Shoe print Identification from Images with Convolutional Neural Network

Chengran Li

A thesis submitted to the Auckland University of Technology

in partial fulfilment of the requirements for the degree of

Master of Computer and Information Sciences (MCIS)

2018

School of Engineering, Computer and Mathematical Sciences

Abstract

Shoe print identification/classification technology is a great significance in crime scene investigation. Traditional shoe print identification and classification technology relied on the experience of investigators. More recently, machine learning technology has brought new research direction and impetus to shoe print identification technology.

This thesis presents a method for classifying shoe print images based on convolutional neural networks (CNN). The main contributions of this thesis are listed as follows. (1) The research includes surveys and summarises existing classification methods of shoe print and proposes: the implementation of a CNN in the classification of shoe prints. (2) The traditional machine learning method artificial neural network (ANN) is used as the classification accuracy benchmark for CNN. (3) The traditional machine learning method support vector machine (SVM) provides a second classification accuracy benchmark for CNN. (4) This research uses CNN to establish an image classification model, optimising the neural network to have higher classification accuracy. The results show that CNN has an outstanding performance in binary classification: the accuracy of classification reaches 99.91%; the sensitivity of CNN reaches 100%; and the specificity reaches 97.05%. These results surpass the other approaches (ANN and SVM). (5) The research successfully visualized the CNN model, which included features extracted from different layers, the kernel visualization, and the kernel heat map.

Keywords: Pattern Identification, Shoe print identification, Convolutional neural network, Neural network Visualisation.

Attestation of Authorship

I hereby declare that this submission is my work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.

Signature: Date: November 2018

Acknowledgement

First and foremost, I am very grateful to my primary supervisor Professor Ajit Narayanan, who inspired, educated and supported me throughout the year, so that I could have the opportunity to complete a master's degree under his supervision.

I also want to thank his student, Nidhi Gowdra, who is very experienced in setting up neural network hyper-parameters. He was of enormous help to me.

I would like to express my heartfelt thanks to Dr. Adam Kortylewski and Prof. Thomas Vetter from the University of Basel, Switzerland. They provided a significant dataset for research, without which my experiment could not have been completed so smoothly.

I wish to express my sincere appreciation to Sharda Mujoo from the Faculty of Design and Creative Technologies at Auckland University of Technology. She provided me with a lot of guidance through the school's official support at all stages of my study.

Finally, I am delighted to give thanks to my parents, who provided not only funds for my scientific research, but also the love and virtue of the family from distant China. Their emotional support, wisdom, foresight, and judgment are priceless to me.

My supervisor Professor Ajit Narayanan, my parents, my college mates - these people gave me the courage, confidence and ambition during the research journey, I cannot thank them enough.

Table of Contents

1. INTRODUCTION 11

1.1 BACKGROUND AND MOTIVATION 11

1.2 CONTRIBUTIONS 13

1.3 STRUCTURE OF THIS THESIS 13

2. LITERATURE REVIEW 16

2.1 A SURVEY OF SHOE PRINT RECOGNITION 16

2.1.1 Introduction 16

2.1.2 Shoe print collection and image preprocessing 17

2.1.3 Verification algorithm 22

2.1.4 Conclusion 26

2.2 A SURVEY OF FACE RECOGNITION WITH CONVOLUTIONAL NEURAL NETWORK 27

2.2.1 Introduction 27

2.2.2 Deep learning model with CNN 29

2.2.3 Application of CNN in face recognition 33

2.2.4 Dataset collection 37

2.2.5 Conclusion and discussion 38

2.3 THE LITERATURE REVIEW OF DEEP NEURAL NETWORK AND TENSORFLOW 39

2.3.1 Introduction 39

2.3.2 The characteristics of TensorFlow 42

2.3.3 The limitations of the previous platform 44

2.3.4 Acceleration of hardware in machine learning 45

2.3.5 Conclusion 48

3. RESEARCH QUESTION AND METHODOLOGY 49

3.1 METHODOLOGY 49

3.2 RESEARCH QUESTIONS 49

3.3 RESEARCH OBJECTIVES 50
3.4 THE EXPERIMENTAL HYPOTHESIS 50
3.5 EXPERIMENTAL DESIGN 51
3.6 DATA COLLECTION AND ANALYSIS TOOLS 51
3.7 DEVELOPMENT TESTING AND VALIDATION OF NEURAL NETWORKS 51
3.8 LIMITATION OF THE STUDY 52

4. RESEARCH METHOD AND TECHNIQUES 54

4.1 ARTIFICIAL NEURAL NETWORK 54
4.2 SUPPORT VECTOR MACHINE 55
4.3 CONVOLUTIONAL NEURAL NETWORK 55
4.3.1 Characteristics of convolutional neural network 55
4.3.2 The topological structure of convolutional neural network 56
4.4 DERIVATION OF CONVOLUTION NEURAL NETWORK THEORY 61
4.4.1 Neuron 61
4.4.2 Forward propagation 62
4.4.3 The backpropagation 63
4.4.4 Convolution layer gradient calculation 65
4.4.5 The gradient calculation of the sub-sampling layer 67
4.4.6 Combination of feature graphs 69

5. EXPERIMENT DESIGN AND EXPERIMENT 71

5.1 DATASET 71
5.2 ARTIFICIAL NEURAL NETWORK 71
5.2.1 Experimental Design Principles 71
5.2.2 Experimental Design 72
5.3 SUPPORT VECTOR MACHINE 74
5.4 CONVOLUTIONAL NEURAL NETWORK EXPERIMENTAL DESIGN 75

6. RESULT AND DISCUSSION 78

6.1 ARTIFICIAL NEURAL NETWORK 78

6.2 SUPPORT VECTOR MACHINE 80

6.3 CONVOLUTIONAL NEURAL NETWORK 81

6.3.1 Experimental discussion 81

6.3.2 Neural Network Visualization 85

6.4 SUMMARY 88

7. CONCLUSION 90

7.1 SUMMARY OF THE THESIS STAGES 90

7.2 CONCLUDING REMARKS 91

7.3 RECOMMENDATIONS FOR FUTURE RESEARCH 92

REFERENCES 94

8. APPENDIX 103

8.1 TRAIN.PY 103

8.2 MODEL.PY 105

List of Figures

FIGURE 2-0 109
FIGURE 2-1 110
FIGURE 2-2 111
FIGURE 2-3 112
FIGURE 2-4 113
FIGURE 2-5 114
FIGURE 3-1 115
FIGURE 4-0 116
FIGURE 4-1A 117
FIGURE 4-1B 118
FIGURE 4-2 119
FIGURE 4-3 120
FIGURE 4-4 120
FIGURE 4-5 120
FIGURE 5-1 121
FIGURE 6-1 122
FIGURE 6-2 123
FIGURE 6-3 124
FIGURE 6-4 124
FIGURE 6-5 125
FIGURE 6-6 125
FIGURE 6-7 125
FIGURE 6-8 125
FIGURE 6-9 126
FIGURE 6-10 126
FIGURE 6-11 126
FIGURE 6-12 127

FIGURE 6-13 128

FIGURE 6-14 128

FIGURE 6-15 129

FIGURE 6-16 130

FIGURE 6-17 131

FIGURE 6-18 132

FIGURE 6-19 133

FIGURE 6-20 134

FIGURE 6-21A 135

FIGURE 6-21B 136

FIGURE 6-22 137

FIGURE 6-23 137

List of Tables

TABLE 2-1 138

TABLE 2-2 139

TABLE 4-1 140

TABLE 6-1 141

TABLE 6-2 141

TABLE 6-3 141

TABLE 6-4 142

TABLE 6-5 142

1. Introduction

1.1 Background and motivation

Shoe print identification is one of the oldest technologies used to judge whether a person had been present at a crime scene, dating back to before the Southern Song Dynasty (960-1297A.D.) (Ci, 1235). Now, the same is achieved through the computer to identify the shoe from an image and determine the shoe print owner. Shoe print identification is an important research direction of a criminal investigation. The shoe print can be broadly broken into two classes: 1) those including 3-dimensional information (such as shoe print on the sand or dirt); and 2) those containing two-dimensional information (shoe print on the hard floor). Shoe print is a common biometric characteristic in the scene of a crime, more frequent in fact than a fingerprint (Bodziak, 2000). Statistical techniques have been used extensively in the process of image classification and have a long history. Early studies used a manual classification method for the shoe mark image. It was an encoding form similar to the ASCII coding table for classification of a shoe image such as wave, rectangular, cube, circle pattern. (Xiao, 2007) This process is complicated, and some practical difficulties exist in processing. Existing research in the field focuses on a content-based image retrieval system (Dai, 2010; Arumugam & Rathinavel, 2011) which uses three layers to implement shoe print identification.

A study of the traditional shoe print showed how it was divided into two parts or more, the front and the heel or the front, middle and the heel (Heinonenb, 1996). The first trial to separate shoe print into some parts is found in a 1990s study (Keyser, 1996). The researchers set up a shoe database in cooperation with the Netherland police; each shoe consisted of more than 80 characters, stored in the database. Each character was labelled on the sole and was represented by a small shape, such as circles, rectangles, and hexagons in the image classification section; they used an

image processing package ‘Khoros’ (Konstantinides, 1994). This method can deal with two-valued images, black and white images.

In any image database, dimension reduction is essential (Hamiane, 2008). It can extract characteristics efficiently and reduce computational complexity in classification. Discrete Cosine Transform (DCT) (Levine, 2001), Linear Discriminant Analysis (LDA) (Sirvoich, 1990) and Principal Component Analysis (PCA) (Pentland, 1991; Belhumeur, Hespanha, & Kriegman, 1997) are the main techniques used for data reduction and feature extraction in the appearance based approaches. DCT, Eigen faces and Fisher faces (Lu, Plataniotis, & Venetsanopoulos, 2003) have proven to be the most successful of all the others. It was noticed in the image processing literature that most effort is put mainly into developing a characteristic extraction method and creating powerful classifiers. For example, Euclidean Distance Classifier, Hidden Markov Models (HMMs) (Aboulnasr & Othman, 2003) and Support Vector Machine (SVM). Convolutional neural networks and deep learning development provide a new idea for pattern recognition in images (Lawrence, 1997). In convolutional neural networks, the generic descriptors extracted are very powerful. Through using a linear SVM classifier, the feature representation is extracted from a neural network intermediate layer (Razavian, 2014). Using simple augmentation techniques, such as jittering, enhances the outcome (Everingham, Gool, Williams, Winn, & Zisserman, 2012). The results show that the features obtained from deep convolutional network learning should be the primary candidate for most visual recognition tasks (Razavian, 2014). These general classification methods enabled the processing of an image to allow recognition automatically, which was an important achievement.

When the original image has been processed as a grain image (grayscale), the Gabor transform can provide another multi-resolution feature of a shoe print (Woźniak, 2017), which enables rotation and intensity invariant shoe print matching images (Creighton, 2017). Compared with the Fourier Transform (Remes & Senko, 2016)

and its Power Spectral Density (Gómez, 2017), Gabor transform represents an improvement in accuracy. An especially unclear or fragmented image, even with various interference factors, can be managed well so Gabor transform is taking shoe print recognition in a new direction.

1.2 Contributions

The main contributions of this study are as follows. The first is an investigation and summary of the existing shoe print analyses and shoe print classification methods. Second, an artificial neural network (ANN) is proposed for classifying shoe print images. Third, a support vector machine (SVM) is proposed for classifying shoe print images. ANN and SVM are used as a benchmark for convolutional neural networks. Fourth, a model of the convolutional neural network is proposed to deal with the problem of shoe print classification. Fifth, the existing convolutional neural network model is optimised, and the different performances of convolutional neural networks under different parameters are discussed. Sixth, a visualisation of neural networks is presented that shows the shape and function of the layer and the convolution nucleus of the neural network at different angles.

1.3 Structure of this thesis

The second part of the thesis introduces the development of shoe print identification technology. Traditional shoe-print identification technology divides images to extract their features. These techniques also use a variety of classifications and definitions,

using a variety of methods and algorithms and will be highlighted in 2.1.
Using convolutional neural networks to identify facial images is a popular research area. In the process of studying face classification, researchers have experimented with a variety of deep learning models. These models can be used in this research because facial image and shoe print image have common relevance in some areas and the details are described in Chapter 2.2.

This study requires deep neural network technology. The convolutional neural network model (compared to ANN) is relatively new. It is essential to have a strong platform to provide technical support. We studied the Tensorflow which was developed by Google as the main implementation platform for our research. The development of TensorFlow and the present situation is summarised in Chapter 2.3.

In the third chapter of the thesis, the researcher sets out the problems that need to be solved in this study and discusses its methodology. In this section, the research process is outlined.

Chapter 4 explains the research technique used in this thesis. The core of the technology - artificial neural networks, support vector machines, convolutional neural networks - will be discussed in detail. Especially focused on in this research are convolutional neural networks in terms of the neurons, forward propagation theory and back propagation theory.

The fifth chapter is the experimental design. The researcher has designed several experimental approaches to improve the credibility of the research methods used in this experiment. Specific details will be discussed in this chapter.

The sixth chapter of the thesis is the significant section of the study: the results and discussion of the experiment. In this stage, the advantages and disadvantages of the three methods are compared, and the superiority of convolutional neural network model

is analysed. The researcher visualises convolutional neural networks and tries to understand the functionality of convolutional layers and filters.

The last chapter of this thesis is the conclusion. It discusses the results of this study, summarises the shortcomings of this study, and suggests future research directions.

Note that all tables and figures are located in separate appendix to ensure clarity of presentation and intelligibility.

2. Literature Review

2.1 A Survey of shoe print recognition

2.1.1 Introduction

Shoe print is the mark left on the surface of the sole. When a person walks, his sole will exert pressure on the floor and leave a mark. Shoe print plays a significant role in the detection of new criminal cases (Gharsa, 2008). The shoe print which is left at the crime scene will express various kinds of information about criminals (Sawyer, 1995). According to the size of the shoe print, the height and weight of the suspect can be obtained (Sawyer, 1995). Most shoe brands have their unique texture; the modern classification standard can be used to determine the suspect's social status and income level (Rathinavel, 2011). The faster classification and identification of shoe prints obtained at the crime scene will effectively improve the efficiency of solving cases (Horswell & Cordiner, 2002).

A shoe print can be mainly divided into two parts, the toe and the heel. It also can be divided into four parts which are tip, forefoot, bow and heel (Guan, Li, & Zhong, 2008). In many cases, the footwear stamps collected at the crime scene are incomplete, perhaps only a quarter complete. Using the limited original evidence to infer the classification of the entire shoe print is complicated in the traditional manual identification stage, and it takes much time to identify and match. Some researchers have begun to use part of the shoe print to match some or all of the shoe prints by using sub-space approaches. In their experiments, the accuracy of complete shoe prints was 65% to 87%, and 55% to 78% in partial shoe prints. Nibouche, Bouridane, Gueham, and Laadjel (2009) propose a solution to rotated partial shoe prints retrieval, based on the combined use of local features.

This survey is divided into three parts. The first part focuses on the collection technology of the shoe print. The second part lists the preprocessing technology of the

shoe print image. The third part discusses the various techniques and algorithms used in the identification and classification system of the shoe print. Finally, we provide some concluding observations and further instructions.

2.1.2 Shoe print collection and image preprocessing

Shoe print collection and image preprocessing are two main stages of image identification; these two stages have the same goal, which is to improve the position of shoe printing in the image. Whether it is a manual identification or machine-classified, the researchers want the shoe prints to be clearly shot in the picture, so that the judgment will be more accurate.

Shoe print collection can be divided into two types: (1) using the high precision shoe prints provided by the manufacturer to create a database; and (2) collecting the shoe prints from the scene of the crime.

Shoe print collection

Most shoe prints provided by shoe manufacturers are highly accurate pictures (Rathinavel, 2011). The picture will contain many details; the size, colour and material of the shoe will be shown in the image. The texture of the sole will be visible. Some shoe factories even offer shoe print templates with 3D information such as texture depth. If it is hard to obtain the shoe print template provided by the manufacturer, using digital camera shooting is a standard way to expand the shoe print database. The existing photographic equipment has been able to display the high precision details of shoe printing.

According to Shuhui et al.'s study (2003), shoe prints are separated into two categories: plain shoe prints and three-dimensional shoe prints (Gao, 2003). The impression of the plain shoe only exerts the force on the surface of the object and does not change the carrying object significantly. Plain shoe prints can only reflect the structural features of the convex part of the sole. When high-heeled shoes are encountered, the characteristics of the middle-shoe cannot be expressed, and the entire shoe print will appear to be

interrupted. Therefore, the plain print is often incomplete, and its features are incomplete, too. When the photographs of the soles are compared with the shoe print, the inside and the outside are opposite (left and right side). A shoe print sample must be produced from the forming conditions of the plain shoe print.

A three-dimensional shoe print is an object that a shoe acts on, and the shoe causes the object to undergo a significant deformation. When stereoscopic photographs are compared with a sole image, their concave-convex surfaces are reversed, and the recessed part of the shoe print is precisely the protruding part of the sole. For verification, plaster or other plastic materials can be used to make a three-dimensional shoe print model, which can be consistent with the photos of the sole.

The collection of images at the crime scene will be more difficult and require more skill for on-site investigators. Due to the different materials on the tread surface, shoe print collection requires the use of different technologies. Depending on the surface material being stamped on, there are three main techniques for collecting shoes: (1) on sandy land; (2) soft materials, mainly on the surface of carpets, beds, quilts, sofas; and (3) on a hard surface, mainly refers to the ceramic, glass, wall and other materials (Crookes, 2007).

On the surface of a soft texture, a puzzle will be encountered - that the shoe print colour is too similar to the background. It can lead to the inability to extract the shoe print from the computer. Peng et al.'s research focuses on photography (2005). They used the difference in reflection and absorption of different wavelengths of UV light from various substances and recorded the reflection of ultraviolet light on shoe prints. Peng used ultraviolet rays with a wavelength of 253 nm to irradiate shoe prints. Since the flannel carrying the shoe prints absorbs most of the ultraviolet light, the dust is not substantially absorbed. It gives a higher contrast between the dust shoe print and the background. So that can get the details of shoe prints. This method is suitable for all photography that needs to remove traces of background colour.

Shoe prints left on hard surfaces are usually better collected, but criminals often use water to wash the leftovers. This makes shoe collection on hard surfaces also complicated (Yang, 2010). Ordinary shoe prints are due to subtle dust and water stains left on the tile surface. There is no colour difference between the shoe print and the background, and the above-mentioned ultraviolet colour separation photographic method cannot work efficiently. Yang adopted the technique of reflected light and obtained more successful images of unclear shoe prints (2010). There are three main types of shoe prints: more water, less water, and completely dry. A shoe mark with much moisture can be regarded as a three-dimensional shoe print, and a shadow of the water droplets irradiated by a single-side light is used to take a shoe print. The camera needs to be directly above the tile, and the angle between the light source and the camera at 70° to 90°. When shooting shoes with less moisture, the light source needs to be moved up, and the angle between the lens and the light source set between 0 and 38°. When the shoe print is completely dry, the distance between the light source and the shoe print needs to be increased when taking pictures, and the angle between the light source and the ground set between 60° and 90°. The camera takes a photo on the other side of the light source. However, the samples obtained by this photographic means are deformed. It is necessary to place a scale next to the shoe print to facilitate later correction.

Image preprocessing

Segmenting the target image from a complex background is a critical step in the image processing flow. So far, no method can accurately extract the target from the background. What the researchers do is to get as much detail as possible from the original picture. In shoe print identification, the required preprocessing steps include image enhancement, noise elimination, colour image conversion to grayscale, grayscale conversion to the binary image, contour extraction, and other steps.

Step one: Shoe print contour extraction and tilt correction.

The light should come as the primary consideration during the shooting. In order to get a clearer shoe print texture, the technical personnel need to adjust the shooting angle so

the shoe print can be seen at various angles. When the shoe image is preprocessed, the image is rotated to make the top of the shoe point upwards. However, it cannot ensure that all the images have the same inclination, which directly affects the automatic recognition accuracy, so the tilt angle needs to be detected, and the rotation for the image corrected. The idea of Guan et al. (2008) was that the longest distance in a shoe print image is the length of the shoe, which is longer than the length of any other two points in the shoe print. The angle between the long line and the vertical axis is the angle of the image deflection. This angle is also the point of view that needs to be correctly turned.

To achieve this effect, Guan et al. worked with a four-part process. The first involves extracting the outline of the shoe print (Guan, Li, & Zhong, 2008), recording the first black pixel point on each row from the left and the first black pixel on the right side of each row. Through this algorithm, the outline of the shoe print can be stored in a two-dimensional matrix:

$$A = \begin{bmatrix} a_{h,0}, a_{h\,1} \\ a_{h+1,0}, a_{h+1,1} \\ \cdots \\ a_{k\,0}, a_{k,1} \end{bmatrix}. \qquad (2.1)$$

Where *ax0* stores the vertical coordinates of the first black pixel point on the left, *ax1* stores the vertical coordinates of the first black pixel point on the right; *h* is less than *x* less than *k*, *h* indicates the first black pixel point that appears on a line by line, and *k* is used to represent the cross coordinates of the last black pixel point (see Figure 2-0).

After the shoe print is extracted, Guan et al. calculated the shoe length (the longest axis) – the second part of the process. The distance between the longest two points of the shoe is used as the length of the shoe, and the formula is used: $l = \{ max(distance ((i, ai0) , (j, aj, 1)) \mid h \le i \le k, i \le j \le k \}$. Among them: distance *((I, ai0), (J, aj. 1))* the distance between the outline points of coordinates *(I, ai0)* and coordinates *(J, AJ, 1)*; *h* is the

transverse coordinates of the top black pixels in the outline; *k* is the transverse coordinates of the bottom black pixels in the outline; the ai0 is the column coordinates of the left wheel profile of the row coordinates; *ai1* is the right wheel of the row mark. The column coordinates are the outline point. To improve the execution efficiency and avoid duplicated two-wheel profile points, the contour points on the left of the longest axis and all the right contour points on the right side of the longest axis are more than equal to all the right contour points of the left profile point line (also see Figure 2-0).

The third part is to calculate the angle α between the longest axis and the horizontal X-axis.

The fourth part is if α is less than 90°, then the angle of the rotation of the image is 90° - α (anticlockwise), if the α > 90°, then the image rotates clockwise at the α - 90°.

Step two: image enhancement.

Image enhancement is employed to highlight specific information in a particular image according to specific requirements while removing unnecessary information to improve the recognition rate of images. Guan et al. pointed out that the use of Retinex image enhancement algorithm will have a good effect in dealing with shoe print images (2008). Retinex has good results in sharpening, colour constancy, dynamic range compression, colour fidelity, and so on. The colour image itself contains a large amount of colour information, the space occupied by the image is ample, and the processing takes a much longer time. However, in shoe print identification, what the researchers need is the relationship between the effective pixels of each shoe print, without the need for colour information. Converting colour images into black and white pictures is an effective way to reduce image size. During the transformation from a grayscale image to a two-value map, the maximum entropy principle is used to select the threshold value to cut the image. After segmentation, the image is transformed into a two-value image with a white background and a black shoe print.

Step three: image feature extraction

The rotated image needs to re-extract features. There are two main methods for extraction. One is extracting the outline only, in accordance with Guan et al. (2008), and the subsequent identification is also based on the critical points of the shoe print outline. These researchers divided the shoe print into four areas: the tip of the shoe, the sole, the bow, and the heel. Each area occupies part of the shoe print. According to the above-proposed method, they calculated the width of each area. If only using the four parameters of shoe length, palm width, bow width and heel width, describing the outline characteristics of the shoe print is not enough; the two shoe print outlines with entirely identical four parameters will appear completely different, as shown in Figure *2-1*.

The main reason for this situation is that the position of the palm width, width of the bow, and the width of the heel relative to the entire shoe are not taken into account. That is the distance or angle of each feature point. Guan et al. not only considered the length and width of each part but also the distance of each feature point. The distance reflects the position of the palm width, bow width, and heel width relative to the entire shoe. The shape characteristics of the shoe print are described as S = (CD/AB, EF/AB, GH/AB, AI/AB, AJ/AB, AK/AB, CI/CD, EJ/EF/GK/GH).

The second method of extracting image features is to record the position of each point in the image. The abscissa and ordinate of each point are arranged in a row as a sample. The sample with the largest length is used as the length of the entire matrix, and the rest is filled with 0.

2.1.3 Verification algorithm

Identification and verification are the two phases of the entire identification system. In verification, only a yes/no response is required. In identification, the shoe print must be identified by matching against a database entry (Adams, 2009). The verification requires high accuracy, and the identification needs to have fast identification speed with high accuracy. Only in this way can the theory be used in actual projects. However, it is regrettable that existing researchers rarely record the speed of identification and

verification. This is because computer hardware is very different. Machines at different price points will have utterly different verification speeds. On the other hand, most researchers are only aiming for an improvement of one of the algorithms. Compared with the traditional model, the improved system efficiency after the optimised algorithm is obvious. In this section, the advantages of various algorithms will be compared. Verification and identification approaches are mainly divided into pattern-based, subspace-based, and other complicated methods. The survey here is also based on these three categories.

Pattern-based approaches:

The shoe print identification technology based on the pattern is the most traditional shoe print identification method, and shoe printing itself is a collection of various patterns. According to the study by Gao et al., today's police department in China is still based on the preservation and identification of shoe prints (2003), which are mainly based on the different pattern features of different regions after the segmentation of the shoes and the identification of different images: the damaged shape of the shoe tip, the main bearing area of the shoe, the shape of the shoe bow, and the wear degree of the heel. These indicators can be used as a basis for judging a person's walking habits. Soles vary from person to person as a result of the amount and style of walking. The damage from the shoe tip area usually reflects the characteristics of a person when starting a step. The damage pattern of the toe is mainly circular, triangular or irregular. The abrasion of the sole reflects the main force exerted by a person while walking. According to the different shapes of the pressure surface, it is possible to learn about the change of the centre of gravity of his/her body when walking. For the same reason, the wear of shoe bows and heels can also be recognized as revealing attributes of the person.

According to different parameters of the shoe print profile, the classification of shoe prints has reached a better level (Guan, Li, & Zhong, 2008). Similarly, the shoe print is divided into four parts, and the information of the area, width, and length of each part

can also be used as the attribute of classification. To improve the efficiency of identification, Guan at al. not only match the properties of the width and area and the data in the database, but also deal with the offset of the shoe palm and the shoe bow relative to the whole shoe, and thus get a more accurate value.

Subspace-based approaches

Compared to the traditional way of classifying shoes based on patterns, the subspace-based approach is more varied. Scale-Invariance Feature Transform (SIFT) is one of the popular methods for detecting key points of an object (Li, 2011). According to the research of Li (2011)., SIFT can identify the same object in another image by rotating, translating and illuminating the shoe print on the premise of invariable scale. The same method is used in Dong's research (2017). SIFT can rely only on partial patterns to match the complete pattern. Moreover, this algorithm makes full use of the grayscale statistical characteristics of the image and avoids any influence on the picture quality due to factors such as local environment, light, and noise. The overall performance of the SIFT operator is better than other operators, and the extracted feature points have good stability, especially for local tasks such as image identification and image matching.

Based on SIFT, Dong introduced Random Sample Consensus, RANSAC. The algorithm can further improve the overall system accuracy, enhancing the availability and robustness. The experimental results also show that the accuracy of the matching is 91% to 96% using a identification system combining SIFT and RANSAC.

Because the ordinary shot image contains much information such as environment, colour, light, and so on, this information will greatly increase the time of image identification. The researchers want to achieve high-speed matching of existing pictures and pictures in the database, dimensionality reduction is inevitable (Rathinavel, 2011). Principal component analysis (PCA), linear discriminant analysis (LDA), discrete cosine transform (DCT), and single-value decomposition (SVD) are methods that can

effectively reduce the size of the image. Rathinavel pointed out that DCT is conceptually similar to the discrete Fourier transform (DFT), except that the DCT does a better job of concentrating energy into lower order coefficients than does the DFT for image data (2011).

PCA is also known as the Karhunen-Loeve method. This technique can effectively reduce the dimensions of computer vision. In previous studies, PCA has often been used in face identification to reduce the size of data. In the study by Rathinavel , PCA was used to form eigenvectors of the covariance matrix of the set of shoe print images, treating an image as a point in a very high dimensional space (2011).

SVD, such as principal component analysis, digital watermarking, and data compression, has been widely used by past researchers. This has also been of high research value in the field of image identification (Guo, 2018). Images are compressed on a large scale and, after undergoing singular value decomposition, most of the elements in the matrix have a value of 0.

Other Approaches

In the most advanced research, it is evident that single means are not enough to achieve the most accurate and efficient results. Researchers have tried to improve the accuracy of identification by various means. In the study by Sun, canonical correlation analysis (CCA) is used to extract features and form more representative identification vectors (Sun, 2005).

The use of K-SVD dictionary to learn about and solve the problem of image identification is a new way of thinking. Especially in the processing of large sample data sets, dictionary learning will ensure greater progress (Liu & Liu, 2018). Image processing using SVM-PSO and complex image processing methods in SVD and DWT domains also has excellent classification results (Chang, 2016). According to Chang (2016), the accuracy of this method for complex texture image classification reaches 95% to 97%. Chang used SVD to enhance the image texture and extract the DWT and

SVD features in the image. The two features are combined to get the best feature combination (2016).

2.1.4 Conclusion

In the field of shoe print identification, many researchers have designed algorithms based on previous face recognition methods. To improve the accuracy and speed of recognition, the researchers hope to design better algorithms. A combination of multiple algorithms can lead to a breakthrough in accuracy. Before introducing new technologies into the field of shoe print identification, research into the existing models is of great value, and this has also provided the direction for our subsequent research.

