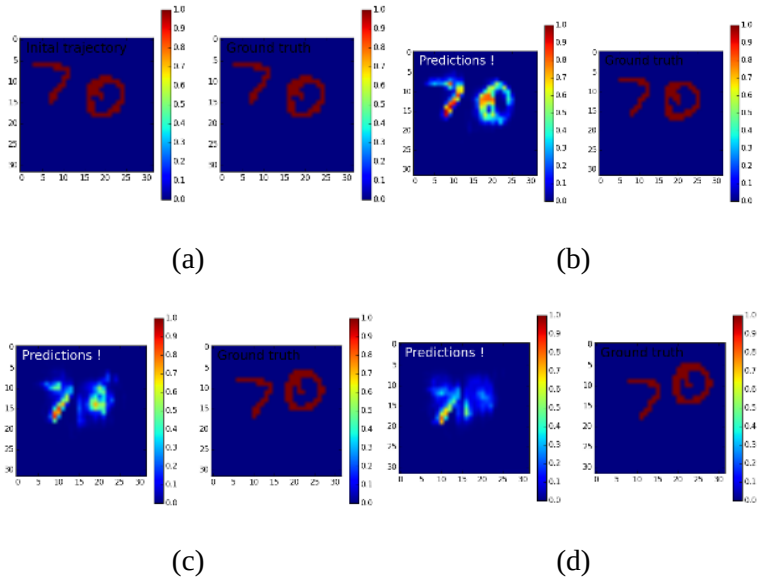


Figure 14.22: (a) Output of full precision of ConvLSTM for actual trajectory of 7th frame on left and ground truth on the right, (b) Predicted frame on the left and ground truth on the right for 8th number frame, (c) Predicted frame on the left and ground truth on the right for 9th number frame and (d) Prediction result for 10th frame.

$$MSE = \frac{1}{m\,n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2 \tag{14.25}$$

The following figure is showing the MSE for moving MNIST dataset where x-axis shows the number of frames and the y-axis shows the MSE with respect to the frame predicted.

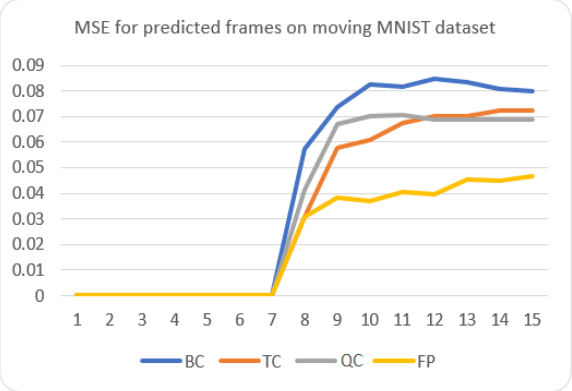


Figure 14. 23: MSE errors for frames prediction on the moving MNIST dataset.

According to Figure 14.23, it can be observed that the full precision ConvLSTM shows better performance in term of MSE compared to BC, TC, and QC. However, TC and QC also show promising results on the frames prediction task.

14.5 Conclusion

In this work, we have proposed efficient quantization approaches for Recurrent Neural Networks (RNNs) including Long Short-Term Memory (LSTM), Gated Recurrent Units (GRU), and Convolutional LSTM (ConvLSTM). The adaptive thresholding methods are proposed based on the basic statistics of the weights of a layer. We have also investigated the performance of approximate normal and uniform distribution of quantized weights for Binary Connect (BC), Ternary Connect (TC), and Quaternary Connect (QC) techniques. The empirical results show that the normal distribution shows better performance against uniform distribution with quantized weights. These proposed quantization methods are tested for sentiment analysis on IMDB sentiment analysis

dataset and frames prediction on moving MNIST dataset. The results show promising performance against full precision for LSTM, GRU, and ConvLSTM. It is noted that this is the first-time experimental evaluation of the performance of the quantized ConvLSTM approach for movie frame generation. In the future, we would like to evaluate the performance of quantized ConvLSTM for more complex datasets.

CHAPTER 15

QUBO ON TRUENORTH

The problems of Artificial intelligence (AI) naturally maps to NP-hard optimization problems. This trend has significance to achieve human-level computation capability from machines. This computational ability can be achieved by developing evolutionary algorithms or mapping those evolutionary algorithms onto new generation computing systems: Quantum or Neuromorphic hardware. In this paper, we implemented the NP-hard optimization problem called Quadratic Unconstrained Binary Optimization (QUBO) problem for the solution of graph problems on the IBM's Neurosynaptic TrueNorth System. We have experimented on different types of graph problems with different levels of complexities and achieved encouraging results on IBM's Neuromorphic TrueNorth chip. Moreover, there are a set of potential applications have been discussed based on this proposed QUBO solution. Along with the QUBO on quantum annealing, it is the important first step towards the solutions of QUBO on Neuromorphic computing systems.

15.1 Introduction

In the field of computer science, evolutionary computing is one of the subfields of artificial intelligence (AI). Evolutionary algorithms include general or optimization problems such as trial and error problem solver, global optimization, and meta-heuristic or stochastic optimization. These evolutionary or recombination approaches makes them less prone to get stuck in local optima than alternative methods. QUBO is a mathematical optimization programming problem, the objective of this approach is to find the minimum (or the maximum) value of a quadratic function with a

finite number of binary variables [372-374]. It has recently become quite heavily used, particularly in quantum computing. It can be defined by equation (15.1) and (15.2).

$$\mathrm{QUBO}(Q \in R^{nxn}; Q(i,j) = Q(j,i)) \tag{15.1}$$

Here Q is a nxn dimensional symmetric matrix with real numbers. The objective is to maximize or minimize the function. Here is the maximization function is given by equation (15.2).

$$\max\ Z = X^T Q X \quad \text{s.t. } X \in \{0,1\}^n \tag{15.2}$$

Here X is a vector with the elements of $\{0,1\}^n$. QUBO is a pattern recognition technique, it is commonly used in the field of machine learning and computer vision applications [372-374]. Even if it cannot provide complete solutions for a particular problem, it can be used for extracting useful features for computer vision applications by discovering persistence. This is a partial assignment of variables which must occur in an assignment to minimize the function. Moreover, persistence has already proved to be particularly useful for medical imaging applications [375]. QUBO is an NP-hard problem and thus is well-suited to algorithms aided by quantum annealing [1]. Furthermore, this is one of the best optimization problems, and can easily be reformulated to solve many applications including VLSI design [376, 377], economics and finance [378], manufacturing [379], data mining [380], computer vision [381], statistical mechanism [382], and discrete mathematics [383].

In this work, we have implemented and experimented with the QUBO problem as a solution for different graph problems and suggested potential applications in different fields. In particular, we have examined how to implement it on IBM TrueNorth Neurosynaptic hardware. The main contributions of this paper are:

- First ever implementation of the solution of QUBO problem on neuromorphic hardware.
- Implemented QUBO for solving graph problems.
- Discussed different potential problems that can be solved using the graph problem approach based on QUBO.

15.2 QUBO Problem and TrueNorth Chip

According to the definition of the QUBO problem in equations (15.1) and (15.2), it has only input values $X \in \{0,1\}^n$ and produces outputs $Y \in \{0,1\}^n$. It has only one symmetric matrix which is $Q \in Z^{nxn}$ to get the overall optimization. On the other hand, according to the architecture of the TrueNorth system, it only takes spike values as inputs $X \in \{0,1\}^n$ and produces the spikes $Y \in \{0,1\}^n$ as output. Moreover, the crossbar weights of TrueNorth chip are also represented with the values of $W_{i,j} \in \{0,1\}^{nxn}$. Only the synaptic weights have signed integer values -128 to 127. Therefore, according to the input and output constraints, the QUBO problem and TrueNorth architecture are reasonably well matched. One complication is that TrueNorth synaptic weights are binned into four types; thus each neuron has limited ability to represent a wide dynamic range of input weights across large numbers of inputs. In addition, according to the TrueNorth internal architecture, we can only map the values of the input symmetric matrix $Q \in Z^{nxn}$ of the QUBO problem as the synaptic weight of neurons which are ranged in signed integers between -128 to 127 on the TrueNorth system [328,329].

In order to use the TrueNorth neuromorphic hardware to map techniques like the QUBO problem (which is already implemented on quantum computing systems), the problem needs are formulated to map onto the architecture of the TrueNorth system. The following sections discuss the formulation process in details of the QUBO problem to map onto IBMs TrueNorth system.

15.3 Neuro-Synaptic Cognitive Chips

In 2015, IBM released the TrueNorth Neurosynaptic cognitive chip. The diagram of the TrueNorth system is shown in Figure 15.1. In this paper, we explore the use of the TrueNorth as an alternative computing system for solving NP-hard problems like QUBO for different applications with very low power and high energy efficiency [328,329]. The basic components of IBM's cognitive computing system are as follows: (a) IBM TrueNorth is based on a non-von Neumann architecture;

(b) has 4096 cores per chip, each core consists of 256 input axons and 256 output neurons connected with a 256x256 crossbar of configurable synapses; (c) each chip contains 1 million programmable neurons and 256 million synapses.

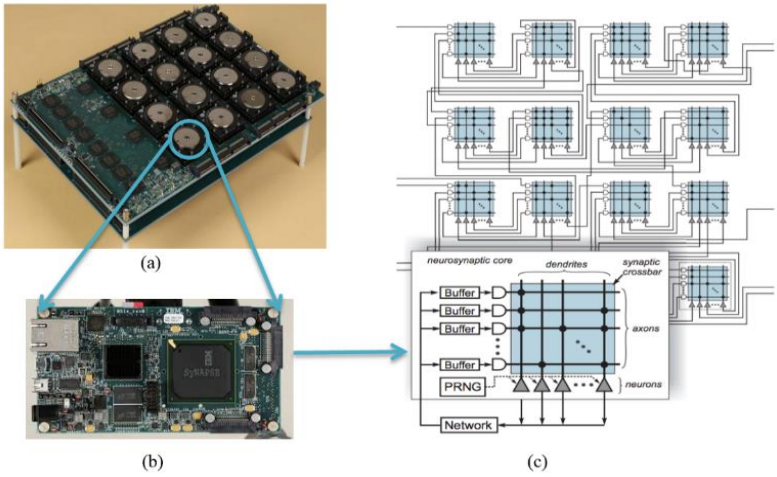


Figure 15.1: IBM's Neurosynaptic Cognitive TrueNorth Chips (a) TrueNorth multi-chip system (b) a single chip and (c) a zoomed-in internal structure of a single core.

Many constraints are considered during implementation of QUBO problem onto Neuromorphic systems. Traditional neural networks and deep learning algorithms are implemented using artificial neurons, whereas IBM's TrueNorth architecture uses versatile spiking neurons. Therefore, it is important as well as challenging, to discover a representation of data in spiking format to use in the TrueNorth system. The spiking neuron model was chosen to balance the dual objectives of capability (from a computational perspective) and cost (from an implementation perspective). The neurons' capability should be sufficient to support useful and potential cognitive algorithms [333], while the cost should be no more than necessary in terms of power, area, and speed. This cognitive architecture allows the possibility of reducing the computation cost in different ways. First, the power supply to the circuits can be turned off while they are quiescent, which reduces total power consumption. Second, the neurons can be implemented in an event-driven fashion that reduces power consumption as well.

In 2016, IBM released two TrueNorth system: the NSe1 single chip system and the NS16e, a 16 chip system where 16 chips are incorporated together for more flexible and more efficient reconfigurable Neuromorphic programming. These systems are capable of providing supercomputing scale computational capability with very low power consumption.

15.4 TrueNorth Architecture and Neuron Convention

In the TrueNorth architecture, each neuron's and synapse's states are updated in every millisecond. This duration is called a tick. From the architecture's point of view, each axon is assigned 0 of 3 axon types, which are used as an index into an "s-value" lookup table. The s-value table is unique to each neuron and provides a signed 9-bit integer synaptic strength to the corresponding synapse. The spikes are generated by neurons and can be the sent to any single axon on the chip. Each neuron in a core can be represented with about 23 individual programmable features such as synaptic weight, crossbar weight, threshold, leak, and reset. The TrueNorth architecture is very efficient because: (i) formation of neurons clusters which are created from inputs of similar pools of axons; (ii) spiking events only, which are sparse with respect to time and the communications among the cores performed through a long-distant communication network; (iii) the active power of this architecture is proportional to the firing activity [328,334,335].

15.4.1 Neurons

There are different types of neuron models that have been used in the TrueNorth system. The Leaky Integrate-and-Fire (LIF) neurons are used in this study. The basic operations are (see equations 15.3-15.5): 1. synaptic integration, 2. leak integration, 3. threshold, 4. spike firing, and 5. reset. In the general case, the LIF neuron model can be described by the following equations 15.3-15.5 [329].

Synaptic integration:

$$V_j(t) = V_j(t-1) + \sum_{i=0}^{N-1} x_i(t)\, s_i \tag{15.3}$$

Leak integration:

$$V_j(t) = V_j(t) - \lambda_j \quad (15.4)$$

Threshold, fire, and reset

$$\text{If} \quad V_j(t) \geq \alpha_j \quad (15.5)$$

Spike

$$V_j(t) = R_j$$

End-if

The parameter $V_j(t)$ represents for the summation of the membrane potential of the j^{th} neuron in the t^{th} timestep, and $V_j(t-1)$ is the sum of the membrane potential of the prior time-step. $x_i(t)$, and s_i are the synaptic inputs as the sum of spike inputs in the current time-step and signed synaptic weights respectively. The leak value λ_j is subtracted in every time-step from the membrane potential. The membrane potential is compared with the threshold voltage α_j every tick. If the membrane potential is greater or equal the desired threshold voltage, the neuron fires a spike and resets the membrane potential to zero.

