A Detachable LSTM with Residual-Autoencoder Features Method for Motion Recognition in Video Sequences

Thesis

Presented in Partial Fulfillment of the Requirements for the Degree Master of Science in the Graduate School of The Ohio State University

By

Sheng Ding, M.S

Graduate Program in Department of
Electrical and Computer Engineering

The Ohio State University

2020

Master's Examination Committee:

Prof. Xiaorui Wang, Advisor

Prof. Wladimiro Villarroel

Abstract

Motion recognition in video sequences is a challenging computer vision problem. Actions are represented as a series of frames in video environments, which can be easily understood by analyzing multiple frames' contents. In this thesis, we recognize human actions in a way similar to our observation of actions in real life, which is exploring the features of consecutive frames and the connection between them.

Traditionally, feature extraction and recognition in video motion recognition are integrated, and the training time is lengthy [1–5]. Especially when new data is given, the time cost of retraining may be days which is too high, and the reliability for a new environment is low. We want to break down this process to reduce the difficulty of training, and at the same time, to find a reliable description of the process of feature extraction.

In this thesis, we propose a detachable training motion recognition method by processing the video data using Residual Block Autoencoder (ResAE) and Long Short-Term Memory (LSTM) network. The proposed method can provide a reliable feature extraction and process long videos by analyzing the features in frame sequences.

Experimental results show acceptable performance over 60% accuracy, which is promising, in action recognition using the proposed method on UCF-101 (action recognition dataset).

This is dedicated to my dear families and friends.

Acknowledgments

Without the help of the following people, I would not have been able to complete my thesis. My heartfelt thanks to:

Dr. Xiaorui Wang, for being my mentor in research. Under his guidance, I learned more professional knowledge and research skills, involved with exciting research topics, but more importantly, I have got the chance to improve myself with a better understanding of the research methodology, ideas development, and research attitude. I feel very grateful for the opportunity to have him as my research advisor.

Dr. Han-Wei Shen, for his valuable suggestions on the algorithm model design and experiment implementation, as well as his great advice and helps with the revision of my thesis writing. I do appreciate him for all the above helpings.

Dr. Yunhao Bai, for his unselfish suggestions and generous helpings in sharing his knowledge and experience of OS concept and framework implementation.

Vita

December 24, 1995 ..... Born - Wuhu, China

Sept, 2014 - June, 2018 ..... B.E. Electrical Packaging,
Beijing Institute of Technology

Aug, 2018 - present ..... M.S. candidate Electrical and Computer Engineering,
Ohio State University

Fields of Study

Major Field: Electrical & Computer Engineering

Table of Contents

List of Tables

Table **Page**

List of Figures

Chapter 1: Introduction

1.1 Background

Motion recognition in video sequences is a challenging computer vision problem, which involves the similarity of visual content, the change of the same motion perspective, the camera's movement with action performer, the scale and posture of actors, and different illumination conditions. In general, human motion is the movement of body parts interacting with objects in the environment. In a video environment, an action is represented by a series of frames, which can be easily understood by analyzing multiple frames' contents [3]. In this thesis, we recognize human actions in a way similar to our observation of actions in real life, which is exploring the features of consecutive frames and the connection within them.

1.2 Problem Statement

The training process is end-to-end in some existing video motion recognition models, which means feature extraction and recognition are encapsulated together for training. Such encapsulation undoubtedly reduces the design requirements for the model because the end-to-end models focus on results more and often ignore the hidden layer. In such a model, we cannot assess the quality of the features extraction;

the training time for months and computing resources (more than 24GB ram) are enormous as well.

We hope to disassemble the feature extraction and recognition, design a neural network structure capable of handling complex frames (complex environment and detail) in the feature extraction part, reduce the size of the feature representation as much as possible, and find a reliable method to evaluate its performance.

1.3 Approach and Result

In this thesis, some modifications were made to the feature extractor, and the whole training process is divided into two parts:1) Frame compression and feature extraction, 2) Recognition based on continuously processed frame.

We designed an LSTM as a recognition component. We want to reduce the training time as much as possible for preprocessing while still having high robustness. To attain that goal, an Autoencoder based on the residual block, similar to the architecture in [3], is presented as a compression/extraction component.

Applying the Hand Gesture Recognition Database, we demonstrate the feasibility of the encoder in feature extraction. By experimenting with the UCF-101 dataset, we evaluate the performance of ResAE+LSTM.

1.4 Thesis Overview

The remaining chapters of this thesis introduce some works already done and document our design methodology, the results of our work, and the conclusion of this research.

Chapter 2 makes a brief review of the previous work in both the traditional method and the modern deep learning method.

Chapter 3 introduces how different model units are built internally. Including the structure design of Autoencoder and LSTM layer. The working principle and mathematical rationality of residual block are also mentioned.

Chapter 4 presents the video dataset we used and some preprocessing of the video frame.

Chapter 5 provides the experimental process and all the experimental data. Among them, the Autoencoder training relevant data accounts for a large proportion as well as details of the dataset generation and usage in model training.

Chapter 6 summarizes the thesis's results and analyzes the model's characteristics, which give pointers to future research based on this exemplary work.

Chapter 2: Related Works

This chapter introduces standard Deep learning models and their utility in Motion Recognition. A general overview of the architectures in different Deep learning models supports a discussion of the problem at hand and the goals of this project.

Researchers have proposed many motion recognition methods based on manual (traditional methods) or deep network approaches in the past decade.

2.1 Traditional Methods

Earlier works were based on mathematical features for non-realistic action, in which actors often performed specific actions in scenes with simple backgrounds. Such systems extract low-level features from video data and then feed them to classifiers such as support vector machines (SVM), decision trees, and k-nearest neighbors (KNN) for action recognition.

