Empirical Analysis for Earlier Diagnosis of Alzheimer's Disease Using Deep Learning

Shouming Sun

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

2020

School of Engineering, Computer and Mathematical Sciences

Abstract

With the aging of contemporary society, cognitive impairment and dementia in the elderly, mainly Alzheimer's disease (AD), have become increasingly serious. As a multifactor, multistage, and clinical syndrome with concomitant diseases, senile cognitive impairment will take progress to irreversible dementia after clinical symptoms appear, eventually lead to death. Alzheimer's disease is currently irreversible, effective treatments lack in clinical practice. The development of a patient's status will go through several stages, so early diagnosis is essential. Early intervention of Alzheimer's disease can effectively slow down the disease progression while reduce the burden on patients' families and our society.

This thesis introduces a method based on deep learning for early diagnosis and screening AD. The method is to slice a 3D magnetic resonance image of a human brain so as to generate a two-dimensional image, then we use an object detection network Faster R-CNN to detect the atrophy of the hippocampus region of human brain to realize the diagnosis of AD. A new network is modified and optimized based on VGG16 as the basic network of Faster R-CNN to extract feature maps and obtain 100% high-precision detection of AD samples. At the same time, 97.67% of the detected image accuracy is obtained for the validation set.

Keywords: Alzheimer's disease, early diagnosis, deep learning, object detection, Faster R-CNN.

Content

List of Figures

List of Tables

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: [illegible] Date: 16 March 2020

Acknowledgment

First of all, I thank my supervisor Dr. Wei Qi Yan. In the early stage of this thesis project, Dr. Yan patiently discussed with me the direction of the dissertation and gave me a lot of scientific and effective suggestions on experimental methods. During my completion of this thesis, he has paid regular attention to the progress of my thesis and provided great help in the problems encountered in my experiments. In addition, the deep learning courses delivered by Dr. Yan gave me a better understanding of deep learning so that I can finish this thesis.

Secondly, I would like to thank Auckland University of Technology (AUT). AUT is a wonderful university, the courses are well designed. The four courses I completed in the first semester have greatly supported my thesis work. In addition, AUT provides great hardware facilities, especially high-performance computers in the laboratory, which greatly reduces the training time of deep learning models.

Finally, I thank my wife for her support and encouragement during my studies, especially for taking care of my family while I was working for this thesis.

Shouming Sun

Auckland, New Zealand

March 2020

Chapter 1

Introduction

This chapter consists of five parts. First, an overview of the current state of Alzheimer's disease, the necessity and importance of early diagnosis, screening using Deep Learning methods will be introduced. Sections 1.2 *and* 1.3 *will list the main research issues to be discussed in this thesis and make meaningful contributions to the field of deep learning. Section* 1.4 *will explain the significance of this research and its implementation. Finally, the detailed content of this thesis and the core content of each chapter will be introduced in Section* 1.5.

1.1 Background and Motivation

Alzheimer's disease (AD) is a disease usually occurred in old age which is a slowly progressive, genetically complex, deteriorating over time and irreversible neurodegenerative disease of the brain (Hampel, et al., 2011). The early symptoms of this disease are mild memory loss and mild language impairment (Braak & Braak, 1995), the clinical manifestations, in turn, are: Easy to forget, irritability, inability to speak normally, loss of long-term memory, loss of motivation for survival, difficulty in taking care of oneself and abnormal behavior, gradually lose physical functions, eventually leading to death (Billones, Demetria, Hostallero, & Naval, 2016). AD is now the fifth reason of death among older Americans (Association, 2017).

It has been more than 110 years since the first recorded patient with Alzheimer's disease, but humans have not overcome the disease (Association, 2016). Although some medicine can delay the progression of AD, such as N-methyl-D-aspartate receptor antagonists and acetylcholinesterase inhibitors, there is no medicine can effectively reverse this progress (Winslow, Onysko, Stob, & Hazlewood, 2011).

Alzheimer's disease not only brings great harm to the patients themselves, but also makes great pain to the patients' families (Brookmeyer, Johnson, Ziegler-Graham, & Arrighi, 2007). They have to bear the pain of their relatives not being able to communicate with themselves normally, and even forget who they are. In addition, they have to pay a lot of time and energy to take care of patients. According to statistics, in 2016, more than 15 million families and other unpaid caregivers in the United States alone, provided more than 230 billion US dollars and approximately 18.2 billion hours of care for patients with Alzheimer's or other dementia (Association, 2017).

At the same time, the treatment and care of patients with Alzheimer's disease also consume a large amount of social medical resources. In the US, the average medical insurance per capita for patients over 65 years old with AD is three times that of other patients. In addition, the payment of Medicaid is more than 23 times. In 2017, the total payment in the U.S. for over 65 years old AD patients including healthcare, long-term care, and hospice were approximately $ 259 billion (Association, 2017).

Statistics in recent years indicate that AD is a rapidly growing global epidemic, with an estimated 5.5 million AD patients in the United States alone in 2017; as the baby boom generation ages, the number of people with AD is expected to grow to 13.8 million in large part by mid-century. On average, one American develops AD every 66 seconds, the rate is accelerating. By 2050, it is expected to be 33 seconds, with one million new cases of AD each year (Association, 2017). Globally, the number of patients with AD reached 24 million in 2011, which is expected to double every 20 years by 2040 (Reitz, Brayne, & Mayeux, 2011).

Although AD cannot be cured at present, timely and accurately screening and diagnosis are still meaningful, because active treatment can delay the progress of AD, thereby reducing the harm caused by the disease to patients and their families (Winslow et al., 2011). In the past, the interpretation of medical images was performed by doctors, but due to the differences, subjectivity, and fatigue of different doctors, humans have limited interpretation of medical images (Greenspan, Van Ginneken, & Summers, 2016).

In recent years, with the great progress and rapid development of machine learning, especially deep learning, computer vision has also played a significant role in the field of medical imaging (Shen, Wu, & Suk, 2017). Among them, deep learning is becoming more and more widely used in medical image recognition by training models on a given data set to complete specific tasks on new data. The traditional medical image recognition methods are based on multifeatured fusion methods, singular value decomposition and wavelet transform methods have low efficiency in feature extraction and limited information to be mined, and the recognition effect is not ideal (Ge & Shen, 2013). Compared with other traditional medical image recognition methods, deep learning can mine potential nonlinear relationships in medical images, feature extraction is much efficient (Bengio, 2009). In recent years, many scientists have applied deep learning to medical image recognition, the work provides an important basis for further clinical applications (Yan Xu et al., 2014). Disease detection and classification are performed based on a group of sample populations to determine whether a sample is diseased or how badly it is, the lesion recognition is generally based on the identification of a lesion and other parts in a sample's own medical image (Razzak, Naz, & Zaib, 2018).

Convolutional neural networks (CNNs) in the field of deep learning have proven to be powerful tools for various computer vision tasks (Islam & Zhang, 2017). Because deep CNNs can automatically learn intermediate and advanced abstractions that the human eyes cannot recognize from raw data (such as medical images) (Lee et al., 2017). Medical image analysis teams around the world are rapidly entering the field and applying CNN and other deep learning methods to various applications (Greenspan et al., 2016). In addition, the application of convolutional neural networks to pattern classification of digital images has achieved high average accuracy (Liu, Yan, & Yang, 2018). Compared with traditional machine learning methods, deep learning is a revolutionary method (Le Roux & Bengio, 2008). Instead of separating features using classifier, it can learn the parameters of neural networks directly from the image without human involvement (Ji, Liu, Yan, & Klette, 2019).

In this project, a deep learning-based method is implemented to screen and diagnose the images of human brain magnetic resonance imaging (MRI) to reduce the workload of doctors and obtain better diagnostic results (Batchelor et al., 2002). The medical image-based diagnosis task can be regarded as a classification task, which is to classify the processed medical image (Mohamed et al., 2018). In this experiment, 3D MRI images are sliced into two-dimensional images for classification.

With the development of AD, atrophic changes occur in the hippocampus of the human brain, these changes can be measured on MRI (Killiany et al., 2002). Therefore, this experiment uses deep learning methods to observe the hippocampal atrophy, so as to achieve the diagnosis and screening of AD. Faster R-CNN network for object detection is used in this experiment, so that unnecessary information in the image is filtered out, the classification process focused only on the Region of Interest (ROI) which refers to the hippocampus region in this experiment, thereby improved the detection accuracy. A newNetwork-Faster R-CNN network from VGG16 is proposed in this thesis, which improves the precision of AD without significantly increasing the amount of calculation.

1.2 Research Question

Based on the Deep Learning method to diagnose MRI images, according to the research hypothesis, this thesis mainly raises the following research question:

Question:

"Which deep learning algorithms can get a more accurate result for diagnosing Alzheimer's disease?"

The fundamental aim of this thesis is to develop an algorithm based on deep learning to achieve an accurate diagnosis of Alzheimer's disease by using human brain MRI images.

1.3 Contributions

The main purpose of this project is to design a deep learning-based diagnostic method for diagnosing Alzheimer's disease, which could assist doctors to reduce workload while screening the disease for existing brain MRI images. This experiment has achieved the superior results of improving AD diagnosis precision by adjusting and optimizing the existing Faster R-CNN basic network. The experiment is divided into five parts: 1) download the original MRI data from ADNI; 2) preprocessing the original data (slicing, screening, and labelling) to generate data set; 3) pre-training the existing network; 4) optimization and adjustment; 5) evaluating the training model.

After optimization and adjustment, the model trained by the newNetwork-Faster R-CNN network achieves 100% prediction precision for AD samples and reaches a relatively high accuracy of 97.67% for detection.

1.4 Objectives of This Thesis

Although Alzheimer's disease cannot be cured, it can be diagnosed early. If AD can be found early, it will bring much benefits. First of all, early diagnosis of AD can explain the symptoms and signs that patients have already appeared, so that patients can get timely medical care and relevant welfare support (Aël Chetelat & Baron, 2003). Those seeking assistance and getting a diagnosis will be accessible to the treatment, care and related information they need. Second, early diagnosis of AD can improve the quality of treatment (Querbes et al., 2009), because dementia is a syndrome with multiple etiologies, early diagnosis can identify those that can be treated and the factors that are rapidly deteriorating, as well as examine the patient's medications to obtain better quality of treatment. Third, the early diagnosis of AD can delay the clinical course, taking anti-dementia drugs early can effectively delay the decline of patients' cognitive ability (Leifer, 2003). In addition, early diagnosis of AD can avoid or reduce future costs. According to the 2010 Global AD Report in high-income countries, the average social cost of $ 32,865 per person is with dementia per year (de Vugt & Verhey, 2013).

The purpose of this thesis is to implement the early diagnosis of AD and reduce the workload of doctors by using deep learning. At the same time, the existing MRI images can be quickly screened to predict the risk of disease in potential patients in advance.

1.5 The Structure of This Thesis

The structure of this thesis is as follows:

In the second chapter, the previous literature will be reviewed and discussed, including traditional diagnosis methods of AD. In addition, we will also introduce the relevant content of deep learning, including convolutional neural networks, loss functions, optimization algorithms, classic convolutional neural networks, and object detection networks.

In the third chapter, the research methods of this thesis will be introduced, including research design, network design, validation methods, adjustment and optimization methods.

In the fourth chapter, the experimental process and results of this thesis will be stated, including the collection of raw data, data preprocessing, experimental hardware and software environment, model training process, and comparative analysis of experimental results.

In Chapter 5, the experimental results of this thesis will be analyzed and discussed by using the test sample comparison method.

In Chapter 6, the summary of this thesis and plans for our future work will be presented.

Chapter 2

Literature Review

Through a comprehensive review of research issues and a reasonable review of previous research, the focus of this thesis is on the early diagnosis and screening Alzheimer's disease by using object detection networks in deep learning. In this chapter, we will review and summarize the past few Achievements in research on medical images based on deep learning.

2.1 Introduction

Since the German doctor, Alzheimer published the first disease case report under his name in 1906, the diagnostic criteria for Alzheimer's disease have been continuously improving (Selkoe, 2001). From AD which was considered to be an unusual cause of adult dementia for most of the 20th century to the National Institute of Aging-Alzheimer's Association (NIA-AA) 2011 Working Group and the International Working Group (IWG) Guidelines for biomarkers as part of the diagnosis (Karlawish, Jack Jr, Rocca, Snyder, & Carrillo, 2017). With the continuous development and progress of human technology in the fields of computer vision and medical imaging, the use of computers for medical image analysis has become an indispensable tool and technical means for clinical disease diagnosis, medical research and treatment. (Miller & Brown, 2018).

In recent years, deep learning, especially deep convolutional neural network, has rapidly developed into a research hotspot in the field of medical image analysis (Shen et al., 2017). It can automatically extract hidden data that cannot be recognized by the human eye from big data in medical images disease diagnostic characteristics (Ortiz, Munilla, Gorriz, & Ramirez, 2016).

This chapter will introduce general medical images and common diagnostic methods of Alzheimer's disease. The basics of Deep Learning will be introduced later, including the construction of convolutional neural networks, loss functions and optimizers. Finally, the typical convolutional neural networks and object detection networks will be introduced.

2.2 Common Medical Images

2.2.1 Magnetic Resonance Imaging (MRI)

MRI image is measurement based on the principle that the magnetic resonance signals generated by hydrogen nuclei in human tissues, organs and lesions under the influence of external strong magnetic fields are different in size. The information received by the external MRI signal detector is 3D imaged by a computer reconstruction (Ogawa, Lee, Kay, & Tank, 1990). It can provide very clear human soft tissue anatomy and lesion images. Moreover, it can represent accurate images of brain anatomy without using ionizing radiation (Giedd, 2004).

2.2.2 Computer Tomography (CT)

CT uses a precise collimated X-ray beam to scan a section of a certain thickness of a part of the human body, a detector rotating with the radiation beam receives X-rays transmitted through the section. Finally, computers, utilizing X-ray signal, received by the detector, reconstruct a 3D image of the corresponding cross-section of a human body (Ciernik et al., 2003). It has sub-millimeter-level spatial resolution, can provide clear human bone tissue anatomy and lesion images, and has been widely used in a variety of clinical disease examinations and auxiliary diagnosis (Senohradski, Karovic, & Miric, 2001).

2.2.3 X-Ray Image

Medical X-ray imaging is to generate an electronic density measurement image of different tissues and lesions of the human body (Marchesini et al., 2003). X-ray based imaging includes 2D computer radiography, digital X-ray photography, digital subtraction angiography and mammography, 3D spiral computer tomography, etc., has been widely used in orthopedics, lung, breast and cardiovascular diseases, and clinical diseases detection and auxiliary diagnosis, but 2D X-ray images cannot provide 3D information of human tissues and organs and lesions (Shamir et al., 2008).

2.2.4 Ultrasound Imaging

Ultrasound imaging refers to scan human body with an ultrasound beam, receiving and processing reflected signals to obtain images of internal organs (Jensen, Nikolov, Gammelmark, & Pedersen, 2006). In recent years, ultrasound imaging technology has continued being developed, new ultrasound imaging technologies such as 3D color ultrasound, ultrasound holography, intrabody ultrasound imaging, color Doppler imaging, and ultrasound biological microscope have appeared (Fenster, Downey, & Cardinal, 2001).

2.2.5 Positron Emission Tomography (PET) Image

PET uses positron information emitted when a tracer labelled with a radioactive element such as F18 is decaying (Gambhir, 2002). Therefore, PET image is a measure of the radioactivity of the corresponding tracer and can provide tumor biology information on characteristics (e.g., glucose metabolism, hypoxia, proliferation, etc.), the standard intake can be used to clinically determine the benign/malignant tumor (Gambhir, 2002). PET can provide more intuitive and accurate radiobiological and visual biological characteristics than MRI and CT (Duara et al., 1986).

2.2.6 Pathological Image

Pathological images refer to the removal of a size of diseased tissue from a patient, the tissue is made into pathological sections using eosin and hematoxylin staining methods, and then microscopic imaging techniques are used to image microscopic cells and glands (Ruiz et al., 2007). By analyzing pathological images, doctors can explore the cause, pathogenesis, and pathogenesis of the lesions to make pathological diagnosis. (Luo et al., 2017).

2.3 Diagnostic Methods for Alzheimer's Disease

2.3.1 Detecting Changes in The Electrophysiology of Human Brain

Studies have shown that Alzheimer's disease affects electromagnetic activity in the brain. Compared with healthy older people, patients with Alzheimer's have undergone significant changes in their electromagnetic activity (Pazo‐Alvarez, Amenedo, & Cadaveira, 2004). Therefore, in the diagnosis of Alzheimer's diseases, a new technique combining transcranial magnetic stimulation (TMS) and electroencephalogram (EEG) measures is used in disease detection. The advantage of TMS / EGG is that it can directly and non-invasively perturb the human cerebral cortex without the need for subject cooperation (Ferreri et al., 2003).

2.3.2 Non-invasive Techniques to Find Biomarkers of The Disease

Alzheimer's detection methods based on molecular-level biomarkers are currently widely studied. Many biomarkers, such as tau and b-amyloid 42, measured from cerebrospinal fluid (CSF), the fluid surrounding the cerebral cortex, are closely related to the disease. A major challenge with these biomarker detection methods is that collecting samples from CSF is an invasive measurement, limiting their usefulness in early diagnosis. As blood sample testing will not be considered an invasive technique, it will be an excellent source for detecting Alzheimer's disease. The role of metabolites and protein compounds in Alzheimer's disease has been studied from blood samples, thus, this method is also promising (Geekiyanage, Jicha, Nelson, & Chan, 2012).

2.4 Hippocampus

The hippocampus is an important part of the brain of humans and vertebrates (Xu, et al., 2000). It is a part of the limbic system of the brain which is located below the cerebral cortex and plays a role in learning, memory, stress regulation, and spatial navigation(Johnston & Amaral, 2004). The hippocampus is named because it looks like a sea horse, which appears in pairs in all animals with the hippocampus and locates in the left and right brain hemisphere (Pennanen et al., 2004).

In Alzheimer's disease, the hippocampus is the first area to be damaged (Du et al., 2004). With the development of the disease, the hippocampus gradually shrinks, and its clinical manifestations are memory loss and loss of direction perception.

The hippocampus is the only tissue in the brain that can generate neurons. In other words, it has the ability to repair itself, and can make new neurons in the hippocampus.

Studies have shown that the rate of neuron generation in the hippocampus decreases slightly with age. In the brain tissue of Alzheimer's patients, the rate of neuron production in the hippocampus drops sharply (Tejani-Butt, Yang, & Pawlyk, 1995).

2.5 Deep Learning

Deep learning is a subfield of machine learning. Deep learning has more hidden layers, which uses multiple cascaded nonlinear processing units for feature extraction and transformation (Ngiam, et al., 2011). Compared with Machine Learning, deep learning has the following advantages. First, deep learning has a very strong learning ability. From the results of learning, deep learning performs better (Schmidhuber, 2015). Second, deep learning has wide coverage and good adaptability (Carin & Pencina, 2018).

Deep learning neural networks have many layers and wide widths, can theoretically be mapped to arbitrary functions, so they can solve very complex problems (R. Wu, Yan, Shan, Dang, & Sun, 2015). Third, deep learning is data-driven and has great prospects (Y. Shen et al., 2017). Deep learning is highly data-dependent, the relevant experiments show that the larger the amount of data, the better it performs. Some tasks such as image recognition, face recognition, and natural language processing have even surpassed human performance (Yan, Yoshua, & Geoffrey, 2015). Fourth, deep learning has good portability (M. Wu & Chen, 2015). Due to the outstanding performance of deep learning, currently many frameworks can be used to deploy deep learning, such as MATLAT, TensorFlow and Pytorch, etc., and these frameworks are compatible with many platforms, such as Windows, Linux and Mac OS.

2.6 Convolutional Neural Network

A convolutional neural network is a network consisting of at least one convolutional layer and a fully connected layer for network output, including association weights and pooling layers (Kalchbrenner, Grefenstette, & Blunsom, 2014). CNN can effectively reduce the dimensionality of a large amount of data information in an image to obtain a small amount of data while effectively retaining the feature of the image, compared with other deep learning models, convolutional neural networks can have better results in image recognition and classification (Krizhevsky, Sutskever, & Hinton, 2012).

2.6.1 Convolutional Layer

The first layer of CNN after the input layer is usually a convolution layer (Bayar & Stamm, 2016). The image of the input layer can be regarded as an array of pixel values of $W \times H \times C$, where W and H are the width and height of the image, C is the channel of the image. For an RGB image, C =3 usually. The convolution layer will use an $N \times N$ matrix as a filter (usually N is 3, 5 or 7) to perform the convolution operation from the upper left corner of the image, that is, the values in the matrix are multiplied by the corresponding values in the image covered by the filter, then sum up all the products to form the convolution value at the location of the filter.

After that, the filter will perform a convolution operation from left to right and from top to bottom according to the stride (the stride is usually 1), finally obtain a new array as the input of the next layer. Among them, the size of convolution kernel and the number of filters is manually set. During the initialization process, the weight parameters will be randomly generated, they will be continuously optimized in the subsequent training process to achieve the best classification performance.

The convolution process is to use these weights to continuously multiply the RGB values of these pictures so as to extract visual data information (LeCun & Bengio, 1995). The convolutional layer not only extracts the picture information, but also achieves the effect of dimensionality reduction (Huang, Liu, Van Der Maaten, & Weinberger, 2017).

2.6.2 Activation Layer

An activation layer is usually applied after each convolutional layer. It is used to introduce nonlinear features to a system that has just performed a linear computation operation in a convolutional layer (Krupa, Wiest, Shuler, Laubach, & Nicolelis, 2004). Previously, nonlinear equations such as hyperbolic tangent and sigmoidal functions were used, but scientists have found that Rectified Linear Units (ReLU) (Nair & Hinton, 2010) layer performs much better and can improve computing efficiency because neural networks can reduce greatly the training time without significant change of accuracy. It can also help alleviate the problem of gradient disappearance, because the gradient disappears exponentially in the layer, the training speed of the network is very slow (Zeiler et al., 2013). The ReLU layer applies the function $f(x) = max\ (0, x)$ to the inputs. That is, all negative activations become zero. This layer could increase the nonlinear characteristics of the model and the entire neural network, will not affect the receptive field of the convolutional layer (Nair & Hinton, 2010).