According to this chapter's survey, there are still no research organisations or individuals using artificial neural networks to classify shoe prints in the literature published so far. So in this study, we will use traditional machine learning methods to classify shoe prints, using the obtained accuracy as the benchmark for the convolutional neural network training. We chose the artificial neural network and support vector machine as the two methods to classify the existing datasets, hoping to get a good benchmark.

In addition, the survey revealed that there are still no organisations or individuals using the convolutional neural network for shoe printing classification either. In order to understand the current situation of convolution neural network in pattern classification, some research needs to be carried out into the convolution neural network in facial recognition in order to obtain some methods and means for use when applying CNN to shoe print recognition. This is the subject of the next chapter.

2.2 A Survey of Face Recognition with Convolutional Neural Network

2.2.1 Introduction

With the development of traditional machine learning technology, deep learning can produce more appropriate expressions from one layer to multi-layers, and, as a result, has been widely used in many areas. The traditional deep learning algorithm loses the structure information of the original image in pattern recognition. This leads to a decline in the recognition accuracy (Yin, Wang, & Wang, 2015). However, the convolution neural network not only inherits the advantages of the automatic extraction of features from the traditional deep learning neural network but also guarantees the spatial information of the original data by the local receptive field (Yin, Wang, & Wang, 2015). During sub-sampling, Local Correlation Principle (LeCun, Boser, Denker, & Henderson, 1989) is used to reduce the amount of data processing while preserving the structure information. The parameters needed in the training process are reduced, so it has a better effect in various recognition fields by sharing weights, and it still has a high level of robustness after the image shift, scale, and distortion invariance.

Convolution neural network has apparent advantages in the face recognition area. Convolution neural network can reduce the complex structure of the traditional neural network as far as possible. Convolution neural network can be used as input directly by using raw data (such as pixel value or original images), avoiding the extra data preprocessing as in traditional recognition algorithms (LeCun, Bottou, Bengio, & Haffner, 1998). There are two main reasons for focusing on facial recognition technology. First, the shoe print and the human face both have clear edges; in the same way that the human face may be oval or round, the shoe print also has a unique shape. Digits or articles of clothing, for example, do not share this feature. Second, the leather-grain of sole has huge difference from other shoes. CNN may extract more features than fingerprint technologies.

The earliest use of the convolution neural network can be traced back to the 1980s York University professor LeCun et al. who used CNN for handwritten digital recognition and as a tool for bank identification of bill data (LeCun, Boser, Denker, & Henderson, 1989). In 2012, in Canada, Hinton and others got the best results on ImageNet and they used a deeper convolution neural network (Hinton & Salakhutdinov, 2006; Hinton, Osindero, & Teh, 2006; Hinton & Krizhevsky, 2012). Facebook also achieves 97.25% accuracy in face recognition using deep convolution neural network (Taigman & Yang, 2014). Team Face++ from China won the championship by using Pyramid CNN in face recognition public data LFW (Fan & Cao, 2014). In 2012, researchers at the GoogleX laboratory built the world's largest artificial neural network, Google brain, using 1,000 computers (16K CPU in total). They extracted nearly ten million still images from YouTube videos as training sets. The trained Google brain can automatically classify faces, human beings, animals and other types of images from the Internet video (Sungjoon, 2013).

Face recognition technology is a significant branch of machine learning. A face recognition system usually includes face image acquisition, image preprocessing, face feature extraction and feature classification. Among these stages, feature extraction plays a core role in a face recognition model and affects the recognition rate of the whole system. Because of this, the construction and extraction of facial expression features received extensive attention. Images of faces contain many feature points that can be extracted by CNN; the complexity of the image is incomparable with digital or character images. This is very similar to the image of the shoe print, which also contains a lot of random feature points. Some of these features can be extracted by neural networks, while others are more difficult. Based on facial recognition studies (referred to below), we can understand the methods used to process complex images.

Researchers have designed a range of methods including: Active Appearance Model (AAM) (Cootes, 2001), Gabor Features transfer (Gu, 2012), Local Binary Pattern (LBP) (Shan, 2009), Histograms of Oriented Gradients (HOG) (Wang, 2013), and Local

Discriminative Component Analysis (LDCA) (Jiang, 2014). The common point of these methods is use features which, to a certain extent, are human-made and so lose the original feature information. Alternatively, assuming that the features are independent of each other, these features cannot express the actual practical environment. In recent years, in the machine learning area, feature learning algorithms have appeared (Bengio, 2013). These include: Local Linear Embedding (LLE) (Roweis, 2000), Laplacian Eigenmap (LE) (Belkin, 2003), ISO metric MAP ping (ISOMAP) (Tenenbaum, 2000), Uncorrelated Locality Sensitive Discriminant Analysis (ULSDA) (Lu G. , Neonatalpainexpresionrec- ognition based on uncorelated loclaity sensitivediserimi- nantanlaysis, 2013), Two-dimensional Locality Preserving Discriminant Analysis (2D-LPDA) and (Lu G. , 2014). These methods abandon manual marking and the obvious feature extraction methods, and build a multilayer neural network so that the machine can learn the more essential features from sample data, which make these features more generalizable and characterisable.

This survey mainly investigates several recent studies on the realisation of facial recognition using convolution neural networks and introduces the different performance of various algorithms or combination of algorithms in identifying facial features. The first part of the chapter is an introduction to the deep learning model and the theory of convolution neural networks. The second part introduces approaches and implementation methods used by researchers in the process of facial recognition research. The third part introduces several common facial recognition databases. The fourth part of the chapter discusses the methods that can be used in subsequent research.

2.2.2 Deep learning model with CNN

Deep learning is essentially a general term for a class of training methods that have deep structure models. The deep structure is relative to the shallow structure. The shallow structure model usually contains nonlinear feature transformations that are not more than one or two layers, such as Gaussian Mixture Model (GMM), Support Vector

Machine (SVM), and Multi-layer Perceptron (MLP). The related research has proved that the shallow structure has a good effect on uncomplicated and constrained data (Lu , 2016). However, when researchers must deal with complex data in the real world, such as voice, natural sound, natural image, and video, these models will have problems related to the inability to capture expressions. The deep learning structure model is characterised by hierarchical representation. Regarding learning about highly nonlinear relationships and complex function representation in large data sets, multi-layer structures are more capable than shallow structural models.

Typical deep learning models include Deep Belief Networks (DBN) (Hinton & Salakhutdinov, 2006), Stacked Auto-encoder (SAE) (Hinton, Osindero, & Teh, 2006) and Convolutional Neural Networks (CNN). DBN consists of several structural units stacked, and the structural unit is usually Restricted Boltzmann Machine (RBM). RBM is a special form of Boltzmann Machine; variables between the graph model connection forms are restricted, only visible layer nodes and hidden layer nodes have connection weights, whereas there is no connection between two visible layer nodes or two hidden layer nodes. The number of neurons in the visible layer of each RBM cell in the stack is equal to the number of neurons in the hidden layer of the previous RBM unit. DBM automatically learns the abstract features of different levels from bottom to top, and finally obtains the nonlinear description of features. It expresses an automatic feature extraction process that does not depend on human-marked. DBN has been successfully applied to many fields such as handwritten digit recognition. However, DBN ignores the two-dimensional spatial structure information of the image and the local structure between adjacent pixels, and it is difficult to learn the local features of the face image. Moreover, the learning processing of DBN is slow, and the improper selection of parameters will lead to the convergence of the learning to the local optimal solution. The structure of Stacked Auto-encoder (SAE) is similar to DBN and is stacked by several structural units. However, the difference between the two is that the structural unit of SAE is Auto-encoder instead of RBM.

The convolution neural network is a deep neural network containing a convolution layer, whose template was initially inspired by neuroscience, mimicking the axon and nucleus (cell body) in nerve cells which are on the visual cortex to process visual information. Axon responds to edge information from different directions, and the cell body accumulates the output of single cells nearby. It became known as the Hubel-Wiesel structure (Hubel & Wiesel, 1962). CNN includes a multi-stage Hubel-Wiesel structure. Each stage usually contains basic convolution operations that simulate axon and pooling operations that simulate cell body. In CNN, the units in the image (local receptive fields) are input, and they are the lowest in the network hierarchy. The information is transmitted to different layers, each layer using a digital filter to obtain the most valuable characteristics of the observed data. This method can obtain significant characteristics of the observed data for image shift, scale, and distortion invariance, because the local receptive fields of the image allow neurons or processing units to access the most salient features, such as directional edges or corner points.

The basic structure of the convolution neural network is: data input layer (the dimension centre of input data to 0, remove many deviations, avoid impact results); convolution computing layer (CONV, linear involution, summation); motivation layer (ReLU, one of the motor function); pooling layer (marked as POOL, take the region average or maximum value, effectively reduce the matrix size, reduce the full connection layer parameters); full connection layer (FC, output a program to select the category of an n-dimensional vector); and output layer (OUTPUT recognition classification results) (Xu & Liu, 2018).

The basic learning processing is:

1) The image is processed into the neural network model by the input layer preprocessing; 2) the convolution layer of the filter is placed in the convolution operation because the filter in each convolution operation has the ability of local perception so that the neural network can perceive the local characteristics of the image;

3) the motivation function of the convolutional operator processes the results; and 4) the processing result is output to the neuron, all the neurons output in the layer constitutes the feature map of this layer, set as the input of the next layer. Each neuron in the next layer is connected to the local receptive fields of the previous layer.

It is assumed that the attribute *a* is on convolutional layer *b* which can be explained by X^{l}_{j}, then:

$$x_j^l = f\left(\left(\sum_{i=M} i \cdot K_{i \cdot j}^l\right) + b_j^i\right) \tag{2.2}$$

Formula: $f(.)$ is the activition function, M as the input of the previous layer of the set of feature graphs. K is the weighted value of the convolution kernel, and b is the only variable that can be added to the convolution feature chart X, so that can reduce the amount of computation in the next stage. After the convolution features are extracted from the image, a computational layer, which is used to compute the local mean and secondary feature extractions, is needed. It is called the sub-sampling layer, and the lower sampling layer can reduce the feature dimension extracted by the convolution layer and reduce the resolution of the feature.

Assuming that the feature map j of the sampling layer l is represented by X, then:

$$x_j^l = f\left(\beta_j^l down\left(x_i^{l-1}\right) + b_j^l\right) \tag{2.3}$$

In the formula: *down ()* is a sub-sampling function, β is the unique product variable of the feature graph, b is the only addition and subtraction variable of the convolution feature graph. Multiple convolution layers and a sub-sampling layer be have operate it according to the experiment requirement.

The training of the convolution neural network is set out below.

1) The forward propagation stages. Sample x is a sample of the training sample set. y is the corresponding category label, and the X is input from the input layer to the convolution neural network model, and then the output of the current layer is calculated by the motivation function of the current layer. In the convolution neural network, except for the last layer, the output of each layer is used as the input for the next layer and passed down by layers. The Softmax layer is finally obtained. The output of the layer is Y. Y is an n-dimensional vector. Every dimension in the n-dimensional vector represents the probability that x becomes the corresponding category.

2) The stage of error propagation. Calculate the error between the output Y from Softmax layers and vector y from the labelled class of given samples. The weight value parameter is adjusted by minimising the mean square error cost function.

2.2.3 Application of CNN in face recognition

The complexity of face images brings new inspiration to research on shoe printing. In recent years, facial recognition technology has made great progress. The face image contains many random feature points, and the shoe print image also contains many random feature points. Thus, it can be said that there is a certain correlation between the two technologies. The researcher tried to investigate existing methods of facial recognition in the hope of finding the right way to use them, drawing on their techniques, for the study of shoe print recognition

Improved Fisher criterion with CNN

Sun used an improved Fisher criterion in CNN (Sun F. , 2015). The traditional deep learning technique decreases rapidly when the number of data set samples drops. According to Sun's statistics (2015), using the classic CNN algorithm, the image recognition accuracy is 88.6%, when the training sample number is 60,000. However, when the data set sample number is reduced to 20,000, the recognition accuracy of the pattern falls to 70.2%. When the number of training samples is below 10,000, the accuracy is less than 20%. Sun effectively use the deep learning technology to extract

features automatically and put forward a CNN algorithm based on an improved Fisher criterion (2015). The constraint criterion based on Fisher is adopted in the adjustment of the weight value of the backward propagation. The minimisation of error is considered in the iterative adjustment of weights. It is important to keep the sample distance small within the class, and considerable distance between the classes so that the weight can be more quickly approximated to the optimal classification of the best value. When the sample size or the number of training iterations is not enough, the Fisher criterion can effectively improve the system recognition rate.

Deep consecutive convolutional neural network

In order to improve the extraction ability of convolution neural networks, Niu and Chen put forward a neural network model for the continuous use of multiple convolution layers (Niu & Chen, 2016). The model uses a small scale convolution kernel and a consecutive two convolution layer to deal with the issue of excessive convolutional unit quantity. This method, combined with dropout technology, can increase the non-linear expressive ability of the model and reduce the interdependence between neurons.

Deep convolutional neural network

Unlike the traditional CNN model, Oxford University's Parkhi and Omkar tried to use a much deeper CNN model (2015), which maximises the ability of neural networks to extract and learn samples. They used up to 37 layers of neural networks (Table 2-1). This neural network contains 11 blocks, each containing a linear operator (CONV) and several nonlinearities, such as Max Pooling and ReLU. The first eight are called convolution blocks because the operators are linear operators. The last three blocks are fully connected; they are the same as the first eight convolution layers, but the size of the filter matches the input layer.

After 37 layers of neural network training, the final output will be 4096-dimensional descriptor vectors. Compared with other algorithm models, using this very deep neural network, the number of samples in the training set is one per cent of the other methods. Moreover, the accuracy is very similar. It shows that a convolution neural network can be used to establish appropriate structures and suitable training methods. It also can have a good result even without any modifier algorithm (Table 2-2).

Multimodal deep face representation

The traditional CNN model focuses on two-dimensional images and has an outstanding result in recognition accuracy. Although the CNN model can effectively solve the problem of image shift, zoom and shrink, the details are missing, so this still cannot satisfy the actual requirement. The face images obtained in daily life will vary greatly depending on the angle, light, and colour in the shooting. The same face can also be large changed by the expression. The robustness of the CNN model has been questioned. Ding's research aimed to try to solve this problem (2015). Instead of using a single image to train the neural network, they used multiple-angle images of the same person to create a face model multiple time. They firstly used OpenGL to simulate a photo of the three-dimensional model and then create Multi-angle images as input. The researchers built different blocks; each block should contain two to three convolution layers and a pooling layer (total 12 convolution layers and five pooling layers). The human face model created in this way is robust. Moreover, it is suitable for multi-angle images. It should be recorded that this method would have the best accuracy in supervised training - the correct rate is over 98%.

Domain-specific data augmentation in CNN

In recent years, much of the human face recognition progress has been due to the expansion of the dataset. Facebook created labelled Faces in the Wild (LFW), which contains 4.4 million face images (Taigman & Yang, 2014). After that, to study VGG-face representation, Parkhi and Omkar. trained 2.6 million samples (2015). Moreover, face++ 's team used 5 million photos for their Megvii System (Zhou & Cao, 2015). And the latest Google FaceNet used 200 million tagged images (Schroff & Kalenichenko, 2015). Collecting and labelling these images is a huge expense, and this is not at all affordable to most researchers.

For this reason, Masi used a new method to improve the image sampling rate and reduce the number of samples in the dataset (2016). The traditional CNN model does not require preprocessing of samples. CNN's unsupervised training model automatically

extracts valuable features and feeds back to the entire neural network. It is also the reason why the researchers increased the size of the dataset - more samples will repair a few samples of abnormal conditions to prevent overfitting. Masi used image preprocessing technology, data enhancement, especially in specific areas of image enhancement, such as the eye, mouth and other parts. They used face synthesis technology to enlarge, significantly, the original data set; for example, they constantly modified the emotion appearing on the mouth to create a new sample. This approach ensures the sample quantity of the dataset, enhances the system robustness and reduces the possibility of overfitting (2016).

2.2.4 Dataset collection

Besides the databases mentioned above from Google's, Facebook's and Face++'s database, some databases, as detailed below, also have significant research value.

1. FERET face database (Gross, 2005)

Created by the FERET project, it contains 14,051 multi-pose, illuminated grey-scale face images, which is the most widely used in the face recognition field.

In one of the face databases, most of them are Westerners, and each of them contains relatively simple changes in face images.

2. MIT face database (MIT, 2018)

Created by MIT Media Lab, there are 2,592 different poses, light and size facial images of 16 volunteers.

3. Yale face database

Created by the computing vision and control centre of Yale University, 165 images of 15 volunteers, including illumination, expression and posture changes, are presented.

4. Yale face database B

This database contains 5,850 multi-pose and multi-illumination images of 10 people. The images of attitude and illumination change are collected under strictly controlled conditions. It is mainly used for modelling and analysis of light and attitude problems. Further application of the database is limited by the small number of people.

5. PIE face database

PIE was created by the Carnegie Mellon University, including 41,368 facial images, light and facial images of 68 volunteers. The attitude and illumination change images are also collected under strict control and have gradually become an important test set in the field of face recognition.

6. KFDB face database

There are 1,000 people, a total of 52,000 multi-pose, multi-illuminated, multi facial images, in which the volunteer's attitude and illumination condition changed are collected under strict rules. The volunteers are mainly Korean.

7.XM2VTS face database

This includes 295 people in four different periods of image and voice video clips. Each period, each person is recorded two head rotating video clips and six voice video clips. In addition, a three-dimension model of 293 of them is also available.

2.2.5 Conclusion and discussion

Section 2.2. briefly introduced the theoretical knowledge of convolution neural networks and the various existing research methods. Since LeCun proposed the traditional convolution neural network in 1989 (LeCun, Boser, Denker, & Henderson, 1989), researchers have continued to put forward new ideas in all aspects. CNN rely on many samples to achieve an unsupervised learning process. CNN can automatically extract parameters from samples without any pretreatment and get better recognition results. The dataset used by the researchers rose from LeCun's 60,000 patterns to 200 million facial images used by the Google FaceNet team. There has been an amazing rise in accuracy. However, another part of the research is thinking about how to use the existing, even smaller datasets to get similar precision, image enhancement and image synthesis technology. The face area is constantly modified to expand the existing dataset. The Fisher Criterion is also a way to reduce the number of sample sets; that is, the methods in the traditional recognition model can also be used in the CNN, such as Simultaneous Feature and Dictionary Learning (Lu, 2017). The CNN is not kept constant since the new structure is still proposed continuously. It is feasible to add multiple continuous coiling layers and increase the number of the whole neural network. Compared to two-dimensional images, the three-dimensional image can contain more information and details, use two-dimensional photos to simulate the stereoscopic shape and establish the multimodal structure. It is also an effective method to improve the robustness of the CNN model.

2.3 The Literature Review of Deep Neural Network and TensorFlow

2.3.1 Introduction

The convolution neural network model is a particular kind of deep neural network model (Zhang, 2017). Its particularity is embodied in two aspects: on the one hand, its neuron link is not fully connected; and, on the other hand, the weight of the connection between some neurons in the same layer is shared (the same). CNN's incomplete connection and weight sharing network structure make it more similar to the biological neural network, reduces the complexity of the neural network model, reduces the number of weights, and solves the problem of excessive computation of the weights of traditional neural networks. The basic structure of the CNN consists of two special neuronal layers, one of which is the convolutional layer; the input of each neuron is connected with the local part of the previous layer, and the local characteristics are extracted. The second is the pooling layer, which is used to compute the local sensitivity and two-time feature extraction. These two feature extraction structures reduce the resolution of features and reduce the number of parameters that need to be optimised.

In terms of the current convolution neural network, when defining the objective function of solving weights, the general definition of reconstruction error is minimal, or the actual output value and the label error are minimal. This deep neural network learning model requires many tag samples for training. Moreover, the time complexity is very high, and it often requires tens of thousands of iterations to get better recognition performance. Then in practice, the cost of sample tagging is high, too, whereas the requirement of time complexity is also very demanding (for example, it is sometimes necessary to recognise in real time).

The Google Brain Project began in 2011 to explore the use of a large scale of deep neural networks. This was both for Google's product development and for scientific research. As part of the early work of the project, researchers from Google Brain set up

DistBrief, the first generation of neural network research and inference systems that can be customised (Dean & Corrado, 2012). DistBrief provides an outstanding service; the entire system is widely used in Google's multiple projects, including unsupervised learning (Le & Ranzato, 2012), language recognition and expression (Mikolov & Chen, 2013; Vinyals & Kaiser, 2014), building models, images classification and recognition (Frome & Corrado, 2013; Szegedy & Liu, 2015), classified video files (Karpathy & Toderici, 2014), behavior monitoring (Angelova & Krizhevsky, 2015), as well as Google search (Clark, 2015). However, because the DistBrief system has shortcomings, the Google Brain team developed a new neural network system, TensorFlow. TensorFlow is deployed in some Google products, including Google search (Clark, 2015), Google ads, voice recognition system (Hasim & Andrew, 2015; Beaufays, 2015), Google Photos (Rosenberg, 2013) and Google Street View (Goodfellow & Bulatov, 2014). TensorFlow is a second-generation product that implements and deploys a large machine learning model (Abadi & Agarwal, 2016). The expression of the calculation in TensorFlow is similar to a data flow graph model and uses the data flow graphs for different OS platforms, from running on mobile platforms such as Android and iOS to running on a desktop system that contains one or more GPU. Having a system that spans such a wide platform will significantly simplify the difficulty of machine learning in actual use due to different platforms. A massive training mission deployed on a small system will have significant maintenance costs and errors. The TensorFlow computation is represented as a stateful data flow diagram, which allows researchers to focus on the use of flexible and rapidly generated new models and to apply models to experiments. TensorFlow model is sufficiently robust and high-performing for the production and training of machine learning patterns. In order to allow TensorFlow to be deployed on larger neural networks, users can express the various model using the replication and parallel execution of the core data flow graph, while the multiple computing devices included in the model collaborate to complete the updated state. A variety of methods can achieve moderate changes in computation and attempt to bring about small effects (Dean, Corrado, & Monga, 2012). When the user is employing TensorFlow, parameter settings allow for some flexibility,

and users can easily express and utilise these settings in larger deployments. Compared with DistBrief, the TensorFlow model is more flexible, has significantly improved performance, and it also supports a wider range of hardware platforms and more diverse models and algorithms. Google's internal customers shifted to using the new platform after the TensorFlow release. These customers use TensorFlow in research and production, the tasks and reasoning of which are diverse, and can be used on mobile phone vision models for large-scale deep neural network training, with over tens of millions of neurons using parallel computing in hundreds of devices (Javier & Ignacio, 2015). Today, TensorFlow mainly focuses on machine learning and deep neural networks, but in the future TensorFlow will be used in other fields, including different types of machine learning algorithms, and possibly in other forms of digital computing, such as accounting calculations or financial analysis.

The open source version of TensorFlow can only be run on a single machine. However, it supports parallelisation of multiple processors (CPUs or GPUs) on a single machine. The TensorFlow version, which is capable of distributed computing on a group of machines, was released towards the end of 2016. This is the standard development time for deep learning tools. Most current frameworks do not support distributed computing, but TensorFlow does. After the release of TensorFlow in November 2015, several problems, such as uptime and extensive memory use, caused TensorFlow to be compared unfavourably with other state-of-the-art deep learning frameworks. Google acknowledged these initial performance problems, and many of them were resolved in a new version of 0.6 released in December 2015. This rapid turnaround indicates that Google is committed to supporting and developing this open source project.

Section 2.3.2 below will introduce the advantages and characteristics of TensorFlow. Section 2.3.3 will identify the weaknesses of TensorFlow's previous generation of product, DistBrief, and the reason for Google Brain developing a new generation of deep learning platforms. Section 2.3.4 will introduce the direction that the TensorFlow or machine learning model needs to be optimised when running on a small device. The

last part of the article is the existing summary and discussion.

2.3.2 The characteristics of TensorFlow

The abstract representation of the processing model is handled automatically by the framework. This makes TensorFlow and Theano particularly suitable for the development of new models using gradient-based optimisation. The deep neural network architecture, which also includes other types of models, falls into this category. The main disadvantage of Theano is that it takes time to compile the symbolic model. TensorFlow significantly improved this bottleneck. Another advantage of TensorFlow is that it comes with an assistive tool called TensorBoard for in-depth simulation of training progress. The structure of the calculation diagram can be studied interactively, and how the parameters and model performance change in the training iterations (Abadi & Agarwal, 2016). TensorBoard visualization provides a modular representation of a complex model. It facilitates the global representation of the model, debugging, and checking to gain insight during model development (see Figure 2-3).

The primary component in the TensorFlow system is the client, which communicates with the central server using a session interface; one or more worker processes, each of which is responsible for the quorum access to one or more computing devices (such as the CPU kernel or GPU card) and the graphics nodes on those devices that perform the main instructions. The client has a local and distributed implementation of the TensorFlow interface. When clients, primary servers, and workers are running on a single computer in the context of a single operating system process (there may be multiple devices), a local implementation is used, for example, many GPU cards are installed on the computer. The distributed system could share the most of the code from localhost, but TensorFlow extends support for environments where clients, primary servers, and workers can have different processes on different computers. In a TensorFlow distributed environment, these various tasks are containers in jobs

managed by the cluster dispatch system.

TensorFlow is computed using data flow diagrams, where the nodes represent mathematical operations, and the lines in the graph represent the interactions between multidimensional arrays (Tensor). This is described as follows:
Data Flow graph - mathematical calculations are expressed using the nodes and changes of a directed graph. The nodes in the graph represent mathematical operations that can represent the endpoints of data input and output. The edges represent the relationships between the nodes, and the interaction between the transfer operations is tensor in the graph. Once a node is connected to a data stream, the node is assigned to either asynchronous (between nodes) or parallel (within the node) of the computing device.

TensorFlow has the following advantages:
Mobility - TensorFlow is not a regular neural network library. If the user want to express any calculation as a data flow diagram, he/she can use TensorFlow. The user constructs a graph, writes the inner layer loop code to drive the computation; TensorFlow can help to assemble the child graph. To define a new operation, the user only needs to write a Python function.
Adaptability - TensorFlow can be used on different devices, such as CPU, GPU, mobile devices and cloud platforms.
Automatic recurrence relation: TensorFlow's automatic difference function is beneficial to many graph-based machine learning algorithms.
Multiple programming languages available - TensorFlow is easy to use, with Python APIs and C + + APIs. Other languages can rely on the clear wrapper and Interface Generator (SWIG) tool to use the C + + API.
Optimisation Performance - TensorFlow enables the full use of hardware resources, can be the graphic computing units which is automatically allocated to different devices.

2.3.3 The limitations of the previous platform

TensorFlow is a DistBrief successor and a neural network system for distributed systems. DistBrief uses the parameter server architecture, which is limited (Chen & Li, 2015; Jia & Shelhamer, 2014). This architecture restricts the upgrade of these systems, especially in other systems based on DistBrief. In a parameter server architecture, a project consists of two sets of disjoint processes: a stateless worker process that performs most of the calculations while the model is being trained, and a stateful parameter server process parameter that maintains the current version of the model. The DistBrief programming model is similar to Caffe (Jia & Shelhamer, 2014): the user defines a neural network as a forward-free graph of a layer terminated by a loss function. One layer is the composition of mathematical operators; for example, a fully connected layer multiplies its input by increasing the weighting matrix, adding a bias vector, and applying the non-linear function (such as Sigmoid) to the result. A loss function is a scalar function that quantifies the difference between a predicted value (a given input data point) and the ground state truth. In the fully connected layer, the weighting matrix and the bias vector are the parameters, and the learning algorithm is updated to minimise the value of the loss function. By backpropagation (Rumelhart & Hinton, 1988), DistBrief uses the DAG structure and knowledge of the layer's semantics to compute gradients for each model parameter. Because parameter updates in many algorithms are interchangeable and have weak consistency requirements, the worker process can independently compute the upgrade and update its current state. However, the model still has shortcomings, and it uses Python scripting interface to compose a predefined layer, which is simple, fast, but does not have enough flexibility for advanced users: 1) cannot define a new layer; 2) refines training algorithm, 3) defines a new training algorithm. For traditional neural networks, many models use stochastic gradient descent (SGD) as the losing function, but this is not enough for researchers, who want to use more optimised algorithms (such as Mini-batch SGD). DistBrief work follows a fixed execution pattern: loads a batch of input data and current parameter

values, calculates the loss function (passed forward over the network), calculates the gradient (pass backwards) of each parameter, and writes the gradient back to the parameter server. This model is suitable for training simple feedforward neural networks, but for more advanced models (such as recursive neural networks, including loops (Jordan, 1986)), DistBrief cannot achieve the original results. In adversarial networks, two related networks are alternately trained (Goodfellow, 2014); and in the reinforcement learning model, the loss function is computed by some agents in a separate system, such as a video game emulator (Mnih & Kavukcuoglu, 2015). Also, there are many other machine learning algorithms, such as expectation maximisation, decision forest training, and potential allocation, which are not suitable for training the same model as neural networks but can also benefit from a common, optimised distribution. In addition, researchers have designed a new platform for DistBrief: a large multicore server cluster (Dean & Corrado, 2012). They want to be able to increase support for GPU acceleration, which will effectively lift convolution operations (Krizhevsky, Sutskever, & Hinton, 2012). However, DistBrief is still a heavyweight system that is suitable for training deep neural networks in huge datasets and is difficult to extend to other environments. This is especially so because many users want to adjust their models locally on a GPU-driven workstation, and then scale the same code to train on more massive datasets. After training the model on the cluster, the next step is to push the model to production, which may involve integrating the model into an online service or deploying it to a mobile device for offline execution. Based on the requirements mentioned above, Google Brain in 2016 released the TensorFlow. TensorFlow provides a programming model that can also run in all environments.