For instance, the geometrical properties of space-time volume (STV), called action sketch, were analyzed by Yilmaz and Shah [1]. They stacked body contours in the time axis by capturing direction, speed, and shape of STV for action recognition.

Hu et al. [2] used two types of features: motion history image (MHI) and histogram of oriented gradients feature (HOG). The former is the foreground image subtracted from the background scene, while the latter refers to the magnitudes and directions

of edges. Then these features are fused and classified by simulated annealing with multiple instances learning SVM (Smile-SVM).

However, these methods, based on mathematical features, have certain limitations. For example, the STV-based approach is not useful for identifying the actions of multiple people in one scene. The technique based on MHI only works well in simple datasets. To deal with complex datasets, we need a hybrid approach that combines different functions and preprocessing, which will increase the computational complexity of the target system [6].

These limitations can make lengthy videos and real-time applications with continuous video streaming difficult.

2.2 Deep nets in Motion Recognition

In addition to the motion recognition methods based on manual features, several methods based on deep learning have been proposed in recent years. Deep learning has shown significant progress in many areas, such as image classification, person recognition, object detection, speech recognition, and bioinformatics [7].

The deep learning architectures for action recognition could be classified into three categories based on feature extraction and fusion in different domains. Simonyan and Zisserman [4] raised an architecture which is a two-stream convolutional network based on two separate recognition streams (spatial and temporal), combined by late fusion. The spatial stream performs action recognition from still video frames, while the temporal stream is trained to recognize action from motion in the form of dense optical flow (Figure 2.1). However, the precomputing of optical flow requires extra GPU running time and storage space, which has become the two-stream algorithm's

bottleneck. Besides, the traditional optical flow calculation method is entirely independent of the two-stream framework, not end-to-end training, result in that the motion information in advance is not optimal [8].

Figure 2.1: Optical Flow

The second architecture is focusing on the three-dimensional convolution kernel. Similar to still images, a video is a set of consecutive pictures with some inner correlation in the time domain. As an extension of 2D ConvNets, 3D ConvNets is more suitable for learning spatiotemporal features, which means it does some help in video processing. In [5], Tran and his colleagues proposed a new model C3D based on 3D ConvNets. Simultaneously, due to the good performance in still image recognition, ResNet which is an improved architecture of ConvNets came into the researchers' view [9, 10].

The problem is that unlike the motion recognition in still images, the motion recognition based on video should have the ability to represent the evolution of short-term small motions and long-term representation. Some motions can be reliably distinguished through the movements captured from successive frames, such as short-term actions, but some kinds of motions require the overall features of a more lasting video like long-term actions. To illustrate the above issues, we show some example videos from UCF-101 [11] dataset in Figure 2.2 and 2.3. As shown in Figure 2.2 , "Playing Piano" and "Archery" can be easily recognized through the information of a static frame or small motion between consecutive frames. However, sometimes short clips

are not sufficient for classifying similar classes (High Jump vs. Long Jump), as shown in Figure 2.3. Therefore, it is critical to exploit the complementary nature of the single static frame, the short and long-term temporal evolution, and the video-level representations. With that in mind, Liu et al. [12] proposed Temporal Convolutional 3D Network (T-C3D) based on the Multi-clip C3D model [13] architecture.

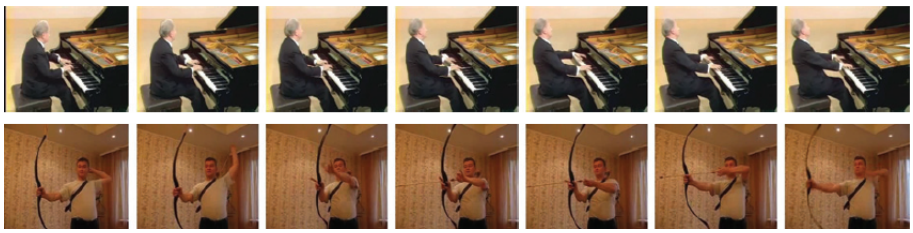

Figure 2.2: Playing Piano and Archery

Figure 2.3: High Jump and Long Jump

Unlike the first two models, the third architecture exploits the convolution unit and Recurrent Neural Network (RNN), which has "time depth" and forms implicit component representation in the time domain [14]. A significant limitation of simple RNN models that strictly integrate state information over time is known as the "vanishing gradient" effect: in practice, the ability to backpropagate an error signal through a long-range temporal interval becomes increasingly complex [15]. Long Short-Term Memory (LSTM) units, first proposed in [16], are recurrent modules that enable long-time learning. LSTM units have hidden state augmented with nonlinear mechanisms to allow the state to propagate without modification, be updated, or be

reset, using simple learned gating functions [17]. LSTMs have recently been demonstrated to be capable of large-scale learning of speech recognition [18] and language translation models [19, 20].

Considering that the LSTM unit is a deep learning architecture with cumulative effects, if the input size of the LSTM unit is too large ($\geq 10^4$), each unit will correspondingly have more losses, thus affecting the final output. In [6], Ullah et al. used CNN as a feature extractor so that a 112x112 image can be compressed into a 1x1000 feature matrix. However, the whole process is end-to-end, which means the convolution nets and LSTM are trained together. Even regardless of how much time is spent in the training process, the performance of the feature extractor is also an unknown factor.

Compared to general CNN nets, though Autoencoder is also a lossy compression process, training allows its compression results to be restored as much as possible to the original input, which can be a good indicator of feature extraction [21, 22].

Chapter 3: Model Architecture and Details

This chapter introduces different model units and their inner structure in the proposed framework, including features extraction through Resnet-Autoencoder for frames and recognizing actions from the sequence of frames in video using LSTM.