2.6.3 Pooling Layer

In general, after the ReLU layers, pooling layer which is also called the down sampling layer may be selected to reduce the computational cost and control overfitting (Giusti, Cireşan, Masci, Gambardella, & Schmidhuber, 2013). In this category, there are also several layers to choose from, such as max pooling, average pooling, and L_2-norm pooling, the most popular of which is max pooling (Giusti et al., 2013). It uses a filter (usually 2×2) and a stride of the same length. It is then applied to the input and the maximum number in each subregion of the output filter convolution calculation (Yu, Wang, Chen, & Wei, 2014).

2.6.4 Dropout Layer

To control the overfitting, the dropout layer is usually applied (Gal, Hron, & Kendall, 2017). It discards a random activation parameter set, that is, sets these activation parameters to 0 in the forward pass so as to ensure that the neural network will not affect the training samples overmatching, which will alleviate overfitting issues (Ba & Frey, 2013).

2.6.5 Fully Connected Layer

After feature extraction from the previous layer, the network will use a Fully Connected Layer to map higher-level activation mappings to the classification of the output layer and generate an *n*-dimensional vector, where *n* is the number of output layer classifications (Xu, Zhang, Gu, & Pan, 2019). This *n*-dimensional vector represents the probability of the detected image in *N* classifications (C.-L. Zhang, Luo, Wei, & Wu, 2017).

2.6.6 Transposed Convolution Layer

The convolution operation can be regarded as a downsampling process, so the transposed convolution on the sampling can be regarded as the inverse process of the convolution operation, which is generally used for upsampling (Su, Sun, Liu, Zhai, & Jing, 2019). It is becoming more common in recently proposed convolutional neural networks, especially in generative adversarial network (GAN) (Ledig et al., 2017), a transposed convolutional layer appears in the up-sampling part of the generator network to restore the reduced dimensionality (Gao, Yuan, Wang, & Ji, 2019).

The transposed convolution is derived as follows. Suppose the size of the convolution operation is 4×4 and the element matrix is

$$X = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ x_5 & x_6 & x_7 & x_8 \\ x_9 & x_{10} & x_{11} & x_{12} \\ x_{13} & x_{14} & x_{15} & x_{16} \end{bmatrix} \tag{2.1}$$

The size of the convolution kernel is 3×3, the element matrix is

$$W = \begin{bmatrix} w_{0,0} & w_{0,1} & w_{0,2} \\ w_{1,0} & w_{1,1} & w_{1,2} \\ w_{2,0} & w_{2,1} & w_{2,2} \end{bmatrix} \tag{2.2}$$

where *strides*=1, *padding*=0, that is *i*=4, *k*=3, *s*=1, *p*=0.

According to the convolution operations, the size of the output image is 2×2.

$$o = \frac{i+2p-k}{s} + 1 \tag{2.3}$$

Expand the input element matrix into a column vector ***X***,

$$X = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & x_{16} \end{bmatrix}^T \tag{2.4}$$

Expand the element matrix of the output image into a column vector ***Y***,

$$Y = \begin{bmatrix} y_1 & y_2 & y_3 & y_4 \end{bmatrix}^T \tag{2.5}$$

For the input element matrix *X* and the output element matrix *Y*, the matrix operation is used to describe this process:

$$Y = CX \tag{2.6}$$

By derivation, the sparse matrix *C* can be gotten:

$$C = \begin{bmatrix} w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 \\ 0 & 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} \end{bmatrix} \tag{2.7}$$

The operation of transposed convolution is to inverse this matrix calculation process, that is, to obtain *X* through *C* and *Y*. According to the size of each matrix, the calculation process can be easily obtained, which is the operation of transposed convolution:

$$X = C^T Y \tag{2.8}$$

However, if it is substituted into the numerical calculation, the operation of transposed convolution only restores the size of the matrix *X*, but it cannot restore the value of each element of *X*.

2.7 Classic Convolutional Neural Networks

2.7.1 AlexNet

In 2012, the large-scale convolutional neural network AlexNet proposed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton won the championship of the ImageNet LSVRC competition, its accuracy rate surpassed the second place (Fairuz, Habaebi, & Elsheikh, 2018). This caused a great sensation in the academic world and opened the era of deep learning. Although many faster and more accurate convolutional neural network structures have subsequently appeared, AlexNet is still worthy of learning and reference as a pioneer. It sets the tone for subsequent CNNs and even other networks such as R-CNN (Muhammad, Ab Nasir, Ibrahim, & Sabri, 2018).

AlexNet contains a total of 5 convolutional layers and 3 fully connected layers (Minhas, Javed, Irtaza, Mahmood, & Joo, 2019). Each convolutional layer contains the activation function ReLU and the Local Response Normalization (LRN) layers, performs downsampling through the max pooling layer (Patino-Saucedo, Rostro-Gonzalez, & Conradt, 2018). In order to reduce overfitting in fully connected layers, AlexNet has adopted a dropout layer which has proven to be very effective (Krizhevsky et al., 2012).

Before AlexNet, the activation function in neural networks generally selected sigmoid (·) or tanh (·). However, AlexNet chose ReLU which made it much faster than traditional neural networks. Experiments have shown that four-layer convolutional nerves were tested on the CIFAR-10 data set when the training set error rate reaches 25%, the speed using the ReLU activation function is six times faster than tanh (Krizhevsky, et al., 2012).

In addition to the ReLU activation function, AlexNet also uses Local Response Normalization (Hong-meng, Di, & Xue-bin, 2017). This normalization operation achieves the form soft lateral inhibition, which is also behavioral inspiration of real neurons. The equation for LRN is

$$b_{x,y}^{i} = a_{x,y}^{i} / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a_{x,y}^{j})^2 \right)^{\beta} \tag{2.9}$$

where $a^{i}{}_{x,y}$ represents the value produced by the number *i* convolution at (*x, y*) and then the result of applying the ReLU activation function, *n* represents several adjacent convolution kernels, *N* represents the total number of convolution kernels in this layer. *k, n, α, β* are hyperparameters, the values are obtained on the experimental validation set.

In order to prevent overfitting, AlexNet uses dropout layer after the first and second fully connected layers, utilizes a 50% probability to make the output of the neuron as 0, which breaks the fixed dependency between the neurons and makes the learned parameters more robust. In addition, it doubles the speed of iterative convergence (Alom et al., 2018).

2.7.2 VGG

The VGG network was proposed by the Visual Geometry Group Oxford in 2014 (Day, Horzinek, Schultz, & Squires, 2016). Compared with the previous state-of-the-art network structure, the error rate has dropped significantly. It achieved the second place in the ILSVRC 2014 competition in classification project and the first place in the positioning project (Dutta & Zisserman, 2019). VGG is developed based on the AlexNet which has done more in-depth research on the depth and width of deep neural networks than AlexNet (Sengupta, Ye, Wang, Liu, & Roy, 2019). It is an industry generally believe that deeper networks have stronger expression capabilities and can complete more complex tasks (Simonyan & Zisserman, 2014).

Compared with AlexNet, VGG has the following improvements: First, VGG removes the LRN layer, because developers found that the role of LRN in deep networks is not obvious, they simply removed it. Second, VGG uses a smaller 3×3 convolution kernel, while AlexNet uses a larger convolution kernel, e.g., 11×11 convolution kernel, VGG has fewer parameters than AlexNet. Third, the pooling kernel used by VGG becomes smaller. The pooling kernel in VGG is 2×2, the stride is 2, and the AlexNet pooling kernel is 3×3, with a step size of 2.

VGG is very scalable, which is generalized when it is migrated to other image data. At the same time, though ResNet and Inception networks have high accuracy and simpler network structures based on feature extractors, VGG is always a good choice (Alippi, Disabato, & Roveri, 2018).

2.7.3 ResNet

After VGG, as the network structure continues to deepen, the gradient will gradually disappear during the backpropagation process, resulting in the inability to effectively adjust the weights of the previous network layer, because the accuracy rate is gradually saturated, and decrease rapidly which is called degradation problem, but this is not caused by overfitting, because the accuracy is higher based on training set than the validation set (Z. Wu, Shen, & Van Den Hengel, 2019). With the emergence of residual networks (ResNets) in 2015, the problems have been alleviated. ResNet can simplify network training and optimize easily which has also achieved higher accuracy while deepening the network structure (He, Zhang, Ren, & Sun, 2016).

ResNet creatively employs a shortcut connection to provide identity mapping (Akiba, Suzuki, & Fukuda, 2017). When backpropagating during training, it passes the gradient of the next layer to the previous layer, thereby solving the problem of gradient vanishing in deep networks. The principle of the residual block is shown in Figure 2.1 (Chen, Xie, Zhang, & Xu, 2017).

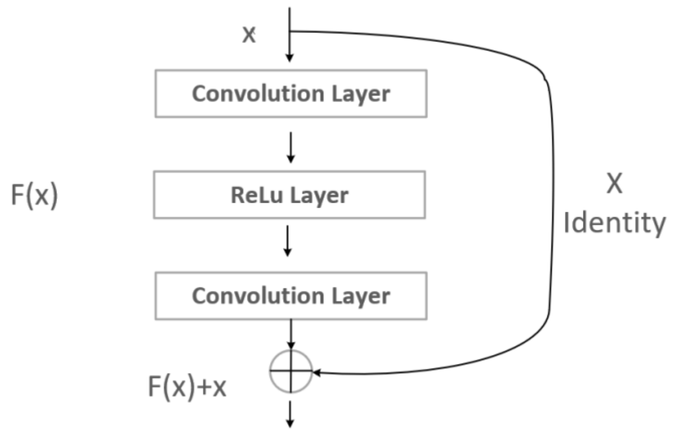


Figure 2. 1 Residual block

It is assumed that the input is x, the output of the convolution layer is $F(x)$, it is added to x as the mapping input, and the resultant output $H(x) = F(x) + x$ is passed to the next layer. This is much easier than matching an identity map through a bunch of nonlinear layers, it will not add extra parameters and calculations to the network. At the same time, it can greatly increase the training speed and improve the training effect of the model when the number of layers deepens, this simple structure can well solve the problem of gradient vanishing (He et al., 2016).

ResNet has two types of residual blocks. The first type is suitable for training shallow networks, if the network is deeper (more than 50 layers), the second type (bottleneck) is recommended. Moreover, the two types have similar time complexity.

2.7.4 DenseNet

The emergence of ResNet allows the network structure to be developed in a deeper direction. The problem is the limitation of memory or video memory, as the network deepens, the parameters that need to be processed are also increasing (Zhu & Newsam, 2017). DenseNet can effectively control the number of parameters while deepening the network. Unlike ResNet, DenseNet is a special type of convolutional neural network with dense connections. In such a network, there is a direct connection between any two network layers, that is, the input of each layer in the network is the output of all previous layers. At the same time, the feature maps extracted from this layer will also be passed as input directly to all subsequent layers. (Zhang, Lu, Li, Kim, & Wang, 2019).

If the transformation function of layer *i* is *H_i* , corresponding to one or two batch-normalization, ReLU, and Convolution operations, the output is *x_i*, then we can use a very simple equation to describe each layer of DenseNet

$$x\_i = H\_i\,([x\_0, x\_1, \ldots, x\_\{i\text{-}1\}]) \quad (2.10)$$

DenseNet differs from other networks in two main ways. First, it allows each layer in the network to be directly connected to its previous layers so as to achieve the reuse of features (Tang, et al., 2019). Second, it designs each layer of the network to be particularly narrow, that is, only very few feature maps are learned (the most extreme case is that each layer only learns one feature map) to achieve the purpose of reducing redundancy. The first point is the premise of the second point, without dense connections, it is impossible to design the network too narrow, otherwise underfitting will occur during training (Gottapu & Dagli, 2018).

DenseNet has the following advantages. First, the parameters of DenseNet are greatly reduced. Experiments have shown that the same accuracy rate is achieved based on the ImageNet data set, DenseNet only requires less than half of the parameters (Huang, et al., 2017). This is of great significance to the industry because small models can significantly reduce storage overhead and save bandwidth. Second, DenseNet requires less computation. When the accuracy equivalent to ResNet is achieved, DenseNet requires only half of the calculation amount of ResNet (Huang, et al., 2017). Nowadays, the requirement for computational efficiency in practical applications of deep learning is very strong, efficient models will be much meaningful to the industry.

Third, DenseNet has very good anti-overfit performance, which is especially suitable for applications with relatively scarce training data. It is well known that insufficient training data sets can easily lead to overfitting. Experiments show that DenseNet reduces the error rate of the best previous results from 28.20% to 19.64%

based on the CIFAR data set without data enhancement (Huang et al., 2017). This achievement can be achieved because the features extracted from each layer of the convolutional neural network can be equivalent to a nonlinear transformation of the input data. As the depth increases, the complexity of the transformation also gradually increases (more complex non-linear functions).

Compared to general neural networks that directly depend on the features of the last layer (the highest complexity) of the network, DenseNet can comprehensively utilize features with low complexity in the shallow layer, it is easier to obtain a smoother model with better generalization performance decision function.

2.8 Loss Function

The loss function is an important concept in deep learning, which reflects the degree to which the model fits the data. In general, the smaller the loss function, the better the model fits the data. At the same time, when the loss function is relatively large, it is expected that the corresponding gradient will also be relatively large so that the update will be faster when the gradient decreases. Besides, the loss function plays an important role in model training. The training process guides the network parameter learning through backpropagating the errors generated by using prediction samples and real sample labels to obtain the optimal model. The loss functions used in classification problems are listed as follows.

2.8.1 Zero-one Loss

Zero-one loss is a relatively simple loss function (Kohavi & Wolpert, 1996). If the predicted value is not equal to the target value, it is 1, otherwise, it is 0, that is

$$\ell(y_i, \hat{y}_i) = \begin{cases} 1 & y_i \neq \hat{y}_i \\ 0 & y_i = \hat{y}_i \end{cases} \tag{2.11}$$

The significance of this loss function is that if the prediction is wrong, the value of the loss function is 1; if the prediction is correct, the value of the loss function is 0. However, the loss function does not consider the degree of errors between the predicted value and the true value, that is, if the prediction is wrong, the prediction error is almost as same as the difference. In addition, the loss function does not contain input information, it cannot be used to backpropagation (Domingos & Pazzani, 1997).

2.8.2 Cross Entropy Loss Function

The labels of real samples in the binary classification problem model are 0 and 1, which represent negative and positive classes, respectively. At the end of the model, a sigmoid function is usually used to output a probability value. This probability reflects the probability that the prediction is positive (Hu, et al., 2018). The expression and graph of the sigmoid function are shown as

$$g(s) = \frac{1}{1+e^{-s}} \tag{2.12}$$

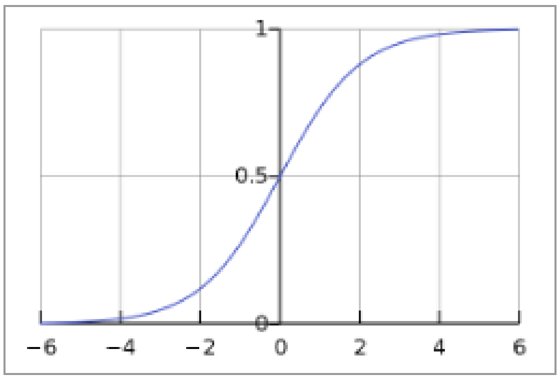


Figure 2. 2 The sigmoid function

where s is the output of the previous layer of the model, the sigmoid function has the following characteristics: If $s = 0$, $g(s) = 0.5$; if $s >> 0$, $g(s) \approx 1$; if $s << 0$, then $g(s) \approx 0$. Obviously, $g(s)$ maps the linear output of the previous stage to a numerical probability between 0 and 1. Here $g(s)$ is the model prediction output in the cross entropy. Figure 2.2 shows a graph of the sigmoid function.

The prediction output (the output of the Sigmoid function) characterizes the probability that the current sample label is

$$\hat{y} = P(y = 1|x) \quad . \tag{2.13}$$

Obviously, the probability, the current sample label is 0, is expressed as

$$1 - \hat{y} = P(y = 0|x) \quad . \tag{2.14}$$

From the perspective of maximum likelihood, the two cases are integrated together

$$P(y|x) = \hat{y}^y \cdot (1 - \hat{y})^{1-y} \quad . \tag{2.15}$$

For a model, it is desirable that the larger the probability *P* (*y* | *x*) the better. The log function is *P (y | x)* because the log operation does not affect the monotonicity of the function itself. There are

$$log\ P(y|x) = log(\hat{y}^y \cdot (1 - \hat{y})^{1-y}) = ylog\ \hat{y} + (1 - y)log(1 - \hat{y}) \tag{2.16}$$

It is expected log *P* (*y* | *x*) to be as large as possible; in other words, its negative value *-log P (y | x)* is as small as possible. Then, we can introduce the loss function *Loss* = *-log P* (*y* | *x*), the expression of this loss function is

$$L = -[ylog\ \hat{y} + (1 - y)log\ (1 - \hat{y})] \quad . \tag{2.17}$$

This is a single-sample loss function, for calculating the total loss function of *N* samples, we superimpose *N* losses, as shown in eq.(2.18), the total loss function is

$$L = -\sum_{i=1}^{N} y^{(i)} log\ \hat{y}^{(i)} + (1 - y^{(i)})log\ (1 - \hat{y}^{(i)}) \quad . \tag{2.18}$$

2.9 Optimizers

2.9.1 Stochastic Gradient Descent

Stochastic gradient descent (SGD) is an iterative method that is applied to optimize differential objective functions. This method iteratively updates the weight and bias terms by calculating the gradient of the loss function based on the mini-batch. SGD surpasses the simple gradient descent method on the highly nonconvex loss surface. This simple mountain climbing technique has dominated modern non-convex optimization.

$$\theta_{\ell+1}=\theta_{\ell}-\alpha\nabla E(\theta_{\ell}) \tag{2.19}$$

where ℓ is the number of iterations, α is the learning rate, θ is the parameter vector, and $E(\theta)$ is the loss function. In the stochastic gradient descent algorithm, only one sample is used to adjust θ for each update. Therefore, stochastic gradient descent will bring further problems, because the calculated result is not an accurate gradient. For the optimization problem, though the loss function obtained by each iteration is not in the direction of global optimization, the direction of the large whole is toward the global optimization solution, and the final result is often near the global optimal solution (Loshchilov & Hutter, 2016).

2.9.2 Stochastic Gradient Descent with Momentum

In practical applications, stochastic gradient descent algorithms often do not find the smallest gradient. It may go beyond the minimum value, and also cause the consequences of too much calculation and long training time. (Murphy, 2012). Therefore, the stochastic gradient descent with momentum (SGDM) is updated to

$$\theta_{\ell+1}=\theta_{\ell}-\alpha\nabla E(\theta_{\ell})+\gamma(\theta_{\ell}-\theta_{\ell-1}) \tag{2.20}$$

where γ refers the contribution of the previous gradient step to the current iteration. In this way, during the training process, if the gradient always declines in a direction, then let the update speed in this direction be faster. If the gradient descent is always swinging in a certain direction, then let the updated speed in this direction be slower.

2.9.3 AdaGrad

Adaptive gradient, or AdaGrad is proposed to solve the problems of SGD and SGDM (Klein, Pluim, Staring, & Viergever, 2009). It can adjust different learning rates for each different parameter, update frequently changed parameters in smaller steps and sparse parameters are updated in larger steps. It deals with the learning rate component by dividing the learning rate by the square root of *S*, which is the cumulative sum of the current and past squared gradients (Duchi, Hazan, & Singer, 2011)

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{S_t + \epsilon}} \cdot \frac{\partial L}{\partial w_t} \tag{2.21}$$

where

$$S_t = S_{t-1} + \left[\frac{\partial L}{\partial w_t}\right]^2 \tag{2.22}$$

and *S* initialised to 0.

2.9.4 Root Mean Square Propagation

Root mean square propagation (RMSProp) is an adaptive learning rate algorithm. Unlike the SGDM algorithm, which uses a single learning rate, RMSpop uses a moving average of the element-wise squares of the parameter gradients as the learning rate,

$$v_\ell = \beta_2 v_{\ell-1} + (1-\beta_2)[\nabla E(\theta_\ell)]^2 \tag{2.23}$$

where β_2 is the attenuation rate of the average whose values are 0.9, 0.99, and 0.999. In addition, the corresponding average lengths of the square gradients are equal to 1 / (1-β_2), the updates are 10, 100, and 1000, respectively. The RMSProp algorithm employs this moving average to normalize the update of each parameter separately

$$\theta_{\ell+1} = \theta_\ell - \frac{\alpha \nabla E(\theta_\ell)}{\sqrt{v_\ell} + \epsilon} \tag{2.24}$$

where $\varepsilon>0$ is a small constant and is added to avoid division by zero. Using RMSProp algorithm can effectively change the learning rate of training to eliminate swing in gradient descent.

2.9.5 Adam

Adam is another adaptive learning algorithm with rate optimization similar to RMSProp. It adds a momentum term when the parameters are updated and retain the element-wise moving average of the squared values and the parameter gradients,

$$\begin{aligned} m_\ell &= \beta_1 m_{\ell-1} + (1-\beta_1)\nabla E(\theta_\ell) \\ v_\ell &= \beta_2 v_{\ell-1} + (1-\beta_2)[\nabla E(\theta_\ell)]^2 \end{aligned} \tag{2.25}$$

Adam updates the network parameters by employing the moving averages as

$$\theta_{\ell+1} = \theta_\ell - \frac{\alpha m_l}{\sqrt{v_l} + \epsilon} \tag{2.26}$$

where ε >0 is a small constant and is added to avoid division by zero. We see from the expression that the calculation of the updated step size can be adaptively adjusted from the two angles of gradient mean and gradient square, instead of being directly determined by the current gradient (Kingma & Ba, 2014).