2.3.4 Acceleration of hardware in machine learning

In 2018 since the release of TensorFlow, the deep learning technology has gradually changed people's understanding and view of machine learning. Using mobile devices to make judgments about the real world - such as facial recognition, text and speech recognition – can be achieved. Unlike traditional models, which rely heavily on local

computing, new technologies increasingly rely on cloud computing resources to get results from the way local computing joins. Even the new computational model requires only a small amount of computation, and even on the handheld device the model can be completed the model and the target classified and synthesised. Lane and Warden . suggested that mainstream learning and classification techniques will appear on embedded devices or mobile devices and that the accuracy and robustness of deep neural networks will also be passed on to home devices, office equipment, automobiles and mobile phones (2018). And because these devices are more human-like, they may even offer better experiences and better performance than before (e.g., in machine translation, image understanding, and speech synthesis). Examples of mainstream learning and classification techniques include Microsoft's seeing AI, a mobile visual processing program that can accurately describe the current environment to visually impaired people. Another example is Babylon from Babylon Health Inc., a medical assistant who can provide adequate medical diagnosis and advice based on the patient's situation.

In order to improve the efficiency of these mobile terminals, Andreas et al. (2018) put forward the model of hardware acceleration, and the new acceleration mode modifies the computational structure. The new structure accepts 16 activations and 16 weights. It multiplies these pairs and then uses the adder tree to reduce the 16 products to add the result to the output register. The hardware can calculate output activation for multiple cycles. Accelerators can use several units to handle more activation and weight pairs per cycle. Because the convolution layer usually has more than one filter, each filter can be assigned a separate unit, and all units reuse the same 16 activations. As memory access is much more expensive than typical calculations in modern semiconductor technology, reusing data is desirable.

DaDianNao is an accelerator based on the structure in Figure 2-4 (Chen, Luo, Liu, Zhang, & He, 2014), using the active reuse advantage in the convolution layer to

balance the computing and communication requirements with the new resources. The DaDianNao contains 256 processing units. Each unit can handle a single filter; in total DaDianNao can handle more than 4,000 products, each of which contains 256 parts. Depending on the settings, different results will be available.

An ideal accelerator would be: 1) specialised enough to provide the desired performance level; 2) versatile enough to support a broader range of application classes; and 3) designed for the future so as to enhance flexibility.

In all the CNN studies, many multiplications are ineffective because they involve 0-value activation. More multiplication operations can be avoided if the activation input value is close to 0. In different networks or layers, this 'close enough' is different. Andreas et al. (2018) developed an empirical approach to finding thresholds for each layer. This invalid multiplication represents an opportunity to improve performance. However, leveraging the performance is the challenge of large-scale data parallel engines. To improve any performance, a method is needed to facilitate other useful calculations to replace this invalid calculation. Unfortunately, just checking whether activation is invalid will cost as much time as doing multiplication, whereas, activation requires another data access. Fortunately, input, in each CNN layer, which is the previous layer of output. Therefore, at the output of each layer, the output place the active activation function tightly in memory so that the next layer of processing can proceed smoothly without having to check for invalid computations or perform additional memory accesses. The Cluvlutin5 is such a design, and Figure 2-4 reports its performance improvements to DaDianNao.

Activation can be considered a probability that an image feature appears in a location unless the image is filled with this feature; most of this feature appears in a low probability, that is, activation is 0 (ReLU) or close to 0 (Sigmoid). A large amount of invalid activation is an intrinsic attribute of a neural network. The efficient inference

engine also skips 0 activations and uses sparse weights to perform a single multiplication accumulation unit. CNN, even for such a sparse neural network, is the only neural network that processes many weights into 0, skipping the invalid weight and activation.

2.3.5 Conclusion

The deep network represented by convolution neural network signalled a breakthrough in the artificial neural network field, which was dormant for many years yet, more recently, has attracted new research hotspots. The feature of the global training, which is based on the local receptive fields, hierarchical structure, feature extraction and classification process of CNN, makes it widely used in the area of image recognition.

CNN is the best-known model of deep learning, which belongs to the discriminant mode and is mainly used in machine vision field. Since CNN has been proposed, deep learning has made rapid progress in the areas of image identification and classification, target detection and localisation, and even played an invaluable role in the field of the human-computer game. The standard CNN is a special feedforward neural network model, which usually has a comparatively deep structure, and is composed of the input layer, convolution layer, pooling layer, full connection layer and the output layer.

TensorFlow is a second-generation machine learning platform developed by DistBrief. It is a relatively high order machine learning library; researchers can efficiently use it to design a neural network structure, without having to write C + + and Cuda code for efficient implementation. TensorFlow supports automatic derivation, the researchers do not need to pass the reverse propagation solution gradient, the core code in C + + simplifies the complexity of the online deployment so that the mobile phone tablet and other memory and CPU resources with sensitive equipment can also run complex models. In addition to the core code of the C + + interface, TensorFlow also supports Python, Java and other interfaces.

3. Research Question and Methodology

3.1 Methodology

In this chapter, the goal is to present the necessary links between all the elements of this study together with its methodology.

3.2 Research Questions

Putting forward research questions is the first step in all aspects of scientific research. This section corresponds to Figure 3-1. Research questions and problem definitions are based on the past and current shoe print identification technology and convolutional neural networks and other technologies explored in Chapter 2's literature review . The purpose is to understand existing methods – such as means of classification and image processing - and their functions in terms of their advantages and limitations.

In this thesis, we have collected a few of the current research results in shoe print identification area. We need a basic research target and link it to existing outcomes in this area. Our work begins with the information we have gathered and the results that others have already researched from our surveys and literature review. This, in turn, enables us to introduce new methods of image classification. For example, in this study's experiment, the convolution neural network is used to classify the shoe print image.

The purpose of this thesis is to explore the identification and classification of the image for shoe prints by using deep learning, convolutional neural networks (CNN). The overall research questions are three.

The first question, what is the performance of traditional pattern classification methods on shoe prints identification?

The data sets are categorised using traditional machine learning methods. When using traditional methods, the accuracy of the classification, the time consumed, the merits

and demerits of the structure will be discussed. The second question: does CNN have better performance on shoe print identification? Using CNN, we employ more massive datasets to classify images. Moreover, the advantages and disadvantages between the convolutional neural network and the original machine learning method are discussed. The third research question of this thesis: can the CNN performance be improved furhter with fine-tuning?

This thesis will also list the direction of research after this research (section 7.2); for example, the classification of the type of shoe print, or the changes from shoe print image collection that comes from a different substance.

3.3 Research Objectives

The goal of these experiments is to achieve the binary classification of images: shoe print or not shoe print. We try to propose an efficient CNN classification model to improve the accuracy of the classification as much as possible and try to improve the robustness of the model. At the same time, the objective of these experiments is to understand the performance of a convolutional neural network when it is not so accurate when dealing with the positional relationship between lines and patterns.

3.4 The Experimental Hypothesis

Research Hypothesis.

First, we hypothesize that previous research in face identification using CNNs is also applicable to shoe print identification.

Second, the researchers hypothesize that the convolution neural network can have excellent performance and accuracy when it identifies the shoe print.

This means that we need to hypothesize that CNN has a significant advantage over traditional machine learning methods such as artificial neural networks.

3.5 Experimental Design

To verify that our hypotheses are correct or incorrect, we have designed multiple experiments and used different methods to obtain the results. Datasets will be tested in artificial neural networks, support vector machines and convolutional neural networks. We also designed an experiment to optimise the neural network and compare it to the previous neural network to detect the results of the optimisation. The specific experimental content will be discussed in the sixth chapter.

3.6 Data Collection and Analysis Tools

A variety of image classification tools are used in this experiment. In the artificial neural network classification, we use MATLAB pattern identification toolbox as the primary research tool. In the study of support vector machine, we used MATLAB classification learner as a research tool because it contains a variety of binary classification algorithms and can be very intuitive to get the results and accuracy of the classification.

The new model was developed in the Python language. Moreover, the new model uses TensorFlow as the primary research tool. Among this tool many highly efficient APIs are included, which can help us to use and analyse CNN better.

Moreover, at the end of the neural network visualisation phase, we used Keras as a visual tool. Compared to the traditional TensorFlow platform, Keras's visualisation function is more powerful.

3.7 Development Testing and Validation of Neural Networks

The stage after data collection and analysis is the instrument development test and instrument validation in the methodology. Implementation is one of the essential steps in this study. At this stage, a new convolutional neural network model is selected and designed. It is based on preliminary experimental research. In the test and validation phase, we use one dataset to discuss and evaluate different methods to study the performance and characteristics of each. The data is inputted into a different method model to verify the performance of its various aspects. In this experiment, the data set is divided into training sets and testing sets for evaluation. The performance of the CNN model will be compared with traditional machine learning methods.
This stage will be discussed in the main text of Chapters 6 and 7.

3.8 Limitation of the Study

The research environment and previous studies limit this thesis such that some research goals may not be achieved. For example, the further classification of the shoe print image and the sample rotation or the classification containing multiple shoe print images are not considered here.
We have put these discussions in the “Future Work” section in Chapter 8.

Another limitation is face identification was used as a guide to the research. This is because, according to the survey, no previous research has used neural networks to classify shoe-print images. This meant that during the survey of research approaches, it was not possible to get a sufficiently valid reference. However, it must be pointed out that there is still a large gap between shoe print and the human face. Both feature extraction and neural network structure, but there are obvious differences. This also leads us to need to repeatedly verify whether the original research approaches in face identification can be used in the new research.

4. Research Method and Techniques

4.1 Artificial Neural Network

Artificial neural network (ANN) is a network structure interconnected by a large number of processing units (neurons), which reflects the essential characteristics of human brain function and is an abstraction, simplification, and simulation of the human brain. ANN information processing is realized by the interaction between neurons; the storage of knowledge and information is the physical relation of network structure distribution, and the learning and processing process of the network depends on the dynamic change of neuron connection weights. Because ANN usually adopts nonlinear function, its dynamic operation constitutes a nonlinear system. The system is characterised by unpredictable, irreversible and multiple factors. Therefore, ANN can simulate large-scale, adaptive, nonlinear complex systems. ANN is widely used in optimisation, pattern identification, knowledge processing, signal processing and other fields.

This experiment uses the pattern identification toolbox in Matlab. ANN is a linear classifier that requires the experimenter to convert the original image into a linear data band and enter the neural network (Figure 4-0). It should be noted that this method is relatively primitive, and the spatial relationship between the upper and lower two pixels is lost in the process of resizing. Moreover, because the neurons of ANN are all connected, there is a large amount of computation after forming the neural network. Before using ANN, we need to add a variety of image preprocessing methods to improve the image identification rate.

The primary purpose of this experiment is to create a baseline value for the subsequent convolution neural network. In previous studies, ANN has been a fundamental tool in image classification. This method can effectively help we understand the advantages of neural networks and the direction of improvement.

4.2 Support Vector Machine

Support vector machine (SVM) is a modelling method based on the small sample, statistical learning theory and structural risk minimisation, which is a research hotspot in the machine learning field after artificial neural network research. In the binary image classification, there is good accuracy. SVM overcomes the shortcomings of other machine learning methods such as overfitting, lack of learning, low generalisation ability and local minimum value. In practical applications, SVM-related parameters generally rely on human experience. The selection of parameters is directly related to the predictive accuracy of SVM.

4.3 Convolutional Neural Network

4.3.1 Characteristics of convolutional neural network

Convolutional Neural Network is one kind of deep neural networks, which combines artificial neural network and deep learning technology. Its particularity is mainly embodied in two aspects as the following paragraphs explain.

First, the neural elements of CNN are not fully connected, and Figure 4-1a illustrates a fully connected network. For images with 1000x1000 pixels, there are one million hidden neurons in each layer of neural networks. Each hidden layer neuron is linked to the one-pixel point in the image, and there is a 1000x1000x1000000=1.00*e*+12 connections. This means that the neural network will have 1.00*e*+12 weights. Figure 5-1b shows a locally connected network, and each node is connected to the 10x10 window near the same location as the previous node, then the one million hidden layer neurons have only 10x10x100000=1.00*e*+08 weights, so that the number of local connection neural network weights will be one out of 10,000 of the fully connected neural network.

Second, in CNN, the weights of connections between certain neurons are shared. In a locally connected neural network, each neuron of the hidden layer is connected only to

the 10x10 image area; each neuron has 100 connection parameters. However, in CNN, these 100 parameters are the same, that is, each neuron uses the same convolution kernel convoluting image, the weights are shared. A convolution kernel provides a feature of an image. If it was needed to extract many different features, users could add several filters. Various parameters of the filter indicate the extraction of different image features. The different characteristics of images can be obtained by using different filters to convolute images. So, 100 convolution kernels have 100 feature maps, and these 100 feature maps form one single layer of neurons. Each convolution kernel shares 100 parameters, with a total of 10,000 parameters.

In CNN, the number of parameters in a hidden layer is related to the size of the filter and the type of filter, regardless of the number of neurons in the hidden layer. The number of neurons in the hidden layer is related to the original image, the size of the filter, and the sliding step length of the filter in the image. For example, the original image is 1000*1000 pixels, and the filter size is 10x10. If the filter does not overlap, that is, the sliding step is 10, the number of neurons in the hidden layer can be (1000x1000)/(10x10) = 10,000 neurons. The non-full connection and weighted value sharing network structure of a CNN reduces the number of weights and reduces the complexity of the network model, which is very important for the deep structure. The three essential ideas of the CNN are local receptive fields, weighted value sharing, and space sampling. These three theories combine to let CNN achieve translational, proportional scaling, skewing or other forms of high-level invariance.

4.3.2 The topological structure of convolutional neural network

Part One:

Convolution neural network is a multi-layered neural network; each layer consists of the two-dimensional plane, each two-dimensional plane is composed of several independent neurons. The network contains some simple neurons and complex neurons.

Simple neurons combine to form S-plane, and S-plane to form S-layer. Similarly, C-planes, which are composed of complex neurons, have similar structures in C-layers. Any intermediate stage in the network is made up of S-layer and C-layer, and the sample feature extraction step is embedded in the interconnected structure of the network Structure.

Usually, S-layer is a feature extraction layer, and each neuron's input is connected to the local receptive field in the previous layer, and the local characteristics are extracted. Once the local feature is extracted, the position relation between it and other local features is determined. C-layer is a feature map layer; each computing layer is composed of multiple features, every feature map is a plane, the weights of all neurons on the plane are shared, which have the same degree of displacement or rotation invariance. Because of the neuron sharing weights of the same mapped polygons, the parameters in the network are greatly reduced. Each feature extraction layer (S-layer) is followed by a computed layer (C-layer) that is used, twice, to extract the local average. This unique two-time feature extraction structure enables the network to have a high distortion tolerance to the input sample when it is identified. The structure of most CNN is similar.

The CNN structure in our experiment is shown in Table 4-1 and Figure 4-2. Except for the input layer, it contains eight layers, which receives a 64x64 pixel image area to determine whether it is a shoe print or non-shoe print. The Conv1 layer to Pool2 layer contains a series of planes that can perform convolution and pooling operations. These planes are called feature maps, and they are responsible for extracting and combining a set of appropriate features. ReLU functions are used as an activation function for each layer in the two convolutional layers C1 and two fully connected layers fc1, fc2. The neural network adds the dropout layer to increase the classification accuracy of the network and reduce overfitting. The last layer of Softmax layer uses the features extracted from the front layer to classify tasks.

Each unit in the same layer accepts input from a small neighbourhood unit in the previous layer. The idea of connecting local receptive field units is largely inspired by Hubel and Wiesel (1962) and their notion of local sensitivities. Neurons can extract basic visual features, such as image edges and endpoints. These features can be combined with subsequent layers to detect higher-level features. Distortions or input displacements can cause significant changes in the location of the feature. Moreover, a primary feature detector that is useful for a part of the image is likely to be useful for an entire image. The units within the same layer are organized in space and all units within that space share weights. Therefore, each feature map had a fixed feature detector, which is equivalent to a trained convolution kernel and applied to the previous layer of space. Each layer uses several feature graphs (with different weights vectors) to allow multiple features to be detected at each location. These features are graphically formed as convolutional layer Ci.

Once a feature is detected, its exact position will become less critical. Its approximate position is relative to other characteristics. The absolute position is likely to be different in different shoe print images. Various features may have different coding positions in the feature map, and a simple way to reduce positional precision is to reduce the spatial resolution of the feature map. Therefore, a lower sampling layer Pi usually follows each convolutional layer Ci; the lower sampling layer performs local maximization and down-sampling operations, reduces the resolution of the feature map and, therefore, reduces the sensitivity to output such as translation, skew, scale change, and rotation.

Part Two:

After confirming the basic structure of the neural network, we have carried out a large amount of research to determine the number of layers, the number of filters, the number of neurons, and the size of the filter. The C1 layer consists of three feature graphs (three channels), each of which is connected to the input image pixels of the 3x3x3. The field

of the adjacent units of each feature graph is concentrated in the corresponding adjacent units of the input image. The size of the feature map is 62x62x3 pixels, regardless of the expansion of the boundary. Each feature plot unit calculates the weighted sum of its inputs, including 27 (3x3x3) coefficients and one bias. Therefore, the Conv1 layer has a 90(3x30) parameters which can be trained. The P1 layer consists of a 64x3 feature map, each corresponding to the C1 layer's feature map. The receptive field for each unit is the 2x2 field of the previous layer corresponding to the feature map. Each unit calculates its four-input means, multiplied by a training bias, plus a training bias, which passes the result to the activation function ReLU. Adjacent kernels have the distinct adjoining field of receptive. Therefore, the feature map of the lower sampling layer has half rows and columns of the previous layer's feature graph. As a result, the Pool1 layer has 64 feature graphs of size 32x32 pixels, and the 128 (64x2) can be trained parameters. The successive alternation of the convolution layer and the lower sampling layer result in the double cone. At each level, the number of feature graphs is increased, and the resolution is decreased.

The Conv2 layer is a convolution layer with a 16x3 feature map. Each element of each feature map connects to a subset of the Pool1 layer feature map at the same location as a 3x3 neighbouring unit. This implementation corresponds to a 3x3x3-trained convolution kernel, which is followed by a trained bias. Here, the output of the different feature graphs is fused to facilitate the combination of different features to extract more complex information. Each of the 64 down-sampled feature plots of the P1 layer serves as input to eight different feature graphs of the C2 layer, which has the first eight feature graphs of the C2 layer. Each of the 64 feature graphs of P1 produces an additional eight features of the C2 layer for every eight combinations.

The P2 layer is an sub-sampling layer with 16 feature mappings. The 2x2 receptive field of each unit corresponds to the area of the above C2 layer feature map, just like P1 and C1. Therefore, P2 has 16 feature maps of size 16x16, 32x3 (16x2x3) can be trained parameters. In the P2 layer, a series of disjoint and stable low-dimensional

features are extracted and used by simple MLP for classification. The FC1 and FC2 layers contain standard neural units. These layers act as a classifier, and the front layers act as feature extractors. In the FC1 layer, each of the 128 neurons is fully connected to all the units of the corresponding P2 feature map. The FC1 units perform a classical dot product operation between their input vector and the weight vector. FC2 and FC1 have the same size and number of neurons. There is a dropout layer after two fully connected layers; the samples passed through the FC2 layer are discarded at a certain scale to reduce the model overfitting. Finally, the data after the dropout layer is Softmax, and the kernel of each classification is calculated at the Softmax layer. The output of the neuron of the Softmax layer is used for classification; if its value is less than 0.5, the input image is nonshoe print, if its value is greater than 0.5, then the input image is a shoe print.

With regard to our learning strategy, all weights are calculated based on gradient learning, using a modified version of the inverse propagation algorithm. The main change here is the calculation of the local gradient of the inverse propagation error signal of the shared weights. Considering that each feature map contains a simple neuron (with multiple instances), the local gradient of the neuron is simply the sum of the local gradients of all its instances. During the training, the network output response is less than 0.5 for non-shoe printing, more than 0.5 for shoe print.

Because all weights can be learned, the system is a feature extractor that synthesizes its specific problems. Although this network structure uses many connections, weight sharing reduces the number of parameters, the computational capacity of the machine, and the gap between training errors and test errors, and thus has a better predictive ability. Local receptive field, weight sharing, and down-sampling provide many advantages to solve two critical problems: robustness and good generalization. It is important, though, to consider the impossibility of acquiring all possible variations of the shoe-print pattern within a limited-scale training set.

The topology proposed has another interesting feature. In most image-based methods,

to search for a shoe print of a given size, the network must copy (or scan) all the images in the image. In our method, there is no local preprocessing sub-window, the original image is passed directly to the network. Because each layer essentially performs small-size convolution in parallel, a large portion of the calculation is the same for each convolution in two adjacent processing windows. This redundancy is eliminated by making convolution in each layer. The whole calculation is included in the transmission path of the convolution and the nonlinear transformation of the whole image. On each layer of the transmission path, the entire image convolution and the nonlinear transform of the small size kernel can be executed in parallel efficiently.

4.4 Derivation of Convolution Neural Network Theory

4.4.1 Neuron

Neurons are the basic elements of neural networks. Neurons have three basic characteristics: weighted, summed, and transferred. Chart x_1 ... x_n represents the input from the previous layer, the line represents the corresponding weight of the input neuron, b is the threshold, $f(.)$ is the activation function, and y is the output component of the neuron. We use net_j to represent the activation of a unit j, x_i represents the input of this unit, and w_{ij} represents the weight of the corresponding input, then it can be expressed as (LeCun, Bottou, Bengio, & Haffner, 1998):

$$net_j = \sum_{i=1}^{n} x_i w_{ij} + b \tag{4.1}$$

Each neuron corresponds to an output component, which is the value that it gets after the activating function and can be represented as:

$$y_j = f(net_j) \tag{4.2}$$

because the output is only active or inactive:

$$f(net) = \begin{cases} 1 & net \geq 0 \\ 0 & net < 0 \end{cases} \quad (4.3)$$

ReLU function is represented as (Glorot, Bordes, & Y., 2011):

$$f(x) = max(0, x) \quad (4.4)$$

$$max(0, w^T x + b) \quad (4.5)$$

4.4.2 Forward propagation

BP algorithm can solve the learning problem of neuron weights in the hidden layer of multilayer feedforward networks systematically. BP algorithm is a kind of natural extension of gradient descent criterion based on error; the basic idea is that the input data is transmitted through the neural network, and finally a result is obtained. If the actual result is not the same as the expected result, the error is propagated in some form through the hidden layer to the input layer. In the process of propagation, the error is assigned to each neuron unit of each layer according to certain rules, and the error of all the neurons in the first layer is formed as a graph (sensitivity map). Each neuron unit will update the weights of each unit according to these errors. The process of correcting forward propagation and reverse propagation is repeated until the input error of the result is reduced to the threshold of our acceptable range, or the number of repetitions reached the predetermined number of times.

We take the three-layer neural network as an example, as shown in Figure 4.5. A three-layer network consists of an input layer, an output layer, and a hidden layer in the middle. Forward propagation is the transfer of data from the input layer to the hidden layer, and then from the hidden layer to the output layer, and finally a result. The neuron state of each layer affects only the neurons in the next layer. If the final actual result is not

the same as the expected result, it will be converted to the error reverse propagation process. The process of training and learning of BP Neural network is the process of forwarding propagation and reverse propagation until the satisfactory result is obtained at the end.

We hypothesize that the input layer has *s* units, the hidden layer has *q* units, the output layer has *c* units, v_{ji} represents the input unit *i* and the hidden unit *j* connection weights. The w_{kj} represents the weighted value of the hidden unit *j* and the output unit *k* connection. The activation function used by the hidden layer is represented by $f_1()$. The output layer is represented by $f_2()$. So, for each value of the input x_i,
Each unit value in the hidden layer is:

$$z_k = f_1\left(\sum_{i=0}^{s} v_{ji}x_i\right), j = 1,2,\dots,q \tag{4.6}$$

The value of each unit of the output layer is:

$$y_j = f_2\left(\sum_{j=0}^{q} w_{kj}z_k\right), k = 1,2,\dots,c \tag{4.7}$$

According to the formula 2.8 and formula 2.9, we can get the approximate output of BP network once. The above operation is equivalent to obtaining a mapping function that maps the *s*-dimensional space into a *c*-dimensional space.

4.4.3 The backpropagation

After a forward propagation, we need to measure this result. We define an error to describe the state of this network. The process of reverse propagation is to pass the error to the previous layer so that each neuron in the upper layer can update its weight according to the error condition. We use the squared error cost function. According to Bouvrie's research (2006), for multi-class problems with *c* classes, *N* training samples, the error function is:

$$E = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{c}(t_k^n - y_k^n)^2 \tag{4.8}$$

Where t_k^n is the *k*th dimension of the corresponding label of *n* samples, y_k^n is the *k*th output of the corresponding network of the nth sample.

The error of all training sets is only the sum of the two errors of each training, and, in line with Bouvrie (2006), the error of the *n*th sample is expressed as:

$$E^N = \frac{1}{2}\sum_{k=1}^{c}(t_k^n - y_k^n)^2 = \frac{1}{2}\|t^n - y^n\|_2^2 \quad (4.9)$$

The backpropagation algorithm is based on a gradient descent algorithm, the purpose of calculating the global error is to adjust the unit weight to the direction of the error reduction. The current output layer can be expressed as:

$$x^l = f(u^l), u^l = W^l x^{l-1} + b^l \quad (4.10)$$

Backpropagation error is the basic sensitivity of each neuron; that is, how much *b* changes, thus how much the error will change. The error is based on the rate of change. We use ∂ to denote the base sensitivity:

$$\frac{\partial E}{\partial b} = \frac{\partial E}{\partial u}\frac{\partial u}{\partial b} = \delta \quad (4.11)$$

Because $\frac{\partial E}{\partial b} = 1$, so

$$\frac{\partial E}{\partial b} = \frac{\partial E}{\partial b} = \delta \quad (4.12)$$

We can roll out, the base sensitivity $\frac{\partial u}{\partial b} = \delta$ and error *E* are equal to the reciprocal $\frac{\partial E}{\partial u}$ of all input *u* in a node. This countdown allows high-level errors to be propagated back to the lower levels. Bouvrie's (2006) relational formula is:

$$\delta^l = (w^{l+1})^T \delta^{l+1} \circ f'(u^l) \quad (4.13)$$

Here "。" represents the multiplication of each element. According to formula 4.13, the neuron sensitivity of the output layer can be expressed as:

$$\delta^l = f'(u^l) \circ (y^n - t^n) \tag{4.14}$$

Finally, using *delta* rules to update the weights of each neuron, for the first *l* layer, the derivative of the error for each weight of the layer is the cross-multiplication of the input and the sensitivity of the layer. The learning rate of the partial derivative multiplied by the negative number is the weight of this neuron and is updated.

$$\frac{\partial E}{\partial w^l} = x^{l-1}(\delta^l)^T \tag{4.15}$$

$$\Delta w^l = -\eta \frac{\partial E8}{\partial w^l} \tag{4.16}$$

4.4.4 Convolution layer gradient calculation

Reverse propagation will update the convolution layer; in the convolution layer, the previous layer of the feature map using the convolution kernel can be learned, and, from the consequent results of the activation function, we can get the output feature map. Each output feature graph may have multiple input feature graphs combined.

According to Bouvrie (2006), usually there is:

$$x_j^l = f\left(\sum_{i \in M_j} x_i^{l-1} * k_{\mathrm{ij}}^L + b_j^l\right) \tag{4.17}$$

Where x_j^l represents the *l*-layer *j* feature diagram, $f(.)$ represents an activation function. M_j represents a collection of input graphs, * represents convolution operations, *k* for convolution kernels, and *b* for biasing items. The more common combinations of input feature graphs are, “one to two”, and “one to three”. One to two means that a feature graph on the current layer has a combination of two feature graphs from the previous layer. One to three means that a feature graph on the current layer is composed of three feature graphs from the previous layer. Each output feature graph will be added a bias item *b*, for different output mappings, the input uses different convolution kernels. In other words, although the output mapping *j,k* is obtained by summing the input mappings *i*, they do use different convolution kernels for the input mapping *i*.

We assume that each convolutional layer *l* will be followed by a lower sample layer *l*+1. According to the backpropagation algorithm, to calculate the sensitivity of the *l*-layer neurons, it is necessary to multiply each element in the next layer with the function defined on the l+1 layer and then sum the values. The sum is then multiplied by the activation function of the current layer, it less than the partial derivative of the input *u*. When the convolution layer is followed by a lower sampling layer, a pixel in the sensitive graph δ is connected to a neuron in the convolution output map. To better calculate the sensitivity of the *l* layer, we can sample the residual plot of the lower sample layer so that the two layers are the same size. Then it will take the sample after the sensitive graph δ multiply the *l*-layer activation function of the partial derivative. And the weight defined in the next sample layer is β. Therefore, when calculating residuals, it is also necessary to enlarge the result to β-times. Each feature graph *j* in the convolution layer is computed once and corresponds to the following sample layer (Bouvrie, 2006):

$$\delta_j^l = \beta_j^{l+1}(f'(u_j^l) \circ up(\delta_j^{l+1})) \tag{4.18}$$

The $up(.)$ in formula 4.18 is the up-sampling operation, and the process is to put each element of the input layer in a simple n-repetition in both vertical and horizontal directions, where n is the sampling from a multiple at that time.

$$p(x) = x \otimes 1n \times n \tag{4.19}$$

For the sensitivity of known graphs, we can directly calculate the gradient of the bias of the feature graph in the current layer according to formula (4.11):

$$\frac{\partial E}{\partial b_j} = \sum_{u,v} (\delta_j^l)_{uv} \tag{4.20}$$

According to the principle of reverse propagation, we can calculate the gradient of all the weights of the kernel in the same way as the method of the biased term:

$$\frac{\partial E}{\partial K_{ji}^l} = \sum_{u,v} (\delta_j^l)_{uv} (P_i^{l-1})_{uv} \tag{4.21}$$

wherein, *(P)* indicates that the kernel *K* is multiplied from one area in the convolution process of the feature map *xj*.

