

Real-Time Face Detection and Recognition Based on Deep Learning

Hui Wang

A thesis submitted to Auckland University of Technology
in partial fulfillment of the requirements for the degree of
Master of Computer and Information Sciences (MCIS)

2018
School of Engineering, Computer and Mathematical Sciences

Abstract

Face recognition is one of the most important applications in video surveillance and computer vision. However, the conventional algorithms of face recognition are susceptible to multiple conditions, such as lighting, occlusion, viewing angle or camera rotation. Therefore, face recognition based on deep learning can greatly improve the recognition speed and compatible external interference. In this thesis, we use convolutional neural networks (ConvNets) for face recognition, the neural networks have the merits of end-to-end, sparse connection and weight sharing.

The purpose of this thesis project is to identify the name of different people based on location of the detected box of a face. Then, we can obtain recognition results with different confidence under various distances.

This thesis presents different methods with comparisons, namely, comparing the training results and the test results of different parameters under the same model, training results of the same test video under different models. We find that the recognition accuracy of this model is mainly affected by face proportion and the number of samples. If we get larger proportion of a face on screen, then we have higher recognition accuracy; if we obtain much greater number of samples, we can get higher recognition accuracy.

In this work, we first collect sufficient samples as our dataset and use the suitable model embedded in the platform Google TensorFlow to complete the training and test. We collected five different faces and obtained 500 images on each face as training set, each of which can be cropped and rotated by using 50 different angles of the picture having a human face, of which 40 for training, 10 for verification.

The use of neural networks for face recognition improves the speed of recognition. The contributions of this thesis are: (1) The use of elliptical markers can identify a human face including rotation and position. (2) The confidence of human face recognition is mainly affected by the proportion of face occupied on the screen.

Keywords: CNNs, face recognition, data augmentation, SSD, Inception v2

Table of Contents

Abstract	I
List of Figures	IV
List of Tables.....	VI
Attestation of Authorship	VII
Acknowledgement.....	VIII
Chapter 1 Introduction.....	1
1.1 Background and Motivation	2
1.2 Research Questions	4
1.3 Contribution.....	5
1.4 Objective of This Thesis.....	5
1.5 Structure of This Thesis.....	6
Chapter 2 Literature Review	7
2.1 Introduction	8
2.2 Meta-architectures	10
2.2.1 Single Shot Detector (SSD).....	14
2.2.2 Faster R-CNN.....	16
2.3 Feature Extractor of CNNs Model	18
2.3.1 LeNet-5.....	19
2.3.2 Dan Ciresan Net	19
2.3.3 AlexNet	20
2.3.4 VGGNet.....	20
2.3.5 Inception.....	21
2.3.6 ResNet	23
2.4 Overfitting Problems and Solutions	24
2.4.1 Data Augmentation.....	25
2.4.2 Dropout.....	26
2.5 Activation and Loss Function.....	27
2.6 Multilayer Perceptron.....	28
Chapter 3 Methodology	30
3.1 Research Designing	31
3.1.1 Data Sources and Data Collection	32
3.1.2 Face Labelling	33
3.1.3 Data Augmentation.....	36
3.1.3.1 Scaling Conversion.....	38

3.1.3.2 Rotation Conversion.....	39
3.1.3.3 Cropping and Flipping.....	40
3.1.3.4 Summary	40
3.2 The Principle of CNNs	41
3.3 Model Design of DNNs.....	43
Chapter 4 Results.....	50
4.1 Training Model.....	51
4.2 Comparison and Analysis of the Four Models	52
4.3 Effect of Face Ratio on Accuracy	58
4.4 Classification Results	62
4.5 Limitations of the Research.....	63
Chapter 5 Analysis and Discussions.....	65
5.1 Analysis	66
5.1.1 SSD Inception (v2) Model.....	66
5.1.2 Simplified SSD model.....	68
5.1.3 Comparisons	70
5.1.3.1 Comparisons of Architecture.....	70
5.1.3.2 Comparisons of Training Time	71
5.1.3.3 Activation and Loss Function Comparisons.....	73
5.1.3.4 Comparisons of Loss Value.....	74
5.1.3.5 Precision Comparisons	77
5.2 Discussions	78
Chapter 6 Conclusion and Future Work	79
6.1 Conclusion.....	80
6.2 Future Work	80
References	82

List of Figures

Figure 2.1 Flowchart of DNNs fundamental.....	11
Figure 2.2 Schematic diagram of DNNs meta-model.....	11
Figure 2.3 Schematic diagram of modern DNNs meta-model.....	12
Figure 2.4 Faster R-CNN architecture of the feature concatenation scheme.....	17
Figure 2.5 Flowchart of object detection of Faster R-CNN.....	17
Figure 2.6 Classic Inception module.....	22
Figure 2.7 The new version of Inception module.....	23
Figure 3.1 The steps to achieve face recognition.....	31
Figure 3.2 Data collection by taking a video.....	33
Figure 3.3 Label the face position.....	34
Figure 3.4 Marked files saved in text format.....	35
Figure 3.5 Contents of the labelled files.....	35
Figure 3.6 Flowchart of data augmentation.....	36
Figure 3.7 Schematic diagram of data augmentation.....	41
Figure 3.8 The working principle of CNNs.....	42
Figure 3.9 The working principle of feature extraction of CNNs.....	42
Figure 3.10 Model training structure.....	43
Figure 3.11 Model design structure.....	44
Figure 4.1 The resultant example of the test video.....	51
Figure 4.2 Training curve: $box=1$ and $w=1$	53
Figure 4.3 Training curve: $box=1$ and $w=10$	54
Figure 4.4 Training curve: $box=2$ and $w=1$	55
Figure 4.5 Training curve: $box=2$ and $w=10$	56

Figure 4.6 Accuracy comparisons of four models on the validation set.....	57
Figure 4.7 Recognition results.....	58
Figure 4.8 Face with different sizes.....	59
Figure 4.9 The relationship between accuracy and facial proportion	60
Figure 4.10 The relationship between accuracy and facial angle.....	61
Figure 4.11 Results of face recognition and classification.....	63
Figure 5.1 Flowchart of SSD Inception (v2) model.....	66
Figure 5.2 Architecture of SSD Inception (v2)	68
Figure 5.3 Architecture of simplified SSD network model.....	69
Figure 5.4 Results of loss function of the model SSD Inception (v2)	75
Figure 5.5 Results of loss function of the Simplified SSD Model I.....	75
Figure 5.6 Results of loss function of the Simplified SSD Model II.....	76
Figure 5.7 Results of loss function of the Simplified SSD Model III.....	76
Figure 5.8 Results of loss function of the Simplified SSD Model IV.....	77

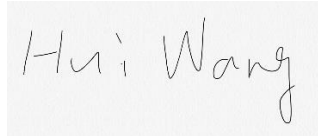
List of Tables

Table 3.1 Classes in our experiment.....	32
Table 4.1 Comparisons of training results of the four models.....	52
Table 5.1 The distribution of SSD Inception (v2)	67
Table 5.2 Comparisons of architectures.....	71
Table 5.3 Comparisons of training process.....	72
Table 5.4 Comparisons of activation function.....	73
Table 5.5 Comparisons of loss function.....	74
Table 5.6 Comparison of precisions.....	77

Attestation of Authorship

I hereby declare that this submission is my own 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 acknowledgments), 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:

A rectangular box containing a handwritten signature in black ink that reads "Hui Wang".

Date: 21 June 2018

Acknowledgement

This research was completed in response to the requirements of the Master of Computer and Information Science (MCIS) program at the School of Computer and Mathematical Sciences (SCMS) in the Auckland University of Technology. First, I would like to thank my parents for supporting me to study in Auckland and give me financial help. I also want to thank my friends who supported me in my life and study.

In this research project, my deepest thanks are given to my primary supervisor Dr Wei Qi Yan. I thank him very much for giving me technical and academic teaching. Under his leadership, I learned a lot of knowledge in deep learning and I am confident I can achieve my master's degree with his assistance. In addition, I would also like to appreciate my second supervisor Professor Nikola Kasabov who gave me invaluable comments on my work. In addition, I am very grateful to my third supervisor Dr Michael Watts, who gave invaluable advice on my thesis.

Hui Wang

Auckland, New Zealand

21 June 2018

Chapter 1

Introduction

The first chapter of this thesis consists of five sections. In the first section, the background and motivation are introduced, face recognition can be used not only for access control systems, but also for supermarket shopping, such as payment methods, to facilitate ordinary people's lives. The second section and the third section of this thesis will introduce the main research questions and contributions of this thesis. The fourth section will introduce the objective. Finally, in the fifth section, we will mainly outline the structure of this thesis.

1.1 Background and Motivation

With rapid development of computer technology, computers are becoming intelligent. Artificial Intelligence (AI) has also become a crucial branch of computer science, it has also become the core of contemporary high-tech, its application also involves various fields. Among them, computer vision (CV) is one of the important areas of artificial intelligence (Rautaray & Agrawal, 2015), it attempts to obtain information from images or data, uses computer algorithms to identify or track images and videos, and then performs image processing to achieve the purpose of making computers replace human eyes (Koch, 2018) (Yan, 2017).

Nowadays, face recognition is also widely used in our ordinary life. There are many types of applications of face recognition, e.g., gender, identity, age and emotion are the most important characteristics of human (Lawrence, Campbell, & Skuse, 2015) (Gu, Nguyen, & Yan, 2016). Face recognition mainly includes three key aspects: geometric structure (Gao, Huang, Gao, Shen, & Zhang, 2015), subspace local features (Liao, Hu, Zhu, & Li, 2015), and deep learning (Parkhi, Vedaldi, & Zisserman, 2015). Traditional face recognition algorithms, such as LBP, PCA, and LDA, have drawbacks in feature extraction and recognition (Dewangan & Verma, 2016). Because they need to make some effective features to make the face recognizable; each feature is separated, usually the test results are not optimal. Deep learning, as a new technology in recent years, has made great contributions in speech recognition, image recognition and license plate recognition, and has played a key role in the process. So, this thesis will mainly introduce the use of deep learning to achieve face recognition and confirm the identity of the objects.

With the development of neuroscience, computer scientists have found that brain signals are transmitted through a complex structure, if time permits, some of the characteristics can be used to understand the signals, which lead to the emergence of deep learning for the establishment and simulation of human brain for analysis and learning (Kriegeskorte, 2015). Convolution Neural Networks (ConvNets) have been successfully applied to visual imagery in the past few years, such as flame detection (Shen, Chen, Nguyen, Yan, 2018) (Jiao, Weir, & Yan, 2011), image denoising (Liu, Yan, & Yang, 2018), integrated multi-scale event verification (Gu, Yang, Yan, & Klette,

2017) and license plate recognition (Li, Nguyen, & Yan, 2018). One of the most important factors is the need to provide a large amount of training data. But in face recognition, due to lack of large scale of data sets, some experiments will be limited (Karpathy, Toderici, Shetty, Leung, Sukthankar, & Fei-Fei, 2014).

From 2012, AlexNet defeated traditional algorithms in the field of image recognition in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), deep learning algorithms have gradually become the mainstream of image recognition. Although deep learning has been widely used in recent years, it also has a long history of development (Sivaramakrishnan, Antani, Candemir, Xue, Abuya, Kohli, & Thoma, 2018).

In 1943, the neuroscientist McCulloch and logic scholar Walter Pitts proposed that the first artificial neuron model: MP model, they tried to connect the basic units together to understand how human brain produced highly complex patterns. This has made a significant contribution to the development of neural networks (Bressloff, Ermentrout, Faugeras, & Thomas, 2016). In 1958, Rosenblatt published the infected person's speech based on the MP model, greatly developed the neural network theory and applied it to the real problems (Ayouche, Aboulaich, & Ellaia, 2017). In 1986, Rumelhart et al. proposed backpropagation algorithm, which is an important method in neural networks to calculate the error of neurons after data processing (Raza & Khosravi, 2015) (Wang & Yan, 2016). This algorithm is still the most popular one now and is one of the most widely used artificial neural networks in artificial intelligence (Yan, Chambers, & Garhwal, 2015).

In recent years, deep learning algorithms such as ConvNets, Recurrent Networks (RNNs) have achieved great success in experiments (Zheng, Yan, & Nand, 2018). The convolutional layer extracts visual features through local connections and weight sharing. It has the advantage of feature extraction through dimension reduction of convolutional layers. After nonlinear mapping, the network can automatically form feature extractors and classifiers adapted to the task from training samples. ConvNets is essentially a forward feedback network, aiming at simplifying the pre-processing operation and simulating the alternating cascade structure of the simple and complex cells used for feature extraction (Cao, Liu, Yang, Yu, Wang, Wang, & Ramanan, 2015).

Face recognition, as biometrics, is a significant components of video surveillance and visual security which is wildly applied to identification today (Cui & Yan, 2016).

In some large shopping malls, face recognition is used to retrieve goods, monitor the passengers in the shopping mall, and provide users with more convenient services. At the entrance of the school or company, face recognition is used to implement access control system, which prevents the entry of foreign persons and ensures the security of premise. With the development of technology, face recognition can also be used to deblock devices, such as smart phones and computers. This can better protect the privacy of users, improve the security of the data. Face recognition, as a special part of human-computer interaction through a computer identifies users, serves them with great convenience (Parmar & Mehta, 2013).

Therefore, with the improvement of image and video processing technology, the appearance of deep learning has made a great contribution to computer vision. This also makes it possible to achieve face recognition with better algorithms and models. After surveying the literature, the study of this thesis has been achieved.

1.2 Research Questions

Face recognition has been widely used in recent years; extracting facial features and classifying a given face are the basic procedure of face detection and recognition. Therefore, the research questions in this thesis are:

Question:

What kind of technology and methods can achieve face recognition?

Although face recognition technology matured, there also has a big gap with fingerprints and retinas in recognition. After a full conclusion, we find that the main reasons that may affect face recognition are:

- 1) Uncertainty in the acquisition of face images, such as the direction of light, and the intensity of light, etc.
- 2) Face diversity, such as beards, glasses, hairstyles, etc.
- 3) Uncertainty of human faces, such as facial expressions, etc.

Because there are multiple factors affecting the recognition results, the recognition results will be more complicated in the process of face recognition. According to these

existing factors, we tried to study the following questions and the sub-question of the above main question is:

“What kind of algorithm is suitable for face recognition? Which method can make recognition faster and compatible with the diversity of face changes?”

In the process of recognition, because the proportion of face occupied by screen is various, the degree of confidence will be different. Therefore, we have extended the following research question:

“What is the relationship between confidence and the proportion of faces?”

Since the core idea of this thesis is to find more effective methods to improve the accuracy of recognition results, we need to choose more advanced models and algorithms to achieve it.

1.3 Contribution

The contribution of this thesis is based on deep learning for face recognition. We will conduct the experiments through real-time recognition; for example, when people move close to a camera, the system will verify the proportion of a human face and its confidence, we will take the proportion as a major reason. The experiment has four parts: 1) collect the data, 2) accept the command parameters, 3) define the neural network model, 4) training model.

Moreover, finding suitable algorithms which are suitable for face recognition from deep learning is also introduced in this thesis. Comparing and analysing the accuracy of different parameters will be conducted in the same model framework.

In addition, because this thesis uses neural networks to study, the focus of this thesis is on the models established by using SSD. At the end of this thesis, we will also compare the results of models by using the same dataset to prove the suitability of our algorithm.

1.4 Objective of This Thesis

Firstly, we need to collect large-scale data sets of human faces under the requirement of deep learning. But we think that it is too slow and may waste our time; in terms of data sets, our objective is to use reasonable data augmentation methods to segment the collected images so as to reach the number of training datasets.

Secondly, under the condition of external disturbance factors, we tried to find an algorithm to avoid the influence of environment and light and explore whether the system can recognize the occluded face and the rotated face position.

Finally, we will compare experimental results and explore the effect of facial proportions on confidence in recognition.

1.5 Structure of This Thesis

This thesis consists of six chapters:

In the second chapter, we mainly introduce literature review. First, we will introduce feature extraction method of convolutional neural networks and list the relatively new algorithms and models used in recent years, outline the working principles, and then propose solutions for overfitting problems. Therefore, the second chapter is more about learning and understanding the results of previous studies and summing up experiences to better conduct the next research.

In the third chapter, we will discuss the methodology of this thesis. These include data collection methods, data augmentation methods, and model design methods. The design and implementation of the experimental process will also be listed.

In the fourth chapter, the results of the experiment are mainly introduced, including training and test results of the proposed model, which also includes analysing the results of the experiment under the conditions of four different parameters. We will also show the figures and tables with an intuitive explanation.

In the fifth chapter, we will analyse and discuss our experimental results and compare the results with different models. To prove the advantages of our algorithm, the conclusions and future work will be presented in Chapter 6.

Chapter 2

Literature Review

After in-depth study and understanding of the previously studied algorithms and structures, we know that the core of this thesis is based on test video to identify people, including face location and face classification. Therefore, this chapter will introduce and summarize a completely new algorithm for face recognition in deep learning and outline its principle.

2.1 Introduction

Deep Neural Networks (DNNs) have a development history of more than 10 years, but it has not been enough attention in the scientific research field until the concept of deep learning was presented. The DNNs was originated in 1943 when Walter Pitts and Warren McCulloch created a computer model based on human brain neural network that eventually became a hot topic after a century of development (McCulloch & Pitts, 1943). The Neocognitron was the first artificial neural network that introduced convolutional neural networks (CNNs), where the receptive field of a convolutional unit gave weight vector (Fukushima & Miyake, 1982). In 2006, Hinton presented the concept of a fast learning algorithm for deep belief net, and he also presented the deep learning methods and improvement of DNNs training model (Hinton, Osindero, & Teh, 2006). He presented two main arguments of deep learning in the literature, both of them are the multi-layer DNNs model, which has a strong feature learning ability and the method of layer-by-layer training to achieve the deep learning model. However, deep learning remains in theoretical stage until Hinton with his team win the championship of ImageNet Large Scale Visual Recognition Challenge (ILSVRC) by using Alexnet Model using a deep learning algorithm in 2012, it is an important milestone in DNN's history. They got the accuracy of 84.7% in the ILSVRC, which is far better than the previous best result of 74.3%. The deep learning has aroused widespread concern after the ILSVRC and began to develop rapidly (Schmidhuber, 2015).

There are four reasons that why AlexNet can succeed and DNN becomes one of the most popular topics: 1) big data training with millions ImageNet image data, 2) assisting with GPU accelerated training, 3) methods of prevent over-fitting, e.g., dropout and data augmentation, 4) development of nonlinear activation function, e.g., ReLU (Krizhevsky, Sutskever, & Hinton, 2012).

In this thesis, we will also discuss the methods of preventing overfitting of data augmentation and dropout operation (Lee, Gallagher, & Tu, 2016). In addition, this thesis will introduce the operation of DNNs structure.

The research field of this thesis is object detection, which includes object identification and classification (Satat, Tancik, Gupta, Heshmat, & Raskar, 2017). In recent years, DNN has been applied to object detection. Object detection using DNNs

has been developed rapidly based on the success of AlexNet model, and many models have been developed and popularized after AlexNet model, such as Region Convolutional Neural Network (RCNN), Spatial Pyramid Pooling (SPP-net), Fast RCNN, Faster RCNN, You Only Look Once (YOLO), and Single Shot MultiBox Detector (SSD) (Huang, Rathod, Sun, Zhu, Korattikara, Fathi, & Murply, 2017). Among them, from RCNN to Faster RCNN, they elaborate the model optimization process. The Faster RCNN has a good accuracy, but the effect on real-time detection is still not ideal (Zhang, Lin, Liang, & He, 2016); the main feature of YOLO model is very fast for detection, but the accuracy of small object detection of YOLO is also not good (Molchanov, Vishnyakov, Vizilter, Vishnyakova, & Knyaz, 2017). The SSD model is one of the first attempts to use a ConvNet's pyramidal feature hierarchy, which adds the ability to map features onto multiscale feature maps (Lin, Dollár, Girshick, He, Hariharan, & Belongie, 2017); it is a very advanced object detection model, which takes into account the detection accuracy, it is one of the best choices for real-time object detection.

DNNs has promoted the third wave of artificial intelligence research (LeCun, Bengio, & Hinton, 2015). The achievements made by using DNNs have also proved its broad prospects for development in recent years (Jordan & Mitchell, 2015). The DNN software framework plays a key role in the development and application of neural network models, many software frameworks are proposed and have been widely used, such as TensorFlow (Abadi, Barham, Chen, Chen, Davis, Dean, & Kudlur, 2016), Caffe, Torch7 (Jia, Shelhamer, Donahue, Karavev, Long, Girshick, & Darrell, 2014), Theano (Bergstra, Breuleux, Lamblin, Pascanu, Delalleau, Desjardins, & Kaelbling, 2011), Keras, Lasagne (Shatnawi, Al-Bdour, Al-Ourran, & Al-Avvoub, 2018), and Chainer (Tokui, Oono, Hido, & Clayton, 2015). Among them, we choose TensorFlow as our framework in this thesis.

TensorFlow plays a decisive role in development and application of CNNs, which has integrated most common units in deep learning framework (Shi, Wang, Xu, & Chu, 2016), it is also independent on open source framework for DNNs model based on C++, python (Flores, Barrón-Cedeño, Rosso, & Moreno, 2011), and CUDA (Kirk, 2007). One of the important features of TensorFlow is flexible portability, which makes it easy to deploy the same codes to multiple CPUs and GPUs with no modification. Another reason of choose TensorFlow as our framework is that TensorFlow library has the

DNNs model of the most widely used software framework. Deep learning framework as the core role of DNNs models must be explicit before implementation of DNNs model. In a typical DNNs framework model, such as TensorFlow, it is constituted by two phases “Define” and “Run”, namely, Define-and-Run (Tokui, Oono, Hido, & Clayton, 2015).

In the Define phase, the DNNs model constructs a computational graph based on specific inter-layer connections, initial weights, and activation functions. After the computational graph has been built in memory of computer, the forward computation and backward computation are set; then, it will enter the “Run” phase. In the Run phase, it repeats training the set of training samples in the computational graph, finally achieves the goal of reducing the loss function and optimizing the results of DNNs model.

2.2 Meta-architectures

The architecture of deep learning consists of a computational graph that is most conveniently constructed by composing layers with other layers (Perez, 2017). Many processes have been made remarkable achievement on object detection since the use of convolutional neural networks (Schmidhuber, 2015). The working principle of DNNs model is to collect the original data set, after duplicated layers of extraction and optimization, eventually minimized the loss function and obtain the best object detection results, we can see the flowchart in Figure 2.1. The meta-models of DNNs are objectives, optimizers, activations, metrics, and layers. Normally, an objective is a function and an optimizer is an algorithm. Consequently, we can deduce the figure of DNNs meta-model, see Figure 2.2. With the rapid development of DNNs, the objective function has become a neural network and the optimizer has also become a neural network. Thus, DNNs meta-model will be evolved from Figure 2.2 into Figure 2.3.