15.4.2 Crossbar weight of TrueNorth system

The crossbar weights $w_{i,j} \in \{0,1\}$ of the neurosynaptic core are 0 or 1 (representing active or inactive states) and are represented using a single bit per weight. Moreover, each active synapse can have one of the four possible values as synaptic weight $s_j^{G_i}$ depending on the axon type. In this implementation, the crossbar weight of a core $w_{i,j}$ is structured according to the weight matrix of the input graph. Each element of the respective row of the input weight matrix is assigned as synaptic weight $S_{,j}$, which is shown in Figure 15.3.

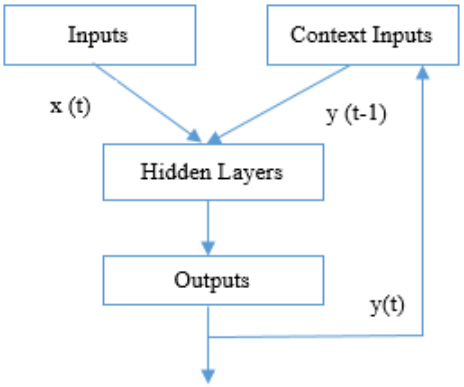


Figure 15.2: Vanilla Recurrent Neural Network.

15.5 Networks Structure

15.5.1 Recurrent neural networks(RNNs)

RNN has been used for implementing the overall proposed system. In a fully recurrent neural network, a Multi-layer Perceptron (MLP) with the previous set of hidden unit's activations feedback into the network along with the inputs (see Figure 15.2). According to the definition based on the structure of recurrent neural networks, the Jordan Recurrent Neural Network (JRNN) is used in this study [18]. In JRNN, the outputs of the neural network are used as inputs, with inputs of t^{th} time-step. Since the TrueNorth system only deals with spikes as inputs and outputs, we have implemented a spiking form of JRNN here, named Vanilla RNN in Fig 2. According to Figure 15.2, the inputs and outputs of the recurrent network are represented with $x(t)$ and $y(t)$ respectively. Here $h(t)$ is used for the outputs of the hidden layer and the context inputs of this recurrent network are represented with $y(t-1)$. This is the delayed version of output $y(t)$.

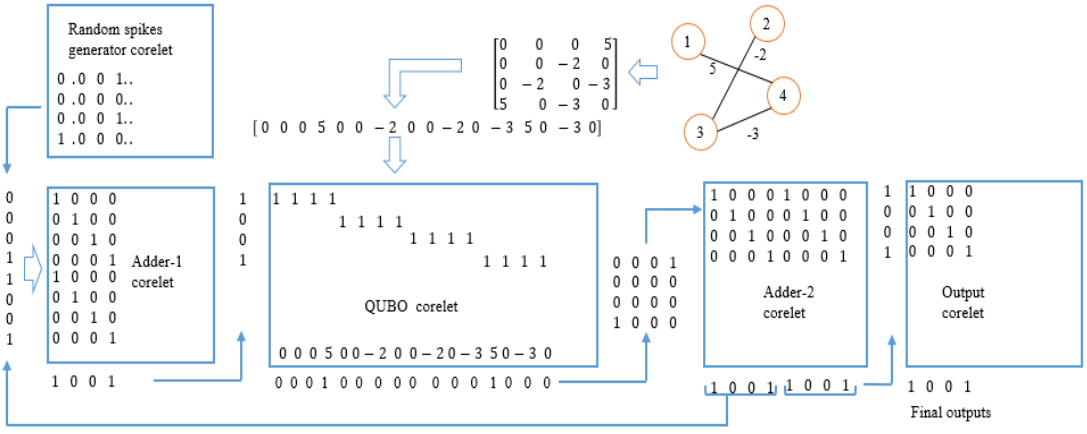


Figure 15.3: Overall implementation diagram for the QUBO problem.

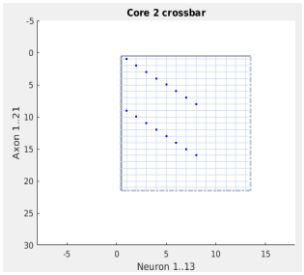
We have considered the weight matrices of W_{IH}, W_{HO}, and W_{OI} for inputs to hidden, hidden to outputs, and outputs to inputs respectively. A recurrent neural network can then be expressed using Eq. 15.6 and 15.7.

$$h(t) = f_H(W_{IH}x(t) + W_{OI}y(t-1)) \tag{15.6}$$

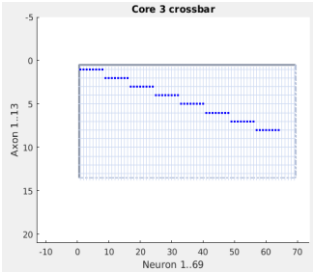
$$y(t) = f_O\left(W_{HO}h(t)\right) \tag{15.7}$$

15.5.2 Network Structure on TrueNorth system

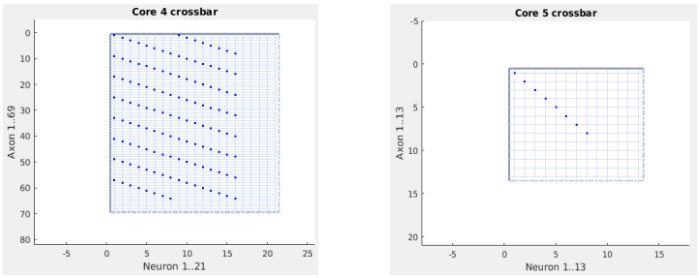
A diagram of the recurrent network on TrueNorth implementation with operational details is shown in Fig 3. In the TrueNorth chip, the recurrent inputs of adder-1 have a two time-tick delay on outputs from adder-2 in the very end of the implementation flow diagram in Figure 15.3. Four types of corelets are used for different operations: random spike generator, adder corelet, QUBO corelet, and output corelet. The random spike-generator generates the random spikes as inputs to the system. The number of spikes has been generated with respect to the dimensions of the input weight matrix based on a number of nodes in an input graph. For example, if the input dimension of the weight matrix is 4x4, it means that the number of nodes in the graph is 4. Nodes are connected with each other through positive or negative weights as shown in Fig 3. Thus, the number of input spikes is 4 for this graph. We have run for 50 and 100 time-ticks in this implementation. The adder (Adder-1) corelet accumulates the input spikes coming from a random spike generator in the present time-tick with the previous time-tick output spikes coming from the second adder named adder-2 close to the output unit. The synaptic weights are assigned after vectorization of the input weight matrix in row-major order. QUBO corelets perform the thresholding operation with respect to the value of the membrane potential of neurons. Since the QUBO corelet produces the outputs with respect to individual elements of the weight matrix, therefore the outputs need to be summed up for each column. Adder-2 performs the addition operation on the outputs on the QUBO corelet.



(a)



(b)



(c) (d)

Figure 15.4. Cores structure for 8 input nodes: (a) Adder corelet for adding inputs and recurrent inputs (b) QUBO corelet: crossbar weight respect to the rows of input weight matrix and encoded matrix element is used as synaptic weight (c) Produce two sets of outputs after addition on QUBO corelet outputs; one set of outputs are used as recurrent inputs and another set of outputs are used as input of output corelet. (d) Output corelet produces the final output.

There are two sets of outputs generated using Adder-2. One set of outputs is used for recurrent inputs and another output set is shown as final outputs. The output corelet shows the final spikes as outputs. The internal core structure of the TrueNorth implementation with respect to the different operations are shown in Figure 15.4. In this figure, the internal core structures have been shown for better understanding for an 8x8 input weight matrix for the solution of an 8 nodes input graph.

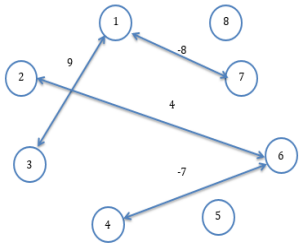
15.6 Results and Discussion

In the study, we have experimented with the QUBO problem for solving different kinds of graph problems. The experiments have been conducted for different graph problems with different numbers of nodes and different levels of structural complexity of graphs. The entire process has been executed and evaluated on IBM's Neuromorphic TrueNorth system. All of the experiments have been conducted on Lawrence Livermore National Laboratory's Surface cluster and TN clusters. The entire model is written in MATLAB, using the integrated programming environment called corelet programming for IBM's Neurosynaptic system [334]. There are two implementation platforms. The first simulation platform is called Neurosynaptic Simulator for Corelet System (NSCS) and another is in actual hardware. The exact same program can be run on the actual TrueNorth chip or simulator depending upon a flag that is assigned "TN" or "NSCS" respectively.

It is noted that the outputs of both environments are identical. In this experiment, we tested our system in both environments and the results which are represented here are tested on IBM's TrueNorth system. The experimental results are described in detail in the following paragraphs.

15.6.1 Input graph 1

The first input graph and the corresponding weight matrix of the graph are shown in Figs. 5(a) and (b) respectively. According to the graph, the expected solution set is {1, 2, 3, and 6}.

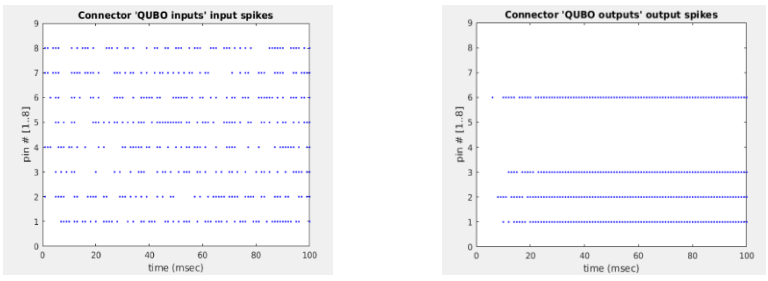


0	0	9	0	0	0	-8	0
0	0	0	0	0	4	0	0
9	0	0	0	0	0	0	0
0	0	0	0	0	-7	0	0
0	0	0	0	0	0	0	0
0	4	0	-7	0	0	0	0
-8	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(a) (b)

Figure 15.5: (a) Input weighted graph and (b) weight matrix for a weighted graph.

Figure 15.6(a) shows the random eight-input spikes over a 100 time-tick period. The number of spikes is selected with respect to the number of nodes in the input graph. Figure 15.6(b) shows the final outputs spikes on the TrueNorth system. The spikes of respective pin numbers show the active node on the input graph. According to the outputs in Figure 15.6(b), it is clearly seen that only pins {1, 2, 3, and 6} are active. In this implementation, a zero value has been assigned as the threshold of neurons.

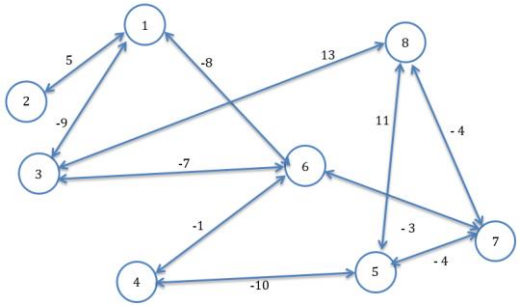


(a) (b)

Figure 15.6. (a) Random input spikes (b) TrueNorth outputs {1, 2, 3, and 6}.

15.6.2 Input graph 2

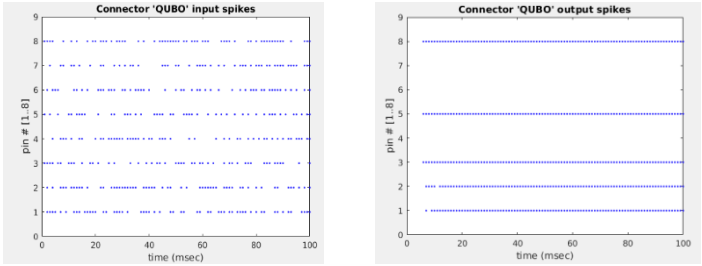
The graph in Figure 15.7(a) has the same number of input nodes as the graph in Figure 15.5(a) but is a more complex graph. This graph contains more node to node connections compared to the graph in Figure 15.7(a). The weight matrix of the above graph is given in Figure 15.7(b). The solution of this graph is {1, 2, 3, 5, and 8}. The outputs of the TrueNorth system shows the exact solution of {1, 2, 3, 5, and 8} as spiking form in Figure 15.8(b)



0	5	-9	0	0	-8	0	0
5	0	0	0	0	0	0	0
-9	0	0	0	0	-7	0	13
0	0	0	0	-10	-1	0	0
0	0	0	-10	0	0	-4	11
-8	0	-7	-1	0	0	-3	0
0	0	0	0	-4	-3	0	-4
0	0	13	0	11	0	-4	0

(a) (b)

Figure 15.7: (a) Input weighted graph (b) Input weight matrix for the above-weighted graph.