3.1 Structure Overview

Figure 3.1 illustrates the framework of the proposed ResAE-LSTM network. First, $\mathcal{S}$ frames are uniformly selected from the video to form a clip which represents the entire video. For example, for a 151 frames video, the small clip with 15 frames is formed by picking the frame with a 10 frames interval (10,20,...,150). We extract ResAE features for all frames from these small clips. Second, the feature representing the sequence of action are fed to the proposed LSTM in $\mathcal{S}$ chunks, where each chunk is the features representation of the video frame and input to one RNN step. The final state of LSTM is obtaining video-level scores and analyzing for final recognition of an action in a video.

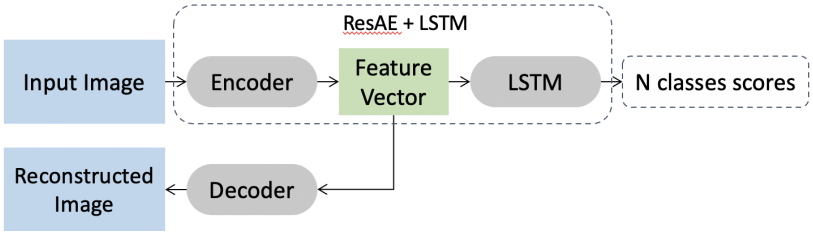

Figure 3.1: Model Structure

3.2 Feature Extraction

The feature extraction of the proposed method is completed by the encoder in the Autoencoder. We evaluate whether the feature matrix is useful and train the Autoencoder through the restoration of the decoder to the feature matrix.

3.2.1 Autoencoder

We want to discover a "proper" way to extract features for consecutive frames. In the traditional CNN+LSTM model, as mentioned in Chapter 2.2, the output of CNN is not evaluable. Though it dramatically reduces the size of input images, it may bring some unexpected noise or unnecessary information to the LSTM net and affect the training process, recognition output either.

Autoencoder is an unsupervised artificial neural network that learns how to compress and encode data efficiently then learns how to reconstruct the data back from the reduced encoded representation to a representation that is as close to the original input as possible.

Unlike other Feedforward Neural Networks (FNN), as shown in Figure 3.2, which focus on the Output Layer and error rate, Autoencoder focuses on the Hidden Layer. Given the example in the Modified National Institute of Standards and Technology (MNIST) dataset, the output of the Hidden Layer is a compressed feature matrix,

as shown in Figure 3.3. This feature matrix is reliable because it can be restored to the original image, while the output of the FNN softmax layer is just the result of multiple convolutional layers which we can not evaluate.

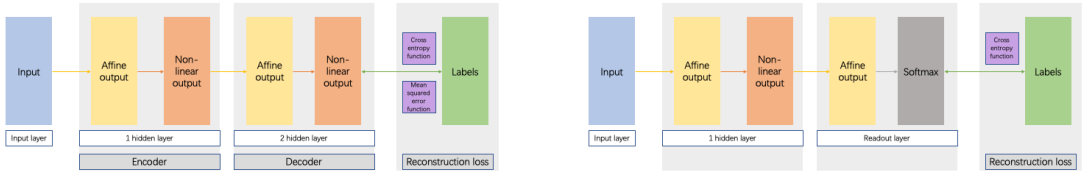

Figure 3.2: FNN Autoencoder vs. FNN Classfication

The mathematical expression for Autoencoder is as follows, where ϕ and ψ represent encoder and decoder; X and $\mathcal{L}$ represent the input and loss function respectively.

$$\phi, \psi = argmin_{\phi,\psi}\mathcal{L}(X, (\phi \circ \psi)X) \tag{3.1}$$

It can be seen that Autoencoder is an enhanced Principal Component Analysis (PCA) element: It has some nonlinear transformation units, so the learned code can be more refined and have more vital expression ability for input.

3.2.2 Residual Block

Resnet was proposed to solve the problem of network degradation. The degradation problem refers to the fact that the model accuracy is not always improved with the increase of network depth, and this problem is not caused by overfitting. Because after deepening the network, not only the test error is rising, but also the training error is rising. The rising error for both training and test may be due to the fact that deeper networks are accompanied by gradient disappearance/explosion problems, which hinder the convergence of the network.

He et al. [9] have proposed that the solution to network degradation is to build an identity mapping, as shown in Figure 3.4. The original network input was x, and the output was $\mathcal{H}(x)$. Assume that $\mathcal{H}(x) = \mathcal{F}(x) + x$, so the network just needs to learn to output a residual $\mathcal{F}(x) = \mathcal{H}(x) - x$. The authors suggest that learning the residual $\mathcal{F}(x)$ is much easier than learning the original feature $\mathcal{H}(x)$ directly. According to the authors' analysis, the deeper model should have a training error no greater than its shallower counterpart if the added layers can be formed as identity mappings. However, it is not easy to approximate identity mappings by multiple nonlinear layers, causing the degradation problem. By applying residual learning reformulation, the solvers may easily approach the weights of the nonlinear layers toward zero to approach identity mappings, if they are optimal.

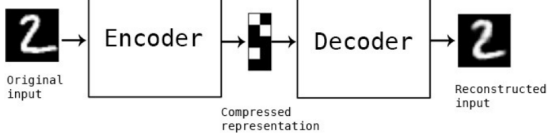

Figure 3.3: Autoencoder Example in MNIST

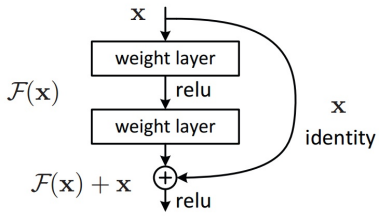

Figure 3.4: Identity Mapping

3.2.3 Structure Design

The architecture of Residual-Autoencoders we designed in this thesis is presented in Figure 3.5.