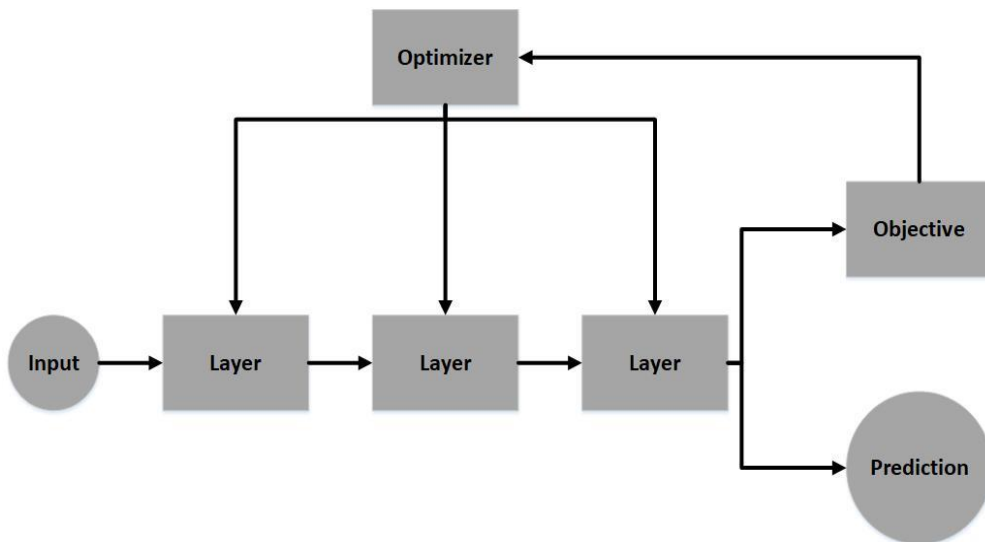


Figure 2.1 Flowchart of DNNs fundamental

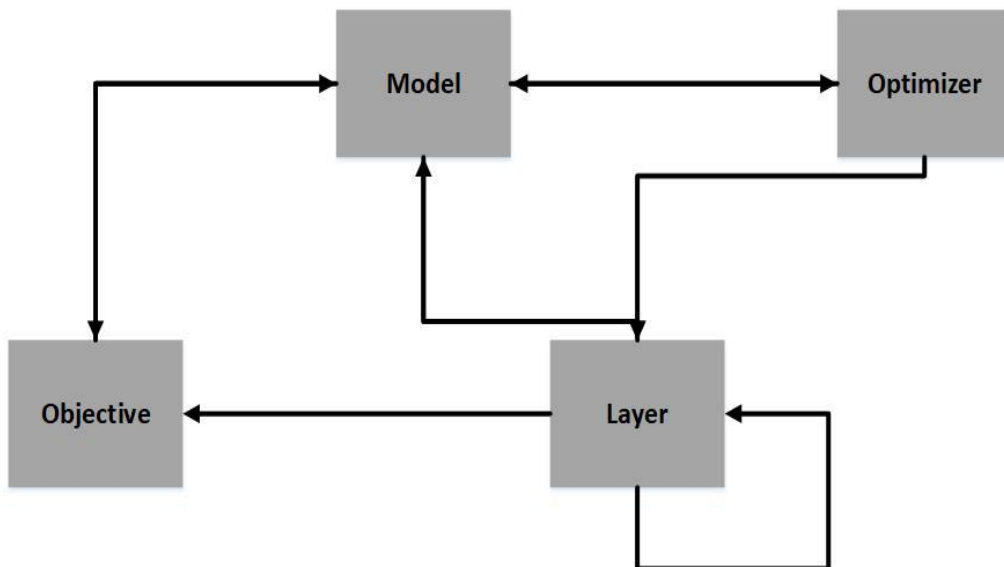


Figure 2.2 Schematic diagram of DNNs meta-model

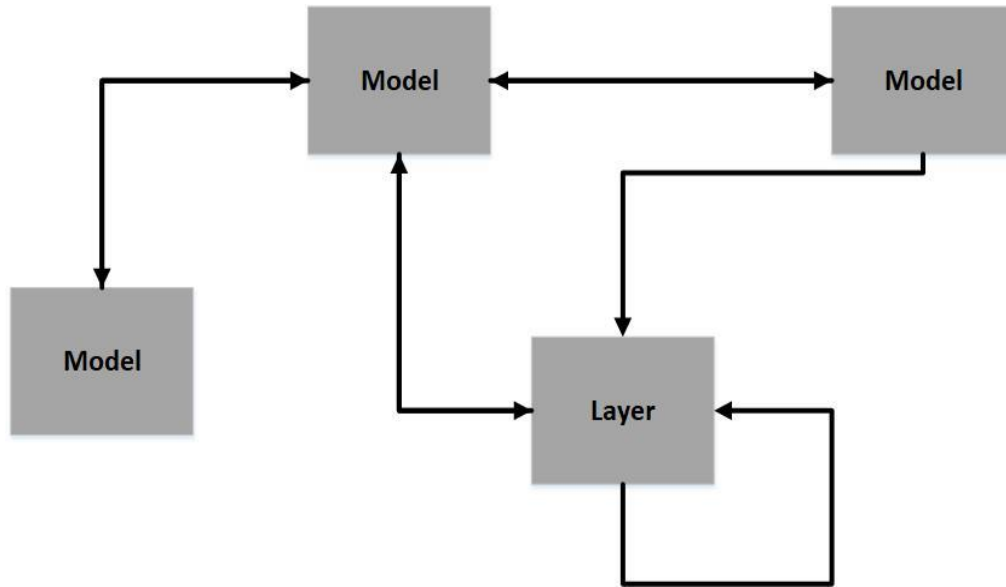


Figure 2.3 Schematic diagram of modern DNNs meta-model

Modern CNNs meta-architecture (Huang, Rathod, Sun, Zhu, Korattikara, Fathi, & Murphy, 2017) (Yadav & Binay, 2017) (Chin, Yu, Halpern, Genc, Tsao, & Reddi, 2018) (Oquab, Bottou, Laptev, & Sivic, 2014) of object detection is mainly represented by SSD (Liu, Anguelov, Erhan, Szegedy, Reed, Fu, & berg, 2016), Faster R-CNN (Ren, He, Girshick, & Sun, 2015), R-FCN (Dai, Li, He, & Sun, 2016), Multibox (Erhan, Szegedy, Toshev, & Anguelov, 2014), and YOLO (Redmon, Divvala, Girshick, & Farhadi, 2016).

In this project of detecting the effect of facial area ratio and angle on face recognition, the two most important requirements are real-time performance and average precision (Ren, Nguyen, & Yan, 2018). Unfortunately, only the papers of SSD (Liu, Anguelov, Erhan, Szegedy, Reed, Fu, & berg, 2016), R-FCN (Dai, Li, He, & Sun, 2016), and YOLO (Redmon, Divvala, Girshick, & Farhadi, 2016) discussed the running time on details. The work (Huang, Rathod, Sun, Zhu, Korattikara, Fathi, & Murphy, 2017) retest the frame-rate of SSD, R-FCN, Faster R-CNN, statistics a full picture of the speed and accuracy trade-off in a unified platform environment.

Around these modern meta-architecture, SSD and YOLO are using a single feedforward convolutional network to directly predict object (Kong, Sun, Yao, Liu, Lu, & Chen, 2017); in CVPR2017 (Huang, Rathod, Sun, Zhu, Korattikara, Fathi, & Murphy, 2017), SSD is defined as the classification of all CNNs meta-architecture that used single feedforward convolutional network to directly classify default boxes offset

without requiring the second stage per-proposal classification operation. Thereby, the meta-architecture of YOLO is also a kind of SSD. However, the dominant idea of object detection is to use the Faster R-CNN architecture, which requires the second stage per proposal classification operation (Huang, Rathod, Sun, Zhu, Korattikara, Fathi, & Murphy, 2017).

In the Faster R-CNN architecture, object detection requires two stages that region proposal network (RPN) and box proposal network (Ren, Zhu, & Xiao, 2018). The stage of RPN (Cai, Fan, Feris, & Vasconcelos, 2016) is used to predict class-agnostic box proposals by using feature extractors, such as Inception v2 (Tri, Duong, Van, Van, Nguyen, Toan, & Snasel, 2017). The stage of box proposal network is used to crop features from the same intermediate feature map, and then feed to the remainder of the feature extractor for predicting the class and class-specific box (Zagoruyko, Lerer, Lin, Pinheiro, Gross, Chintala, & Dallar, 2016). The running time of box proposal network depends on the number of regions proposed by the RPN (Huang, Rathod, Sun, Zhu, Korattikara, Fathi, & Murphy, 2017). Faster-RCNN has the good average precision (mAP) on object detection, but it cannot be achieved real-time detection due to requirement of two stages in each prediction (Wu, Iandola, Jin, & Keutzer, 2017). However, SSD is able to realize real-time object detection by applying the approach of single feedforward predict classes and default boxes offsets, thus, we are going to focus on applying the method of SSD meta-architectures.

In CNN meta-architectures, it is typical to have a collection of boxes overlaid on the image at different spatial locations, scales and aspect ratios, namely, default boxes (Liu, Anguelov, Erhan, Szegedy, Reed, Fu, & Berg, 2016). Some papers are named as priors (Erhan, Szegedy, Toshev, & Anguelov, 2014) (Bao & Chung, 2018) or anchors (Ren, He, Girshick, & Sun, 2016). The CNNs model on object detection requires to make two predictions for each default boxes, respectively, a discrete class prediction for each anchor, and a continuous prediction of an offset to shift the default boxes fit the ground truth bounding box (Erhan, Szegedy, Toshev, & Anguelov, 2014). There have a lot of default box in each image, if a default box matches the class offsets of ground truth bounding box, which becomes positive default boxes; otherwise, it calls negative default boxes (Liao, Shi, Bai, Wang, & Liu, 2017) (Shi, Bai, & Belongie, 2017). In SSD meta-architecture, each box is out of k at a given location, it computes c class scores and four offsets related to the original default box shape (Liu, Anguelov, Erhan, Szegedy,

Reed, Fu, & Berg, 2016). For each default box, it predicts both the shape offsets and confidences (Nguyen, YOsinski, & Clune, 2015) for different object categories.

2.2.1 Single Shot Detector (SSD)

SSD as recently developed the state-of-the-art convolutional network for real-time object detection (Liu, Anguelov, Erhan, Szegedy, Reed, Fu, & Berg, 2016) (Poirson, Ammirato, Fu, Liu, Kosecka, & Berg, 2016), it discretizes the output space of bounding box into a set of default boxes at different scales and aspect ratios in each feature map coordinate; object detection in a single shot without intermediate stage of detecting parts or initial bounding boxes eliminates proposal generation and subsequent pixel, or feature resample stage. In the time of object prediction, the network generates scores for each object category in each default box (Liao, Shi, Bai, Wang, & Liu, 2017); SSD network will also adjust each default box to better match the shape of the object. The fundamental of SSD is combination and improvement of YOLO (Redmon, Divvala, Girshick, & Farhadi, 2016) and R-FCN (Dai, Li, He, & Sun, 2016) architecture.

SSD is broadly defined as an architecture that uses a single feedforward convolutional network for object detection without requiring the second stage per-proposal classification operation (Huang, Rathod, Sun, Zhu, Korattikara, Fathi, & Murphy, 2017), which indicates that the SSD represents all single shot convolutional network, includes YOLO (Redmon, Divvala, Girshick, & Farhadi, 2016) and Overfeat (Sermanet, Eigen, Zhang, Mathieu, Fergus, & LeCun, 2014). However, we further study the single shot feedforward convolutional network (Yan, Chambers, 2013) and need to understand SSD (Liu, Anguelov, Erhan, Szegedy, Reed, Fu, & Berg, 2016) and YOLO (Redmon, Divvala, Girshick, & Farhadi, 2016) architecture respectively, then, build a suitable architecture for real-time face recognition.

For the SSD network architecture, the feature layer of size $m \times n$ with p channel, the kernel size is $3 \times 3 \times p$ for predicting parameters of a potential detection that a score for a category or a shape offset related to the default box coordinates (Chen, Papandreou, Kokkinos, Murphy, & YUille, 2016) (Hager, Dewan, & Stewart, 2004). The kernel is the core element that generates the default box (Nie, Zhang, Niu, Dou, & Xia, 2017).

The YOLO network architecture has a similar approach to SSD network architecture, and the main difference between these two is that SSD uses six fully convolutional layer for object detection, and YOLO uses two fully connected layers instead of the convolutional filter in output layers (Redmon, Divvala, Girshick, & Farhadi, 2016) (Liu, Anguelov, Erhan, Szegedy, Reed, Fu, & Berg, 2016). Therefore, the improvements of SSD from YOLO are mainly attributed to three aspects: (1) it uses a small convolutional filter to predict object categories and offsets for default boxes coordinates, (2) it utilizes separate filters for different aspect ratio detections, (3) it carries out detection at multiple scales by using these filters to multiple feature maps from the later stages of the network (Jeong, Park, & Kwak, 2017).

The traditional SSD (Liu, Anguelov, Erhan, Szegedy, Reed, Fu, & Berg, 2016) and YOLO (Redmon, Divvala, Girshick, & Farhadi, 2016) architectures enable real-time detection because they are using grid cells of feature maps proposal generator instead of region proposal network (Huang, Rathod, Sun, Zhu, Korattikara, Fathi, & Murphy, 2017).

Among them, the YOLO model uses a 7×7 grid cells feature map for proposal generating and the SSD model uses multiple feature maps from different feature layers, such as 8×8 grid cells of feature map and 4×4 grid cells of feature map (Redmon, Divvala, Girshick, & Farhadi, 2016) (Liu, Anguelov, Erhan, Szegedy, Reed, Fu, & Berg, 2016). In YOLO model, each grid cell predicts the object, and totally it has 98 proposal generators (Kong, Yao, Chen, & Sun, 2016). The grid cells of YOLO output results are confidence and position coordinate which is a great attempt and achieved good detection results (Redmon, Divvala, Girshick, & Farhadi, 2016). However, the accuracy of small object detection in YOLO is not satisfactory because feature maps is divided into fixed 7×7 grid cells (Al-masni, Al-antari, Park, Gi, Kim, Rivera, & Kim, 2018). The lower layer feature map can capture more details of the input object (Long, Shelhamer, & Darrell, 2015), thus utilizes the lower layer feature map with fewer grid cells to improve the accuracy of small object detection based on this conclusion (Wang, Ouyang, Wang, & Lu, 2015). Simultaneously, the upper feature map has an irreplaceable advantage in the detection of large objects (Cong & Xiao, 2014). Thereby, the SSD model applies both the lower and upper feature maps for object detection, such as 8×8 feature map and 4×4 feature map (Liu, Anguelov, Erhan, Szegedy, Reed, FU, & Berg, 2016).

Similar as YOLO, the output values of feature map of SSD are also location coordinate and confidence, which define the coordinates and categories of default boxes (Liu, Anguelov, Erhan, Szegedy, Reed, Fu, & Berg, 2016). The confidences output all object categories by c_1, c_2, \dots, c_p , where p represents category; the localization outputs the coordinates of default boxes by using the centre (c_x, c_y) , width (w) , and height (h) . Therefore, the loss function of SSD is a weighted sum of localization loss and the confidence loss:

$$L(x, c, l, g) = \frac{1}{N} (L \cdot \text{conf}(x, c) + a \cdot L \cdot \text{loc}(x, l, g)) \quad (2.1)$$

In the equation above, N is the number of match default boxes, the loss result is 0 when the $N = 0$; l refers to the predicted box; g represents the ground truth box; and a stands for the weight term. The localization loss of SSD is applying the Smooth L_1 loss function (Gkioxari, Girshick, & Malik, 2015), and the confidence loss is utilizing the Softmax loss function (Zhang, Zhang, Jing, Li, & Yang, 2017) (Zheng, Javsumana, Romera-Paredes, Vineet, Su, Du, & Torr, 2015) (Liu, Wen, Yu, & Yang, 2016). The Softmax loss function (Gu, Yang, Kong, Yan, & Klette, 2017) will also apply to CNNs model, and activation functions and loss function are also an important part of the CNNs model, we will talk about separately in below of this thesis.

2.2.2 Faster R-CNN

The Faster R-CNN (Ren, He, Girshick, & Sun, 2015) is the state-of-the-art convolutional network for object detection, which has the highest accuracy of object detection since the development from R-CNN and Fast R-CNN (Gkioxari, Girshick, & Malik, 2015) (Girshick, 2015). At the prediction phase, it has two stages of region proposal network (RPN) and box proposal (Gu, Yang, Yan, Li, & Klette, 2017).

The object detection of Faster R-CNN (Ren, He, Girshick, & Sun, 2015) can be divided into five main parts: 1) input the training set of object images into the Faster R-CNN model for feature extraction; 2) apply RPN generates proposal sliding windows, each image generates 300 sliding windows; 3) map sliding windows to the last layer of the Faster R-CNN convolutional feature map; 4) generate a fixed size feature map through the pooling layer; 5) utilize Softmax loss function and Smooth L_1 loss function

(Liu, Wen, Yu, & Yang, 2016) (Gkioxari, Girshick, & Malik, 2015) combines classifier and bounding box regression together.

Similar to the SSD model, the Faster R-CNN also is employed as an end-to-end network (Serban, Sordoni, Bengio, Gourville, & Pineau, 2016), but it applies fully-convolutional network (Long, Shelhamer, & Darrell, 2015) for the output layer. The difference is to train the fully-convolutional network by using back-propagation and stochastic gradient (LeCun, Boser, Denker, Henderson, Howard, Hubbard, & Jackel, 1989). The feature concatenation of Faster R-CNN (Cai, Fan, Feris, & Vasconcelos, 2016) is the approach that improves the RoI pooling (Girshick, 2015) by combining the feature maps of multiple convolution layers for lower and upper layer features. The Faster R-CNN network architecture is shown in Figure 2.4.

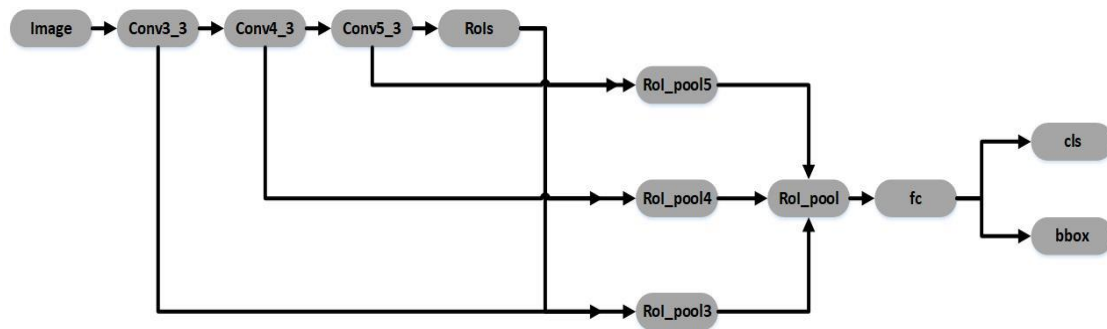


Figure 2.4 Faster R-CNN architecture of the feature concatenation scheme

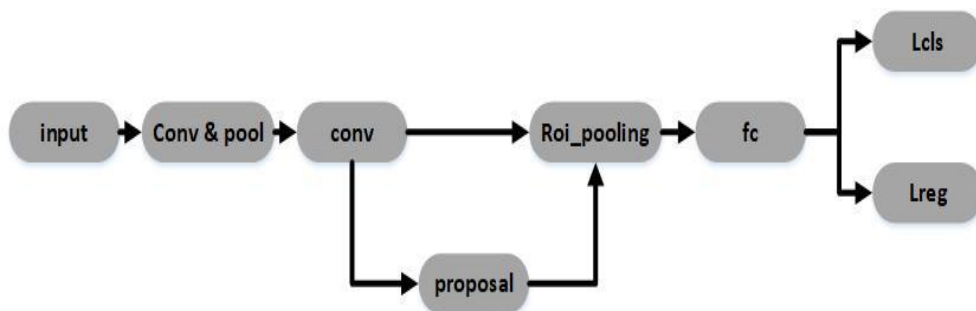


Figure 2.5 Flowchart of object detection of Faster R-CNN

The feature concatenation training of Faster R-CNN (Ren, Zhu, & Xiao, 2018) is shown in Figure 2.4, which has four stages: 1) training the RPN separately, the network parameter load from the per-training model, 2) training the Fast R-CNN network separately, the training parameter loaded from the output of RPN, 3) training the RPN again, the parameters are shared between the Fast R-CNN and the RPN, 4) the parameters of the second RPN output to ROI pooling layer (Ren, He, Girshick, & Sun,

2015). It is noteworthy that the hardware requirements of training environment of Faster R-CNN also have very high threshold, such as it requires at least the level of NVIDIA Titan GPU for training (He, Gkioxari, Dollar, & Girshick, 2017). Faster R-CNN is a sophisticated and rigorous network model that achieves the highest accuracy of object detection by using the design of architecture, though it is not suitable for real-time applications (Akselrod-Ballin, Karlinsky, Alpert, Hasoul, Ben-Ari, & Barkan, 2016).

2.3 Feature Extractor of CNNs Model

Meta-architecture of deep learning can be regarded as the concrete application model of deep learning, its essence is based on the development of deep learning feature extraction model (Huang, Rathod, Sun, Zhu, Korattikara, Fathi, & Murphy, 2017). Convolutional feature extraction network model develops from the initial LeNet5 model (LeCun, Bottou, Bengio, & Haffner, 1998) to AlexNet model (Krizhevshy, Sutskever, & Hinton, 2012), Network-in-Network model (Lin, Lin, Zhou, & Tang, 2014), VGG model (Lin & Yuan, 2016), GoogLeNet model (Szegedy, Liu, Jia, Sermanet, Reed, Anguelov, & Rabinovich, 2015), and ResNet model (He, Zhang, Ren, & Sun, 2016). We conclude that the depth and complexity of convolutional networks are increasing in order to improve the accuracy of object detection, but the training time also increases with the increasing model complexity.

Nonetheless, the development of convolutional feature extraction model has entered a bottleneck, the VGG-19 (Lin & Yuan, 2016) did not achieve good detection accuracy even if it has most operations. Convolutional feature extraction of Inception, ResNet and VGG model is the state-of-the-art technology according to the top one accuracy of object detection (Rajalingham, Issa, Bashivan, Kar, Schmidt, & DicCarlo, 2018). However, all feature extraction models are widely used in various meta-architectures which represent the basic model of convolutional models.

Among them, the meta-architecture of SSD (Liu, Anguelov, Erhan, Szegedy, Reed, Fu, & Berg, 2016) and Faster R-CNN (Ren, He, Girshick, & Sun, 2015) uses the VGG-16 network as a base, the meta-architecture of YOLO (Redmon, Divvala, Girshick, & Farhadi, 2016) uses the GoogLeNet model as a base, the meta-architecture of MultiBox (Erhan, Szegedy, Toshev, & Anguelov, 2014) applies the Inception model as a base, the meta-architecture of R-FCN (Dai, Li, He, & Sun, 2016) utilizes the ResNet-101 as

a base. The convolutional feature extraction network, as the theoretical basis of meta-architecture, needs to profound understanding before we construct an approach of object detection model.

2.3.1 LeNet-5

LeNet-5 is a ground-breaking convolutional neural network architecture that was released in 1998 (LeCun, Bottou, Bengio, & Haffner, 1998). It greatly promoted the development of convolutional deep neural networks. Most convolutional deep learning models are based on the core ideas of LeNet-5 in recent years (Tang, Lu, Wang, Huang, & Li, 2015). In 1998, there was no approach of GPU acceleration training, the saving of parameters and computation is an important feature of LeNet-5. Therefore, LeNet-5 proposed three main features of CNNs: locally connected neural network, weight shared, and subsampling (LeCun, Bottou, Bengio, & Haffner, 1998). The features of the architecture of LeNet-5 model can be summarized as five parts: 1) LeNet-5 convolutional network using three layers that convolution layer, pooling layer, and non-linearity function; 2) the subsample of LeNet-5 using spatial average of maps; 3) the non-linearity activation function applies Tanh or Sigmoids function; 4) the final classifier applies the multi-layer neural network(MLP); 5) it applies sparse connection matrix between each layer in order to avoid large computational cost (LeCun, Jackel, Bottou, Cortes, Denker, & Vapnik, 1995). LeNet-5 is an epoch-making convolutional feature extraction network, which key features have been used in the state-of-the-art convolutional networks.

2.3.2 Dan Ciresan Net

Dan Ciresan Net (Ciresan, Meier, Gambardella, & Schmidhuber, 2010) is the first training of deep neural network under GPU environment, which was proposed in 2010. This network model implemented both forward and backward training on a NVIDIA GTX 280 graphic processor, which can be trained up to nine layers of this neural network. In the test results, the training speed of Dan Ciresan Net neural network increased more than 10 times faster. This is a great attempt to greatly enhance the training velocity of deep neural network under the GPU environment; it opens a new chapter for the development of deep neural networks. After that, most of the deep neural

network models using the GPU environment have been developed to train their datasets (Chen & Lin, 2014).

2.3.3 AlexNet

In the introduction of DNNs above, we briefed that AlexNet win the object detection on ILSVRC and the deep neural network entered the public's awareness, which promoted the third wave of artificial intelligence research (Lee, Chen, Yu, & Lai, 2018). AlexNet is created based on the LeNet-5 model which improves the 5-convolution layer to 7-convolution layer, and it can extract more complex high-dimensional image features from the image (Krizhevsky, Sutskever, & Hinton, 2012). Overall, the architecture of AlexNet model and its features can be summarized as four parts: 1) compared with the sigmoid function of LeNet-5, AlexNet utilizes ReLU functions as activation function to reduces the amount of computation; 2) it applies Dropout technique that temporarily deletes partial neurons randomly, which reduces overfitting effectively; 3) it utilizes max-pooling technique to reduce the computations of convolutional layer; 4) it uses double GPU NVIDIA GTX 580 graphic processor, further improves the training velocity of CNNs (Krizhevsky, Sutskever, & Hinton, 2012). AlexNet is the first deep neural network based on the ILSVRC model whose object detection performance surpasses the traditional algorithms in whole aspects. Its appearance represents the object detection field where deep neural network begins to dominate (Khan & Yong, 2017). Simultaneously, artificial intelligence started a new development.