2.10 Region of Interest

There are pretty rich information in an image, but computers cannot distinguish which information is important. The region that contains important information and needs to be processed is outlined in machine vision and image processing; in the form of boxes, circles, ellipses, irregular polygons, etc. from the processed images are called Region of Interest (RoI). Using ROI to define the target may reduce calculations and increase accuracy (Poldrack, 2007).

2.11 Object Detection

Object detection is an important field in computer vision. Unlike the classification task, which only cares about the entire image, object detection focuses on specific object targets and obtains location and classification information of the target object at the same time. Classic detection models can be divided into two-stage detection models (e.g., R-CNN, Fast R-CNN, and Faster R-CNN) and one-stage detection models (e.g., YOLO and SSD).

2.11.1 Regions with CNN Features (R-CNN)

R-CNN algorithm was firstly proposed in 2014, which is the pioneering work of the two-stage detection algorithm. In this algorithm, CNN can be used to locate and segment objects based on regions, when supervised training samples are sparse, pretrained models on additional data can achieve good results after fine-tuning operations (Girshick, Donahue, Darrell, & Malik, 2014).

R-CNN splits object detection into two procedures. One is to deal with regions that may include objects based on the image (that is, the local cropping of the image, known as the Region Proposal). The second is to run on these proposed regions with the best performance classification network (AlexNet) to get the category of object in each area.

In R-CNN algorithm, IoU is used to evaluate the accuracy of region proposal and ground truth. IoU calculates the ratio of the area where the two regions intersect with their sum to describe the coincidence of the two regions. The choice of IoU threshold has a significant impact on the results. When IoU is greater than 0.5, the region proposal is considered as a positive sample, when IoU is less than 0.1, it is thought as a background class. Proposals between these two thresholds, namely, hard negative will be ignored because it encompasses both positive sample and background information.

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union} \tag{2.27}$$

R-CNN algorithm will continuously be used to adjust the Region Proposal to match the Ground Truth which is called bounding-box regression during the training process. As a result, the repeated calculations through the three training models (Proposal, Classification, Regression) will lead to evaluation problems. Nevertheless, R-CNN still widely influences the depth model in detection tasks, and many subsequent algorithms are also aimed at improving this work.

2.11.2 Fast R-CNN

The reason why R-CNN is time-consuming is that CNN is performed separately on each Proposal, without using shared computing (Girshick, 2015). Fast R-CNN adopts basic networks to run the calculation on the entire image, and transfers it to the R-CNN sub-network, thereby shares most of the calculations and effectively reduces the training time (Wang, Shrivastava, & Gupta, 2017).

Fast R-CNN algorithm uses feature extractor to obtain a feature map from the input image and employs the selective search algorithm to map RoI to the feature map. Then RoI pooling is used to perform a pooling operation based on each RoI to obtain feature vectors of equal length. These feature vectors are sorted into positive and negative samples, and then batched into a parallel R-CNN sub-network, which performs classification and regression calculations at the same time (Girshick, 2015). Fast R-CNN algorithm combines proposal, feature extractor, object classification, and location into a unified overall structure, and improves feature utilization efficiency through shared convolution calculations (Ullah, Xie, Farooq, & Sun, 2018).

2.11.3 Faster R-CNN

Faster R-CNN is the foundation work of the two-stage detection method. It proposes to use the regional proposal networks (RPN) network to replace the selective search algorithm so that the detection task can be finished end-to-end by using the neural network. Due to the characteristics of convolution calculation shared with RCNN, the amount of calculation introduced by RPN is very small. In this way, Faster R-CNN achieves high accuracy while ensures the running speed (Ren, He, Girshick, & Sun, 2015).

RPN network models the task of proposal as the problem of binary classification (whether it is an object) (Jiang & Learned-Miller, 2017). The first step is to generate anchor boxes of different sizes and length-to-width ratios on a sliding window, then the positive and negative of these anchor boxes are calibrated according to the set IoU threshold and Ground Truth (GT). Therefore, the input data that is passed into the RPN network is sorted into anchor boxes (coordinates) and whether each anchor box has objects (two-class labels).

RPN network maps each sample into a probability and four coordinates (Zhang, Lin, Liang, & He, 2016). The probability reflects the opportunity that the anchor box has an object and the four coordinates are used to define the position of the object. Finally, the loss of binary classification and coordinate regression are unified and used as the target of training the RPN network. After the RPN network training is completed, the generated region proposal is filtered according to the probability, passed through a similar labelling process, then transferred to the R-CNN sub-network. After that, the multitask loss is applied to combine the two losses for multiclassification and coordinate regression calculation (Ren et al., 2015).

Compared to the two-stage detection model, the single-stage detection model does not have an intermediate region detection. It directly obtains prediction results from pictures and is also known as a region-free method.

2.11.4 You Only Look Once (YOLO)

YOLO is the pioneer of the single-stage detection method, which expresses the detection as a unified, end-to-end regression problem. The advantages of YOLO are fast, relatively few errors, and good generalization performance (Redmon, Divvala, Girshick, & Farhadi, 2016).

The workflow of YOLO is briefly described as follows: First, YOLO will scale and divide the picture into equally divided grids, each grid will be assigned to the sample to be predicted according to the IoU of ground truth. Secondly, the convolutional neural network modified by GoogLeNet is used to calculate each grid, predicts a conditional probability for each category, and generates several boxes based on the grid. Each box predicts five regression values, four values represent its location, and the fifth value represents the probability of the box containing the object and the accuracy of the position (represented by IoU). Finally, YOLO uses non-maximum suppression (NMS) filters to get the final prediction box. The calculated equation is

$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}} \tag{2.28}$$

The loss function used by YOLO is divided into three parts: Coordinate error, object error, and category error. In order to balance the effects of class imbalance and large and small objects, weights are added to the loss function.

YOLO proposed a new idea of single-stage detection, compared with the two-stage detection method, the advantage of speed is obvious, and its real-time characteristics are impressive. However, YOLO itself has shortcomings, such as the coarse meshing and the number of boxes generated by each mesh, which limits the detection of small-scale objects and similar objects.

2.11.5 Single Shot Multibox Detector (SSD)

SSD is another typical single-stage detection algorithm, which differs from YOLO mainly in two aspects. First, SSD uses a multiscale feature map. It utilizes different convolutional segments based on VGG and outputs a feature map to the regressor to improve the detection accuracy of small objects. Second, SSD uses more anchor boxes, which generates boxes of different sizes and length-to-width ratios at each grid point and then predicts the probability based on the box (YOLO bases on the grid).

The SSD detection model is generally composed of the backbone network and detection head. As a feature extractor, the backbone network extracts representations of different sizes and multiple levels of abstraction in the image. The detector learns category and location associations based on these representations and supervision information. The two tasks of category prediction and location regression performed by detection head are often performed in parallel, jointly trained with the loss that constitutes multitasking (Liu et al., 2016).

SSD is an early model of a single-stage detection model, which has an order of magnitude faster than the two-stage model, meanwhile achieves the accuracy of two-stage model, so most subsequent single-stage model work is based on SSD improvements.

2.12 Related Work

A method is published for early diagnosis of Alzheimer's disease based on deep learning (Ji, Liu, Yan, & Klette,2019) . They employed the MRI images of human brain as the original data, segmented the three-dimensional MRI images during the data processing, selected the gray and white matter as the deep learning data set. In the method, eResNet50, eNASNet, and eMobileNe were implemented as basic classifiers and trained by using end-to-end process. After the training process is completed, the output results of the three basic network trainings were ensembled together to improve the classification accuracy. The method classified the three categories of the data set in pairs and achieved the accuracy of 97.65% for AD/MCI and 88.37% for MCI/CN. The approach did not classify all categories in the data set at one time and the preprocessing of the original data was relatively complicated.

Chapter 3

Methodology

This chapter mainly describes the method for diagnosing Alzheimer's disease by using Faster R-CNN in visual object recognition. In addition, this chapter details the evaluation methods depicted in this thesis.

3.1 Research Design

Since the original data processing in the previous approach is relatively complicated, this experiment designed a method for deep learning training without extracting grey matter and white matter after slicing a three-dimensional MRI image.

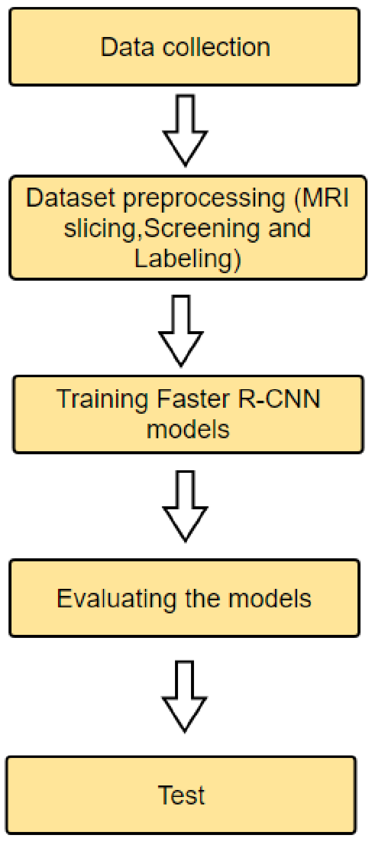

Data collection
Dataset preprocessing (MRI slicing,Screening and Labeling)
Training Faster R-CNN models
Evaluating the models
Test

Figure 3. 1 Our research design process

For a medical diagnosis, accuracy is more important than the speed, because a wrong diagnosis can have serious consequences for patients. The less serious consequence is that an ordinary person is misdiagnosed as sick, which can cause them psychological stress and shortages of medical resources (Schladt, Schneider, Schild, & Tremel, 2011). The serious consequence is to diagnose the sick person as normal, which will delay the patient's treatment. Therefore, the method chosen in this experiment is based on Faster R-CNN on MATLAB platform, though it is very time-consuming for training and test. Figure 3.1 shows the process of this experiment.

For Faster R-CNN network, it needs a traditional CNN network as the basic network for feature extraction. Its structure is as shown in Figure 3.2.

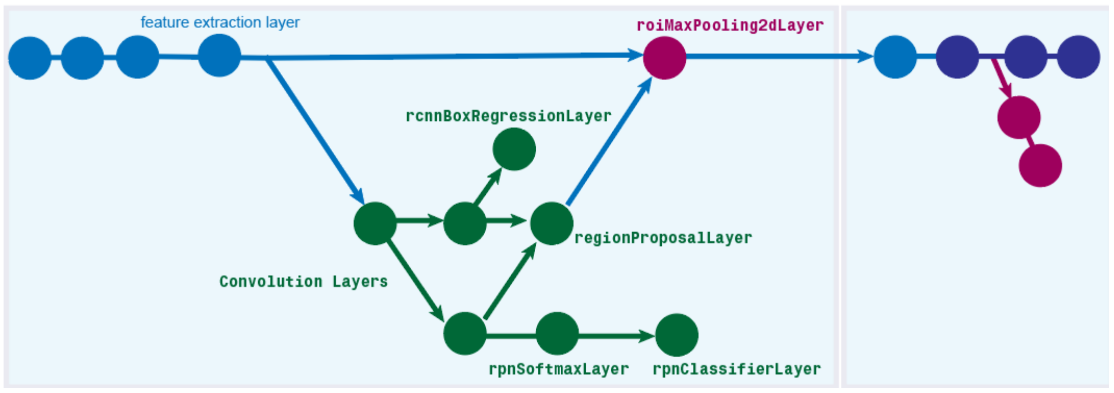


Figure 3. 2 The structure of Faster R-CNN network

We see from Figure 3.2, Faster R-CNN network is mainly divided into four parts according to its functionalities.

First, the feature extraction layer and its previous layers are working for feature extraction in the entire network. The feature maps, extracted through these layers, will be shared with the Region Proposal Network and subsequent fully connected layers.

Second, the Region Proposal Network extracts feature maps based on the feature extraction layer to generate region proposals. It uses softmax layer to determine whether anchors belong to the background or foreground, gives the corresponding labels [-1,0,1], and uses bounding box regression to modify the anchors so as to obtain accuracy proposals. The structure of RPN is shown in Figure 3.3.

After completed the extraction of feature maps and generating region proposals, they are sent to the RoI pooling layer as input. This layer combines the input information to extract the proposal feature maps, and sends the results to the subsequent fully connected layers to determine the target classification. That is, after passing the RPN and RoI Pooling layers, Faster R-CNN network determines that the object classification is based on the characteristics of ROI rather than the characteristics of the entire image. Figure 3.4 shows the input-output relationship of the ROI pooling layer.

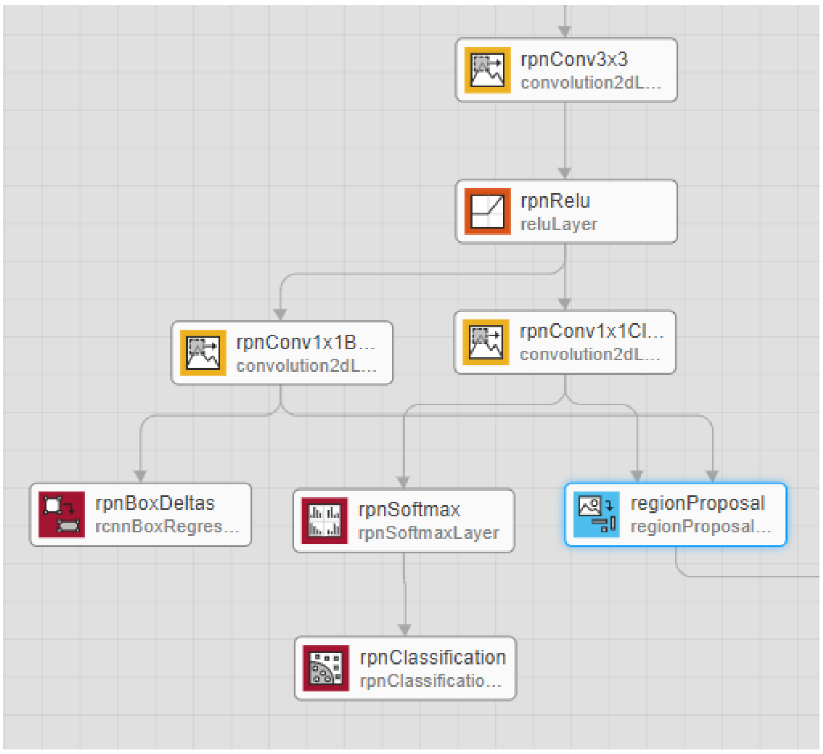


Figure 3. 3 The structure of RPN

The last part is to determine the target classification according to the proposal feature maps and use the bounding box regression once again to obtain the final accurate position of the detection box. The network structure of this part is as shown in Figure 3.5, the left branch is classification, and the right branch is bounding box regression.

The Faster R-CNN network based on the MATLAB platform supports a total of 11 existing network structures as the basic network for feature extraction. The list is shown in Table 3.1.

3.2 Pretraining

In this experiment, 60% of the preprocessed data is selected as the training set, the remaining 40% of the data set is used as the validation set. We use the training set to pretrain the 11 basic networks supported by Faster R-CNN to discover a basic network which is suitable for this data set. During the pretraining process, each network trains only 10 epochs. After the pretraining is completed, the validation method is used to validate the validation set.

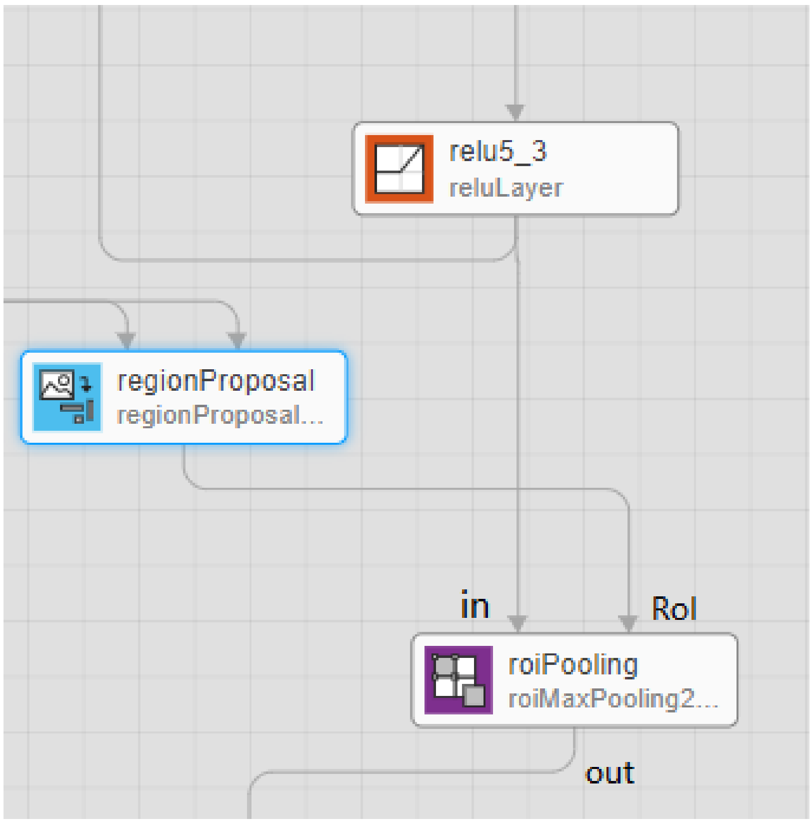


Figure 3. 4 The input-output relationship of the RoI Pooling layer

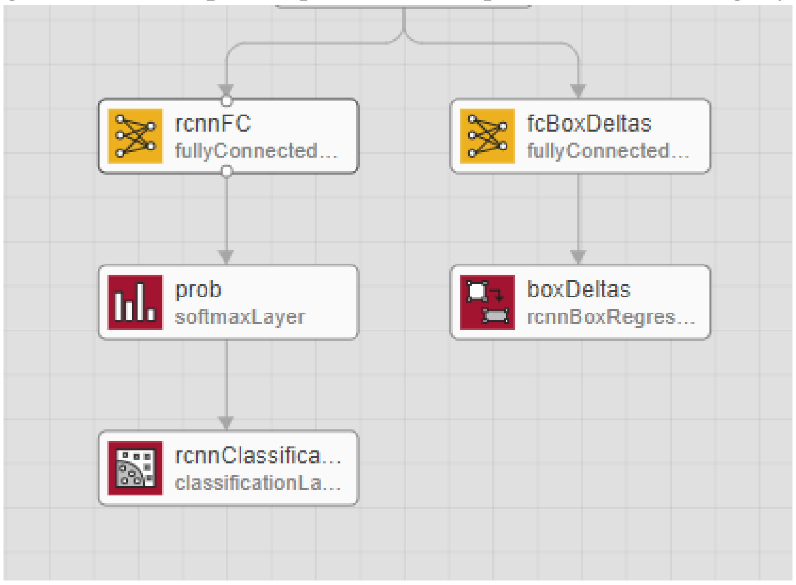
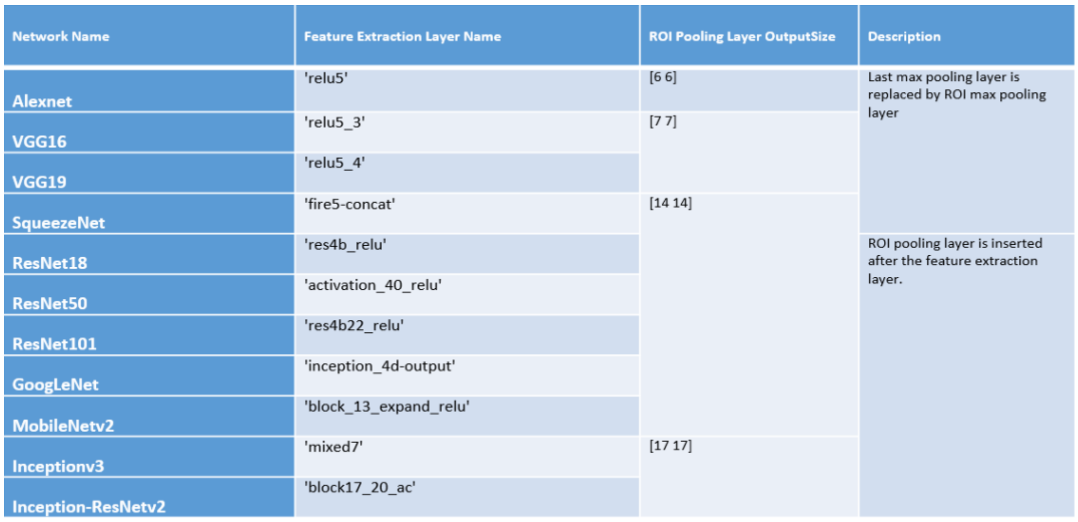


Figure 3. 5 Classification and bounding box regression in Faster R-CNN



Network Name	Feature Extraction Layer Name	ROI Pooling Layer OutputSize	Description
Alexnet	'relu5'	[6 6]	Last max pooling layer is replaced by ROI max pooling layer
VGG16	'relu5_3'	[7 7]	
VGG19	'relu5_4'		
SqueezeNet	'fire5-concat'	[14 14]	
ResNet18	'res4b_relu'		ROI pooling layer is inserted after the feature extraction layer.
ResNet50	'activation_40_relu'		
ResNet101	'res4b22_relu'		
GoogLeNet	'inception_4d-output'		
MobileNetv2	'block_13_expand_relu'		
Inceptionv3	'mixed7'	[17 17]	
Inception-ResNetv2	'block17_20_ac'		

Table 3. 1 Basic networks supported by MATLAB in Faster R-CNN

3.3 Validation Method

Since the process of Faster R-CNN training does not support the validation set, in this experiment, the performance of our training model needs to be evaluated after training. Each image in the validation set will be verified by the trained model. The detection result has three outputs: bboxes, scores, and labels.