In the encoder part, a 4-layer structure is used. The second and third layers are convolutional layers with the residual block (Figure 3.6), called downsampling layers, whose strides are set to 2. Each time a downsampling layer is passed, the frame

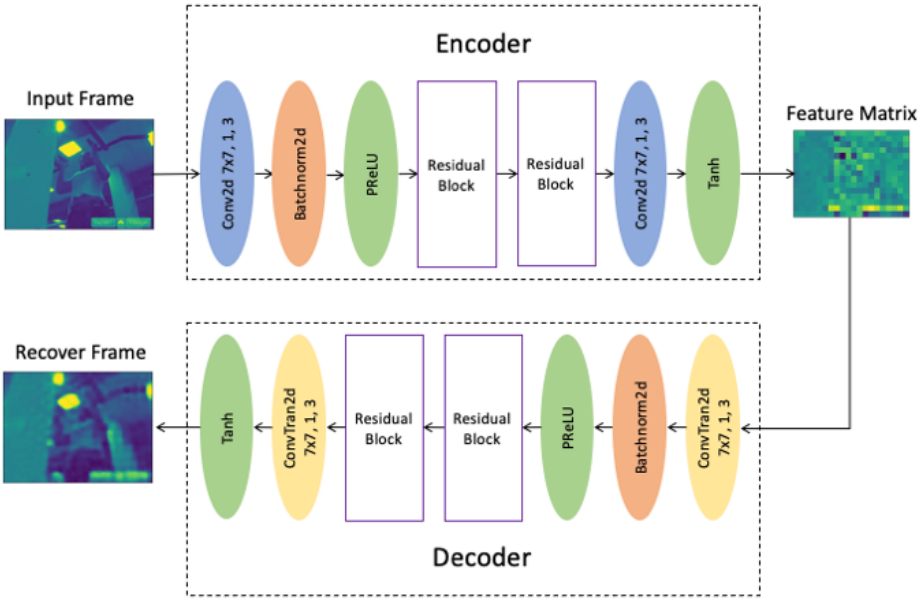

Figure 3.5: Autoencoder Structure

size becomes $1/4 * 1/4$ of the original one. Two downsampling layers mean that the final feature matrix is $1/256$ of the original frame. The dropout layer in the residual block with rate $= 0.5$ is used to improve the robustness and reduce the probability of overfitting [23]. The first and last convolutional layers do not change the frame size but expand the channels of training samples in order to preserve more information during size reduction.

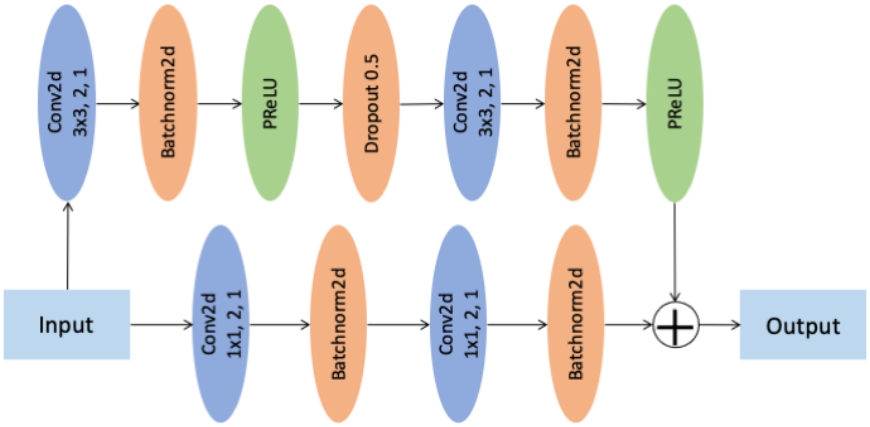

Figure 3.6: Residual Block Structure

To make the combination of functions work better, we set an activation layer after every convolution layer. The reason why we choose PRelu, as shown in Figures 3.5 and 3.6, rather than Relu is that comparing to get rid of all negative correlation elements, the retention of some of them is more conducive to feature extraction because the negative correlation also expresses correlation in a sense [24].

In the decoder part, which is symmetric to the encoder in process, transposed convolution is adopted to decode as the replacement of convolution in the decoder.

Regarding the choice of the loss function, considering that image coding restoration is essentially a regression problem, in [12], McCaffrey mentioned that Mean Squared Error could provide several local optimization solutions, convenient for the training of the model.

3.3 Recognition

3.3.1 RNN and LSTM

Humans don't start their thinking from scratch every second. Recurrent neural networks were invented according to this characteristic. They have chain structure loops, which can obtain input and give an output at each sequence node [25].

RNNs are able to connect previous information to the present task, such as using previous video frames to inform the understanding of the present frame. However, when it comes to a long sequence that a single chain structure cannot hold all the information, RNNs become unable to learn to connect the information.

As shown in Figure 3.7, X_0 and X_1 are two inputs at the very beginning of the sequence and do have a high correlation with the h_{t+1}. When t is large enough,

module A cannot remember the information all the way down to sequence nodes 0 and 1 which will certainly affect the result of predicting h_{t+1}.

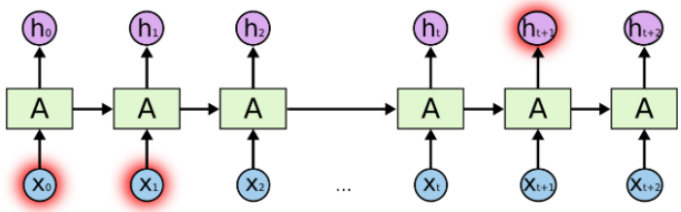

Figure 3.7: RNN Long-Term Dependencies

Long Short-Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber (1997), and were refined and popularized by many people later on.