4.4.5 The gradient calculation of the sub-sampling layer

After the feature is obtained by convolution, a classifier is chosen to use these characteristics to classify. Although, theoretically, we can use all the extracted features to train the classifier, such as the Softmax classifier used in this Project, this poses a computational challenge. For example, for an image of a 100x100 pixel, assuming we have 400 features defined on the 10x10 input, then each feature an image that will be convolution with one (100-10+1) x (100-10+1) = 8281-dimensional convolution feature, and we have 400 features, so each sample gets a convolution eigenvector of the 8281x400 dimension. Training a classifier for such a large feature consumes a lot of resources and is likely to have an overfit.

To solve this problem, we need a way to use a small number to represent so much data. Considering that we use convolution features to represent an image because the image has a 'static' property, this means that features that are useful in one image area are still useful in another area of the image. To describe larger images, it is common to aggregate the characteristics of different locations. The most common method is to describe the area using the average or maximum value of a feature in the image area. These summary statistics features not only have a much lower dimension but also improve the results and can effectively avoid the case of overfitting. This kind of operation is called 'pooling'. Depending on the method used, it can be divided into average pooling and maximum pooling.

The usual use of convolutional neural networks is the maximum pooling, which is a nonlinear down-sampling. The basic idea of max-pooling is to divide the image into rectangular regions that have no intersection with each other and output the maximum

value for each region. The max-pooling has two advantages which ensure that it is an excellent next sampling method: (1) reduces the complexity of upper layer computation; and (2) provides translation invariance.

The sub-sampling layer is the result of sampling under the input image. If there are 'n' input feature graphs, it will produce 'n' output feature graph; the output feature will be much smaller than the input feature graph (Bouvrie, 2006):

$$x_j^l = f\left(\beta_j^l down\left(x_i^{l-1}\right) + b_j^l\right) \tag{4.22}$$

where *down (.)* represents the sampling layer. If it is divided into a few *n* x *n* areas, the sample of the image size in different dimensions is the original 1/*n*. Also, each output layer will have a bias β and an additional bias *b* that can be multiplied.

The difficulty of the gradient calculation of the sub-sampling graph is to calculate the sensitive graph because there are only β and *b* parameters that can be learned here. It is assumed that the upper and lower layers of the sampling layer are convolutional layers. If the next sampling layer is an all-connected layer, then this sensitive graph can be obtained by the inverse propagation algorithm. When we calculate the gradient of the nucleus, we need to indicate which area of the input image corresponds to the specified pixel of the next layer. The weights that are multiplied by the input and output are the weights of the convolution kernels, so it is also possible to use formulas (4.23) (Bouvrie, 2006):

$$\delta_j^l = f'\left(u_j^l\right) 0 conv2\left(\delta_j^{l+1}, rot180\left(k_j^{l+1}\right), 'full'\right) \tag{4.23}$$

in the process of calculating β and *b*, the gradient of the additional bias still uses the formula 4.24 to calculate the sum of all the elements of the sensitive legend:

$$\frac{\partial E}{\partial b_j} = \sum\nolimits_{u,v} \left(\delta_j^l\right)_{uv} \tag{4.24}$$

Multiplied by the bias β and the original bottom-sampled graph of the current layer in the forward propagation (no additional biased lower-sample plots) means that saving

the map during forwarding propagation can be easier to calculate. We define a d_j^l to represent the original sampling layer:

$$d_j^l = down\left(x_j^{l-1}\right) \tag{4.25}$$

then, the biased β can be calculated by the formula:

$$\frac{\partial E}{\partial \beta_j} = \sum_{u,v} \left(\delta_j^l o d_j^l\right)_{uv} \tag{4.26}$$

4.4.6 Combination of feature graphs

Sometimes, when generating a feature map, the feature map is composed of different input layers according to a certain set of rules, which is very advantageous to the effect of generating graphs. In most cases, however, most of the input graphs used to produce the output diagram are manually selected. During training, how is the fusion method learned?. Following Bourvie (2006), here the a_{ij} is used to represent the weight of the input graph *i* when generating the output graph *j*, then the generation of the generated graph *J* can be expressed as (Bouvrie, 2006):

$$x_j' = f_{\left(\sum_{i=1}^{N_{in}} a_{ij}\left(x^{L-1} * k_i^1\right) + b_j^l\right), S \cdot t \cdot} \sum_i a_{ij} = 1, and 0 \le a_{ij} \le 1 \tag{4.27}$$

where the constraints can make a_{ij} equal to a set of related weights c_{ij} to Softmax function of the demerit, thus the formula is established:

$$a_{ij} = \frac{exp(c_{ij})}{\Sigma_k exp(c_{kj})} \tag{4.28}$$

Because the weight set c_{ij} of the constituent diagram *J* is independent of similar sets in other layers, we can say subscript *j* when considering a single feature map. Each diagram is updated in this way unless the index of *j* is different. Bouvrie (2006) instructs that the partial derivative of the Softmax function can be expressed as:

$$\frac{\partial a_k}{\partial c_i} = \delta_{k_i} a_i - a_i a_k \tag{4.29}$$

where δ is usually represented by the Kronecker increment, in the layer l, the derivative of a_i is shown in the formula 4.30:

$$\frac{\partial E}{\partial a_i} = \frac{\partial E}{\partial u^l}\frac{\partial u^l}{\partial a_i} = \sum\nolimits_{u,v} \left(\delta^l o\left(x_i^{l-1} * k_i^l\right)\right)_{uv} \tag{4.30}$$

where δ represents the sensitive graph corresponding to the input layer u. According to the chain rule, we take the final error calculation to the derivation of all weights to get the sensitivity of unit:

$$\frac{\partial E}{\partial c_i} = \sum\nolimits_k \frac{\partial E}{\partial a_k}\frac{\partial a_k}{\partial c_i} = a_i\left(\frac{\partial E}{\partial a_i} - \sum\nolimits_k \frac{\partial E}{\partial a_k} a_k\right) \tag{4.31}$$

5. Experiment Design and Experiment

5.1 Dataset

The dataset used in this experiment is in two parts. The first one was obtained by the researcher through an internet search. The dataset contains a total of 40 pairs of images, including the image of the shoe print and 20 images that do not contain a shoe print. The shoe print has been manually cropped to 50x110,70x176,90x227 - three sizes to produce three datasets to test the efficiency of each method. The design of three sizes is mainly due to the different size of the data set which will greatly affect the operation speed of the neural network. In particular, in the artificial neural networks and support vector machines that will be introduced later, different sample sizes produce completely different results. Also, this dataset will be used to verify the accuracy of convolutional neural networks.

For the second dataset, when training deep neural networks, we used FID-300 datasets from the University of Basel Switzerland (provided by Adam Kortylewski and Thomas Vetter). This dataset was originally used to train compositional Active Basis Model to identify shoe print images in complex background images. The dataset contains a total of 1,175 images for model training. The image is a high-precision shoe print image. There is only shoe print in the image, and the image size is about 200x700. The image is manually labelled as a ‘shoeprint’. Also, the shoe print labelled ‘nonshoeprint’ was collected by researchers from the internet, with a total of 801 samples labelled ‘nonshoeprint’. Therefore, the total number of dataset samples using FID-300 is 1,976.

5.2 Artificial Neural Network

5.2.1 Experimental Design Principles

The main pretreatment process of the artificial neural network involves several image processing steps. This is because all the neurons in the neural network belong to the fully connected structure. As the number of neurons increases, the computational capacity of neural networks increases greatly. It requires a variety of image processing

methods to reduce image size and computational capacity. In the MATLAB pattern identification toolbox, there are too few parameters to adjust, only the number of neurons can be adjusted. The main means to simplify the neural network is to simplify the input values. The original image is an RGB three-channel image. When judging whether the image is a shoe print, black and white images can also produce outstanding results. We tried to convert the original image into a grayscale image, and then convert it to a two-value image after median filtering. The image leaves only one channel, and the image occupies approximately 70% less space. The complexity of the image was greatly reduced.

The change in image size is also a means to greatly reduce the complexity of neural networks. The original image is approximately 100KB, and the original image size in the dataset is not the same. Such datasets cannot be used as input data.

Each data set will contain 40 images, half of which are marked with a shoe print, and 20 are images that do not contain a shoe print. The original image is resized into different sizes to detect the performance of the neural network.

5.2.2 Experimental Design

Image preprocessing:

Step one: Convert original image to grayscale.

The original image is a colour picture; on the RGB three channels, each has value, each pixel on three channels has the values of 0-255. Convert the colour image to grayscale, sum the values on the three channels with a coefficient to get the grayscale value for each pixel (Formula 5.1).

$$gray() = (r \times 0.299) + (g \times 0.587) + (b \times 0.114) \qquad (5.1)$$

Step two: Convert grayscale images into black and white images.

To further reduce the image size and reduce the computational amount of the neural network, convert the grayscale image into the black and white image. The grayscale value of the pixel in the grayscale image uses a threshold value of 127 for two. Pixels

greater than 127 are represented by one, and values less than 127 are replaced with 0, so there are only two values, 0, or 1 for each pixel in the image. The black and white image is written to a new file after median filtering, creating a training set.

Step three: Convert the training set into a table that can be entered into a neural network.

The artificial neural network cannot directly accept the two-dimensional image. It is necessary to re-convert the two-dimensional image into a one-dimensional sequence to be a sample input neural network.

Image training

This experiment used the pattern identification toolbox, which can be modified in the toolbox with only the number of neurons and the size of the dataset. So we designed three datasets to monitor the efficiency of the neural network, resulting in a best-in-use result.

The experiment was designed to train three neural networks in the same set of parameters, record the accuracy, and finally calculate the average value - 75% in a dataset as a training dataset, 20% as a validation set, and 5% as a test set.

Modifiable parameters are the number of neurons in a neural network. We designed the different number of neurons in several groups and obtained the approximate accuracy interval by a few tests. When a more precise range was obtained, the specific number of neurons was then selected.

First group: the dataset used is 50x110, the number of neurons used is 50,80,100;

Second group: the dataset used is 50x100, the number of neurons used is 200,300,400;

Third group: the dataset used is 50x100, the number of neurons used is 3300,3500,4000;

Fourth group: the data set used is 70x176, the number of neurons used is 6200,7000,7500;

Fifth group: the dataset used is 90x227, the number of neurons used is 10000;

Sixth group: the dataset used is 90x227, the number of neurons used is 5000,5500,6000,7000,8000.

In the first two groups of experiments, we found that when the neurons are too few, the exact value of the neural network is too low, or even lower than the random value (because it is only a binary classification, to determine whether the image is a shoe print, random value of 50%). We changed the original neuron increase strategy so that the number of neurons increased significantly.

Specific results pertaining to these experiments will be discussed in Chapter 6.

5.3 Support Vector Machine

In this experiment, the purpose of SVM was mainly to put forward a benchmark, which was then used to judge the merits and demerits of convolutional neural networks.

The data we use is 90x227. We used the classification learner toolbox in MATLAB as an instrument for this experiment. The test set used was a dataset with a picture size of 90x227. Because SVM and ANN have the same requirements for image preprocessing, it was necessary to reduce the information contained in the image as much as possible. So, we directly passed the processed data in the ANN into SVM. Moreover, 5-part cross-validation was used.

In this experiment, we tried a variety of SVM methods: Linear SVM, quadratic SVM, Cubic SVM, Fine Gaussian SVM, Medium Gaussian SVM and Coarse Gaussian SVM. A variety of functions represent different classification methods. The experiment was repeated several times to determine the correct rate for each classifier.

5.4 Convolutional Neural Network Experimental Design

Through the means of both the previous artificial neural network and support vector machine, we obtained the following accuracy: the artificial neural network has 89% of the correct rate; and the support vector machine classification accuracy reached 95%. We first need to verify whether the accuracy of CNN can reach or exceed 95%. A total of 40 datasets used in previous artificial neural networks and support vector machines were used to verify that the neural network works correctly. The second step uses the FID-300 dataset, which contains a total of 1,976 samples, and is employed to test the learning ability of neural networks.

We now select the neural network that consists of two convolutional layers, two pooled layers, and two fully connected layers. However, the existing neural network structure is based on past studies, and it cannot be concluded that this structure is the best. After the CNN classifies accuracy that exceeds the artificial neural network and the support vector machine, we will try to improve the existing CNN in terms of training accuracy and learning efficiency.

During the first training of CNN, the setting of different hyper-parameters will be the main direction of adjustment. The learning rate, batch size, and maximum training times will be the main adjusted hyper-parameters. Moreover, in the further adjustment, the optimisation model is the primary experimental target, at this time the number of neurons in the whole connected layer and the number of fully connected layers will also be adjusted parameters.

Our existing structure consists of two fully connected layers. Of these, there are 128 neurons at each fully connected layer. The main experimental direction of the current experiment is to change the structure of two full connection layers and the number of neurons. According to previous studies, full-attached layer computing would consume

much computational time and computer resources. Simplifying the full-connection layer can effectively increase the computational efficiency of the entire neural network; the number of neurons decreased from 128 to 64,32,16,8 and 4. The idea is to select one or two for the number of fully connected layers. In this way, the model needs to run 12 times altogether.

We have been able to use the artificial neural network to get 90% classification accuracy, and the support vector machine to get 95% of the classification accuracy.

As per Figure 5-1:

b) In the convolutional neural network model, our base is the correct rate of more than 95%, which need to be higher than two baseline approaches. In this experiment, we use the convolutional neural network to identify and classify the success rate of more than 95%. We first use the image size of 90x227,40 and a map of the data set – used to verify the neural network model – can be run.

d) Otherwise, we would need to fix the code constantly.

c) After successful verification, we use FID-300, which contains 1,976 images of the data set. A better result can be achieved with multiple workouts.

f) If the accuracy is less than 95%, we would need to do some image preprocessing on the samples in the FID-300 data set. The images in this dataset are all original colour images; they are not black and white images as they were previously processed. Image preprocessing can simplify the difficulty of image classification. Images preprocessing methods includ but not limited to use grayscale image, black white image, image enhancement methods.

e) When the experimental success rate exceeds 95%, we can try to keep the existing precision in the case by simplifying CNN models. If the accuracy after simplification can be improved, the new model is a more optimal solution.

h) If the result does not reach the original correct rate after simplification, we can try to use a more complex model to observe the change in accuracy. Such an experimental process can ensure that we can get a more appropriate neural network model.

6. Result and Discussion

ANN and SVM have been used to provide benchmarks because they are available in open source toolboxes, with clear experiment steps and mathematical theory, and, as reported here, good identification accuracy. ANN is fundamental in image classification. SVMs are useful for dealing for overfitting problems. They provide two well established pattern recognition technologies for baseline results.

We did not compare against SIFT/RANSAC, because the paper which used SIFT-RANSAC does not provide the whole dataset and does not describe the methods adopted to obtain the results. There were only basic mathematical formulae and results, and so this approach does not have the condition of reproducibility. In addition, our ANN accuracy reached 90% and SVM reached 95%, which is not much worse than SIFT-RANSAC (91%-96%).

6.1 Artificial Neural Network (ANN)

We tested six sets of data altogether. Each group of data was tested three times separately. There is recording the training set and testing set accuracy after each training. As shown in Table 6-1, when trained in the first and second group, the number of neurons was below 500. The results were poor. For first group result, the accuracy was only about 25%. For the second group setting, the accuracy was about 30%. This result was even lower than the random selection (50%).

In the third group of tests, the number of nodes in the neural network was increased from less than 500 to about 3,000. The accuracy of the training set was improved dramatically. It can be noticed that with the number of neurons increasing from 3,300 to 4000, the accuracy of the training set increased from less than 50% to about 70%, which is a very substantial improvement. Using the 70x176 dataset in the fourth set of tests, we tried 6,200, to 7,500 of neurons. The test results were slightly better than random selection, which is 50% accuracy, but this number is not enough for machine learning images classification. In the fifth group tests, we used the 90x227 data set. Moreover, the number of neurons selected for the toolbox can be supported to the maximum value, 10,000. After being trained twice, the accuracy of the training set was

above 70%, which is better than most of the previous tests. The above five groups of tests, all for the pre-test, were designed to get a more appropriate combination of dataset-neurons.

Based on the previous five group tests, a general experience can be obtained. That is, the number of neurons is about one-third in all parameters, which will get a higher accuracy of results.

For the sixth group tests, we used the data set containing the most significant size images of the three datasets, 90x227 pixels. This dataset contains a total of 20,430 variables per sample. Each variable has only two values, 1 or 0. It takes the total variable number of 1/3 values, from 5,000 to 8,000; we designed five sets of tests, each set of configuration training was run three times and counted the average value. The results obtained are set out in Table 6-3.

With the 5,000 neurons setting, the accuracy increased gradually. After exceeding 6,000 neurons, the accuracy of the artificial network started to reduce. When the neural network had 5,500 neurons, we got the best results of all artificial neural networks. The accuracy of classification in the training set is 90%, and in the testing set is 83%.

5,500 neurons are roughly in line with our previous assumptions. That is, in the binary classification of shoe prints, the number of neurons is about one-third of the total number of variables.

However, the disadvantages of artificial neural networks are apparent. First, the computational time is extended due to large-scale computing. In the sixth group

experiment, the running time of each training was about 30 minutes. This time is much more than the CNN spent. Second, because the learning rate is not adjustable, artificial neural networks usually appear as ‘reached the minimum gradient’, even if they spend only one or two epochs. When the number of neurons is few, the neural network would have a considerable iteration time; for example, with the number of neurons at 200, the neural network will run 2,700 times before it stops. However, the accuracy is not optimal. Third, the artificial neural network is prone to overfit or falling into the ‘local optimal solution’. It will result in a low accuracy result after a long period of computation.

6.2 Support Vector Machine (SVM)

We used a variety of support vector machines, but the accuracy was all 95%. We used cross-validation methods to improve the correctness of the classification, similar to artificial neural networks. In a support vector machine, each sample in the dataset will use all the pixels in the image as the categorical variable. The picture size is 90x227; that is, for the image there will be 20,430 categorical variables.

95% accuracy means there is the total number of 40 samples of the classification, 38 are correctly categorized. There are several main reasons for this. First, the different algorithm of support vector machines is essentially different forms of the same binary classification. Therefore, the same dataset classification results will not produce too much deviation. Second, the different SVM classifications are mainly between ‘false positive’ and ‘true negative’. In other words, there may be two, ‘true negative’ or two ‘false positive’, or a ‘true negative’ and a ‘false positive’ (see Figure 6-4). However, the classification of ‘true positive’ is the same. Third, the result of image classification is similar because of multiple images preprocessing. If grayscale images are used, then the calculation cost will be greatly increased. That is what we do not want to see. 95% is a relatively high classification accuracy – exceeding the artificial neural network’s

highest value – thus it provides a new benchmark for the subsequent CNN model.

However, support vector machines still have relative weaknesses compared to neural networks. For example, support vector machines are only good at binary classification. They are weak at handling the complexity output model, not even as good as artificial neural networks. This is detrimental to our subsequent research. Another knotty problem is the traditional machine learning structure is simple, resulting in a very long computational time when dealing with complex image classifications. And, traditional machine learning methods rely heavily on excellent image preprocessing. The result of image preprocessing will affect the result of image classification. Moreover, the method of image preprocessing contains too much uncertainty. For example, in this experiment, we could only prove that using black and white images reduces the computational capacity of machine learning. However, it is not possible to prove that the result of using grayscale images is better or worse than using black and white images. We were not able to extract a standard process of image preprocessing, which is one of the reasons why traditional machine learning methods are very unstable.

6.3 Convolutional Neural Network (CNN)

6.3.1 Experimental discussion

2.Use of convolutional neural network.

The advantages of classification are clear (see Figure 6-5). After a thousand iterations, the accuracy of neural network classification reached 97.48%. Dealing with the daily binary classification problem produced relatively excellent results. The success rate of classified shoe prints and non-shoe prints was compared with the previous two machine learning methods. There was much progress made.

Figures 6-6 to 6-11 provide some visual examples to illustrate the performance of CNN in shoe print classification.

In Figure 6-6, it is evident that most images could be classified correctly, and the accuracy of classification is very high, at least 95%. However, a few images could not be correctly categorised. This is not to say that the image prediction rate fluctuated around 50%, but that the image was completely defined for another category. Figure 6-7, for example, was not a shoe print at all but was considered a shoe print with a chance of more than 95%. In Figure 6-8 it was a shoe print, but the model inferred that there was 87% probability that was not a shoe print. The probability of such a situation was very low; in the verification of 40 images, only two images had a classification anomaly. This may have been due to activated convolution kernel over-learning. We also verified that the partial shoe print images could be correctly categorised. As shown in Figure 6-9, the prediction accuracy of the testing image was about 97%. However, when the image left only the sole part, the correct rate of prediction decreased to 73%. When the sample retained only the heel portion, the prediction accuracy was only 60% (Figures 6-10 & 6-11). This can be explained by the fact that the sole part contains more information and activates more convolution cores. Correspondingly, the heel does not have that much information.

2. CNN optimization

From Table 6-4, Figure 6-12a and Figure 6-12b, neural networks with two full connection layers (Model A) were more complex than the single full connection layer model (Model B), but the results did not improve. Model A reduced the identification accuracy of the neural network. When neurons were 128, the neural network identification accuracy of one single fully-connected layer reached 99.91%. The CNN with two layers of full-connection layers (Model A) had a identification accuracy of only 97.48%. If the accuracy difference between the two neural networks is 2% then that is insufficient to reflect the advantages of Model B. The model B still had a lower data loss rate. In terms of the above two parameters, data loss and accuracy, Model B performed better than model A (see Figure 6-13).

With the neurons decreasing, the identification and classification of the two neural networks - accuracy and data loss rate - became worse. In model A, the identification accuracy was reduced from 97.48% to 62.76%. This result is not even as effective as artificial neural networks or support vector machines. The data loss rate also increased from the previous 0.11 to 0.64. In Model B, with the decrease in the number of neurons, the accuracy of the neural network classification also declined. However, the decrease was tiny compared to the previous neural network. Similarly, the increase in data loss was also small, from 0.03 to 0.27.

From the two graphs in Figure 6-13, Model B had a faster function convergence efficiency than Model A. The number of training iterations increased to about 200 times. The accuracy of the classification in Model B exceeded 95%, whereas Model A's classification accuracy was only about 75%. Although at the end of the training, the accuracy and data loss of the two neural networks were similar.

Through graphs and tables (see Table 6-4, Figure 6-12, 6-13), we can see some of the following problems. First, complex neural networks do not necessarily provide better results or classification accuracy. It is also not possible to allow neural networks to reduce data loss. Second, complex neural networks consume more computational resources. This also means that they consume more computing time. However, the computational efficiency of the whole neural network decreased. Third, more complex neural networks will lose more data when the number of neurons encountered decreases, the accuracy of the decline will be more serious.

Table 6-4, Figure 6-12, 6-13 also showed the neurons in full connection layer, which before the Classified layer of the neural network, will be very important.

The researcher tried to use 512 neurons in Model B. The results of the experiment are illustrated by the graphs in Figure 6-14:

First, it can be seen that increasing neurons will significantly improve the convergence speed of neural network training. Compared with the convergence scale, where the neurons numbered only 128, when the neural network training is around 100 steps, the classify accuracy with 512 neurons exceeded 85%. However, for the Model B with 128 neurons accuracy is less than 80%. Second, these two neural networks, in terms of the result, are similar. Both the classification accuracy and data loss are very similar. Thirdly, while in neural network training, the function had faster convergence speed, in some situations, was not always better. When the accuracy scale quickly converges to a value, after this point, the training is very likely overfitting. The neural network needs to be reduced the learning rate. In this training, we found that a neural network with 512 neurons was overfitted more than once. When training 700 steps, the accuracy of classification was close to 100%. However, in 750 times, there was an over-fitting situation, the accuracy had a significant decline, and then re-ascension. At the last recording point, the value appeared as a sign of decrease.

Compared to Model B with 128 neurons, 512 neurons were overfitted more often. The final result was no better than Model B.

Based on our previous research result and discussions, CNN model B, which contains one layer of fully connected layer and 128 neurons, has obtained the best identification performance. Model B can achieve the goals set at the beginning of the project. When the shoe printing image is identified, the identification accuracy can be achieved by 99.91%.

6.3.2 Neural Network Visualization

Layers visualization

We tried to visualise the neural network and expected to get the analysis and discussion of the neural network from different angles. Neural network visualisation is a very effective method for understanding neural networks, especially deep neural networks. In the deep neural network, because the model is too complicated, the actual means are not enough to let us understand what the machine learns at every step. We are trying to output each layer data in the neural network. Associated with these outputs, it allows us to understand the functions of various layers, filters, and convolution kernel in a neural network. We need to understand how the designed model works and how it functions.

We used the Keras tool to visualise existing neural networks (see Figures 6-15 to 6-18). we visualised in the neural network, what characteristics are extracted are, what is filtered out, and what the output is.

The first image (Figure 6-15) shows the convolution neural network. For different convolution layers, for one picture, the information collected was different. Some convolution kernels were responsible for collecting image colour information; some focused on extracting edge information from images.

At conv1 layer, activation preserved almost all the information from the original image. As the neural network goes deep, activation became more and more abstract, and visually tricky to understand. We cannot see visually that this is a shoe print. The machine began to have a more complex extraction of the shoe print in the image, heel

or toe. The machine had a higher level of concept to learn. Higher levels of activation mean learning less information from the visual content, but the image category related to more and more information. The absorption of activation increased as the depth of the neural network increased. In very deep layers, more kernels were not activated, which also meant that in the input image, there was no pattern that the filter could represent.

Though these visualisations (Figure 6-15 to 6-18), we can understand the deep neural network and that the learning process has an important feature. The features that can be extracted from each layer became more and more abstract as the layers went deeper. For a particular input image, as the layer got darker and deeper the activation output contained less information. However, information is more relevant to the goal. A deep neural network is like a pipeline of information. The raw image data was then filtered repeatedly, filtering out irrelevant information, such as a specific background and visual appearance, from useful information that is magnified and refined. This is similar to the way humans and animals perceive the world. Observing a scene, a few seconds later humans can only remember which abstract object it is, for example, is this a bicycle or a tree? The human does not retain the exact appearance of this object. The brain has learned to completely abstract the visual input of this information and translate it into more advanced visual concepts, while filtering out irrelevant visual details. In this way, human memory always finds it difficult to remember surrounding subject matter or other background information. Machine learning is also similar, the weight of the subject features are steadily strengthened, filtering out the unimportant features.

Kernel visualisation

We visualised the filters in the neural network (see Figures 6-19 and 6-20). The process of convolution is the process of extracting features. Each convolution kernel, or multiple convolution kernels, represents a feature. If an area in an image is more similar to a convolution kernel, the region is closer to the convolution kernel. So, we need to find in an image - the largest output of the convolution kernel. The specific

implementation method is:

Input A noise image (64x64 pixels) *image*, to find a convolution kernel *kernel_filters* to the image gradient:

$$gradients = \frac{\partial(kernel\_filters)}{\partial(image)} \quad (6.1)$$

Then:

$$image = image + learn\_rate * gradients \quad (6.2)$$

Outputs:

We can see from Figure 6-19 and 6-20 that in this neural network, the two convolution layers have different layers of filters so represent different meanings. In the first layer, filters are more like an edge-to-colour extractor. The different edge and colour characteristics from the image were extracted. In the second layer, the filter is more focused on the relationship between the line and the pixel point. The different lines in the image are convoluted and extracted.

Both figures also show that in different layers, the division of the convolution kernel is different. If we try a more complex neural network, the theoretical convolution kernel will extract more abstract information.

Heat map

The heat map can be reflected in CNN (see Figure 6-21a, 6-21b, 6-22, 6-23) with different weights of different convolution kernels. The figures shows how the model made the decision; The model relied the activating feature from feature map set. The more reddish-coloured areas show where the filter region was activated. Moreover, it has larger weight. The features extracted from the convolution kernel of this area are also more critical for classification.

In the dataset that does not contain a shoe print, the filters in the upper right-hand corner of the image were activated, which is also the essential feature source for the image to be classified as ‘nonshoeprint’.

6.4 Summary

Table 6-5 summarises the differences in terms of accuracy, specificity, and sensitivity of three machine learning methods. The value of accuracy increased in the four models established by three different methods. In other words, the classification ability of the model increased. The accuracy rate for ANN was 90%, SVM accuracy was 95%, CNN Model A’s accuracy was 97.48%, CNN Model B’s was 99.91%.

Sensitivity represents the proportion of the true result in all positive samples that were classified, measuring the ability of the classifier to recognize positive samples.
Specificity represents the proportional true result of all negative samples, measuring the classifier ability of the classifier for negative samples. Precision, sensitivity and specificity were also constantly increasing. The sensitivity of CNN-Model B reached 100%, and all the positive samples were correctly classified. But this is also because the classification accuracy of Model B reached 99.91%, that is, only two of the total 1,976 samples were incorrectly classified as ‘false negative’. There was no false positive. In all, these models had higher sensitivity and specificity with the optimization of the model. Also, in the two models of CNN, the sensitivity results were all higher than the specificity, and the identification ability of positive examples was better than the negative examples. This suggests a research direction that could be usefully followed up in future studies.

7. Conclusion

7.1 Summary of the Thesis

This chapter summarises the work carried out to achieve the research objectives described in Chapter 3. This chapter also presents recommendations for future work in this field.

The aim of this thesis was to classify shoe print images using convolutional neural networks. From the results of the experiments carried out herein, it can be seen that there was excellent identification efficiency and precision.

This thesis introduced a new method for the classification of shoe print images. Specifically, a convolution neural network was used to classify images that contain a shoe print. This method (CNN) belongs to a model of image classification in machine learning. The contributions and highlights of this study are outlined below.