The bboxes refers to the position of object (hippocampus region) detected in the input image, with $M\times4$ matrix returns, where M is the number of bounding boxes. Each row of the bbox contains a four-element vector of the form (x , y, *width, height*), where x and y refer to the coordinates of the upper-left pixel of the corresponding bounding box, *width* and *height* stand for the width and height of the bounding box, respectively.

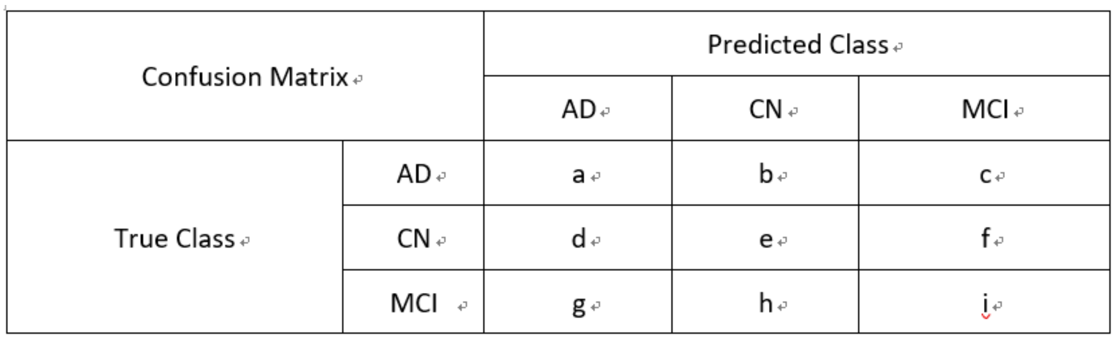
Scores refer to the detection confidence which are returned as $M\times1$ vectors, where M is the number of bounding boxes. The higher this score, the higher the confidence in the detection, the more likely the bounding box contains the detected object (hippocampus region).

Labels refer to the labels of the bounding box which are returned in the form of $M\times1$ classification array of M labels. That means, the classification label of the detected object is the classification of the sample being diagnosed in this experiment.

After the pretraining, due to the performance problems of the model, multiple target objects are detected. In fact, for the validation set of this experiment, only one hippocampal region needs to be detected, so only the one with the highest score of bounding boxes is used to determine the classification among all the detection results.

This experiment uses Faster R-CNN to detect the ROI, the most important factor for validation is the accuracy of classification. Therefore, the validation method of this experiment only evaluates accuracy of the classification, recall and precision of each category. There are three categories in this experiment: Alzheimer’s Disease (AD), Cognitively Normal (CN), and Mild Cognitive Impairment (MCI).

The performance index for evaluating classification tasks is generally the classification accuracy, that is, the ratio of the number of correctly classified samples to the total number of samples for a given data. In this experiment, the detection results of each image in the validation set will be compared with the label, the correct number of all detections will be added and divided by the number of the validation set to obtain the detection accuracy. The confusion matrix is shown in Table 3.2.



Confusion Matrix		Predicted Class		
		AD	CN	MCI
True Class	AD	a	b	c
	CN	d	e	f
	MCI	g	h	i

Table 3. 2 The confusion matrix

The calculation formula for accuracy is shown as

$$Accuracy = \frac{a+e+i}{a+b+c+d+e+f+g+h+i} \quad . \qquad (3.1)$$

Recall refers to the proportion that is predicted to be positive among all positives, recall is a very important indicator in disease diagnosis. For multiclassification, the recall for each category needs to be calculated. The calculations are shown as

$$Recall_{AD}=\frac{a}{a+b+c} \tag{3.2}$$

$$Recall_{CN}=\frac{e}{d+e+f} \tag{3.3}$$

$$Recall_{MCI}=\frac{i}{g+h+i} \tag{3.4}$$

.

Precision refers to the proportion of samples that are truly positive among the samples that are predicted to be positive. For multiclassification, the precision for each class needs to be calculated. The calculations are shown as

$$Precison_{AD}=\frac{a}{a+d+g} \tag{3.5}$$

$$Precison_{CN}=\frac{e}{b+e+h} \tag{3.6}$$

$$Precison_{MCI}=\frac{i}{c+f+i} \tag{3.7}$$

3.4 Tuning and Optimization

After pretraining, the best basic network with good performance will be selected for further tuning and optimization based on the validation results. Tuning and optimization mainly include the following aspects:

First, modifying the basic network of feature extraction improves performance based on the existing data sets. Of course, this may increase the amount of calculations.

Second, adjusting the more accurate RPN method includes reducing the number of proposals to improve accuracy.

Third, improving the classification regression layer includes extracting features and classification by increasing the number of layers.

Finally, adjusting the optimizer, learning rate, and number of epochs achieves the best performance.

In this experiment, Faster R-CNN using VGG16 as the basic network was selected for adjustment and optimization based on the results of pretraining. After the attempts, a transposed convolution layer is added after the relu5_3 layer as the new feature extraction layer. The kernel size is 4 ×4, the number of filters is 512, the stride size is 4 ×4, and the cropping size is 1. The modified part structure diagram of the newNetwork and specific parameters are shown in Figure 3.6 and Figure 3.7. The significance of adding the transposed convolution layer is related to upsampling the feature map extracted through the basic network. At the same time, the parameters in the transposed convolution matrix are continuously optimized with the training process to increase the dimension of the feature map while better retaining the feature information of the image.

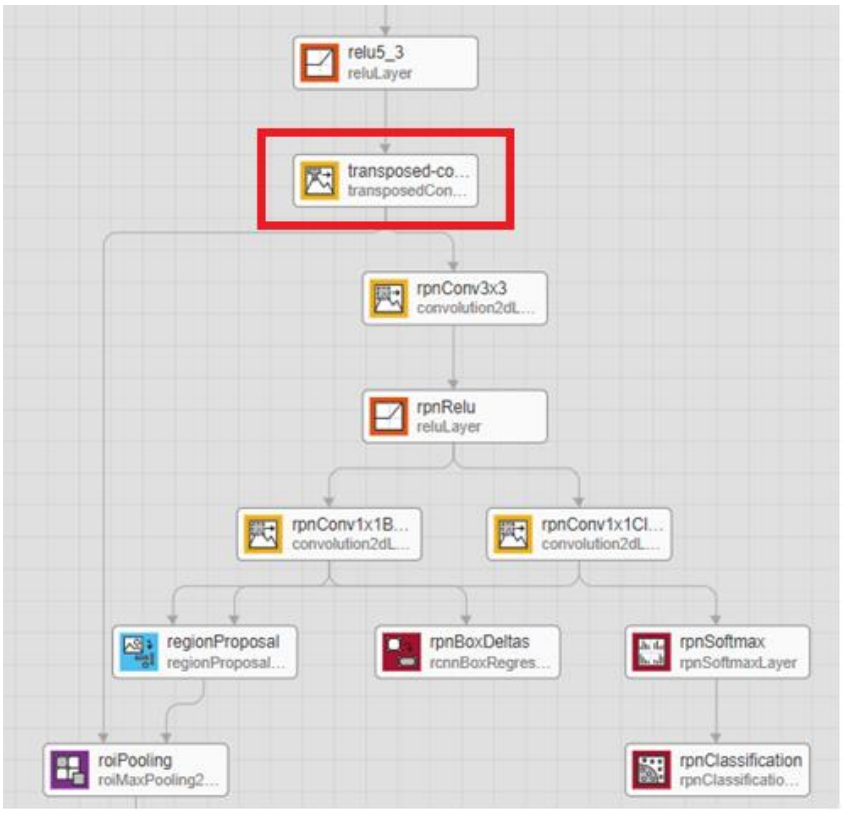
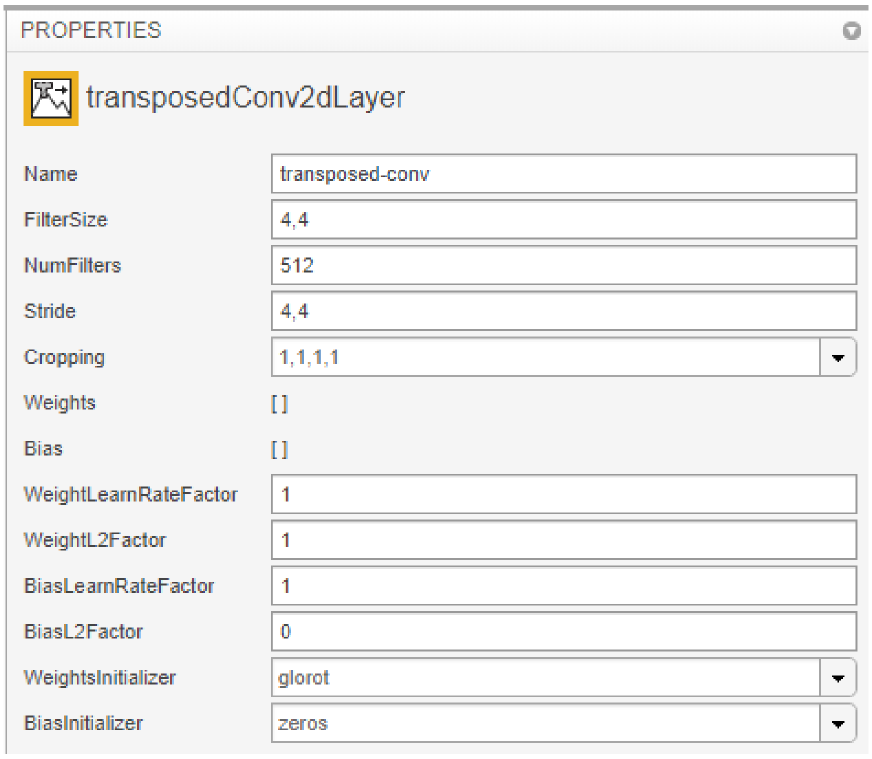


Figure 3. 6 The modified part structure of the newNetwork



PROPERTIES

transposedConv2dLayer

Name	transposed-conv
FilterSize	4,4
NumFilters	512
Stride	4,4
Cropping	1,1,1,1
Weights	[]
Bias	[]
WeightLearnRateFactor	1
WeightL2Factor	1
BiasLearnRateFactor	1
BiasL2Factor	0
WeightsInitializer	glorot
BiasInitializer	zeros

Figure 3. 7 The parameter settings for the transposed convolutional layer

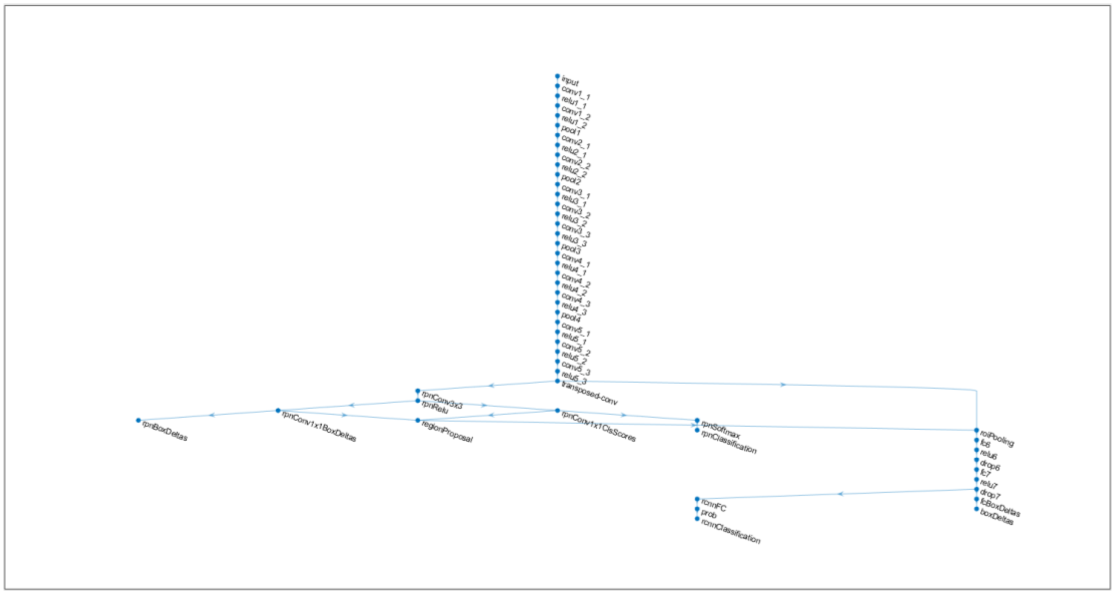
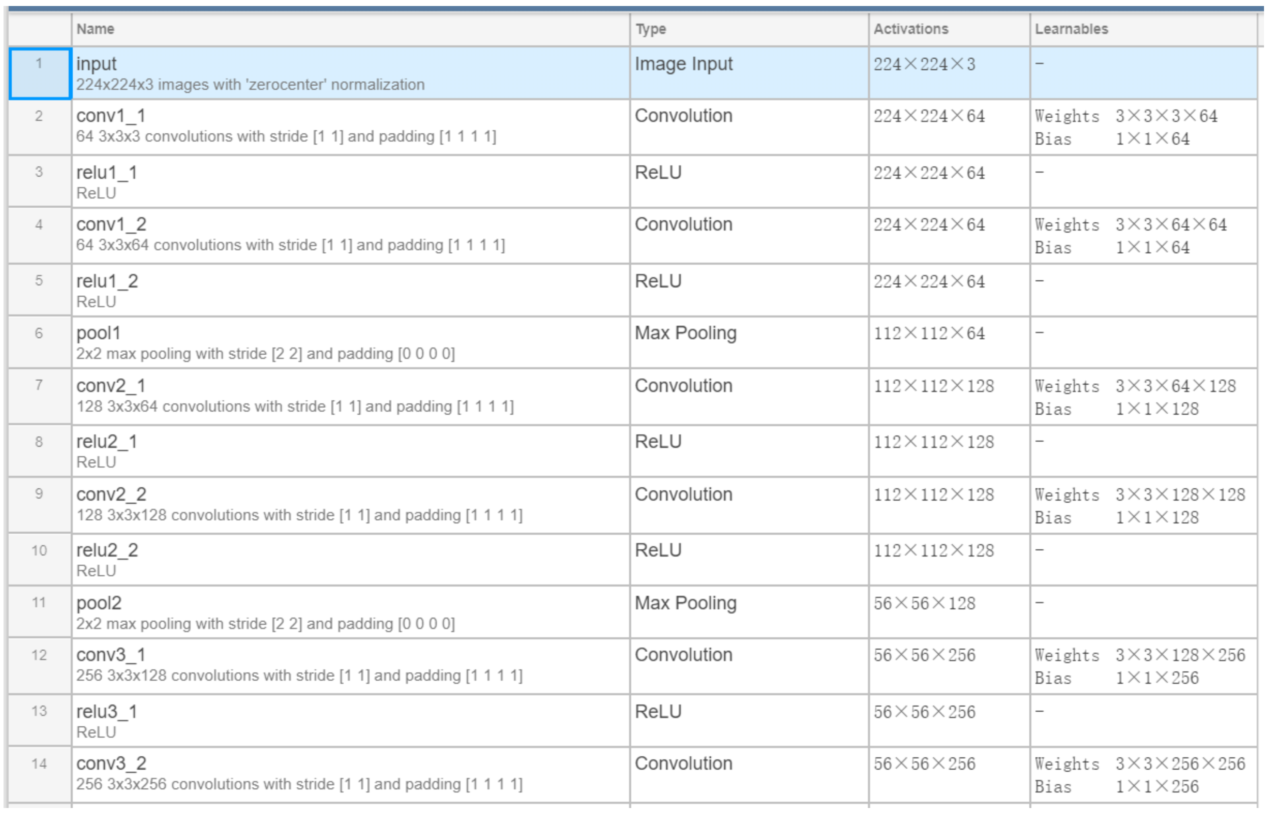
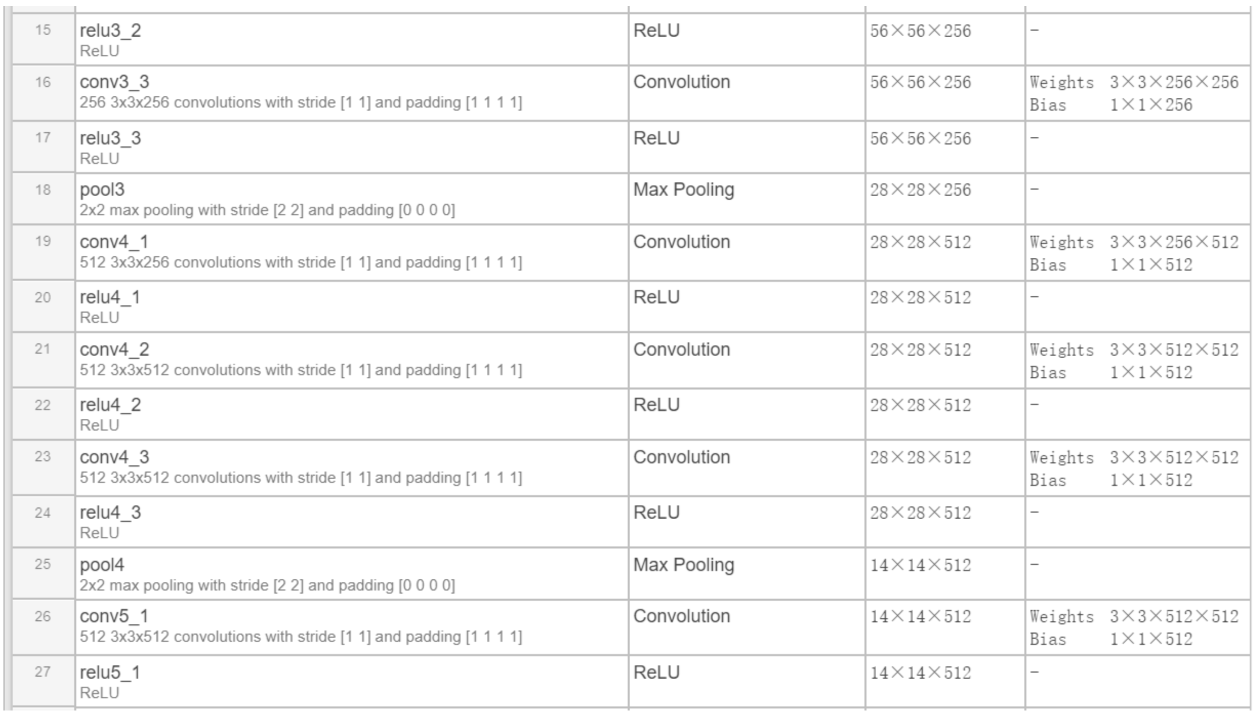
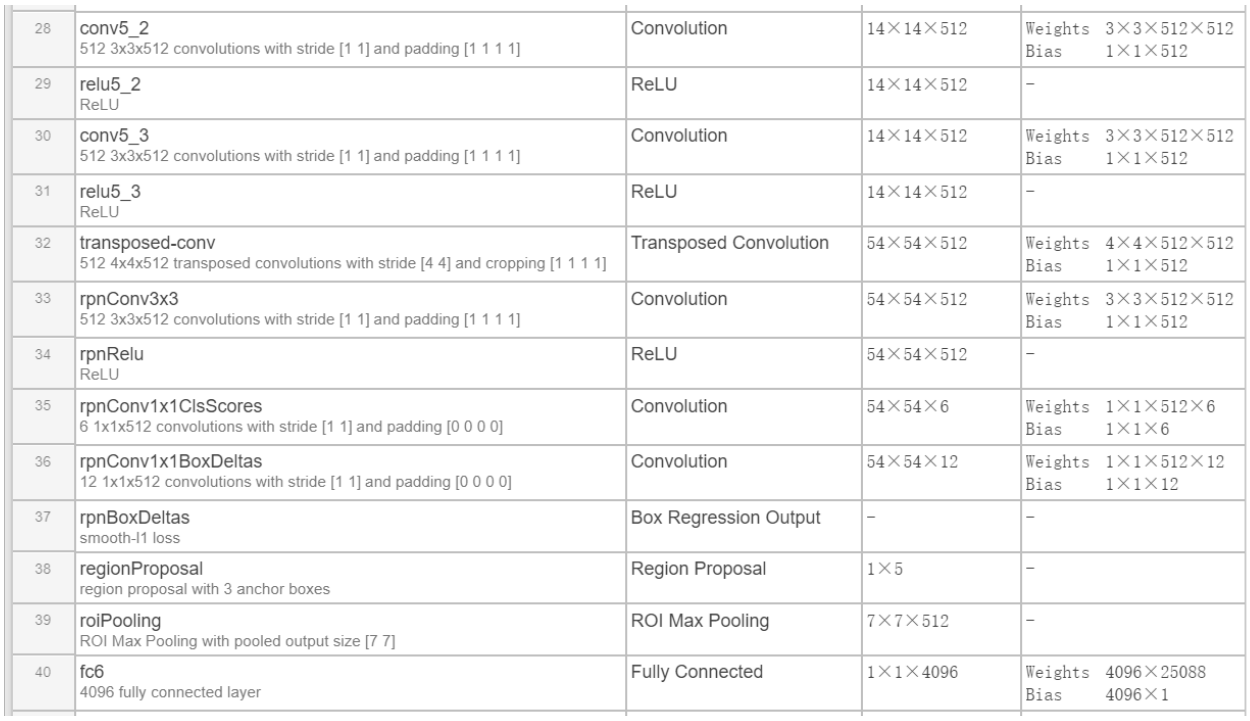


Figure 3. 8 Faster R-CNN network structure based on the newNetwork

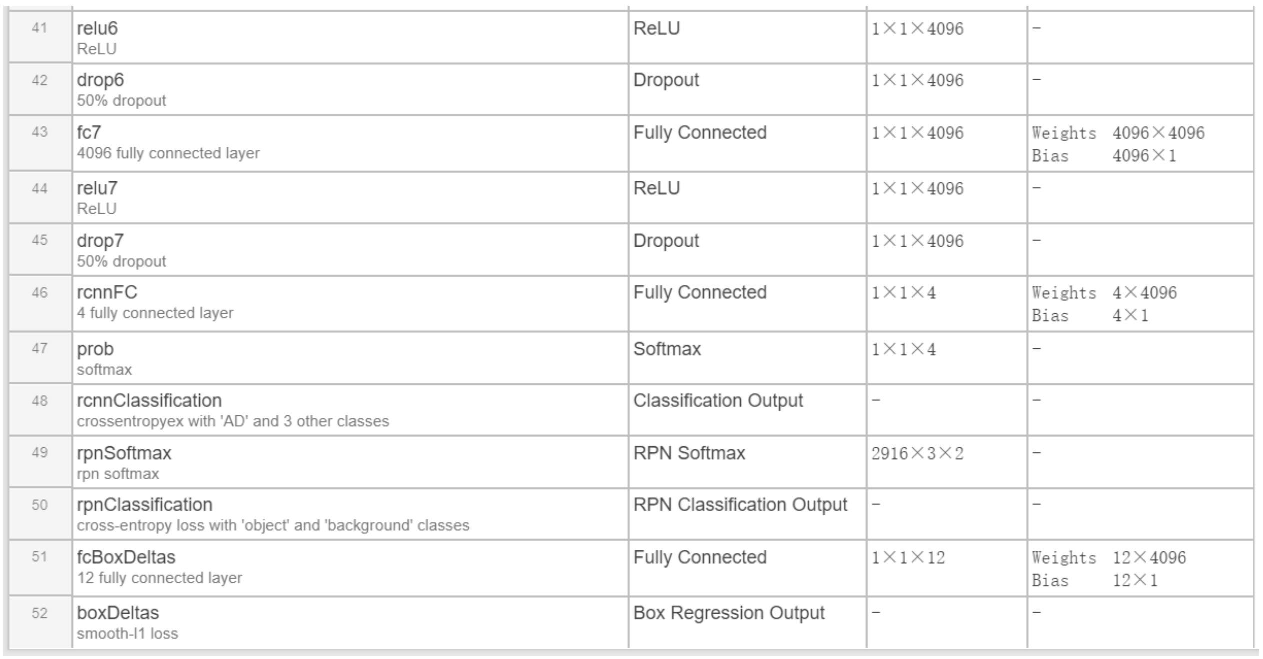
Figure 3.8 shows the network structure of Faster R-CNN based on the newNetwork. Figure 3.9 illustrates the network layer parameters of Faster R-CNN based on the newNetwork.