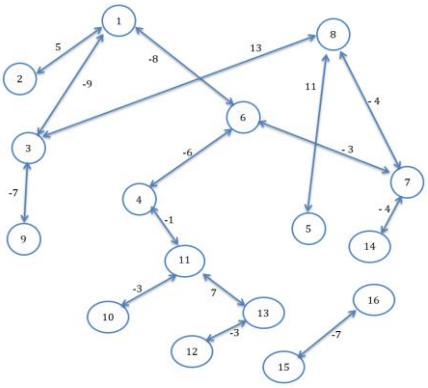


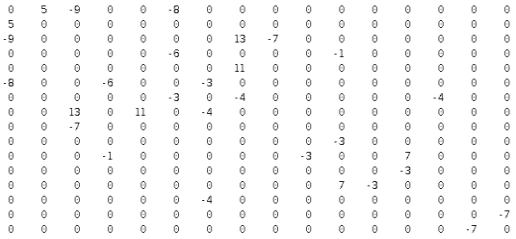
(a) (b)

Figure 15.8: (a) Input spikes (b) TrueNorth output spikes with solution set {1, 2, 3, 5, and 8}.

15.6.3 Input graph 3

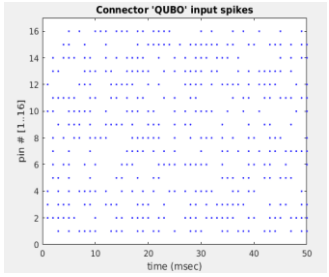
This graph has more nodes and is even more complex in terms of the node to node connectivity. The solution of this graph is {1, 2, 3, 5, 8, 11, and 13}. We have experimented with 16 spikes during a 50 time-tick period for a better understanding of convergence. The TrueNorth system provides accurate solution set {1, 2, 3, 5,8,11, and 13}, as shown in Figure 15.10. (b).

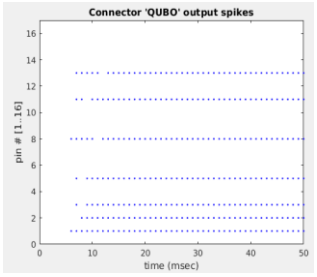




(a) (b)

Figure 15.9: (a) Input graph (b) The weight matrix of the above graph.





(a) (b)

Figure 15.10: (a) Input spikes (b) TrueNorth output spikes with active pins set {1, 2, 3, 5, 8, 11, and 13}.

15.6.4 Input graph 4

We have again considered another graph with eight nodes on it. However, the exceptionality of this input graph is that it has two identical solutions on it which are shown in Figure 15.11. The solution set of this input graph is {1, 2, 5, and 6}. Random inputs are considered for this experiment over a period of 50-time ticks. It is clearly seen that the proposed system provides an accurate solution set for this input graph as shown in Figure 15.12 (b).

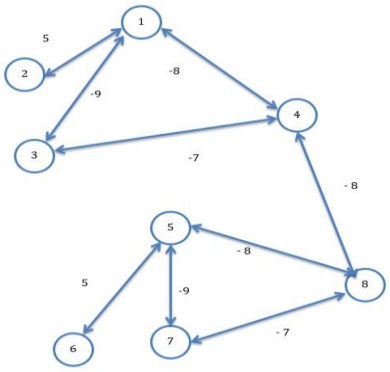
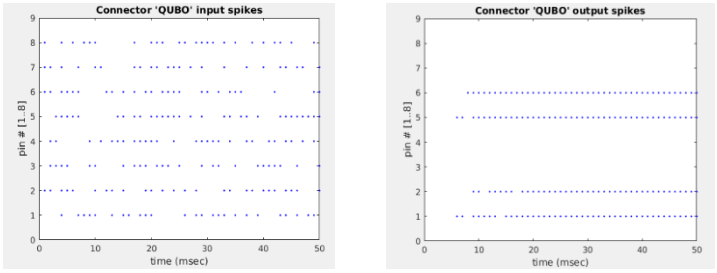


Figure 15.11: Input graph with two identical solutions.



(a) (b)

Figure 15.12: (a) Input spike (b) TrueNorth output spikes for 50 time ticks with solution set of {1, 2, 5, and 6}

From the above experimental results, it is clearly observed that the solutions are converged within 10 to 20 time-tick depending upon the complexity of graph as well as the pattern of random input spikes. This is quite fast in terms of convergence. Furthermore, the QUBO problem can be implemented for a minimization value function. The proposed technique can be easily implemented for minimization approach by changing the sign of individual elements of the input weight matrix. A hierarchical approach for multi-level thresholding using QUBO on TrueNorth: in most of the cases for solving a particular problem, we usually have predefined values for making the decision in most real-life problems. In the TrueNorth system, we can easily implement multi-level thresholding after encoding the values according to a neuron's capacity to -128 to 127 in the present architecture. First, the highest value will be considered as a threshold, and the active nodes will be shown as results. Then the highest value of the matrix elements is replaced by zero and

applied to the second level for thresholding. The same approach can be repeated as required. Therefore, we can easily implement a hierarchical thresholding approach for making a multi-level decision of any task.

15.7 Potential Applications

The solution of the QUBO problem on the TrueNorth system can be used for different applications like other applications [335,384,385]. Several prospective applications are discussed in detail in the following paragraphs.

15.7.1 Image segmentation

In IBM's Neuromorphic system, image segmentation has been implemented using a convolutional approach which is computationally expensive and needs more cores. Image segmentation can be easily implemented using this proposed approach on the TrueNorth chip. For example: in case of skin color model for face detection, if we considered the RGB color image then based on the values of R, G, and B according to the skin color model, a segmentation operation can be performed for detecting faces in an input image on the TrueNorth system.

15.7.2 Vehicle density or speed limit estimation and traffic monitoring system

Consider different areas or particular highways as nodes of a graph for a traffic monitoring system. Based on the highest capacity or density of vehicles, the vehicle density can be estimated for traffic monitoring systems on TrueNorth. In the TrueNorth implementation, the quantized values of the number of vehicles of a node can be used as synaptic weights. Then the desired threshold can be applied for synaptic weights. For measuring different levels of density, the different QUBO problems for different levels of thresholding can be implemented on TrueNorth. TrueNorth based systems will be very fast and it will be very convenient in term of power efficiency. The conceptual diagram is shown in Figure 15.13.

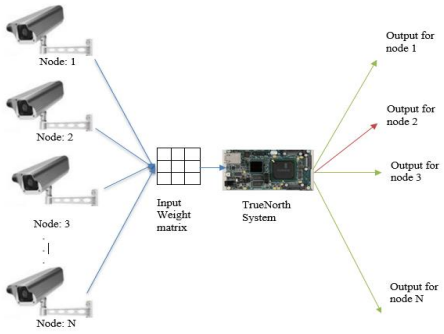


Figure 15.13: Conceptual diagram for TrueNorth based traffic monitoring system.

15.7.3 Cellular network user density estimation

Individual cellular network towers can be considered as nodes of a graph problem (see Figure 15.14). The predefined value for the highest number of user capacity under a cellular network tower can then be used as the threshold for the node representing that tower. The overflow of user density can be estimated based on the current number of users under that network tower in a particular time.

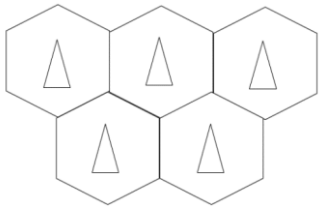


Figure 15.14: Diagram for a cellular network node for user density estimation.

15.7.4. Server or supercomputing monitoring system

Any supercomputing center or computer cluster needs to be monitored for different parameters such as bandwidth, power consumption, heat, and load balancing. If each machine or a cluster is considered as a node, then based on the predefined value of the different parameter, the proposed system can be easily implemented to make decisions or predict which parameter exceeds the expected values. Therefore, we can take immediate action with respect to the problems.

15.7.5 True weather monitoring system

The proposed solution for the QUBO problem can be applied to forecasting weather information such as temperature, humidity, and wind speed. Depending on the different environments or locations, the value of weather forecasting parameters can vary. For example, if there is a predefined range of values of wind speed for tornadoes or cyclones, it can be easily implemented in massive scale, very fast, energy efficient, and distributed solution for weather monitoring system. There are many more possible applications of this solution such as airplane scheduling systems, tree problem solutions, and many more.

15.8 Power Consumption

The TrueNorth system is very efficient in term of power consumption compared to traditional computing systems. A traditional computing system, such as CPUs and GPUs, easily consume around 100W or more power. On the other hand, an entire TrueNorth chip consumes only up to 100mW to operate 4096 cores. It should be noted that about 50% of this power is passive power in the TrueNorth system. In this implementation, we have only used 5 cores for 16x16 input matrix dimensions or smaller. This means that this system required only $0.5 * 100\text{mW} + (5/4096) * 50\text{ mW} = 50.061\text{ mW}$. If we analyze the power consumption more precisely with respect to neurons for 16x16 input matrix, then we have used only 16 neurons in the random number generator, 256 neurons in the QUBO core, (16+32) neurons in the adder core, and 16 neurons in the output core. Therefore, we have used only $(16 + 256 + 48 + 16) = 336$ neurons across 5 cores. The actual power consumption is $0.5 * 100\text{mW} + [\frac{\{(\frac{1}{4096}) * 50\text{ mW}\}}{256} * 336] = 50.016\text{ mW}$. Even if there is an overhead to use cores, there is still an order of magnitude power consumption difference between the Neuromorphic computing system and traditional computing system. This extremely low power system can be used for low-power devices or applications such as mobile devices, sensor applications, and robotics.

15.9 Conclusion and Future Works

This work represents a very important first step towards the solution of Quadratic Unconstrained Binary Optimization (QUBO) technique as a solution of graph problems on the extremely low power TrueNorth Neurosynaptic computing system. It shows promising experimental results for different types of graph problems with different levels of constraints and complexity. Moreover, the potential applications of this proposed technique have been discussed for different fields of monitoring, tracking, etc. Presently we have been working on better implementations of the QUBO problem on TrueNorth with new techniques for solving graph problems using more nodes. As future work, we would like to implement this proposed approach for prospective applications described in the paper. We further want to experiment with this QUBO solution for determining molecular similarity.

CHAPTER 16

CONCLUSION

In this thesis, we have developed and evaluated several improved Deep Convolutional Neural Network (DCNN) models including the Inception Recurrent Convolutional Neural Network (IRCNN), Improved Recurrent Residual CNN (IRRCNN), Densely Connected Recurrent Network (DRCN), Recurrent Residual U-Net (R2U-Net), R2U-Net based regression named University of Dayton Network (UD-Net). These models are applied for classification, segmentation, and detection tasks in the field of computer vision and Bio-medical imaging. First, the IRCNN, IRRCNN, and DCRN are proposed for classification tasks and evaluated on computer vision and Bio-medical imaging, the experimental results show superior performance against equivalent Inception, Residual, Inception Residual Networks in object classification tasks. To generalize IRRCNN model, the IRRCNN model is applied for White Blood Celle (WBC), Red Blood Celle (RBC), Breast Cancer, and Nuclei classification problems and achieved state-of-the-art testing accuracy compared to recently published results. Second, we have proposed R2U-Net and applied for medical image segmentation tasks which show state-of-the-art performance compared to SegNet, and U-Net models for retinal blood vessel segmentation, skin cancer segmentation, and lung segmentation problems. Third, we have developed a regression model for detection task with R2U-Net which is named UD-Net that is tested for nuclei and lymphocyte detection shows superior performance against existing machine learning and deep learning-based approaches. In addition, we have evaluated our proposed models for seven different tasks in digital pathology including Invasive ductal carcinoma (IDC) detection, Lymphoma classification, Epithelium segmentation, Tubule segmentation, Nuclei segmentation, Lymphocyte detection and Mitosis detection for

computational pathology. In all cases, our models provide better performance compared to existing machine learning and deep learning-based approaches. Furthermore, we have implemented and evaluated energy efficient DCNN models object classification problem on IBN's Neuromorphic system. The Convolutional Sparse Coding (CSC) is implemented on IBM's TrueNorth system for the very first time and shows promising sparse reconstruction results on MNIST and CIFAR-10 datasets. The deep versus wide DCNN models are implemented and evaluated on TrueNoeth and concludes that for the higher number of classes, the wider DCNN model outperforms compared to deeper models on IBM's TrueNorth system. Finally, we have proposed effective quantization methods for different Recurrent Neural Networks (RRN) and have solved NP-hard Quadratic Unconstrained Binary Optimization (QUBO) problem on IBM's TrueNorth Neuromorphic system.

REFERENCES

1. Alom, Md Zahangir, Tarek M. Taha, Christopher Yakopcic, Stefan Westberg, Mahmudul Hasan, Brian C. Van Esesn, Abdul A. S. Awwal, and Vijayan K. Asari. "The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches." *arXiv preprint arXiv:1803.01164* (2018).
2. Jump, *Schmidhuber, J. (2015). "Deep Learning in Neural Networks: An Overview". Neural Networks.* ***61****: 85–117.*
3. Litjens, Geert, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen AWM van der Laak, Bram van Ginneken, and Clara I. Sánchez. "A survey on deep learning in medical image analysis." *Medical image analysis* 42 (2017): 60-88.
4. Greenspan, Hayit, Bram van Ginneken, and Ronald M. Summers. "Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique." *IEEE Transactions on Medical Imaging* 35, no. 5 (2016): 1153-1159.
5. Esteva, Andre, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, and Sebastian Thrun. "Dermatologist-level classification of skin cancer with deep neural networks." *Nature* 542, no. 7639 (2017): 115.
6. Han, Zhongyi, Benzheng Wei, Yuanjie Zheng, Yilong Yin, Kejian Li, and Shuo Li. "Breast cancer multi-classification from histopathological images with structured deep learning model." *Scientific reports* 7, no. 1 (2017): 4172.
7. Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet classification with deep convolutional neural networks. In NIPS, pp. 1106–1114, 2012.

8. Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. CoRR, abs/1311.2901, 2013. Published in Proc. ECCV, 2014.
9. Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556*(2014).
10. Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.
11. He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
12. https://www.prnewswire.com/news-releases/deep-learning-in-medical-imaging-a-300m-market-by-2021-300408645.html
13. https://www.cancer.gov/research/key-initiatives/moonshot-cancer-initiative
14. Arulkumaran, Kai, et al. "A brief survey of deep reinforcement learning." *arXiv preprint arXiv:1708.05866* (2017).
15. Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." *Nature* 518.7540 (2015): 529-533.
16. Mnih, Volodymyr, et al. "Playing Atari with deep reinforcement learning." *arXiv preprint arXiv:1312.5602* (2013).
17. Fukushima, K., & Miyake, S. (1982). Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In Competition and cooperation in neural nets (pp. 267-285). Springer Berlin Heidelberg.
18. LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324.
19. Lin, Min, Qiang Chen, and Shuicheng Yan. "Network in network." *arXiv preprint arXiv: 1312.4400* (2013).
20. Springenberg, Jost Tobias, et al. "Striving for simplicity: The all convolutional net." arXiv preprint arXiv: 1412.6806 (2014).

21. He, Kaiming, et al. "Identity mappings in deep residual networks." European Conference on Computer Vision. Springer International Publishing, 2016.

22. Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." arXiv preprint arXiv: 1502.03167 (2015).

23. Szegedy, Christian, et al. "Rethinking the inception architecture for computer vision." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016.

24. Szegedy, Christian, et al. "Inception-v4, inception-resnet and the impact of residual connections on learning." arXiv preprint arXiv: 1602.07261 (2016).

25. Zagoruyko, Sergey, and Nikos Komodakis. "Wide residual networks." arXiv preprint arXiv: 1605.07146 (2016).

26. Huang, G., Liu, Z., Weinberger, K.Q., van der Maaten, L., 2016. Densely connected convolutional networks. arXiv preprint arXiv:1608.06993 .

27. Larsson, Gustav, Michael Maire, and Gregory Shakhnarovich. "FractalNet: Ultra-Deep Neural Networks without Residuals." *arXiv preprint arXiv:1605.07648* (2016).

28. Sze, Vivienne, et al. "Efficient processing of deep neural networks: A tutorial and survey." *Proceedings of the IEEE*105.12 (2017): 2295-2329.

29. Van Essen, Brian, et al. "LBANN: Livermore big artificial neural network HPC toolkit." *Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments*. ACM, 2015.

30. Chen, Xue-Wen, and Xiaotong Lin . “Big Data Deep Learning: Challenges and Perspectives” IEEE Access in date of publication May 16, 2014.

31. Russakovsky, Olga, et al. "Imagenet large scale visual recognition challenge." *International Journal of Computer Vision* 115.3 (2015): 211-252.

32. Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234-241. Springer, Cham, 2015.

33. Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431-3440. 2015.

34. Zacharaki, Evangelia I., Sumei Wang, Sanjeev Chawla, Dong Soo Yoo, Ronald Wolf, Elias R. Melhem, and Christos Davatzikos. "Classification of brain tumor type and grade using MRI texture and shape in a machine learning scheme." *Magnetic resonance in medicine* 62, no. 6 (2009): 1609-1618.

35. Suk, Heung-Il, and Dinggang Shen. "Deep ensemble sparse regression network for Alzheimer's disease diagnosis." In *International Workshop on Machine Learning in Medical Imaging*, pp. 113-121. Springer, Cham, 2016.

36. Anthimopoulos, Marios, Stergios Christodoulidis, Lukas Ebner, Andreas Christe, and Stavroula Mougiakakou. "Lung pattern classification for interstitial lung diseases using a deep convolutional neural network." *IEEE transactions on medical imaging* 35, no. 5 (2016): 1207-1216.

37. Shen, Wei, Mu Zhou, Feng Yang, Caiyun Yang, and Jie Tian. "Multi-scale convolutional neural networks for lung nodule classification." In *International Conference on Information Processing in Medical Imaging*, pp. 588-599. Springer, Cham, 2015.

38. Lakhani, Paras, and Baskaran Sundaram. "Deep learning at chest radiography: automated classification of pulmonary tuberculosis by using convolutional neural networks." *Radiology* 284, no. 2 (2017): 574-582.

39. Zhang, Zizhao, Yuanpu Xie, Fuyong Xing, Mason McGough, and Lin Yang. "Mdnet: A semantically and visually interpretable medical image diagnosis network." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6428-6436. 2017.

40. Sirinukunwattana, Korsuk, Shan E. Ahmed Raza, Yee-Wah Tsang, David RJ Snead, Ian A. Cree, and Nasir M. Rajpoot. "Locality sensitive deep learning for detection and

classification of nuclei in routine colon cancer histology images." *IEEE transactions on medical imaging* 35, no. 5 (2016): 1196-1206.

41. Habibzadeh, Mehdi, Adam Krzyżak, and Thomas Fevens. "White blood cell differential counts using convolutional neural networks for low resolution images." *International Conference on Artificial Intelligence and Soft Computing*. Springer, Berlin, Heidelberg, 2013.

42. Drozdzal, Michal, et al. "The importance of skip connections in biomedical image segmentation." *International Workshop on Large-Scale Annotation of Biomedical Data and Expert Label Synthesis*. Springer International Publishing, 2016.

43. Chen, Hao, et al. "Dcan: Deep contour-aware networks for accurate gland segmentation." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016.

44. McKinley, Richard, et al. "Nabla-net: A Deep Dag-Like Convolutional Architecture for Biomedical Image Segmentation." *International Workshop on Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*. Springer, Cham, 2016.

45. Çiçek, Özgün, et al. "3D U-Net: learning dense volumetric segmentation from sparse annotation." *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer International Publishing, 2016.

46. Q. Dou, L. Yu, H. Chen, Y. Jin, X. Yang, J. Qin, and P.-A. Heng, “3D deeply supervised network for automated segmentation of volumetric medical images,” Medical Image Analysis, vol. 41, pp. 40–54, 2017.

47. Li, Wenqi, et al. "On the Compactness, Efficiency, and Representation of 3D Convolutional Networks: Brain Parcellation as a Pretext Task." *International Conference on Information Processing in Medical Imaging*. Springer, Cham, 2017.

48. Kamnitsas, Konstantinos, et al. "Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation." *Medical image analysis* 36 (2017): 61-78.

49. Roth, Holger R., et al. "Deeporgan: Multi-level deep convolutional networks for automated pancreas segmentation." *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, Cham, 2015.

50. Chen, Hao, et al. "Voxresnet: Deep voxelwise residual networks for volumetric brain segmentation." *arXiv preprint arXiv:1608.05895* (2016).

51. Milletari, Fausto, Nassir Navab, and Seyed-Ahmad Ahmadi. "V-net: Fully convolutional neural networks for volumetric medical image segmentation." *3D Vision (3DV), 2016 Fourth International Conference on*. IEEE, 2016.

52. Liang, Ming, and Xiaolin Hu. "Recurrent convolutional neural network for object recognition." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015.

53. McCulloch, Warren S., and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity." *The bulletin of mathematical biophysics* 5.4 (1943): 115-133.

54. Rosenblatt, Frank. "The perceptron: A probabilistic model for information storage and organization in the brain." *Psychological review* 65.6 (1958): 386.

55. Minsky, Marvin, and Seymour Papert. "Perceptrons." (1969).

56. Ackley, David H., Geoffrey E. Hinton, and Terrence J. Sejnowski. "A learning algorithm for Boltzmann machines." *Cognitive science* 9.1 (1985): 147-169.

57. Hinton, Geoffrey E., Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." *Neural computation* 18.7 (2006): 1527-1554.

58. *Bengio, Yoshua; LeCun, Yann; Hinton, Geoffrey (2015). "Deep Learning". Nature.* ***521****: 436–444. doi:*10.1038/nature14539.

59. G.-B Huang, Q.Y. Zhu, C.-K Siew, "Extreme learning machine: theory and applications," Neurocomputing, 70, pp. 489–501, 2006.

60. G.-B Huang, D. H. Wang, and Y. Lan, "Extreme learning machines: a survey," Int. J. Mach Learn Clybern., 2(2), pp. 107–122, 2011.

61. G.-B Huang, L. Chen, C.-K Siew, “Universal approximation using incremental constructive feedforward networks with random hidden nodes,” IEEE Trans Neural Netw.,17(4), pp. 879–92,2006.

62. G.-B Huang, H. Zhou, X. Ding, R. Zhang, “Extreme Learning machine for regression and multiclass classification,” IEEE Trans. Syst. Man Cybern. Part B Cybern. 42(2), pp. 513–529, 2012.

63. G. Feng, G.-B. Huang, Q. Lin, and R. Gay, “Error minimized extreme learning machine with growth of hidden nodes and incremental learning,” IEEE Transactions on Neural Networks, vol. 20, no. 8, pp. 1352–1357, 2009.

64. W. Deng, Q. Zheng, and L. Chen, “Regularized extreme learning machine,” IEEE CIDM., pp. 389–395, 2009.

65. Zhang, Xingcheng, Zhizhong Li, Chen Change Loy, and Dahua Lin. "Polynet: A pursuit of structural diversity in very deep networks." In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3900-3908. IEEE, 2017.

66. Sabour, Sara, Nicholas Frosst, and Geoffrey E. Hinton. "Dynamic routing between capsules." *Advances in Neural Information Processing Systems*. 2017.

67. LeCun, Y., L. Bottou, and G. Orr. "Efficient BackProp in Neural Networks: Tricks of the Trade (Orr, G. and Müller, K., eds.)." *Lecture Notes in Computer Science* 1524.

68. Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." International conference on artificial intelligence and statistics. 2010.

69. He, Kaiming, et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." *Proceedings of the IEEE international conference on computer vision*. 2015

70. Mishkin, Dmytro, and Jiri Matas. "All you need is a good init." *arXiv preprint arXiv:1511.06422* (2015).

71. Lavin, Andrew. "Fast algorithms for convolutional neural networks." *arXiv preprint arXiv , ICLR 2016*

72. V. Nair and G. Hinton, Rectified linear units improve restricted boltzmann machines. Proceedings of the 27th International Conference on Machine Learning (ICML-10). 2010.

73. He, Kaiming, et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." *Proceedings of the IEEE international conference on computer vision*. 2015.

74. Clevert, Djork-Arné, Thomas Unterthiner, and Sepp Hochreiter. "Fast and accurate deep network learning by exponential linear units (elus)." *arXiv preprint arXiv:1511.07289* (2015).

75. Li, Yang, et al. "Improving Deep Neural Network with Multiple Parametric Exponential Linear Units." *arXiv preprint arXiv:1606.00305* (2016).

76. Jin, Xiaojie, et al. "Deep Learning with S-shaped Rectified Linear Activation Units." *arXiv preprint arXiv:1512.07030* (2015).

77. Xu, Bing, et al. "Empirical evaluation of rectified activations in convolutional network." *arXiv preprint arXiv:1505.00853* (2015)

78. He, Kaiming, et al. "Spatial pyramid pooling in deep convolutional networks for visual recognition." *European Conference on Computer Vision*. Springer, Cham, 2014.

79. Yoo, Donggeun, et al. "Multi-scale pyramid pooling for deep convolutional representation." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2015.

80. Graham, Benjamin. "Fractional max-pooling." *arXiv preprint arXiv:1412.6071* (2014).

81. Lee, Chen-Yu, Patrick W. Gallagher, and Zhuowen Tu. "Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree." *International Conference on Artificial Intelligence and Statistics*. 2016.

82. Hinton, Geoffrey E., et al. "Improving neural networks by preventing co-adaptation of feature detectors." *arXiv preprint arXiv:1207.0580* (2012).

83. Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research* 15.1 (2014): 1929-1958.

84. Wan, Li, et al. "Regularization of neural networks using dropconnect." *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*. 2013.

85. Bulò, Samuel Rota, Lorenzo Porzi, and Peter Kontschieder. "Dropout distillation." *Proceedings of The 33rd International Conference on Machine Learning*. 2016.

86. Ruder, Sebastian. "An overview of gradient descent optimization algorithms." *arXiv preprint arXiv:1609.04747* (2016).

87. Ngiam, Jiquan, et al. "On optimization methods for deep learning." Proceedings of the 28th International Conference on Machine Learning (ICML-11). 2011.

88. Koushik, Jayanth, and Hiroaki Hayashi. "Improving Stochastic Gradient Descent with Feedback." *arXiv preprint arXiv:1611.01505* (2016). (ICLR-2017)

89. G. Zhao, Z. Shen, and Z. Man, "Robust input weight selection for well-conditioned extreme learning machine," International Journal of Information Technology, vol. 17, no. 1, 2011.

90. P. L. Bartlett, "The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network," Information Theory, IEEE Transactions on, vol. 44, no. 2, pp. 525–536, 1998.

