EFFECT OF ENHANCEMENT ON CONVOLUTIONAL NEURAL NETWORK
BASED MULTI-VIEW OBJECT CLASSIFICATION

Thesis
Submitted to
The School of Engineering of the
UNIVERSITY OF DAYTON

In Partial Fulfillment of the Requirements for
The Degree of
Master of Science in Electrical Engineering

By
Zhiyuan Xie
Dayton, Ohio
May, 2018
EFFECT OF ENHANCEMENT ON CONVOLUTIONAL NEURAL NETWORK
BASED MULTI-VIEW OBJECT CLASSIFICATION

Name: Xie, Zhiyuan

APPROVED BY:

___________________________               ___________________________
Vijayan K. Asari, Ph.D.               Eric J. Balster, Ph.D.
Advisory Committee Chairman               Committee Member
Professor               Associate Professor
Dept. Electrical & Computer               Dept. Electrical & Computer
Engineering               Engineering

___________________________               ___________________________
Theus H. Aspiras, Ph.D.               Eddy M. Rojas, Ph.D., M.A., P.E.
Committee Member               Dean, School of Engineering
Research Scientist               Professors
Dept. Electrical & Computer               Dept. Electrical & Computer
Engineering               Engineering

___________________________               ___________________________
Robert J. Wilkens, Ph.D., P.E.               Eddy M. Rojas, Ph.D., M.A., P.E.
Associate Dean for Research and               Dean, School of Engineering
Innovation               Professors
Professor               Professors
School of Engineering               School of Engineering
ABSTRACT

EFFECT OF ENHANCEMENT ON CONVOLUTIONAL NEURAL NETWORK BASED MULTI-VIEW OBJECT CLASSIFICATION

Name: Xie, Zhiyuan
University of Dayton
Advisor: Dr. Vijayan K. Asari

The main goal of this thesis is classification of multi-view objects by using convolutional neural networks (CNN), and evaluation of the recognition performance on images preprocessed by enhancement technologies such as multilevel windowed inverse sigmoid (MWIS) function and locally tuned sine nonlinearity (LTSN) technique. Humans can easily recognize objects in different observational directions, but machines cannot achieve this easily. The convolutional neural network (CNN), which has successfully been used to do visual imagery analysis, is a deep learning, feed-forward neural network that collects features of an image and classify them accordingly. A multi-layer CNN architecture is designed for multi-view object classification by appropriately choosing the number of layers, the sequence of layers cascading, and size of the filters. It is expected that the enhanced images exhibit stronger features. Therefore, we apply image enhancement techniques before the convolutional neural network to observe the recognition performance. The datasets used for performance evaluation in this work are from the Columbia Object Image Library (COIL-100) and
Multi-view Car dataset. It is observed that the preprocessing by image enhancement can provide improved performance in some cases of the smaller training set. Research work is in progress to modify the CNN architecture to see the impact of recognition performance for multi-view object classification. Advanced non-linear enhancement technologies might also be investigated to see the effectiveness in classification.
ACKNOWLEDGEMENTS

Although the process is full of difficulty, I finished my thesis with the help from my family, my advisor, the Vision Lab crew and my friends. Therefore, I would like to say thank you to:

My Parents, thank you for raising me up, and giving me everything.

My advisor Dr. Vijayan Asari and the Vision Lab crew, who gave me a lot of advice, ideas and encouragements during my research, and guided me to success. I cannot forget Dr. Vijayan Asari’s words: “You are not going to try it, you are going to make it.”

My friends, who listened to my worries and cheered me on to face the challenges.
# TABLE OF CONTENTS

ABSTRACT .................................................................................................................................iv

ACKNOWLEDGEMENTS ..............................................................................................................vi

LIST OF FIGURES ......................................................................................................................ix

LIST OF TABLES ......................................................................................................................xi

NOMENCLATURE ...................................................................................................................xiii

CHAPTER 1: INTRODUCTION ..............................................................................................1

1.1 Specific Objectives ...........................................................................................................5

1.2 Overview ...........................................................................................................................6

CHAPTER 2: BACKGROUND AND RELATED WORK .........................................................7

2.1 Background of Convolutional Neural Network ..............................................................7

2.1.1 LeNet5 ........................................................................................................................7

2.1.2 Dan Ciresan Net and AlexNet ...................................................................................9

2.2 Related Work of Classification of Multi-view Images of Objects ..............................10

2.3 Image Enhancement .......................................................................................................11

CHAPTER 3: CONVOLUTIONAL NEURAL NETWORK .......................................................12

3.1 Convolutional Layer ......................................................................................................12

3.2 Pooling Layer .................................................................................................................17

3.3 Rectified Linear Unit (ReLU) Layer ..............................................................................22
CHAPTER 4: PROPOSED STRUCTURE OF CONVOLUTIONAL NEURAL NETWORK .......................................................... 29

CHAPTER 5: IMAGE ENHANCEMENT .......................................................... 33
  5.1 Multilevel Windowed Inverse Sigmoid Function (MWIS) ................. 33
    5.1.1 MWIS Part 1: Adaptive Intensity Enhancement ...................... 33
    5.1.2 MWIS Part 2: Contrast Enhancement ..................................... 37
    5.1.3 MWIS Part 3: Color Restoration ............................................ 37
  5.2 Locally Tuned Sine Nonlinearity Technique (LTSN) ......................... 38
    5.2.1 LTSN Part 1: Adaptive Intensity Enhancement ...................... 38
    5.2.2 LTSN Part 2: Contrast Enhancement ..................................... 40
    5.2.3 LTSN Part 3: Color Restoration ............................................ 40
  5.3 Results of Enhancement .................................................................. 41

CHAPTER 6: EXPERIMENTS AND RESULTS ............................................. 43
  6.1 Results of Coil-100 Dataset ........................................................... 43
  6.2 Results of Multi-view Car Dataset .................................................. 47
  6.3 Analysis and Discussion .................................................................. 51

CHAPTER 7: CONCLUSION AND FUTURE WORKS ................................. 55

BIBLIOGRAPHY .................................................................................. 57
LIST OF FIGURES

Figure 1.1: Object images of Coil-100 dataset ..............................................................3
Figure 1.2: Camera azimuth of Coil-100 dataset ..............................................................3
Figure 1.3: All images of Object 10 ..................................................................................4
Figure 1.4: Images of Multi-view Car dataset .................................................................4
Figure 1.5: Example images of car 1 of Multi-view Car dataset ......................................5
Figure 2.1: LeNet 5 ........................................................................................................8
Figure 2.2: AlexNet .........................................................................................................10
Figure 3.1: (a) a 4 x 4 convolutional window with stride 1. (b) a 4 x 4 convolutional window with stride 2 .................................................................13
Figure 3.2: An example of down-sampling of forwarding of convolution with stride 2 ......................................................................................................................15
Figure 3.3: Example for rotating 180 degree .................................................................16
Figure 3.4: An example of up-sampling of backward of convolution with stride 2 ........................................................................................................................17
Figure 3.5: Example of Max pooling and Average pooling ............................................19
Figure 3.6: (a) 4 x 4 pooling window with stride 4. (b) 4 x 4 pooling window with stride 2 ..........................................................................................................................20
Figure 3.7: Example to show the error back propagation of Max pooling layer ......21
Figure 3.8: Example of back propagation of Average pooling .................................21
Figure 3.9: Example of ReLu and its back propagation .........................................23
Figure 3.10: Fully-connected layers .......................................................................24
Figure 4.1: Proposed structure of convolutional neural network ...........................29
Figure 5.1: Original image, MWIS adaptive intensity enhanced image, MWIS
correspondence enhanced image, and MWIS color restored image of object 1 of
Coil-100 dataset ........................................................................................................41
Figure 5.2: Original image, MWIS adaptive intensity enhanced image, MWIS
correspondence enhanced image, and MWIS color restored image of car 2 of
Multi-view Car dataset .............................................................................................41
Figure 5.3: Original image, LTSN adaptive intensity enhanced image, LTSN
correspondence enhanced image, and LTSN color restored image of object 1 of
Coil-100 dataset ........................................................................................................42
Figure 5.4: Original image, LTSN adaptive intensity enhanced image, LTSN
correspondence enhanced image, and LTSN color restored image of car 2 of
Multi-view Car dataset .............................................................................................42
Figure 6.1: The cost curve of one training process ....................................................44
LIST OF TABLES

Table 2.1: The connection table of S2 layer and C3 layer of LeNet5 ..................8

Table 4.1: Layers and sizes of each stage of convolutional neural network ............30

Table 6.1: Testing results of 8 view images for training of Coil-100 dataset ..........45

Table 6.2: Testing results of 12 view images for training of Coil-100 dataset ..........45

Table 6.3: Testing results of 18 view images for training of Coil-100 dataset ..........46

Table 6.4: Testing results of 24 view images for training of Coil-100 dataset ..........47