	Name	Type	Activations	Learnables
1	input 224x224x3 images with 'zerocenter' normalization	Image Input	224×224×3	–
2	conv1_1 64 3x3x3 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	224×224×64	Weights 3×3×3×64 Bias 1×1×64
3	relu1_1 ReLU	ReLU	224×224×64	–
4	conv1_2 64 3x3x64 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	224×224×64	Weights 3×3×64×64 Bias 1×1×64
5	relu1_2 ReLU	ReLU	224×224×64	–
6	pool1 2x2 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	112×112×64	–
7	conv2_1 128 3x3x64 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	112×112×128	Weights 3×3×64×128 Bias 1×1×128
8	relu2_1 ReLU	ReLU	112×112×128	–
9	conv2_2 128 3x3x128 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	112×112×128	Weights 3×3×128×128 Bias 1×1×128
10	relu2_2 ReLU	ReLU	112×112×128	–
11	pool2 2x2 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	56×56×128	–
12	conv3_1 256 3x3x128 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	56×56×256	Weights 3×3×128×256 Bias 1×1×256
13	relu3_1 ReLU	ReLU	56×56×256	–
14	conv3_2 256 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	56×56×256	Weights 3×3×256×256 Bias 1×1×256
15	relu3_2 ReLU	ReLU	56×56×256	–
16	conv3_3 256 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	56×56×256	Weights 3×3×256×256 Bias 1×1×256
17	relu3_3 ReLU	ReLU	56×56×256	–
18	pool3 2x2 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	28×28×256	–
19	conv4_1 512 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	28×28×512	Weights 3×3×256×512 Bias 1×1×512
20	relu4_1 ReLU	ReLU	28×28×512	–
21	conv4_2 512 3x3x512 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	28×28×512	Weights 3×3×512×512 Bias 1×1×512
22	relu4_2 ReLU	ReLU	28×28×512	–
23	conv4_3 512 3x3x512 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	28×28×512	Weights 3×3×512×512 Bias 1×1×512
24	relu4_3 ReLU	ReLU	28×28×512	–
25	pool4 2x2 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	14×14×512	–
26	conv5_1 512 3x3x512 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	14×14×512	Weights 3×3×512×512 Bias 1×1×512
27	relu5_1 ReLU	ReLU	14×14×512	–



28	conv5_2 512 3x3x512 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	14×14×512	Weights 3×3×512×512 Bias 1×1×512
29	relu5_2 ReLU	ReLU	14×14×512	–
30	conv5_3 512 3x3x512 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	14×14×512	Weights 3×3×512×512 Bias 1×1×512
31	relu5_3 ReLU	ReLU	14×14×512	–
32	transposed-conv 512 4x4x512 transposed convolutions with stride [4 4] and cropping [1 1 1 1]	Transposed Convolution	54×54×512	Weights 4×4×512×512 Bias 1×1×512
33	rpnConv3x3 512 3x3x512 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	54×54×512	Weights 3×3×512×512 Bias 1×1×512
34	rpnRelu ReLU	ReLU	54×54×512	–
35	rpnConv1x1ClsScores 6 1x1x512 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	54×54×6	Weights 1×1×512×6 Bias 1×1×6
36	rpnConv1x1BoxDeltas 12 1x1x512 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	54×54×12	Weights 1×1×512×12 Bias 1×1×12
37	rpnBoxDeltas smooth-l1 loss	Box Regression Output	–	–
38	regionProposal region proposal with 3 anchor boxes	Region Proposal	1×5	–
39	roiPooling ROI Max Pooling with pooled output size [7 7]	ROI Max Pooling	7×7×512	–
40	fc6 4096 fully connected layer	Fully Connected	1×1×4096	Weights 4096×25088 Bias 4096×1



41	relu6 ReLU	ReLU	1×1×4096	–
42	drop6 50% dropout	Dropout	1×1×4096	–
43	fc7 4096 fully connected layer	Fully Connected	1×1×4096	Weights 4096×4096 Bias 4096×1
44	relu7 ReLU	ReLU	1×1×4096	–
45	drop7 50% dropout	Dropout	1×1×4096	–
46	rcnnFC 4 fully connected layer	Fully Connected	1×1×4	Weights 4×4096 Bias 4×1
47	prob softmax	Softmax	1×1×4	–
48	rcnnClassification crossentropyex with 'AD' and 3 other classes	Classification Output	–	–
49	rpnSoftmax rpn softmax	RPN Softmax	2916×3×2	–
50	rpnClassification cross-entropy loss with 'object' and 'background' classes	RPN Classification Output	–	–
51	fcBoxDeltas 12 fully connected layer	Fully Connected	1×1×12	Weights 12×4096 Bias 12×1
52	boxDeltas smooth-l1 loss	Box Regression Output	–	–

Figure 3. 9 Faster R-CNN network layer parameters based on the newNetwork

3.5 Test Method

To verify the specific performance of the training model, a sample from each classification outside the data set is sliced, the sliced image is used for the trained model. The test results are output graphically to view whether the test results are correctly classified and whether the hippocampal region is correctly detected. Figure 3.10 shows the process of the test method.

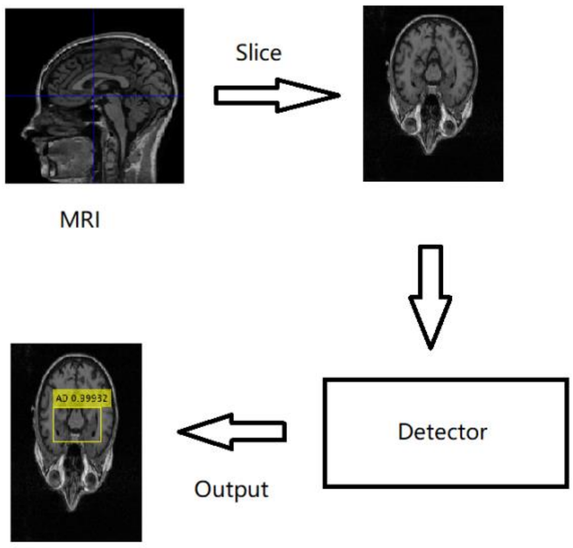


Figure 3. 10 The process of the test method

Chapter 4

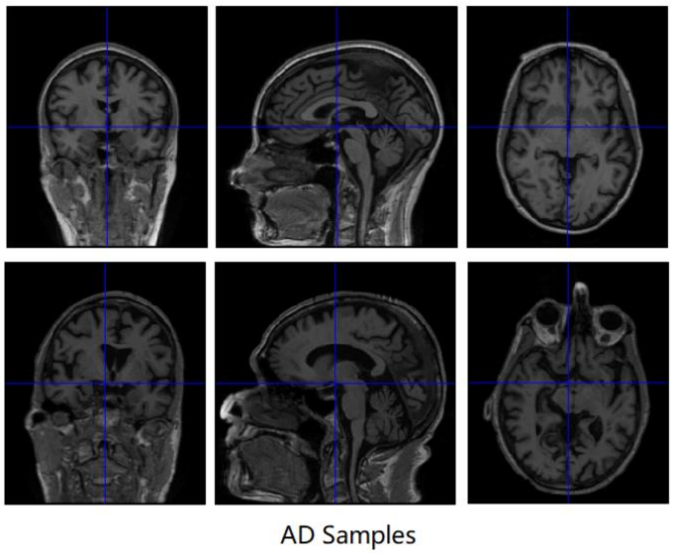
Experiments and Results

The main content of this chapter is to introduce the whole experimental process, including hardware and software environment of the experiment, the setting of experimental parameters, the collection and preprocessing of the data set, the optimization of the experimental model, etc. In addition, a comparison of experimental results is listed at the end of this chapter.

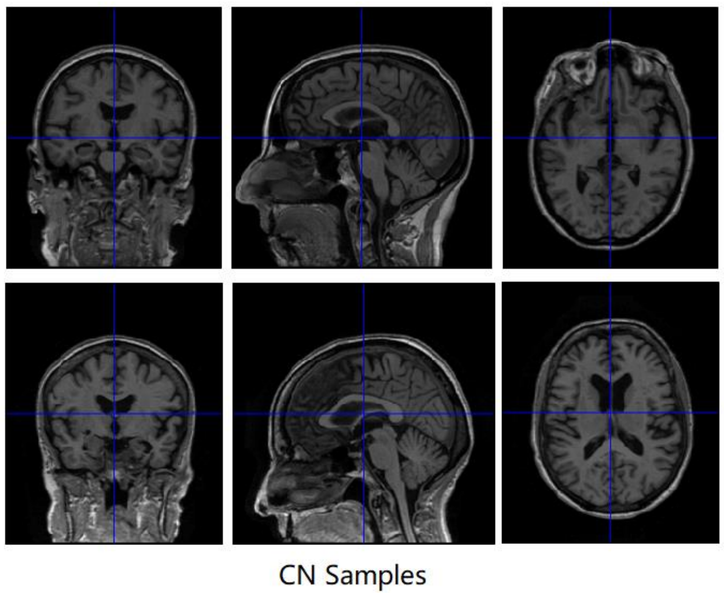
4.1 Data Collection

Alzheimer's Disease Neuroimaging Initiative (ADNI) is linked to experimental data to determine the relationship between cognitive, clinical, biochemical biomarkers, imaging and genetic across Alzheimer's disease, early diagnosis and tracking of this disease. ADNI is committed to detect AD at the earliest possible stage (before dementia) by applying new diagnostic methods at the earliest possible stage.

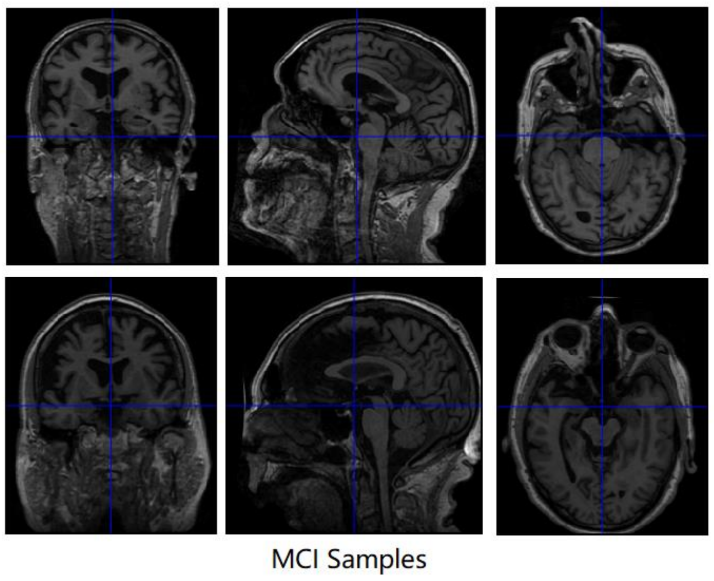
The data set for this experiment is a part of the collection of ADNI 1. All raw data are MRI images related to the head of patients or volunteers. There are three categories in ADNI 1 data set: Alzheimer’s Disease (AD), Cognitively Normal (CN), and Mild Cognitive Impairment (MCI) (Gauthier et al., 2006). Amongst them, AD indicates that patients with Alzheimer disease have been diagnosed, CN indicates normal cognitive status, CN participants are the controls used in the ADNI study and they show no signs of depression, mild cognitive impairment or dementia. MCI indicates mild cognitive impairment patients, MCI participants can basically take care of their daily activities, there is no significant level of damage in other cognitive areas, there are no signs of dementia. The MRI image preview is shown in Figure 4.1.



(a)



(b)



(c)

Figure 4. 1 Examples of MRI

4.2 Data Set Preprocessing

Since the limit capability of current deep learning to deal with 3D images (Mohamed, et al., 2018), such as MRI, the original data needs to be preprocessed. The preprocessing of the data set in this experiment is divided into three parts: MRI slicing, screening, and labelling.

4.2.1 MRI Slicing

MRI imaging uses layer-by-layer scanning to synthesize all layer images into a 3D image, so the inverse operation is performed during the preprocessing of the image data, that is, the MRI image is layered (sliced) along the bottom to the top of the head. In this experiment, a piece of MATLAB code is used for slicing. Each MRI sample is cut into approximately 256 slices, however, only a small part of the slice contains the hippocampus, so only 56 slices of each sample are selected for manual screening. Even if so, there is no guarantee that all the remaining images contain clear information about the hippocampus, because the hippocampus is located at the height of human eyes. For example, the first image in Figure 4.2 cannot find the hippocampus clearly. An example of reserved slices is shown in Figure 4.2.

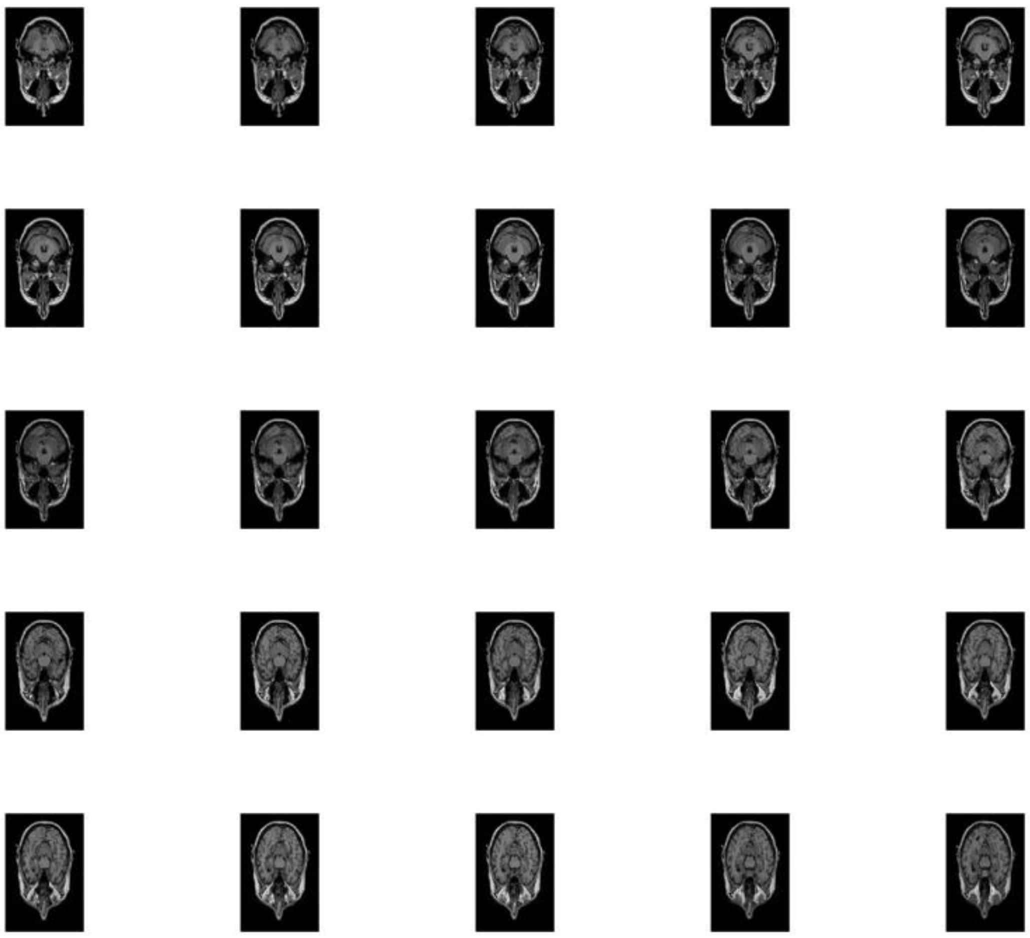


Figure 4. 2 Examples of slice of MRI

4.2.2 Screening

Since the samples show different positions of the scans, even if a part of the slice image is retained when slicing, it cannot guarantee that all the reserved slices contain hippocampus, so that manual screening is required. After screening, 10 pictures containing clear hippocampus are selected from each sample.

In this experiment, 100 samples were selected for each category. After the screening, about 10 slices were selected for each sample, a total of 3343 images were used as the data set.

4.2.3 Labelling

Although the filtered data is stored in different categories, all the images look too similar and some interference factors also affect the accuracy of the diagnosis, such as the shape of the skull, individual differences, etc, thus, simple image classification algorithms cannot achieve the diagnosis of Alzheimer's disease. A simple classification experiment was implemented based on the screened data set, but the accuracy of the diagnosis was unsatisfactory.

Image Labeler is one of the useful tools of MATLAB for data labelling, which can be used to define rectangular ROI labels, scene labels, pixel ROI labels, polyline ROI labels, and use these labels to interactively mark other data. In this experiment, Image Labeler marks up a rectangular ROI which contains the hippocampus and pons of the human brain image. The labelling process is shown in Figure 4.3.

After labelling, the three categories of data sets are merged by using MATLAB to form a data set for training and test. This data set contains a total of 3343×4 samples. Among them, the first column of data represents the path information of each image, the remaining three columns represent the coordinate positions of the three classified ROIs. In addition, each row of data corresponds to the information of each image. An example of the dataset is shown in Figure 4.4.