91. H. T. Huynh and Y. Won, "Weighted least squares scheme for reducing effects of outliers in regression based on extreme learning machine," J. Digital Content Technol. Appl.(JDCTA), vol. 2, no. 3, pp. 40–46, 2008.

92. A. L. B. Barros and G. A. Barreto, "Building a robust extreme learning machine for classification in the presence of outliers," in Hybrid Artificial Intelligent Systems, ser. Lecture Notes in Computer Science, J.-S. Pan, M. Polycarpou, M. Woniak, A. C. Carvalho,

H. Quintin, and E. Corchado, Eds. Springer Berlin Heidelberg, vol. 8073, pp. 588–597, 2013.

93. P. Horata, S. Chiewchanwattana, and K. Sunat, “Robust extreme learning machine,” Neurocomputing, vol. 102, pp. 31–34, 2013.

94. W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, “Face recognition: A literature survey,” ACM Computing Survey, 34(4), pp. 399–485, 2003.

95. T. Ahonen, A. Hadid, and M. Pietikainen, “Face description with local binary patterns: Application to face recognition,” IEEE TPAMI, 28(12), 2006.

96. Anna Bosch, Andrew Zisserman and Xavier Munoz, “Representing shape with a spatial pyramid kernel,” International Conference on Image and Video Retrieval, pp. 401-408, 2007.

97. L. Shen, and L. Bai, “A review on Gabor wavelets for face recognition,” Pattern Analysis and Application, 9, pp. 273–292, 2006.

98. Yale face database, < http://vision.ucsd.edu/content/yale-face-database>.

99. X. Liu, T. Chen, and B. V. K. Vijaya Kumar, “Face authentication for multiple subjects using eigenflow,” Pattern. Recog. 36(2), pp. 313–328, 2003.

100. F. S. Samaria, A. C. Harter, “Parameterisation of a stochastic model for human face identification,” Proceedings of the Second IEEE Workshop on Applications of Computer Vision, pp. 138-142, 1994.

101. Dalal N, Triggs B (2005) Histograms of oriented gradients for human detection. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE, vol 1, pp 886–893

102. Hettich S, Bay SD (1999) The uci kdd archive. http://kdd.ics.uci.edu

103. Jolliffe I (2002) Principal component analysis. Wiley Online Library

104. D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, “Deep big simple neural nets excel on handwritten digit recognition,” Neural Computation, vol. 22, no. 12, pp. 3207-3220, Dec. 2010.

105. U. Meier, D. Ciresan, L. Gambardella, and J. Schmidhuber, “Better digit recognition with a committee of simple neural nets,”, 2011 International Conference on in Document Analysis and Recognition (ICDAR), pp. 1250-254, Sept. 2011.

106. W. Song, S. Uchida, and M. Liwicki, “Comparative study of part-based handwritten character recognition methods,” in Document Analysis and Recognition (ICDAR), 2011 International Conference on, pp. 814-818, Sept. 2011.

107. B. B. Chaudhuri, U. Pal, “A complete printed Bangla OCR system,” Pattern Recognition, vol. 31, pp. 531–549, 1998.

108. U. Pal, “On the development of an optical character recognition (OCR) system for printed Bangla script,” Ph.D. Thesis, 1997.

109. U. Pal, B.B. Chaudhuri, “Indian script character recognition: a survey,” Pattern Recognition ,vol. 37, pp. 1887–1899, 2004.

110. U. Pal, B. B. Chaudhuri, “Automatic recognition of unconstrained offline Bangla hand-written numerals,” T. Tan, Y. Shi, W. Gao (Eds.), Advances in Multimodal Interfaces, Lecture Notes in Computer Science, vol. 1948, Springer, Berlin, pp. 371–378, 2000.

111. K. Roy, S. Vajda, U. Pal, B.B. Chaudhuri, “A system towards Indian postal automation,” Proceedings of the Ninth International Workshop on Frontiers in Handwritten Recognition (IWFHR-9), pp. 580–585, October 2004.

112. U. Pal, A. Belad, Ch. Choisy, “Touching numeral segmentation using water reservoir concept,” Pattern Recognition Lett. vol. 24, pp. 261-272, 2003.

113. C.-L. Liu and C. Y. Suen, “A new benchmark on the recognition of handwritten bangla and farsi numeral characters,” Pattern Recognition, vol. 42, no. 12, pp. 3287–3295, 2009.

114. B. B. Chaudhuri, “A complete handwritten numeral database of bangla – a major indic script,” Tenth International Workshop on Frontiers in Handwriting Recognition, Oct. 2006.

115. O. Surinta, L. Schomaker, and M. Wiering, “A comparison of feature and pixel-based methods for recognizing handwritten bangla digits,” IEEE 12th International Conference on Document Analysis and Recognition (ICDAR), pp. 165–169, Aug. 2013.

116. J.-W. Xu, J. Xu, and Y. Lu, “Handwritten Bangla digit recognition using hierarchical Bayesian network,” IEEE 3rd International Conference on Intelligent System and Knowledge Engineering, vol. 1, pp. 1096–1099, Nov. 2008.

117. Volume 4, Issue 6, June (2016).

118. Kiran Banjare, Sampada Massey "Handwritten Numeric Digit Classification and Recognition: Recent Advancements"International Journal of Emerging Technologies in Engineering Research (IJETER)

119. N. Das 1, B. Das, R. Sarkar, S. Basu, M. Kundu, M. Nasipuri, “Handwritten Bangla Basic and Compound character recognition using MLP and SVM classifier,” Journal of Computing, vol. 2, Feb. 2010.

120. S. Basu, N. Das, R. Sarkar, M. Kundu, M. Nasipuri, D. K. Basu “An MLP based Approach for Recognition of Handwritten ‘Bangla’ Numerals,” Proc. 2nd Indian International Conference on Artificial Intelligence, pp. 407-417, Dec. 2005.

121. N. Das, R. Sarkar, S. Basu, M. Kundu, M. Nasipuri, and D.K. Basu, “A genetic algorithm based region sampling for selection of local fatures in handwritten digit recognition application,” Applied Soft Computing, vol. 12, no. 5, pp. 1592-1606, 2012.

122. Md. M. Rahman, M. A. H. Akhand, S. Islam, P. C. Shill, “Bangla Handwritten Character Recognition using Convolutional Neural Network,” *I. J. Image, Graphics and Signal Processing*, vol. 8, pp. 42-49, July 15, 2015.

123. G. E. Hinton, P. Dayan, B. J. Frey, and R. Neal, The wake-sleep algorithm for self-organizing neural networks. Science, 268, pp. 1158-1161, 1995.

124. P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, “Extracting and composing robust features with denoising autoencoders,” Proceedings of the Twenty-fifth International Conference on Machine Learning, pp. 1096–1103, 2008.

125. K. Fukushima, “A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,” Biological Cybernetics, vol. 36, no. 4, pp.193–202, 1980.

126. Dan Cireşan and J. Schmidhuber, “Multi-column Deep Neural Networks for Offline Handwritten Chinese Character classification,” IDSIA Technical Report, No. IDSIA-05-13, 2013.

127. In-Jung Kim and Xiaohui Xie “Handwritten Hangul recognition using deep convolutional neural networks” Accepted and published in International Journal on Document Analysis and Recognition (IJDAR) vol 18, issue 1, pages 1-15, March 2015.

128. T. K. Bhowmik, P. Ghanty, A. Roy and S. K. Parui, *SVM-based hierarchical architectures for handwritten Bangla character recognition*, International Journal on Document Analysis and Recognition, vol. 12, no. 2, pp. 97-108, 2009.

129. S. Basu, N. Das, R. Sarkar, M. Kundu, M. Nasipuri and D. K. Basu, *A hierarchicalapproach to recognition of handwritten Bangla characters*, Pattern Recognition, vol. 42, pp. 1467–1484, 2009.

130. U. Bhattacharya, M. Shridhar, S. K. Parui, P. K. Sen and B. B. Chaudhuri, *Offline recognition of handwritten Bangla characters: an efficient two-stage approach*, Pattern Analysis and Applications, vol. 15, no. 4, pp. 445-458, 2012.

131. Das, Nibaran, Reddy, Jagan Mohan, Sarkar, Ram, Basu, Subhadip, Kundu, Mahantapas, Nasipuri, Mita, and Basu, Dipak Kumar. A statistical topological feature combination for recognition of handwritten numerals. Applied Soft Computing, 12(8):2486 – 2495, 2012a. ISSN 1568-4946.

132. Khan, Hassan, Al Helal, Abdullah, Ahmed, Khawza, et al. Handwritten bangla digit recognition using sparse representation classifier. In 2014 International Conference on Informatics, Electronics & Vision (ICIEV),, pp. 1–6, 2014.4.3

133. Alom, Md Zahangir, et al. “Handwritten Bangla Digit Recognition Using Deep Learning,” *arXiv preprint arXiv:1705.02680* (2017).

134. Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." *Science*, 313.5786 (2006): 504-507.

135. Collobert, Ronan, and Jason Weston. "A unified architecture for natural language processing: Deep neural networks with multitask learning." *Proceedings of the 25th international conference on Machine learning*. ACM, 2008.

136. Manning, Christopher D., et al. "The Stanford CoreNLP natural language processing toolkit." *ACL (System Demonstrations)*. 2014.

137. Hinton, Geoffrey, et al. "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups." *IEEE Signal Processing Magazine* 29.6 (2012): 82-97.

138. Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." *arXiv preprint arXiv: 1312.5602* (2013).

139. Lillicrap, Timothy P., et al. "Continuous control with deep reinforcement learning." *arXiv preprint arXiv: 1509.02971* (2015).

140. Riesenhuber, M., & Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nature neuroscience*, *2*(11), 1019-1025.

141. T.S. Lee, and D. Mumford. "Hierarchical Bayesian inference in the visual cortex." Journal of the Optical Society of America (*JOSA) A* 20.7 (2003): 1434-1448.

142. Lee, H., Grosse, R., Ranganath, R., & Ng, A. Y. (2009, June). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations.

In *Proceedings of the 26th annual international conference on machine learning* (pp. 609-616). ACM.

143. Deco, G., and T.S. Lee (2004). The role of early visual cortex in visual integration: a neural model of recurrent interaction. *European Journal of Neuroscience, 20*(4), 1089-1100.

144. DiCarlo, James J., Davide Zoccolan, and Nicole C. Rust. "How does the brain solve visual object recognition?." *Neuron* 73.3 (2012): 415-434.

145. McCulloch, Warren S., and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity." *The bulletin of mathematical biophysics* 5.4 (1943): 115-133.

146. B. Fernandez, A. G. Parlos, and W. Tsai. Nonlinear dynamic system identification using artificial neural networks (anns). In International Joint Conference on Neural Networks (IJCNN), pages 133–141, 1990.

147. D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, 1986.

148. Szegedy, Christian, et al. "Rethinking the inception architecture for computer vision." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016.

149. Iandola, Forrest N., et al. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size." *arXiv preprint arXiv: 1602.07360* (2016).

150. Urban, Gregor, et al. "Do Deep Convolutional Nets Really Need to be Deep and Convolutional?." *arXiv preprint arXiv:1603.05691* (2016).

151. Clevert, Djork-Arné, Thomas Unterthiner, and Sepp Hochreiter. "Fast and accurate deep network learning by exponential linear units (elus)." *arXiv preprint arXiv: 1511.07289* (2015).

152. Koushik, Jayanth, and Hiroaki Hayashi. "Improving Stochastic Gradient Descent with Feedback." *arXiv preprint arXiv: 1611.01505* (2016).

153. Ngiam, Jiquan, et al. "On optimization methods for deep learning." *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. 2011.

154. O'Reilly, R. C., Wyatte, D., Herd, S., Mingus, B., & Jilk, D. J. (2013). Recurrent processing during object recognition. *Frontiers in psychology*, *4*, 124.

155. Wyatte, D., Curran, T., & O'Reilly, R. (2012). The limits of feedforward vision: Recurrent processing promotes robust object recognition when objects are degraded. *Journal of Cognitive Neuroscience*, *24*(11), 2248-2261

156. Liao, Q., & Poggio, T. (2016). Bridging the gaps between residual learning, recurrent neural networks and visual cortex. *arXiv preprint arXiv:1604.03640*.

157. Pinheiro, Pedro HO, and Ronan Collobert. "Recurrent Convolutional Neural Networks for Scene Labeling." *ICML*. 2014.

158. Donahue, Jeffrey, et al. "Long-term recurrent convolutional networks for visual recognition and description." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

159. Bishop, Christopher M. "Pattern recognition." *Machine Learning* 128 (2006): 1-58.

160. Kussul, Ernst, and Tatiana Baidyk. "Improved method of handwritten digit recognition tested on MNIST database." *Image and Vision Computing* 22.12 (2004): 971-981.

161. Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical report, 2009.

162. Netzer, Yuval, et al. "Reading digits in natural images with unsupervised feature learning." (2011).

163. Sutskever, Ilya, et al. "On the importance of initialization and momentum in deep learning." *ICML (3)* 28 (2013): 1139-1147.

164. Goodfellow, Ian J., et al. "Maxout Networks." *ICML (3)* 28 (2013): 1319-1327.

165. Wan, Li, et al. "Regularization of neural networks using drop-connect." *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*. 2013.

166. Stollenga, Marijn F., et al. "Deep networks with internal selective attention through feedback connections." *Advances in Neural Information Processing Systems*. 2014.