2.3.4 VGGNet

VGGNet (Lin & Yuan, 2016) was a new model of deep neural networks released by the University of Oxford in 2014, which uses a small 3×3 convolution kernel in per convolution layer to extract minutiae and arrange these small convolutional kernels as a sequence of convolutions. It seems to be contrary to the principles of LeNet-5 and AlexNet, where VGGNet proposes to use a 3×3 small convolutional kernel while AlexNet use 7×7 , 9×9 and 11×11 large convolutional kernel. The fundamental of VGGNet is to convolve the original image using 3×3 small convolution kernels, and then apply 3×3 convolution kernels continuously for multiple convolutions (Chu,

Ouyang, Li, & Wang, 2016). The approach of multiple 3×3 small convolution kernels can simulate the effect of large convolution kernel for local feature extraction, thus greatly reduce computation parameters and the time of object detection. Because only 9 weight parameters are used for 3×3 convolution kernel, the 7×7 convolution kernel uses 49 weight parameters (Wu, Leng, Wang, Hu, & Cheng, 2016).

VGGNet (Lin & Yuan, 2016) pointed out that the use of large convolution kernel will waste lots of time, small convolution kernel can reduce the computation parameters to save computational overhead. However, if we utilize small convolution kernels, it will also lead to increase the training time and require high performance of hardware but reduce prediction time and computation parameters of overall.

2.3.5 Inception

Inception model, also known as GoogLeNet, is a deep neural network structure released by Google in 2014 (Szegedy, Liu, Jia, Sermanet, Reed, Anguelov, & Rabinovich, 2015). The first version of Inception model shown in Figure 2.6 is a 1×1 , 3×3 , or 5×5 convolution kernel series parallelly combined deep neural network (Szegedy, Liu, Jia, Sermanet, Reed, Anguelov, & Rabinovich, 2015). Its main core is the 1×1 convolution kernel, which effectively reduces the number of features for parallel convolution kernels. The idea of 1×1 convolution kernel was inspired by the Network-in-Network model (Lin, Chen, & Yan, 2013), known as “bottleneck layer”. The bottleneck layer reduces the number of operational complexities of each convolution layer, thus shortens the time of computations (Szegedy, Vanhouche, Ioffe, Shlens, & Wojna, 2016).

Suppose ConvNet has 256 features coming in and 256 features coming out, then, it only uses 3×3 convolution kernel which has to perform $256\times 256\times 3\times 3$ convolutions; while using bottleneck layer can reduce computation parameters, it shrinks one of the 256 features to $256/4$ in 1×1 convolution kernel which performs $256\times 64\times 1\times 1$ convolutions, then uses 64 convolution on all Inception branches utilizing 3×3 convolution kernel, which needs $64\times 64\times 3\times 3$ convolutions, and then takes advantage of again a 1×1 convolution kernel for transfer 64 features to 256 which performs $64\times 256\times 1\times 1$ convolutions. Without the bottleneck layer only using 3×3 convolution kernel, it must perform $256\times 256\times 3\times 3$ convolutions; while with the bottleneck layer

parallel combined 3×3 convolution kernel, it must perform $256 \times 64 \times 1 \times 1 + 64 \times 64 \times 3 \times 3 + 64 \times 256 \times 1 \times 1$ convolutions which reduce almost 10 times.

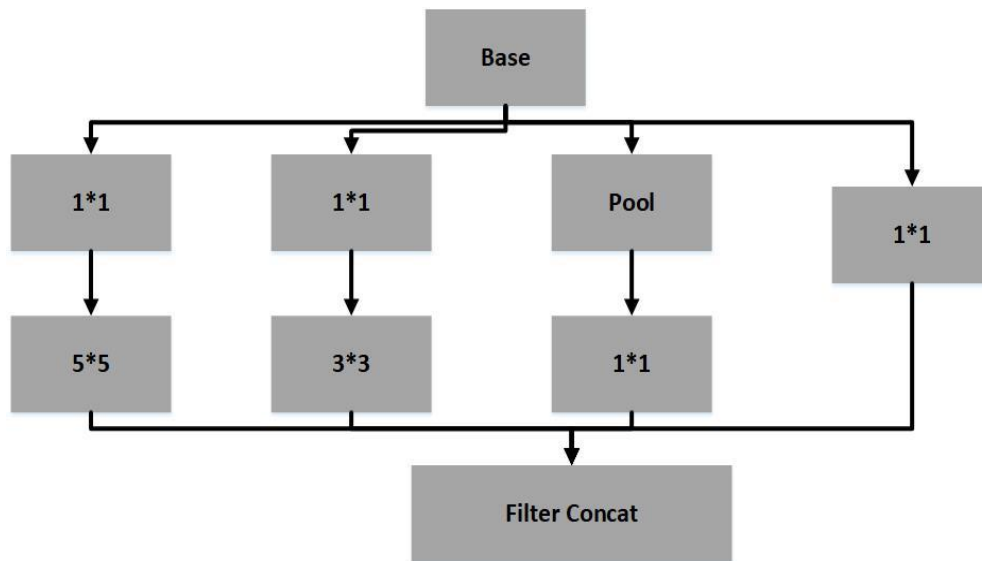


Figure 2.6 Classic Inception module

The bottleneck layer is the core idea of Inception, we define the parallel combined methods, the purpose is to reduce the operation parameters (Lin & Chen, 2015). Christian and his Google team introduced the concept of batch-normalized Inception (Szegedy, Ioffe, Vanhoucke, & Alemi, 2017) in 2015. Batch-normalization computes the mean and standard-deviation of all feature maps and output them to a layer. It corresponds to “whitening” (Hoffer Ailon, 2015) the data, it makes all the neural maps respond in the same range with zero mean. The batch-normalization helps training as the next layer which does not have to learn offsets in the input data, and then it can focus on combine features. The new version of Inception module is shown in Figure 2.7, its main improvement can divide into four parts: 1) it adds feature maps before the pooling layer for constructing networks that balance depth and width; 2) it increases the depth of neural networks and the number of features; 3) it enriches the combination of features before next layer by increasing width of the layers; 4) it employs only 3×3 and 1×1 convolution kernels by using VGGNet (Szegedy, Ioffe, Vanhoucke, & Alemi, 2017). The classifier of Inception is using pooling layer and Softmax layer, which is similar to VGGNet.

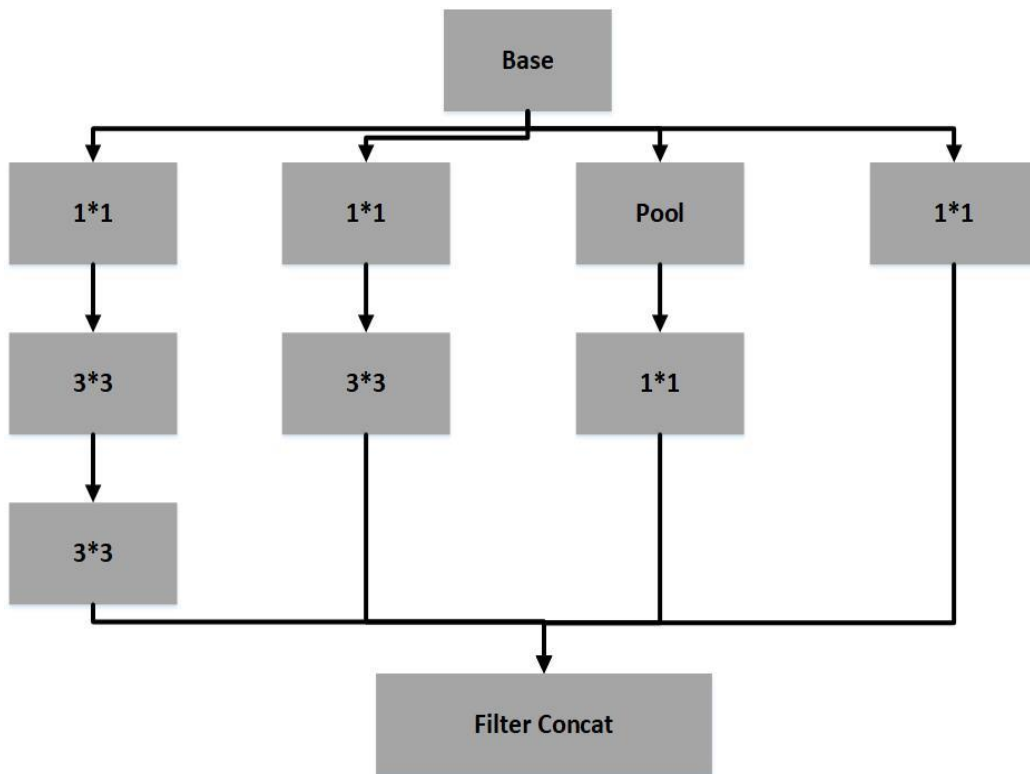


Figure 2.7 The new version of Inception module

2.3.6 ResNet

ResNet is a deep neural network model released in 2015 (He, Zhang, Ren, & Sun, 2016). The main feature of ResNet is the depth of network model which is able to achieve more than 1000 layers network model. The core idea of ResNet is to feed the output of two successive convolutional layer and bypass the input of the next layers (Toderici, Vincent, Johnston, Hwang, Minnen, Shor, & Covell, 2017). With the development of deep neural networks and graphic process performance, the depth of neural network model increases accordingly. The depth of neural model deepens and leads to difficulty increasing in optimization, which limits the accuracy of object detection and recognition. Thus, the ResNet is provided a new idea for solving complex deep neural network model (He, Zhang, Ren, & Sun, 2016). It is similar as Inception, ResNet also applies the bottleneck layer to reduce operation parameters and uses pooling layer plus Softmax as final classifier. ResNet has both parallel and serial modules which the income as parallel and the output of each modules connect in series (He, Zhang, Ren, & Sun, 2016). ResNet may be widely used with the development of neural network

models. However, ResNet has some undeniable disadvantages for our research, which requires high standard of operating environment and takes long time for training.

2.4 Overfitting Problems and Solutions

Overfitting corresponds too closely or exactly to a particular set of data and may fail to fit additional data or predict future observations reliably (Hitchcock & Sober, 2004). The concrete performance of overfitting is that the function of models perfectly matches with the data of training set, but the prediction results are much lower than the expected value. The primary reason caused overfitting is that the architecture of deep neural network models is too large and involves too many feature latitudes, this leads to a perfect fitting of the functions of model with training dataset while predictive accuracy cannot reach the expected since training dataset has their own features (Hinton, Vinals, & Dean, 2015).

Underfitting has similar problem to overfitting, while the reason for underfitting is that the model is too simple that leads to the model cannot fit the training dataset well, moreover, predictive accuracy cannot reach the expected value. With the development of deep neural networks, the feature extraction models are also much more perfect, thereby, the problem of under-fitting does not occur any more (Ghiassi, Saidance, & Zimbra, 2005). However, the feature extraction models are more and more complex, which lead to the overfitting becoming a factor that cannot be neglected.

How to prevent overfitting is being paid more and more attention. Currently, it mainly has six popular overfitting solutions (Clark, 2004), which are cross-validation, train with more data, features removal, early stopping, regularization, and ensembles. Among them, cross-validation is a powerful measure to prevent overfitting. In the deep neural network models, it optimizes training results through training the dataset repeatedly (Clark, 2004). Therefore, all the deep neural networks are using cross-validation method to reduce the overfitting; for the methods of training with more data, it can help our algorithms detect the signal better. In deep learning field, the method of training with more data is called Data Augmentation.

The main idea of Data augmentation is to generate more similar datasets through scaling, rotation and interception of the image, ultimately achieve the effect of

expanding the training dataset (Ding, Chen, Liu, & Huang, 2016). Data augmentation is a simple implementation and an excellent way to reduce overfitting. The method of removing features is also an effective approach to reduce overfitting. The specific implementation method is to simplify the feature extraction model according to training dataset. In the application of feature extraction model, many deep neural models are too complex for a training dataset, we can simplify the feature extraction model to reduce the overfitting. However, the underlying architecture of feature extraction models is complex; thus, this method is difficult to be implemented; the method of temporarily shield neurons is called dropout in deep neural networks (Schmidhuber, 2015). It reduces overfitting by randomly deleting some neurons temporarily, which has excellent achievement through the results of AlexNet model (Krizhevskv, Sutskever, & Hinton, 2012). This method has been widely used in deep learning, we will also introduce on details in this thesis.

The method of early stopping is to find the best training time, because sometimes the accuracy of training does not necessarily cause longer training time. With the deep neural models having gradually solved this problem, the long training time of the state-of-the-art model will not reduce training accuracy.

The method of regularization is reducing the overfitting by using the functions of models; the method of ensembles is one of machine learning algorithms for combining prediction from multiple separate models (Wan, Zeiler, Zhang, LeCun, & Fergus, 2013).

2.4.1 Data Augmentation

The number of images in dataset is an important factor that determines the accuracy of recognition in deep learning. The method of Data Augmentation is to enhance the training data by using artificially transformations, in order to achieve the purpose of reducing overfitting and improving the accuracy of recognition results (Ding, Chen, Liu, & Huang, 2016). The two distinct forms of Data Augmentation respectively are to generate image translations and horizontal reflections and alter the intensities of the RGB channels in training images. Both methods are able to convey multiple images from the original one with very little computation; the transformed images do not need to be stored on disk which can be generated before deep learning training (Krizhevsky, Sutskever, & Hinton, 2012).

For the form of generating image translations and horizontal reflections, if the training dataset is with images having the resolution 256×256 , we can randomly extract a fixed number of images (224×224) and horizontal reflection of these images, then, scale all images to a uniform size. In supervised training, all the images we cropped need to include the labelled region; hence, we can extract different labelling size and angles from one image.

For the form of altering the intensities of RGB channels in training images (Krizhevsky, Sutskever, & Hinton, 2012), we perform PCA on the set of RGB values and convert RGB to HSV colour space throughout the training dataset. This scheme approximately captures the principal components of natural images, the intensity and colour of object are invariable.

2.4.2 Dropout

The key idea of dropout is to take a large model that overfits easily and repeatedly sample and train smaller sub-models from it. Dropout was presented by Hinton in 2012. The Dropout is a neural network unit temporarily discarded from the deep learning model in accordance with a certain probability in the training process of deep learning. It should be noticed that the temporarily discarded neuron parameters are merely hidden in this training phase, the essence is to ignore the part of feature classifier, which means that the part of the hidden layer nodes tends to zero in each cycle of training. This approach can reduce the interaction in feature classifier, thus, effectively diminishes the overfitting phenomenon.

We also think that Dropout is an average model, each sample inputs into the neural network and its corresponding network structure is different, but all these different neural network structures share the weight of hidden layer nodes at the same cycle. This technology is used to improve the performance of deep learning in a variety of applications, such as image recognition, digital recognition, speech recognition, object classification and data analysis of computational biology (Gal & Ghahramani, 2016). Dropout is widely used in the field of DNNs, which directly shows the superiority of this technology in improving the accuracy of verification results. But the drawbacks of Dropout are also noteworthy, which greatly increases the time of data training and the complexity of the nonlinear activation function (Maalei, Tagougui, & Kherallah, 2016).

However, both Dropout and Data Augmentation are effective methods to reduce overfitting.

2.5 Activation and Loss Function

The main function of activation functions (Specht, 1990) is to provide the nonlinear modelling capabilities of a neural network model. If there is no activation function in neural network model, then, the model can only express linear mapping which achieves the effect of single layer network. Therefore, deep neural network only has a layered nonlinear learning ability by adding the activation function. The loss function (Murata, Yoshizawa, & Amari, 1994) is used to estimate the difference between the predicted value and the ground truth in a neural model, which is a non-negative numerical function. The loss function is the main approach to inspect the training results of deep neural network model, the smaller loss function of this model is with a better robustness (Quang, Chen, & Xie, 2014). The activation and loss functions are the core component of deep neural network model; thus, choice of the suitable function is crucial for constructing deep learning model.

In the current deep learning model, the LeNet-5 (LeCun, Bottou, Bengio, & Haffner, 1998) used the Sigmoid and Softmax activation function (Marreiros, Daunizeau, Kiebel, & Friston, 2008) with the loss function of MLE (Maximum Likelihood Estimation criterion) (Wood, 2011); the AlexNet (Krizhevsky, Sutskever, & Hinton, 2012) and VGGNet (Lin & Yuan, 2016) apply the ReLu nonlinearity activation function (Jin, Xu, Feng, Wei, Xiong, & Yan, 2016) with the cross entropy softmax loss function (DeBoer, Kroese, Mannor, & Rubinstein, 2005), it is noteworthy that VGGNet (Lin & Yuan, 2016) can be regarded as an upgraded version of AlexNet (Krizhevsky, Sutskever, & Hinton, 2012); the Inception (Szegedy, Liu, Jia, Sermanet, Reed, Anguelov, & Rabinovich, 2015) utilizes the Sigmoid activation function (Marreiros, Daunizeau, Kiebel, & Friston, 2008) with the cross entropy softmax loss function (DeBoer, Kroese, Mannor, & Rubinstein, 2005); the ResNet (He, Zhang, Ren, & Sun, 2016) applies ReLu activation function (Jin, Xu, Feng, Wei, Xiong, & Yan, 2016) with the loss function of cross entropy softmax.

The function of Sigmoid nonlinear activation function is $a(x) = 1/(1 + e^{-x})$. Deep learning model enters a parameter into the Sigmoid function, then compresses it into a

range of 0 to 1. In the Sigmoid function, the inactive parameter indicated as 0 and the fully saturated of model indicated as 1. The Sigmoid has two disadvantages: 1) Sigmoid saturation and kill gradients; 2) Sigmoid outputs are not zero-centered (Rawlings, Woodland, & Craford, 2006). In addition, the Tanh activation function is very similar to Sigmoid, but it compresses the input parameter signal into a range of -1 to 1 (Wan, Zeiler, Zhang, LeCun, & Fergus, 2013).

The function of ReLu nonlinear activation function is $f(x) = \max(0, x)$. Deep neural network model enters a parameter signal into the ReLu function; the output equals to the input if greater than 0; and the output is equal to 0 if less than 0. The advantages of the ReLu function are: 1) the convergence of this model is much faster than the Sigmoid function; 2) the function will not be saturated; 3) the operation is less than the Sigmoid function (Zhang & Woodland, 2016). But its drawback is also obvious that ReLu is fragile in training time which easily leads to neuron necrosis.

The loss function of cross entropy is indicated as $C = -\frac{1}{n} \sum_x (y \ln a + (1 - y) \ln(1 - a))$, where C is the loss value, x stands for the input sample, y represents the predicted value, a means the actual output signal of neuron which $a = \sigma(z)$, n refers to the total number of samples. The cross-entropy function (Maas, Hannun, & Ng, 2013) applies classification approach to the computation of loss values, if the output of Softmax function (Kivinen & Warmuth, 1998) on the verge of 1, then it is corresponding to correct class label 1 which indicates $y = 1$; otherwise, the output value of Softmax function on the verge of 0, then it is corresponding to incorrect class label 0 which indicates $y = 0$. If the difference between the output signal parameters of the Softmax function classifier, the loss function value is big. In addition, loss function based on cross entropy is able to offset the shortcoming of saturation, thus, Sigmoid activation function usually matches with the cross-entropy loss function (Dunne & Campbell, 1997).

2.6 Multilayer Perceptron

Multilayer Perceptron (MLP) (Taud & Mas, 2018) can be interpreted as an artificial neural network, which contains input layer, hidden layer, and output layer. The layers are connected by using fully-connected layer in MLP, the simplest MLP can have only

one hidden layer. In CNNs model, an MLP consists of multiple fully-connected layers with nonlinear activation functions (Lin, Chen, & Yan, 2013). It is noteworthy that images input to the MLP layer in the CNNs model are an abstract one from convolution layers which describes that this feature is invariant to variations of the same concept after the feature extraction from convolution layers (Bengio, Courville, & Vincent, 2013).

In MLP, the function is described (Perez, 2017), if the size of input vector is x , the size of the output vector is $f(x)$, bias vectors are $b^{(1)}$ and $b^{(2)}$, weight matrices are $W^{(1)}$ and $W^{(2)}$, activation functions are G and s , then the matrix notation of MLP is

$$f(x) = G(b^{(2)} + W^{(2)}(s(b^{(1)} + W^{(1)}x))). \quad (2.2)$$

The output vector of MLP is obtained as

$$o(x) = G(b^{(2)} + W^{(2)}h(x)) \quad (2.3)$$

In order to train the MLP layer, the DNN model needs to learn all the parameters, and the set of parameters is $\{W^{(2)}, b^{(2)}, W^{(1)}, b^{(1)}\}$. In addition, the output of MLP layer also is the CNNs model results.

Chapter 3

Methodology

The purpose of this chapter is to introduce the methodology we used for our experiments that contributed to the final outcomes. This includes data sources, data augmentation methods, and model design. We will clearly explain the details and process of the method.

3.1 Research Designing

Since the purpose of this thesis is to achieve face recognition, we listed the steps of face recognition as shown in Figure 3.1. In this figure, we listed seven steps to show the structure of this research.

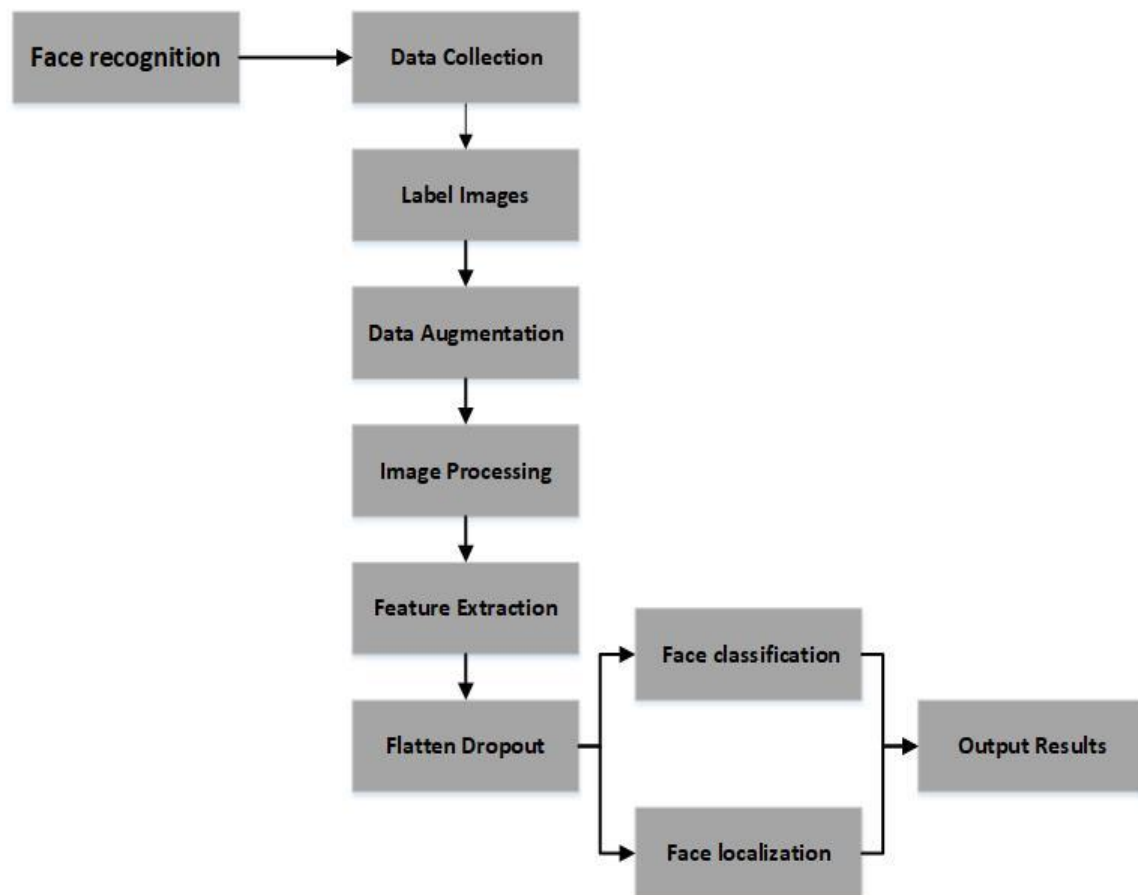


Figure 3.1 The steps to achieve face recognition

In the flowchart of face recognition as shown in Figure 3.1, the first three steps are used to prepare the data for training, the last four steps mainly refer to the workflow inside the model. The combination of the two parts can achieve face recognition.

In our experiment, we chose to collect data sets through ourselves. Next, we will mark the face location in the dataset and then put our original dataset into the data augmentation program to increase the number of datasets. After the external data is processed, the internal model of image processing will tackle the input images again. The model uses the convolutional layer to extract features of the face, the feature extraction results is input to the fully connected layer through the Dropout process to perform face classification and localization.

3.1.1 Data Sources and Data Collection

The core issue of this thesis is the use of deep learning to detect and recognize faces. The primary problem of deep learning is how to train data, so we need to prepare training data and mark the location and classification of faces in each image. Because it is difficult to find public datasets that meet our requirements, so we choose to collect data by ourselves.