Table 6.5: Testing results of 36 view images for training of Coil-100 dataset ..........47

Table 6.6: Testing results of 8 view images for training of Multi-view Car dataset .........................................................................................................................................48

Table 6.7: Testing results of 12 view images for training of Multi-view Car dataset .........................................................................................................................................49

Table 6.8: Testing results of 18 view images for training of Multi-view Car dataset .........................................................................................................................................50

Table 6.9: Testing results of 24 view images for training of Multi-view Car dataset .........................................................................................................................................50

Table 6.10: Testing results of 36 view images for training of Multi-view Car dataset ........................................................................................................................................51

Table 6.11: The average error rate of Coil-100 dataset in ascending order ..........52
Table 6.12: The average error rate of Multi-view Car dataset in ascending order

<table>
<thead>
<tr>
<th>Order</th>
<th>Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>52</td>
</tr>
</tbody>
</table>
NOMENCLATURE

\( W \) Size of Input of Convolutional Layer
\( F \) Size of filter
\( P \) Number of Zero Padding
\( S \) Stride
\( X_k \) Input of the \( k \)th Stage
\( \Omega_{k,l} \) Weights of The \( l \)th Layer of \( k \)th Stage
\( T_{k,l} \) Bias of The \( l \)th Layer of \( k \)th Stage
\( Y_{k,l} \) Output of The \( l \)th Layer of \( k \)th Stage, Input of The \( l+1 \)th Layer of \( k \)th Stage
\( y \) Single Element of Matrix \( Y \)
\( z \) Output of Softmax Function
\( t \) Excepted Output
\( E_{k,l} \) Error
\( \Delta \) Difference, Error from Previous Layer
\( d_{in} \) Number of Input of Each Filter
\( d_{out} \) Number of Output of Each Filter
\( r \) Red Component of Color Image
\( g \) Green Component of Color Image
\( b \) Blue Component of Color Image
$G$  Gaussian Function

c  Scale, Gaussian Surround Space Constant

$I$  Grayscale Image

$L$  Illumination

$\alpha, \beta$  Curve Adjustment Parameter of MWIS transfer function

$M,N$  Size of The Grayscale Image

$m,n$  Size of Sub-Image

$V$  Contract Enhancement Grayscale Image

$\theta$  Ratio of Surrounding Intensity

$q$  Locally Adaptive Tunable Parameter

$u$  Constant for Computing Locally Adaptive Tunable Parameter

$\lambda$  Constant to Adjust Color Restoration
CHAPTER 1
INTRODUCTION

In recent years, neural networks have become more and more popular because of their outstanding performance in the object classification area. The convolutional neural network (CNN) \([1, 2, 3, 4]\) is a deep learning, feed-forward neural network that has excellent performance in visual imagery analysis area. The idea of the connectivity pattern between neurons of the CNN came from the organization of the animal visual cortex \([5, 20]\). For human vision, different observational directions of objects can get different views. Human can easily recognize objects in different observational directions, but machines cannot achieve this easily. Therefore, multi-view object classification have been researched for many years. To solve this problem, we apply CNN to do classification of the multi-view images of objects. Then, we try to improve the classification performance by adding image enhancement techniques before CNN. CNN collects features of the image. It is expected that the enhanced image exhibits stronger features, this is why we apply image enhancement techniques to the CNN to observe the effect. The training and testing input images of the CNN are original images or enhanced images, which are processed by a multilevel windowed inverse sigmoid (MWIS) function \([23]\) or a locally tuned sine nonlinearity (LTSN) technique \([24]\).
Therefore, the first goal of this thesis research work is classification of multi-view objects, and the second goal is comparison and analysis of the results of original input images and enhanced input images.

For the image datasets in this thesis research work, the first dataset is the Columbia Object Image Library (COIL-100) [6] that includes the RGB images of 100 objects. In this dataset, each object has 72 images, taken from 72 poses that are sampled every 5 degrees. The second dataset is the multi-view car dataset which includes 2299 multi-view images of 20 cars. The images of this dataset are not sampled in a fixed angle; therefore, the recorded angles of images of the multi-view car dataset are approximated in the results. The following figures show some images or information of these two datasets. Figure 1.1 shows the images for every objects of Coil-100 dataset, Figure 1.2 is the camera azimuth of Coil-100 dataset, Figure 1.3 shows all images of Object 10 in COIL-100 dataset, Figure 1.4 shows the images of all cars in the Multi-view Car dataset [26], and Figure 1.5 shows some example images of car 1 in Multi-view Car dataset. For each experiment, only a part of each dataset are used as training images, and rest are used for testing.
Figure 1.1: Object images of Coil-100 dataset

Figure 1.2: Camera azimuth of Coil-100 dataset
Figure 1.3: All images of Object 10

Figure 1.4: Images of Multi-view Car dataset
1.1 Specific Objectives

1. Perform a detailed literature survey on CNN.

2. Design and program a CNN, then, test its performance for multi-view objects (Coil-100 and multi-view car dataset).

3. Modify the CNN architecture with different layer-connecting strategies.

4. Select the most appropriate filter size and stride in the convolutional and pooling layers by conducting different experiments.

5. Test the performance of the modified CNN architecture with Coil-100 and multi-view car datasets.
6. Investigate various non-linear image enhancement algorithms for improvement of features in the imagery.

7. Apply the enhancement algorithms and test the performance of modified CNN with enhanced images.

1.2 Overview

In chapter 2, we introduce the background and history about CNN and some research on multi-view object classification and image enhancement. In chapter 3, the functions, structures, forward, back propagation process of each layer of CNN and initialization of training are included. Chapter 4 presents the detailed structure of CNN. In chapter 5, the process of MWIS and LTSN are introduced. Chapter 6 includes experiments of the CNN by using different numbers and types of images to train and their testing results. In the final chapter, we present a conclusion about this research and also a plan for work in the feature.
CHAPTER 2
BACKGROUND AND RELATED WORK

In this chapter, the background and related work of the convolutional neural network, multi-view object classification, and image enhancement will be introduced.

2.1 Background of Convolutional Neural Network

This part includes the development of the convolutional neural network [7] and its basic structure. The LeNe5 [1], Dan Ciresan Net [8] and Alex net will be introduced in following sub-parts.

2.1.1 LeNet5

In 1990s, Yann LeCun created one of the first convolutional neural networks. This network was named LeNet5 [1] which is shown in Figure 2.1. The LeNet5 was created for hand-writing numbers classification. This network has 7 layers, and the inputs are 32 × 32 zero-padded grayscale images of hand-writing digits from MNIST dataset.
Figure 2.1: LeNet5

The first layer C1 is a convolutional layer, that has 6 kernels of $5 \times 5$ filters, and each $5 \times 5$ filter is connected with the input layer. This means the outputs of the C1 layer are six $28 \times 28$ feature maps of the input image. The second layer S2 is a pooling layer. This layer does sub-sampling of feature maps from C1. The size of the sub-sampling window is $2 \times 2$ and the stride is 2, which means each $28 \times 28$ feature map is sub-sampling to $14 \times 14$ feature map. The third layer C3 is a non-complete connection convolution layer, and has 16 feature maps. Each C3 feature map is connected with several feature maps of S2 through several $5 \times 5$ filters. Table 2.1 shows the connection. The output feature map size is $10 \times 10$.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The fourth layer S4 is also a pooling layer. This layer does sub-sampling of feature maps from C3. The size of the sub-sampling window is $2 \times 2$ and the stride is 2, which means each $10 \times 10$ feature map is sub-sampling to a $5 \times 5$ feature map. The fifth layer, F5, is a complete connection layer. Each one of this layer’s 120 feature maps is connected with every feature map of S4 through $5 \times 5$ filters, and the output feature map size is $1 \times 1$. The sixth layer, F6, is a fully-connected layer that has 84 units, and each unit is connected with every feature map of C5. The last layer is the output layer, which includes 10 Euclidean Radial Basis Function units (RBF). Each unit is connected with every unit of F6.

### 2.1.2 Dan Ciresan Net and AlexNet

From 1998 to 2010, the ability of neural network was not noticed by most people. But in 2010, the publication of Dan Ciresan let people realize the power of the neural network [8]. The Dan Ciresan Net has 9 layers and is based on a NVIDIA GTX 280 graphic processor.

In 2012, a deep convolution neural network named AlexNet [9] and based on GPUs NVIDIA GTX 580 was created by Alex Krizhevsky. Alex applied Rectified linear units, dropout technique, and overlapping pooling to this network. Figure 2.2 shows the structure of AlexNet.
AlexNet uses GPU to increase the speed of computation and learn much more complex images. The success of AlexNet pushed the development of the convolutional neural network.

2.2 Related Work of Classification of Multi-view Images of Objects

For related work, there are some publications of classification of multi-view objects by using convolutional neural network or other methods.