First, chapter 2 introduced the fields of shoe print identification and face identification, and surveyed TensorFlow. The identification of shoe prints has a very long tradition, mainly used in crime scene investigation and research. The shoe print picture is accepted as significant evidence. The public security sector has long been committed to creating a shoe print database to improve the efficiency of classifying such evidence. Previously, the classification and identification of shoe printing depended mainly on artificial methods (e.g. human eyes). The use of machine learning to classify shoe print images is also a new area of research that has emerged in recent years and has not yet been used in neural networks. We investigated the use of CNN in the research on facial identification in the field of image identification because, to date, neural networks have not been central to research into shoe print identification. We also studied the development and status of TensorFlow.

Second, chapter three identified the problems that we needed to address in this study. The plan was to use artificial neural networks, support vector machines, and CNN on

the shoe print image classification.

Third, chapter 4 summarised the different machine learning models, learning algorithms, and algorithmic theories. Then, the experimental method of the CNN model used in this thesis was analysed critically.

Forth, briefly introduced the theoretical knowledge and operational method of the artificial neural network, and employed ANN to classify shoe print image datasets. The obtained results were used as the precision benchmark for CNN. This contribution corresponds to the first study question.

Fifth, the support vector machine's theory and function were briefly documented. A variety of support vector machines were introduced, and images were classified using multiple SVM. The classification results became another benchmark for CNN. This contribution corresponds to the second study question.

Sixth, in Chapter 5, a CNN was proposed to deal with the binary classification of shoe print images. The experimental results and analysis were discussed in Chapter 6. This contribution answered the study of question three. In addition, this chapter explored the multi-classification of shoe print images, the extraction of shoe prints from complex images, and the identification of the position of shoes printed in complex images. These are promising methods that merit further investigation in future research.

7.2 Concluding Remarks

This thesis explored the possibility of realising the application of the convolutional neural network to a new field. Shoe print images can be compared with the human facial images and fingerprint images. However, while there are many similarities, there are also many differences. In contrast to other images, the shoe print pattern does not have a regular, definite shape. Points, lines, circles, and polygons do not have a specific

position or distribution. This is detrimental to machine learning.

It was determined through the experimentation in this thesis that CNN, which are extracted from facial recognition technology, still have a high performance when classifying shoe print images. The local receptive field of convolutional neural networks provides an excellent learning ability. The work of different convolution kernels is different.. They are activated when different images are processed. This can be very clearly expressed in the heat map. The different neural network models bring significant changes to the accuracy of the classification. A complex model is not necessarily better than a small classification of simple models. CNN is a powerful classification model. In this experiment, the classification success rate reached 99.91%. This is a good result, but it can also prove that the potential of CNN is not entirely realised. We can use it to classify more complex datasets, for example, the multi-classification of shoe print images. The result may suggest a limitation to the research insofar as the objective of the experiment was too simple. If we were to start over again, it would be more effective if more complex classification requirements were established. Moreover, in terms of the data used in this experiment, it should be noted that there was only one shoe print in the image.

In sum, it can be argued that CNN is suitable for the classification of shoe prints. In the previous survey in chapter 2, we also found that in machine learning, the use of other machine learning methods for image classification also has outstanding results. If the experiment were to be repeated, we would try to use other machine learning methods as benchmarks for CNN. We also recognise that merely changing the number of nodes in one layer of CNN is not enough to judge CNN's merits and demerits. Using different CNN should focus more on the use of entirely different structures. For example, residual neural networks (RNN) or the Inception V3 model could be used in order to reflect the different CNN in dealing with complex problems of different performance.

7.3 Recommendations for Future Research

We have also briefly mentioned the direction of development in this area. The main points are as follows.

- This experiment has only two outputs - a shoe print or not a shoe print. Subsequent studies could focus on classifying more complex data, for example, different brands or different shoe types.
- The dataset used in this image is very concise, the shoe print is very prominent, and there is only one shoe print in the image. Future studies could try to use more complex images involving, say, a shoe print that is extracted from a complex site or an image that contains multiple shoe prints. An example here could be some shoe prints in a piece of sand.
- The CNN used in this experiment is relatively simple. Subsequent experiments could attempt to use more layers or more efficient convolutional neural network models.
- The experiment did not verify the rotation, distortion, and deformation of the image. We can learn from other studies (such as facial recognition) that CNN has a good performance in dealing with such problems. However, in the field of shoe printing identification, we still need to do experiments to prove the performance of neural networks.

After the several above goals have been achieved, we can establish the relationship between the shoe print and the person's posture, behaviour and personal information. This information could be obtained through setting up a shoe printing analysis system. In this way, shoe printing identification may enable the shoe print owner to be established. This would represent a huge step forward.

References

Abadi, M., & Agarwal, A. (2016). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems., (p. arXiv:1603.04467).

Aboulnasr, T., & Othman, H. (2003). A separable low complexity 2D HMM with application to face recognition. *IEEE Trans. Pattern Anal. Machine Intell., Vol 25, No.10, pp.1229-1238.*

Aboulnasr, T., & Othman, H. (2003). A separable low complexity 2D HMM with application to face recognition. *IEEE Trans. Pattern Anal. Machine Intell., Vol 25, No.10, pp.1229-1238.*

Adams, K. (2009, July). A survey of palmprint recognition. *ELSEVIER*, pp. 1408-1418.

Andreas, M., Jorge, A., Patrick, J., & et.al. (2018). Exploiting Typical Values to Accelerate Deep Learning. *Computer*, pp. 18 - 30.

Angelova, A., & Krizhevsky, A. (2015). Pedestrian detection with a large-field-of-view deep network. *Robotics and Automation (ICRA)* (pp. 704–711). IEEE International Conference on.

Arumugam, S., & Rathinavel, S. (2011). Full Shoe Print Recognition based on Pass Band DCT and Partial Shoe Print Identification using Overlapped Block Method for Degraded Images. *International Journal of Computer Applications (0975 – 8887)*.

Beaufays, F. (2015). *The neural networks behind Google Voice transcription.*

Belhumeur, P., Hespanha, J., & Kriegman, D. (1997). Eigenfaces versus fisherfaces: Recognition using class specific Linear projection. *IEEE Trans. Pattern Anal. Mach. Intell., Vol.19, No.7, pp. 711-720*.

Belkin, M. (2003). Laplacian eigenmaps for dimensionality reduction and data repersentation. *Neural Computation*, pp. 1373-1396.

Bengio. (2013). Representation learning: A review and new perspectives. *IEEE Transaction on pattern analysis and machine intelligence*, pp. 1798-1828.

Bodziak, W. (2000). Footwear Impression Evidence Detection Recovery and

Examination.

Bouvrie, J. (2006). Notes on Convolutional Neural Networks. *(Unpublished)*.

Chang, B. (2016, June). SVM-PSO based rotation-invariant image texture classification in SVD and DWT domains. *Engineering Applications of Artificial Intelligence*, pp. 96-107.

Chen, T., & Li, M. (2015). MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems. *Proceedings of LearningSys.*

Chen, Y., Luo, T., Liu, S., Zhang, S., & He, L. (2014). DaDianNao: A Machine-Learning Supercompute. *Microarchitecture (MICRO) 014 47th Annual IEEE/ACM International Symposium on*, (pp. 609-622).

Ci, S. (1235). *XI Yuan Lu.* China.

Clark, J. (2015). *Google turning its lucrative web search over to AI machines.*

Cootes, T. (2001). Active apprearance model. *IEEE Transsction on Pattern Analysis and Machine Intelligence*, pp. 681-685.

Creighton, D. S. (2017, Sep 22). The application of a pre-positioned upper cervical traction mobilization to patients with painful active cervical rotation impairment: A case series. *Journal of Back and Musculoskeletal Rehabilitation*, pp. 1053-1059.

Crookes, D. (2007, Jan). Local image Features for Shoeprint Image Retrieval . *DBLP*.

Dai, X. (2010). Content-Based Image Retrieval Method and its Application to Shoeprint Identification. *978-1-4244-3709-2/10/ IEEE*.

Dean, J., & Corrado, G. S. (2012). Large scale distributed deep networks. *NIPS.* Google Research PDF.

Dean, J., & Corrado, G. S. (2012). Large scale distributed deep networks. *Proceedings of NIPS*, (pp. 1232–1240).

Dean, J., Corrado, G. S., & Monga, R. (2012). Large scale distributed deep networks. *NIPS*, (pp. 70-72).

Ding, C. (2015). Robust Face Recognition via Multimodal Deep Face Representation. *IEEE TRANSACTIONS ON MULTIMEDIA*, (pp. 2049-2058).

Dong, Y. (2017, Mar). Footprint Image Matching Algorithm Based on SIFT and

RANSAC. *JOURNAL OF HENAN UNIVERSITY OF ENGINEERING*.

Everingham, M., Gool, L. V., Williams, C. K., Winn, J., & Zisserman, A. (2012, June). The PASCAL Visual Object Classes Challenge. *International Journal of Computer Vision*, pp. 303–338.

Fan, H., & Cao, Z. (2014). Learing deep face representation. *arXiv preprint*, p. 2802.

Frome, A., & Corrado, G. S. (2013). A deep visual-semantic embedding model. *Advances in Neural Information Processing Systems*, pp. 2121–2129.

Gómez, C. M. (2017, Jan). Absolute Power Spectral Density Changes in the Magnetoencephalographic Activity During the Transition from Childhood to Adulthood. *Brain Topography*, pp. 87-97.

Gao, S. (2003, Feb). Research on shoe print file management and computer aided retrieval system. *Journal of Chinese People's Public Security Uni*.

Gharsa, A. (2008, October). A novel technique for automatic shoeprint image retrieval. *Forensic Science International*, pp. Pages 10-14.

Glorot, X., Bordes, A., & Y., B. (2011). Deep sparse rectifier neural networks. *Proc. 14th International Conference on Artificial Intelligence and Statistics*, (pp. 315–323).

Goodfellow. (2014). Generative adversarial nets. *Proceedings of NIPS*, (pp. 2672–2680).

Goodfellow, I., & Bulatov, Y. (2014). Multi-digit number recognition from Street View imagery using deep convolutional neural networks. *International Conference on Learning Representations*, (p. arxiv.org/pdf/1312.6082).

Gross, R. (2005). *Face Databases, Handbook of Face Recognition.* Retrieved from FACE RECOGNITION HOMEPAGE: http://www.face-rec.org/databases/

Gu, W. (2012). Facial expression recognition using redial encoding of local Gabor features and classifier Synthesis. *Pattern Recognition*, pp. 80-91.

Guan, Y., Li, C.-h., & Zhong, Z.-m. (2008, Aug). Research and realization of recognition for shoe soles based on outline feature. *Application Research of Computers*.

Guo, S. (2018). SVD-based burning state recognition in rotary kiln using machine

learning. *Industrial Electronics and Applications (ICIEA), 2017 12th IEEE Conference on* (pp. 18-20). IEEE.

Hamiane, G. A. (2008, October). A novel technique for automatic shoeprint image retrieval. *Forensic Science International, Volume 181*, pp. 10-14.

Hasim, S., & Andrew, S. (2015). *Google Voice Search: faster and more accurate.*

Hinton, G., & Krizhevsky, A. (2012). ImageNet Classification with deep convolutional neural network. *Proc of NIPS.*

Hinton, G., & Salakhutdinov, R. R. (2006). Reducing the diemensionality of data with neural network. *Science*, pp. 504-507.

Hinton, G., Osindero, S., & Teh, Y. (2006, July). A fast learning algorithm for deep belief nets. *Neural Cumputation*, pp. 1527-1555.

Horswell, J., & Cordiner, S. (2002). Forensic Comparison of Soils by Bacterial Community DNA Profiling. *Journal of Forensic Sciences, Vol. 47*(2), pp. 350-353.

Hubel, D. H., & Wiesel, T. N. (1962). Receptive fields, binocular interaction adn functional architecture in the cats visual cortex. *The Journal of Physiology*, pp. 106-154.

J. Lu, K. P. (2003). Face recognition using LDA based algorithms. *IEEE Transactions on Neural Networks 14(1), pp.195-200.*

Javier, G.-D., & Ignacio, L.-M. (2015). Frameby-frame language identification in short utterances using deep neural networks. *Neural Networks*, pp. 64:49–58.

Jia, Y., & Shelhamer, E. (2014). Caffe: Convolutional architecture for fast feature embedding. *Proceedings of ACM Multimedia*, (pp. 675–678 arxiv.org/abs/1408.5093.).

Jiang, B. (2014). A Local Discriminative component analysis algorithm for facial expression recognition. *Acts Electronica Sinics*, pp. 155-159.

Jordan, M. I. (1986). Serial order: A parallel distributed processing approach. *ICS report 8608, Institute for Cognitive Science*.

Karpathy, A., & Toderici, G. (2014). Large-scale video classification with convolutional neural networks. *In Computer Vision and Pattern Recognition*

(CVPR) (pp. 1725–1732). IEEE.

Krizhevsky, A. (2014). One weird trick for parallelizing convolutional neural networks. *arXiv preprint*, p. arxiv.org/abs/1404.5997.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Proceedings of NIPS*, (pp. 1106–1114 papers.nips.cc/paper/4824- imagenet-classification-with-deep-convolutionalneural-networks.pdf).

Lawrence, S. (1997, JANUARY). Face Recognition: A Convolutional Neural-Network Approach. *IEEE TRANSACTIONS ON NEURAL NETWORKS, 8*, p. 98.

Le, Q., & Ranzato, M. (2012). Building high-level features using large scale unsupervised learning. *ICML 2012.*

LeCun, Y., Boser, B., Denker, J. S., & Henderson, D. (1989, 4). Backpropagation applied to handwritten zip code recognition. *Neural Computation*.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998, November). Gradient-Based Learning Applied to Document Recognition. *PROCEEDINGS OF THE IEEE*, pp. 0018–9219/98.

Levine, Z. H. (2001). Face recognition using he discrete cosine transform. *Int. J. Comput. Vis., Vol.43, No.3, pp. 167-188*.

Li, Z. (2011). Research of shoeprint image stream retrival algorithm with scale-invariance feature transform. *Multimedia Technology (ICMT), 2011 International Conference on*, (pp. 26-28).

Liu, i., & Liu, W. (2018, Jan). Image-set based face recognition using K-SVD dictionary learning. *International Journal of Machine Learning and Cybernetics*, pp. 1-14.

Lu, G. (2013). Neonatalpainexpresionrec- ognition based on uncorelated loclaity sensitivediserimi- nantanlaysis. *Journal of Naming University of Posts and Telecommunications:Natural Science Edition*, pp. 1-7.

Lu, G. (2014). Feature extraction based on two—dimensionla locality preserving discriminnat analysis. *Journal of Naming University of Posts and Telecommunications:Natural Science Edition*, pp. 1-8.

Lu, G. (2016). Convolutional neural network for facial expression recognition. *Journal*

of Naming University of Posts and Telecommunications:Natural Science Edition, pp. 17-22.

Lu, J. (2017). Simultaneous Feature and Dictionary Learning for Image Set Based Face Recognition. *IEEE TRANSACTIONS ON IMAGE PROCESSING*, (pp. 4042-4054).

Lu, J., Plataniotis, K., & Venetsanopoulos, A. (2003). Face recognition using LDA based algorithms. *IEEE Transactions on Neural Networks 14(1), pp.195-200.*

Mark, E., Luc, V., Christopher, K. I., John, W., & Andrew, Z. (2012, June). The PASCAL Visual Object Classes Challenge. *International Journal of Computer Vision*, pp. 303–338.

Masi, I. (2016). Do We Really Need to Collect Millions of Faces for Effective Face Recognition? *European Conference on Computer Vision ECCV 2016: Computer Vision – ECCV 2016*, (pp. 579-596).

Mikolov, T., & Chen, K. (2013). Efficient estimation of word representations in vector space. *International Conference on Learning Representations: Workshops Track*, (p. arxiv.org/abs/1301.3781).

MIT. (2018). *Face Databases* . Retrieved from MIT.edu: http://web.mit.edu/emeyers/www/face_databases.html#wild

Mnih, V., & Kavukcuoglu, K. (2015). Human-level control through deep reinforcement learning. *Nature*, pp. 518(7540):529–533 dx.doi.org/10.1038/nature14236.

Nibouche, O. (2009, Sept.). Rotation Invariant Matching of Partial Shoeprints. *International Machine Vision and Image Processing Conference.*

Niu, L., & Chen, X. (2016, 11). Model construction and preformance analysis for dee consecutive conbolutional neural network. *Journal of Shenyang University of Technology*, pp. 662-666.

Parkhi, & Omkar, M. (2015). Deep Face Recognition. *Proceedings of the British Machine Vision*, 6.

Peng, W. (2005, Feb). Method for extracting dust shoeprint on the surface of flannel cloth. *Criminal technology*.

Pentland, M. T. (1991.). Eigenfaces for Recognition. *Journal of Cognitive Neuroscience,*

Vol.3, No.1, pp. 71-86, .

Philip M. Remes, M. W. (2016, Sep 27). Methods of operating a fourier transform mass analyzer . *Nature Journal*.

Rathinavel, S. (2011, July). Full Shoe Print Recognition based on Pass Band DCT and Partial Shoe Print Identification using Overlapped Block Method for Degraded Images. *International Journal of Computer Applications*, pp. 16-21.

Razavian, A. S. (2014). CNN Features off-the-shelf: an Astounding Baseline for Recognition. *Computer Vishion Fundation*.

Remes, P. M., & Senko, M. W. (2016, Sep 27). Methods of operating a fourier transform mass analyzer. *Nature Journal*.

Rosenberg, C. (2013). *Improving Photo Search: A step across the semantic gap.*

Roweis, S. T. (2000). Nonlinear Dimensionality reduction by locally linear embedding. *Science*, pp. 2323-2326.

Rumelhart, D. E., & Hinton, G. E. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, pp. 213–220.

Sawyer, N. (1995). “Shoe-fit” - a computerised shoe print database. *European Convention on Security and Detection*, (pp. 86 – 89).

Schroff, F., & Kalenichenko, D. (2015). a unified embedding for face recognition and clustering. *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, (pp. 815–823).

Shan, C. (2009). Facial expression recognition based on local binary patterns: A comprechensive study. *Image and vision Computing*, pp. 803-816.

Sirvoich, M. K. (1990). Application of the karhunenLoeve Procedure for the characterization of human faces, —. *IEEE Trans. Pattern Anal. Mach. Intell., Vol.12, No.1, pp. 103-108*.

Sun, F. (2015). Deep Convolution Neural Network Recognition Algorithm Based on Improved Fisher Criterion. *JOURNAL OF BEIJING UNIVERSITY OF TECHNOLOGY*, pp. 835-841.

Sun, Q.-S. (2005, Dece). A new method of feature fusion and its application in image recognition. *Pattern Recognition*, pp. 2437-2448.

Sungjoon, C. (2013). Human behaivor prediction for smart homes using deep learning. *Proc of the 22nd IEEE international Symposium on Robot and human Interactive Communication*, (pp. 173-179).

Szegedy, C., & Liu, W. (2015). Going deeper with convolutions. *CVPR'2015*, p. arxiv.org/abs/1409.4842.

Taigman, Y., & Yang, M. (2014). DeepFace: Closing the gap to human-level perfoemance in Face vierification. *Proc of IEEE conference on Computer vision and pattern recognition*, (pp. 1701-1708).

Tenenbaum, J. B. (2000). A global geometric framwork for nonlinear dimensionality reduction. *Science*, pp. 2319-2323.

Vinyals, O., & Kaiser, K. T. (2014). Grammar as a foreign language. *Technical report*, p. arXiv:1412.7449.

Wang, X. (2013). Feature Fusion of hog and wld for ficial expression recognition. *IEEE/SICE International Symposium on System Inergration*, (pp. 227-232).

Warden, P., & Lane, N. D. (2018, 05 24). The Deep (Learning) Transformation of Mobile and Embedded Computing. *Computer*, pp. 12 - 16.

Woźniak, M. (2017, Jun 17). Voice recognition through the use of Gabor transform and heuristic algorithm. *International Journal of Electronics and Telecommunications*.

Xiao, R. (2007). Shoeprint recognition and its application.

Xu, X., & Liu, H. (2018). Facial expresion recognition based on convolutional neural network. *Foreign Electronic Measurement Technology*.

Yang, Y. (2010, Mar). The shooting technique of the water trace shoe print on the white ceramic tile. *Journal of Chinese Criminal Police Institute*.

Yin, B.-c., Wang, W.-t., & Wang, L.-c. (2015). Review of deep learning. *Journal of Beijing University of Technology*, pp. 48–59.

Zhang, M. (2017). The framework of TensorFlow form Google. *Computer and Application of China*, pp. 58-60.

Zhou, E., & Cao, Z. (2015). Naive-deep face recognition: touching the limit of LFW benchmark or not? *arXiv preprint: arXiv:1501.04690*.

8. Appendix

8.1 train.py

```
1. #!/usr/bin/env python
2. ## input files
3. import os
4. import numpy as np
5. import tensorflow as tf
6. import input_data
7. import model
8.
9. # variables declare
10. N_CLASSES = 2  # 2 classes
11. IMG_W = 64  # resize images
12. IMG_H = 64
13. BATCH_SIZE = 10
14. CAPACITY = 200
15. MAX_STEP = 1000
16. learning_rate = 0.00001  # usually less than 0.0001
17.
18. # get batch
19. train_dir = 'C:/Users/ChengranLi/Desktop/world-
    master_test/input_data'  # load dir
20. logs_train_dir = 'C:/Users/ChengranLi/Desktop/world-
    master_test/save'  # logs dir
21.
22. # train, train_label = input_data.get_files(train_dir)
23. train, train_label, val, val_label = input_data.get_files(train_dir, 0.3)
24. # training data & label
25. train_batch, train_label_batch = input_data.get_batch(train, train_label, IM
    G_W, IMG_H, BATCH_SIZE, CAPACITY)
26. # testing data & label
27. val_batch, val_label_batch = input_data.get_batch(val, val_label, IMG_W, IMG
    _H, BATCH_SIZE, CAPACITY)
28.
29. # training operation define
30. train_logits = model.inference(train_batch, BATCH_SIZE, N_CLASSES)
31. train_loss = model.losses(train_logits, train_label_batch)
32. train_op = model.trainning(train_loss, learning_rate)
33. train_acc = model.evaluation(train_logits, train_label_batch)
34.
35. # testing operation define
36. test_logits = model.inference(val_batch, BATCH_SIZE, N_CLASSES)
```

```
test_loss = model.losses(test_logits, val_label_batch)
test_acc = model.evaluation(test_logits, val_label_batch)

# summary logs
summary_op = tf.summary.merge_all()

# start a session
sess = tf.Session()
# define a writer to write log files
train_writer = tf.summary.FileWriter(logs_train_dir, sess.graph)

# define a saver to save model after training
saver = tf.train.Saver()
# Initialization
sess.run(tf.global_variables_initializer())
# queue coordinator
coord = tf.train.Coordinator()
threads = tf.train.start_queue_runners(sess=sess, coord=coord)

# start to training
try:
    # operating traning as MAX_STEP
    for step in np.arange(MAX_STEP):
        if coord.should_stop():
            break
        _, tra_loss, tra_acc = sess.run([train_op, train_loss, train_acc])

        # every 50 step print current data loss and acc. save log and write to writer
        if step % 10 == 0:
            print('Step %d, train loss = %.2f, train accuracy = %.2f%%' % (step, tra_loss, tra_acc * 100.0))
            summary_str = sess.run(summary_op)
            train_writer.add_summary(summary_str, step)
        # save model every after 100 steps
        if (step + 1) == MAX_STEP:
            checkpoint_path = os.path.join(logs_train_dir, 'model.ckpt')
            saver.save(sess, checkpoint_path, global_step=step)

except tf.errors.OutOfRangeError:
    print('Done training -- epoch limit reached')

finally:
    coord.request_stop()
```

8.2 model.py

```
1.  # ==============================================================================

2.  import tensorflow as tf
3.
4.
5.  # ==============================================================================

6.  # CNN definiton
7.  # input parameters: images, image batch、4D tensor、tf.float32、
    [batch_size, width, height, channels]
8.  # return parameters: logits, float、 [batch_size, n_classes]
9.  def inference(images, batch_size, n_classes):
10.     # CNN with (conv+pooling) x2, full_connection x2, dropout x1, softmax x1

11.     # conv1
12.     # 64 3x3 fillters（3channel）,
    padding='SAME', conv_image size is same as ori_image size after padding, ac
    t_Func_relu()
13.     with tf.variable_scope('conv1') as scope:
14.         weights = tf.Variable(tf.truncated_normal(shape=[3, 3, 3, 64], stdde
    v=1.0, dtype=tf.float32),
15.                               name='weights', dtype=tf.float32)
16.
17.         biases = tf.Variable(tf.constant(value=0.1, dtype=tf.float32, shape=
    [64]),
18.                              name='biases', dtype=tf.float32)
19.
20.         conv = tf.nn.conv2d(images, weights, strides=[1, 1, 1, 1], padding='
    SAME')
21.         pre_activation = tf.nn.bias_add(conv, biases)
22.         conv1 = tf.nn.relu(pre_activation, name=scope.name)
23.
24.     # maxpool1
25.     # 3x3 max pooling, strides=2
26.     with tf.variable_scope('pooling1_lrn') as scope:
27.         pool1 = tf.nn.max_pool(conv1, ksize=[1, 3, 3, 1], strides=[1, 2, 2,
    1], padding='SAME', name='pooling1')
28.         norm1 = tf.nn.lrn(pool1, depth_radius=4, bias=1.0, alpha=0.001 / 9.0
    , beta=0.75, name='norm1')
29.
30.     # conv2
```

```
    # 16 3x3 fillters（3channel），
padding='SAME'，  conv_image size is same as ori_image size after padding, ac
t_Func_relu()
    with tf.variable_scope('conv2') as scope:
        weights = tf.Variable(tf.truncated_normal(shape=[3, 3, 64, 16], stdd
ev=0.1, dtype=tf.float32),
                              name='weights', dtype=tf.float32)

        biases = tf.Variable(tf.constant(value=0.1, dtype=tf.float32, shape=
[16]),
                             name='biases', dtype=tf.float32)

        conv = tf.nn.conv2d(norm1, weights, strides=[1, 1, 1, 1], padding='S
AME')
        pre_activation = tf.nn.bias_add(conv, biases)
        conv2 = tf.nn.relu(pre_activation, name='conv2')

    # maxpool2
    # 3x3 max pooling，strides=2
    # pool2 and norm2
    with tf.variable_scope('pooling2_lrn') as scope:
        norm2 = tf.nn.lrn(conv2, depth_radius=4, bias=1.0, alpha=0.001 / 9.0
, beta=0.75, name='norm2')
        pool2 = tf.nn.max_pool(norm2, ksize=[1, 3, 3, 1], strides=[1, 1, 1,
1], padding='SAME', name='pooling2')

    # Fc3
    # 128 neurons ，reshape maxpool2 output to liner，act func relu()
    with tf.variable_scope('local3') as scope:
        reshape = tf.reshape(pool2, shape=[batch_size, -1])
        dim = reshape.get_shape()[1].value
        weights = tf.Variable(tf.truncated_normal(shape=[dim, 128], stddev=0
.005, dtype=tf.float32),
                              name='weights', dtype=tf.float32)

        biases = tf.Variable(tf.constant(value=0.1, dtype=tf.float32, shape=
[128]),
                             name='biases', dtype=tf.float32)

        local3 = tf.nn.relu(tf.matmul(reshape, weights) + biases, name=scope
.name)

    # Fc4
    # 128 neurons act func relu()
```

```
65.     with tf.variable_scope('local4') as scope:
66.         weights = tf.Variable(tf.truncated_normal(shape=[128, 128], stddev=0
    .005, dtype=tf.float32),
67.                               name='weights', dtype=tf.float32)
68.
69.         biases = tf.Variable(tf.constant(value=0.1, dtype=tf.float32, shape=
    [128]),
70.                              name='biases', dtype=tf.float32)
71.
72.         local4 = tf.nn.relu(tf.matmul(local3, weights) + biases, name='local
    4')
73.
74.     # dropout layer
75.     with tf.variable_scope('dropout') as scope:
76.         drop_out = tf.nn.dropout(local4, 0.8)
77.
78.     # Softmax regression
79.     with tf.variable_scope('softmax_linear') as scope:
80.         weights = tf.Variable(tf.truncated_normal(shape=[128, n_classes], st
    ddev=0.005, dtype=tf.float32),
81.                               name='softmax_linear', dtype=tf.float32)
82.
83.         biases = tf.Variable(tf.constant(value=0.1, dtype=tf.float32, shape=
    [n_classes]),
84.                              name='biases', dtype=tf.float32)
85.
86.         softmax_linear = tf.add(tf.matmul(local4, weights), biases, name='so
    ftmax_linear')
87.
88.     return softmax_linear
89.
90.
91. # --------------------------------------------------------------------------
    ---
92. # loss:
93. # trans in parameters: logits,labels(0 or 1)
94. # return: loss
95. def losses(logits, labels):
96.     with tf.variable_scope('loss') as scope:
97.         cross_entropy = tf.nn.sparse_softmax_cross_entropy_with_logits(logit
    s=logits, labels=labels,
98.                                                                        name=
    'xentropy_per_example')
99.         loss = tf.reduce_mean(cross_entropy, name='loss')
```

```
100.            tf.summary.scalar(scope.name + '/loss', loss)
101.        return loss
102.
103.
104.    # ------------------------------------------------------------------------
    --
105.    # loss optimal
106.    # trans in parameters: loss,learning_rate
107.    # return: train_op, trans it into sess.run for training
108.    def trainning(loss, learning_rate):
109.        with tf.name_scope('optimizer'):
110.            optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate)
111.            global_step = tf.Variable(0, name='global_step', trainable=False)

112.            train_op = optimizer.minimize(loss, global_step=global_step)
113.        return train_op
114.
115.
116.    # ------------------------------------------------------------------------
    -
117.    # Accuracy
118.    # trans in parameters: logits labels(1 or 0)
119.    # return: accuracy
120.    def evaluation(logits, labels):
121.        with tf.variable_scope('accuracy') as scope:
122.            correct = tf.nn.in_top_k(logits, labels, 1)
123.            correct = tf.cast(correct, tf.float16)
124.            accuracy = tf.reduce_mean(correct)
125.            tf.summary.scalar(scope.name + '/accuracy', accuracy)
126.        return accuracy
```