Since we mainly investigate the influence of face proportion on confidence and accuracy, a proportion of face images is the data we need to collect. That is, we need to collect faces of different sizes as datasets. Of course, the data collected using different devices will have different resolutions, and if the pixels of the collected face images are clearer, the recognition precision will also be increased. Therefore, the performance of the identification is closely related to the quality of the dataset. It is worth mentioning that this thesis uses the rear camera of Apple iPhone 7 plus which has 12 million pixels as a device for collecting data. We can't deny that if we use better pixels and resolution devices to collect datasets, we will get better recognition results.

There are five categories in the data set we collected. Table 3.1 indicates that we have five participants in the experiment. Because in this thesis we collected the data by ourselves, there are five classes we collected, each class represents one category.

Table 3.1 Classes in our experiment

Classes	Names
1	Person No.1
2	Person No.2
3	Person No.3
4	Person No.4
5	Person No.5

The data we collected is shown in Figure 3.2. We collected face data by taking a video of each person and store each video in a folder with their own names. In order to get a better training effect, we should pay attention to the following points when collecting datasets:

- The picture is clear, and the movement is small, avoiding the dynamic blurring of the image.
- Various positions of a person's face in an image, people can walk a few steps.
- Faces are big enough in size and need to have images taken at different distances.
- Faces are big enough in forms, which means, taking under different expressions and actions.

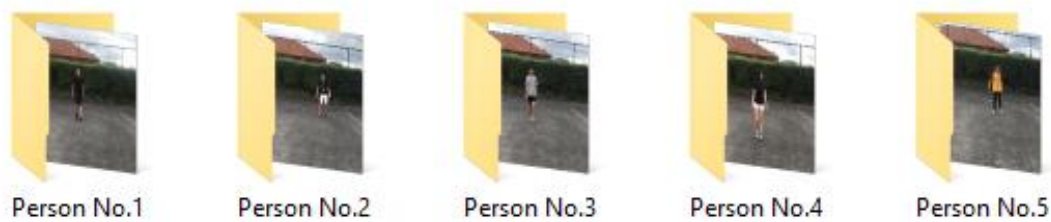


Figure 3.2 Data collection by taking a video

After the video was collected, we used a tool to acquire frames in the video. Each video took an average of 120 images. Because we used five people in our dataset, we have a total of 600 images. The 600 images will be automatically saved in the folder. In order to ensure quality of the dataset, we select 100 individuals from 120 images, among which 100 images are selected for each person for 500 images in total.

3.1.2 Face Labelling

After collecting the original data, we need to label face position on the collected images. Face detection is a branch task of object detection, a rectangular bounding box is generally used to mark the position of a human face in the traditional method. However, the human face can be approximated as an ellipse. Using the ellipse mark not only can accurately confirm the position of the human face, but also can recognize that the human face rotates in the image. The ellipse labelling method of bounding box has been applied to traffic-sign recognition and the medical application of vessel detection, which has achieved high precision and practical application effect.

For the traffic-sign detection, previous research adopts the ellipse labels bounding box with circle signs, and label arbitrary “vertices” along the bounding box of the ellipse label (Zhu, Liang, Zhang, Huang, Li, & Hu, 2016). For the medical application, Smistad and Løvstakken proposed utilize the ellipse labels method for vessel detection

in ultrasound images (Smistad & Løvstakken, 2016). The method of ellipse label is also suitable for the face recognition filed based on the shape of the human face, thus we adopted the bounding box of ellipse label.

We use a tool to mark ellipse shapes based on human faces. Our labelling procedure is shown in Figure 3.3. When we label face, we need to specify the positions of three points, which are the two endpoints of the major axis of the ellipse and one endpoint of the minor axis. We see from Figure 3.3, using a rectangular mark will not reflect the tilt angle of a human face.



Figure 3.3 Labelling the face position

After we label the face location, these labelled files are automatically deposited to a folder and archived in text format. As shown in Figure 3.4, these are our marked files.

Person No.1_0117.txt	12/03/2018 2:12 PM	Text Document	1 KB
Person No.1_0118.txt	12/03/2018 2:12 PM	Text Document	1 KB
Person No.1_0119.txt	12/03/2018 2:12 PM	Text Document	1 KB
Person No.1_0120.txt	12/03/2018 2:12 PM	Text Document	1 KB
Person No.2_0001.txt	12/03/2018 2:13 PM	Text Document	1 KB
Person No.2_0002.txt	12/03/2018 2:13 PM	Text Document	1 KB
Person No.2_0003.txt	12/03/2018 2:13 PM	Text Document	1 KB
Person No.2_0004.txt	12/03/2018 2:13 PM	Text Document	1 KB
Person No.2_0005.txt	12/03/2018 2:13 PM	Text Document	1 KB
Person No.2_0006.txt	12/03/2018 2:13 PM	Text Document	1 KB
Person No.2_0007.txt	12/03/2018 2:13 PM	Text Document	1 KB
Person No.2_0008.txt	12/03/2018 2:13 PM	Text Document	1 KB
Person No.2_0009.txt	12/03/2018 2:14 PM	Text Document	1 KB
Person No.2_0010.txt	12/03/2018 2:14 PM	Text Document	1 KB
Person No.2_0011.txt	12/03/2018 2:14 PM	Text Document	1 KB
Person No.2_0012.txt	12/03/2018 2:14 PM	Text Document	1 KB
Person No.2_0013.txt	12/03/2018 2:14 PM	Text Document	1 KB
Person No.2_0014.txt	12/03/2018 2:14 PM	Text Document	1 KB
Person No.2_0015.txt	12/03/2018 2:15 PM	Text Document	1 KB
Person No.2_0016.txt	12/03/2018 2:15 PM	Text Document	1 KB
Person No.2_0017.txt	12/03/2018 2:15 PM	Text Document	1 KB
Person No.2_0018.txt	12/03/2018 2:15 PM	Text Document	1 KB

Figure 3.4 Marked files saved in text format

Figure 3.5 shows the contents of the labelled file, which contains the category name of objects, and the coordinates of the labelled face. Among them, x_c , y_c represent the centre point of the ellipse, a is the semi-major axis of the ellipse, b is the semi-minor axis of the ellipse, and t refers to the rotation angle.

```

Person No.1.txt x
1 type:Person No.1
2 xc:597.707547
3 yc:141.783019
4 a:27.981278
5 b:23.314984
6 t:1.473543

```

Figure 3.5 Contents of the labelled files

However, elliptical markers will bring new difficulties in data augmentation. We will address and solve these problems below in this thesis.

3.1.3 Data Augmentation

After marking the data set, we got 500 original images which have been labelled. However, this number is not enough for training, if the number of our data set is very small, the model training will be inadequate, which will affect the final accuracy of the model. In order to make the training dataset fully trained, we tried a method to improve the data and increase the amount of data.

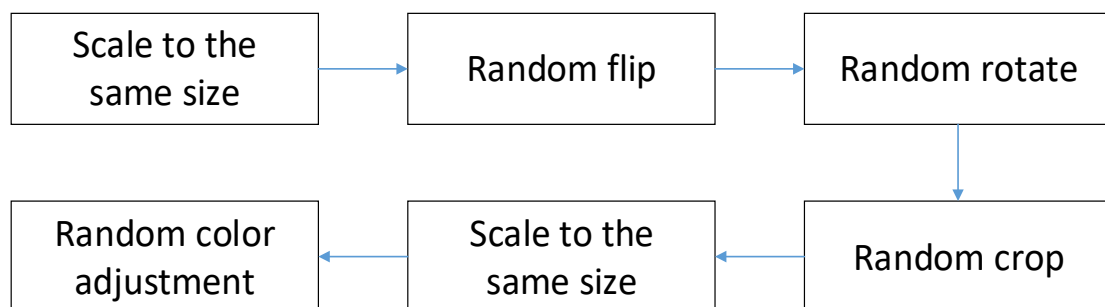


Figure 3.6 Flowchart of data augmentation

Figure 3.6 above shows the flowchart of data augmentation. Then, we will introduce the details of each step one by one.

- For the collected data, there will inevitably be different sizes. The format of the video input is generally different, such as Full HD (1920×1080), HD (1280×720), qHD (960×540), and nHD (640×360). The first thing we need to commitment is to cut the collected data into uniform sizes. Due to computer hardware limitations, we chose nHD (640×360) as the input size of the DNNs.
- Due to our camera that might be reversal during the time of data collection, the image will also appear inconsistent. In order to ensure the recognition accuracy, we randomly flipped the images involved in the training to improve the accuracy of verification and enabled it to recognize a variety of situations.
- We use the ellipse shape to label people’s faces so that the faces can be recognized when the face is tilted. Since the images are randomly selected frames from the video, the non-tilted human face occupies the main part of that dataset. For example, if there is an image from one of the 100 images, the face is labelled. Even if the model does not consider the labelled faces, the precision can reach 99%. This will not predict the tilted faces very well. So, we need to increase the number of tilted faces.

- Next, we will perform randomly cropping the image. The purpose is to let a face occupy various positions in the image to enhance the size and position of the face. After cropped, the position of the face and its ratio to the picture have been changed.
- After cropping, we need to scale the cropped image down to our desired size so that it can be used as training data.
- Considering that different cameras will have different colour when shooting, in order to simulate the colour difference, we convert the RGB colour to the HSV colour space and randomly adjust the saturation (S) and value (V) component.

In the steps listed, we take the colour conversion into consideration, but in image processing, we also need to alter the position of the bounding box because we are using the elliptical bounding box. The focus of this problem is on how to calculate the corresponding parameters of the ellipse after the transformation. Then, we will introduce the calculation of this ellipse transformation.

Equation (3.1) is used for the ellipse of the major axis on the x -axis

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \quad (a \geq b), \quad (3.1)$$

where a is the semi-major axis of the ellipse and b is the semi-minor axis of the ellipse. When $a=b$, the ellipse becomes a circle. This elliptical parametric equation can be written as

$$\begin{cases} x = a \cos t \\ y = b \sin t \end{cases} \quad t \in [0, 2\pi]. \quad (3.2)$$

Now we rotate the ellipse shown in equation (3.2), set the angle between the long axis of the ellipse and the x -axis as ϕ ; then, the parameter equation can be written as

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos \phi & -\sin \phi \\ \sin \phi & \cos \phi \end{pmatrix} \begin{pmatrix} a \cos t \\ b \sin t \end{pmatrix}. \quad (3.3)$$

Then, we shift centre of the ellipse to (x_c, y_c) , and finally get the parameter equation for the ellipse

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos \phi & -\sin \phi \\ \sin \phi & \cos \phi \end{pmatrix} \begin{pmatrix} a \cos t \\ b \sin t \end{pmatrix} + \begin{pmatrix} x_c \\ y_c \end{pmatrix}. \quad (3.4)$$

Therefore, any ellipse can be uniquely determined by using these five parameters $E = [x_c, y_c, a, b, \phi]$. The difficulty of the problem lies in what values should be taken for the corresponding five parameters after the original image is scaled, rotated, and cropped.

3.1.3.1 Scaling Conversion

First, we perform the scaling conversion $S(s_x, s_y)$, if the transformation equation is

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}, \quad (3.5)$$

where (x, y) is the original coordinates and (x', y') is the transformed. Then the parametric equation of the scaled ellipse is

$$\begin{aligned} \begin{pmatrix} x' \\ y' \end{pmatrix} &= \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \left(\begin{pmatrix} \cos \phi & -\sin \phi \\ \sin \phi & \cos \phi \end{pmatrix} \begin{pmatrix} a \cos t \\ b \sin t \end{pmatrix} + \begin{pmatrix} x_c \\ y_c \end{pmatrix} \right) = \\ & \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \begin{pmatrix} \cos \phi & -\sin \phi \\ \sin \phi & \cos \phi \end{pmatrix} \begin{pmatrix} a \cos t \\ b \sin t \end{pmatrix} + \begin{pmatrix} s_x x_c \\ s_y y_c \end{pmatrix}. \end{aligned} \quad (3.6)$$

As we see after the conversion

$$\begin{aligned} x'_c &= s_x x_c \\ y'_c &= s_y y_c \end{aligned} \quad (3.7)$$

However, the new a', b', ϕ' cannot be directly determined, which means that we cannot find a group of a', b', ϕ', t' directly so that equation (3.8) establishes:

$$\begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \begin{pmatrix} \cos \phi & -\sin \phi \\ \sin \phi & \cos \phi \end{pmatrix} \begin{pmatrix} a \cos t \\ b \sin t \end{pmatrix} = \begin{pmatrix} \cos \phi' & -\sin \phi' \\ \sin \phi' & \cos \phi' \end{pmatrix} \begin{pmatrix} a' \cos(t+t') \\ b' \sin(t+t') \end{pmatrix} \quad (3.8)$$

However, we calculate the corresponding a', b', ϕ' from the definition of long and short axes. Let

$$\begin{aligned} u &= s_x(a \cos \phi \cos t - b \sin \phi \sin t) \\ v &= s_y(a \sin \phi \cos t + b \cos \phi \sin t), \\ f(t) &= u^2 + v^2 \end{aligned} \quad (3.9)$$

Calculate

$$\begin{aligned} f(t_a) &= \max f(t) \\ f(t_b) &= \min f(t) \end{aligned} \quad (3.10)$$

Then, we get

$$a' = \sqrt{f(t_a)}, b' = \sqrt{f(t_b)}, \phi' = t_a. \quad (3.11)$$

Although theoretical calculations of t_a, t_b are more complex, numerical calculations are very simple. We use the numerical method to calculate t_a and t_b .

In summary, scaling $S(s_x, s_y)$ for the ellipse $E = [x_c, y_c, a, b, \phi]$ can get a new ellipse

$$E' = [s_x x_c, s_y y_c, \sqrt{f(t_a)}, \sqrt{f(t_b)}, t_a]. \quad (3.12)$$

3.1.3.2 Rotation Conversion

After scaling conversion, we consider the rotation conversion $R(\theta, w, h)$. Because the rotation of this image is on the centre of the image to rotated, our ellipse is the origin of the upper-left corner of the image, the x axis is the width direction, and the y axis is the height direction. Therefore, the result of the rotation is not only related to rotation angle θ , it is but also related to the size of the image $[h, w]$.

Rotating the centre of image $C = (w/2, h/2)$ by θ degree, as a transformation of the ellipse from the xOy coordinate system to the $x'Cy'$ coordinate system, we rotate the new origin by θ degree, and converse back to the xOy coordinate. Therefore, the transformed ellipse is shown in equation (3.13)

$$\begin{aligned} \begin{pmatrix} x' \\ y' \end{pmatrix} &= \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \left(\begin{pmatrix} \cos \phi & -\sin \phi \\ \sin \phi & \cos \phi \end{pmatrix} \begin{pmatrix} a \cos t \\ b \sin t \end{pmatrix} + \begin{pmatrix} x_c - w/2 \\ y_c - h/2 \end{pmatrix} \right) + \begin{pmatrix} w/2 \\ h/2 \end{pmatrix} = \\ & \begin{pmatrix} \cos(\phi+\theta) & -\sin(\phi+\theta) \\ \sin(\phi+\theta) & \cos(\phi+\theta) \end{pmatrix} \begin{pmatrix} a \cos t \\ b \sin t \end{pmatrix} + \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} x_c - w/2 \\ y_c - h/2 \end{pmatrix} + \begin{pmatrix} w/2 \\ h/2 \end{pmatrix}. \end{aligned} \quad (3.13)$$

Then, we see

$$\begin{aligned} a' &= a, b' = b, \phi' = \phi + \theta \\ \begin{pmatrix} x_{c'} \\ y_{c'} \end{pmatrix} &= \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} x_c - w/2 \\ y_c - h/2 \end{pmatrix} + \begin{pmatrix} w/2 \\ h/2 \end{pmatrix}. \end{aligned} \quad (3.14)$$

3.1.3.3 Cropping and Flipping

Compared to the scaling and rotation transformations, the calculation of ellipse parameters under cropping and flipping changes is relatively simple.

Suppose the cropped image centred on (x_0, y_0) and its size is $[h', w']$, then the transformed ellipse is

$$E' = [x_c - x_0, y_c - y_0, a, b, \phi]. \quad (3.15)$$

The left and right flips are about $x = w/2$ for axisymmetric transformation, so the transformed ellipse is

$$E' = [w - x_c, y_c, a, b, \pi - \phi]. \quad (3.16)$$

3.1.3.4 Summary

After solving the problems above, we achieve data augmentation. Because our original training dataset has 500 images, there are 100 images for each of the five classes. After we have augmented the data, it will generate 50 different samples for the same original samples. Among them, 40 augmentation samples are generated for training, and 10 augmentation samples are produced for verification. So, in our training dataset, our training set contains 20,000 samples for training and 5000 samples for verification. This increases the number of training sets, includes various aspects of the face in the image; it can fully identify faces in various situations. The effect of the data augmentation is shown in Figure 3.7. We get multiple samples with different angles, different sizes, and different positions using one of our samples.

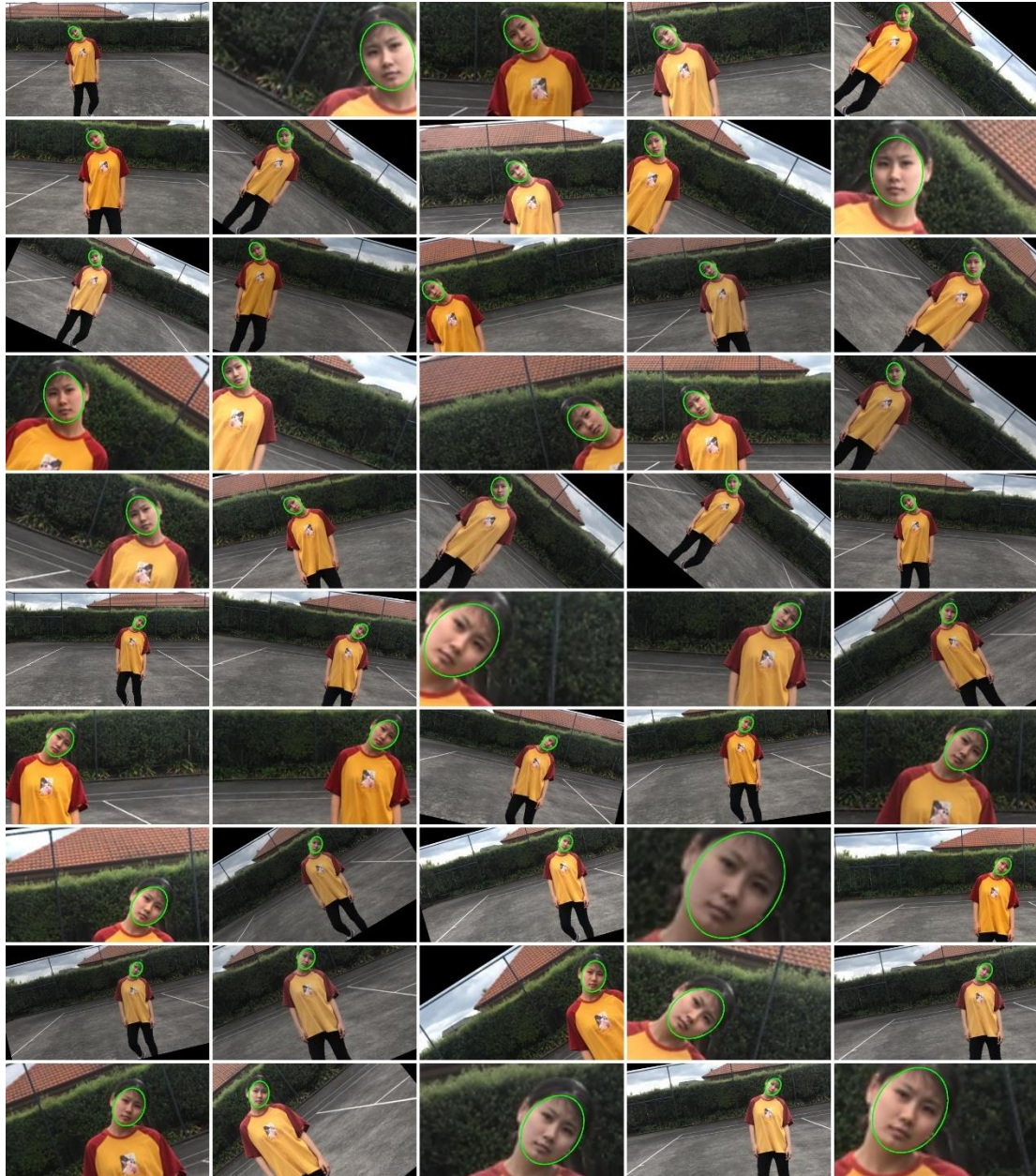


Figure 3.7 Schematic diagram of data augmentation

3.2 The Principle of CNNs

The principle of CNNs in our model is shown in Figure 3.8, which can be divided into convolution layer, pooling layer, and fully connected MLP. The convolution layer convolves the image into different feature maps according to the number of convolution kernels to achieve the purpose of feature extraction. We used 5×5 and 3×3 convolution kernels in the convolution layers, and each convolution layer follows a pooling layer in common CNNs model to reduce neuron parameters. The fully connected layer is the output layer of the CNNs, which adopts the cross entropy as the neurons output function

of classification and utilizes the smooth L1 as the neurons output function of localization. In addition, we deploy the Dropout operation in the fully-connected layer to reduce the overfitting of model training.

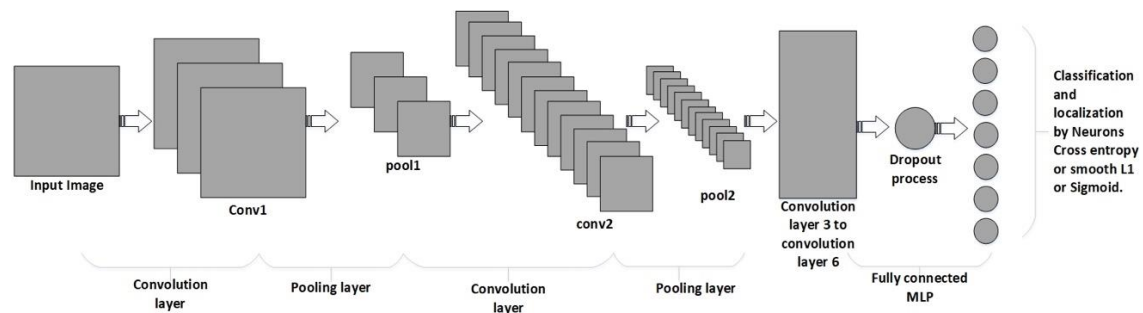


Figure 3.8 The working principle of CNNs

The principle of the feature extraction process is shown in Figure 3.9. When the original image input to the CNNs model, which convolve the images into different feature maps through the 3×3 and 5×5 convolution kernels. It adopts the activation function of Leaky_ReLU to activate the CNNs model and utilizes the 2×2 pooling layer with stride of 2 to reduce the neuron parameters. The activation function converts the linear function of images into a neuron parameter of nonlinear complex function so as to improve the expression ability of the neural network model. The purpose of the pooling layers is to reduce neuron parameters, which are applied in the most of convolutional neural networks.

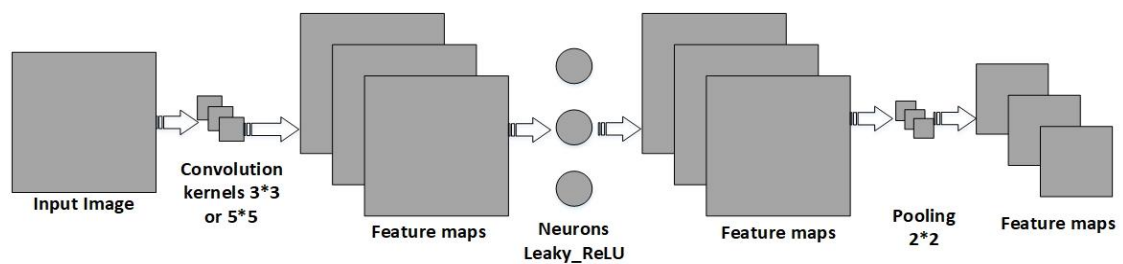


Figure 3.9 The working principle of feature extraction of CNNs

In summary, the types of neurons are the neurons output function and the neurons activation function; the model connections by convolution kernels, neurons activation function, pooling, dropout, and neurons output function; the learning rule of CNNs can be divided into feature extraction and prediction, which convolution layer and pooling layer are responsible for feature extraction, fully-connected MLP is responsible for prediction.

Each step in the model training overview diagram above is a part of the model training including the input processing layer and the convolutional layer, where the input processing layer refers to the processing of the image by the system, the convolutional network layer refers to the use of convolutional layers for feature extraction of human faces. The detailed workflow of the convolution layer is described on details. After feature extraction, we use dropout operation to prevent overfitting. In this process, some neural nodes are randomly deleted. Afterwards, a fully connected layer is used to implement the location and classification of human faces. Ultimately, face position prediction and recognition can be achieved.