LSTMs are explicitly designed to avoid the long-term dependency problem.

3.3.2 Mathematical Analysis

All recurrent neural networks have the form of a chain of repeating modules of a neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer. LSTMs also have this chain-like structure, but the repeating module has a different structure which makes the whole process into three stages: forget gate stage, select memory stage, and output stage, as shown in the middle module of the LSTM in Figure 3.8. To better explain how it works, all the variables used and their definitions are shown in Table 3.1.

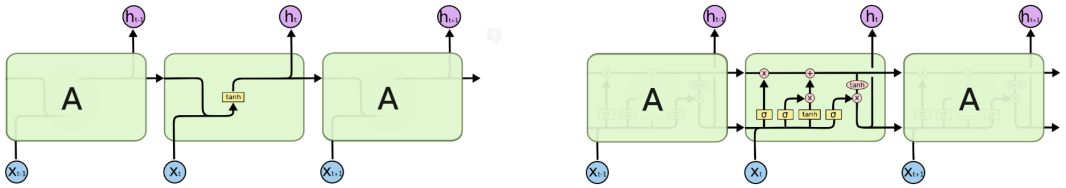

Figure 3.8: The Repeating Module in A Standard RNN Contains A Single Layer vs. An LSTM Contains Four Interacting Layers

x_t	Input at step t
$h_{t-1,t}$	Hidden state at step $t-1, t$
$C_{t-1,t}$	Cell state at step $t-1, t$
$W_{f,i,C,o}$	Trainable Weight in forget, input gate, Cell state and output layer
$b_{f,i,C,o}$	Trainable Bias in forget, input gate, Cell state and output layer
f_t	Forget items
i_t	Old information needs to be updated
$\tilde{C}_t$	New information needs to be added
o_t	Output selector
σ, tanh	Nonlinear sigmoid function, Activation function

Table 3.1: Description of Input and Output Parameters Used in The Proposed LSTM for Recognition.

The forget gate layer (Figure 3.9) is used to decide what information should be thrown away from the cell state. This decision is made by a sigmoid layer. It looks at h_{t-1} and x_t and outputs a number between 0 and 1 for each number in the cell state C_{t-1}.

The relation is given by the following equation:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3.2}$$

The memory select layer (Figure 3.10) is used to decide what new information can be stored in the cell state. It has two parts. First, a sigmoid layer decides which

values will be updated. Next, a tanh layer creates a vector of new candidate values, $\widetilde{C}_t$, that could be added to the state. In the end, by combining these two layers, the whole process creates an update to the state.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{3.3}$$

$$\widetilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{3.4}$$

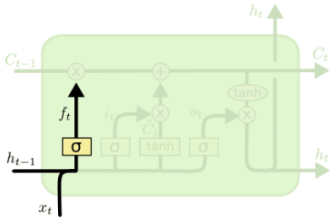

Figure 3.9: Forget Gate Layer

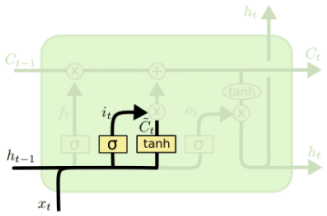

Figure 3.10: Memory Select Layer

After going through the forget gate layer and the memory select layer, cell state, C_{t-1}, can be updated into C_t (Figure 3.11).

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t \tag{3.5}$$

Finally, the output layer chooses what to output (Figure 3.12).

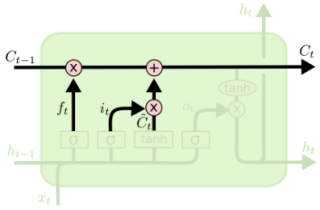

Figure 3.11: Update Cell State

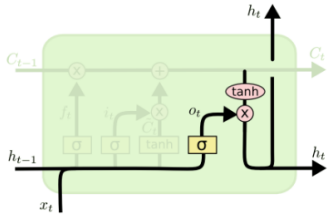

Figure 3.12: Output Layer

This output will be based on the filtered cell state. First, getting through a sigmoid layer to decide what parts of the cell state to output. Then, put the cell state

through tanh and multiply it by the output of the sigmoid gate, so that it can only output the parts we decided to.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{3.6}$$

$$h_t = o_t * \tanh(C_t) \tag{3.7}$$

Chapter 4: Dataset Design and Application

This chapter introduces the dataset used in this thesis. Apart from the necessary information, how is the video preprocessed will also be mentioned.

4.1 UCF-101

UCF-101 is an action recognition data set of real action videos, collected from YouTube, having 101 action categories [11].

The action categories can be divided into five types: 1) Human-Object Interaction, 2) Body-Motion Only, 3) Human-Human Interaction, 4) Playing Musical Instruments, 5) Sports.

To have better comparability, we choose 16 groups of videos of the second category, which are similar in video environment and mainly focus on human movements.

4.2 Video Preprocess

Considering that the inputs of the Autoencoder are single-frame images, the first step is to decompose the video into pictures according to the number of frames. Since training consists of compressing and recovering, all the images do not need to be tagged compared to the classification task.

4.2.1 Video Clips

A five-second video whose fps is 30 has 150 frames, and most of those frames are highly similar to those adjacent to them in time, which means that it makes no sense to put the entire video into the model. Duplicate images can be discarded with the method of spaced sampling.

In the training process, we took $\mathcal{S} = 15$ frames of each video to form a clip as indicated in Chapter 3.1.

4.2.2 Canny Edge Detection

In certain works mentioned in Chapter 2, such as [4], optical flow is extracted as one of the motion features. The reasonable point of pulling optical flow is that, firstly, it can simplify the data (only considering the optical flow field, that is, the motion information, removing complex environment information) and reduce the computation time. Second, it can make the input more diverse and improve the robustness of the model.