8.3 Figures

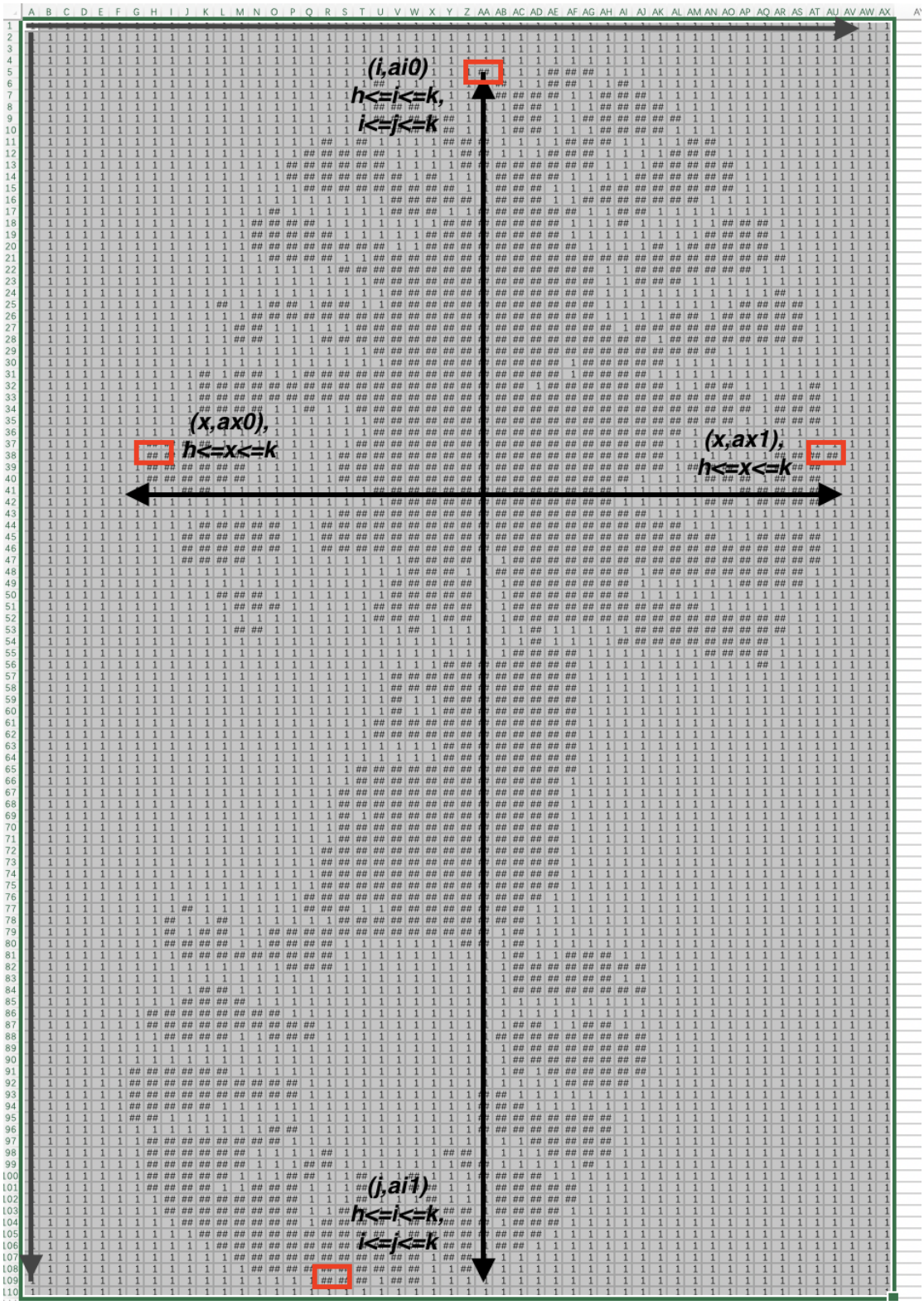


Figure 2-0

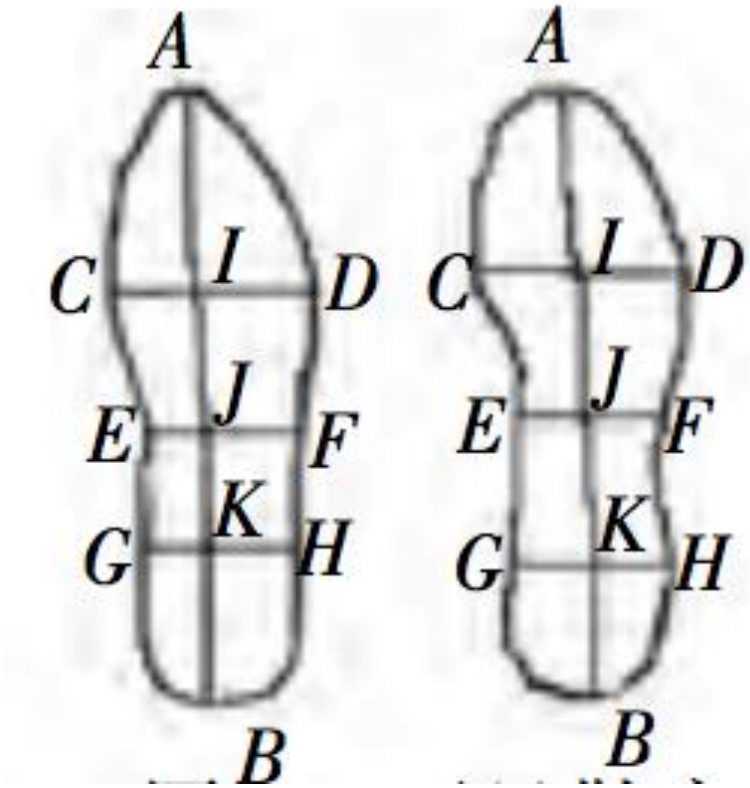


Figure 2-1

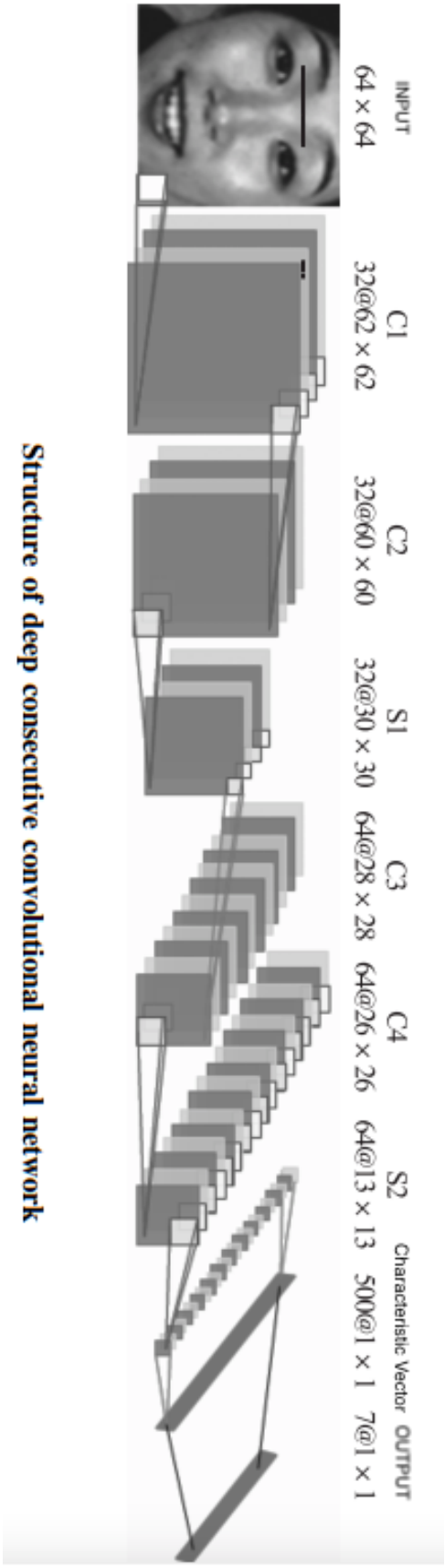

INPUT
64 × 64
C1
32@62 × 62
C2
32@60 × 60
S1
32@30 × 30
C3
64@28 × 28
C4
64@26 × 26
S2
64@13 × 13
Characteristic Vector
500@1 × 1
OUTPUT
7@1 × 1
Structure of deep consecutive convolutional neural network

Figure 2-2

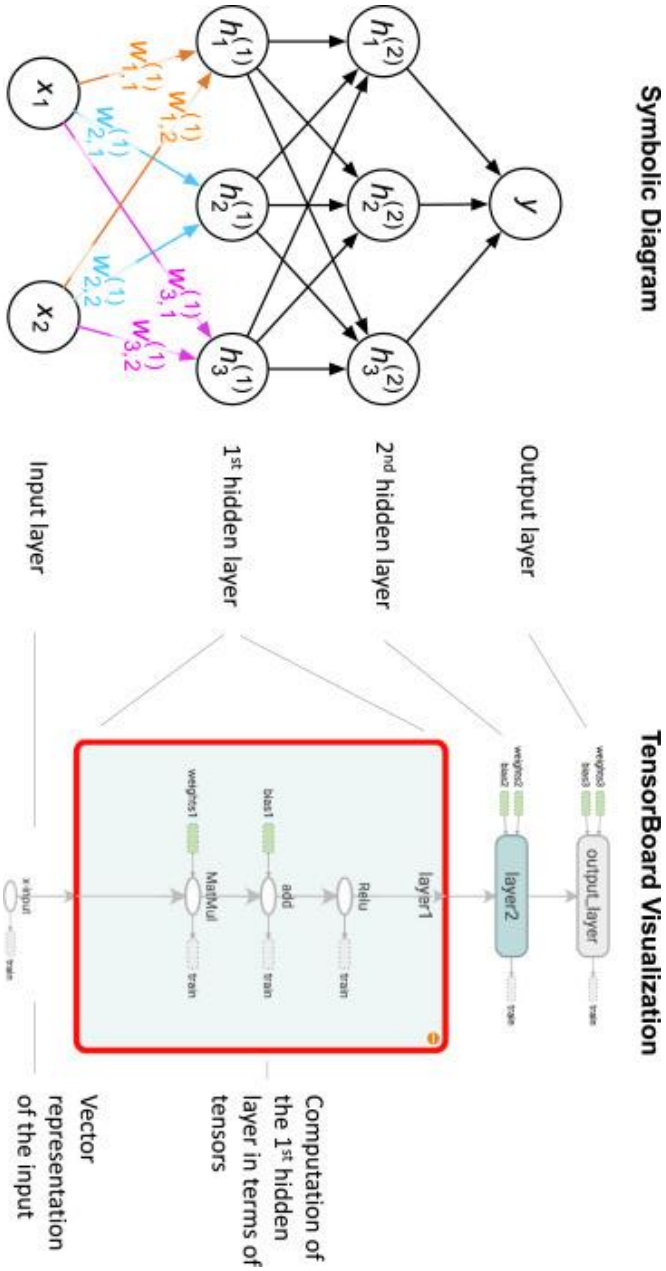


Figure 2-3 An example of a fully connected feed-forward neural network with two hidden layers

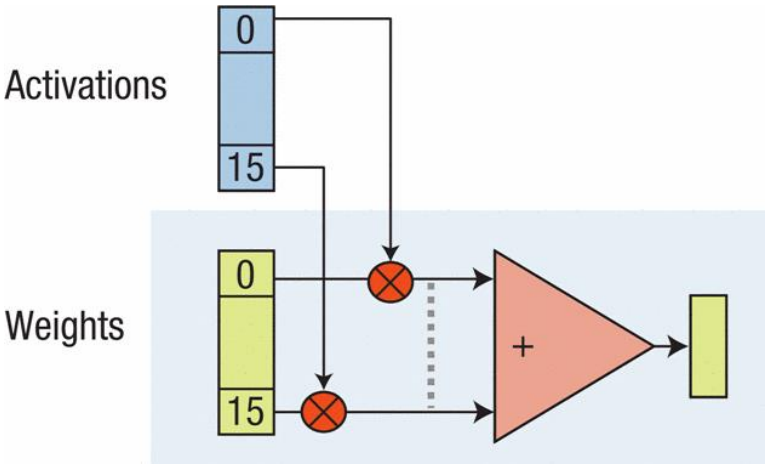


Figure 2-4 The structure of DaDianNao

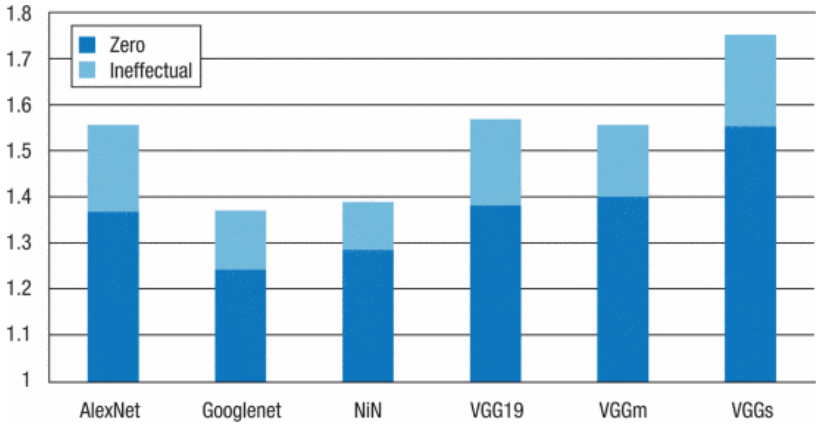


Figure 2-5 Dark blue represents the activation of zero values that can be skipped, and light blue represents the process of calculating thresholds while maintaining accuracy.

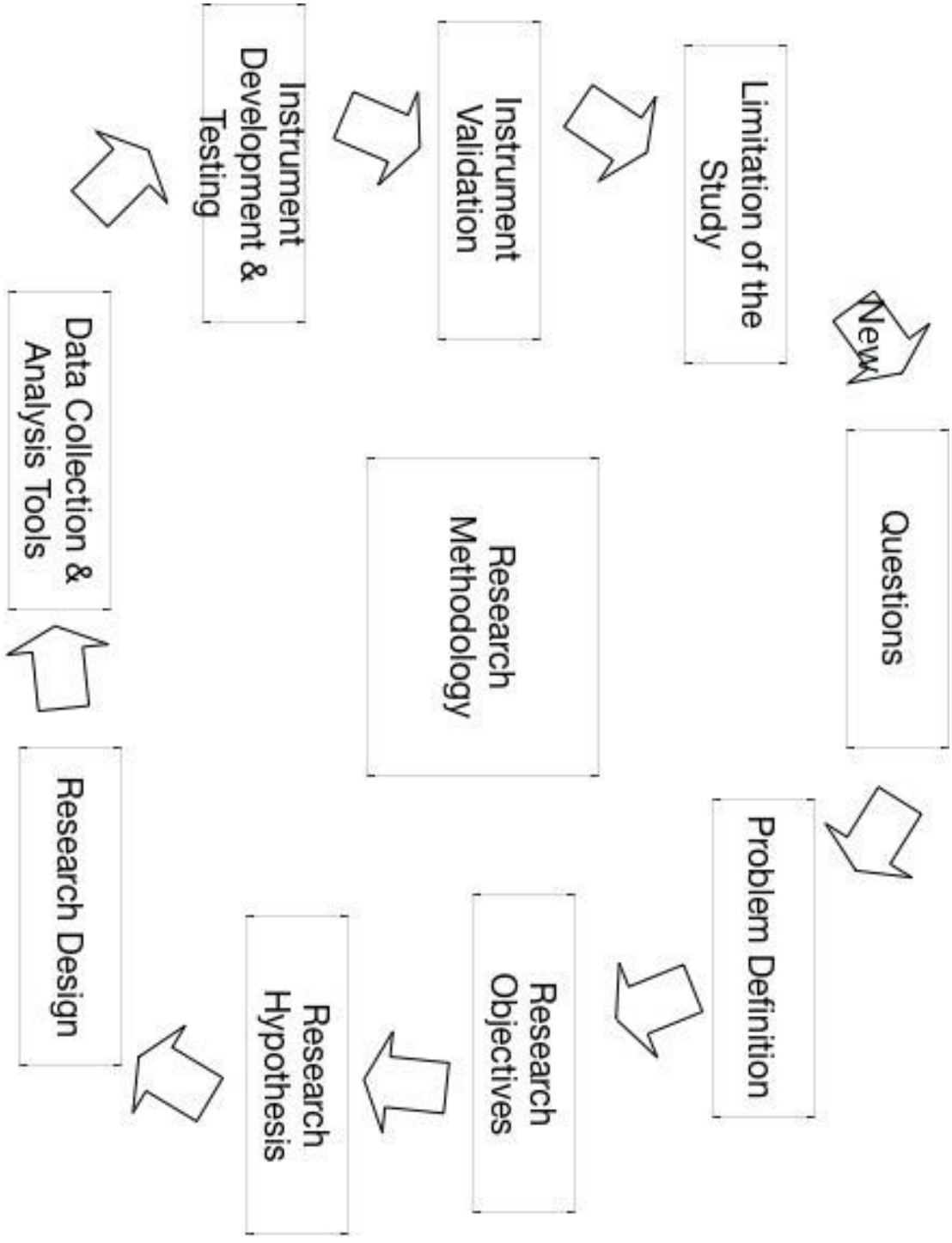


Figure 3-1 Research Methodology

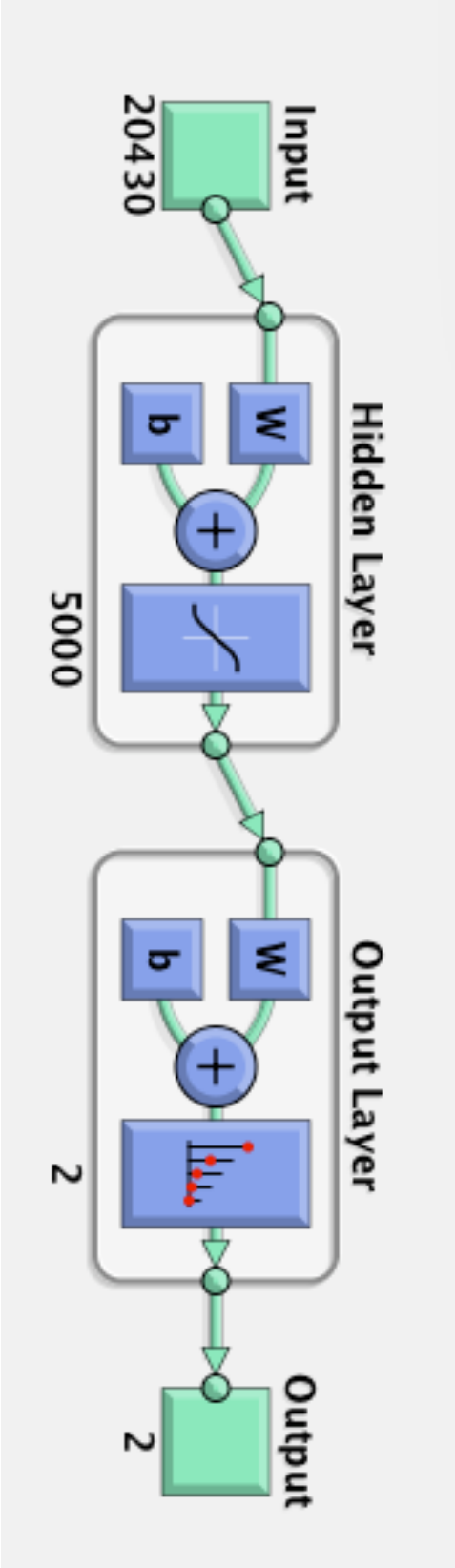


Figure 4-0

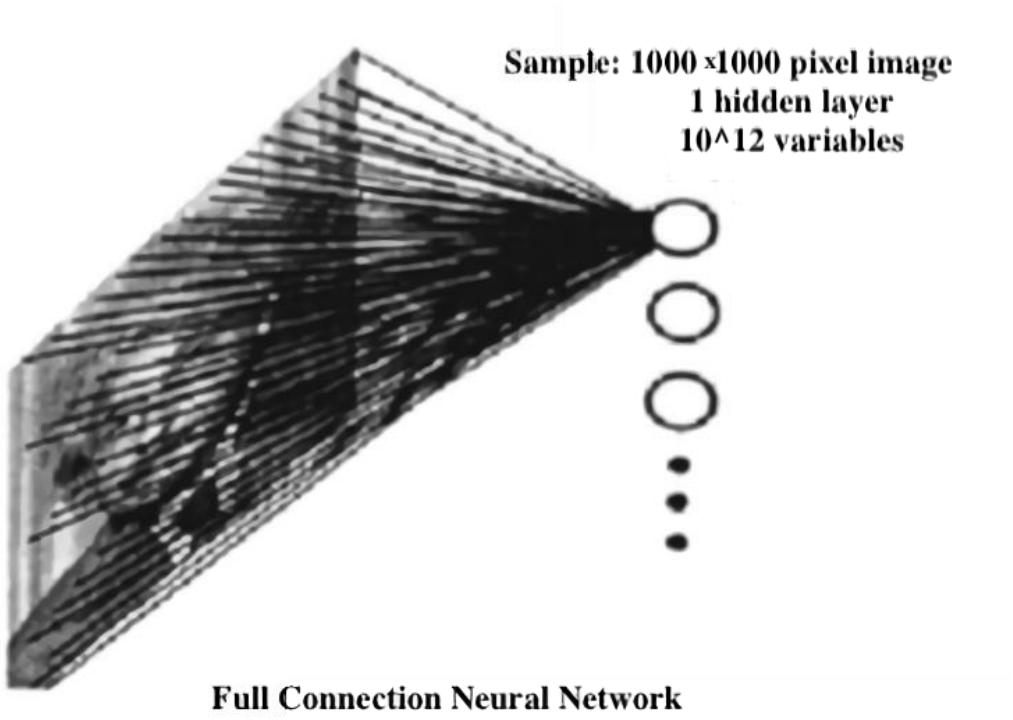

Sample: 1000 ×1000 pixel image
1 hidden layer
10^12 variables
Full Connection Neural Network

Figure 4-1a

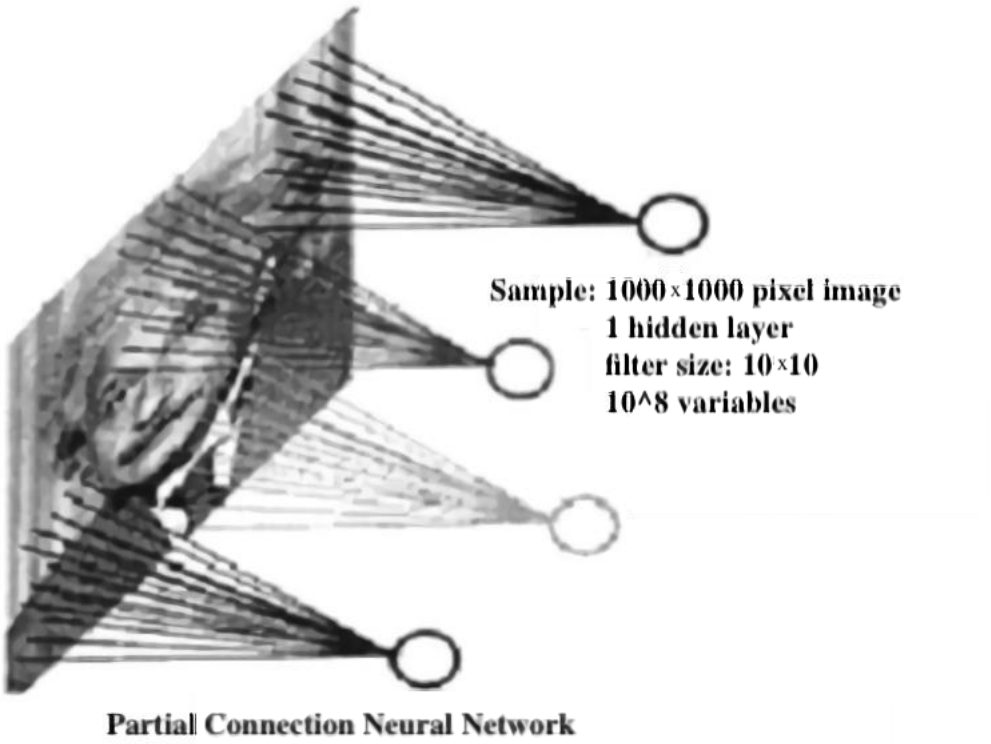

Sample: 1000×1000 pixel image
1 hidden layer
filter size: 10×10
10^8 variables
Partial Connection Neural Network

Figure 4-1b

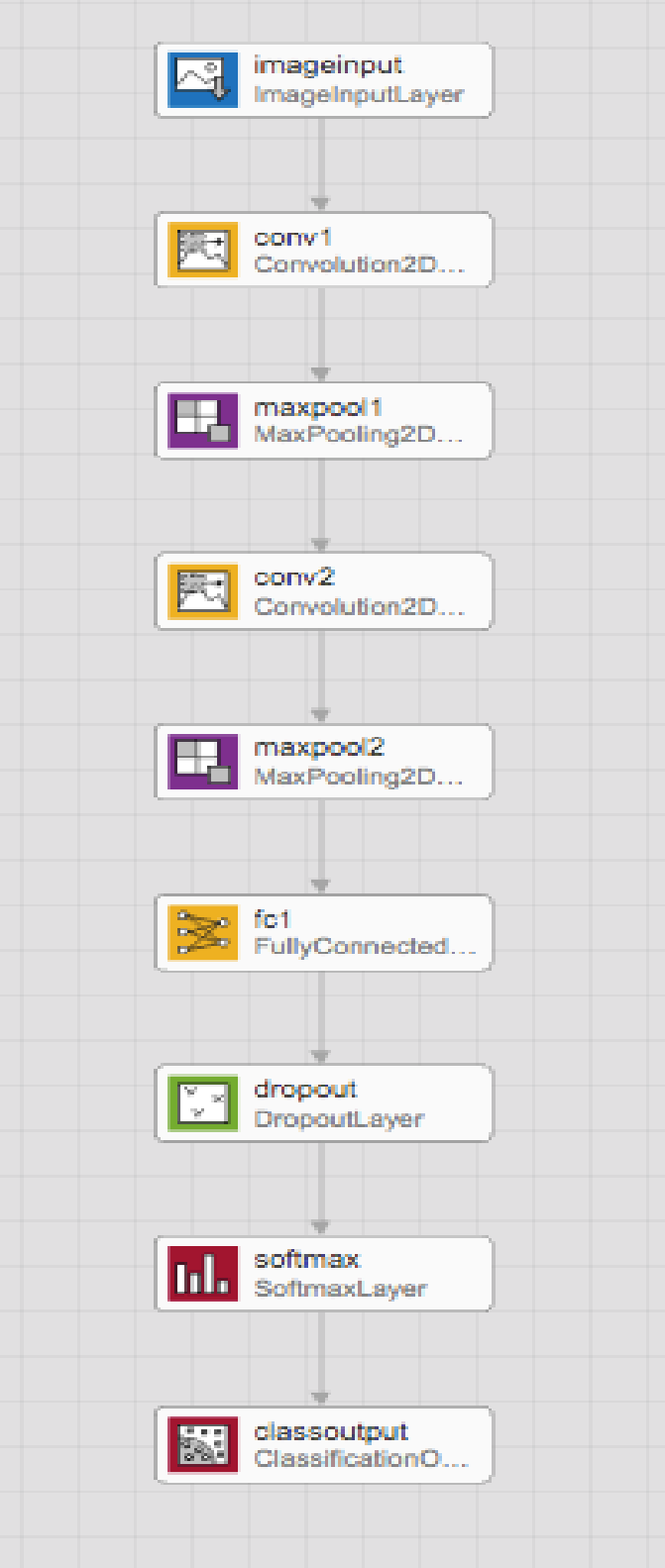


Figure 4-2 and See Table 4-1 for further details.