In 2013, H.B. Kekre, T. K. Sarode, and J. K. Save used independent principal component analysis [10], which is a modified PCA, to achieve the classification of multi-view objects. Then, in 2015, application of deep convolutional wavelets network [11] that is published by S. Hassairi, R. Ejbali and M. Zaied also had a good performance on multi-view objects classification.
2.3 Image Enhancement

The image enhancement methods in this work are MWIS [23] and LTSN [24]. In fact, we also applied Self-tunable transformation function (STTF) [25] to the input images, the enhancement is good, but it also brings noise to some images, this noise affect the training process, therefore, we do not select STTF as the enhancement method.

The MWIS technique is published by K. Vijayan Asari, Ender Oguslu, and Saibabu Arigela in 2006. The LTSN is published by Saibabu Arigela and K. Vijayan Asari in 2008. The processes of both enhancements include three steps, adaptive intensity enhancement, contrast enhancement and color restoration. We will introduce the detail steps of MWIS and LTSN in chapter 5.
CHAPTER 3
CONVOLUTIONAL NEURAL NETWORK

The CNN for multi-view object classification includes Convolution layers, Pooling layers, Rectified linear unit (ReLU) Layers, Fully-connected layers and Softmax layer. Each layer has its function, position, connection forward and back propagation process in this neural network. This chapter focuses on layers’ function, forward, back propagation process and initialization. The position and connection of layers are introduced in chapter 4.

3.1 Convolutional Layer

The input is applied to convolution computation when passing the convolutional layer [9, 12]. The basic 2-dimensional convolution for Convolutional layer are

\[ Y_{k,l}(j,i) = X_k \ast \Omega_{k,l} + T_{k,l} = \sum_{p} \sum_{q} X_k(p,q) \Omega_{k,l}(j-p, i-q) + T_{k,l} \quad (3.1) \]

where \( X_k \) is the input, \( Y_{k,l} \) is output, and \( T_{k,l} \) is bias. This process sums all the values of points within the 2-dimensional convolutional window and bias. After that, the convolutional window is shifted one point and does the summation process again.
The $k$ means $k$th stage, and $l$ means $l$th layer. For example, $X_2$ is input of the stage 2, $Y_{2,1}$ is output of the first layer of stage 2 and also input of the second layer of stage 2. Then, the output of convolution need to pass an activation function:

$$Y_{k,l+1} = f(Y_{k,l})$$ (3.2)

However, the above convolution computation has a fixed stride. We apply convolution with a variable stride rather than convolution with a fixed stride. The stride determine the shifting distance for convolutional window. Figure 3.1 shows the difference between convolution with a stride one and convolution with a stride 2. The red areas are convolutional windows.

Figure 3.1: (a) a $4 \times 4$ convolutional window with stride 1. (b) a $4 \times 4$ convolutional window with stride 2

The adjusting of convolutional stride changes the size of the output feature map, the output size for normal 2- dimensional convolution is
\[
\text{size of output} = W - F + 1 \\
\]

(3.3)

where \(W\) is the original size of input, \(F\) is the size of convolutional windows. Then, the size for 2-dimensional convolution with variable stride is

\[
\text{size of output} = \frac{W - F + 2P}{S} + 1 \\
\]

(3.4)

where \(W\) is the original size of input, \(F\) is the size of convolutional windows, \(P\) is the number of zero padding, and \(S\) is the stride. The zero padding is to make sure the size of output is an integer. For example, if the size of input is \(96 \times 96\), the size of window is \(10 \times 10\), and stride is 3, then the height and width of output are 29.667 without zero-padding. Obviously, this size is not reasonable, therefore, it is necessary to apply a zero-padding process to this convolutional computation. In order to make sure the size is an integer, the \(P\) can be set to 2. Then, the height and width of output are 31.

After the convolution with the variable stride is introduced, how can it be achieved in program? In fact, convolution with variable stride, which is similar to pooling that will be introduced in following part of this chapter, is also a method to reduce the size of feature maps. To achieve convolution with variable stride, we can let the convolutional window move setting stride each time or separate the convolution process to two sub-processes. The two sub-processes are convolution with a stride of 1, and corresponding pooling. The equation of convolution with a stride of 1 has been shown in equation 3.1, so what is the corresponding pooling? Figure 3.2 shows an
example of down-sampling of forwarding of convolution with stride 2. Part (a) of figure 3.2 is the output of convolution with a stride of 1 and also the input of corresponding down-sampling, part (b) is the output of corresponding down-sampling.

Figure 3.2: An example of down-sampling of forwarding of convolution with stride 2

The above part introduced forwarding process of the Convolutional layer. Next, we move to back propagation [28] part. The back propagation process updates the weights and bias, and also propagate errors to previous layer. First, update the weights, the gradient of weights can be defined as:

$$\frac{\partial E_{k,l+1}}{\partial \Omega_{k,l}} = \frac{\partial E_{k,l+1}}{\partial Y_{k,l+1}} \cdot \frac{\partial Y_{k,l+1}}{\partial Y_{k,l}} \cdot \frac{\partial Y_{k,l}}{\partial \Omega_{k,l}} = rot_{180^\circ}(X_k) * [\Delta_{k,l+1} \cdot f'(Y_{k,l})] \quad (3.5)$$

Second, update the bias, the gradient of bias is defined as:
Third, the back propagation of errors is defined as:

\[
\frac{\partial E_{k,l+1}}{\partial X_k} = \frac{\partial E_{k,l+1}}{\partial Y_{k,l+1}} \frac{\partial Y_{k,l+1}}{\partial Y_{k,l}} \frac{\partial Y_{k,l}}{\partial X_k} = \Delta_{k,l+1} \cdot f'(Y_{k,l}) \cdot \text{rot}_{180} \{ \Omega_{k,l} \} \tag{3.7}
\]

In equation 3.5 to 7, \( X_k \) is the input of convolutional layer, \( \Omega_{k,l} \) is the weights of the filter, \( Y_{k,l} \) is output of convolutional layer, \( Y_{k,l+1} \) is the output of activation function, \( E_{k,l+1} \) is the error, \( \Delta_{k,l+1} \) is the difference from previous layer, \( T_{k,l} \) is the bias, and \( f'(\cdot) \) is the differential of the activation function \( f(\cdot) \). The activation function will be introduced in following section of this chapter. Then, equation 3.5 & 3.7 also include the rotation process, its example is shown in Figure 3.3.

![Figure 3.3: Example for rotating 180 degree](image)
The above backpropagation process is applied to convolution with stride 1. Then, if the stride becomes variable, an important step, which is the backpropagation of down-sampling or it can also be called the corresponding up-sampling, should be added before the error backpropagation. The example of corresponding up-sampling is shown in Figure 3.4.

![Figure 3.4: An example of up-sampling of backward of convolution with stride 2](image_url)

### 3.2 Pooling Layer

The Pooling layer [16] is a type of non-linear sub-sampling layer. This layer combines neuron cluster to a single neuron. There are several types of pooling layers, we only describe two type of pooling layers in this part: Max pooling and Average pooling. Max pooling, which chooses the maximum value within the pooling window as output, is defined as:
\[ Y_{k,l+2}(j,i) = \max \{ Y_{k,l+1}(\rho_1(j-1) + 1, \rho_2(i-1) + 1), Y_{k,l+1}(\rho_1(j-1) + 2, \rho_2(i-1) + 1), Y_{k,l+1}(\rho_1(j-1) + 1, \rho_2(i-1) + 2), \ldots, Y_{k,l+1}(\rho_1(j-1) + \rho_1, \rho_2(i-1) + \rho_2) \} \] (3.8)

Average pooling, which computes the Average value within the pooling window as output, is defined as:

\[ Y_{k,l+2}(j,i) = \frac{1}{\rho_1 \times \rho_2} \left( Y_{k,l+1}(\rho_1(j-1) + 1, \rho_2(i-1) + 1) + Y_{k,l+1}(\rho_1(j-1) + 2, \rho_2(i-1) + 1) + Y_{k,l+1}(\rho_1(j-1) + 1, \rho_2(i-1) + 2) + \ldots + Y_{k,l+1}(\rho_1(j-1) + \rho_1, \rho_2(i-1) + \rho_2) \right) \] (3.9)

In equation 3.8&9, \( Y_{k,l+1} \) is the output of activation function and also input of pooling layer, \( Y_{k,l+2} \) is the output of pooling layer, \( \rho_1 \) and \( \rho_2 \) is the size of pooling window.