However, extracting optical flow is quite time-consuming and requires advanced computing equipment. Canny edge detection is one of several methods used to extract motion features besides optical flow. The process result is shown in Figure 4.1.

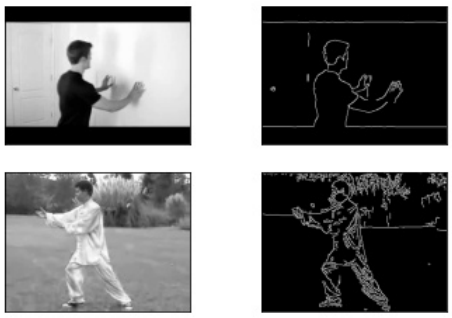

Figure 4.1: Origin Frame vs Canny Edge Frame

By applying edge detection, the human body's motion outline in the complex background can be found. For feature extraction, the lower the noise interference, the better the performance of the Autoencoder [26].

Chapter 5: Experiment and Data

5.1 Experiment Overview

In this chapter, we evaluate the proposed ResAE + LSTM model based on UCF-101. As mentioned in Chapter 3.1, we split the training process into two parts, respectively, representing feature extraction and motion recognition.

Prior to the training of the Autoencoder, we conducted a preliminary experiment on the Hand Gesture Recognition Database to find out the influence of input size on the time of training and the performance of the Autoencoder. This preliminary experiment helps us in scaling video frame.

In the main experiment, we first train the automatic encoder, and the evaluation criterion is the loss rate of the restored frame. We put both the original frame and the edge detection frame into the Autoencoder to obtain two-stream feature matrices.

Then, we conduct LSTM training based on the two-stream feature matrices in the small clips from the video with an average length of 15 frames.

5.2 Autoencoder

5.2.1 Performance Prediction

The training of encoders is a complicated process. We want the coding time to be as short as possible, and the recovery rate to be as high as possible, which means the loss needs to remain low. Intuitively, it has a high demand for the size of the input image and the random preprocessing of the image like flip, not only edge detection.

We used the Hand gesture recognition database [27] which contains 20000 still images in 10 categories for Autoencoder testing and obtained the effect of input size on Autoencoder function and convergence time. The relationship is shown in Figures 5.1 and 5.2.

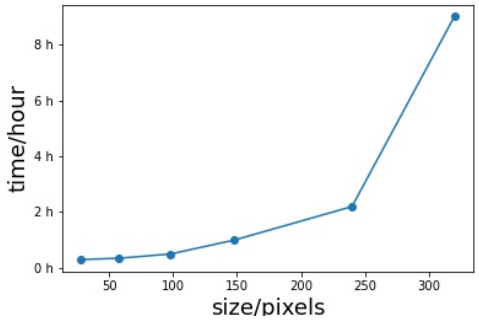

Figure 5.1: Training Time with Size Increasing

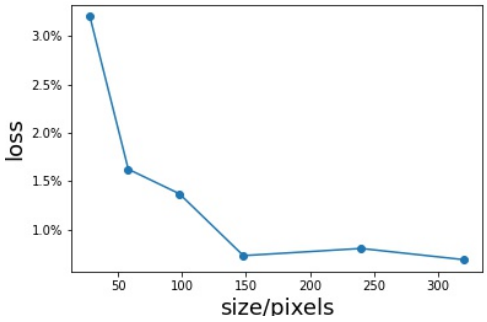

Figure 5.2: Loss with Size Increasing

From these two graphs, we can conclude that as the input size increases, the training cost explodes exponentially. Simultaneously, the recovery difficulty is greater, and the loss rate gradually converges to a certain value.

We present some restoration results to support this view (Figure 5.3). It is evident

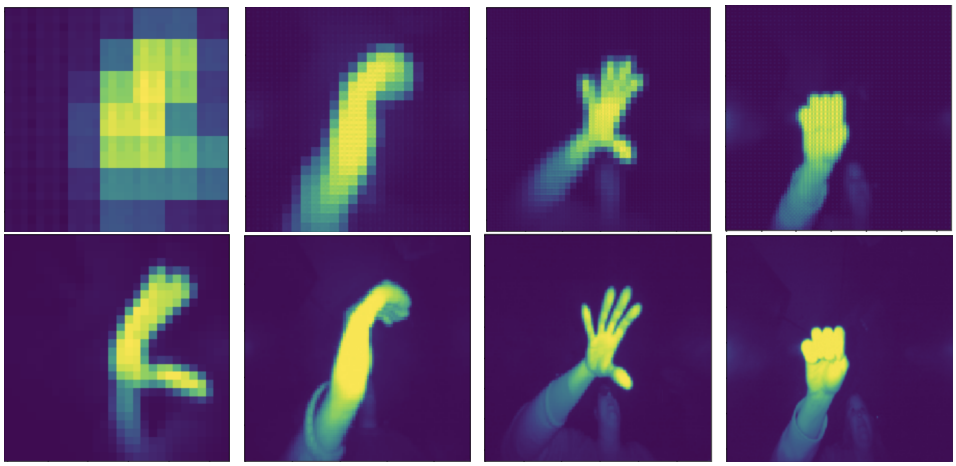

Figure 5.3: Restoration (top) and Original (bottom) Image for Input Size 28*28, 98*98, 148*148, and 320*320

that when input is relatively small, the necessary information is insufficient, and the restoration has a significant loss.

In this case, to achieve the tradeoff of time and loss, it is unnecessary to have a large input size such as 1280*720. A reasonable size range is 200 to 250, within which training is often about a few hours and loss is at a convergent level of about 0.7%.

5.2.2 ResAE Training and Visual Results

We took Body-Motion Only set in UCF-101 as the final training dataset. There are over 1,500 videos in 16 categories in the database. After the videos were decomposed frame by frame, we obtained a total of 323,260 images, of which 90% were used as the training set and 10% as the test set.