Next, we will mainly discuss the details of the model and the internal parameters of the convolutional layer. We see that the model design structure from Figure 3.11.

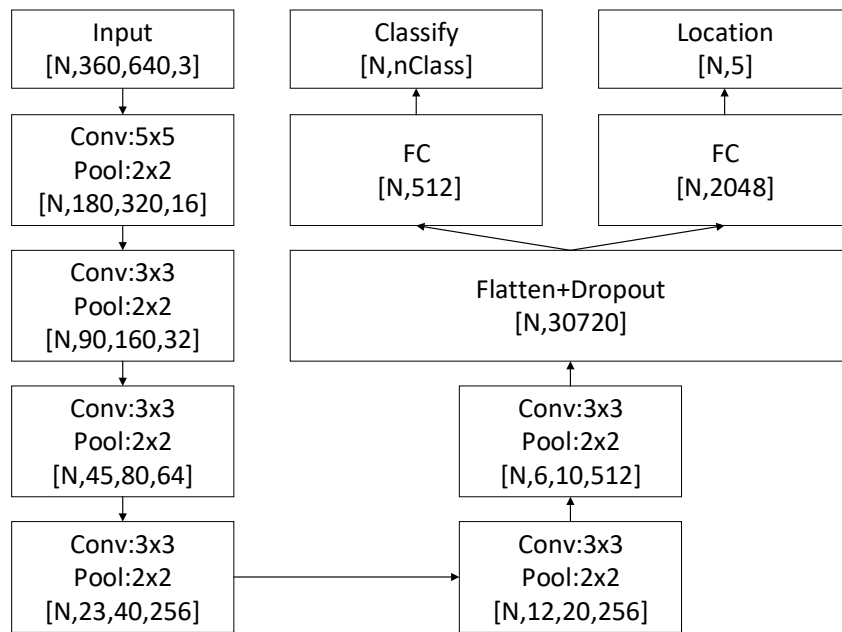


Figure 3.11 Model design structure

In the process of designing the model, we have referenced many models, combined with the complexity of our tasks and our goals as well as the limitations of our operating equipment. We have designed the DNNs network structure shown in Figure 3.11. We have found that such a network structure can well realize the location and recognition of human faces. Then, we will discuss the framework and operate principles of this model.

First, when the DNNs model is used to train the input image, it is not an image followed by an image for training, but training N images in one step. This N can be set as batch size. We see that at the very beginning of this model, we need to input the images, the dimensions of the model parameters are 4D, namely, $N, 360, 640, 3$. Among them, 360×640 is the size of these images, 3 is the number of channels of the image, because the general image is an RGB 3-channel image.

In the following several convolutional layers of the model, we also see that the dimension represents the feature that has heights H and width W , channel number is C . Then, we set up six convolutional layers in our model. In these six convolutional layers, we set up a 5×5 large convolution kernel in the first convolutional layer to capture a wide range of image information and set a 3×3 small convolution kernel at the next five convolutional layers, which is used to capture a small range of image information. In each of the six convolutional layers, a 2×2 maximum pooling layer is set for each convolutional layer. The maximum pooling operation means that for each 2×2 small mesh, the maximum value is taken as the output, and the output also reduces the size of the special name at the same time.

After feature extraction of six convolutional layers, we expand the features of each sample into a vector and convert the features into a matrix. While expanding the vector, we add dropout operation into it. The dropout operation indicates that at the time of output, the values of some nodes are randomly deleted. If we use the dropout before fully-connected layers, we can effectively prevent overfitting.

Then, we constructed two dual-layer MLPs to implement object classification and location. The dimension of the output of the classified MLP is $[N, nClass]$, where $nClass$ represents the number of categories. In our experiment, $nClass=5$. Each row of the output results represents a classification vector

$$p = [p_1, \dots, p_{nClass}], \quad (3.17)$$

where p_k represents the probability of the k -th classification of the input of this sample. Used to measure the gap between the predicted classification and the actual classification of the sample, cross entropy is generally used as a loss function. Suppose that the actual classification of the sample is j , then the predicted classification vector of the sample is

$$q = [q_1, \dots, q_{nClass}]. \quad (3.18)$$

It satisfies that

$$q_k = \begin{cases} 1 & k = j \\ 0 & k \neq j \end{cases}. \quad (3.19)$$

The loss function of classification model is

$$L_{cls} = -\sum_{k=1}^{nClass} q_k \log p_k. \quad (3.20)$$

Since the feature vector of the output of the location MLP is $[N, 5]$, each row of the output result represents the code of an ellipse shape. Let \hat{E} be the ellipse position predicted by the model, E be the actual ellipse position of the sample, then the location loss function of model is

$$L_{loc} = \sum_{k=1}^5 f(\hat{E}_k - E_k). \quad (3.21)$$

We are calculating norms of vector $\hat{E}_k - E_k$. When $f(x) = |x|$, this is a **1**-norm, when $f(x) = x^2$, this is a **2**-norm. The **1**-norm is not prone to gradient explosions, but the derivative is not continuous; the **2**-norm is prone to gradient explosions, but the derivatives are continuous. After considering the two points above, we choose the smooth **1**-norm as f , then

$$f(x) = \begin{cases} |x| - 0.5 & x < 1 \\ 0.5x^2 & x \geq 1 \end{cases}. \quad (3.22)$$

It is worth mentioning that we chose Leaky ReLU as the activation function of the model, where

$$LeakyReLU(x) = \begin{cases} x & x \geq 0 \\ 0.1x & x < 0 \end{cases}. \quad (3.23)$$

In the previous section, we used $E = [x_c, y_c, a, b, \phi]$ as the ellipse code to analyse the new parameters of the ellipse after data augmentation. In actual training, we do not use this code directly, because the size of each component of the code has a different

effect on the position of the ellipse. Thus, we designed two codes for training. The first code is

$$E_1 = [x_c, y_c, a, b, 180\phi/\pi] . \quad (3.24)$$

The only difference between this encoding and the original encoding is that we change the angle of ellipse from radian to angle. Because we believe that the effect of shifting the position of the ellipse by 1 pixel and the angular deviation of the ellipse by 1 degree is approximate. The deviation of 1 radians (approximately 57.3°) is a large deviation.

The second code is

$$E_2 = [x_1, x_2, y_1, y_2, b], \quad (3.25)$$

where $(x_1, y_1)(x_2, y_2)$ is the two endpoints of the long axis and b is the length of the minor axis. This can also uniquely identify an ellipse. Conversion from the original code E to E_2 can be used as

$$\begin{aligned} x_1 &= x_c - a \cos \phi \\ y_1 &= y_c - a \sin \phi \\ x_2 &= x_c + a \cos \phi \\ y_2 &= y_c + a \sin \phi \\ b &= b \end{aligned} . \quad (3.26)$$

The conversion from code E_2 to original code E can be used

$$\begin{aligned} x_c &= \frac{x_1 + x_2}{2} \\ y_c &= \frac{y_1 + y_2}{2} \\ \phi &= \arctan \frac{y_2 - y_1}{x_2 - x_1} \\ a &= \frac{1}{2} \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \\ b &= b \end{aligned} . \quad (3.27)$$

Compared to the first encoding, the theoretical advantage of the second encoding is that each component of the encoding is based on pixels, but for the first encoding, first 4 classifications are in pixels units and the fifth components are in angle units. Therefore, the components of this coding are more uniform.

Finally, the loss function of the model training is

$$L = L_{cls} + wL_{loc}. \quad (3.28)$$

where w is the weight of the location, we will try to take $w \in \{1,10\}$ during our training. The process of DNNs model training is to use the training data and minimize the loss function.

In addition, in order to measure the accuracy of this model, we also need to establish criteria to determine the correct prediction results. In the general object detection task, determining a prediction results requires: (1) the classification result is correct; (2) IoU of the prediction bounding box and the real bounding box is greater than 0.5.

Because the traditional bounding box is a rectangle oriented to the axis, it is relatively easy to calculate the IoU value. Let the model predicting the bounding box be $A = [x_1, x_2, y_1, y_2]$ and the real bounding box be $B = [x_3, x_4, y_3, y_4]$, then

$$IoU(A, B) = \frac{S_{A \cap B}}{S_{A \cup B}} = \frac{S_{A \cap B}}{S_A + S_B - S_{A \cap B}}, \quad (3.29)$$

where $S_{A \cap B}$ denotes the area of the intersection of A and B, $S_{A \cup B}$ denotes the area of the union set of A and B. For the case of a rectangle

$$\begin{aligned} S_A &= (x_2 - x_1)(y_2 - y_1) \\ S_B &= (x_4 - x_3)(y_4 - y_3) \\ S_{A \cap B} &= \max(0, \min(x_2, x_4) - \max(x_1, x_3)) \max(0, \min(y_2, y_4) - \max(y_1, y_3)) \end{aligned} \quad (3.30)$$

But for the ellipses $A = [x_{c1}, y_{c1}, a_1, b_1, \phi_1]$ and $B = [x_{c2}, y_{c2}, a_2, b_2, \phi_2]$, though we can calculate

$$\begin{aligned} S_A &= \pi a_1 b_1 \\ S_B &= \pi a_2 b_2 \end{aligned} \quad (3.31)$$

However, it is very difficult for $S_{A \cap B}$. Therefore, we decided to calculate the IoU of two ellipses using IoU of the ellipse's bounding rectangle. Then, we need to determine the parameter representation of the rectangle of the ellipse $E = [x_c, y_c, a, b, \phi]$.

Examining the parametric equation of the ellipse

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos \phi & -\sin \phi \\ \sin \phi & \cos \phi \end{pmatrix} \begin{pmatrix} a \cos t \\ b \sin t \end{pmatrix} + \begin{pmatrix} x_c \\ y_c \end{pmatrix}. \quad (3.32)$$

We write it as

$$\begin{cases} x = a \cos \phi \cos t - b \sin \phi \sin t + x_c = \sqrt{(a \cos \phi)^2 + (b \sin \phi)^2} \sin \left(t - \arctan \frac{a \cos \phi}{b \sin \phi} \right) + x_c \\ y = a \sin \phi \cos t + b \cos \phi \sin t + y_c = \sqrt{(a \sin \phi)^2 + (b \cos \phi)^2} \sin \left(t + \arctan \frac{a \sin \phi}{b \cos \phi} \right) + y_c \end{cases} \quad (3.33)$$

Obviously, there is

$$\begin{aligned} x_c - \sqrt{(a \cos \phi)^2 + (b \sin \phi)^2} \leq x \leq x_c + \sqrt{(a \cos \phi)^2 + (b \sin \phi)^2} \\ y_c - \sqrt{(a \sin \phi)^2 + (b \cos \phi)^2} \leq y \leq y_c + \sqrt{(a \sin \phi)^2 + (b \cos \phi)^2}. \end{aligned} \quad (3.34)$$

Therefore, we can get the bounding rectangle of ellipse is

$$[x_c - w_c, x_c + w_c, y_c - h_c, y_c + h_c]. \quad (3.35)$$

Among them,

$$\begin{aligned} w_c &= \sqrt{(a \cos \phi)^2 + (b \sin \phi)^2} \\ h_c &= \sqrt{(a \sin \phi)^2 + (b \cos \phi)^2}. \end{aligned} \quad (3.36)$$

After obtained the bounding rectangle of the ellipse, the IoU value of the ellipse can be easily calculated according to the calculation method of the rectangle IoU.

Chapter 4

Results

This chapter mainly shows the results of face recognition of our experiments. The result of face recognition will also be introduced on details. We will also list the comparative evaluations of the four models. In the end, we will select the model with the highest verification set as our final model.

4.1 Training Model

In our experiment, we mainly used three related software to achieve it, which are MATLAB, Python and TensorFlow. The work we did with MATLAB are labelled images and data augmentation, which mainly are used to data processing. It is also used for data input and output, including feeding data into Python and accepting output data from Python; the main work in this experiment is to use Python to implement it, including model construction and training; TensorFlow provides an environmental framework of deep learning; we transfer the model framework using TensorFlow.

According to the model design above, we trained the two sets of codes $E_1 = [x_c, y_c, a, b, 180\phi/\pi]$ and $E_2 = [x_1, x_2, y_1, y_2, b]$, taking weight $w \in \{1,10\}$, and trained a total of four different models.

In our dataset, we have 20,000 training samples, we take a batch size 50, and train 75 epochs. In order to fully train the model, we set the number of training steps to 30,000 steps. Then, we used TensorFlow to implement the model and accelerate training on the NVIDIA GTX 1070 graphics card, each model takes about 4 hours of training, and the four models cost us about 16 hours.

After training, we test results of the training, which means we use the test video to visually reflect the effect. We put the test video in the specified folder, click on the program to run, and the test results will be automatically saved to the program. We found that each person's face is located and classified. An example of this test is shown in Figure 4.1.



Figure 4.1 The resultant example of the test video

From the test results, we found that human face can be well positioned and classified for multiple objects with various distances as well as different face angles, or external conditions such as lighting. It can accurately find a human face and identify the identity of the object. This achieves the purpose of using deep learning for human face detection and recognition.

4.2 Comparison and Analysis of the Four Models

According to the above introduction, we set four models to compare each other. For the same training samples, we compare the accuracy based on the training dataset and the accuracy on the validation set. The training results for the four models are shown in Table 4.1.

Table 4.1 Comparisons of training results of the four models

Models	Bounding Boxes	w	Means	Trainings	Validations
I	$[x_c, y_c, a, b, 180\phi/\pi]$	1	0.9736	0.9947	0.8890
II	$[x_c, y_c, a, b, 180\phi/\pi]$	10	0.9714	0.9935	0.8832
III	$[x_1, x_2, y_1, y_2, b]$	1	0.9718	0.9907	0.8962
IV	$[x_1, x_2, y_1, y_2, b]$	10	0.9731	0.9910	0.9018

Through comparing the four models, we see that accuracy of the four models on the training dataset has reached more than 99%, which indicates that the four models have been fully trained. However, there are some differences with regard to accuracy of the validation set. Among them, assume the ellipse code is E , and the positioning weight $w = 10$ of the model II has the lowest accuracy on the validation set, only 88.32%.

Suppose the ellipse code is E_2 , the positioning weight $w = 10$ of the model IV has the highest accuracy on the validation set which can reach 90.18%. Thus, we use model IV with a higher accuracy as our final model.

In order to better understand the training process and results of the four models, we have the training curve as shown in Figure 4.2~4.5.

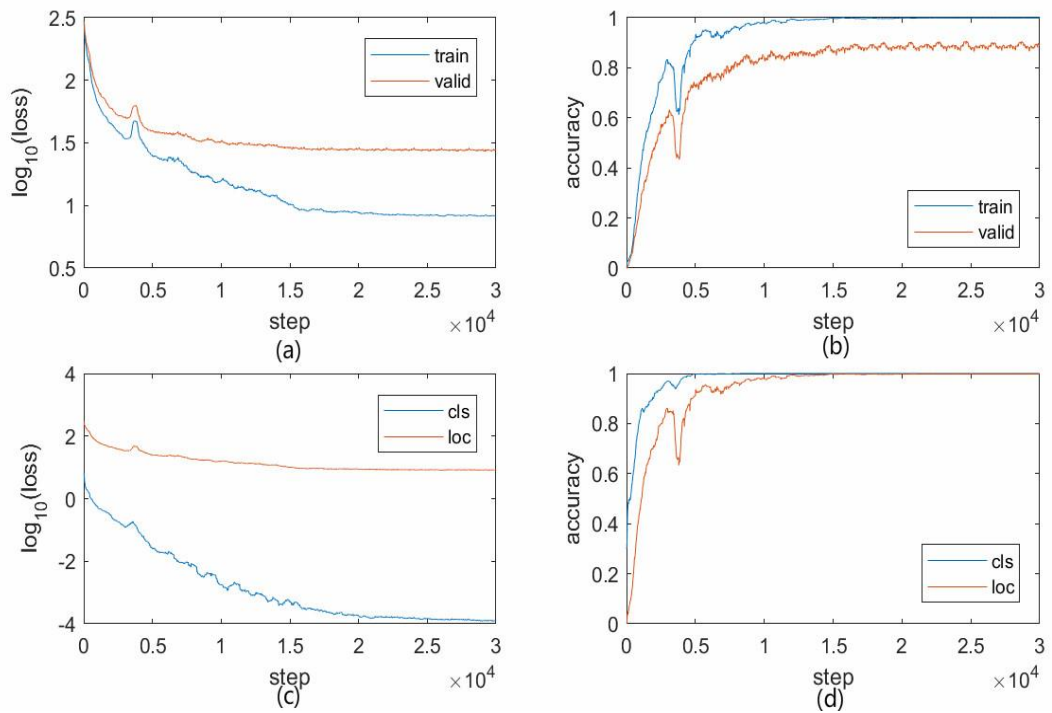


Figure 4.2 Training curve: $box=1$ and $w=1$

- Fig. 4.2 (a) represents the loss function of models based on training dataset and the loss function based on the validation set, which mainly are used to observe whether the model exists overfitting. When the loss function decreases on the training set, the loss function does not increase on the validation set, but also decreases, which indicates that this model does not have to be fitted.
- Fig. 4.2 (b) shows the changes in the accuracy of training dataset and validation dataset. The accuracy of training set is mainly used to observe whether the model has been fully trained. From Figure 4.2(b), we see that the training set is accurate, the accuracy is almost 100%. This shows that this model has been fully trained. Starting from Step 20,000, the accuracy based on the validation set is also basically stable and reaches the rates more than 80%.
- Fig. 4.2 (c) is used to compare the loss function values of the localization and classification on the training set, where the blue line represents the classification loss function value and the red line represents the location loss function value. From Fig. 4.2(c), we see that the decreasing amplitude of the classification loss function is larger than the value of local loss function, which indicates that in the process of training, positioning is more complex than classification; the positioning is more difficult to be trained.

- Fig. 4.2(d) is used to compare the changes in the accuracy of location and classification on the training set. In Fig4.2(d), the blue line increases faster than that of the red line, and the accuracy of classification is also the first to be stable and convergent. The accuracy of this model is affected by the correctness of positioning after approximately 5000 steps.

Next, we will introduce the training curve of $box=1, w=10$ as shown in the Figure 4.3.

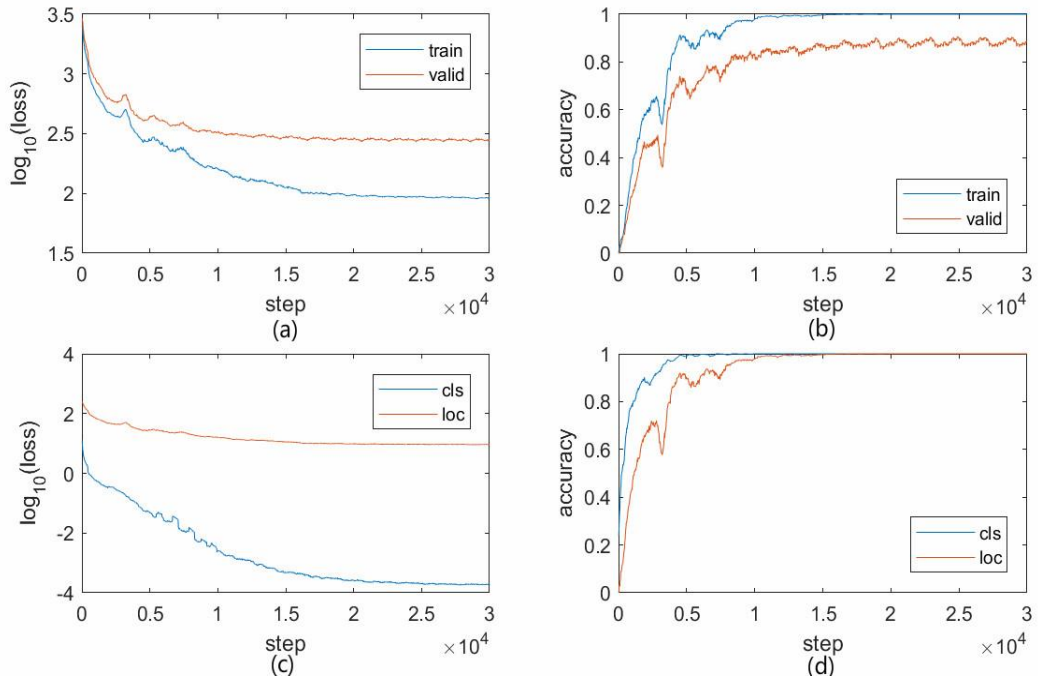


Figure 4.3 Training curve: $box=1$ and $w=10$

- Same as Figure 4.2, Fig. 4.3(a) compares the loss function in the training set and the validation set. The loss functions on the training set and validation set are falling rapidly before 15,000 steps, and slow down after 15,000 steps, which reflects that this model has not been fitted. After 20,000 steps, it slowly and steadily shows that the accuracy of the model is basically stable.
- From Fig 4.3(b), we see that the blue curve represents the accuracy of the training set, the red curve indicates the accuracy of the validation set, and the accuracy on the training set is basically stable after 10,000 steps, nearly 100%, this shows that the model is also fully trained, the accuracy on the validation set fluctuates a little bit after 10,000 steps, and basically converges after 25,000 steps.

- Fig. 4.3(c) is the loss function for classification and location. Comparing with the first model, the loss function of classification obviously decreases than that of the location, which indicates that the classification is easier to be trained.
- Fig. 4.3(d) is the accuracy of comparative location and classification. From approximately 2000 steps, the location accuracy is more than that of classification. The accuracy of classification began to converge at about 5,000 steps, and the accuracy of location start converging at 10,000 steps.

Subsequently, we will discuss the training results of the third model from Figure 4.4.

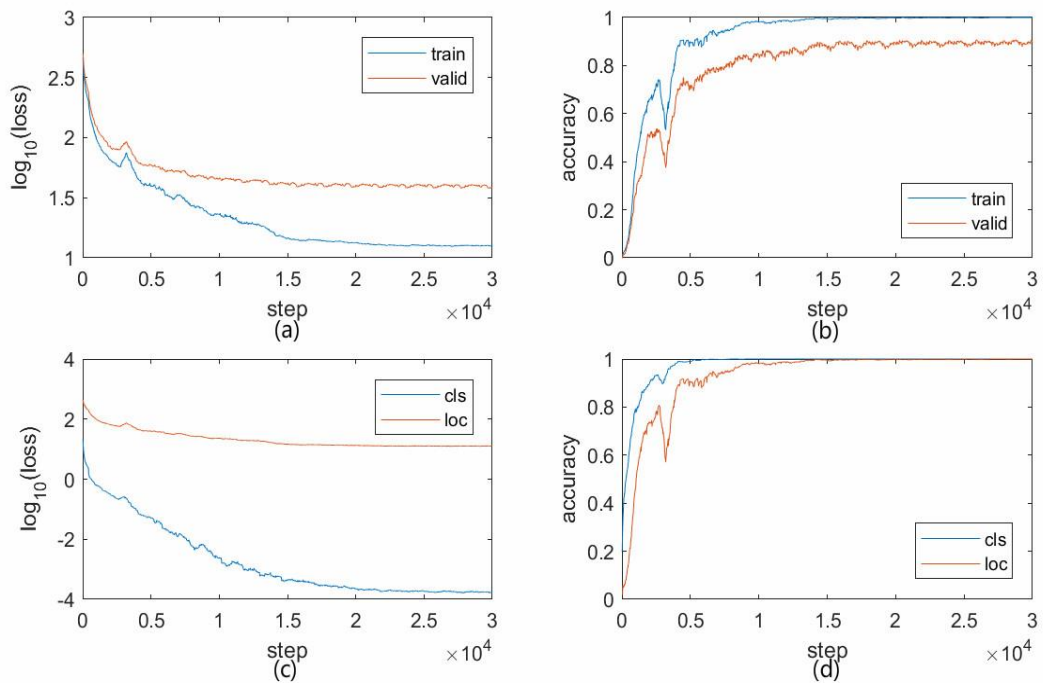


Figure 4.4 Training curve: $box=2$ and $w=1$

- In the comparison of the loss function of Fig. 4.4(a), there is no overfitting in this model. The loss functions of training set and validation set converge basically after 20,000 steps.
- Fig. 4.4(b) is the accuracy based on the training set and the validation set. The training set converges approximately at 15,000 steps, while the validation set tend to approximately at 25,000 steps. Although the loss function based on the training set continues to be decreased, the accuracy of the training set no longer increases in the subsequent training, and the accuracy of the validation set only increases slightly.

- Fig. 4.4(c) compares the loss function values of classification and location, the loss function of classification is still dipping faster than the loss function value of the location, which indicates that the classification is still easy to be trained.
- Fig. 4.4(d) reflects the accuracy of classification and location, the accuracy of the classification also increases faster than the accuracy of the location, and it is more convergent.