Figure 3.5 is an example of Max pooling and Average pooling. The numbers in the pooling window are 12, 21, 30 and 9. The output of Max pooling is 30, which is the maximum value of these numbers, and the output of Average pooling is 18, which is the average value of these numbers.
Similar with convolution window, the pooling window also needs to be shifted according to the set stride. Therefore, we have two kinds of pooling: overlapping [12, 13, 14] and non-overlapping pooling. The pooling, which is overlapped or not, is determined by the size of the filter and shifting stride. In this thesis work, we choose to use overlapping max pooling as the pooling layer. Figure 3.6 shows the difference between overlapping pooling and non-overlapping pooling. There is no overlapping area for (a) 4 × 4 pooling window with stride 4, but (b) 4 × 4 pooling window with stride 2 has overlapping area.
After the introduction of the forwarding process of the pooling layer, we move to the backpropagation part. For Max pooling, the error only goes back to the position of the recorded maximum unit of forwarding process. The example is shown in Figure 3.7, the size of assuming feature before pooling map is $4 \times 4$, the size of pooling window is $2 \times 2$ and the stride is 2. The top part of Figure 3.7 is the forwarding process of max pooling, and the bottom part of Figure 3.7 is the backpropagation of max pooling. The red areas are assumed maximum units. The error only goes back to the position of recorded maximum units, and others are zeros. For the backpropagation of an overlapping max pooling, it is similar with non-overlapping pooling. The only difference is that multi errors may go back to one corresponding position. At this time, just sum the multi errors.
For the back propagation of average pooling, the error is divided by the size of pooling windows, then it goes back to whole pooling area, which means all units obtain the same value. The example of back propagation of Average pooling is shown in Figure 3.8, where the size of input is $4 \times 4$, and the pooling window has the size of $2 \times 2$ and the stride is 2.
3.3 Rectified Linear Unit (ReLU) Layer

Rectified linear unit (ReLu) [15] is a unit with activation function defined as

\[ y_{k,l+1} = f(y_{k,l}) = \begin{cases} 
  y_{k,l} & y_{k,l} \geq 0 \\
  0 & y_{k,l} < 0 
\end{cases} \]  \hfill (3.10)

In this function, the output is equal to input if input \( y_{k,l} \) is larger than or equal to zero. The output is equal to zero if input \( y_{k,l} \) is smaller than zero. The advantages of the rectified linear unit are sparse activation, efficient gradient propagation, and efficient computation. When the input passes the rectified linear unit layer, all the values which are smaller than zero become zeros, therefore, the output has sparse activation and become easy to compute. Then, comparing rectified function with other functions such as sigmoid function:

\[ \text{sigmf}(y_{k,l}) = \frac{1}{1 + e^{-y_{k,l}}} \]  \hfill (3.11)

where \( y_{k,l} \) is input. Sigmoid function kills the gradients, while the rectified linear unit has no vanishing gradient problem.

After the forwarding process, for the backpropagation of ReLu, the differential is defined as:

\[ f'(y_{k,l}) = \begin{cases} 
  1 & y_{k,l} \geq 0 \\
  0 & y_{k,l} < 0 
\end{cases} \]  \hfill (3.12)
where $y_{k,t}$ is input value of forwarding process. According to equation 3.12, the error, that has a corresponding input value of forwarding process is larger than or equal to zero, will keep its value; and the error, that has a corresponding input value of forward process is smaller than zero, will become zero. The example is shown in Figure 3.9, the top part is a forward process of ReLu unit, and the bottom part is a backpropagation process, the gradient is killed when its corresponding input value in forwarding process is smaller than zero.

Figure 3.9: Example of ReLu and its back propagation
### 3.4 Fully-connected Layer

The position of fully-connected layers is behind convolutional layers and pooling layers. Figure 3.10 shows fully-connected layers, where each neuron of the fully-connected layer is connected with every neuron of the previous layer. This design used Multi-layer Perception [18, 19].

![Fully-connected layers diagram](image)

**Figure 3.10:** Fully-connected layers

For Fully-connected layers, one layer is defined as:

\[
Y_{k,l} = \Omega_{k,l} \times X_k + T_{k,l}
\]  

\[
Y_{k,l+1} = f(Y_{k,l})
\]  

(3.13)  

(3.14)
where \( X_k \) is the input, \( Y_{k,l} \) is the computing results of the network and also inputs of activation function, \( T_{k,l} \) is the bias, \( \Omega_{k,l} \) is the weights, \( f() \) is the activation function, and \( Y_{k,l+1} \) is the output of activation function. Then, for the backpropagation of fully-connected layer, the error propagations is defined as:

\[
\frac{\partial E_{k,l+1}}{\partial X_k} = \frac{\partial E_{k,l+1}}{\partial Y_{k,l+1}} \frac{\partial Y_{k,l+1}}{\partial Y_{k,l}} \frac{\partial Y_{k,l}}{\partial X_k} = \Delta_{k,l+1} \cdot f'(Y_{k,l}) \times \Omega_{k,l} \tag{3.15}
\]

The updating weights is defined as:

\[
\frac{\partial E_{k,l+1}}{\partial \Omega_{k,l}} = \frac{\partial E_{k,l+1}}{\partial Y_{k,l+1}} \frac{\partial Y_{k,l+1}}{\partial Y_{k,l}} \frac{\partial Y_{k,l}}{\partial \Omega_{k,l}} = \Delta_{k,l+1} \cdot f'(Y_{k,l}) \times X_k \tag{3.16}
\]

Then, the updating bias is defined as:

\[
\frac{\partial E_{k,l+1}}{\partial T_{k,l}} = \frac{\partial E_{k,l+1}}{\partial Y_{k,l+1}} \frac{\partial Y_{k,l+1}}{\partial Y_{k,l}} \frac{\partial Y_{k,l}}{\partial T_{k,l}} = \Delta_{k,l+1} \times f'(Y_{k,l}) \tag{3.17}
\]

In equation 3.15 to 17, \( X_k \) is the input of network, \( \Omega_{k,l} \) is the weights, \( Y_{k,l} \) is the output of network and also the input of activation function, \( Y_{k,l+1} \) is the output of activation function, \( T_{k,l} \) is the bias, \( E_{k,l+1} \) is the error, \( \Delta_{k,l+1} \) is the difference from previous layer, and \( f'(\cdot) \) is the differential of the activation function.
3.5 Softmax Layer

Softmax layer [17], which can transform the input vector into the output vector of values in the range [0, 1], is used for multiclass logistic regression and defined as:

\[ z_i = \frac{e^{y_{last}^i}}{\sum_{j \in group} e^{y_{last}^j}} \]  \tag{3.18}

where \(y_{last}^i\) is one element of the computing output of the last fully-connected layer and also the input of the softmax layer, and \(z\) is output of softmax function. The summation of output of Softmax function is 1. In fact, it gives a score to each class, and the classification layer use these scores to do the classification. Then, the cross-entropy error function is

\[ E = -\sum_i t_i \log (z_i) \]  \tag{3.19}

where \(t\) is expected output, and \(z\) is output of softmax function. Then, the gradient of backpropagation is obtained by:

\[ \frac{\partial E}{\partial z_i} = -\frac{t_i}{z_i} \]  \tag{3.20}

\[ \frac{\partial z_i}{\partial Y_k} = \begin{cases} z_i(1 - z_i) & i = k \\ -z_i z_k & i \neq k \end{cases} \]  \tag{3.21}
\[ \frac{\partial E}{\partial Y_i} = z_i - t_i \]  

(3.22)

where \( y_{last_i} \) is one element of the computing output of the last fully-connected layer and also the input of the softmax layer, \( z \) is output of softmax function, \( E \) is error, and \( t \) is expected output.

### 3.6 Initialization of Training

A good beginning is half done. To make sure the training process of convolution neural network is successful, the initialization of training is very important.

For initialization of weights of the convolutional neural network, we recommend using the strategy of Xavier Glorot and Yoshua Bengio [22]. The initialized weights of convolutional layers can use the uniform distribution with the range of

\[
\left[ -\frac{\sqrt{6}}{\sqrt{d_{in} + d_{out}}}, +\frac{\sqrt{6}}{\sqrt{d_{in} + d_{out}}} \right]
\]

The \( d_{in} \) is the number of input of each filter, \( d_{out} \) is the number of output of each filter. Then, the initialized weights of fully-connected layers can use the normal distribution with zero mean and standard deviation of

\[
\sqrt{\frac{2}{d_{in} + d_{out}}}
\]
or other small values. Then, the biases can be initialized to zeros or small constant numbers. In fact, unlike sigmoid function, the value of output of ReLu unit has no boundary, a large weight would cause too large of a gradient, and a large input of Softmax function would cause computation problems. For example, if the input of Softmax function is [1000 3000 8000], the $e^{1000}$, $e^{3000}$ and $e^{8000}$ are too large to be computed.
CHAPTER 4

PROPOSED STRUCTURE OF CONVOLUTIONAL NEURAL NETWORK

After the introduction and explanation of every layer in the CNN, this chapter will show our proposed CNN structure. We designed many CNN structures based on theories and experiments. The proposed CNN that was designed after many times of modifying and testing can get the highest accuracy when compared with other versions which was designed. Figure 4.1 shows the 6 stages of our CNN, the inside layers and their size of each stage are shown in Table 4.1.