To prevent overfitting, we set the learning rate to 0.001 and the decay rate to 0.00001. With the time restriction, 10000 iterations are operated with 50 frames per

batch. The testing loss shows that ResAE has good performance (Mean Square Error less than 1%), as show in Figure 5.4).

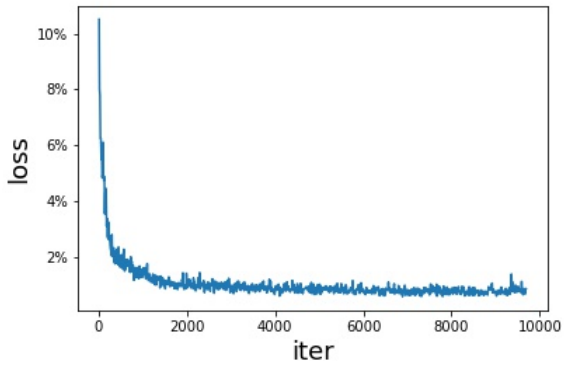

Figure 5.4: Residual-Autoencoder Test Loss

The original frame size is 240*320. After the feature extraction, we get a 15*20 feature matrix. At the same time, we also continue the idea of two-stream; one is the original image with RGB 3 channels, the other is the grayscale image processed by canny edge detection. Because background information is essentially omitted in canny edge detection images, the loss rate of restored images using Canny Edge detection is lower (Figure 5.5, 5.6). We can consider that the motion information contained in the feature matrix of the hidden layer is more effective.

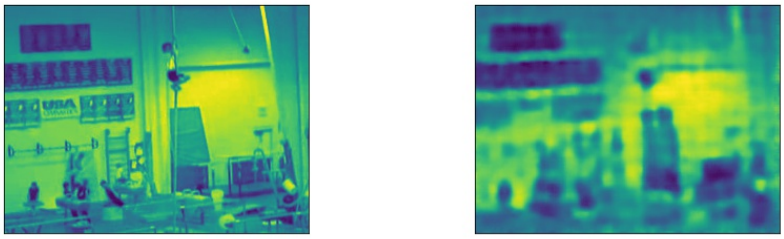

Figure 5.5: Original (left) and Recovered (right) Images (with average loss 0.7%)

In our design, the feature matrix of the original image, which is the input to LSTM, has three channels. We are more concerned about the dynamic components of the picture for the action, so we can consider using only one of the three channels because

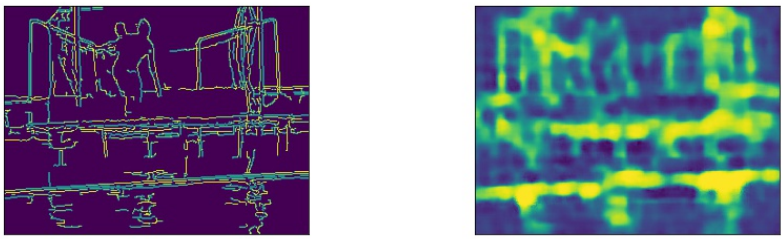

Figure 5.6: Canny Edge (left) and Recovered (right) Images (with average loss 0.3%)

channel decomposition does not affect dynamic features; moreover, the encoding of the edge detection image also compensates for these losses. In Figure 5.7, we can see that all the details are blurred similarly in different channels.

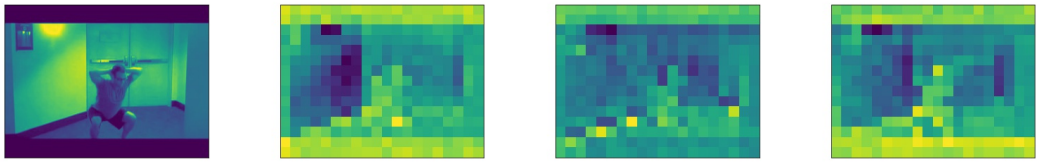

Figure 5.7: Original Frame with Three Channels

5.3 LSTM Training and Visual Results

After the training of Autoencoder, we take one of the feature matrix channels as the input to LSTM. Then, the coded frame is reshaped into a 1-dimension vector before sent into LSTM.

Due to the time limitations, we only trained 600 iterations, which run through half of the whole dataset. In Figure 5.8, the training loss is significantly reduced from 3% to 0.5%. The accuracy of the model prediction reached 60% (Figure 5.9).

Compared with the completed training model like T-C3D with accuracy at 79.7%, the training of ResAE-LSTM is not sufficient due to time and resource constraints. So the final accuracy is not very high (an average accuracy is between 70% and 85%).

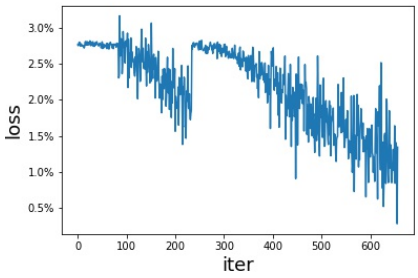

Figure 5.8: LSTM Train Loss

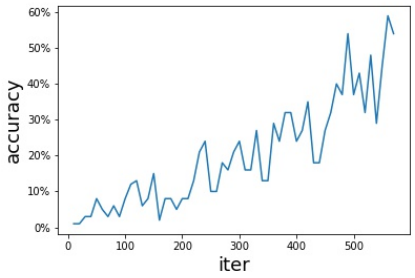

Figure 5.9: LSTM Test Accuracy

However, in the case of only going through half of the data set, the training loss has steadily declined, indicating a stable convergence trend. In Figure 5.9, accuracy is also rising, which may suggest that if more computing resources and time are available, higher accuracy can be achieved.