After that, the analysis results of the fourth model are shown in Figure 4.5. This model is also the model we used because it has the highest accuracy among these four models.

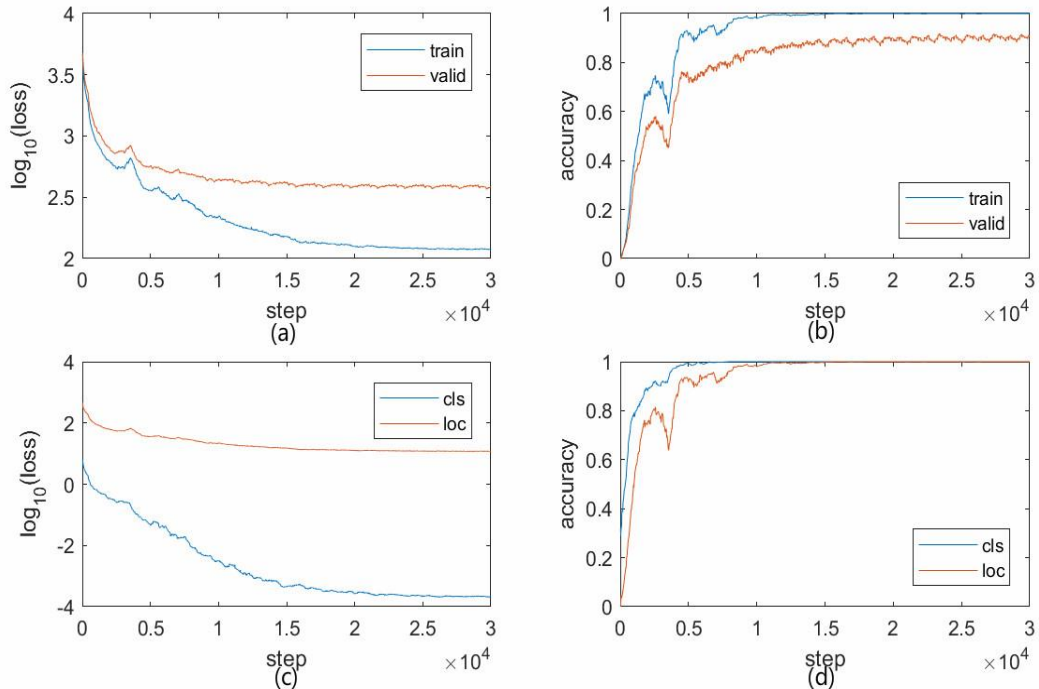


Figure 4.5 Training curve: $box=2$ and $w=10$

- In the fourth model, in the graph that explores the value of loss function, the blue curve represents the value of loss function on the training set, we see that when the blue curve is decreasing, the red curve does not increase, it also steadily declines until convergence.
- Fig. 4.5(b) shows the accuracy based on training set and validation set. We see that the accuracy on the training set converges around 11,000 steps, the training model is fully trained, and the accuracy based on the validation set reaches more than 80% around 10,000 steps. It is slow after 10,000 steps, the final accuracy on validation set reaches about 90%.

- With regard to the loss function for classification and location, the classified curve has a larger decrease than the localized curve; the loss function of the localization has a small decrease, the location is difficult to be trained and can hardly be affected.
- Similarly, we see that in the accuracy of classification and location, the classification is less fluctuating than the location, the classification curve converges at about 5,000 steps, and the location curve converges at about 10,000 steps; there is little difference between these two, but the location curve is still more complicated than that of classification.

It is worth noting that if we use original data to make graphs, the changes in the graphs will not be very intuitively. Each point in the graph is the average of 20 data points of the original data, so that we can better understand the relevant trends.

Next, we will compare the accuracy based on the validation set for these four models. Figure 4.6 shows that the trend of accuracy based on the validation set. In the plot, we see the curves of the four models together so that we can better understand the trend of the curves.

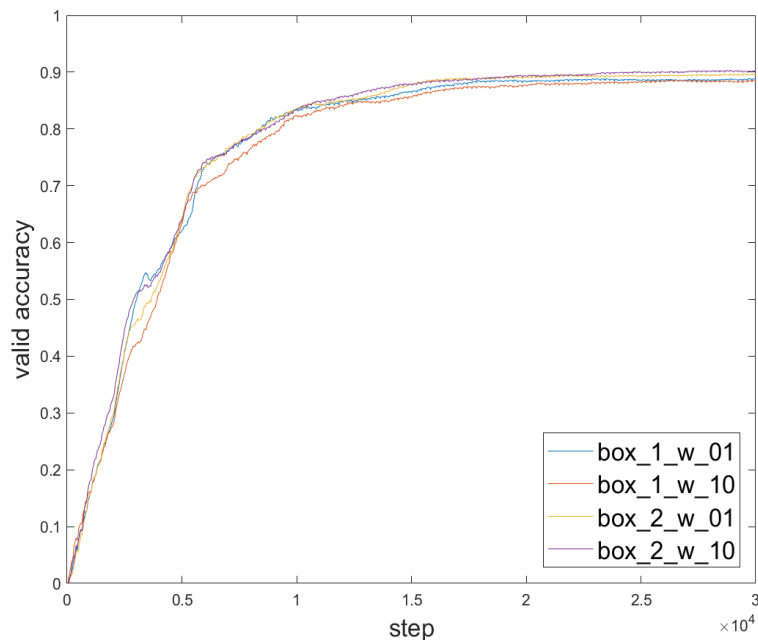


Figure 4.6 Accuracy comparisons of four models on the validation set

In Figure 4.6, the blue curve represents the first model, the red line shows the second model, the orange line indicates the third model, and the purple line is on behalf of the fourth model. Before 6,000 steps, the accuracy of these four models rises rapidly, and then slowly increases until their convergence after 6,000 steps. We easily know from Figure 4.6, the convergence of the four models is basically the same during training, and all models basically converge after 20,000 steps. After 20,000 steps, the strengths and weaknesses of each model are determined. We see that the purple line has the highest accuracy and the best effect. Therefore, we choose model IV as our final model.

4.3 Effect of Face Ratio on Accuracy

After the model training, we choose model IV as our final model. Our model can label every frame of the input video, but the face is difficult to be recognized, because this model has some drawbacks to recognize small objects. Our results are shown in Figure 4.7. Marked results can be divided into three parts:

- Use an ellipse shape to mark the face contour. We use ellipse markers, because it not only reflects the position of human face but also reflects the angle of human face. This will identify face patterns from multiple perspectives.
- The ellipse bounding box.
- The classification of human faces and the possibility of faces belongs to this category. For example: “Person No.5: 1.00” indicates that the probability of this person's face is 100%.



Figure 4.7 Recognition results

Of course, we will get different sizes of a face at different distances. As shown in Figure 4.8, when a person moves closer to the camera, the system will detect the human face with different sizes in different distance. The size of this face in the image is also an important factor that affects the recognition effect:

- If the face is large, there are few samples, the model can obtain more abundant face information. It will be easier to locate and classify the face and obtain better accuracy.
- If the face is small, the number of samples is large, there are a lot of interferences, the face information is relatively vague and results in that the model cannot detect the face accurately, thus it reduces the accuracy.



Figure 4.8 Face with different sizes

Because the purpose of this thesis is to investigate face detection and recognition based on the accuracy, the size of human face can be measured by using proportion of the face to the total size of the image. Let the width and height of the image be w, h , respectively, the ellipse shape of a face is denoted as $\{x_c, y_c, a, b, \phi\}$, then the area of the bounding box of the human face is

$$S_F = 4\sqrt{(a \cos \phi)^2 + (b \sin \phi)^2}\sqrt{(a \sin \phi)^2 + (b \cos \phi)^2}. \quad (4.1)$$

Thus, the area ratio of the face in the image can be obtained by using

$$A = \frac{4}{wh} \sqrt{(a \cos \phi)^2 + (b \sin \phi)^2} \sqrt{(a \sin \phi)^2 + (b \cos \phi)^2}. \quad (4.2)$$

Obviously, if A is larger, which indicates a larger face in the image, people are closer to the camera; otherwise, if A is smaller, which reveals a smaller face is detected in the image, that represents that human face is far from the camera.

In order to better understand the influence of proportion of a human face on the accuracy, we will calculate the recognition accuracy and IoU value for different sizes of faces.

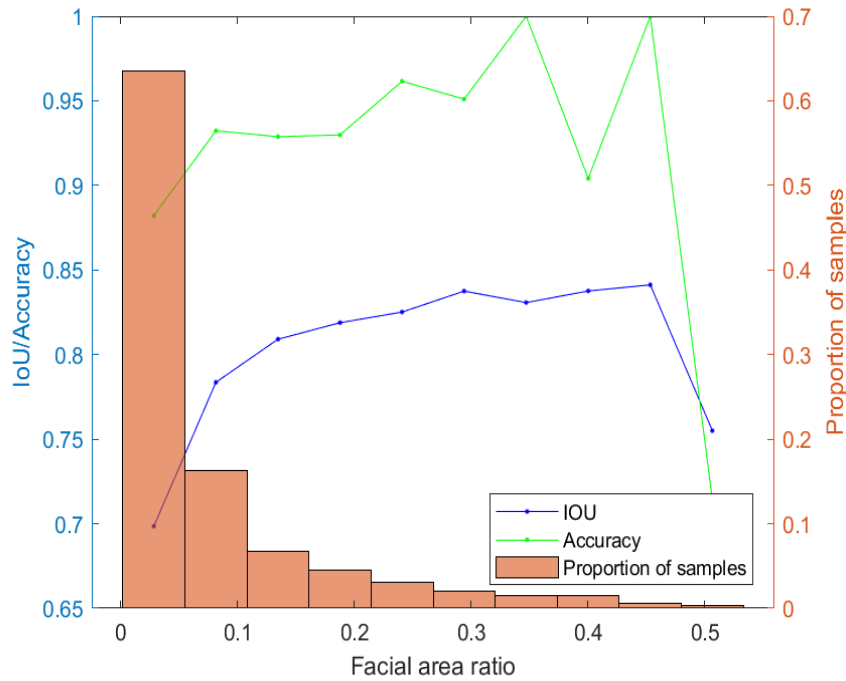


Figure 4.9 The relationship between accuracy and facial proportion

Among them, IoU value can more accurately reflect the accuracy of the recognition because we set the $\text{IoU} > 0.5$ is the correct recognition results, but if the IoU value tends to 1.0, then the bounding box of the face is close to the actual results.

In Figure 4.9, the histograms in Fig. 4.9 represent the proportion of a face having different sizes in the samples. The green line represents the changes in accuracy, meanwhile, the blue line stands for the change in IoU. The y-axis scaling on the left corresponds to the accuracy and the IoU value, the y-axis scaling on the right corresponds to the proportion of the sample.

From the histogram, we see that if the proportion of a human face is small, the number of samples is large; if the ratio of faces increases, the number of samples gradually decrease. At the same time, both the accuracy and the IoU value are increasing first and then decrease. This is because: (1) if the face occupies a large proportion area of the image, it is easily to be recognized and the accuracy is high; (2) the proportion of the face area to the image is large, the number of samples is less, and the insufficient training of the model for this sample. Therefore, we see that the values of accuracy and IoU have been changed around the ratio of 0.4, which begins to decline, because there are few samples, the statistical results fluctuate greatly.

In addition to the influence of face proportions, we also analysed the influence of face angles based on accuracy and IoU. We got the results as shown in Figure 4.10.

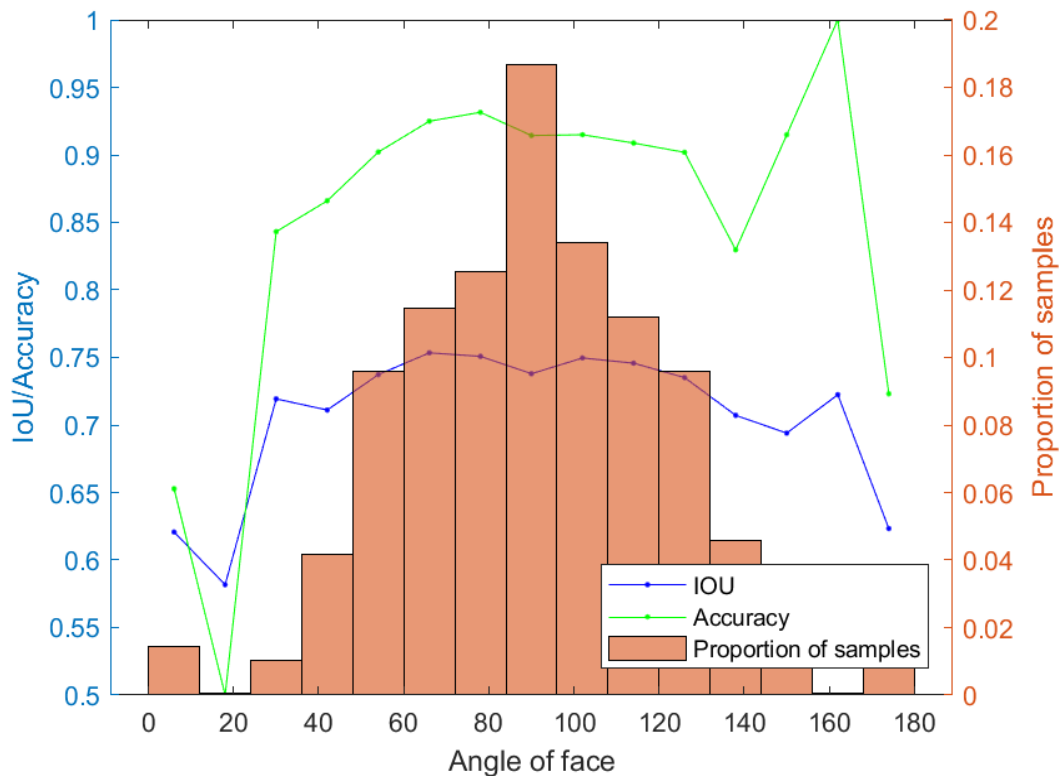


Figure 4.10 The relationship between accuracy and facial angle

We see from Figure 4.10, if the angle of a human face is closer to 90°, the accuracy and IoU value are higher. This is the reason why the number of samples at 90 ° is the biggest one. When the face angle is about 20 ° and 160 °, both the values of accuracy

and IoU are mutated, it is also because there are few samples in these two angles; the accuracy is unstable, and the statistical results fluctuate very much.

To sum up, we conclude:

- The recognition accuracy of this model is mainly affected by the proportion of the face and the number of samples. If the face ratio is large, the recognition accuracy will be higher; if the number of samples is large, the recognition accuracy will be higher.
- Our model has the highest accuracy for face recognition with a medium-sized face and a small angle of inclination, the accuracy becomes lower when the inclination angle becomes larger.

4.4 Classification Results

Classification is also an important part of face recognition because it directly affects the final results. In order to verify the classification results, we use the test video for verification. In this thesis, we have five participants. They are: Person No.1, Person No.2, Person No.3, Person No.4, and Person No.5. Our expectation is that when a person walks in front of the camera, the system will automatically detect the object within a certain distance. Of course, beyond a range, the computer can hardly recognize small faces. We list the example of recognition results in Figure 4.11.





Figure 4.11 Results of face recognition and classification

From Figure 4.11, we show the classification results of Person No.5. The first two images represent the recognition results within the range, and the other two images are the resulting images beyond the identification range because the small object is very difficult to be identified. We also revealed that face proportion plays an important role in recognition, the proportion of faces in the image is small so that it is difficult to be identified. Of course, we also found that in our test video, beyond 4 meters, the face will not be detected.

4.5 Limitations of the Research

Although the model introduced in this thesis has been successfully applied to face recognition, but the face recognition based on deep learning still has some limitations. We will introduce the parts that need improvement below:

- (1) Because the model of deep learning is very flexible; with a large number of architectures and node types, training a large neural network is time-consuming and requires high computer hardware; hence, low budget and low cost are difficult to achieve.
- (2) The outcome of deep learning is based on strict conditions because it requires support from a large amount of data. These data are the criteria for successful model training. This thesis uses data augmentation to enhance the data set; otherwise, the model is difficult to be fully trained.

(3) There are only five participants in this thesis. In future, we will consider more people to participate in this project.

Chapter 5

Analysis and Discussions

In the previous chapter, the experimental results have been presented, but there is no excessive analysis and comparisons. In this chapter, we will focus on analysing the results of experiments and comparing the effects and accuracy of different models. Finally, we will explain the reasons for using this model and the contributions we made in this thesis.

5.1 Analysis

We detailed the direct results of face recognition in the last chapter. In this chapter, we will analyse the accuracy between the different models. At the same time, this chapter explained another model: SSD Inception (v2) we used. Although the results of this model have room to improve them better, the outcomes reflect the availability of our model framework.

5.1.1 SSD Inception (v2) Model

In Chapter 4, our network model and its experimental results were depicted in detail, while we also utilize other CNNs models based on SSD meta-architecture to compare with the model we introduced before. The SSD Inception (v2), as a combined CNNs network model, was used to face recognition by using the same training dataset in this project. The SSD Inception (v2) uses Inception v2 as the basic architecture of the CNNs model and applies the method of SSD meta-architecture to improve the Inception (v2) network model; our purpose is to achieve the better results of object recognition. The flowchart of SSD Inception (v2) is shown in Figure 5.1, which demonstrates the processing steps of face recognition based on SSD Inception (v2) model, including images labelling, random crop and reshape image, feature extraction, predict location, and classify prediction; then, we can achieve face recognition.

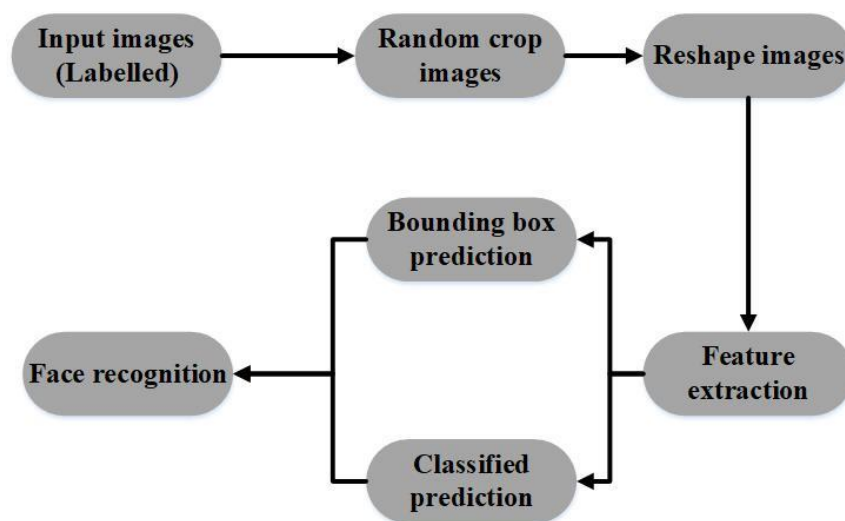


Figure 5.1 Flowchart of SSD Inception (v2) model

The SSD Inception (v2) model is a large CNNs model with 282 layers of convolution layers, the model distribution is shown in Table 5.1. This model is able to be divided into three main parts: image processing, feature extraction, and box predictor. The part of image process is from the 1st convolution layer to the 26th convolution layer, which achieves the effects of data augmentation through adjusting image parameters. The part of feature extraction is from the 27th convolution layer to 258th convolution layer, which extracts the abstract features of images from a deep hidden layer to simulate visual nervous system. The part of box predictor layer is from the 259th convolution layer to 282nd convolution layer, which utilizes the SSD meta-architecture to improve the predict accuracy and frame per second (FPS) of predict results.

Table 5.1 The distribution of SSD Inception (v2)

	Image processing	Feature extraction	Classification and Localization
Layer	1 st to 26 th layer	27 th to 258 th layer	259 th to 282 nd layer
Action	Data augmentation/ adjusting image parameters	Use the base Inception_v2 network	Use SSD method for end-to-end object detection and recognition

The architecture of SSD Inception (v2) is shown in Figure 5.2, which indicates the relationship between image process, feature extraction, and meta-architecture. In the part of image process, the 1st convolution layer to 6th convolution layer execute the action of random crop image, which deal with the shape of objects in images; the 7th convolution layer to 26th convolution layer are responsible for re-adjusting the number of rows, columns, and dimensions of images matrix, which is the processing of reshape. Thus, the part of image processing section mainly performs data augmentation and image adjustment. In the part of feature extraction, it utilizes basic network of Inception (v2) for feature extraction, which involves 231 convolution layers; in the part of box predictor, it applies the method of SSD meta-architecture, which utilizes six different feature layers in the end of the base network of Inception (v2) to predict the location of box. Each box predictor includes four convolution layers, which two convolution layers

are used to implement object classification and two convolution layers are employed to execute object localization.

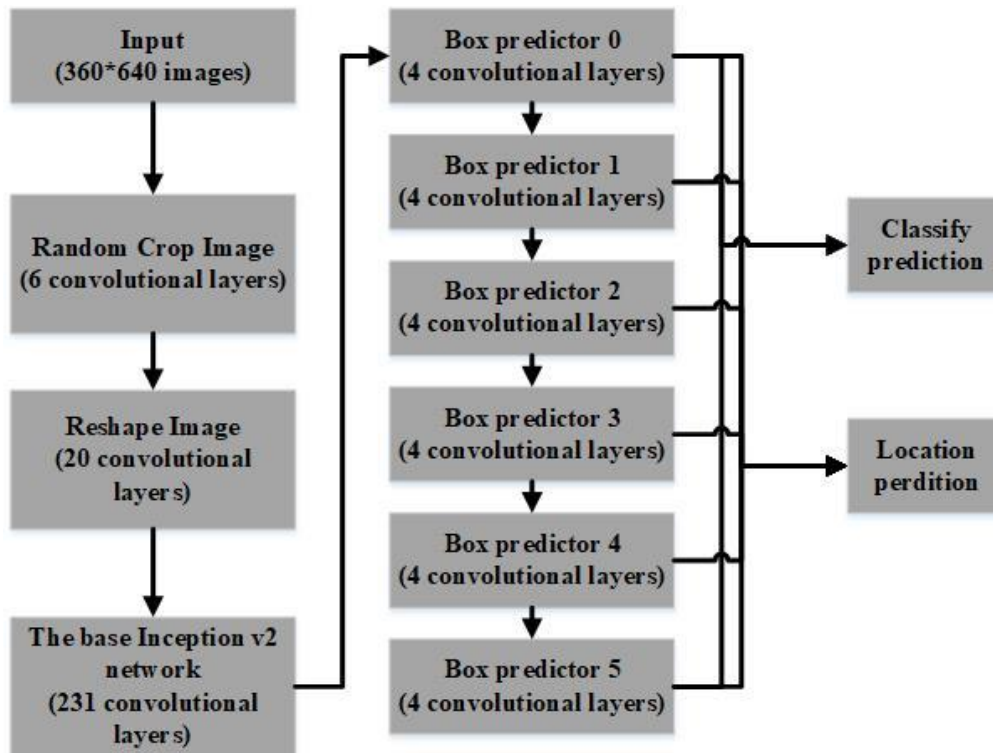


Figure 5.2 Architecture of SSD Inception (v2)

In the model SSD Inception (v2), it costs approximately 82% time in the part of base network for feature extraction and spent about 8% time in the part of SSD meta-architecture for object classification and localization, around 10% time was spent in the part of image processing. The SSD Inception (v2) including the data augmentation section, is difficult to utilize external data augmentation based on the integrity and complexity of its model. In the actual operation of face recognition, its huge network model will also cause overfitting problem, it is also a disadvantage of this model. Therefore, a relatively simple CNNs model might have better accuracy in single object detection.

5.1.2 Simplified SSD model

The simplified SSD network is an end-to-end object detection and recognition model based on the method of SSD meta-architecture, the architecture of the simplified SSD

network model is shown in Figure 5.3. It includes six convolution layers, two fully connected layers, and we dropout optimization between convolution layers and fully connected layers. The convolution layers involving 5×5 convolution kernel and 3×3 convolution kernel to capture the wide and small range of the image information; the 1st convolution layer utilizes 5×5 convolution kernel and 2nd to 6th convolution layers adopt 3×3 convolution kernel. For two fully-connected layers, one of the fully connected layers carries out the prediction of location of bounding box; another fully connected layer executes the classified prediction. We introduced the network model in detail.