167. Lee, Chen-Yu, et al. "Deeply-Supervised Nets." *AISTATS*. Vol. 2. No. 3. 2015.

168. Srivastava, Rupesh K., Klaus Greff, and Jürgen Schmidhuber. "Training very deep networks." *Advances in neural information processing systems*. 2015.

169. J. T. Springenberg and M. Riedmiller. Improving deep neural networks with probabilistic maxout units. In International Conference on Learning Representations (ICLR), 2014

170. Romero, Adriana, et al. "Fitnets: Hints for thin deep nets." *arXiv preprint arXiv: 1412.6550* (2014).

171. Srivastava, Nitish, and Ruslan R. Salakhutdinov. "Discriminative transfer learning with tree-based priors." *Advances in Neural Information Processing Systems*. 2013.

172. Chollet, F. Keras. https://github.com/fchollet/keras, 2016.

173. F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio. Theano: new features and speed improvements. NIPS Workshop on Deep Learning and Unsupervised Feature Learning, 2012

174. https://tiny-imagenet.herokuapp.com

175. Long, J., Shelhamer, E., and Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3431-3440 (2015).

176. Toshev, A., and Szegedy, C., Deeppose: Human pose estimation via deep neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1653-1660 (2014).

177. Dong, C., Loy, C. C., He, K., and Tang, X. , Learning a deep convolutional network for image super-resolution. In European Conference on Computer Vision . Springer International Publishing, pp. 184-199, (2014).

178. Shankar, S., Garg, V. K., and Cipolla, R. (2015). Deep-carving: Discovering visual attributes by carving deep neural nets. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3403-3412, (2015).

179. Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., and Oliva, A., Learning deep features for scene recognition using places database. In Advances in neural information processing systems, pp. 487-495, (2014).

180. Wang, Naiyan, et al. ”Transferring rich feature hierarchies for robust visual tracking.” arXiv preprint arXiv: 1501.04587 (2015).

181. Mao, Junhua, et al. ”Deep captioning with multimodal recurrent neural networks (mrnn).” arXiv preprint arXiv: 1412.6632 (2014).

182. Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R.,and Fei-Fei, L. . Largescale video classification with convolutional neural networks. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pp. 1725-1732, (2014).

183. Ballas, Nicolas, et al. ”Delving deeper into convolutional networks for learning video representations.” arXiv preprint arXiv: 1511.06432 (2015).

184. Collobert, Ronan, and Jason Weston. ”A unified architecture for natural language processing: Deep neural networks with multitask learning.” Proceedings of the 25th international conference on Machine learning. ACM, (2008).

185. Hinton, Geoffrey, et al. ”Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups.” IEEE Signal Processing Magazine 29.6, pp. 82-97, (2012).

186. Mnih, Volodymyr, et al., Playing atari with deep reinforcement learning., arXiv preprint arXiv: 1312.5602 (2013).

187. DiCarlo, James J., Davide Zoccolan, and Nicole C. Rust., How does the brain solve visual object recognition?, Neuron 73.3, pp. 415-434, (2012).

188. McCulloch, Warren S., and Walter Pitts., A logical calculus of the ideas immanent in nervous activity., The bulletin of mathematical biophysics 5.4, pp. 115-133, (1943).

189. J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In CVPR, (2015).

190. Fernandez, Benito, A. G. Parlos, and W. K. Tsai. ”Nonlinear dynamic system identification using artificial neural networks (ANNs).” Neural Networks, 1990., 1990 IJCNN International Joint Conference on. IEEE, 1990.

191. 32. Xie, S., Girshick, R., Dollr, P., Tu, Z., and He, K., Aggregated residual transformations for deep neural networks. arXiv preprint arXiv:1611.05431., (2016)

192. Liao, Q., and Poggio, T. , Bridging the gaps between residual learning, recurrent neural networks and visual cortex. arXiv preprint arXiv:1604.03640,(2016).

193. https://tiny-imagenet.herokuapp.com/ (2017)

194. O’Reilly, R.C., Wyatte, D., Herd, S., Mingus, B., and Jilk, D., Recurrent processing during object recognition. Frontiers in Psychology, 4(124), 1-14. (2013)

195. Md Zahangir Alom, Mahmudul Hasan, Chris Yakopcic, and Tarek M. Taha, Inception Recurrent Convolutional Neural Network for Object Recognition, https://arxiv.org/abs/1704.07709, (2017).

196. https://github.com/fchollet/deep-learning-models/releases/download/v0.2/inceptionv3-weights-tf-dim-ordering-tf-kernels.h5 (2017)

197. Alom, Md Zahangir, et al. "Improved Inception-Residual Convolutional Neural Network for Object Recognition." *arXiv preprint arXiv:1712.09888* (2017).

198. Badrinarayanan, Vijay, Alex Kendall, and Roberto Cipolla. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation." *arXiv preprint arXiv:1511.00561*(2015).

199. Zhang, Zhengxin, Qingjie Liu, and Yunhong Wang. "Road Extraction by Deep Residual U-Net." *arXiv preprint arXiv:1711.10684* (2017).

200. Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." *International Conference on Medical image computing and computer-assisted intervention*. Springer, Cham, 2015.

201. Çiçek, Özgün, et al. "3D U-Net: learning dense volumetric segmentation from sparse annotation." *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer International Publishing, 2016.

202. Milletari, Fausto, Nassir Navab, and Seyed-Ahmad Ahmadi. "V-net: Fully convolutional neural networks for volumetric medical image segmentation." *3D Vision (3DV), 2016 Fourth International Conference on*. IEEE, 2016.

203. Yang, Dong, et al. "Automated anatomical landmark detection ondistal femur surface using convolutional neural network." *Biomedical Imaging (ISBI), 2015 IEEE 12th International Symposium on*. IEEE, 2015.

204. Cai, Yunliang, et al. "Multi-modal vertebrae recognition using transformed deep convolution network." *Computerized Medical Imaging and Graphics* 51 (2016): 11-19.

205. Ramesh, N., J-H. Yoo, and I. K. Sethi. "Thresholding based on histogram approximation." *IEE Proceedings-Vision, Image and Signal Processing* 142.5 (1995): 271-279.

206. Sharma, Neeraj, and Amit Kumar Ray. "Computer aided segmentation of medical images based on hybridized approach of edge and region based techniques." *Proceedings of International Conference on Mathematical Biology', Mathematical Biology Recent Trends by Anamaya Publishers*. 2006.

207. Boykov, Yuri Y., and M-P. Jolly. "Interactive graph cuts for optimal boundary & region segmentation of objects in ND images." *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*. Vol. 1. IEEE, 2001.

208. Litjens, Geert, et al. "A survey on deep learning in medical image analysis." *arXiv preprint arXiv:1702.05747* (2017).

209. Havaei, Mohammad, et al. "Brain tumor segmentation with deep neural networks." *Medical image analysis* 35 (2017): 18-31.

210. S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr, "Conditional random fields as recurrent neural networks," in Proceedings of the IEEE International Conference on Computer Vision, pp. 1529–1537, 2015.

211. L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected CRFs," in ICLR, 2015.

212. Kendall, Alex, Vijay Badrinarayanan, and Roberto Cipolla. "Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding." *arXiv preprint arXiv:1511.02680* (2015).

213. Li, Ruirui, et al. "DeepUNet: A Deep Fully Convolutional Network for Pixel-level Sea-Land Segmentation." *arXiv preprint arXiv:1709.00201* (2017).

214. Kayalibay, Baris, Grady Jensen, and Patrick van der Smagt. "CNN-based Segmentation of Medical Imaging Data." *arXiv preprint arXiv:1701.03056* (2017).

215. Drozdzal, Michal, et al. "The importance of skip connections in biomedical image segmentation." *International Workshop on Large-Scale Annotation of Biomedical Data and Expert Label Synthesis*. Springer International Publishing, 2016.

216. Hoover, A. D., Valentina Kouznetsova, and Michael Goldbaum. "Locating blood vessels in retinal images by piecewise threshold probing of a matched filter response." *IEEE Transactions on Medical imaging* 19.3 (2000): 203-210.

217. Fraz, Muhammad Moazam, et al. "Blood vessel segmentation methodologies in retinal images–a survey." *Computer methods and programs in biomedicine* 108.1 (2012): 407-433.

218. Zhao, Yitian, et al. "Automated vessel segmentation using infinite perimeter active contour model with hybrid region information with application to retinal images." *IEEE transactions on medical imaging* 34.9 (2015): 1797-1807.

219. Cheng, Erkang, Liang Du, Yi Wu, Ying J. Zhu, Vasileios Megalooikonomou, and Haibin Ling. "Discriminative vessel segmentation in retinal images by fusing context-aware hybrid features." *Machine vision and applications* 25, no. 7 (2014): 1779-1792.

220. Soares, João VB, et al. "Retinal vessel segmentation using the 2-D Gabor wavelet and supervised classification." *IEEE Transactions on medical Imaging* 25.9 (2006): 1214-1222.

221. Gutman, David, et al. "Skin lesion analysis toward melanoma detection: A challenge at the international symposium on biomedical imaging (ISBI) 2016, hosted by the international skin imaging collaboration (ISIC)." *arXiv preprint arXiv:1605.01397* (2016).

222. https://www.kaggle.com/kmader/finding-lungs-in-ct-data/data.

223. Dice, Lee R. "Measures of the amount of ecologic association between species." *Ecology* 26.3 (1945): 297-302.

224. Jaccard, Paul. "The distribution of the flora in the alpine zone." *New phytologist* 11.2 (1912): 37-50.

225. Zhao, Yitian, et al. "Automatic 2-D/3-D Vessel Enhancement in Multiple Modality Images Using a Weighted Symmetry Filter." *IEEE transactions on medical imaging* 37.2 (2018): 438-450.

226. Li, Qiaoliang, et al. "A cross-modality learning approach for vessel segmentation in retinal images." *IEEE transactions on medical imaging* 35.1 (2016): 109-118.

227. Azzopardi, George, et al. "Trainable COSFIRE filters for vessel delineation with application to retinal images." *Medical image analysis* 19.1 (2015): 46-57.

228. Roychowdhury, Sohini, Dara D. Koozekanani, and Keshab K. Parhi. Blood vessel segmentation of fundus images by major vessel extraction and subimage

classification." *IEEE journal of biomedical and health informatics* 19.3 (2015): 1118-1128.

229. Liskowski, Paweł, and Krzysztof Krawiec. "Segmenting Retinal Blood Vessels With Deep Neural Networks." *IEEE transactions on medical imaging* 35.11 (2016): 2369-2380.

230. Marín, Diego, et al. "A new supervised method for blood vessel segmentation in retinal images by using gray- level and moment invariants-based features." *IEEE Transactions on medical imaging* 30.1 (2011): 146-158.

231. Fraz, Muhammad Moazam, et al. "An ensemble classification-based approach applied to retinal blood vessel segmentation." *IEEE Transactions on Biomedical Engineering* 59.9 (2012): 2538-2548.

232. Fraz, Muhammad Moazam, et al. "Delineation of blood vessels in pediatric retinal images using decision trees-based ensemble classification." *International journal of computer assisted radiology and surgery* 9.5 (2014): 795-811.

233. Yu, L., Chen, H., Dou, Q., Qin, J., & Heng, P. A. (2017). Automated melanoma recognition in dermoscopy images via very deep residual networks. *IEEE transactions on medical imaging*, *36*(4), 994-1004.

234. Burdick, Jack, et al. "Rethinking Skin Lesion Segmentation in a Convolutional Classifier." *Journal of digital imaging* (2017): 1-6.

235. Hsu, Roy Chaoming, et al. "Contour extraction in medical images using initial boundary pixel selection and segmental contour following." *Multidimensional Systems and Signal Processing* 23.4 (2012): 469-498.

236. Litjens, Geert, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen AWM van der Laak, Bram van Ginneken, and Clara I. Sánchez. "A survey on deep learning in medical image analysis." *Medical image analysis* 42 (2017): 60-88.

237. Habibzadeh, Mehdi, Adam Krzyżak, and Thomas Fevens. "Application of pattern recognition techniques for the analysis of thin blood smear images." *Journal of Medical Informatics & Technologies* 18 (2011).

238. Habibzadeh, Mehdi, Adam Krzyżak, and Thomas Fevens. "White blood cell differential counts using convolutional neural networks for low resolution images." *International Conference on Artificial Intelligence and Soft Computing*. Springer, Berlin, Heidelberg, 2013.

239. Anglin C. Sickle Cell Disease. Journal of Consumer Health on the Internet. 2015;19(2):122–131.

240. Xu, Mengjia, Dimitrios P. Papageorgiou, Sabia Z. Abidi, Ming Dao, Hong Zhao, and George Em Karniadakis. "A deep convolutional neural network for classification of red blood cells in sickle cell anemia." *PLoS Computational Biology* 13, no. 10 (2017): e1005746.

241. Ushizima, Daniela Mayumi, Ana Carolina Lorena, and A. C. P. L. F. De Carvalho. "Support vector machines applied to white blood cell recognition." *Hybrid Intelligent Systems, 2005. HIS'05. Fifth International Conference on*. IEEE, 2005.

242. Shitong, Wang, and Wang Min. "A new detection algorithm (NDA) based on fuzzy cellular neural networks for white blood cell detection." *IEEE Transactions on information technology in biomedicine* 10.1 (2006): 5-10.