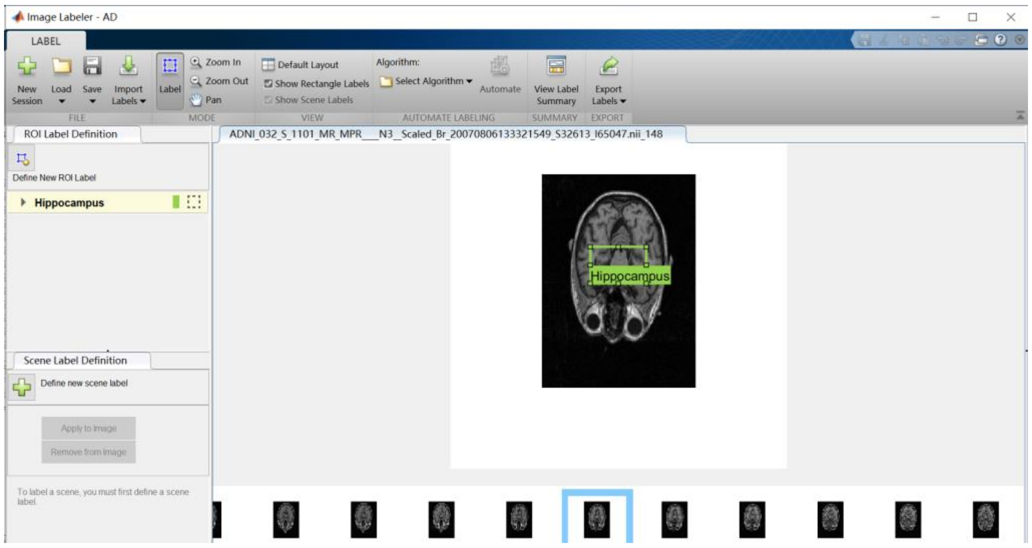
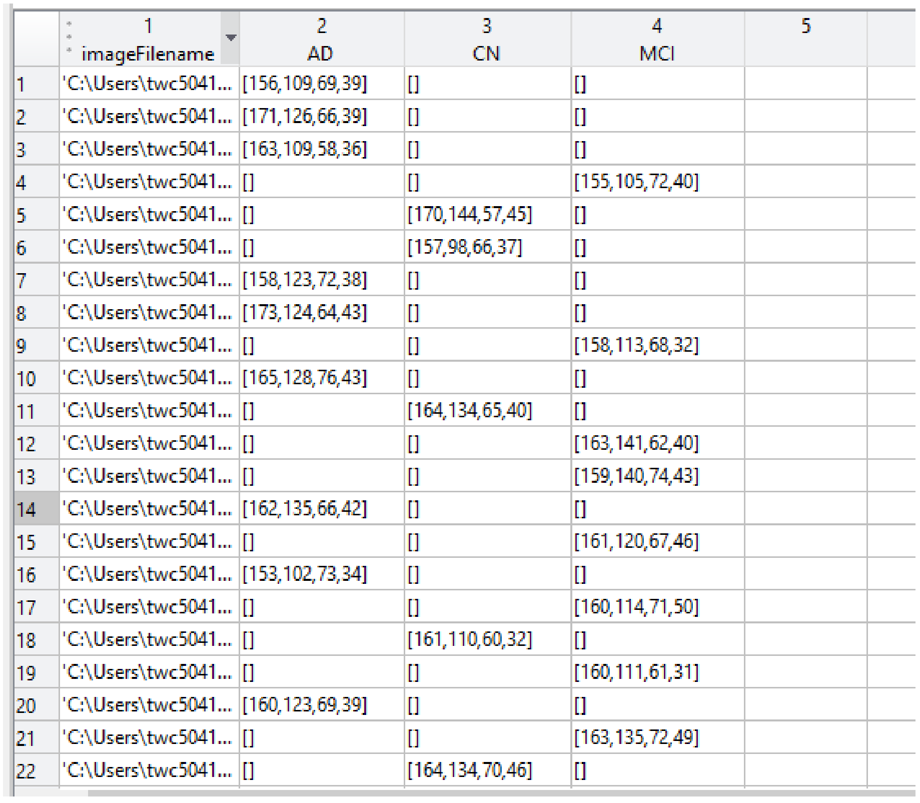


Figure 4. 3 The labeling process



	1 imageFilename	2 AD	3 CN	4 MCI	5
1	'C:\Users\twc5041...	[156,109,69,39]	[]	[]	
2	'C:\Users\twc5041...	[171,126,66,39]	[]	[]	
3	'C:\Users\twc5041...	[163,109,58,36]	[]	[]	
4	'C:\Users\twc5041...	[]	[]	[155,105,72,40]	
5	'C:\Users\twc5041...	[]	[170,144,57,45]	[]	
6	'C:\Users\twc5041...	[]	[157,98,66,37]	[]	
7	'C:\Users\twc5041...	[158,123,72,38]	[]	[]	
8	'C:\Users\twc5041...	[173,124,64,43]	[]	[]	
9	'C:\Users\twc5041...	[]	[]	[158,113,68,32]	
10	'C:\Users\twc5041...	[165,128,76,43]	[]	[]	
11	'C:\Users\twc5041...	[]	[164,134,65,40]	[]	
12	'C:\Users\twc5041...	[]	[]	[163,141,62,40]	
13	'C:\Users\twc5041...	[]	[]	[159,140,74,43]	
14	'C:\Users\twc5041...	[162,135,66,42]	[]	[]	
15	'C:\Users\twc5041...	[]	[]	[161,120,67,46]	
16	'C:\Users\twc5041...	[153,102,73,34]	[]	[]	
17	'C:\Users\twc5041...	[]	[]	[160,114,71,50]	
18	'C:\Users\twc5041...	[]	[161,110,60,32]	[]	
19	'C:\Users\twc5041...	[]	[]	[160,111,61,31]	
20	'C:\Users\twc5041...	[160,123,69,39]	[]	[]	
21	'C:\Users\twc5041...	[]	[]	[163,135,72,49]	
22	'C:\Users\twc5041...	[]	[164,134,70,46]	[]	

Figure 4. 4 Examples of the dataset

4.3 Experimental Environment

Deep learning is increasingly demanding hardware resources for two main reasons. First, the process requires a large amount of data resources, including the data itself and labels of the data, because deep learning requires more training data than other methods due to a large number of parameters and strong expression ability, more training samples usually lead to higher accuracy(Sun, Shrivastava, Singh, & Gupta, 2017). Second, the scale of deep learning networks continues to increase and deepen, so that deep learning has to calculate massive parameters during the training process. Google used a data set of more than 300 million images to train on 50 GPUs using asynchronous parallel computing methods, but they could only train four rounds in two months (Sun et al., 2017).

Deep learning, no matter during training the model or testing the model, requires a large number of computing resources, such as memory, GPU/TPU/NPU, and storage, etc.

Classic deep learning models have a huge number of parameters to obtain strong representation. For example, VGG16 network has more than 10^8 parameters and occupies more than 500 MB of memory space (Simonyan & Zisserman, 2014). This model cannot be used in mobile phones and other hardware resources. Training these models requires more memory to save intermediate results in the learning process (e.g., gradients used to update parameters, activation maps of the processing layers of the model, etc.).

When GPU is used to speed up the training process, the video memory size of the graphic card has become another bottleneck. In some complex tasks, such as UberNet, which integrates recognition, segmentation, and detection tasks into a single network, now if an advanced graphic card does not use a special algorithm, its memory capacity will not even be sufficient to deal with a single input image, which will bring tremendous resource to the learning and test (Kokkinos, 2017).

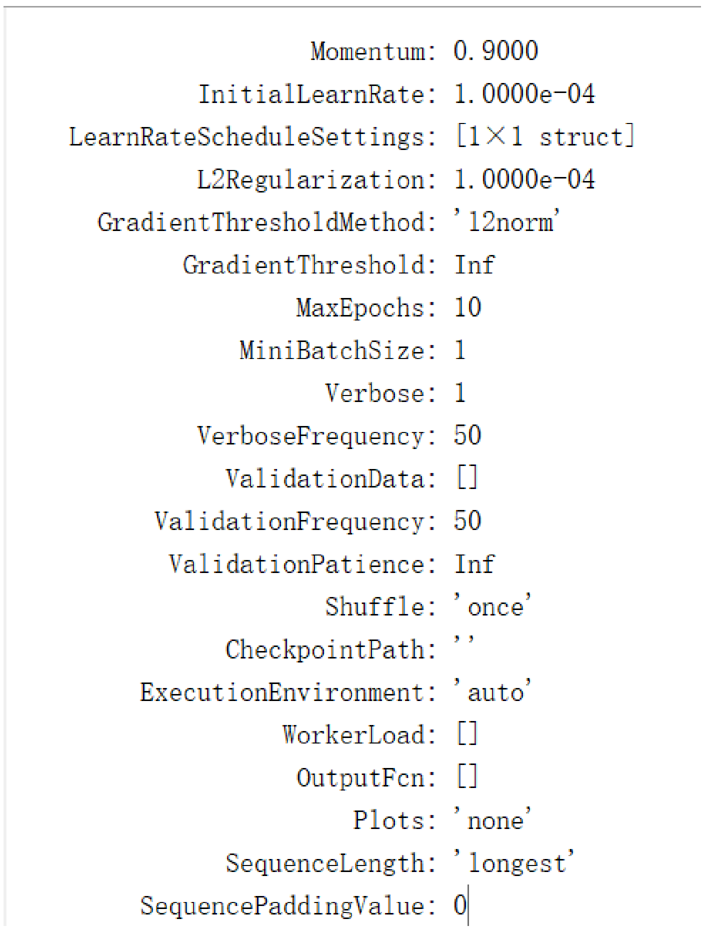
The amount of computation is reflected in CPU and GPU time. Due to a large number of parameters of deep learning models and the requirement for a huge number of floating-point operations, both the training and test phases occupy a large amount of CPU /GPU computing time. Taking training as an example, because a single training set needs to carry out calculations and has a large number of training samples, even if it is accelerated by using multiple GPUs, the learning time of large-scale deep models is often in weeks or even months. When testing, though it can complete many tasks in real time with high-performance GPU card acceleration, on common resource-constrained platforms, the classic deep learning model not only exceeds the memory, storage, and other resources that the platform can provide. Its running time and CPU usage are also far beyond the standard, which has affected the widespread application of deep learning.

The hardware resources used in this experiment include the Intel i7-9750H CPU, 1TB SSD, 16GB of memory, and Nvidia Geforce RTX 2070 graphics card with 8GB of video memory. In addition, the software environment of this experiment is based on MATLAB 2019a version with the Deep Learning Toolbox as the main tool. At the same time, the Parallel Computing Toolbox is also employed to support the parallel computing function of GPUs. In addition, the Computer Vision Toolbox is also installed to support our algorithms and functions in this experiment.

There are several hyperparameters that can be adjusted when using the Deep Learning Toolbox of MATLAB. The hyperparameters in this experiment are listed in Figure 4.5.

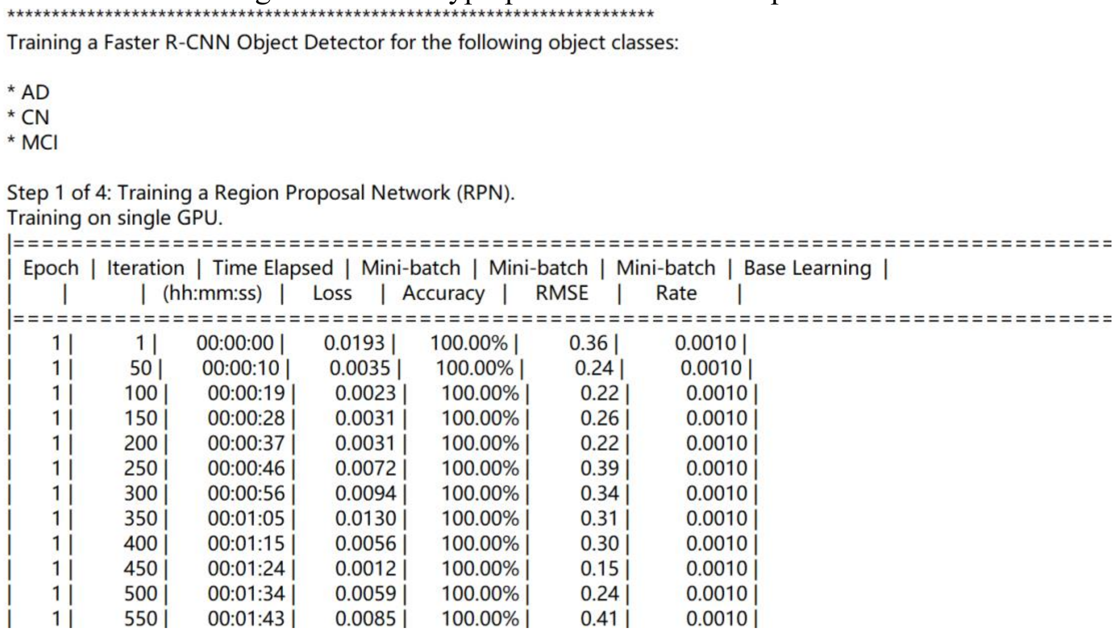
4.4 Experiment Procedure

There are four steps of the training process for Faster R-CNN. The first one is to train a Region Proposal Network (RPN), which uses a pretrained model on ImageNet data set to initialize the feature extraction network and train the RPN network. Figure 4.6 shows the training start state of Faster R-CNN, we see that the object classes are divided into three categories: AD, CN and MCI, the training process is carried out based on the GPU.



```
                   Momentum: 0.9000
            InitialLearnRate: 1.0000e-04
   LearnRateScheduleSettings: [1×1 struct]
            L2Regularization: 1.0000e-04
     GradientThresholdMethod: 'l2norm'
           GradientThreshold: Inf
                   MaxEpochs: 10
               MiniBatchSize: 1
                     Verbose: 1
            VerboseFrequency: 50
              ValidationData: []
         ValidationFrequency: 50
          ValidationPatience: Inf
                     Shuffle: 'once'
              CheckpointPath: ''
        ExecutionEnvironment: 'auto'
                  WorkerLoad: []
                   OutputFcn: []
                       Plots: 'none'
              SequenceLength: 'longest'
        SequencePaddingValue: 0
```

Figure 4. 5 The hyperparameters in this experiment



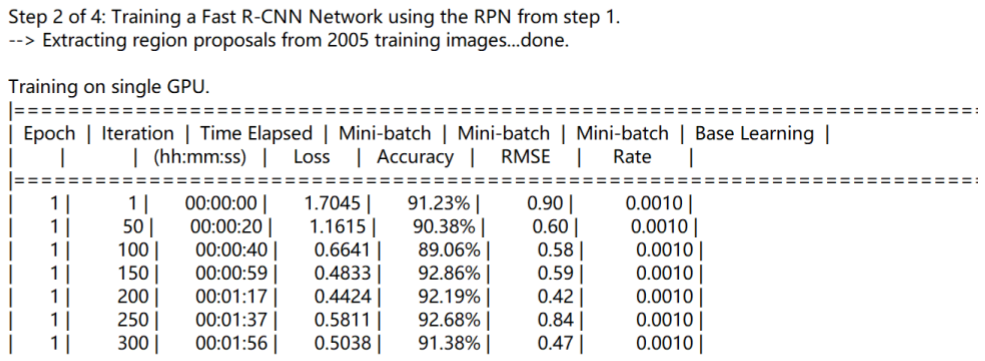
```
*************************************************************************
Training a Faster R-CNN Object Detector for the following object classes:

* AD
* CN
* MCI

Step 1 of 4: Training a Region Proposal Network (RPN).
Training on single GPU.
|=========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |     Loss     |   Accuracy   |     RMSE     |      Rate       |
|=========================================================================================|
|       1 |           1 |       00:00:00 |       0.0193 |      100.00% |         0.36 |          0.0010 |
|       1 |          50 |       00:00:10 |       0.0035 |      100.00% |         0.24 |          0.0010 |
|       1 |         100 |       00:00:19 |       0.0023 |      100.00% |         0.22 |          0.0010 |
|       1 |         150 |       00:00:28 |       0.0031 |      100.00% |         0.26 |          0.0010 |
|       1 |         200 |       00:00:37 |       0.0031 |      100.00% |         0.22 |          0.0010 |
|       1 |         250 |       00:00:46 |       0.0072 |      100.00% |         0.39 |          0.0010 |
|       1 |         300 |       00:00:56 |       0.0094 |      100.00% |         0.34 |          0.0010 |
|       1 |         350 |       00:01:05 |       0.0130 |      100.00% |         0.31 |          0.0010 |
|       1 |         400 |       00:01:15 |       0.0056 |      100.00% |         0.30 |          0.0010 |
|       1 |         450 |       00:01:24 |       0.0012 |      100.00% |         0.15 |          0.0010 |
|       1 |         500 |       00:01:34 |       0.0059 |      100.00% |         0.24 |          0.0010 |
|       1 |         550 |       00:01:43 |       0.0085 |      100.00% |         0.41 |          0.0010 |
```

Figure 4. 6 The first step of training Faster R-CNN

The second step is to train a Fast R-CNN Network using the RPN from Step 1. This step utilizes the model pretrained based on ImageNet to initialize the Fast R-CNN feature extraction network and applies the candidate frames generated by the RPN network trained in Step 1 as input to train a Fast R-CNN network. So far, each layer of the two networks parameters is not shared at all. Figure 4.7 shows the second step of training Faster R-CNN.

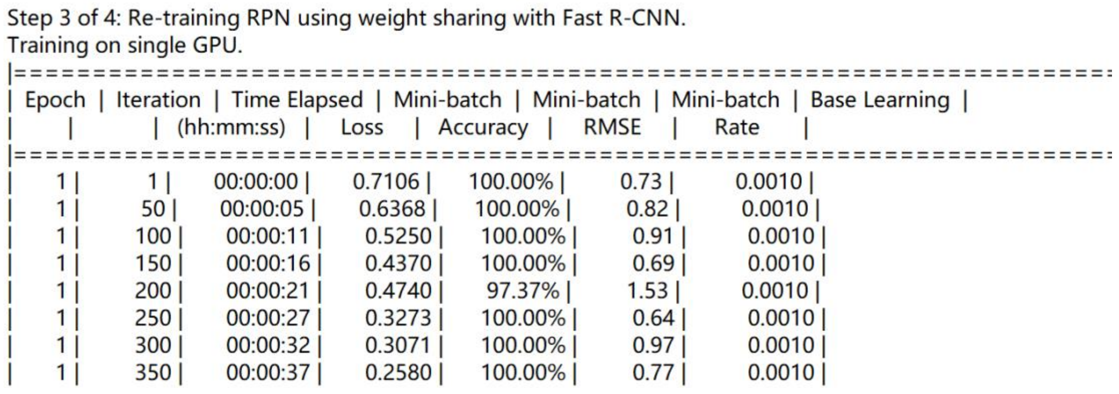


```
Step 2 of 4: Training a Fast R-CNN Network using the RPN from step 1.
--> Extracting region proposals from 2005 training images...done.

Training on single GPU.
|========================================================================:
| Epoch | Iteration | Time Elapsed | Mini-batch | Mini-batch | Mini-batch | Base Learning |
|       |           | (hh:mm:ss)   |    Loss    |  Accuracy  |    RMSE    |     Rate      |
|========================================================================:
|     1 |         1 |     00:00:00 |     1.7045 |     91.23% |       0.90 |        0.0010 |
|     1 |        50 |     00:00:20 |     1.1615 |     90.38% |       0.60 |        0.0010 |
|     1 |       100 |     00:00:40 |     0.6641 |     89.06% |       0.58 |        0.0010 |
|     1 |       150 |     00:00:59 |     0.4833 |     92.86% |       0.59 |        0.0010 |
|     1 |       200 |     00:01:17 |     0.4424 |     92.19% |       0.42 |        0.0010 |
|     1 |       250 |     00:01:37 |     0.5811 |     92.68% |       0.84 |        0.0010 |
|     1 |       300 |     00:01:56 |     0.5038 |     91.38% |       0.47 |        0.0010 |
```

Figure 4. 7 The second step of training Faster R-CNN

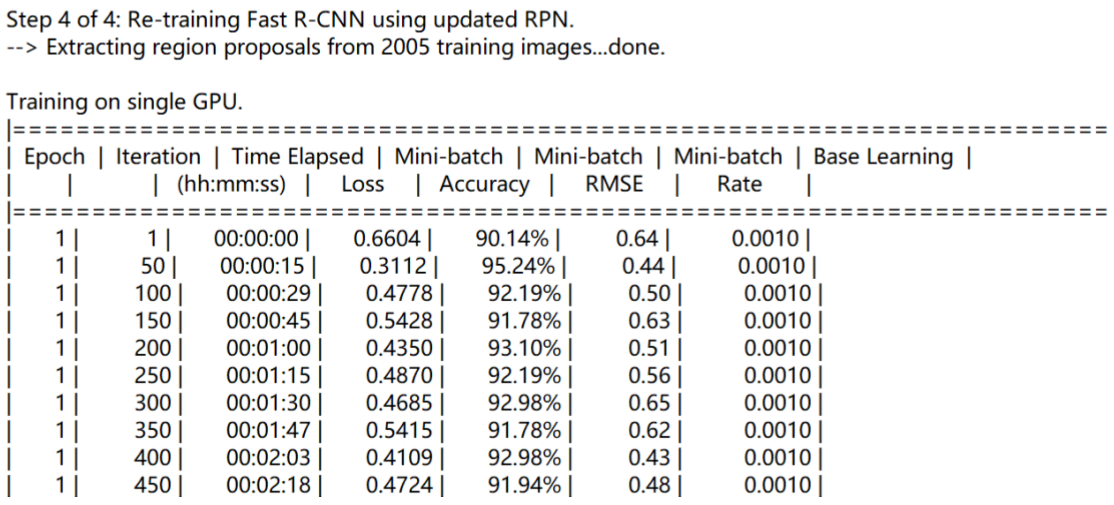
The third step of training is re-training RPN using weight sharing with Fast R-CNN. It uses the parameters of Fast R-CNN network of Step 2 to initialize a new RPN network, we set the learning rate of the feature extraction network parameters shared by RPN and Fast R-CNN to 0. In this way, it fixes the feature extraction network and only learns the characteristics of the RPN network Parameters. At this point, both networks have shared all common convolutional layers. Figure 4.8 shows the third step of training Faster R-CNN.



```
Step 3 of 4: Re-training RPN using weight sharing with Fast R-CNN.
Training on single GPU.
|========================================================================:
| Epoch | Iteration | Time Elapsed | Mini-batch | Mini-batch | Mini-batch | Base Learning |
|       |           | (hh:mm:ss)   |    Loss    |  Accuracy  |    RMSE    |     Rate      |
|========================================================================:
|     1 |         1 |     00:00:00 |     0.7106 |    100.00% |       0.73 |        0.0010 |
|     1 |        50 |     00:00:05 |     0.6368 |    100.00% |       0.82 |        0.0010 |
|     1 |       100 |     00:00:11 |     0.5250 |    100.00% |       0.91 |        0.0010 |
|     1 |       150 |     00:00:16 |     0.4370 |    100.00% |       0.69 |        0.0010 |
|     1 |       200 |     00:00:21 |     0.4740 |     97.37% |       1.53 |        0.0010 |
|     1 |       250 |     00:00:27 |     0.3273 |    100.00% |       0.64 |        0.0010 |
|     1 |       300 |     00:00:32 |     0.3071 |    100.00% |       0.97 |        0.0010 |
|     1 |       350 |     00:00:37 |     0.2580 |    100.00% |       0.77 |        0.0010 |
```

Figure 4. 8 The third step of training Faster R-CNN

The last step of training is re-training Fast R-CNN using updated RPN. At this step, the shared network layers are still fixed. The training process adds the unique network layers of Fast R-CNN to continue training and fine-tuning. So far, the RPN and Fast R-CNN network completely share the parameters. Using Fast R-CNN can complete the candidate frame extraction and target detection functions at the same time. Figure 4.9 shows the last step of training Faster R-CNN.