Figure 4.1: Proposed structure of convolutional neural network
Table 4.1: Layers and sizes of each stage of convolutional neural network

<table>
<thead>
<tr>
<th>Input Stage A</th>
<th>C&amp;P Stage B</th>
<th>C&amp;P Stage C</th>
<th>C Stage D</th>
<th>F Stage E</th>
<th>CF Stage F</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1:128×128×3</td>
<td>C2:6×6×3×6</td>
<td>C5:4×4×64×256</td>
<td>C8:6×6×256×1024</td>
<td>F10:1024 nodes</td>
<td>CF 16</td>
</tr>
<tr>
<td></td>
<td>4 Stride: 2</td>
<td>2 Stride: 2</td>
<td>1 Stride: 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R3</td>
<td>R6</td>
<td>R9</td>
<td>R11</td>
<td>F12:1024 nodes</td>
<td></td>
</tr>
<tr>
<td>P4: 4×4</td>
<td>P7: 4×4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stride: 2</td>
<td>Stride: 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R13</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R14:100 nodes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S15</td>
<td></td>
</tr>
</tbody>
</table>

The proposed structure of the CNN can be described as below:

(a) The First layer of convolutional neural network is Input layer 1 (I1). The size is 128 × 128 × 3, and the input image are 128 × 128 RGB images of objects.

(b) The second layer is Convolutional layer 2 (C2). This layer includes 64 kernels of 6 × 6 × 3 filters with stride 2. Each 6 × 6 × 3 filter can be seen as a filter with three 6 × 6 sub-filters, and each sub-filter is connected with corresponding sub-input layers (The input layer can also be seen as an input with three 128 × 128 sub-inputs, and they correspond with sub-filters in order). The outputs of this layer are 64 feature maps with size of 62 × 62.

(c) The third layer is ReLu layer 3 (R3), ReLu function keeps all the input values that are larger than or equal to zero, and transform all the input values that are
smaller than zero into zeros.

(d) The fourth layer is max pooling layer 4 (P4). The layer includes a $4 \times 4$ max pooling window with stride 2. The outputs of this layer are 64 feature map with size of $30 \times 30$.

(e) The fifth layer is convolutional layer 5 (C5). This layer includes 256 kernels of $4 \times 4 \times 64$ filters with stride 2. Similar to convolutional layer 2, every sub-filter of each filter is connected with corresponding feature maps from Max pooling layer 1. The outputs of this layer are 256 feature maps with size of $14 \times 14$.

(f) The sixth layer is ReLu layer 6 (R6), ReLu function keeps all the input values that are larger than or equal to zero, and transform all the input values that are smaller than zero into zeros.

(g) The seventh layer is max pooling layer 7 (P7), this layer includes a $4 \times 4$ max pooling window with stride 2. The outputs of this layer are 256 feature map with size of $6 \times 6$.

(h) The eighth layer is convolutional layer 8 (C8). This layer includes 1024 kernels of $6 \times 6 \times 256$ filters with stride 1. Similar to convolutional layer 2 & 5, every sub-filter of each filter is connected with corresponding feature maps from Max pooling layer 7. The outputs of this layer are 1024 feature maps with size of $1 \times 1$.

(i) The ninth layer is ReLu layer 9 (R9), ReLu function keeps all the input values that are larger than or equal to zero, and transform all the input values that are smaller than zero into zeros.
(j) The tenth layer is Fully-connected layer 10 (F10). This layer include 1024 nodes, and each node is connected with all the feature maps from ReLu layer 9.

(k) The eleventh layer is ReLu layer 11 (R11), ReLu function keeps all the input values that are larger than or equal to zero, and transform all the input values that are smaller than zero into zeros.

(l) The twelfth layer is Fully-connected layer 12 (F12). This layer include 1024 nodes, and each node is connected with all the nodes from ReLu layer 11.

(m) The thirteenth layer is ReLu layer 13 (R13), ReLu function keeps all the input values that are larger than or equal to zero, and transform all the input values that are smaller than zero into zeros.

(n) The fourteenth layer is Fully-connected layer 14 (F14). This layer has 100 nodes for Coil-100 dataset and 20 nodes for multi-view car dataset, each node is connected with all nodes from ReLu layer 13.

(o) The fifteenth layer is Softmax layer 15 (S15). This layer computes the input vector from previous Fully-connected layer 14 by using the softmax function.

(p) The last layer is Classification layer 16 (CF16). This layer does the classification of images depending on the values of the output vector from the previous Softmax layer.

The testing and evaluation of this CNN structure is provided in chapter 6.
CHAPTER 5

IMAGE ENHANCEMENT

This chapter includes the process and results of the multilevel windowed inverse sigmoid (MWIS) function [23] and locally tuned sine nonlinearity (LTSN) technique [24] to enhance the image.

5.1 Multilevel Windowed Inverse Sigmoid Function (MWIS)

The process of MWIS [23] includes three parts: adaptive intensity enhancement, contrast enhancement, and color restoration. The three steps will be described in following sub-parts.

5.1.1 MWIS Part 1: Adaptive Intensity Enhancement

For the first step, use the NTSC method to convert RGB images to grayscale images:

\[ I(j,i) = 0.2989r(j,i) + 0.5870g(j,i) + 0.1140b(j,i) \quad (5.1) \]

The \( r, g \) and \( b \) here means red, green and blue Component of a color image. Then, the Illumination in an image can be defined as
\[ I(j, i) = R(j, i)L(j, i) \] (5.2)

\( L(j, i) \) is the illumination and \( R(j, i) \) is the reflectance, illumination is the low frequency components of an image. In order to get the illumination, we need to use a multi-scale Gaussian low-pass filter.

\[ G(j, i) = Ke^{-\frac{(j+i)^2}{c^2}}, \quad K = \sum_{j} \sum_{i} G(j, i) = 1 \] (5.3)

\[ L = I \ast G_i \] (5.4)

where the \( G(j, i) \) is a 2D Gaussian function, \( c \) is Gaussian surround space constant, and \( G_i \) is the multi-scale Gaussian filter. The multi-scale Gaussian function can avoid the problem of single scale. For a problem of single scale Gaussian, if \( c \) is smaller, it will cause a bad Global impression and halo artifacts, and a larger \( c \) causes unclear detail.

After illumination is obtained, the averaging weighted illumination can be defined as

\[ L_{ave}(j, i) = \frac{I(j, i) - 204}{51} L(j, i) + \left( 1 - \frac{I(j, i) - 204}{51} \right) L(j, i) \] (5.5)

The averaging weighted illumination process can avoid the effect of illumination of different light resource. The values of weights in equation 5.5 are obtained by illumination estimation (smaller than 80% of the maximum grayscale value). Then, the
dark illumination need to be enhanced and the bright illumination need to be compressed by using the MWIS transfer function

\[
L_n(j, i) = \frac{L_{\text{ave}}(j, i)}{25.5}
\]  

\[
L_{\text{ enh}} = \frac{1}{1 + e^{-\alpha \times L_n}} + \frac{1}{1 + e^{-\beta \times (L_n - 10)}} - 0.5
\]

In equation 5.7, the illumination’s value need to be normalized in to the range of 0 to 10 before enhancement by using equation 5.6. The inputs of 5.7 are sub-images of a normalized grayscale image, the size of sub-images are \( M_{\text{sub}} \times N_{\text{sub}} \). The values of \( M_{\text{sub}} \) and \( N_{\text{sub}} \) can be computed by

\[
M_{\text{sub}} = \frac{M}{16}, \quad N_{\text{sub}} = \frac{N}{16}
\]  

The \( M \) and \( N \) are the size of a grayscale image. The \( \alpha \) and \( \beta \) are curve adjustment parameters, their values can be obtained by

\[
\alpha = \begin{cases} 
\frac{76.5 - L_{m, \text{min}}}{51} & 0 \leq L_{m, \text{min}} \leq 51 \\
0.5 & 51 < L_{m, \text{min}} \leq 127
\end{cases}
\]
\[
\beta = \begin{cases} 
\frac{L_{m,max} - 255}{51} + 1.5 & 204 \leq L_{m,max} \leq 255 \\
0.5 & 128 < L_{m,max} \leq 204 
\end{cases} 
\] (5.10)

where the \( L_m \) is the mean of the sub-image, \( L_{m,min} \) is minimum mean value of sub-images, and \( L_{m,max} \) is maximum mean value of sub-images. Then, for special cases, if \( L_{m,min} \) is larger than 127, or \( L_{m,max} \) is less than 127, then, their \( \alpha \) and \( \beta \) are computed as:

\[
\alpha = \frac{127 - l_m}{63.5} + 1.5 \quad L_{m,min} > 127 
\] (5.11)

\[
\beta = \frac{l_m - 128}{63.5} + 1.5 \quad L_{m,max} < 127 
\] (5.12)

where the \( l_m \) is the image’s global mean. After the above process, the enhanced image can be obtained by

\[
l_{enh}(j,i) = L_{n,enh}(j,i)R(j,i) 
\] (5.13)

where \( L_{n,enh}(j,i) \) is the enhanced illumination and \( R(j,i) \) is the reflectance.
5.1.2 MWIS Part 2: Contrast Enhancement

The first step of contrast enhancement is to get intensity information of the surrounding pixels by using a multi-scale Gaussian filter

\[ I_f = I \ast G_i \]  

(5.14)

Then the ratio of surrounding intensity is defined as

\[ \theta(j, i) = \left( \frac{I_f(j, i)}{I(j, i)} \right)^h \]  

(5.15)

where \( h \) is used to tune this process. The results of contrast enhancement are

\[ V(j, i) = 255 \times l_{en}(j, i)^\theta(j, i) \]  

(5.16)

5.1.3 MWIS Part 3: Color Restoration

The result of MWIS is an enhanced grayscale image. In order to get the color back, the color restoration is defined as:

\[ V_\gamma(j, i) = V(j, i) \frac{I_f(j, i)}{I(j, i)} \lambda_\gamma \]  

(5.17)

In the above equation, \( \gamma \) is equal to \( r, g \) and \( b \). \( r \) means red component of image, \( g \) means green component of image, and \( b \) means blue component of image, and \( \lambda \) is
constant, its value is almost close to 1.