243. Theera-Umpon, Nipon, and Sompong Dhompongsa. "Morphological granulometric features of nucleus in automatic bone marrow white blood cell classification." *IEEE Transactions on Information Technology in Biomedicine* 11.3 (2007): 353-359.

244. Dorini, Leyza Baldo, Rodrigo Minetto, and Neucimar Jeronimo Leite. "Semiautomatic white blood cell segmentation based on multiscale analysis." *IEEE journal of biomedical and health informatics* 17.1 (2013): 250-256.

245. Su, Mu-Chun, Chun-Yen Cheng, and Pa-Chun Wang. "A neural-network-based approach to white blood cell classification." *The scientific world journal* 2014 (2014).

246. Habibzadeh, Mehdi, Adam Krzyżak, and Thomas Fevens. "White blood cell differential counts using convolutional neural networks for low resolution images." *International Conference on Artificial Intelligence and Soft Computing*. Springer, Berlin, Heidelberg, 2013.

247. Gao, Zhimin, Jianjia Zhang, Luping Zhou, and Lei Wang. "Hep-2 cell image classification with convolutional neural networks." In *Pattern Recognition Techniques for Indirect Immunofluorescence Images (I3A), 2014 1st Workshop on*, pp. 24-28. Ieee, 2014.

248. Li, Hongwei, Jianguo Zhang, and Wei-Shi Zheng. "Deep CNNs for HEp-2 cells classification: A cross-specimen analysis." *arXiv preprint arXiv:1604.05816* (2016).

249. https://blog.athelas.com/classifying-white-blood-cells-with-convolutional-neural-networks-2ca6da239331

250. Xu, M., Papageorgiou, D. P., Abidi, S. Z., Dao, M., Zhao, H., & Karniadakis, G. E. (2017). A deep convolutional neural network for classification of red blood cells in sickle cell anemia. *PLoS Computational Biology*, *13*(10), e1005746.

251. Razzak, Muhammad Imran, and Saeeda Naz. "Microscopic Blood Smear Segmentation and Classification Using Deep Contour Aware CNN and Extreme Machine Learning." *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*. IEEE, 2017.

252. Alom, M. Z., Hasan, M., Yakopcic, C., Taha, T. M., & Asari, V. K. (2018). Recurrent Residual Convolutional Neural Network based on U-Net (R2U-Net) for Medical Image Segmentation. *arXiv preprint arXiv:1802.06955*.

253. https://github.com/dhruvp/wbc-classification

254. https://github.com/devanshtandon/RBC-Classification-ConvNet

255. Loukas, C., Spiros Kostopoulos, Anna Tanoglidi, Dimitris Glotsos, C. Sfikas, and Dionisis Cavouras. "Breast cancer characterization based on image classification of tissue sections visualized under low magnification." *Computational and mathematical methods in medicine* 2013 (2013).

256. Elston, Christopher W., and Ian O. Ellis. "Pathological prognostic factors in breast cancer. I. The value of histological grade in breast cancer: experience from a large study with long-term follow-up." *Histopathology* 19.5 (1991): 403-410.

257. Gurcan, M. N., Boucheron, L. E., Can, A., Madabhushi, A., Rajpoot, N. M., & Yener, B. (2009). Histopathological image analysis: A review. *IEEE reviews in biomedical engineering*, *2*, 147-171.

258. McCann, Michael T., John A. Ozolek, Carlos A. Castro, Bahram Parvin, and Jelena Kovacevic. "Automated histology analysis: Opportunities for signal processing." *IEEE Signal Processing Magazine* 32, no. 1 (2015): 78-87.

259. Peikari, Mohammad, Mehrdad J. Gangeh, Judit Zubovits, Gina Clarke, and Anne L. Martel. "Triaging diagnostically relevant regions from pathology whole slides of breast cancer: A texture based approach." *IEEE transactions on medical imaging* 35, no. 1 (2016): 307-315.

260. Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., ... & Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. *Medical image analysis*, *42*, 60-88.

261. Spanhol, Fabio A., Luiz S. Oliveira, Caroline Petitjean, and Laurent Heutte. "A dataset for breast cancer histopathological image classification." *IEEE Transactions on Biomedical Engineering* 63, no. 7 (2016): 1455-1462.

262. Kowal, Marek, Paweł Filipczuk, Andrzej Obuchowicz, Józef Korbicz, and Roman Monczak. "Computer-aided diagnosis of breast cancer based on fine needle biopsy microscopic images." *Computers in biology and medicine* 43, no. 10 (2013): 1563-1572.

263. George, Yasmeen Mourice, Hala Helmy Zayed, Mohamed Ismail Roushdy, and Bassant Mohamed Elbagoury. "Remote computer-aided breast cancer detection and diagnosis system based on cytological images." *IEEE Systems Journal* 8, no. 3 (2014): 949-964.

264. Zhang, Yungang, Bailing Zhang, Frans Coenen, and Wenjin Lu. "Breast cancer diagnosis from biopsy images with highly reliable random subspace classifier ensembles." *Machine vision and applications* 24, no. 7 (2013): 1405-1420.

265. Veta, Mitko, Josien PW Pluim, Paul J. Van Diest, and Max A. Viergever. "Breast cancer histopathology image analysis: A review." *IEEE Transactions on Biomedical Engineering* 61, no. 5 (2014): 1400-1411.

266. Gupta, Vibha, and Arnav Bhavsar. "Breast Cancer Histopathological Image Classification: Is Magnification Important?." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 17-24. 2017.

267. Bayramoglu, Neslihan, Juho Kannala, and Janne Heikkilä. "Deep learning for magnification independent breast cancer histopathology image classification." In *Pattern Recognition (ICPR), 2016 23rd International Conference on*, pp. 2440-2445. IEEE, 2016.

268. Spanhol, Fabio Alexandre, Luiz S. Oliveira, Caroline Petitjean, and Laurent Heutte. "Breast cancer histopathological image classification using convolutional neural networks." In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pp. 2560-2567. IEEE, 2016.

269. Spanhol, Fabio A., Luiz S. Oliveira, Paulo R. Cavalin, Caroline Petitjean, and Laurent Heutte. "Deep features for breast cancer histopathological image classification." In *Systems, Man, and Cybernetics (SMC), 2017 IEEE International Conference on*, pp. 1868-1873. IEEE, 2017.

270. Araújo, Teresa, Guilherme Aresta, Eduardo Castro, José Rouco, Paulo Aguiar, Catarina Eloy, António Polónia, and Aurélio Campilho. "Classification of breast cancer histology images using Convolutional Neural Networks." *PloS one* 12, no. 6 (2017): e0177544.

271. Han, Zhongyi, Benzheng Wei, Yuanjie Zheng, Yilong Yin, Kejian Li, and Shuo Li. "Breast cancer multi-classification from histopathological images with structured deep learning model." *Scientific reports* 7, no. 1 (2017): 4172.

272. Pego A, Aguiar P, Bioimaging 2015: 2015, Available from: http://www.bioimaging2015.ineb.up.pt/dataset.html

273. Kahya, Mohammed Abdulrazaq, Waleed Al-Hayani, and Zakariya Yahya Algamal. "Classification of breast cancer histopathology images based on adaptive sparse support vector machine." *Journal of Applied Mathematics and Bioinformatics* 7.1 (2017): 49.

274. Cruz-Roa, Angel, et al. "Automatic detection of invasive ductal carcinoma in whole slide images with convolutional neural networks." *Medical Imaging 2014: Digital Pathology*. Vol. 9041. International Society for Optics and Photonics, 2014.

275. Abadi, Martín, et al. "Tensorflow: a system for large-scale machine learning." *OSDI*. Vol. 16. 2016.

276. Van Norman, Gail A. "Drugs, devices, and the FDA: part 2: an overview of approval processes: FDA approval of medical devices." *JACC: Basic to Translational Science* 1.4 (2016): 277-287.

277. Rojo, M. G., Punys, V., Slodkowska, J., Schrader, T., Daniel, C., & Blobel, B. (2009). Digital pathology in Europe: coordinating patient care and research efforts. *Studies in health technology and informatics*, *150*, 997-1001.

278. Rojo, M. G. (2012). State of the art and trends for digital pathology. *Stud. Health Technol. Inform*, *179*, 15-28.

279. Pantanowitz, L., Sinard, J. H., Henricks, W. H., Fatheree, L. A., Carter, A. B., Contis, L., ... & Parwani, A. V. (2013). Validating whole slide imaging for diagnostic purposes in pathology: guideline from the College of American Pathologists Pathology and Laboratory Quality Center. *Archives of Pathology and Laboratory Medicine*, *137*(12), 1710-1722.

280. López, Carlos, et al. "Digital image analysis in breast cancer: an example of an automated methodology and the effects of image compression." *Studies in health technology and informatics* 179 (2012): 155-171.

281. Bueno, G., García-Rojo, M., Déniz, O., Fernández-Carrobles, M. M., Vállez, N., Salido, J., & García-González, J. (2012). Emerging trends: grid technology in pathology. *Stud Health Technol Inform*, *179*, 218-29.

282. Gurcan, M. N., Boucheron, L., Can, A., Madabhushi, A., Rajpoot, N., & Yener, B. (2009). Histopathological image analysis: A review. *IEEE reviews in biomedical engineering*, *2*, 147.

283. Fuchs, T. J., & Buhmann, J. M. (2011). Computational pathology: Challenges and promises for tissue analysis. *Computerized Medical Imaging and Graphics*, *35*(7-8), 515-530.

284. Kothari, S., Phan, J. H., Stokes, T. H., & Wang, M. D. (2013). Pathology imaging informatics for quantitative analysis of whole

285. Dalle, Jean-Romain, et al. "Nuclear pleomorphism scoring by selective cell nuclei detection." *WACV*. 2009.

286. Malon, Christopher D., and Eric Cosatto. "Classification of mitotic figures with convolutional neural networks and seeded blob features." *Journal of pathology informatics* 4 (2013).

287. Yuan, Yinyin, et al. "Quantitative image analysis of cellular heterogeneity in breast tumors complements genomic profiling." *Science translational medicine* 4.157 (2012): 157ra143-157ra143.

288. Sharma, Harshita, et al. "A Multi-resolution Approach for Combining Visual Information using Nuclei Segmentation and Classification in Histopathological Images." *VISAPP (3)*. 2015.

289. Wang, Haibo, et al. "Cascaded ensemble of convolutional neural networks and handcrafted features for mitosis detection." *Medical Imaging 2014: Digital Pathology*. Vol. 9041. International Society for Optics and Photonics, 2014.

290. Cruz-Roa, Angel Alfonso, et al. "A deep learning architecture for image representation, visual interpretability and automated basal-cell carcinoma cancer detection." *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, Berlin, Heidelberg, 2013.

291. Singh, Malay, et al. "A study of nuclei classification methods in histopathological images." *International Conference on Innovation in Medicine and Healthcare*. Springer, Cham, 2017.

292. Wienert, Stephan, et al. "Detection and segmentation of cell nuclei in virtual microscopy images: a minimum-model approach." *Scientific reports* 2 (2012): 503.

293. Ciresan, Dan, et al. "Deep neural networks segment neuronal membranes in electron microscopy images." *Advances in neural information processing systems*. 2012.

294. Xing, Fuyong, Yuanpu Xie, and Lin Yang. "An automatic learning-based framework for robust nucleus segmentation." *IEEE transactions on medical imaging* 35.2 (2016): 550-566.

295. Kumar, Neeraj, et al. "A dataset and a technique for generalized nuclear segmentation for computational pathology." *IEEE transactions on medical imaging* 36.7 (2017): 1550-1560.

296. Ho, David Joon, et al. "Nuclei segmentation of fluorescence microscopy images using three dimensional convolutional neural networks." (2017).

297. Cui, Yuxin, et al. "A Deep Learning Algorithm for One-step Contour Aware Nuclei Segmentation of Histopathological Images." *arXiv preprint arXiv:1803.02786* (2018).

298. Ram, Sundaresh, et al. "Joint Cell Nuclei Detection and Segmentation in Microscopy Images Using 3D Convolutional Networks." *arXiv preprint arXiv:1805.02850* (2018).

299. Md Zahangir Alom, Chris Yakopcic, Tarek M. Taha , and Vijayan K. Asari, "Nuclei segmentation with Recurrent Residual Convolutional Neural Networks based U-Net (R2U-Net)," *2018 IEEE National Aerospace & Electronics Conference*, Dayton, Ohio, USA, 23 - 26 July 2018. (NAECON)

300. Arteta, Carlos, et al. "Learning to detect cells using non-overlapping extremal regions." *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, Berlin, Heidelberg, 2012.

301. Fiaschi, Luca, et al. "Learning to count with regression forest and structured labels." *Pattern Recognition (ICPR), 2012 21st International Conference on*. IEEE, 2012.

302. Dong, Bo, et al. "Deep learning for automatic cell detection in wide-field microscopy zebrafish images." *Biomedical Imaging (ISBI), 2015 IEEE 12th International Symposium on*. IEEE, 2015.

303. Xu, Jun, et al. "Stacked sparse autoencoder (SSAE) for nuclei detection on breast cancer histopathology images." *IEEE transactions on medical imaging* 35.1 (2016): 119-130.