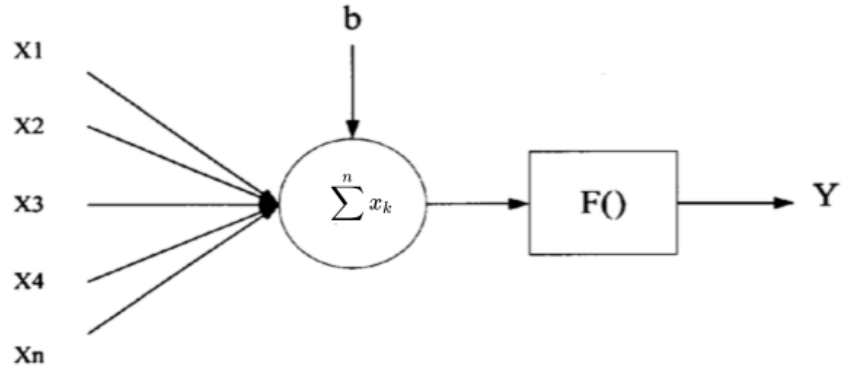


Figure 4-3

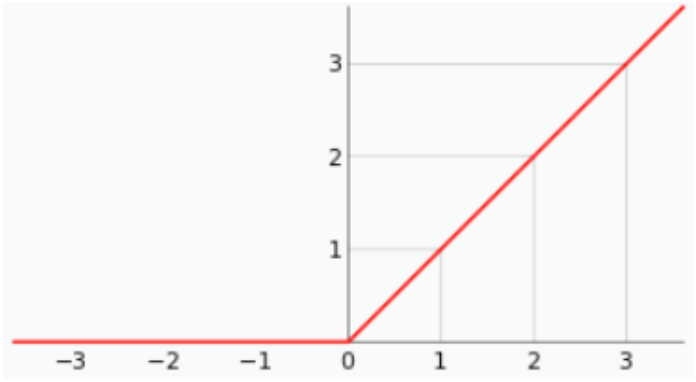


Figure 4-4 ReLu

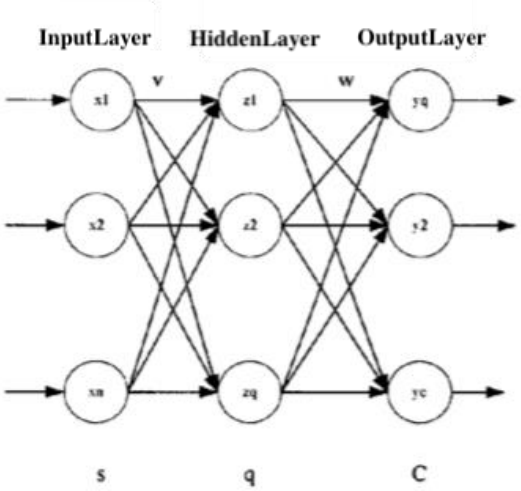


Figure 4-5

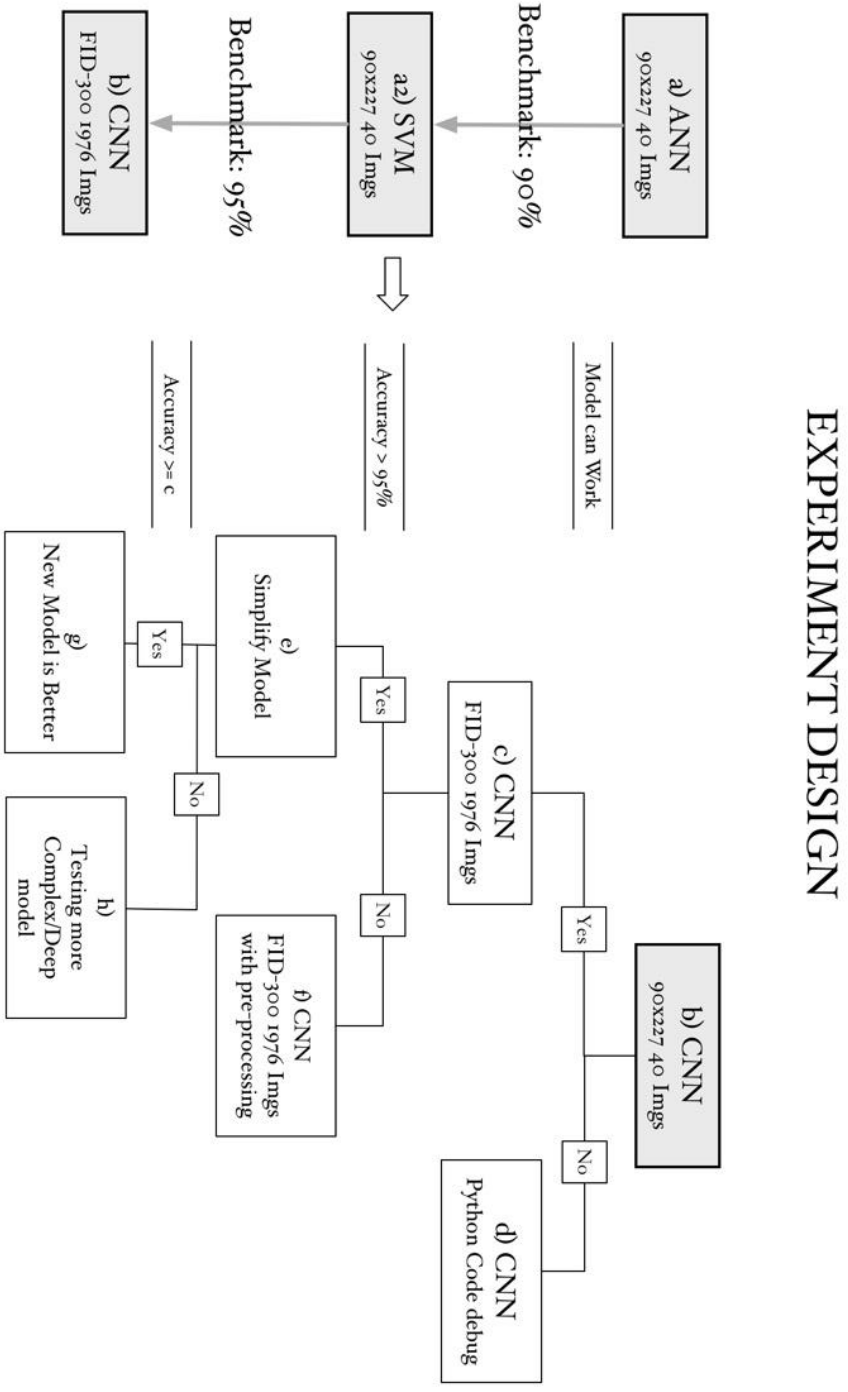

EXPERIMENT DESIGN
a) ANN
90x227 40 Imgs
Benchmark: 90%
a2) SVM
90x227 40 Imgs
Benchmark: 95%
b) CNN
FID-300 1976 Imgs
Model can Work
b) CNN
90x227 40 Imgs
Yes
No
d) CNN
Python Code debug
Accuracy > 95%
c) CNN
FID-300 1976 Imgs
Yes
No
f) CNN
FID-300 1976 Imgs
with pre-processing
Accuracy >= c
e)
Simplify Model
Yes
No
g)
New Model is Better
h)
Testing more
Complex/Deep
model

Figure 5-1

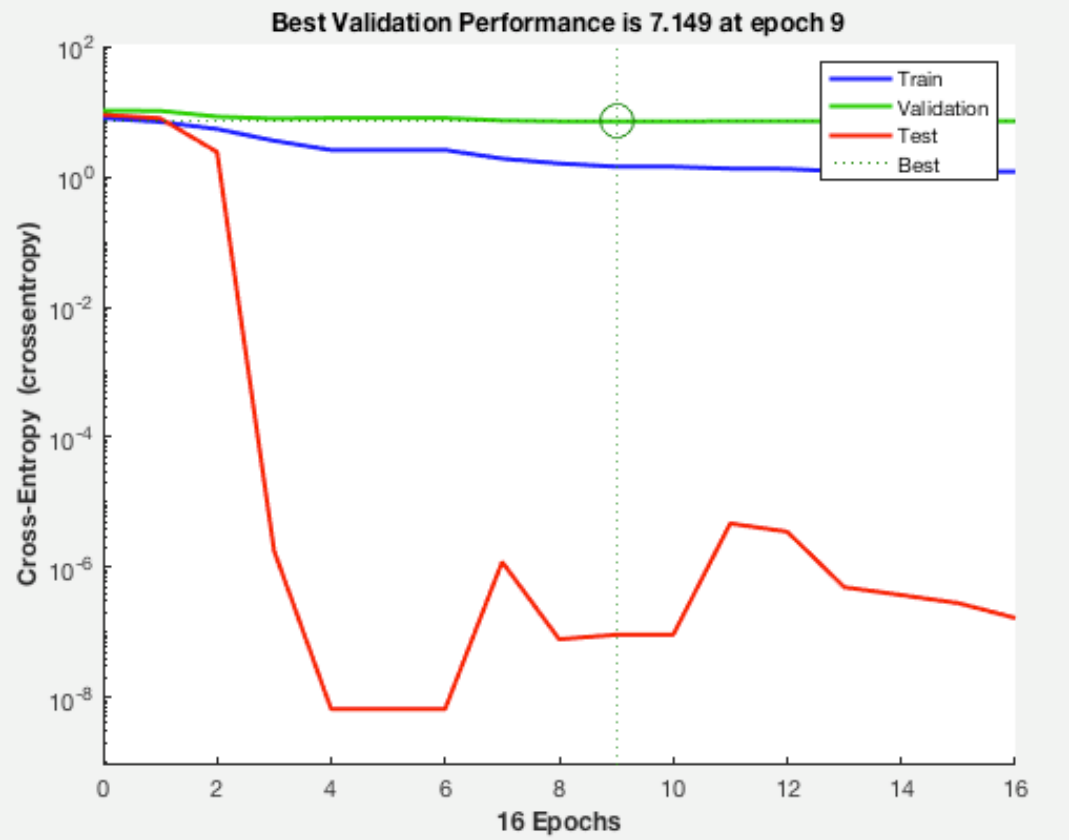

Best Validation Performance is 7.149 at epoch 9
Train
Validation
Test
Best
Cross-Entropy (crossentropy)
16 Epochs

Figure 6-1

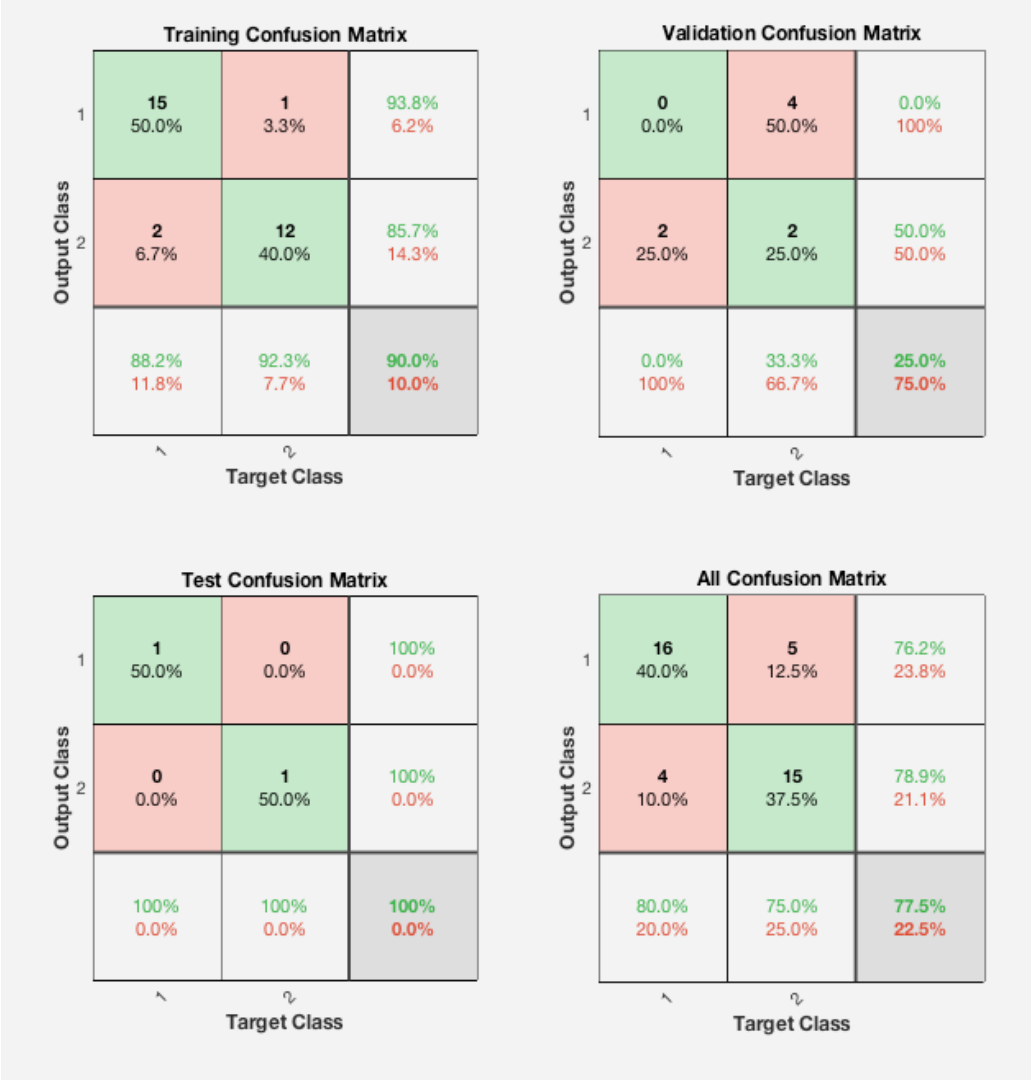

Training Confusion Matrix
15
50.0%
1
3.3%
93.8%
6.2%
2
6.7%
12
40.0%
85.7%
14.3%
88.2%
11.8%
92.3%
7.7%
90.0%
10.0%
Output Class
Target Class
Validation Confusion Matrix
0
0.0%
4
50.0%
0.0%
100%
2
25.0%
2
25.0%
50.0%
50.0%
0.0%
100%
33.3%
66.7%
25.0%
75.0%
Output Class
Target Class
Test Confusion Matrix
1
50.0%
0
0.0%
100%
0.0%
0
0.0%
1
50.0%
100%
0.0%
100%
0.0%
100%
0.0%
100%
0.0%
Output Class
Target Class
All Confusion Matrix
16
40.0%
5
12.5%
76.2%
23.8%
4
10.0%
15
37.5%
78.9%
21.1%
80.0%
20.0%
75.0%
25.0%
77.5%
22.5%
Output Class
Target Class

Figure 6-2

1.1 ☆ SVM Last change: Linear SVM	Accuracy: **95.0%** 20429/20429 features
1.2 ☆ SVM Last change: Quadratic SVM	Accuracy: **95.0%** 20429/20429 features
1.3 ☆ SVM Last change: Cubic SVM	Accuracy: **95.0%** 20429/20429 features
1.4 ☆ SVM Last change: Fine Gaussian SVM	Accuracy: **95.0%** 20429/20429 features
1.5 ☆ SVM Last change: Medium Gaussian SVM	Accuracy: **95.0%** 20429/20429 features
1.6 ☆ SVM Last change: Coarse Gaussian SVM	Accuracy: **95.0%** 20429/20429 features

Figure 6-3

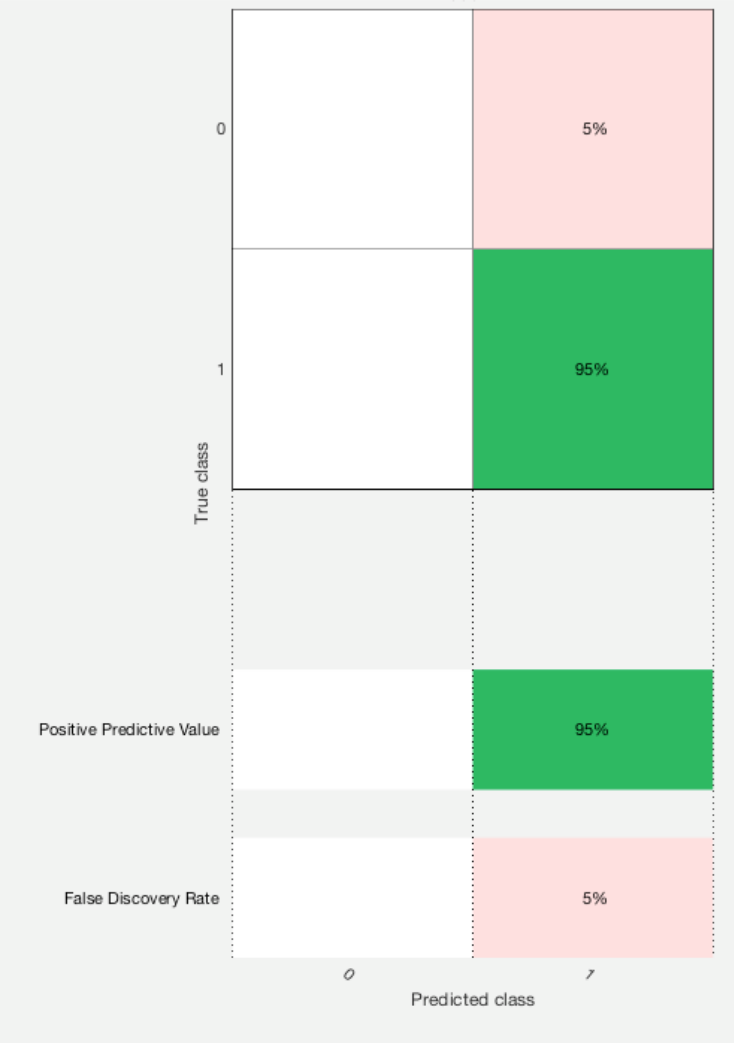


Figure 6-4

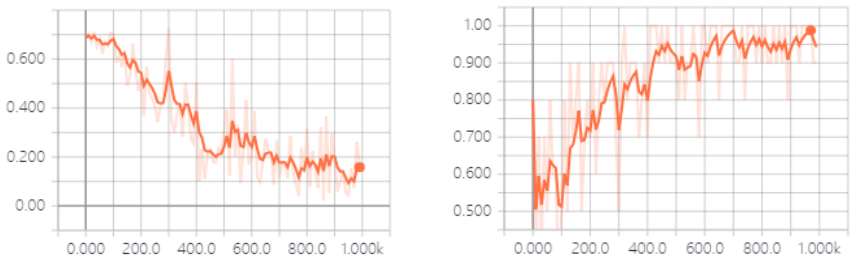


Figure 6-5 Loss and accuracy scale with TWO hidden layers and 128 neurons

Figure 6-6

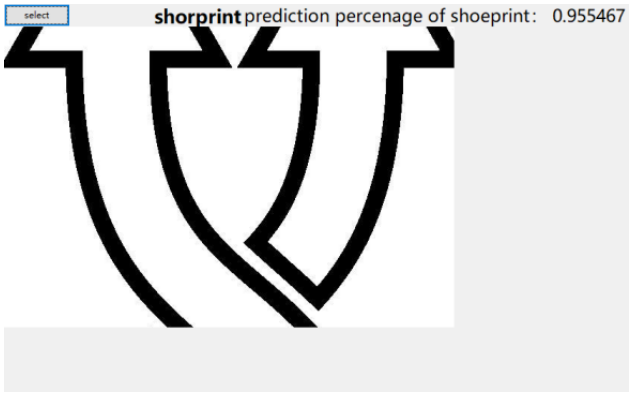


Figure 6-7

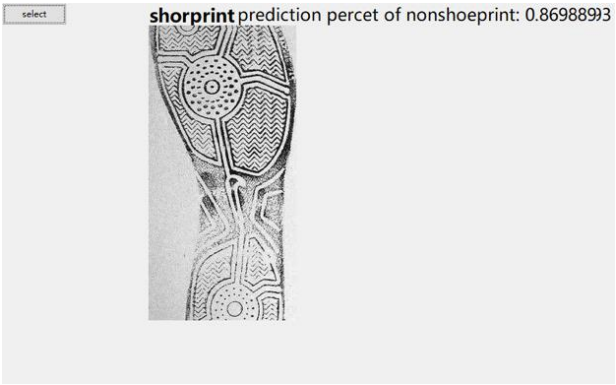


Figure 6-8

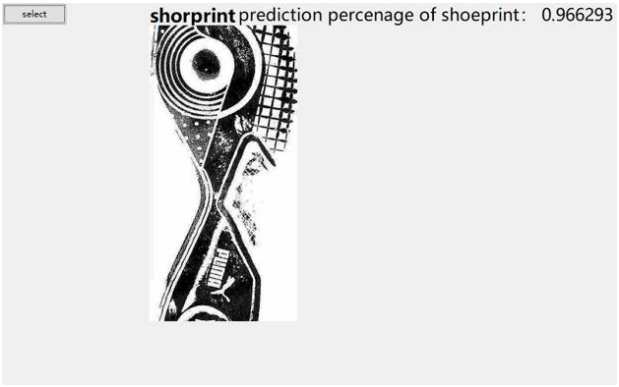


Figure 6-9

Figure 6-10

Figure 6-11

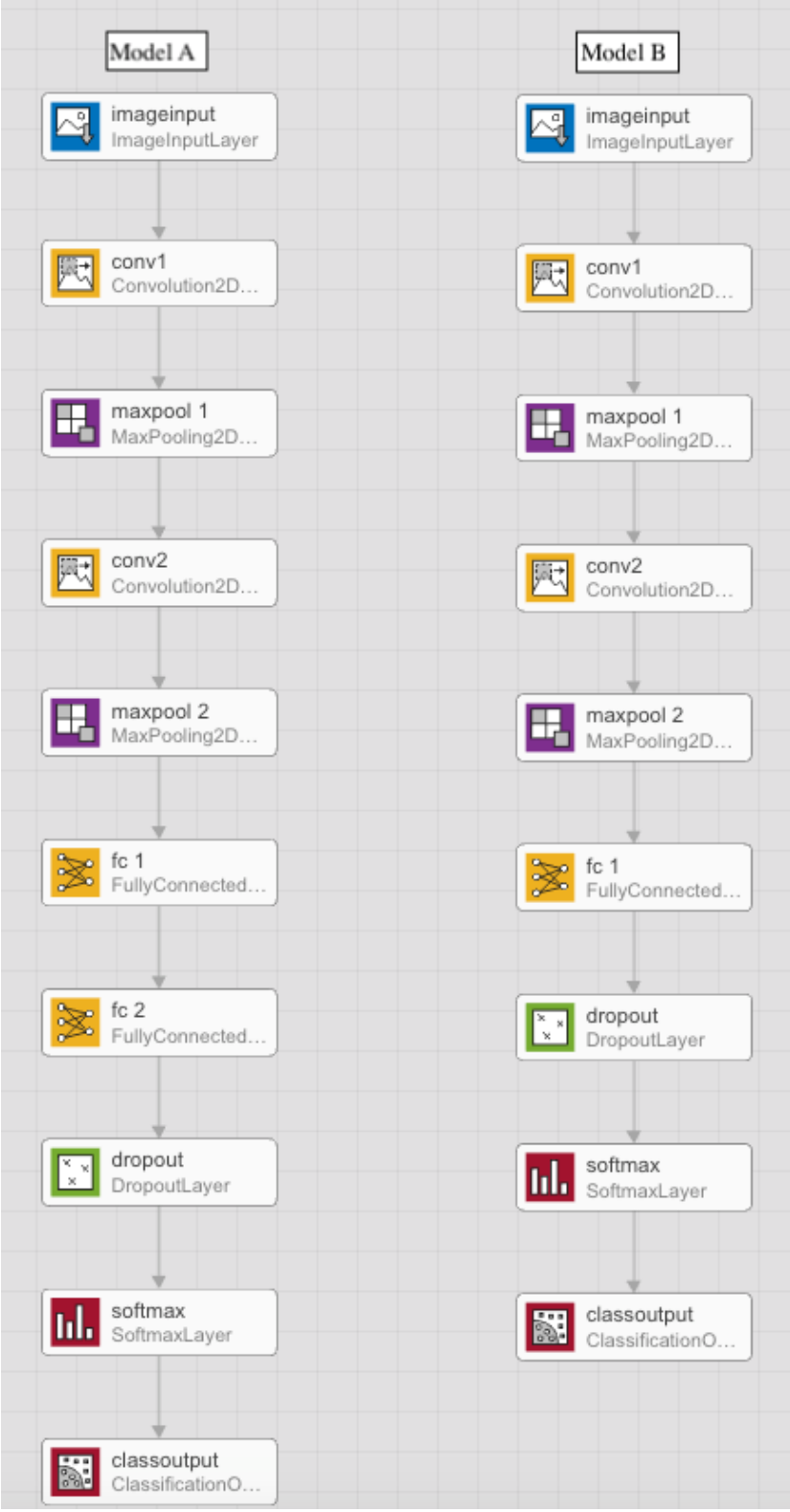

Model A
imageinput
ImageInputLayer
conv1
Convolution2D...
maxpool 1
MaxPooling2D...
conv2
Convolution2D...
maxpool 2
MaxPooling2D...
fc 1
FullyConnected...
fc 2
FullyConnected...
dropout
DropoutLayer
softmax
SoftmaxLayer
classoutput
ClassificationO...
Model B
imageinput
ImageInputLayer
conv1
Convolution2D...
maxpool 1
MaxPooling2D...
conv2
Convolution2D...
maxpool 2
MaxPooling2D...
fc 1
FullyConnected...
dropout
DropoutLayer
softmax
SoftmaxLayer
classoutput
ClassificationO...

Figure 6-12a

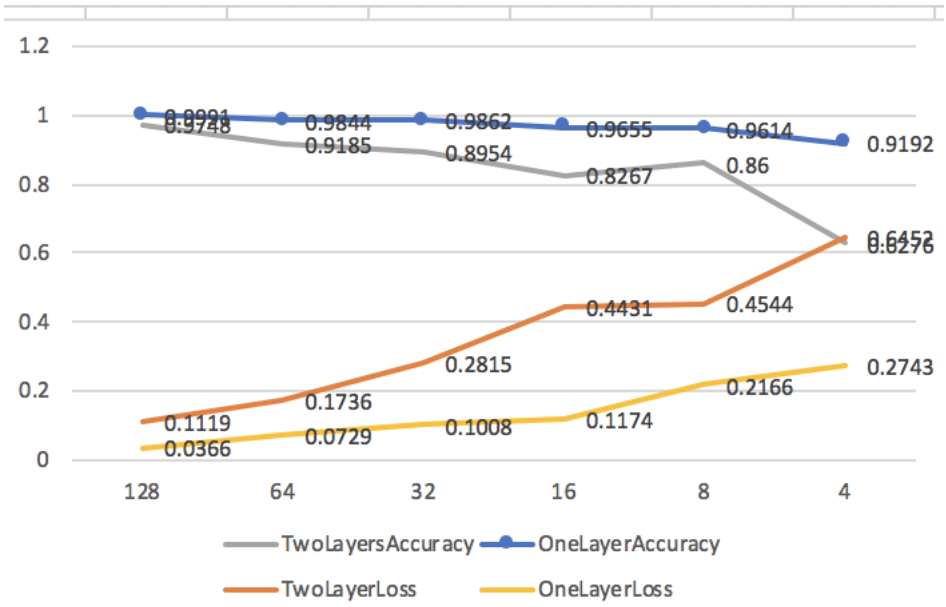


Figure 6-12b

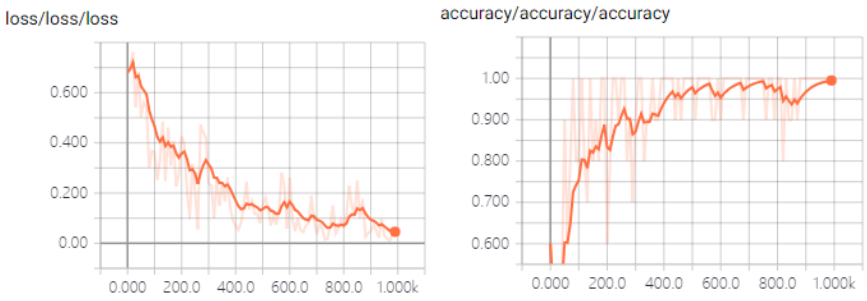


Figure 6-13 Loss and accuracy with one hidden layer with 128 neurons

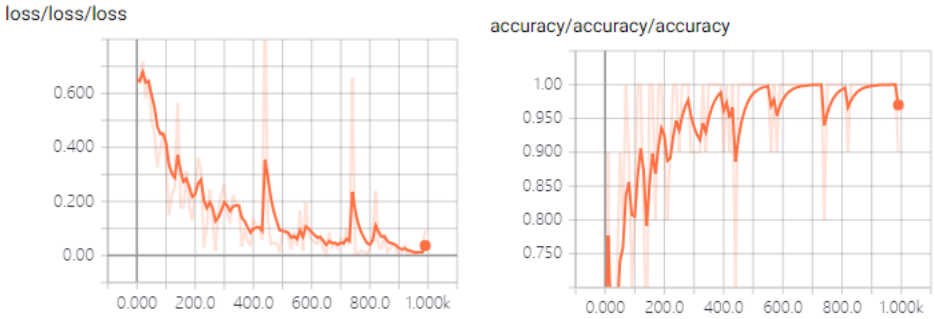


Figure 6-14 Loss and accuracy scale with one hidden layer and 512 neurons

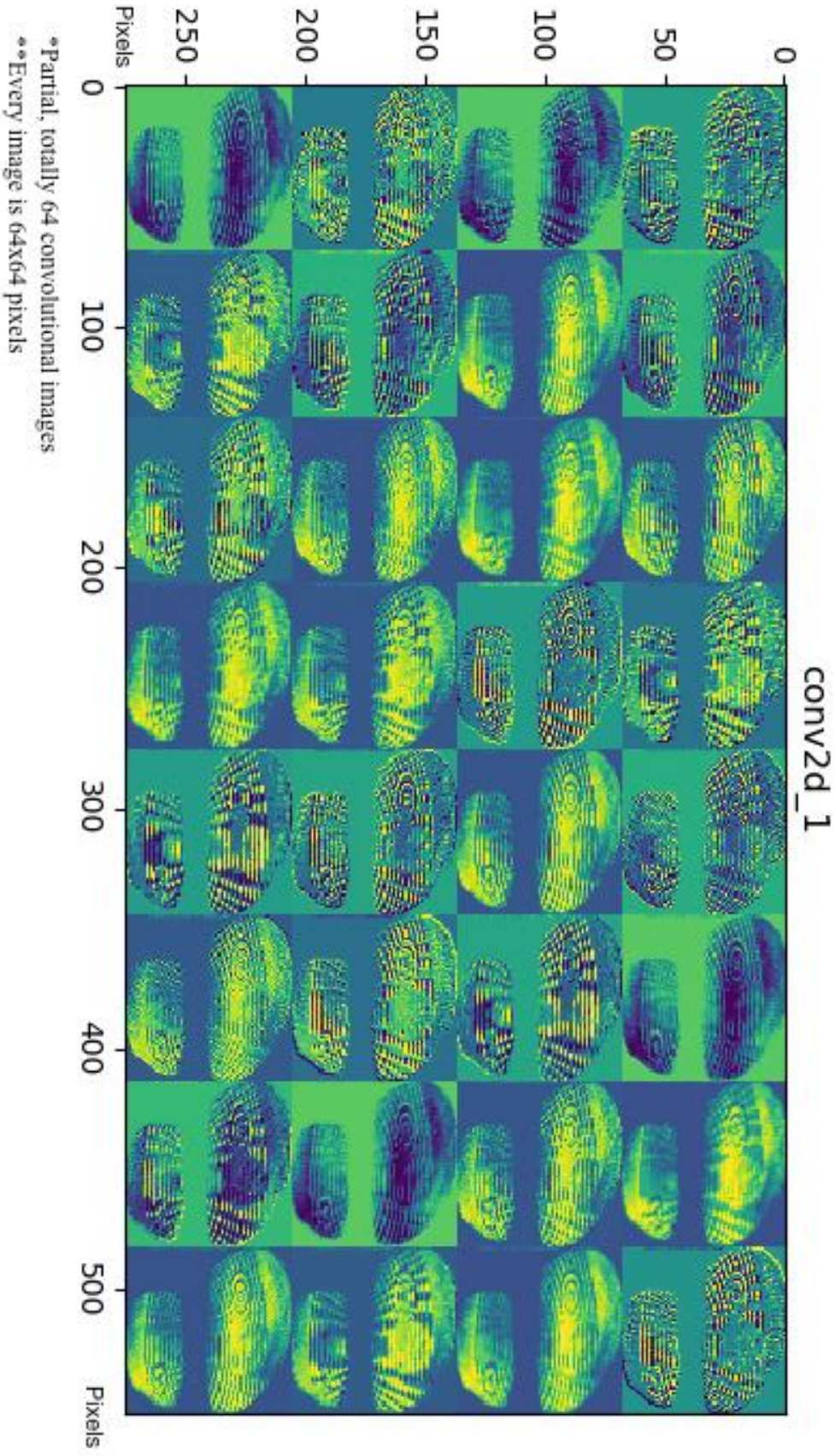

conv2d_1
Pixels
Pixels
0
50
100
150
200
250
0
100
200
300
400
500
*Partial, totally 64 convolutional images
**Every image is 64x64 pixels