5.2 Locally Tuned Sine Nonlinearity Technique (LTSN)

The LTSN [24] also includes three parts, adaptive intensity enhancement, contrast enhancement, and color restoration. The three steps will be described in following sub-parts.

5.2.1 LTSN Part 1: Adaptive Intensity Enhancement

For the first step, we use NTSC method to convert RGB image to grayscale images

\[ I(j, i) = 0.2989r(j, i) + 0.5870g(j, i) + 0.1140b(j, i) \]  \hspace{1cm} (5.18)

where \( r \) means red component of image, \( g \) means green component of image, and \( b \) means blue component of image. Then, the image needs to be normalized to range of 0 to 1.

\[ I_n = \frac{I(j, i)}{255} \]  \hspace{1cm} (5.19)

After that, we use a non-linear transfer function to decrease the illumination value of high-illumination pixels, and increase illumination value of low-illumination pixels. This non-liner transfer function is defined as:
\[ I_e(j, i) = \sin^2 \left( I_n(j, i)^Q \times \frac{\pi}{2} \right) \] 

\[ Q = \tan \left( \frac{I_{Mn}(j, i) \times \pi}{u_1} \right) + u_2 \]

In the above equations, the \( I_n \) is the normalized image, \( Q \) is the locally adaptive tunable parameter, \( u_1 \) and \( u_2 \) are constants, \( u_1 \) has a value range of 2.1 to 2.4, \( u_2 \) has a value of 2.25. The value of \( u_1 \) and \( u_2 \) both come from their experiment. For a special case of extreme dark regions, when the mean value is smaller than 0.2, use another equation to compute the locally adaptive tunable parameter:

\[ Q = \frac{\log(2 \times I_{Mn}(j, i)) + 2}{2} \]

Then, \( I_{Mn} \) is the normalized \( I_M \), which means that the value of \( I_M \) is converted into range of 0 to 1, and \( I_M \) is obtained by

\[ I_M = I \ast G_i \]

\[ G(j, i) = Ke^{-\frac{(j+i)^2}{c^2}}, \quad K = \sum_j \sum_i G(j, i) = 1 \]

where the \( G(j, i) \) is a 2D Gaussian function, \( c \) is Gaussian surround space constant, and \( G_i \) is the multi-scale Gaussian filter. The multi-scale Gaussian function can avoid the
problem of single scale. For the problem of single scale Gaussian, if \( c \) is smaller, it will cause a bad Global impression and halo artifacts, and a larger \( c \) causes unclear detail.

### 5.2.2 LTSN Part 2: Contrast Enhancement

The center-surround contrast enhancement of LTSN is defined as

\[
V(j, i) = 255 \times I_{\text{enh}}(j, i)^{\theta(j, i)}
\]  

(5.25)

\[
\theta(j, i) = \left( \frac{l_M(j, i)}{I(j, i)} \right)^h
\]  

(5.26)

In above equations, \( V(x, y) \) is a contract enhancement grayscale image, \( \theta(j, i) \) is the ratio of surrounding intensity, \( h \) is used to tune the ratio of surrounding intensity.

### 5.2.3 LTSN Part 3: Color Restoration

The results of LTSN are an enhanced grayscale image. In order to get the color back, the color restoration is defined as:

\[
V_\gamma(j, i) = V(j, i) \frac{l_\gamma(j, i)}{I(j, i)} \lambda_\gamma
\]  

(5.27)

In the above equation, \( \gamma \) is equal to \( r, g \) and \( b \). \( r \) means red component of image, \( g \) means green component of image, \( b \) means blue component of image, and \( \lambda \) is constant. Its value is almost close to 1.
5.3 Results of Enhancement

As shown in Figure 5.1 to 4, there are Original images, and enhanced images of each step of image enhancement from the coil-100 dataset and multi-view car dataset.

Figure 5.1: Original image, MWIS adaptive intensity enhanced image, MWIS contrast enhanced image, and MWIS color restored image of object 1 of Coil-100 dataset

Figure 5.2: Original image, MWIS adaptive intensity enhanced image, MWIS contrast enhanced image, and MWIS color restored image of car 2 of Multi-view Car dataset
Figure 5.3: Original image, LTSN adaptive intensity enhanced image, LTSN contrast enhanced image, and LTSN color restored image of object 1 of Coil-100 dataset

Figure 5.4: Original image, LTSN adaptive intensity enhanced image, LTSN contrast enhanced image, and LTSN color restored image of car 2 of Multi-view Car dataset
CHAPTER 6
EXPERIMENTS AND RESULTS

This chapter 6 presents some experiments, results, and discussions for object classification of CNN that is introduced and described in chapters 3 & 4. The input images are original images, LTSN enhanced images and MWIS enhanced images.

6.1 Results of Coil-100 Dataset

The different experiments have different number of images for training and testing, and different type of input images. The first dataset used in this work is the Columbia Object Image Library (Coil-100) and includes the RGB image of 100 objects. Each object has 72 images taken from 72 poses and sampled every 5 degrees. For training and testing, we select the images based on the angle of the dataset. In order to make sure the angle differences of all training images are the same, we select training images in the sample angles of 45, 30, 20, 15, and 10 degrees. This means the numbers of training images of experiments are 8, 12, 18, 24, and 36, and the rest of the images in the dataset are testing images. Each of the groups of each experiment will include three error rates, which are the error rates for original images, MWIS enhanced images and LTSN enhanced images. Figure 6.1 shows the cost curve of one training process, the equation of cost is defined as:
\[ E = - \sum_i t_i \log(z_i) \] (6.1)

where the \( z \) is output of softmax layer, and \( t \) is the expected output.

Figure 6.1: The cost curve of one training process

For the evaluation of accuracy, the following Table 6.1 to 6.5 show the testing error rates of coil-100 dataset. For the first experiment, the number of training images is 8 and the results are shown in Table 6.1, 5 groups of LTSN image have the lowest error rate, and 2 groups of original image have the lowest error rate. The LTSN image has the lowest average error rate.
Table 6.1: Testing results of 8 view images for training of Coil-100 dataset

<table>
<thead>
<tr>
<th>Group Number</th>
<th>Angle of Training Image</th>
<th>Testing error rate (original image)</th>
<th>Testing error rate (MWIS)</th>
<th>Testing error rate (LSTN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5: 45: 320</td>
<td>7.64%</td>
<td>3.11%</td>
<td>2.44%</td>
</tr>
<tr>
<td>2</td>
<td>10: 45: 325</td>
<td><strong>3.75%</strong></td>
<td>5.66%</td>
<td>5.91%</td>
</tr>
<tr>
<td>3</td>
<td>15: 45: 330</td>
<td>4.06%</td>
<td>4.53%</td>
<td><strong>3.81%</strong></td>
</tr>
<tr>
<td>4</td>
<td>25: 45: 340</td>
<td>5.84%</td>
<td>5.81%</td>
<td><strong>5.12%</strong></td>
</tr>
<tr>
<td>5</td>
<td>30: 45: 345</td>
<td>7.55%</td>
<td>8.00%</td>
<td><strong>3.34%</strong></td>
</tr>
<tr>
<td>6</td>
<td>35: 45: 350</td>
<td>4.88%</td>
<td>4.70%</td>
<td><strong>2.81%</strong></td>
</tr>
<tr>
<td>7</td>
<td>40: 45: 355</td>
<td><strong>2.88%</strong></td>
<td>3.98%</td>
<td>3.67%</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td></td>
<td>5.23%</td>
<td>5.11%</td>
<td><strong>3.87%</strong></td>
</tr>
</tbody>
</table>

Table 6.2 shows the testing results of 12 view images for the training of Coil-100 dataset. 2 groups of MWIS image have the lowest error rate, original and LSTN image both have 1 lowest error rate group. MWIS image has the lowest average error rate.