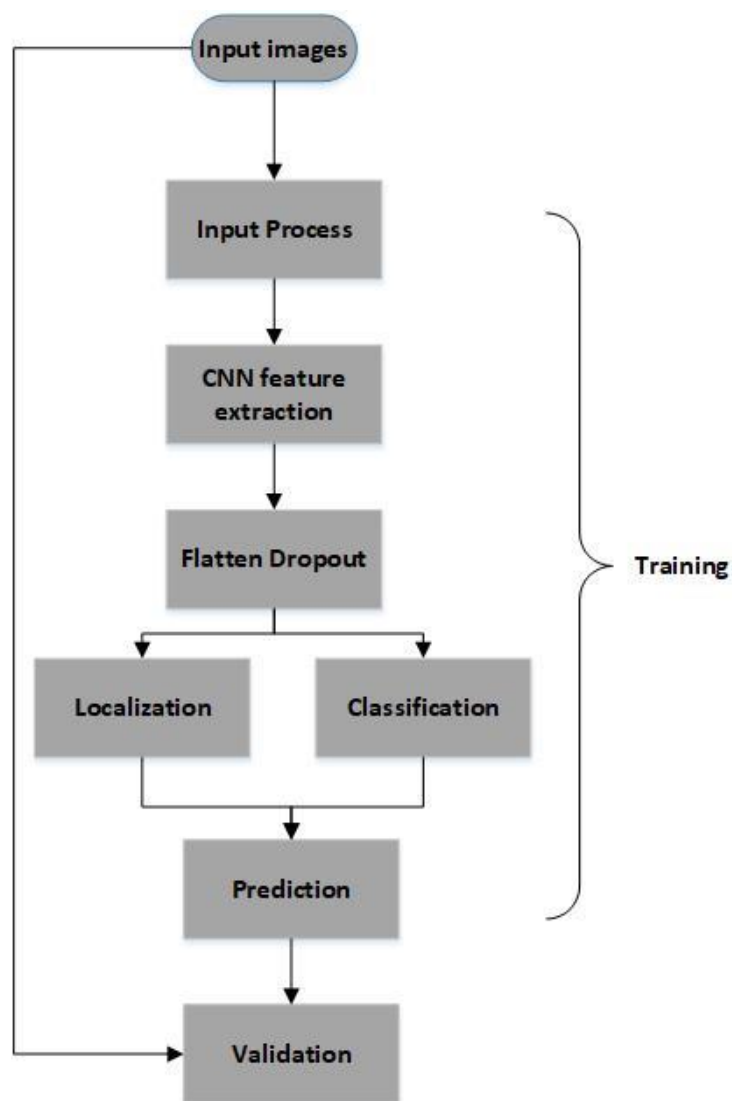


Figure 5.3 Architecture of simplified SSD network model

This model is simpler than SSD Inception (v2), the overfitting is relatively easy to prevent. We adopted the dropout method in the end of convolution layers to optimize the output results in fully-connected layers. Therefore, this network model can effectively avoid overfitting under the sufficient samples of the training dataset. We utilized the external data augmentation method based on the low complexity of this model. We have labelled 500 facial images and each image generates 50 images by scaling to the same size, random flip, random rotate, random crop, scale to the same size, and random color adjustment. Finally, we used 20,000 images to training. The best accuracy of four different models is more than 90%, which proves the correctness of our inference.

5.1.3 Comparisons

5.1.3.1 Comparisons of Architecture

We described the SSD Inception (v2) model and Simplified SSD model in this Chapter. We are going to compare these two models with each other in this section. The comparison is shown in Table 5.2. The SSD Inception (v2) has internal data augmentation in network model, while its difficulty is to use external software for data augmentation, which limits the amount of supervised training dataset; the simplified SSD network model has not internal data augmentation and its facility to adopt external software for data augmentation. At this point, the SSD Inception (v2) model outperforms Simplified SSD network model based on its internal image process, while higher operation leads to overfitting. Therefore, it is difficult to exert on its performance if there is no sufficient amount of training dataset. In additional, the SSD Inception (v2) utilizes the base inception (v2) network for feature extraction, while the Simplified SSD model adopts the six-convolution layer network by using TensorFlow.

Thus, the SSD Inception (v2) model is more difficult to be optimized for overfitting issue based on its large amount of feature extraction layers, while the Simplified SSD network model is able to be outstanding avoid the overfitting issue. The SSD Inception (v2) model adopts the fully convolutional layer for object classification and localization by using six different size of feature map, while the Simplified SSD network model

utilizes two fully connected layers; the SSD Inception (v2) adopted 1×1, 2×2, 3×3 convolution kernel to capture image information, and the Simplified SSD model applied 3×3, 5×5 convolution kernels. Both of network models apply the method of SSD meta-architecture, which utilizes end-to-end object detection and recognition to ensure high FPS of verification.

Table 5.2 Comparisons of architectures

	SSD Inception (v2)	Simplified SSD model
Data augmentation	Internal (1 st layer to 26 th layer)	External software
Feature extraction	231 convolution layers	6 convolution layers
Box prediction	24 fully convolution layers	2 fully connected layers
Convolution kernel	1×1 conv; 2×2 conv; 3×3 conv	5×5 conv; 3×3 conv

5.1.3.2 Comparisons of Training Time

The learning rate indicates the effective training time of the model, if the effective training time is shorter, the model is trained faster. If the learning rate is infinitely close to 0 and no longer fluctuate denotes the final training time. We show the comparison of training process of two models in Table 5.3, which includes training time, training steps, effective training time, and batch size. Amongst them, the setting of batch size depends on the performance of the training machine. We found that if we set the batch size to 20 in SSD Inception (v2) model and the batch size to 50 in the Simplified SSD model, the algorithm can obtain the best and fastest training results. We got the final training time based on these batch size selected.

Table 5.3 Comparisons of training process

	Training time	Training steps	Effective training time	Batch size
SSD Inception (v2)	20h 9m 42s	17.31k	2h 27m 38s	20
Simplified SSD Model I	3h 32m 1s	30.00k	2h 8m 5s	50
Simplified SSD Model II	3h 32m 17s	30.00k	2h 8m 54s	50
Simplified SSD Model III	3h 32m 5s	30.00k	2h 8m 51s	50
Simplified SSD Model IV	3h 33m 39s	30.00k	2h 9m 31s	50

From Table 5.3, we find that the total training time, effective training time, and steps for both CNNs models. Among them, the effective training time refers to the training time before the value of the learning rate converges and tends to 0. Since the internal structure of SSD Inception (v2) is too large, we are not sure whether the computer configuration can afford training, so, we did not set a certain number of training steps for this model. After training for 20 hours, we stopped training. At this time, the number of training steps reached 17,310. However, we found that the training time with fluctuating learning rate was only 2hours and 27minutes; the Simplified SSD Model I took more than 3 hours after 30,000 training steps; the simplified SSD Model II spent 3 hours after 30,000 training steps; the Simplified SSD Model III took more than 3 hours through 30,000 training steps; the Simplified SSD Model IV used more than 3 hours after 30,000 steps training.

According to Table 5.3, it is able to summarize that the training time of the Simplified SSD network models is shorter than that of SSD Inception (v2) model, while the training steps of the Simplified SSD network models are much more than the SSD Inception (v2) model. The simplified network model trained more steps with less time, because the network model of SSD Inception (v2) is much more complex than the Simplified SSD model. However, the effective training time of Simplified SSD network model and SSD Inception (v2) model are very close, the shortest effective training time

of Simplified SSD network Model I used more than 2 hours; the effective training time of SSD Inception (v2) model cost two and half hours. Therefore, learning rate of the Simplified SSD network model is slightly better than the SSD Inception (v2) model, the training efficiency is better than SSD Inception (v2).

5.1.3.3 Activation and Loss Function Comparisons

- **Activation Function**

The activation function of the Simplified SSD network is adopted Leaky ReLU; the SSD Inception (v2) utilizes ReLU. The comparisons of activation function of two CNNs model is demonstrated in Table 5.4. The Leaky ReLU function and ReLU_6 function is shown below,

$$LeakyReLU(x) = \begin{cases} x & x \geq 0 \\ 0.1x & x < 0 \end{cases} \quad (5.1)$$

$$ReLU_6(x) = \max(0, 6, x) \quad (5.2)$$

Table 5.4 Comparisons of activation function

Models	Activation Function
The Simplified SSD model	Leaky_ReLU
SSD Inception (v2)	ReLU_6

The ReLU_6 is defined by using TensorFlow, which is able to solve the gradient vanishing problem; if the training parameter deactivated, then it is difficult to be reactivated. The input parameter might be deactive, when the input parameter is less than 0 or greater than 6, the Leaky ReLU can be used to solve the de-active problem effectively.

- **Loss Function**

The comparison of loss function is shown in Table 5.5. We find that the loss function of the Simplified SSD network model is utilized cross entropy and smooth L1 while the SSD Inception (v2) uses sigmoid and smooth_L1. The equation of cross entropy and smooth L1 of the Simplified SSD model shows in equation (5.3) and (5.4).

$$L_{cls} = - \sum_{k=1}^{n_{class}} q_k \log p_k \quad (5.3)$$

$$SmoothL1(x) = \begin{cases} |x| - 0.5 & |x| > 1 \\ 0.5 x^2 & |x| \leq 1 \end{cases} \quad (5.4)$$

where the classification loss function of SSD Inception (v2) model is

$$Sigmoid(x) = \frac{1}{1 + e^{-x}} \quad (5.5)$$

Table 5.5 Comparisons of loss function

	Classification/Localization	Loss function	Formula
Simplified SSD model	Classification loss function	Cross entropy	$L_{cls} = - \sum_{k=1}^{nClass} q_k \log p_k$
	Localization loss function	Smooth L1	$f(x) = \begin{cases} x - 0.5 & x > 1 \\ 0.5 x^2 & x \leq 1 \end{cases}$
SSD Inception (v2)	Classification loss function	Sigmoid	$f(x) = \frac{1}{1 + e^{-x}}$
	Localization loss function	Smooth L1	$f(x) = \begin{cases} x - 0.5 & x > 1 \\ 0.5 x^2 & x \leq 1 \end{cases}$

5.1.3.4 Comparisons of Loss Value

We also analyzed the loss value results of the Simplified SSD network model and the SSD Inception (v2) network model. The loss function graph of SSD Inception (v2) model is demonstrated in Figure 5.4, the loss function of the Simplified SSD network models is displayed in Figure 5.5 to Figure 5.8. The loss function measures the unit of SSD Inception (v2) model and the Simplified SSD model are different because they adopted different loss functions. From the comparisons, we find that the loss value of SSD Inception (v2) model starts from around 32, and the final value of loss function around 1.86; the Simplified SSD Model I has the lowest value of loss function, the value of loss function starts around 180, and the final value of loss function is less than 9; the final value of loss function of the Simplified SSD Model II is 94.87, and the training time is longer than the Model I; the final value of loss function of the Simplified SSD Model III is around 13.33 and the Model IV is around 136. Therefore, the

Simplified SSD models are better than the SSD Inception (v2) model in the value of loss function, because the loss function of the Simplified SSD model is relatively low. However, because the training process is prone to occurring overfitting problems, this does not mean that the training effect of the Simplified SSD model is better than that of SSD Inception (v2). We also need to compare the final accuracy of the model. In the following, we will mainly introduce the prediction precision of the two models.

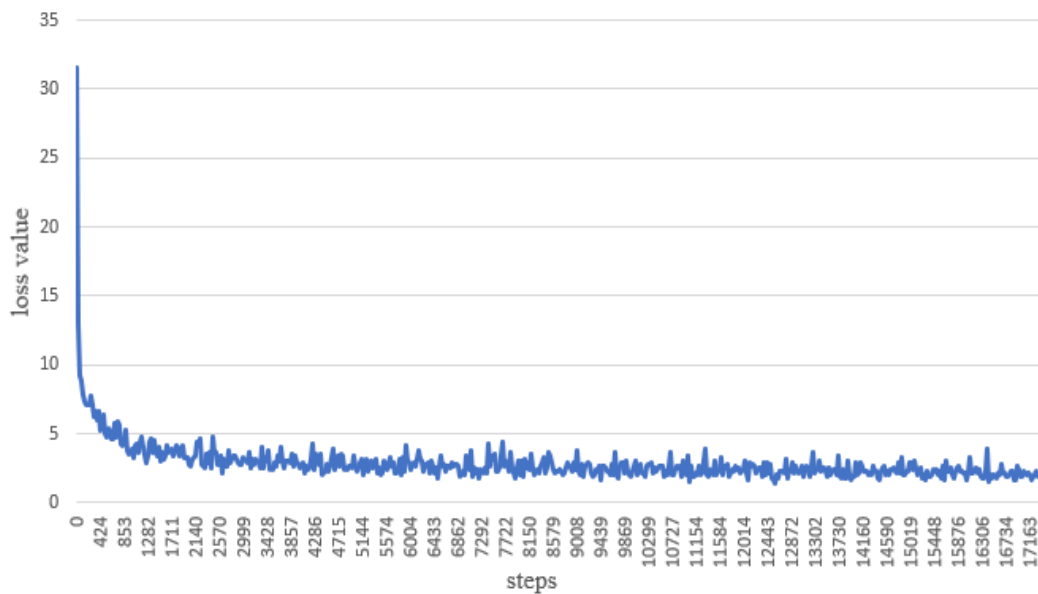


Figure 5.4 Results of loss function of the model SSD Inception (v2)

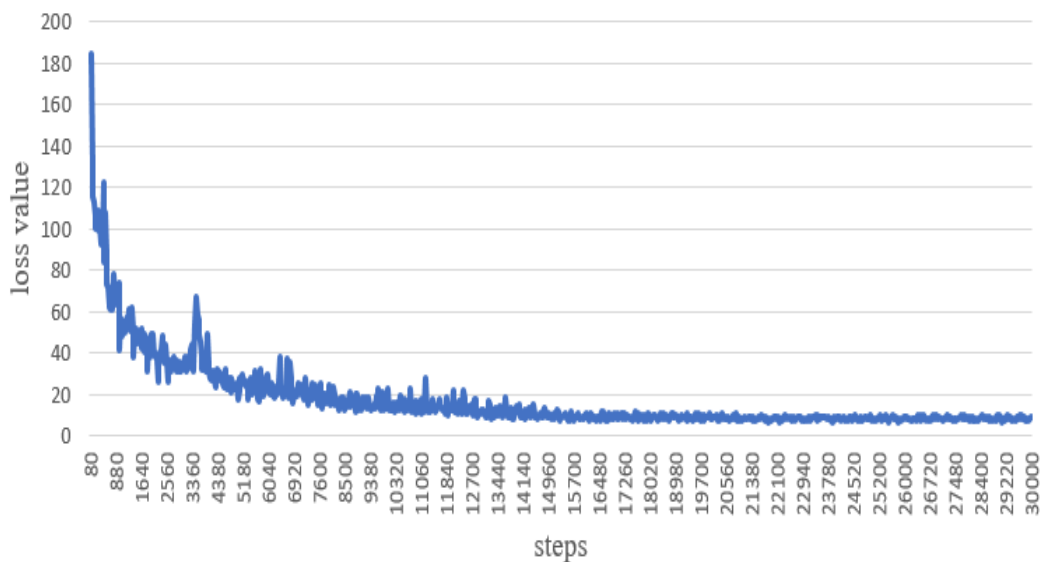


Figure 5.5 Results of loss function of the Simplified SSD Model I

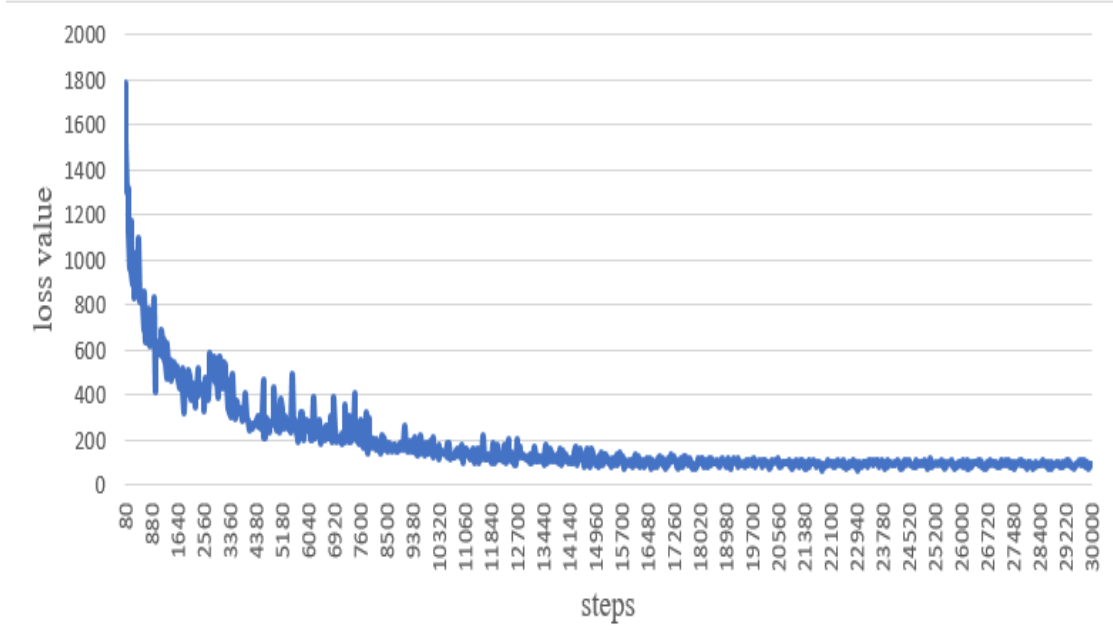


Figure 5.6 Results of loss function of the Simplified SSD Model II

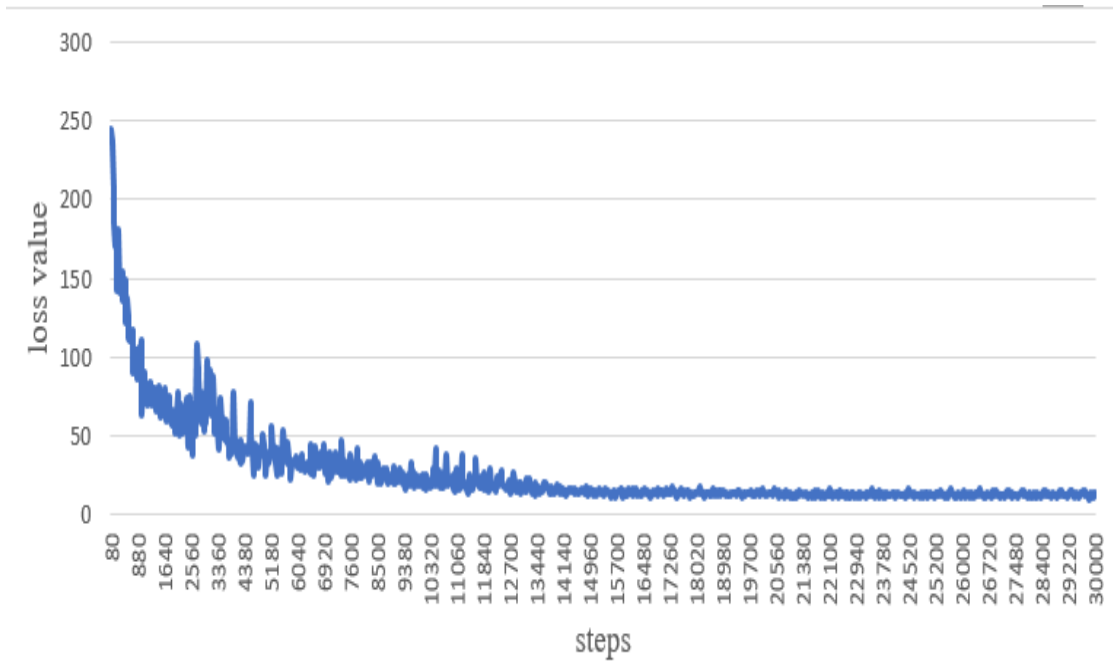


Figure 5.7 Results of loss function of the Simplified SSD Model III

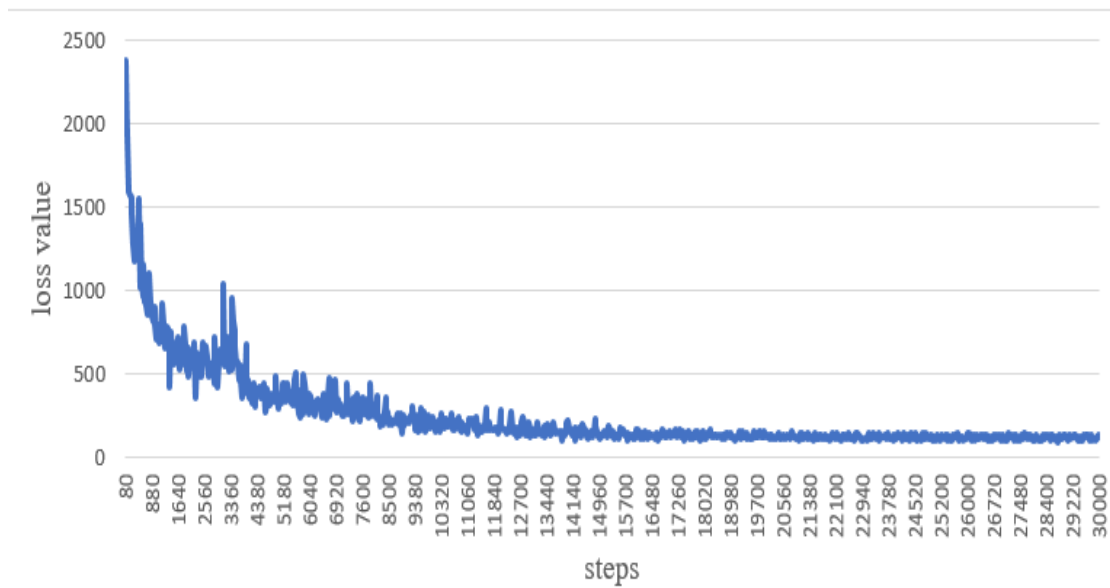


Figure 5.8 Results of loss function of the Simplified SSD Model IV

5.1.3.5 Precision Comparisons

In this thesis, the same training images and verification images are adopted in the Simplified SSD model and the SSD Inception (v2) model. After training, we got the precision of the Simplified SSD model of four models with different parameters and the SSD Inception (v2). The comparisons with regard to precisions for two network models are shown in Table 5.6. From the model comparisons, we found that the precision of the Simplified SSD models is much higher than that of SSD Inception (v2). In four Simplified SSD models, the Model IV has the highest precision, which can reach 90.18%. Therefore, the Simplified SSD models are more suitable than SSD Inception (v2) network model for face recognition.

Table 5.6 Comparison of precisions

Models	Precisions
SSD Inception (v2)	65%
Simplified SSD Model I	88%
Simplified SSD Model II	88%
Simplified SSD Model III	89%
Simplified SSD Model IV	90%

The SSD Inception (v2) model is more complex than that of the Simplified model, but the verification accuracy is lower than that of the Simplified SSD model. The reasons can be divided into threefold: 1) SSD Inception (v2) model has high complexity, which is difficult to achieve external data augmentation and the internal image processing is also difficult to be controlled; it is easy to cause insufficient training of data sets; 2) the SSD Inception (v2) model has a higher depth and more parameter, which makes it more difficult to optimize the overfitting problem; 3) the depth of CNNs network model is hard to avoid the vanishing gradient problem.

5.2 Discussions

From the analysis of this thesis, we found that the SSD Inception (v2) model is more complex than the Simplified SSD model, because SSD Inception (v2) contains 282 convolution layers, while the Simplified SSD model has only 6 convolution layers. The Simplified SSD model dealt with the input images from the external data augmentation including 20,000 images, while the training dataset of SSD Inception (v2) model only is able to improve it by using the internal image process. However, the precision of the Simplified SSD model is better than the SSD Inception (v2) model because large models have complex structures that are prone to solving different problems.

We compared the Simplified SSD model and SSD Inception (v2) model, we found that the simplified model has shorter training time and higher accuracy. Therefore, a complex CNNs model is not necessarily better than a simple CNNs model, the suitability of the training dataset and the CNNs model is also important. With the development of deep learning model, the depth and complexity of CNNs models are continuously and constantly being developed, but the precision of validating has always been difficult to achieve a breakthrough improvement, which requires high performance of the hardware for training. It reflects the bottleneck in the development of deep learning in CNNs model.

Chapter 6

Conclusion and Future Work

We have elaborated on the model and algorithm on details. After verification, we know that our model can be used to recognize human faces. In this chapter, we will summarize this thesis and our results; we will also depict our future work and the possible improvements in this thesis.

6.1 Conclusion

The purpose of this thesis is to identify the identity of each person in front of a camera. The main analysis is the classification of a human face and the influence of proportion of the human face on the accuracy. We propose five classes; each class represents a classification. Under the same model framework, we compared four models with different parameters and summarized the best model as our final model. During the analysis phase, we compared the training results of different models under a unified dataset. We have found that our model using deep learning can detect and recognize faces very well. After completing face recognition, our main contributions are summarized as follows:

We have found that the proportion of a face affects the accuracy. When a person is closer to the camera, the face becomes larger, the confidence of the program tends to higher and the recognition result is good; when the person is far away from the camera, the confidence gradually decreases; the system cannot detect it until the face is small. Therefore, we find that our model has certain limitations for detecting small objects.