304. Xie, Weidi, J. Alison Noble, and Andrew Zisserman. "Microscopy cell counting and detection with fully convolutional regression networks." *Computer methods in biomechanics and biomedical engineering: Imaging & Visualization* 6.3 (2018): 283-292.

305. Sirinukunwattana, Korsuk, et al. "Locality sensitive deep learning for detection and classification of nuclei in routine colon cancer histology images." *IEEE transactions on medical imaging* 35.5 (2016): 1196-1206.

306. Yuan, Yinyin, et al. "Quantitative image analysis of cellular heterogeneity in breast tumors complements genomic profiling." *Science translational medicine* 4.157 (2012): 157ra143-157ra143.

307. Sirinukunwattana, Korsuk, David RJ Snead, and Nasir M. Rajpoot. "A novel texture descriptor for detection of glandular structures in colon histology images." *Medical*

Imaging 2015: Digital Pathology. Vol. 9420. International Society for Optics and Photonics, 2015.

308. Xie, Yuanpu, et al. "Beyond classification: structured regression for robust cell detection using convolutional neural network." *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, Cham, 2015.

309. Xu, Jun, et al. "Stacked sparse autoencoder (SSAE) for nuclei detection on breast cancer histopathology images." *IEEE transactions on medical imaging* 35.1 (2016): 119-130.

310. Kuse, Manohar, et al. "Local isotropic phase symmetry measure for detection of beta cells and lymphocytes." *Journal of pathology informatics* 2 (2011).

311. https://www.kaggle.com/c/data-science-bowl-2018

312. Janowczyk, Andrew, and Anant Madabhushi. "Deep learning for digital pathology image analysis: A comprehensive tutorial with selected use cases." Journal of pathology informatics 7 (2016).

313. Naylor, Peter, et al. "Nuclei segmentation in histopathology images using deep neural networks." Biomedical Imaging (ISBI 2017), 2017 IEEE 14th International Symposium on. IEEE, 2017.

314. Sirinukunwattana, Korsuk, David RJ Snead, and Nasir M. Rajpoot. "A random polygons model of glandular structures in colon histology images." Biomedical Imaging (ISBI), 2015 IEEE 12th International Symposium on. IEEE, 2015.

315. Basavanhally, Ajay, et al. "Incorporating domain knowledge for tubule detection in breast histopathology using O'Callaghan neighborhoods." Medical Imaging 2011: Computer-Aided Diagnosis. Vol. 7963. International Society for Optics and Photonics, 2011.

316. Zhang, Chen, Peng Li, Guangyu Sun, Yijin Guan, Bingjun Xiao, and Jason Cong. "Optimizing fpga-based accelerator design for deep convolutional neural networks." In *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 161-170, 2015.

317. Ren, Ao, et al. "Designing Reconfigurable Large-Scale Deep Learning Systems Using Stochastic Computing." *2016 IEEE International Conference on Rebooting Computing. IEEE*. 2016.

318. LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." *Nature*521.7553: 436-444 in 2015.

319. Y. Marc'Aurelio Ranzato, L. Boureau, and Y. LeCun, ``Sparse feature learning for deep belief networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 20, pp. 1185-1192 in 2007.

320. Carvalho, Edigleison Francelino, and Paulo Martins Engel. "Convolutional Sparse Feature Descriptor for Object Recognition in CIFAR-10." Intelligent Systems (BRACIS), 2013 Brazilian Conference on. IEEE, 2013.

321. Lee, Honglak, et al. "Efficient sparse coding algorithms." Advances in neural information processing systems. 2006.

322. Cevher, Volkan, Stephen Becker, and Mark Schmidt. "Convex optimization for big data: Scalable, randomized, and parallel algorithms for big data analytics." *IEEE Signal Processing Magazine* 31.5: 32-43, in 2014.

323. Wohlberg, Brendt. "Efficient convolutional sparse coding." 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2014.

324. Szlam, Arthur, Koray Kavukcuoglu, and Yann LeCun. "Convolutional matching pursuit and dictionary training." *arXiv preprint arXiv:1010.0422* 4 in 2010.

325. Chen, Bo, et al. "Deep learning with hierarchical convolutional factor analysis." *IEEE transactions on pattern analysis and machine intelligence*35.8: 1887-1901 in 2013

326. Zhou, Yin, et al. "Classification of histology sections via multispectral convolutional sparse coding." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014.

327. Bristow, Hilton, Anders Eriksson, and Simon Lucey. "Fast convolutional sparse coding." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013.

328. M. Zeiler, D. Krishnan, G. Taylor, and R. Gergus. “ Deconvolution network” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2010.

329. Merolla, Paul A., et al. "A million spiking-neuron integrated circuit with a scalable communication network and interface." *Science* 345.6197: 668-673 in 2014.

330. Cassidy, Andrew S., et al. "Cognitive computing building block: A versatile and efficient digital neuron model for neurosynaptic cores." *Neural Networks (IJCNN), the 2013 International Joint Conference on*. IEEE, 2013.

331. Merolla, Paul, et al. "A digital neurosynaptic core using embedded crossbar memory with 45pJ per spike in 45nm." *Custom Integrated Circuits Conference (CICC), 2011 IEEE*. IEEE, 2011.

332. Esser, Steven K., et al. "Convolutional Networks for Fast, Energy-Efficient Neuromorphic Computing." *arXiv preprint arXiv:1603.08270* (2016).

333. Gerstner, Wulfram, and Werner M. Kistler. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

334. Esser, Steven K., et al. "Cognitive computing systems: Algorithms and applications for networks of neurosynaptic cores." *Neural Networks (IJCNN), the 2013 International Joint Conference on*. IEEE, 2013.

335. Amir, Arnon, et al. "Cognitive computing programming paradigm: a corelet language for composing networks of neurosynaptic cores." *Neural Networks (IJCNN), the 2013 International Joint Conference on*. IEEE, 2013.

336. Diehl, Peter U., et al. "Truehappiness: Neuromorphic emotion recognition on truenorth." *arXiv preprint arXiv:1601.04183* (2016).

337. Alom, Md Zahangir, and Tarek M. Taha. "Network intrusion detection for cyber security on neuromorphic computing system." *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, 2017.

338. Alom, Md Zahangir, et al. "Quadratic Unconstrained Binary Optimization (QUBO) on neuromorphic computing system." *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, 2017.

339. Alom, Md Zahangir, et al. "Convolutional sparse coding on neurosynaptic cognitive system." *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, 2017.

340. Sawada, Jun, et al. "Truenorth ecosystem for brain-inspired computing: scalable systems, software, and applications." *High Performance Computing, Networking, Storage and Analysis, SC16: International Conference for*. IEEE, 2016.

341. Ren, Shaoqing, et al. "Faster R-CNN: Towards real-time object detection with region proposal networks." *Advances in neural information processing systems*. 2015.

342. Coates, Adam, et al. "Deep learning with COTS HPC systems." *International Conference on Machine Learning*. 2013.

343. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition.*Proceedings of the IEEE*, November 1998.

344. Gupta, Suyog, et al. "Deep learning with limited numerical precision." *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*. 2015.

345. Courbariaux, Matthieu, Yoshua Bengio, and Jean-Pierre David. "Binaryconnect: Training deep neural networks with binary weights during propagations." *Advances in Neural Information Processing Systems*. 2015.

346. Zhu, Chenzhuo, et al. "Trained ternary quantization." *arXiv preprint arXiv:1612.01064* (2016).

347. Ba, Jimmy, and Rich Caruana. "Do deep nets really need to be deep?." *Advances in neural information processing systems*. 2014.

348. Urban, Gregor, et al. "Do Deep Convolutional Nets Really Need to be Deep and Convolutional?." *arXiv preprint arXiv:1603.05691* (2016).

349. LeCun, Yann, et al. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86.11 (1998): 2278-2324.

350. S. A. Nene, S. K. Nayar and H. Murase, ”Columbia Object Image Library (COIL-20)," Technical Report CUCS-005-96, February 1996.

351. S. A. Nene, S. K. Nayar and H. Murase, "Columbia Object Image Library (COIL-100),"Technical Report CUCS-006-96, February

352. Alom, Md Zahangir, et al. “Object recognition using cellular simultaneous recurrent networks and convolutional neural network.” *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, 2017.

353. Han, Song, Huizi Mao, and William J. Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." *arXiv preprint arXiv:1510.00149* (2015).

354. Qiu, Jiantao, et al. "Going deeper with embedded FPGA platform for convolutional neural network." *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2016.

355. He, Kaiming, and Jian Sun. "Convolutional neural networks at constrained time cost." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015.

356. 13. Lin, Zhouhan, et al. "Neural networks with few multiplications." *arXiv preprint arXiv:1510.03009* (2015).

357. 14. Courbariaux, Matthieu, Jean-Pierre David, and Yoshua Bengio. "Training deep neural networks with low precision multiplications." *arXiv preprint arXiv:1412.7024* (2014).

358. Kim, Minje, and Paris Smaragdis. "Bitwise neural networks." *arXiv preprint arXiv:1601.06071* (2016).

359. Dettmers, Tim. "8-Bit Approximations for Parallelism in Deep Learning." *arXiv preprint arXiv:1511.04561* (2015).

360. Rastegari, Mohammad, et al. "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks." *arXiv preprint arXiv:1603.05279*(2016).

361. He, Qinyao, et al. "Effective Quantization Methods for Recurrent Neural Networks." *arXiv preprint arXiv:1611.10176*(2016).

362. Kapur, Supriya, Asit Mishra, and Debbie Marr. "Low Precision RNNs: Quantizing RNNs Without Losing Accuracy." *arXiv preprint arXiv:1710.07706* (2017).

363. Zhou, Shu-Chang, et al. "Balanced quantization: An effective and efficient approach to quantized neural networks." *Journal of Computer Science and Technology* 32.4 (2017): 667-682.

364. Elman, Jeffrey L. "Finding structure in time." *Cognitive science*14.2 (1990): 179-211.

365. S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In S. C. Kremer and J. F. Kolen, editors, A Field Guide to Dynamical Recurrent Neural Networks. IEEE Press, 2001.

366. Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.

367. Chung, Junyoung, et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling." *arXiv preprint arXiv:1412.3555* (2014).

368. Jozefowicz, Rafal, Wojciech Zaremba, and Ilya Sutskever. "An empirical exploration of recurrent network architectures." *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*. 2015.

369. Xingjian, S. H. I., et al. "Convolutional LSTM network: A machine learning approach for precipitation nowcasting." Advances in neural information processing systems. 2015.

370. Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). Learning Word Vectors for Sentiment Analysis. *The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).*

371. Srivastava, Nitish, Elman Mansimov, and Ruslan Salakhudinov. "Unsupervised learning of video representations using lstms." *International conference on machine learning*. 2015.

372. Gers, Felix A., Nicol N. Schraudolph, and Jürgen Schmidhuber. "Learning precise timing with LSTM recurrent networks." *Journal of machine learning research* 3.Aug (2002): 115-143.

373. Tom Simonite (8 May 2013). "D-Wave's Quantum Computer Goes to the Races, Wins". MIT Technology Review. Retrieved 12 May 2013.

374. Boros, Endre, Peter L. Hammer, and Gabriel Tavares. "Local search heuristics for quadratic unconstrained binary optimization (QUBO)." *Journal of Heuristics* 13, no. 2 (2007): 99-132.

375. Wang, Di, and Robert Kleinberg. "Analyzing quadratic unconstrained binary optimization problems via multicommodity flows." *Discrete Applied Mathematics* 157.18 (2009): 3746-3753.

376. Raj, Ashish, et al. "Bayesian parallel imaging with edge-preserving priors."*Magnetic Resonance in Medicine* 57.1 (2007): 8-21.

377. Boros, E., P.L. Hammer, M. Minoux and D. Rader. Optimal cell flipping to minimize channel density in vlsi design and pseudo-Boolean optimization. Discrete Applied Mathematics 90, (1999), pp. 69–88.

378. Ju¨nger, M., A. Martin, G. Reinelt and R. Weismantel. Quadratic 0-1 optimization and a decomposition approach for the placement of electronic circuits. Mathematical Programming 63, (1994), pp. 257–279.

379. McBride, R.D. and J.S. Yormark. An implicit enumeration algorithm for quadratic integer programming. Management Science 26, (1980), pp. 282–296.

380. Kubiak, W. New results on the completion time variance minimization. Discrete Applied Mathematics 58, (1995), pp. 157–168.

381. Wang, H., B. Alidaee and G. Kochenberger. Evaluating a clique partitioning problem model for clustering data mining. Tech. Rep. HCES-06-04, Hearin Center for Enterprise Science, University of Mississippi (2004).

382. Raj, A., G. Singh and R. Zabih. MRF's for MRI's: Bayesian reconstruction of MR images via graph cuts. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR'06) (2006).

383. Phillips, A.T. and J.B. Rosen. A quadratic assignment formulation for the molecular conformation problem. Journal of Global Optimization 4, (1994), pp. 229– 241.

384. Picard, J.C. and H.D. Ratliff. Minimum cuts and related problems. Networks 5, (1975), pp. 357–370.

385. Jordan, Michael I., and David E. Rumelhart. "Forward models: Supervised learning with a distal teacher." *Cognitive science* 16.3 (1992): 307-354.

386. Hernandez, Maritza, et al. "A Novel Graph-based Approach for Determining Molecular Similarity." *arXiv preprint arXiv:1601.06693* (2016).