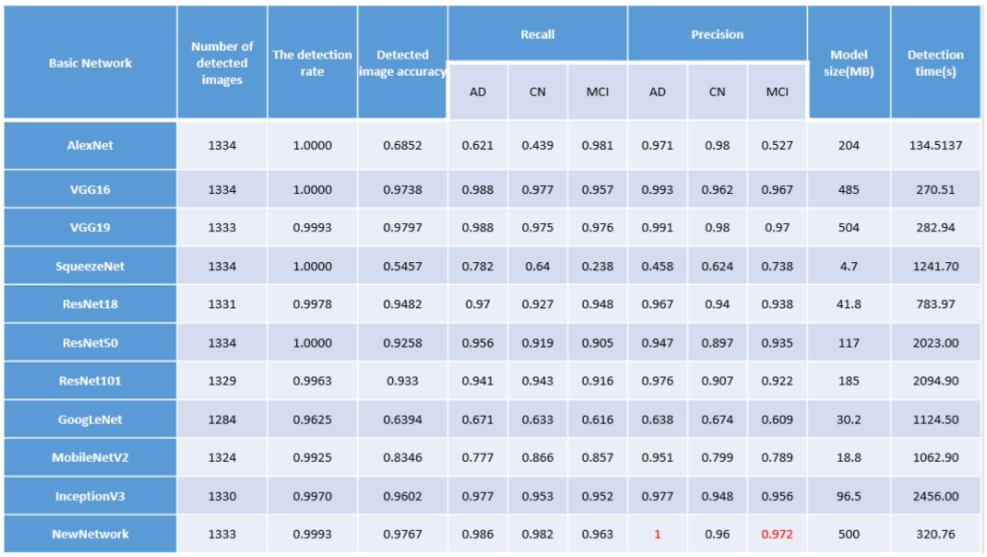


```
Step 4 of 4: Re-training Fast R-CNN using updated RPN.
--> Extracting region proposals from 2005 training images...done.

Training on single GPU.
|=====================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |     Loss     |   Accuracy   |     RMSE     |      Rate       |
|=====================================================================================|
|       1 |           1 |       00:00:00 |       0.6604 |       90.14% |         0.64 |          0.0010 |
|       1 |          50 |       00:00:15 |       0.3112 |       95.24% |         0.44 |          0.0010 |
|       1 |         100 |       00:00:29 |       0.4778 |       92.19% |         0.50 |          0.0010 |
|       1 |         150 |       00:00:45 |       0.5428 |       91.78% |         0.63 |          0.0010 |
|       1 |         200 |       00:01:00 |       0.4350 |       93.10% |         0.51 |          0.0010 |
|       1 |         250 |       00:01:15 |       0.4870 |       92.19% |         0.56 |          0.0010 |
|       1 |         300 |       00:01:30 |       0.4685 |       92.98% |         0.65 |          0.0010 |
|       1 |         350 |       00:01:47 |       0.5415 |       91.78% |         0.62 |          0.0010 |
|       1 |         400 |       00:02:03 |       0.4109 |       92.98% |         0.43 |          0.0010 |
|       1 |         450 |       00:02:18 |       0.4724 |       91.94% |         0.48 |          0.0010 |
```

Figure 4. 9 The fourth step of training Faster R-CNN

4.5 Experimental Results



Basic Network	Number of detected images	The detection rate	Detected image accuracy	Recall			Precision			Model size(MB)	Detection time(s)
				AD	CN	MCI	AD	CN	MCI		
AlexNet	1334	1.0000	0.6852	0.621	0.439	0.981	0.971	0.98	0.527	204	134.5137
VGG16	1334	1.0000	0.9738	0.988	0.977	0.957	0.993	0.962	0.967	485	270.51
VGG19	1333	0.9993	0.9797	0.988	0.975	0.976	0.991	0.98	0.97	504	282.94
SqueezeNet	1334	1.0000	0.5457	0.782	0.64	0.238	0.458	0.624	0.738	4.7	1241.70
ResNet18	1331	0.9978	0.9482	0.97	0.927	0.948	0.967	0.94	0.938	41.8	783.97
ResNet50	1334	1.0000	0.9258	0.956	0.919	0.905	0.947	0.897	0.935	117	2023.00
ResNet101	1329	0.9963	0.933	0.941	0.943	0.916	0.976	0.907	0.922	185	2094.90
GoogLeNet	1284	0.9625	0.6394	0.671	0.633	0.616	0.638	0.674	0.609	30.2	1124.50
MobileNetV2	1324	0.9925	0.8346	0.777	0.866	0.857	0.951	0.799	0.789	18.8	1062.90
InceptionV3	1330	0.9970	0.9602	0.977	0.953	0.952	0.977	0.948	0.956	96.5	2456.00
NewNetwork	1333	0.9993	0.9767	0.986	0.982	0.963	1	0.96	0.972	500	320.76

Table 4. 1 Results of experiment

This experiment tested various basic networks supported by the Faster R-CNN network of MATLAB. The basic networks performed poorly on the data set of this experiment, such as AlexNet, SqueezeNet, and GoogLeNet. However, in these networks, VGG, ResNet, Inceptionv3, and the newNetwork obtained high accuracy. As shown in Table 4.1, the results of all basic networks implemented in this experiment with model size and detection time.

A total of 1334 images are used for verification in this experiment. We see from the experimental results that the Faster R-CNN network based on VGG19, ResNet18, ResNet101, GoogLeNet, MobileNetv2, Inceptionv3, and the newNetwork cannot detect all the images. Among them, GoogLeNet has the lowest detection rate with 96.25%, only 1284 of 1334 validation images are detected. In contrast, detection models based on AlexNet, VGG16, SqueezeNet, and ResNet50 detected all the validation images. The Faster R-CNN network based on VGG19 and the newNetwork detected almost all images, the performance is almost excellent.

In terms of the accuracy of the detected images, the performance of detectors based on VGG16, VGG19, the newNetwork is excellent with the accuracy of more than 97%. In addition, they have the highest relatively high detection precision for AD and MCI samples, which is crucial in disease detection. Figure 4.10, Figure 4.11, and Figure 4.12 show the confusion matrixes based on the detection results of the three networks.

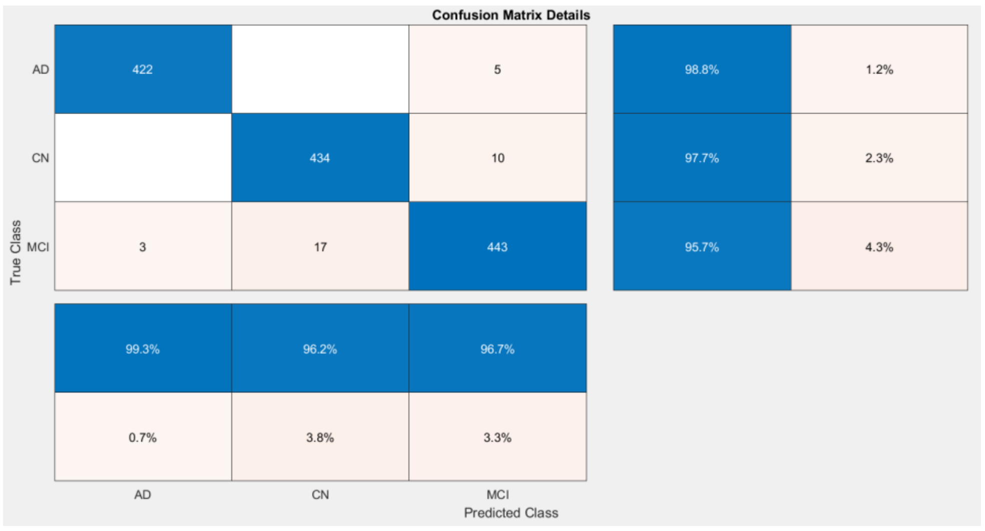
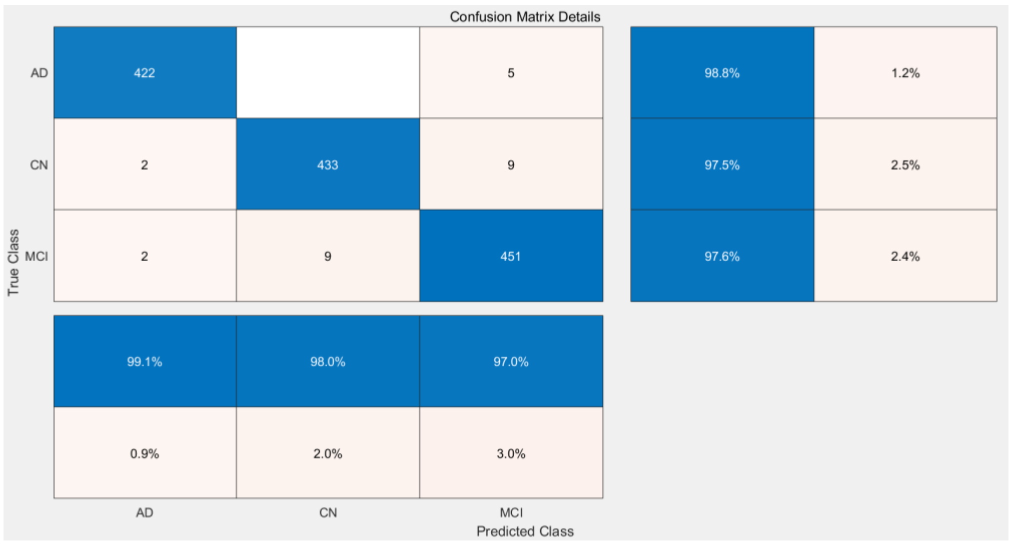


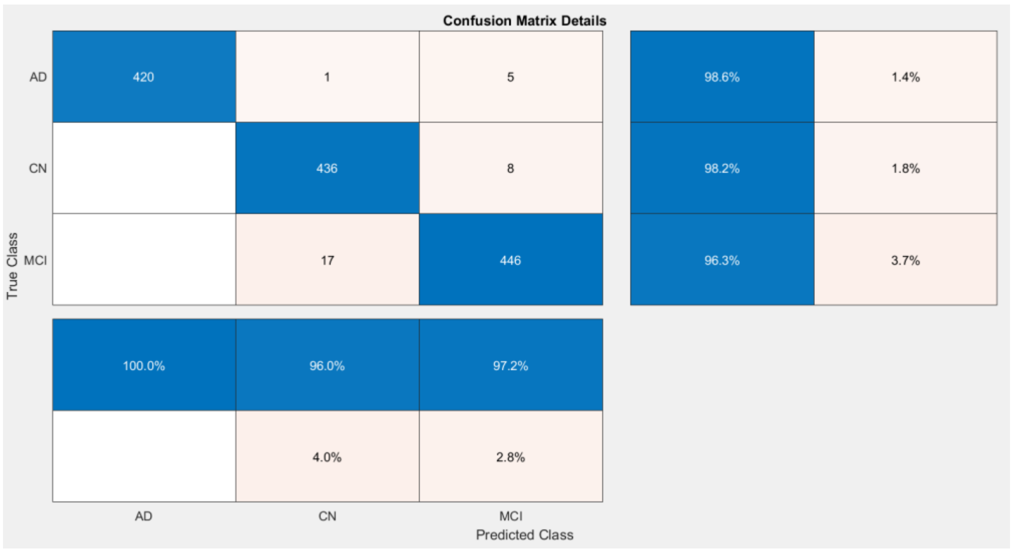
Figure 4. 10 The confusion matrix of the result of VGG16 + Faster R-CNN


Confusion Matrix Details

True Class	AD	CN	MCI		
AD	422		5	98.8%	1.2%
CN	2	433	9	97.5%	2.5%
MCI	2	9	451	97.6%	2.4%
	99.1%	98.0%	97.0%		
	0.9%	2.0%	3.0%		
	AD	CN	MCI		

Predicted Class

Figure 4. 11 The confusion matrix of the result of VGG19 + Faster R-CNN


Confusion Matrix Details

True Class	AD	CN	MCI		
AD	420	1	5	98.6%	1.4%
CN		436	8	98.2%	1.8%
MCI		17	446	96.3%	3.7%
	100.0%	96.0%	97.2%		
		4.0%	2.8%		
	AD	CN	MCI		

Predicted Class

Figure 4. 12 The confusion matrix of the result of newNetwork + Faster R-CNN

From the confusion matrix, we see that the detector based on the newNetwork has achieved 100% detection precision for AD samples, which means, it has correctly detected all AD samples. This is significant for the diagnosis and screening of Alzheimer's disease.

In terms of model size and detection time, the detector models based on the VGG basic networks have large sizes, which require about 500MB of storage space, but the detection time for the entire validation set is relatively short.

Chapter 5

Analysis and Discussions

In this chapter, dialectical comparative analysis of the results of all models in the experiment is performed. In addition, in the last section of this chapter, a discussion of experimental results is presented.

5.1 Analysis

In order to verify the robustness of various models, this experiment selected one image of each category of samples for test.

Faster R-CNN + AlexNet as the base network has only five convolutional layers, the number of network layers is not deep enough and the number of filters is small, this results in poor performance of feature extraction. Although it can detect all the verification images, the accuracy is only 62.1%. From the test image in Figure 5.1, we see that AlexNet + Faster R-CNN fail to correctly classify AD samples and detect hippocampus region. It detects many regions that do not contain hippocampus and even searches the regions outside the effective image area as hippocampus regions. On the other hand, due to the shallow number of neural network layers and the small number of calculation parameters, AlexNet + Faster R-CNN performs well in model size and detection time.

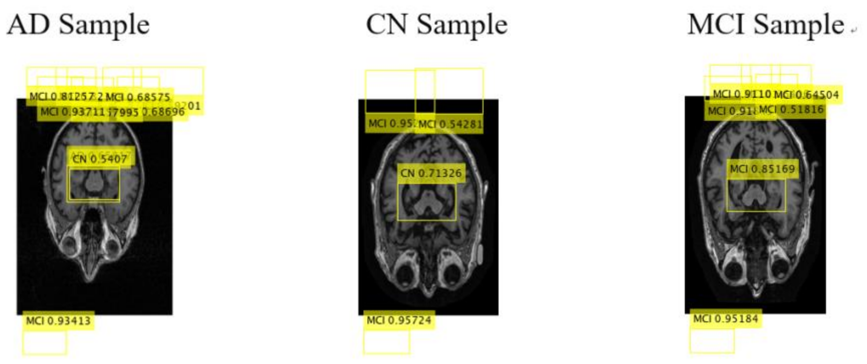


Figure 5. 1 The examples of test results of AlexNet+Faster R-CNN

The network structure of GoogLeNet+Faster R-CNN which has 155 layers is much more complicated than AlexNet+Faster R-CNN. The Inception structure used by GoogLeNet provides a wider network structure by using 1×1, 3×3, and 5×5 convolution kernels. However, such a complex feature extraction network has not achieved good results in the data set of this experiment and its detection rate is the lowest among all models. From the test example shown in Figure 5.2, we see that GoogLeNet+Faster R-CNN fails to correctly classify the CN and MCI samples, the confidence of the bounding box contains the hippocampus region is also low.

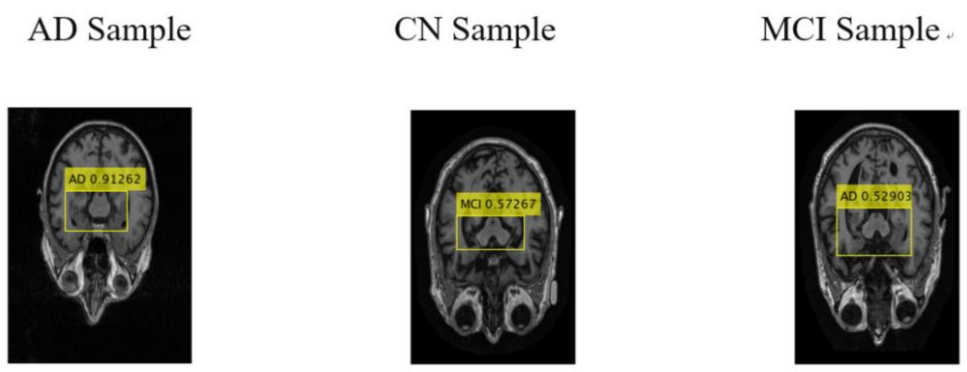


Figure 5. 2 The examples of test results of GoogLeNet-Faster R-CNN

Compared to GoogLeNet, Inceptionv3 has structural changes. It uses the RMSProp optimizer and turns the 7×7 convolution into a 1×7 and 7×1 convolution kernel stack (Szegedy, Vanhoucke, Ioffe, Shlens, & Wojna, 2016). The number of network layers of Inceptionv3+Faster R-CNN has also increased to 327 layers. At the same time, the detection rate and accuracy of the validation set in this experiment have also been improved. However, because the network structure is too deep, the amount of calculation increases, the detection time for the entire validation set is also the longest. Figure 5.3 shows the examples of test results of Inceptionv3+Faster R-CNN, the accuracy is much better than GoogLeNet+Faster R-CNN.

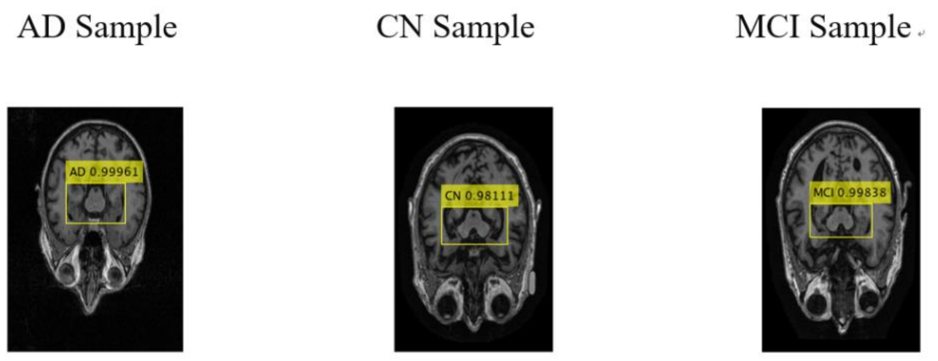


Figure 5. 3 Examples of test results of Inceptionv3+Faster R-CNN

As a lightweight network, MobileNetv2 uses inverted residuals, which is different from the traditional residual block in which the number of feature map channels is reduced before the 3×3 convolution layer, and increased after the 3×3 convolution layer. Inverted residuals use the opposite approach, first increase the number of channels in the feature map and then compress the number of channels after passing through a 3×3 convolution layer (Sandler, Howard, Zhu, Zhmoginov, & Chen, 2018).

In addition, MobileNetv2 uses linear bottlenecks to avoid damages to feature maps using ReLU after channel number compression. In this way, even though MobileNetv2+Faster R-CNN has a network depth of 177 layers, its model size can still be controlled to be very small, only 18.8MB. However, because MobileNetv2 was originally designed to run on mobile devices, the number of channels extracted by the feature map is not enough, which leads to average performance on the data set of this experiment. Figure 5.4 shows the examples of test by using MobileNetv2+Faster R-CNN.

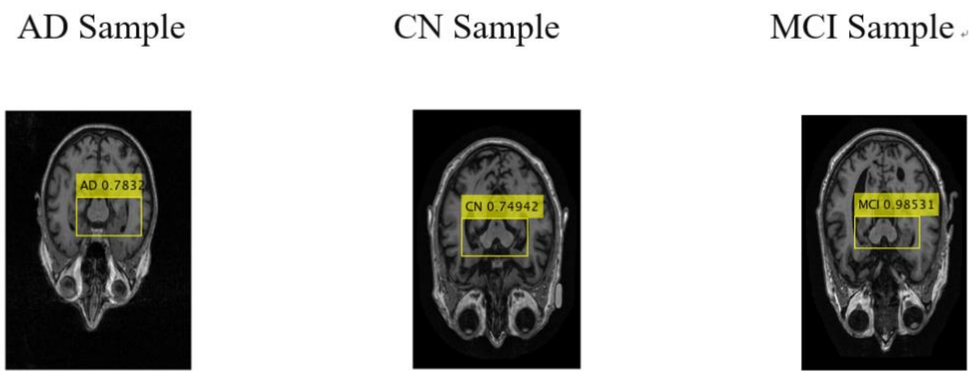


Figure 5. 4 Examples of test results of MobileNetv2+Faster R-CNN

Like MobileNetv2, SqueezeNet is also designed to minimize the amount of calculation and learnable parameters for model training and test, thereby reduce the size of the model and facilitate the preservation and transmission of the model. SqueezeNet focuses on the application of the embedded environment, the size of the model can achieve only about 2% of the AlexNet model while the same level of accuracy as AlexNet (Iandola et al., 2016). This reflects in the results of this experiment. However, SqueezeNet exchanges for a smaller number of parameters by increasing the depth of the network, though it can reduce the parameters of the network, it loses the network parallelism and the test time will be longer. Figure 5.5 shows the examples of test results of SqueezeNet-Faster R-CNN. Like AlexNet+Faster R-CNN, SqueezeNet+Faster R-CNN also suffers from inaccurate classification and confusion of bounding box predictions. It performs poorly in the data set of this experiment.

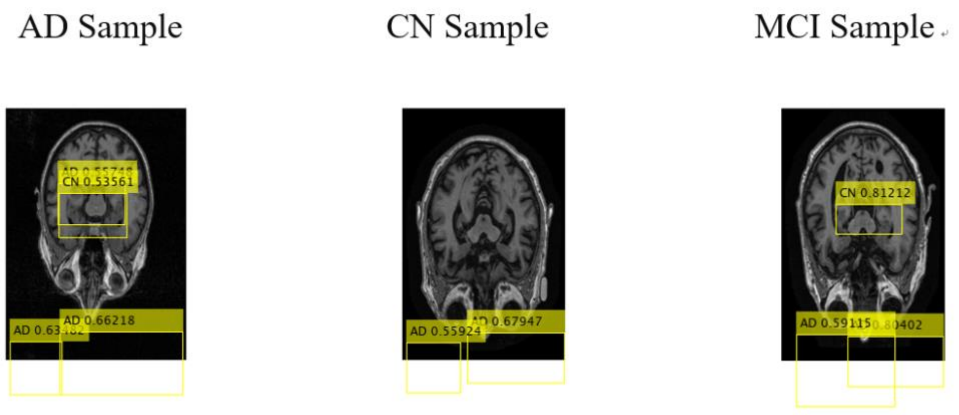


Figure 5. 5 The examples of test results of SqueezeNet+Faster R-CNN

ResNet networks have always been considered as excellent feature extraction networks. They also perform very well on the data set of this experiment and have achieved relatively high accuracy. ResNet18+Faster R-CNN uses different shallow residual blocks, rather than the other two ResNet networks, two 3×3 convolution kernels are used in each residual block. The accuracy of the detected image was 94.82% based on the data set of this experiment, but its image detection rate was lower than that of ResNet50 with deeper network layers.