Figure 6-15

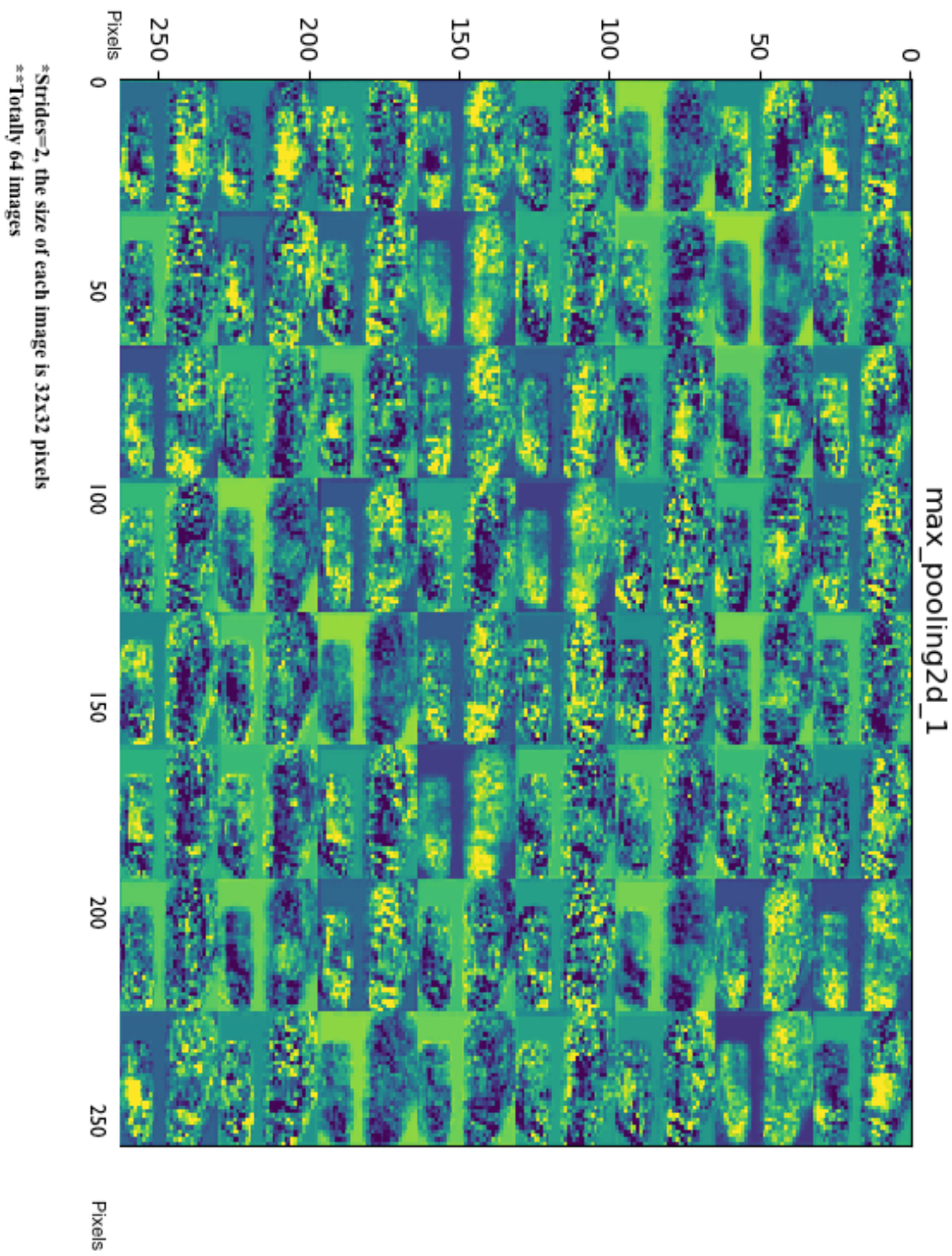

max_pooling2d_1
Pixels
0
50
100
150
200
250
*Strides=2, the size of each image is 32x32 pixels
**Totally 64 images

Figure 6-16

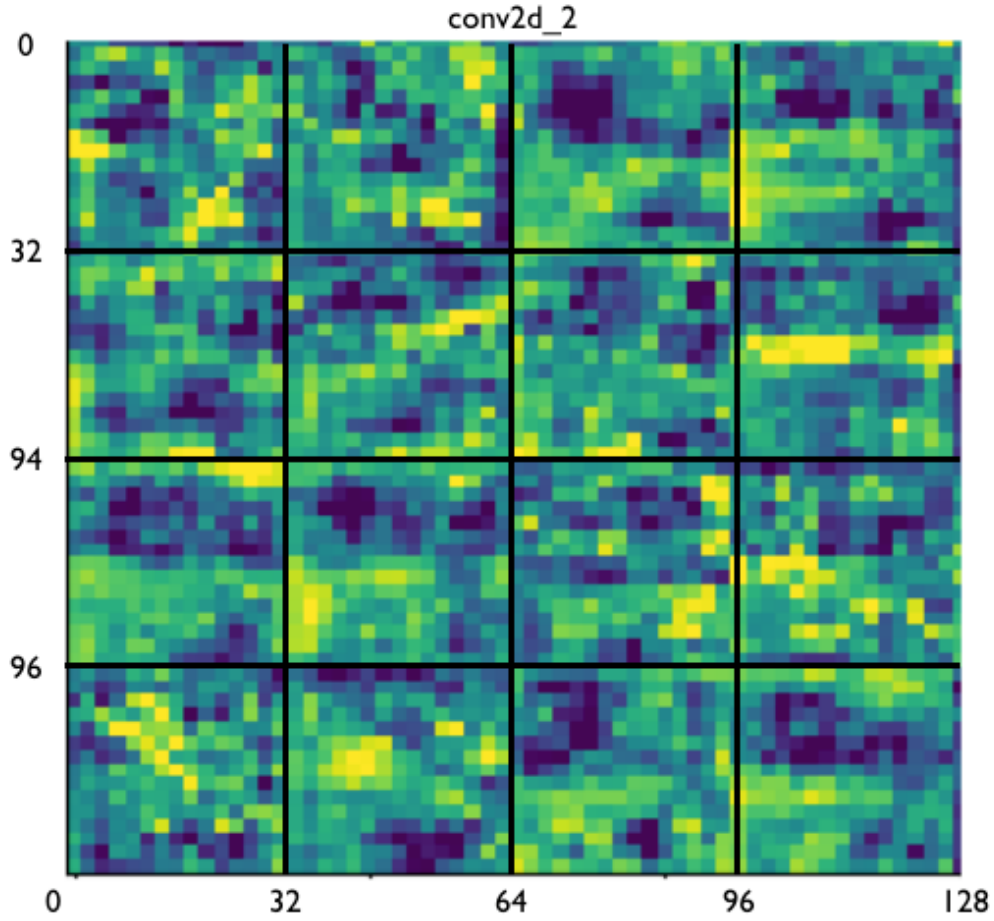
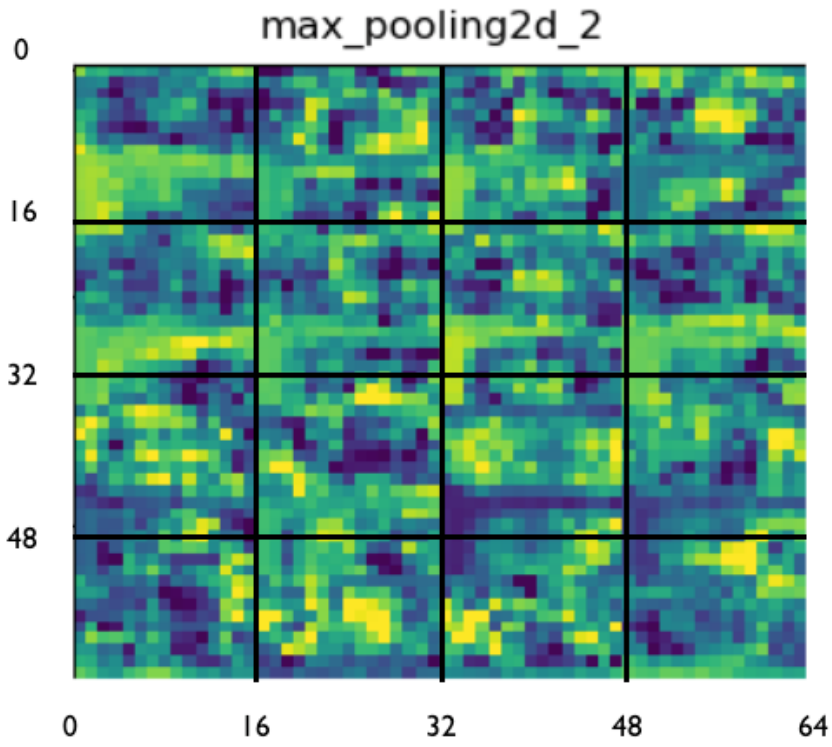
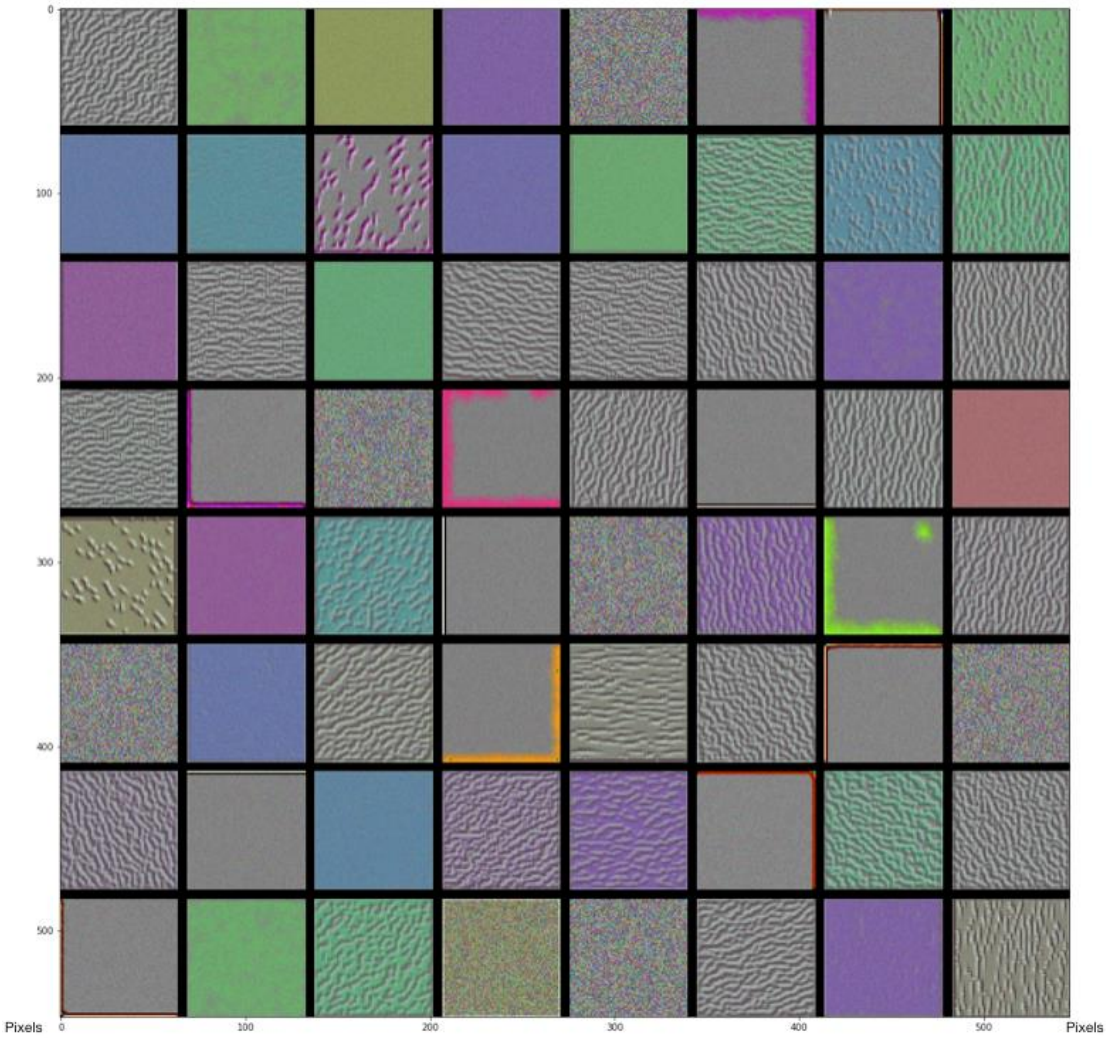

conv2d_2
0
32
94
96
0
32
64
96
128
*Totally 16 fillers
**Each convolutional image is 32x32 pixels

Figure 6-17



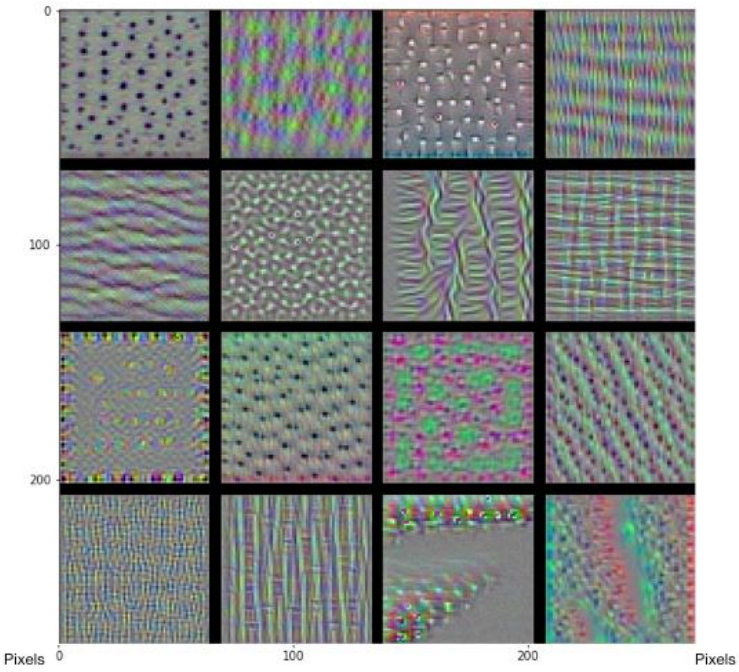
*Strides=2, the size of each image is 16x16 pixels
**Totally 16 images

Figure 6-18



*Each output of noisy image is 64x64 pixels
**Totally 64 filters in Conv1 layer

Figure 6-19 Conv1 Filter



*Each output of noisy image is 64x64 pixels
**Totally 16 filters in Conv2 layer

Figure 6-20

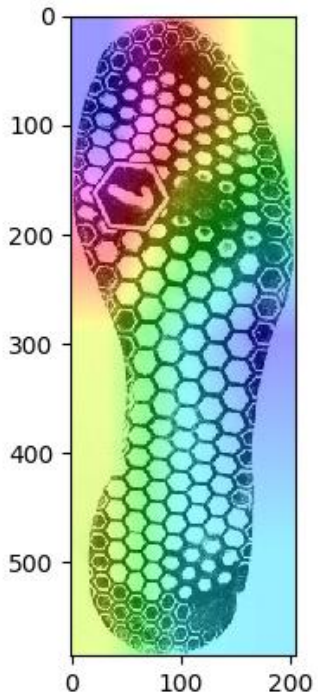


Figure 6-21a

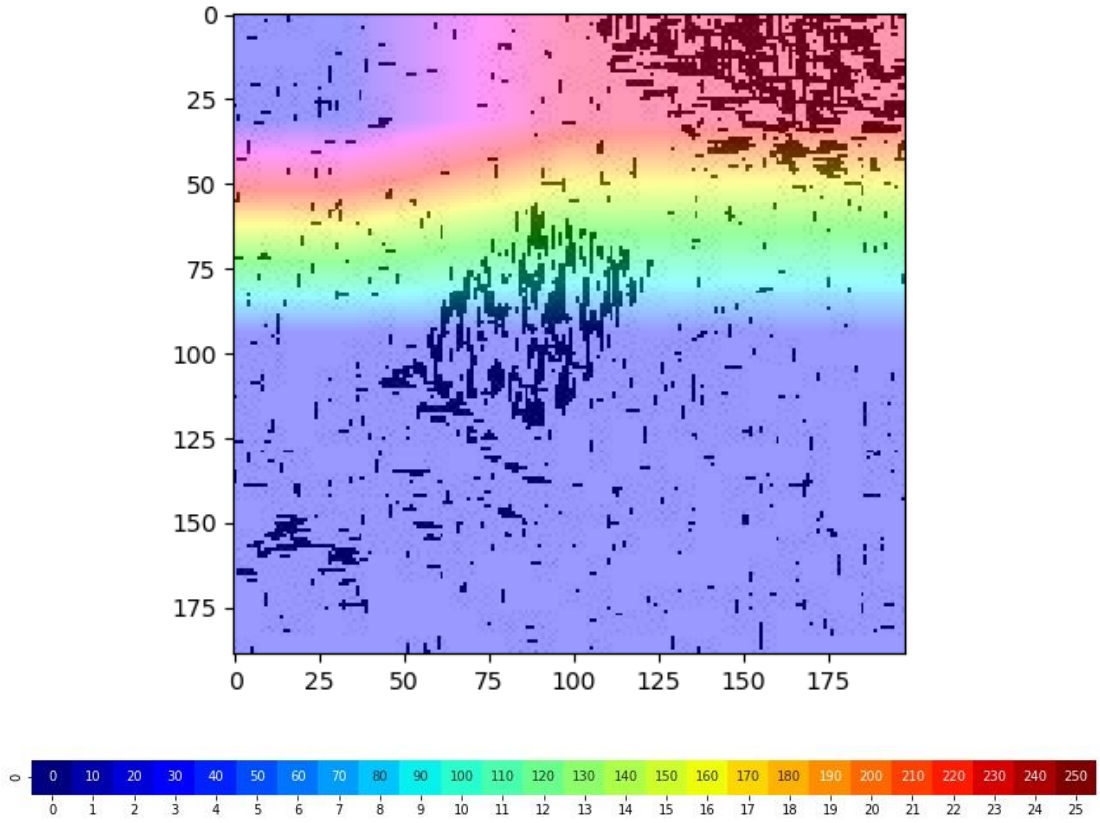

0
25
50
75
100
125
150
175
0 25 50 75 100 125 150 175
0
0 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200 210 220 230 240 250
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

Figure 6-21b

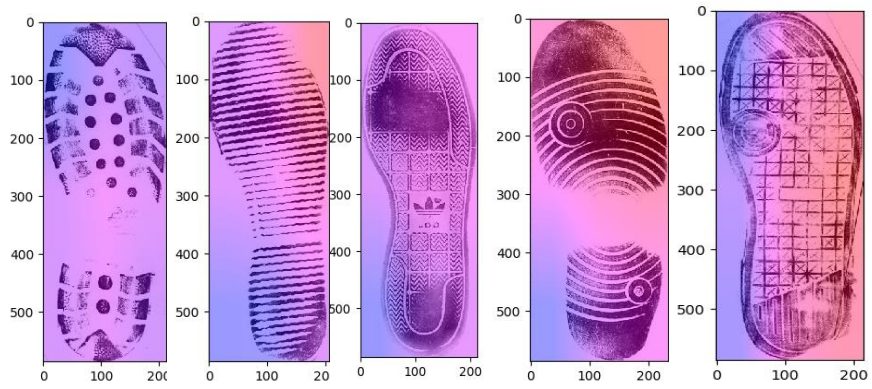


Figure 6-22

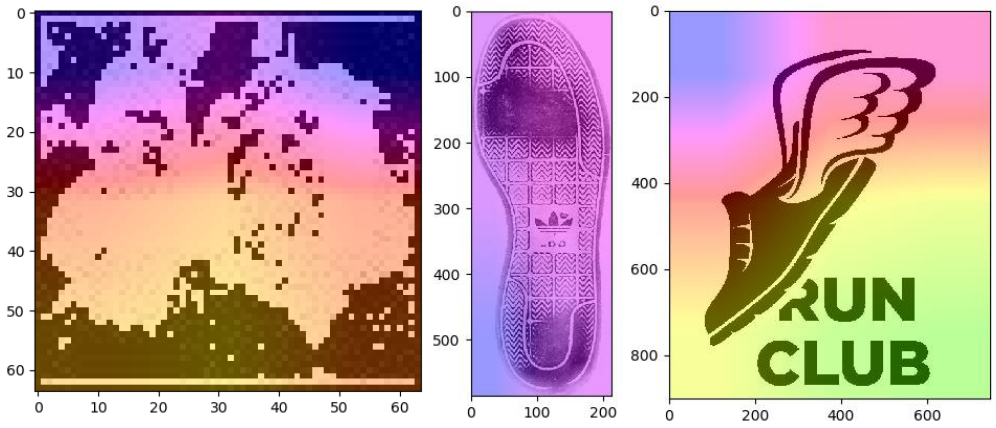


Figure 6-23

8.4 Tables

layer	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
type	input	conv	relu	conv	relu	mpool	conv	relu	conv	relu	mpool	conv	relu	conv	relu	conv	relu	mpool	conv
name	–	conv1_1	relu1_1	conv1_2	relu1_2	pool1	conv2_1	relu2_1	conv2_2	relu2_2	pool2	conv3_1	relu3_1	conv3_2	relu3_2	conv3_3	relu3_3	pool3	conv4_1
support	–	3	1	3	1	2	3	1	3	1	2	3	1	3	1	3	1	2	3
filt dim	–	3	–	64	–	–	64	–	128	–	–	128	–	256	–	256	–	–	256
num filts	–	64	–	64	–	–	128	–	128	–	–	256	–	256	–	256	–	–	512
stride	–	1	1	1	1	2	1	1	1	1	2	1	1	1	1	1	1	2	1
pad	–	1	0	1	0	0	1	0	1	0	0	1	0	1	0	1	0	0	1

layer	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
type	relu	conv	relu	conv	relu	mpool	conv	relu	conv	relu	conv	relu	mpool	conv	relu	conv	relu	conv	softmx
name	relu4_1	conv4_2	relu4_2	conv4_3	relu4_3	pool4	conv5_1	relu5_1	conv5_2	relu5_2	conv5_3	relu5_3	pool5	fc6	relu6	fc7	relu7	fc8	prob
support	1	3	1	3	1	2	3	1	3	1	3	1	2	7	1	1	1	1	1
filt dim	–	512	–	512	–	–	512	–	512	–	512	–	–	512	–	4096	–	4096	–
num filts	–	512	–	512	–	–	512	–	512	–	512	–	–	4096	–	4096	–	2622	–
stride	1	1	1	1	1	2	1	1	1	1	1	1	2	1	1	1	1	1	1
pad	0	1	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0

Table 2-1 Network configuration.

2

No.	Method	Images	Networks	Acc.
1	Fisher Vector Faces [21]	-	-	93.10
2	DeepFace [29]	4M	3	97.35
3	Fusion [30]	500M	5	98.37
4	DeepID-2,3		200	99.47
5	FaceNet [17]	200M	1	98.87
6	FaceNet [17] + Alignment	200M	1	99.63
7	Deep Convolutional Neural Network	2.6M	1	98.95

Table 2-2 Accuracy of Approaches

	NAME	TYPE	ACTIVATIONS	LEARNABLES
1	imageinput 64x64x3 images with 'zerocenter' normalization	Image Input	64×64×3	-
2	conv1 64 3x3x3 convolutions with stride [1 1] and padding 'same'	Convolution	64×64×64	Weights 3×3×3×64 Bias 1×1×64
3	maxpool1 3x3 max pooling with stride [2 2] and padding 'same'	Max Pooling	32×32×64	-
4	conv2 16 3x3x64 convolutions with stride [1 1] and padding 'same'	Convolution	32×32×16	Weights 3×3×64×16 Bias 1×1×16
5	maxpool2 3x3 max pooling with stride [2 2] and padding 'same'	Max Pooling	16×16×16	-
6	fc1 128 fully connected layer	Fully Connected	1×1×128	Weights 128×4096 Bias 128×1
7	fc2 128 fully connected layer	Fully Connected	1×1×128	Weights 128×128 Bias 128×1
8	dropout 80% dropout	Dropout	1×1×128	-
9	softmax softmax	Softmax	1×1×128	-
10	classoutput crossentropyex	Classification Output	-	-

Table 4-1 Structure of CNN and See Figure 4-2 for further details.

Group one				
Datasets	50 x 110			
Nodes	Accuracy(%)			Accuracy (Average)
50	23.5	20	30.5	24.66666667
80	22.67	N.A	N.A	22.67
100	23	25	40	29.33333333
Group two				
200	34.5	30	60	41.5
300	33.5	30.67	37.5	33.89
400	34.5	20	30.75	28.41666667

Table 6-1

Group Three								
Dataset	50 x 100							
	Accuracy in 3 times(%)						Accurancy(AVG)	
Nodes	Training	Testing	Training	Testing	Training	Testing	Training	Testing
3300	33.3	100	50	0	41.7	0	41.6666667	33.3333333
3500	88.9	100	50	0	33.3	100	57.4	66.6666667
4000	66.7	100	88.9	100	63.9	50	73.1666667	83.3333333
Group Four								
Dataset	70 x 176							
	Accuracy in 3 times(%)						Accurancy(AVG)	
Nodes	Training	Testing	Training	Testing	Training	Testing	Training	Testing
6200	36.11	0	80.6	100	64.4	100	60.37	66.6666667
7000	63.9	0	58.3	100	47.3	0	56.5	33.3333333
7500	47.3	100	75	0	52.8	100	58.3666667	66.6666667
Group Five								
Dataset	90 x 227							
	Accuracy in 3 times(%)						Accurancy(AVG)	
Nodes	Training	Testing	Training	Testing	Training	Testing	Training	Testing
10000	66.7	100	77.8	100	N.A.	N.A	72.25	100

Table 6-2

Group Six								
Dataset	90 x 227							
	Accuracy in 3 times(%)						Accurancy(AVG)	
Nodes	Training	Testing	Training	Testing	Training	Testing	Training	Testing
5000	90	50	43.3	50	63.3	50	65.5333333	50
5500	90	100	83.3	50	96.7	100	90	83.3333333
6000	96.7	0	76.7	100	90	100	87.8	66.6666667
7000	53.3	0	46.7	100	90	50	63.3333333	50
8000	63.3	100	93.7	50	90	10	82.3333333	53.3333333

Table 6-3

Convolutional Neural Network		
Two Full Connetion Layers		
Nodes	TwoLayersAccuracy	TwoLayerLoss
128	0.9748	0.1119
64	0.9185	0.1736
32	0.8954	0.2815
16	0.8267	0.4431
8	0.86	0.4544
4	0.6276	0.6452
One Full Connection Layer		
Nodes	OneLayerAccuracy	OneLayerLoss
128	0.9991	0.0366
64	0.9844	0.0729
32	0.9862	0.1008
16	0.9655	0.1174
8	0.9614	0.2166
4	0.9192	0.2743
512	0.9698/0.9998	0.0367

Table 6-4

	ANN	SVM	CNN-ModelA	CNN-ModelB
Accuracy	90%	95.0%	97.48%	99.91%
sensitivity	88.20%	95.2%	97.60%	100%
specificity	92.30%	94.7%	95%	97.05%

Table 6-5