Table 6.2: Testing results of 12 view images for training of Coil-100 dataset

<table>
<thead>
<tr>
<th>Group Number</th>
<th>Angle of Training Image</th>
<th>Testing error rate (original image)</th>
<th>Testing error rate (MWIS)</th>
<th>Testing error rate (LSTN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0: 30: 330</td>
<td>2.23%</td>
<td><strong>1.12%</strong></td>
<td>2.75%</td>
</tr>
<tr>
<td>2</td>
<td>5: 30: 335</td>
<td><strong>1.10%</strong></td>
<td>1.70%</td>
<td>1.13%</td>
</tr>
<tr>
<td>3</td>
<td>10: 30: 340</td>
<td>2.67%</td>
<td>1.95%</td>
<td><strong>1.80%</strong></td>
</tr>
<tr>
<td>4</td>
<td>25: 30: 355</td>
<td>2.18%</td>
<td><strong>1.42%</strong></td>
<td>1.98%</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td></td>
<td>2.04%</td>
<td><strong>1.55%</strong></td>
<td>1.92%</td>
</tr>
</tbody>
</table>
Table 6.3 shows the testing results of 18 view images for the training of Coil-100 dataset. 3 groups of LTSN image have the lowest error rate, and original images has 1 lowest error rate group. LTSN image has the lowest average error rate.

Table 6.3: Testing results of 18 view images for training of Coil-100 dataset

<table>
<thead>
<tr>
<th>Group Number</th>
<th>Angle of Training Image</th>
<th>Testing error rate (original image)</th>
<th>Testing error rate (MWIS)</th>
<th>Testing error rate (LTSN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0: 20: 340</td>
<td>0.50%</td>
<td>0.74%</td>
<td>0.33%</td>
</tr>
<tr>
<td>2</td>
<td>5: 20: 345</td>
<td>0.22%</td>
<td>0.31%</td>
<td>0.20%</td>
</tr>
<tr>
<td>3</td>
<td>10: 20: 350</td>
<td>0.35%</td>
<td>0.22%</td>
<td>0.20%</td>
</tr>
<tr>
<td>4</td>
<td>15: 20: 355</td>
<td><strong>0.31%</strong></td>
<td>0.35%</td>
<td>0.52%</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td></td>
<td>0.34%</td>
<td>0.40%</td>
<td><strong>0.31%</strong></td>
</tr>
</tbody>
</table>

Table 6.4 shows the testing results of 24 view images for the training of Coil-100 dataset. 2 groups of original image have the lowest error rate, and LTSN image has 1 lowest error rate group. Original image has the lowest average error rate.
Table 6.4: Testing results of 24 view images for training of Coil-100 dataset

<table>
<thead>
<tr>
<th>Group Number</th>
<th>Angle of Training Image</th>
<th>Testing error rate (original image)</th>
<th>Testing error rate (MWIS)</th>
<th>Testing error rate (LTSN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0: 15: 345</td>
<td>0.23%</td>
<td>0.27%</td>
<td>0.29%</td>
</tr>
<tr>
<td>2</td>
<td>5: 15: 350</td>
<td>0.10%</td>
<td>0.50%</td>
<td>0.35%</td>
</tr>
<tr>
<td>3</td>
<td>10: 15: 355</td>
<td>0.25%</td>
<td>0.27%</td>
<td>0.23%</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>0.19%</td>
<td>0.34%</td>
<td>0.29%</td>
</tr>
</tbody>
</table>

Table 6.5 shows the testing results of 36 view images for training of Coil-100 dataset. The original image has the best performance with 100% accuracy.

Table 6.5: Testing results of 36 view images for training of Coil-100 dataset

<table>
<thead>
<tr>
<th>Group Number</th>
<th>Angle of Training Image</th>
<th>Testing error rate (original image)</th>
<th>Testing error rate (MWIS)</th>
<th>Testing error rate (LTSN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0: 10: 350</td>
<td>0%</td>
<td>0%</td>
<td>0.21%</td>
</tr>
<tr>
<td>2</td>
<td>5: 10: 355</td>
<td>0%</td>
<td>0.03%</td>
<td>0.06%</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>0%</td>
<td>0.02%</td>
<td>0.14%</td>
</tr>
</tbody>
</table>

6.2 Results of Multi-view Car Dataset

Similar to the previous part, the different experiments have different number of images for training and testing, and different type of input images. The second dataset is used in this work is the Multi-view Car dataset, which includes 2299 multi-view images of 20 cars. However, the images of this dataset are not sampled in a fixed angle,
therefore, the recorded angles of images of the multi-view car dataset in this part are approximated angles. Then, the CNN structure needs to be adjusted to fit this dataset, the number of outputs of the last fully-connected layer is reduced to 20, the image of the multi-view car dataset also needs to be resized to fit the CNN.

For the evaluation of accuracy, the following tables 6.6 to 6.10 show the testing error rates of the Multi-view Car dataset. Table 6.6 shows testing results of 8 views images for training of Multi-view Car dataset. The MWIS image has the lowest error rate for 3 groups, original and LTSN image both have one group that has lowest error rate. MWIS image has the lowest average error rate.

Table 6.6: Testing results of 8 view images for training of Multi-view Car dataset

<table>
<thead>
<tr>
<th>Group Number</th>
<th>Approximated Angle of Training Image</th>
<th>Testing error rate (original image)</th>
<th>Testing error rate (MWIS)</th>
<th>Testing error rate (LSTN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10: 45: 325</td>
<td>6.17%</td>
<td>3.23%</td>
<td>4.02%</td>
</tr>
<tr>
<td>2</td>
<td>15: 45: 330</td>
<td>3.69%</td>
<td>6.17%</td>
<td>5.47%</td>
</tr>
<tr>
<td>3</td>
<td>25: 45: 340</td>
<td>5.10%</td>
<td>4.43%</td>
<td>1.45%</td>
</tr>
<tr>
<td>4</td>
<td>30: 45: 345</td>
<td>7.81%</td>
<td>4.07%</td>
<td>8.74%</td>
</tr>
<tr>
<td>5</td>
<td>35: 45: 350</td>
<td>10.14%</td>
<td>3.55%</td>
<td>6.36%</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td></td>
<td>6.58%</td>
<td><strong>4.29%</strong></td>
<td>5.21%</td>
</tr>
</tbody>
</table>

Table 6.7 shows the testing results of 12 views images for training of Multi-view Car dataset. The MWIS image has the lowest error rate for 2 groups, original and LTSN
both have one group that got lowest error rate. MWIS image has the lowest average error rate.

Table 6.7: Testing results of 12 view images for training of Multi-view Car dataset

<table>
<thead>
<tr>
<th>Group Number</th>
<th>Approximated Angle of Training Image</th>
<th>Testing error rate (original image)</th>
<th>Testing error rate (MWIS)</th>
<th>Testing error rate (LTSN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10: 30: 340</td>
<td>2.33%</td>
<td>3.45%</td>
<td>3.06%</td>
</tr>
<tr>
<td>2</td>
<td>15: 30: 345</td>
<td>0.68%</td>
<td>1.94%</td>
<td>0.34%</td>
</tr>
<tr>
<td>3</td>
<td>20: 30: 350</td>
<td>1.75%</td>
<td>0.73%</td>
<td>2.77%</td>
</tr>
<tr>
<td>4</td>
<td>25: 30: 355</td>
<td>6.31%</td>
<td>3.69%</td>
<td>4.32%</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td></td>
<td>2.76%</td>
<td><strong>2.45%</strong></td>
<td><strong>2.62%</strong></td>
</tr>
</tbody>
</table>

Table 6.8 shows the testing results of 18 views images for training of Multi-view Car dataset. The original image has the lowest error rate for 3 groups, MWIS has one group that has lowest error rate. Original image has the lowest average error rate.
Table 6.8: Testing results of 18 view images for training of Multi-view Car dataset

<table>
<thead>
<tr>
<th>Group Number</th>
<th>Approximated Angle of Training Image</th>
<th>Testing error rate (original image)</th>
<th>Testing error rate (MWIS)</th>
<th>Testing error rate (LTSN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0: 20: 340</td>
<td>1.03%</td>
<td><strong>0.62%</strong></td>
<td>0.93%</td>
</tr>
<tr>
<td>2</td>
<td>5: 20: 345</td>
<td><strong>0.21%</strong></td>
<td>0.67%</td>
<td>0.46%</td>
</tr>
<tr>
<td>3</td>
<td>10: 20: 350</td>
<td><strong>0.21%</strong></td>
<td>0.26%</td>
<td>0.31%</td>
</tr>
<tr>
<td>4</td>
<td>15: 20: 355</td>
<td><strong>1.24%</strong></td>
<td>2.78%</td>
<td>1.39%</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td></td>
<td><strong>0.67%</strong></td>
<td>1.08%</td>
<td>0.77%</td>
</tr>
</tbody>
</table>

Table 6.9 shows the testing results of 24 views images for training of Multi-view Car dataset. The original image has the lowest error rate for 2 groups, MWIS image has one group that has lowest error rate. Original image has the lowest average error rate.