Chapter 6: Conclusion

In this thesis, we presented an architecture of a Residual-Autoencoder + LSTM which is a deep learning model, combined with a compression technique for motion recognition.

We gave a detailed design structure and introduced its mathematical principle. Residual-Autoencoder first extracts the feature matrix from the video frames. Then, LSTM is fed with these feature matrix for recognition. We analyzed videos in small clips, where the length of clips depends on S frames, which are picked in some adjustable time interval preprocessing. Due to these properties, the proposed method is capable of learning long term complex sequences in videos.

The feasibility of the new model was verified by testing using the UCF-101 dataset. Our model can evaluate the feature extraction performance of the middle layer with an average loss of 0.7% for the original frame and 0.3% for the Canny Edge detection frame, which improves the reliability of LSTM classification work and achieves a relatively decent 60% accuracy in motion recognition, considering a limited training due to time and resource constraints. In addition to the final accuracy, the advantages of the new model are also reflected in the training process. Compared with T-C3D and other models that take several weeks of training, ResAE+LSTM only takes a few days. At the same time, in the case of motions with the same label but different

backgrounds, it is only necessary to retrain ResAE to extract corresponding features instead of retraining the whole model end-to-end. It can be said that the new model is conducive to transfer learning.

According to the performance and training process of the model in the practical video sets, we found some shortcomings of the new model. The first is the performance symmetry of the encoder and decoder. In our design, the decoder plays a critical role in image restoration due to its symmetry with the encoder structure, which will cause unavoidable interference to the evaluation of feature extraction. In the future, we can simplify the structure of the decoder as much as possible and reduce its impact. Another con is that the structure of LSTM is quite simple in this thesis and does not make full use of two-stream features (original and canny edge detection) of the motion. To improve that, we may try to apply multi-layers LSTM for multi-dimension inputs later, which may increase training time but should help raise the model's accuracy.

Bibliography

[1] A. Yilmaz and M. Shah. Actions sketch: A novel action representation. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 984–989, Jun 2005.

[2] Y. Hu, L. Cao, F. Lv, S. Yan, Y. Gong, and T. S. Huang. Action detection in complex scenes with spatial and temporal ambiguities. *The IEEE International Conference on Computer Vision (ICCV)*, pages 128–135, 2009.

[3] N. Bian, F. Liang, H. Fu, and B. Lei. A deep image compression framework for face recognition. *2019 2nd China Symposium on Cognitive Computing and Hybrid Intelligence (CCHI)*, pages 99–104, 2019.

[4] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. *Advances in Neural Information Processing Systems*, 2014.

[5] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. *The IEEE International Conference on Computer Vision (ICCV)*, Dec 2015.

[6] A. Ullah, J. Ahmad, K. Muhammad, M. Sajjad, and S. W. Baik. Action recognition in video sequences using deep bi-directional lstm with cnn features. *IEEE Access*, 6:1155–1166, 2018.

[7] S. K. Choudhury, P. K. Sa, R. P. Padhy, S. Sharma, and S. Bakshi. Improved pedestrian detection using motion segmentation and silhouette orientation. *Multimedia Tools Appl 77*, page 13075–13114, 2018.

[8] Y. Zhu, Z. Lan, S. Newsam, and A. G. Haauptmann. Hidden two-stream convolutional networks for action recognition. *Asian Conference on Computer Vision*, pages 363–378, 2017.

[9] K. He, X. Zhang, S. Ren, and Sun J. Deep residual learning for image recognition. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016.

[10] Christoph Feichtenhofer, Axel Pinz, and Richard P. Wildes. Spatiotemporal multiplier networks for video action recognition. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[11] S. Khurram, A. R. Zamir, and S. Mubarak. Ucf101: A dataset of 101 human actions classes from videos in the wild, 2012.

[12] K. Liu, W. Liu, C. Gan, M. Tan, and H. Ma. T-c3d: Temporal convolutional 3d network for real-time action recognition. *Thirty-Second AAAI Conference on Artificial Intelligence*, pages 7138–7145, 2018.

[13] Carreira Joao and Zisserman Andrew. Quo vadis, action recognition? a new model and the kinetics dataset. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[14] R. J.Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1989.

[15] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[16] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

[17] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[18] A. Graves and N. Jaitly. Towards end-to-end speech recognition with recurrent neural networks. *International Conference on Machine Learning*, pages 363–378, 2014.

[19] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, 2014.

[20] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *SSST Workshop*, pages 103–111, Oct 2014.

[21] Haojie Liu, Tong Chen, Qiu Shen, Tao Yue, and Zhan Ma. Deep image compression via end-to-end learning, 2018.

[22] Z Cheng, H Sun, Masaru Takeuchi, and Jiro Katto. Performance comparison of convolutional autoencoders, generative adversarial networks and super-resolution for image compression. *The IEEE Conference on Computer Vision and Pattern Recognition Workshops(CVPR)*, 2018.

[23] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[24] James D. McCaffrey. Why you should use cross-entropy error instead of classification error or mean squared error for neural network classifier training. `https://jamesmccaffrey.wordpress.com/2013/11/05/`. Accessed November 5, 2013.

[25] Christopher Olah. Understanding lstm networks. `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`. Accessed August 27, 2015.

[26] S. Arif, J. Wang, T. Ul Hassan, and Z. Fei. 3d-cnn-based fused feature maps with lstm applied to action recognition. *Future Internet*, 2019.

[27] T. Mantecón, C.R. del Blanco, F. Jaureguizar, and N. García. Hand gesture recognition using infrared imagery provided by leap motion controller. *Int. Conf. on Advanced Concepts for Intelligent Vision Systems (ACIVS)*, pages 47–57, 24–27, Oct 2016.