In the comparison of different model frameworks, our model performs better than SSD Inception (v2) which has greater usability in surveillance. Our model has the highest accuracy of 90.18% for different face recognition projects, and each class is well recognized. The recognition distance of SSD Inception (v2) is very short, and the requirements for face data sets are relatively high, the degree of completion under the same data set is not as good as our model.

6.2 Future Work

Our project has a lot of expansion in future, including the following aspects,

(1) Although our model has well validation results, but this may not apply to any kind of training data set. In future, we will optimize our model based on the existing results of this project so that the model can be applied to different data sets.

(2) We can use the same training dataset to test more models, such as FaceNet or MobileNet. This may also include models under different platforms, not limited to

TensorFlow, so as to better select a model that is more stable and suitable for face recognition.

(3) We may add more values which are not only limited to face recognition, but also include the recognition of genders and facial expressions. Of course, this requires us to innovate our datasets and models.

(4) We are trying to use other neural networks to achieve face recognition, not limited to use convolutional neural networks (CNNs) only.

(5) We may use windows of spatiotemporal data instead of feature vector to better express neuron parameters, and we may replace TensorFlow in NeuCube as the model framework in the future.

References

- Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Kudlur, M. (2016). TensorFlow: A System for Large-Scale Machine Learning. In *OSDI* (Vol. 16, pp. 265-283).
- Akselrod-Ballin, A., Karlinsky, L., Alpert, S., Hasoul, S., Ben-Ari, R., & Barkan, E. (2016). A region based convolutional network for tumor detection and classification in breast mammography. In *Deep Learning and Data Labeling for Medical Applications* (pp. 197-205). Springer, Cham.
- Al-masni, M. A., Al-antari, M. A., Park, J. M., Gi, G., Kim, T. Y., Rivera, P., ... & Kim, T. S. (2018). Simultaneous detection and classification of breast masses in digital mammograms via a deep learning YOLO-based CAD system. *Computer methods and programs in biomedicine*, 157, 85-94.
- Ayouche, S., Aboulaich, R., & Ellaia, R. (2017). Partnership credit scoring classification Problem: A neural network approach. *International Journal of Applied Engineering Research*, 12(5), 693-704.
- Bao, S., & Chung, A. C. (2018). Multi-scale structured CNN with label consistency for brain MR image segmentation. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, 6(1), 113-117.
- Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on PAMI*, 35(8), 1798-1828.
- Bergstra, J., Bastien, F., Breuleux, O., Lamblin, P., Pascanu, R., Delalleau, O., ... & Bengio, Y. (2011). Theano: Deep learning on GPUs with Python. In *NIPS 2011, BigLearning Workshop, Granada, Spain* (Vol. 3).

- Bressloff, P. C., Ermentrout, B., Faugeras, O., & Thomas, P. J. (2016). Stochastic Network Models in Neuroscience: A Festschrift for Jack Cowan. *The Journal of Mathematical Neuroscience*, 1(6), 1-9.
- Cai, Z., Fan, Q., Feris, R. S., & Vasconcelos, N. (2016). A unified multi-scale deep convolutional neural network for fast object detection. In European Conference on Computer Vision (pp. 354-370). Springer, Cham.
- Cao, C., Liu, X., Yang, Y., Yu, Y., Wang, J., Wang, Z., ... & Ramanan, D. (2015). Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks. In *IEEE International Conference on Computer Vision* (pp. 2956-2964).
- Cao, G., Xie, X., Yang, W., Liao, Q., Shi, G., & Wu, J. (2018). Feature-fused SSD: fast detection for small objects. In *Ninth International Conference on Graphic and Image Processing (ICGIP 2017)* (Vol. 10615, p. 106151E). International Society for Optics and Photonics.
- Chen, X. W., & Lin, X. (2014). Big data deep learning: challenges and perspectives. *IEEE Access*, 2, 514-525.
- Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2018). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4), 834-848.
- Chin, T. W., Yu, C. L., Halpern, M., Genc, H., Tsao, S. L., & Reddi, V. J. (2018). Domain-Specific approximation for object detection. *IEEE Micro*, 38(1), 31-40.

- Chu, X., Ouyang, W., Li, H., & Wang, X. (2016). Structured feature learning for pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4715-4723).
- Cireşan, D. C., Meier, U., Gambardella, L. M., & Schmidhuber, J. (2010). Deep, big, simple neural nets for handwritten digit recognition. *Neural Computation*, 22(12), 3207-3220.
- Clark, T. E. (2004). Can out-of-sample forecast comparisons help prevent overfitting. *Journal of Forecasting*, 23(2), 115-139.
- Cong, J., & Xiao, B. (2014). Minimizing computation in convolutional neural networks. In *International Conference on Artificial Neural Networks* (pp. 281-290). Springer, Cham.
- Cui, W., & Yan, W. Q. (2016). A scheme for face recognition in complex environments. *International Journal of Digital Crime and Forensics (IJDCF)*, 8(1), 26-36.
- Dai, J., Li, Y., He, K., & Sun, J. (2016). R-FCN: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems* (pp. 379-387).
- De Boer, P. T., Kroese, D. P., Mannor, S., & Rubinstein, R. Y. (2005). A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1), 19-67.
- Dewangan, R., & Verma, S. (2016). A review on the comparison of global features-based techniques LDA, PCA, And LBP Algorithm for face recognition. *i-manager's Journal on Pattern Recognition*, 3(3), 32.

- Ding, J., Chen, B., Liu, H., & Huang, M. (2016). Convolutional neural network with data augmentation for SAR target recognition. *IEEE Geoscience and Remote Sensing Letters*, 13(3), 364-368.
- Dunne, R. A., & Campbell, N. A. (1997). On the pairing of the Softmax activation and cross-entropy penalty functions and the derivation of the Softmax activation function. In *Aust. Conf. on the Neural Networks, Melbourne* (Vol. 181, p. 185).
- Erhan, D., Szegedy, C., Toshev, A., & Anguelov, D. (2014). Scalable object detection using deep neural networks. In *the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2147-2154).
- Flores, E., Barrón-Cedeño, A., Rosso, P., & Moreno, L. (2011). Towards the detection of cross-language source code reuse. In *International Conference on Application of Natural Language to Information Systems* (pp. 250-253). Springer, Berlin, Heidelberg.
- Fukushima, K., & Miyake, S. (1982). Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets* (pp. 267-285). Springer, Berlin, Heidelberg.
- Gal, Y., & Ghahramani, Z. (2016). A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems* (pp. 1019-1027).
- Gao, Q., Huang, Y., Gao, X., Shen, W., & Zhang, H. (2015). A novel semi-supervised learning for face recognition. *Neurocomputing*, 152, 69-76.

- Ghiassi, M., Saidane, H., & Zimbra, D. K. (2005). A dynamic artificial neural network model for forecasting time series events. *International Journal of Forecasting*, 21(2), 341-362.
- Girshick, R. (2015, December). Fast R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*, (pp. 1440-1448). IEEE.
- Gkioxari, G., Girshick, R., & Malik, J. (2015). Contextual action recognition with R-CNN. In *IEEE ICCV* (pp. 1080-1088).
- Gu, D., Nguyen, M., Yan, W. Cross models for twin recognition. *International Journal of Digital Crime and Forensics*, 8 (4), 26-36.
- Gu, D., Nguyen, M., & Yan, W. (2016). Cross models for twin recognition. *International Journal of Digital Crime and Forensics (IJDCF)*, 8(4), 26-36.
- Gu, Q., Yang, J., Kong, L., Yan, W. Q., & Klette, R. (2017). Embedded and real-time vehicle detection system for challenging on-road scenes. *Optical Engineering*, 56(6), 063102.
- Gu, Q., Yang, J., Yan, W. Q., Li, Y., & Klette, R. (2017). Local Fast R-CNN flow for object-centric event recognition in complex traffic scenes. In *Pacific-Rim Symposium on Image and Video Technology* (pp. 439-452). Springer, Cham.
- Gu, Q., Yang, J., Yan, W. Q., & Klette, R. (2017). Integrated multi-scale event verification in an augmented foreground motion space. In *Pacific-Rim Symposium on Image and Video Technology* (pp. 488-500). Springer, Cham.
- Hager, G. D., Dewan, M., & Stewart, C. V. (2004). Multiple kernel tracking with SSD. In *CVPR 2004* (Vol. 1, pp. I-I). IEEE.

- He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017, October). Mask R-CNN. In *ICCV* (pp. 2980-2988). IEEE.
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *IEEE CVPR* (pp. 770-778).
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Identity mappings in deep residual networks. In *European Conference on Computer Vision* (pp. 630-645). Springer, Cham.
- Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*(7), 1527-1554.
- Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the Knowledge in a Neural Network. *stat*, *1050*, 9.
- Hitchcock, C., & Sober, E. (2004). Prediction versus accommodation and the risk of overfitting. *The British Journal for the Philosophy of Science*, *55*(1), 1-34.
- Hoffer, E., & Ailon, N. (2015). Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition* (pp. 84-92). Springer, Cham.
- Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., ... & Murphy, K. (2017). Speed/accuracy trade-offs for modern convolutional object detectors. In *IEEE CVPR*.
- Jeong, J., Park, H., & Kwak, N. (2017) Enhancement of SSD by concatenating feature maps for object detection. *image*, *1024*, 19.

- Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., ... & Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In *ACM Multimedia* (pp. 675-678). ACM.
- Jiao, Y., Weir, J., & Yan, W. (2011). Flame detection in surveillance. *Journal of Multimedia*, 6(1).
- Jin, X., Xu, C., Feng, J., Wei, Y., Xiong, J., & Yan, S. (2016). Deep learning with S-Shaped rectified linear activation units. In *AAAI* (pp. 1737-1743).
- Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245), 255-260.
- Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., & Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE CVPR* (pp. 1725-1732).
- Koch, M. (2018). Artificial intelligence is becoming natural. *Cell*, 173(3), 531-533.
- Khan, S., & Yong, S. P. (2017). A deep learning architecture for classifying medical images of anatomy object. In *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), 2017* (pp. 1661-1668). IEEE.
- Kirk, D. (2007). NVIDIA CUDA software and GPU parallel computing architecture. In *ISMM* (Vol. 7, pp. 103-104).
- Kivinen, J., & Warmuth, M. K. (1998). Relative loss bounds for multidimensional regression problems. In *Advances in neural information processing systems* (pp. 287-293).

- Kong, T., Sun, F., Yao, A., Liu, H., Lu, M., & Chen, Y. (2017). Ron: Reverse connection with objectiveness prior networks for object detection. In *IEEE CVPR* (Vol. 1, p. 2).
- Kong, T., Yao, A., Chen, Y., & Sun, F. (2016). Hypernet: Towards accurate region proposal generation and joint object detection. In *IEEE CVPR* (pp. 845-853).
- Kriegeskorte, N. (2015). Deep neural networks: a new framework for modeling biological vision and brain information processing. *Annual Review of Vision Science, 1*, 417-446.
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems* (pp. 1097-1105).
- Lawrence, K., Campbell, R., & Skuse, D. (2015). Age, gender, and puberty influence the development of facial emotion recognition. *Frontiers in Psychology, 6*, 761.
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436-444.
- LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation, 1*(4), 541-551.
- LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278-2324.
- LeCun, Y., Jackel, L. D., Bottou, L., Cortes, C., Denker, J. S., Drucker, H., ... & Vapnik, V. (1995). Learning algorithms for classification: A comparison on handwritten

- digit recognition. *Neural networks: the statistical mechanics perspective*, 261, 276.
- Lee, C. Y., Gallagher, P. W., & Tu, Z. (2016). Generalizing pooling functions in convolutional neural networks: mixed, gated, and tree. In *Artificial Intelligence and Statistics* (pp. 464-472).
- Lee, S. J., Chen, T., Yu, L., & Lai, C. H. (2018). Image classification based on the boost convolutional neural network. *IEEE Access*, 6, 12755-12768.
- Liao, M., Shi, B., Bai, X., Wang, X., & Liu, W. (2017). TextBoxes: A Fast Text Detector with a Single Deep Neural Network. In *AAAI* (pp. 4161-4167).
- Liao, S., Hu, Y., Zhu, X., & Li, S. Z. (2015). Person re-identification by local maximal occurrence representation and metric learning. In *IEEE CVPR* (pp. 2197-2206).
- Lin, B. Y., & Chen, C. S. (2015, November). Two parallel deep convolutional neural networks for pedestrian detection. In *IEEE IVCNZ* (pp. 1-6).
- Lin, Y. P., Lin, Y. X., Zhou, Q., & Tang, D. P. (2014). Nafion/Thionine/Platinum Nanowires-Modified Electrode for Electrocatalyzing Glucose. *Journal of Electrochemistry*, 6, 007.
- Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature pyramid networks for object detection. In *CVPR* (Vol. 1, No. 2, p. 4).
- Lin, Z., & Yuan, C. (2016). A very deep sequences learning approach for human action recognition. In *International Conference on Multimedia Modelling* (pp. 256-267). Springer, Cham.

- Li, P., Nguyen, M., & Yan, W. Q. (2018). Rotation correction for license plate recognition. In *4th International Conference on Control, Automation and Robotics (ICCAR)* (pp. 400-404). IEEE.
- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. In *ECCV* (pp. 21-37). Springer, Cham.
- Liu, W., Wen, Y., Yu, Z., & Yang, M. (2016). Large-Margin Softmax Loss for Convolutional Neural Networks. In *ICML* (pp. 507-516).
- Liu, Z., Yan, W. Q., & Yang, M. L. (2018). Image denoising based on a CNN model. In *4th International Conference on Control, Automation and Robotics (ICCAR)* (pp. 389-393). IEEE.
- Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *IEEE CVPR* (pp. 3431-3440).
- Lu, H., Li, Y., Chen, M., Kim, H., & Serikawa, S. (2017). Brain intelligence: go beyond artificial intelligence. *Mobile Networks and Applications*, 1-8.
- Lu, J., Shen, J. Yan, W., Boris, B. An Empirical Study for Human Behaviour Analysis, *International Journal of Digital Crime and Forensics* 9 (3), 11-17
- Maalej, R., Tagougui, N., & Kherallah, M. (2016). Online Arabic handwriting recognition with dropout applied in deep recurrent neural networks. In *Document Analysis Systems (DAS)* (pp. 417-421). IEEE.
- Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML* (Vol. 30, No. 1, p. 3).

- Marreiros, A. C., Daunizeau, J., Kiebel, S. J., & Friston, K. J. (2008). Population dynamics: variance and the sigmoid activation function. *Neuroimage*, 42(1), 147-157.
- McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4), 115-133.
- Molchanov, V. V., Vishnyakov, B. V., Vizilter, Y. V., Vishnyakova, O. V., & Knyaz, V. A. (2017). Pedestrian detection in video surveillance using fully convolutional YOLO neural network. In *Automated Visual Inspection and Machine Vision II* (Vol. 10334, p. 103340Q). International Society for Optics and Photonics.
- Murata, N., Yoshizawa, S., & Amari, S. I. (1994). Network information criterion-determining the number of hidden units for an artificial neural network model. *IEEE Transactions on Neural Networks*, 5(6), 865-872.
- Nguyen, A., Yosinski, J., & Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 427-436).
- Nie, G. H., Zhang, P., Niu, X., Dou, Y., & Xia, F. (2017). Ship Detection Using Transfer Learned Single Shot Multi Box Detector. In *ITM Web of Conferences* (Vol. 12, p. 01006). EDP Sciences.
- Oquab, M., Bottou, L., Laptev, I., & Sivic, J. (2014). Learning and transferring mid-level image representations using convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 1717-1724). IEEE.

- Parkhi, O. M., Vedaldi, A., & Zisserman, A. (2015). Deep Face Recognition. In *BMVC* vol. 1, No. 3, p. 6.
- Parmar, D. N., & Mehta, B. B. (2013). Face Recognition Methods & Applications. *International Journal of Computer Technology and Applications*, 4(1), 84.
- Poirson, P., Ammirato, P., Fu, C. Y., Liu, W., Kosecka, J., & Berg, A. C. (2016). Fast single shot detection and pose estimation. In *International Conference on 3D Vision (3DV)*, (pp. 676-684). IEEE.
- Perez, C. E. (2017). The deep learning AI playbook: strategy for disruptive artificial intelligence (pp. 125-132), Intuition Machine Inc.
- Quang, D., Chen, Y., & Xie, X. (2014). DANN: a deep learning approach for annotating the pathogenicity of genetic variants. *Bioinformatics*, 31(5), 761-763
- Rastegari, M., Ordonez, V., Redmon, J., & Farhadi, A. (2016). Xnor-net: ImageNet classification using binary convolutional neural networks. In *European Conference on Computer Vision* (pp. 525-542). Springer, Cham.
- Rautaray, S. S., & Agrawal, A. (2015). Vision based hand gesture recognition for human computer interaction: a survey. *Artificial Intelligence Review*, 43(1), 1-54.
- Rawlings, A. L., Woodland, J. H., & Crawford, D. L. (2006). Telerobotic surgery for right and sigmoid colectomies: 30 consecutive cases. *Surgical Endoscopy and Other Interventional Techniques*, 20(11), 1713-1718.

- Raza, M. Q., & Khosravi, A. (2015). A review on artificial intelligence-based load demand forecasting techniques for smart grid and buildings. *Renewable and Sustainable Energy Reviews*, 50, 1352-1372.
- Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *IEEE CVPR* (pp. 779-788).
- Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91-99).
- Ren, Y., Nguyen, M., & Yan, W. Q. (2018). Real-Time recognition of series seven New Zealand banknotes. *International Journal of Digital Crime and Forensics (IJDCF)*, 10(3), 50-65.
- Ren, Y., Zhu, C., & Xiao, S. (2018). Object Detection Based on Fast/Faster RCNN Employing Fully Convolutional Architectures. *Mathematical Problems in Engineering*, 2018.
- Satat, G., Tancik, M., Gupta, O., Heshmat, B., & Raskar, R. (2017). Object classification through scattering media with deep learning on time resolved measurement. *Optics Express*, 25(15), 17466-17479.
- Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61, 85-117.
- Serban, I. V., Sordoni, A., Bengio, Y., Courville, A. C., & Pineau, J. (2016). Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models. In *AAAI* (Vol. 16, pp. 3776-3784).

- Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., & Lecun, Y. (2014). Overfeat: Integrated recognition, localization and detection using convolutional networks. In *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*.
- Shatnawi, A., Al-Bdour, G., Al-Qurran, R., & Al-Ayyoub, M. (2018). A comparative study of open source deep learning frameworks. In *International Conference on Information and Communication Systems (ICICS)*, (pp. 72-77). IEEE.
- Shen, D., Chen, X., Nguyen, M., & Yan, W. Q. (2018). Flame detection using deep learning. In *4th International Conference on Control, Automation and Robotics (ICCAR)* (pp. 416-420). IEEE.
- Shi, B., Bai, X., & Belongie, S. (2017). Detecting oriented text in natural images by linking segments. In *CVPR* (Vol. 3).
- Shi, S., Wang, Q., Xu, P., & Chu, X. (2016). Benchmarking the state-of-the-art deep learning software tools. In *Cloud Computing and Big Data (CCBD)*, (pp. 99-104). IEEE.
- Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1), 1929-1958.
- Sivaramakrishnan, R., Antani, S., Candemir, S., Xue, Z., Abuya, J., Kohli, M., ... & Thoma, G. (2018). Comparing deep learning models for population screening using chest radiography. In *Medical Imaging 2018: Computer-Aided Diagnosis* (Vol. 10575, p. 105751E).

- Smistad, E., & Løvstakken, L. (2016). Vessel detection in ultrasound images using deep convolutional neural networks. In *Deep Learning and Data Labelling for Medical Applications* (pp. 30-38).
- Specht, D. F. (1990). Probabilistic neural networks. *Neural networks*, 3(1), 109-118.
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Frank Wang, Y. C. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2016). Inception-v4, inception-ResNet and the impact of residual connections on learning. In *AAAI* (Vol. 4, p. 12).
- Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *IEEE CVPR* (pp. 2818-2826).
- Tang, A., Lu, K., Wang, Y., Huang, J., & Li, H. (2015). A real-time hand posture recognition system using deep neural networks. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(2), 21.
- Taud, H., & Mas, J. F. (2018). Multilayer Perceptron (MLP). In *Geomatic Approaches for Modelling Land Change Scenarios* (pp. 451-455). Springer, Cham.
- Toderici, G., Vincent, D., Johnston, N., Hwang, S. J., Minnen, D., Shor, J., & Covell, M. (2017). Full resolution image compression with recurrent neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (pp. 5435-5443). IEEE.
- Tokui, S., Oono, K., Hido, S., & Clayton, J. (2015). Chainer: a next-generation open source framework for deep learning. In *Proceedings of Workshop on Machine*

Learning Systems (LearningSys) in the Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS) (Vol. 5).

Tri, N. C., Duong, H. N., Van Hoai, T., Van Hoa, T., Nguyen, V. H., Toan, N. T., & Snasel, V. (2017). A novel approach based on deep learning techniques and UAVs to yield assessment of paddy fields. In *International Conference on Knowledge and Systems Engineering (KSE)*, (pp. 257-262). IEEE.

Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., & Fergus, R. (2013). Regularization of neural networks using DropConnect. In *International Conference on Machine Learning* (pp. 1058-1066).

Wang, J., Yan, W., Kankanhalli, M., Jain, R., Reinders, M. (2003) Adaptive monitoring for video surveillance, *IEEE Conference on Information, Communications and Signal Processing*.

Wang, J., Kankanhalli, M., Yan, W., Jain, R. (2003) Experiential sampling for video surveillance. In *ACM SIGMM international workshop on Video surveillance*, 77-86.

Wang, J., & Yan, W. Q. (2016). BP-neural network for plate number recognition. *International Journal of Digital Crime and Forensics (IJDCF)*, 8(3), 34-45.

Wang, L., Ouyang, W., Wang, X., & Lu, H. (2015). Visual tracking with fully convolutional networks. In *IEEE CVPR* (pp. 3119-3127).

Wood, S. N. (2011). Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(1), 3-36.

- Wu, B., Iandola, F., Jin, P. H., & Keutzer, K. (2017). SqueezeNet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object Detection for Autonomous Driving. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 129-137).
- Wu, J., Leng, C., Wang, Y., Hu, Q., & Cheng, J. (2016). Quantized convolutional neural networks for mobile devices. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4820-4828).
- Yadav, N., & Binay, U. (2017). Comparative Study of Object Detection Algorithms. *International Research Journal of Engineering and Technology (IRJET)*, (Vol. 4, pp. 586-591).
- Yan, W. Q. (2017). *Introduction to Intelligent Surveillance: Surveillance Data Capture, Transmission, and Analytics*. Springer.
- Yan, W. Q., & Chambers, J. (2013). An empirical approach for digital currency forensics. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, (pp. 2988-2991).
- Yan, W. Q., Chambers, J., & Garhwal, A. (2015). An empirical approach for currency identification. *Multimedia Tools and Applications*, 74(13), 4723-4733.
- Yan, W., Kankanhalli, M. (2015) Face search in encrypted domain. In *Pacific-Rim Symposium on Image and Video Technology*, 775-790.
- Yan, W., Kankanhalli, M., Wang, J., Reinders, M. (2003) Experiential sampling for monitoring. In *ACM SIGMM workshop on Experiential telepresence*, 70-72.

- Zagoruyko, S., Lerer, A., Lin, T. Y., Pinheiro, P. H., Gross, S., Chintala, S., & Dollar, P. (2016). A MultiPath Network for Object Detection. In *British Machine Vision Conference* (No. EPFL-CONF-224546). BMVA Press.
- Zhang, C., & Woodland, P. C. (2016). DNN speaker adaptation using parameterised sigmoid and ReLU hidden activation functions. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 5300-5304). IEEE.
- Zhang, K., Zhang, D., Jing, C., Li, J., & Yang, L. (2017). Scalable Softmax loss for face verification. In *International Conference on Systems and Informatics (ICSAI)*, (pp. 491-496). IEEE.
- Zhang, L., Lin, L., Liang, X., & He, K. (2016). Is Faster R-CNN doing well for pedestrian detection. In *ECCV* (pp. 443-457). Springer, Cham.
- Zheng, K., Yan, W. Q., & Nand, P. (2018). Video dynamics detection using deep neural networks. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(3), 224-234.
- Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., ... & Torr, P. H. (2015). Conditional random fields as recurrent neural networks. In *IEEE ICCV* (pp. 1529-1537).
- Zhu, Z., Liang, D., Zhang, S., Huang, X., Li, B., & Hu, S. (2016). Traffic-sign detection and classification in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2110-2118).