Table 6.9: Testing results of 24 view images for training of Multi-view Car dataset

<table>
<thead>
<tr>
<th>Group Number</th>
<th>Approximated Angle of Training Image</th>
<th>Testing error rate (original image)</th>
<th>Testing error rate (MWIS)</th>
<th>Testing error rate (LTSN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0 : 15: 345</td>
<td><strong>0.22%</strong></td>
<td>1.10%</td>
<td>0.38%</td>
</tr>
<tr>
<td>2</td>
<td>5: 15: 350</td>
<td><strong>0.05%</strong></td>
<td>0.16%</td>
<td>0.22%</td>
</tr>
<tr>
<td>3</td>
<td>10: 15: 355</td>
<td>0.38%</td>
<td><strong>0.11%</strong></td>
<td>0.33%</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td></td>
<td><strong>0.22%</strong></td>
<td>0.46%</td>
<td>0.31%</td>
</tr>
</tbody>
</table>
Table 6.10 shows the testing results of 36 views images for training of Multi-view Car dataset. For first group, the three input images get the same error rate which is 0.06%. In fact, since the number of testing image is 1579, the 0.06% error rate means there is 1 error classification \( \frac{1}{1579} = 6.3331 \times 10^{-4} \), this error is caused by object occlusion problem. For the second group, original and MWIS image both get 0% error rate. Then, for the average error rate, original and MWIS image are both lowest.

<table>
<thead>
<tr>
<th>Group Number</th>
<th>Approximated Angle of Training Image</th>
<th>Testing error rate (original image)</th>
<th>Testing error rate (MWIS)</th>
<th>Testing error rate (LTSN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0: 10: 350</td>
<td>0.06%</td>
<td>0.06%</td>
<td>0.06%</td>
</tr>
<tr>
<td>2</td>
<td>5: 10: 355</td>
<td>0%</td>
<td>0%</td>
<td>0.06%</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>0.03%</td>
<td>0.03%</td>
<td>0.06%</td>
</tr>
</tbody>
</table>

6.3 Analysis and Discussion

According to the results of testing, the classification of multi-view objects is successful with acceptable error rates. The error rate is decreased significantly by increasing the number of training images. The accuracy can arrive at 100% for coil-100 dataset when the number of training images arrives at 36, and almost 100% accuracy for Multi-view Car dataset. The error rate is not very high when the numbers of training images are small. However, we still try to find the possibility of improvement by observing the effect of image enhancement techniques. For the effect of image
enhancement, the Tables 6.11 & 6.12 show the average error rates of Coil-100 dataset and Multi-view Car dataset in ascending order.

Table 6.11: The average error rate of Coil-100 dataset in ascending order

<table>
<thead>
<tr>
<th>Coil-100 dataset</th>
<th>The average error rate in ascending order</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>8 views</td>
<td>LTSN (3.87%)</td>
</tr>
<tr>
<td>12 views</td>
<td>MWIS (1.55%)</td>
</tr>
<tr>
<td>18 views</td>
<td>LTSN (0.31%)</td>
</tr>
<tr>
<td>24 views</td>
<td>Original (0.19%)</td>
</tr>
<tr>
<td>36 views</td>
<td>Original (0%)</td>
</tr>
</tbody>
</table>

Table 6.12: The average error rate of Multi-view Car dataset in ascending order

<table>
<thead>
<tr>
<th>Multi-view Car dataset</th>
<th>The average error rate in ascending order</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>8 views</td>
<td>MWIS (4.29%)</td>
</tr>
<tr>
<td>12 views</td>
<td>MWIS (2.45%)</td>
</tr>
<tr>
<td>18 views</td>
<td>Original (0.67%)</td>
</tr>
<tr>
<td>24 views</td>
<td>Original (0.22%)</td>
</tr>
<tr>
<td>36 views</td>
<td>Original (0.03%) = MWIS (0.03%)</td>
</tr>
</tbody>
</table>

According to the average error rate of Coil-100 dataset, the performance of classification is increased when the number of training images is low. For 8 and 18 views’ case, the LTSN has the best performance. For 12 views, the MWIS get the lowest error rate, but if the number of training images is increased after 18 views, the original images become the winner. Then, according to the average error rate of Multi-view Car dataset, the MWIS get the lowest error rate for 8 and 12 views, and LTSN has the medium performance for these two cases. However, if the number of training...
images is increased after 12 views, the original images become the winner. The performance of LTSN and MWIS are different in these two datasets. For the Coil-100 dataset, the LTSN is better than MWIS, but MWIS would become better in the Multi-view Car dataset. Therefore, we can get two points: first, the LTSN and MWIS can provide the positive effect on small numbers of training image case, and second, different enhancement methods have different performance on different types of images.

For the first point, we speculate a possible reason that may have caused this issue. The image enhancement technique brings some noises to images while enhancing its feature. The enhancements of the feature are the positive effect to the classification accuracy, and noises are the negative effect. The missing features in the case of small number of training images is more than the case of large number of training images, thus positive effect of the enhancement of features larger than the negative effect of noises. If the number of training images is large enough, however, the neural network can get enough stimulation from features. At this time, the enhancement of feature would become not that effective, but the negative effects of noises would become the winner.

For the second point, images of Coil-100 have a single background, and the images of Multi-view Car dataset have a more complex background. Therefore, we speculate that MWIS may work better than LTSN in a complex background case, or the LTSN may cause more noises in complex background case.

However, in case of low number of training images, few groups of original images got better performance than enhanced images, and also in cases of large numbers of
training images, the enhanced images have higher accuracy than original images in few groups. So why did this happened? We speculate three reasons: first, the effect of contingency. The initialized weights are random, and the weights at the end of the training process are not unique. In fact, we stop the training process when the training groups get 100 % accuracy and the cost is small enough, but the value of weights are still not unique. The weights can be changed unless the cost is zero, but it is almost impossible to make the cost become zero. Therefore, the error rate of testing images is floated in a range. The second reason is individual differences. The different images are taken from different angles, and they include different information of one object. That difference can affect the experiment results. The third reason is the noise issue. Different images will get different amount of noises. If the training images of some groups get higher amount of noises, this would make the performance of this group become worse. That is also the reason that we do not use the STTF, which has a good enhancement performance, but it could bring much noise to some images.

Therefore, image enhancement can provide improved performance in some cases of the smaller training set, but it still requires improvement and more experiments in the future.
CHAPTER 7
CONCLUSION AND FUTURE WORKS

Since Yann LeCun created one of the first convolutional neural networks, the development of the convolutional neural network has never stopped. In this research work, we applied convolutional neural network to do classification of the multi-view object and compared results of different types of input images. The CNN algorithm is powerful with high accuracy, and the CNN based multi-view object classification is successful.

In experiments, the error rate is decreased significantly by increasing the number of training images. The accuracy is 100% for coil-100 dataset when the number of training images is 36, and almost 100% accuracy for Multi-view Car dataset. The error rate is not high when the numbers of training images are small.

For image enhancement technique, the results of the experiments show that image enhancement technique can provide positive effects in the case of small numbers of the training views. The LTSN is better than MWIS for the Coil-100 dataset, but MWIS would become better for the Multi-view Car dataset. Therefore, we speculate that MWIS may work better than LTSN in a complex background case, or the LTSN may cause more noises in this case. Then, experimental data also show that in small number case, the original images had better performances than enhanced images in some groups.
This may be caused by contingency, or individual differences, or noises. Therefore, it will need more research works in the future.

For future work, we need to find methods to improve the performance of image enhancement on CNN based multi-view object classification. On one hand, there are some other algorithms of image enhancement, advanced algorithms can be found or created to improve the performance. On the other hand, to solve the problem of noise, it need to find or create lower noise image enhancement technique, or use the image noise reduction technology.

The image enhancement technique, which is an external influence, is used as one kind of preprocessing method of the input images of CNN. The CNN can also be improved by internal improvement. First, use the drop-out technique [21] to reduce the overfitting. Second, apply the region convolutional neural network (RCNN) [27]. Third, to improve the performance by adjusting the structure of CNN, this design of CNN comes from the experiment. In fact, we designed several structure before it, and finally chose this design because it has the highest testing accuracy, but it still has plenty of room for upgrade.

In general, the image enhancement techniques can provide improved performance in some cases of the smaller training set of CNN based multi-view object classification, but it still needs more researches in the future. Research work is in progress to modify the CNN architecture to observe the effect of recognition performance for multi-view object classification. Advanced non-linear enhancement technologies might be investigated and tested to see the impaction to classification.
BIBLIOGRAPHY