Figure 5.6 shows the examples of test results of ResNet18+Faster R-CNN. ResNet50+Faster R-CNN and ResNet101+Faster R-CNN use residual blocks suitable for deeper network structures to resolve the gradient vanishing problem (He, et al., 2016). ResNet101+Faster R-CNN has a network structure of 358 layers, which is almost twice the 188 layers of ResNet50-Faster R-CNN. However, the performance of a deeper network structure is not necessarily better. On the data set of this experiment, the detection rate of ResNet101-Faster R-CNN for the validation set image is the lowest one among the three ResNet networks, though the accuracy of the detected image is slightly higher than the other two. In addition, with deepening the network structure, the model size and detection time of ResNet-Faster R-CNN also increase. Figure 5.7 and Figure 5.8 show the examples of test results of ResNet50+Faster R-CNN and ResNet101+Faster R-CNN.

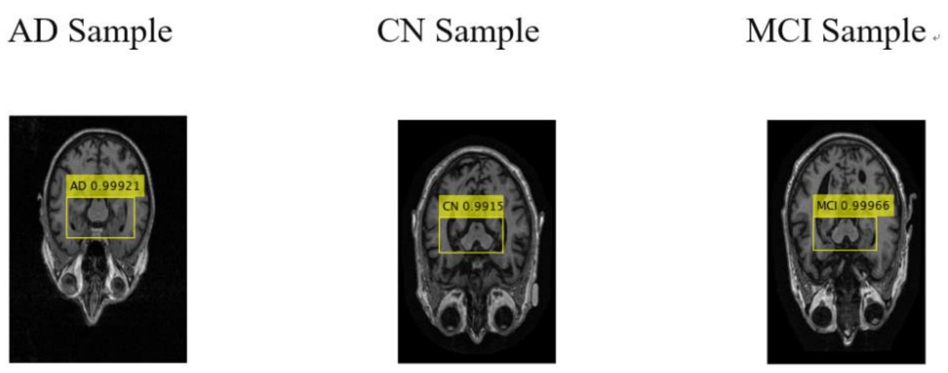


Figure 5. 6 The examples of test results of ResNet18+Faster R-CNN

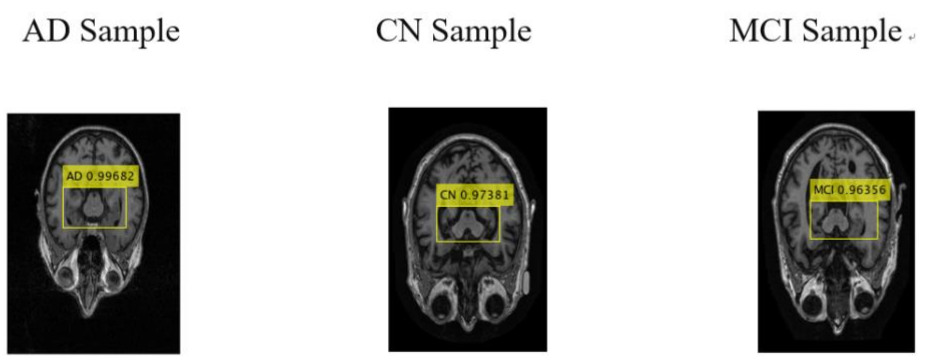


Figure 5. 7 The examples of test results of ResNet50+Faster R-CNN

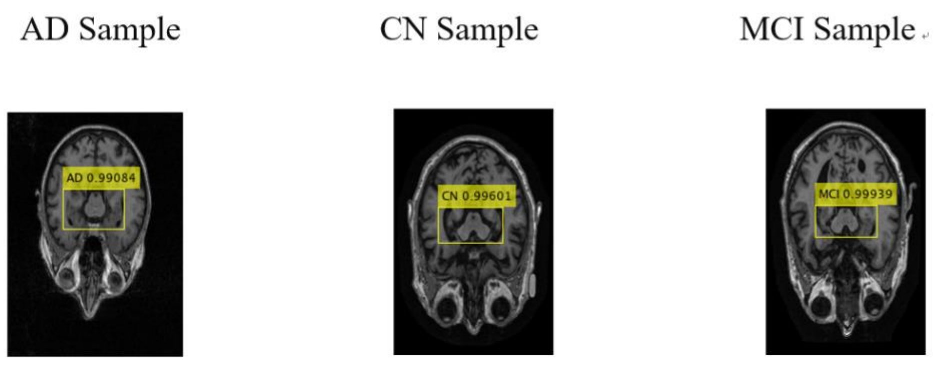


Figure 5. 8 The examples of test results of ResNet101+Faster R-CNN

Although the VGG network has a simple structure, it is a promising network. It can extract useful features of the image well, so it has a wide range of applications in the image field. Each convolutional layer of the VGG network uses the same 3×3 small convolution kernel while using the 2×2 pooling layer with the same parameters to reduce the dimension. With the increase of the number of convolutional layers, the number of channels of the convolution kernel continues to increase, though the structure of VGG is simple, the number of weights is very large. Taken VGG16 as an example, it has reached 138,357,544 parameters. These parameters include convolution kernel weights and fully connected layer weights.

Therefore, the VGG network has two main disadvantages. First, the training time is too long, it is difficult to adjust the parameters. Second, the model size is too large, which requires a large storage space and is not conducive to deployment. For example, it is not conducive to install them in mobile devices and embedded systems (Simonyan & Zisserman, 2014).

In addition, because the VGG network does not have residual blocks like ResNet to effectively transfer gradients, its network layers cannot be constructed too deeply, so VGG16 has only 13 convolutional layers and three fully connected layers, while VGG19 has only 16 convolutional layers and three fully connected layers. But for the data set of this experiment, the performance of VGG is good enough. VGG16+Faster

R-CNN detected all the validation set images and achieved an accuracy of 97.38%, while VGG19+Faster R-CNN only had one image not detected, and obtained the highest detected image accuracy of all models in this experiment, at the same time, they have achieved relatively high detection precision of AD and MCI. Figure 5.9 and Figure 5.10 show the examples of test results of VGG16+Faster R-CNN and VGG19+Faster R-CNN.

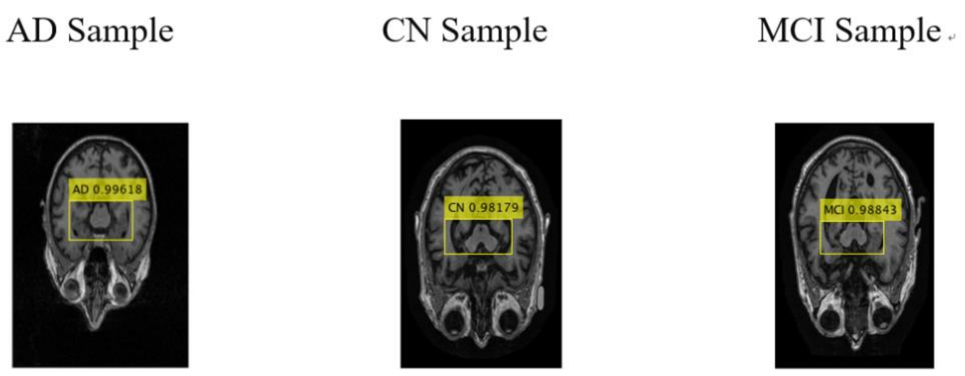


Figure 5. 9 The examples of test results of VGG16-Faster R-CNN

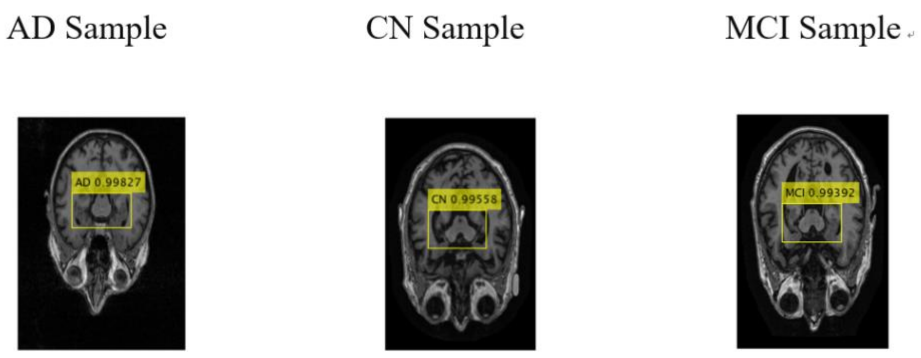
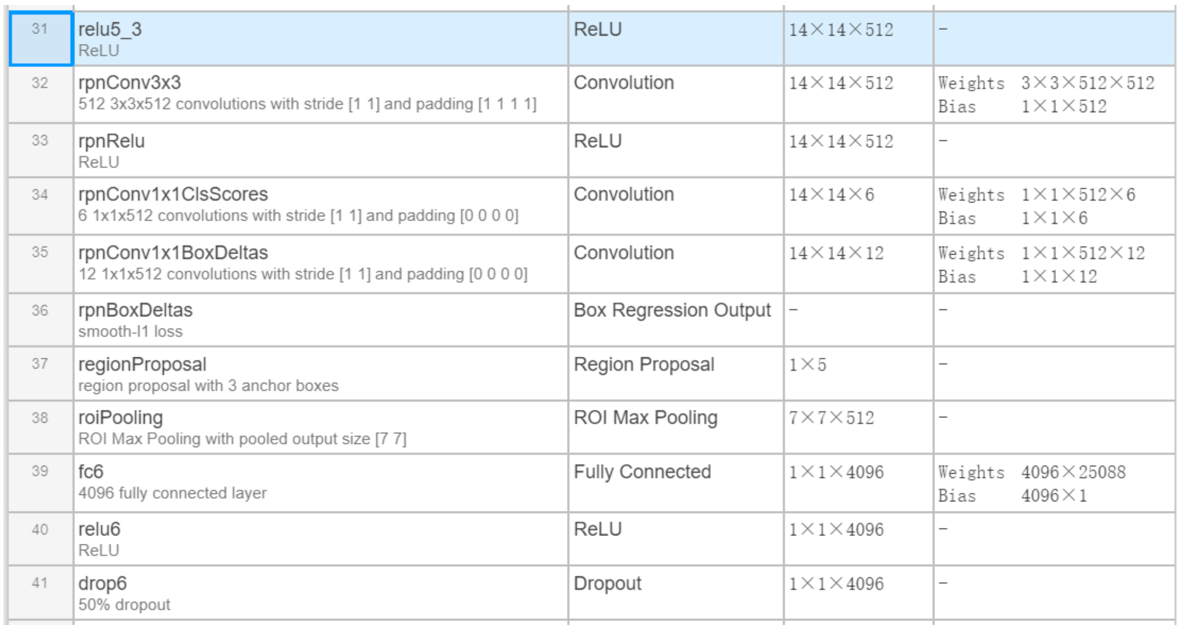


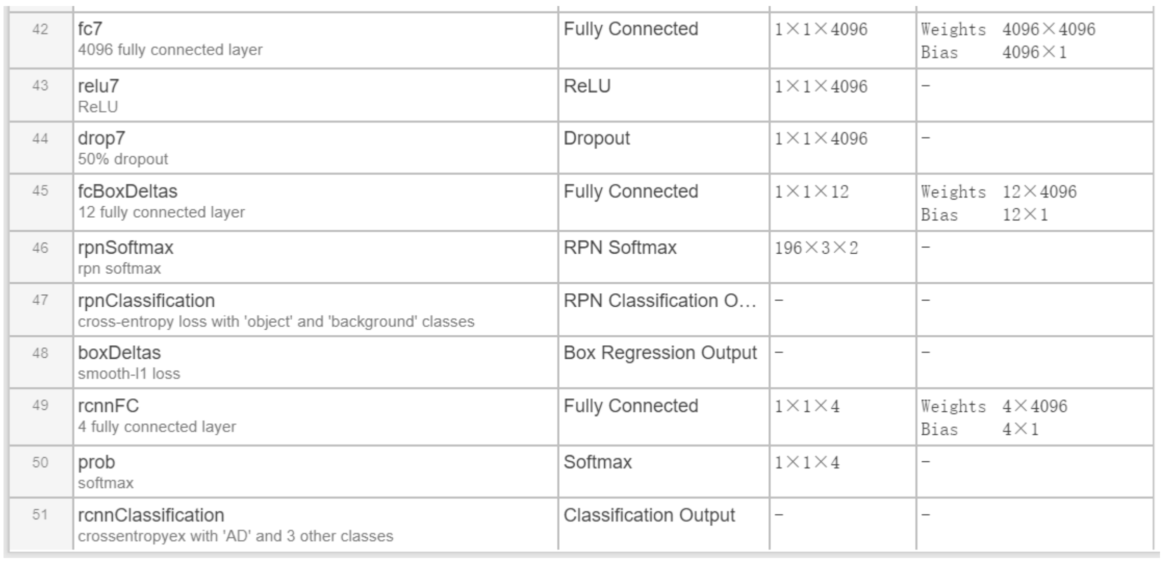
Figure 5. 10 Examples of test results of VGG19-Faster R-CNN

VGG16+Faster R-CNN has excellent phenotypic performance in this experiment. It not only can detect all the images in the validation set but also achieves a relatively high accuracy of the detected images and high precision of the prediction of AD samples. The newNetwork+Faster R-CNN based on the VGG16 network adjustment has been experimentally verified and achieved good performance. The added transposed convolutional layer makes the feature map dimension input to the RPN network increase from the original 14×14×512 to 54×54×512, the increase of this dimension only affects the RPN network.

Figure 5.11 and Figure 5.12 show the parameters after feature extraction layer of VGG16+Faster R-CNN and newNetwork+Faster R-CNN. The effect is that the accuracy of the detected image and the precision of AD and MCI prediction are improved without significantly increasing the amount of calculation. The significance of adding a transposed convolution layer is that it can upsample the feature map extracted by the feature extraction network. The upsampling does not use a predefined interpolation method, and it has learnable parameters. In this way, the upsampled feature map can restore more fine-grained feature degrees compared to the original feature map.

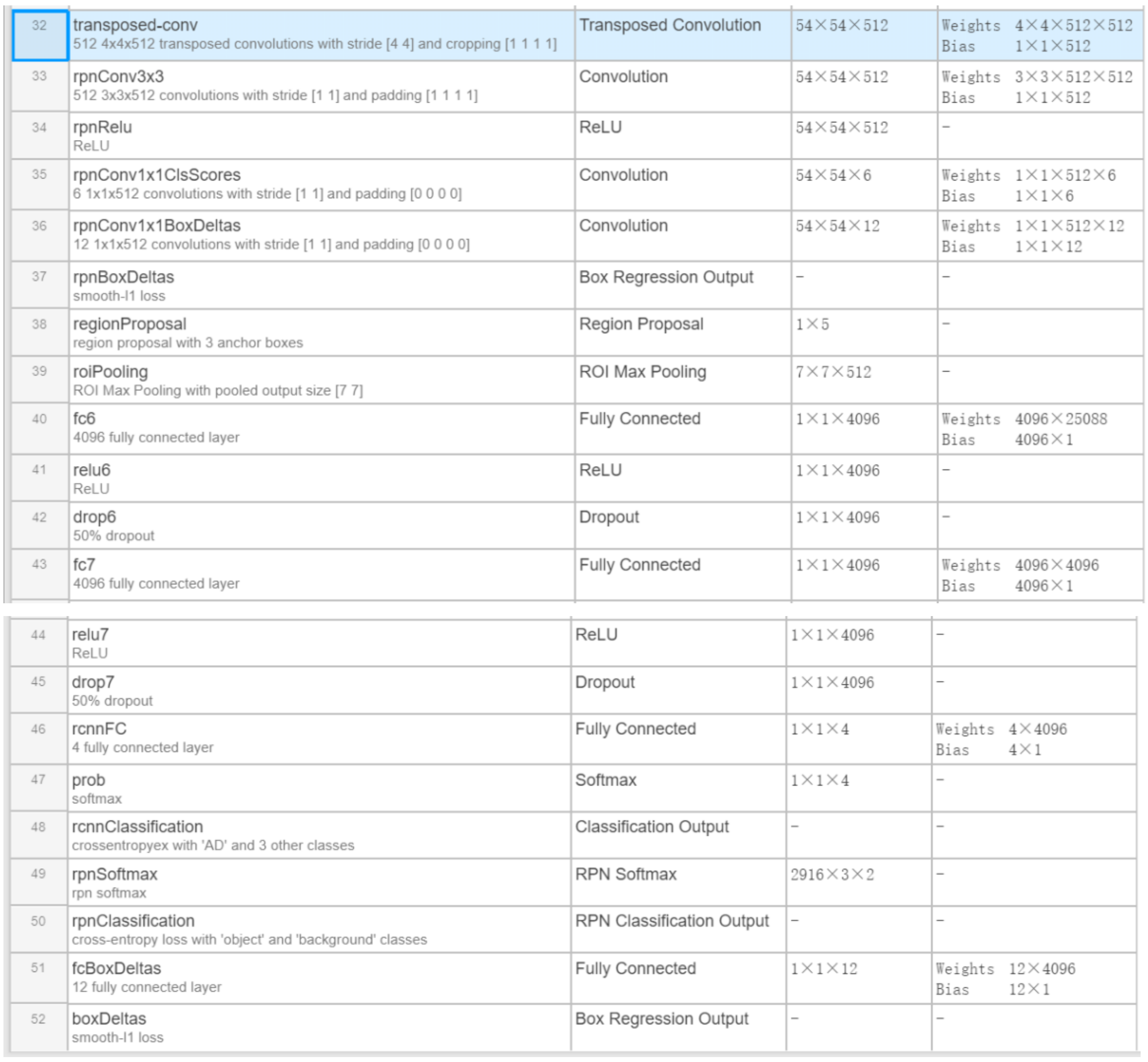


31	relu5_3 ReLU	ReLU	14×14×512	–
32	rpnConv3x3 512 3x3x512 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	14×14×512	Weights 3×3×512×512 Bias 1×1×512
33	rpnRelu ReLU	ReLU	14×14×512	–
34	rpnConv1x1ClsScores 6 1x1x512 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	14×14×6	Weights 1×1×512×6 Bias 1×1×6
35	rpnConv1x1BoxDeltas 12 1x1x512 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	14×14×12	Weights 1×1×512×12 Bias 1×1×12
36	rpnBoxDeltas smooth-l1 loss	Box Regression Output	–	–
37	regionProposal region proposal with 3 anchor boxes	Region Proposal	1×5	–
38	roiPooling ROI Max Pooling with pooled output size [7 7]	ROI Max Pooling	7×7×512	–
39	fc6 4096 fully connected layer	Fully Connected	1×1×4096	Weights 4096×25088 Bias 4096×1
40	relu6 ReLU	ReLU	1×1×4096	–
41	drop6 50% dropout	Dropout	1×1×4096	–



42	fc7 4096 fully connected layer	Fully Connected	1×1×4096	Weights 4096×4096 Bias 4096×1
43	relu7 ReLU	ReLU	1×1×4096	–
44	drop7 50% dropout	Dropout	1×1×4096	–
45	fcBoxDeltas 12 fully connected layer	Fully Connected	1×1×12	Weights 12×4096 Bias 12×1
46	rpnSoftmax rpn softmax	RPN Softmax	196×3×2	–
47	rpnClassification cross-entropy loss with 'object' and 'background' classes	RPN Classification O…	–	–
48	boxDeltas smooth-l1 loss	Box Regression Output	–	–
49	rcnnFC 4 fully connected layer	Fully Connected	1×1×4	Weights 4×4096 Bias 4×1
50	prob softmax	Softmax	1×1×4	–
51	rcnnClassification crossentropyex with 'AD' and 3 other classes	Classification Output	–	–

Figure 5. 11 VGG16+Faster R-CNN parameters after feature extraction layer



32	transposed-conv 512 4x4x512 transposed convolutions with stride [4 4] and cropping [1 1 1 1]	Transposed Convolution	54×54×512	Weights 4×4×512×512 Bias 1×1×512
33	rpnConv3x3 512 3x3x512 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	54×54×512	Weights 3×3×512×512 Bias 1×1×512
34	rpnRelu ReLU	ReLU	54×54×512	–
35	rpnConv1x1ClsScores 6 1x1x512 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	54×54×6	Weights 1×1×512×6 Bias 1×1×6
36	rpnConv1x1BoxDeltas 12 1x1x512 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	54×54×12	Weights 1×1×512×12 Bias 1×1×12
37	rpnBoxDeltas smooth-l1 loss	Box Regression Output	–	–
38	regionProposal region proposal with 3 anchor boxes	Region Proposal	1×5	–
39	roiPooling ROI Max Pooling with pooled output size [7 7]	ROI Max Pooling	7×7×512	–
40	fc6 4096 fully connected layer	Fully Connected	1×1×4096	Weights 4096×25088 Bias 4096×1
41	relu6 ReLU	ReLU	1×1×4096	–
42	drop6 50% dropout	Dropout	1×1×4096	–
43	fc7 4096 fully connected layer	Fully Connected	1×1×4096	Weights 4096×4096 Bias 4096×1
44	relu7 ReLU	ReLU	1×1×4096	–
45	drop7 50% dropout	Dropout	1×1×4096	–
46	rcnnFC 4 fully connected layer	Fully Connected	1×1×4	Weights 4×4096 Bias 4×1
47	prob softmax	Softmax	1×1×4	–
48	rcnnClassification crossentropyex with 'AD' and 3 other classes	Classification Output	–	–
49	rpnSoftmax rpn softmax	RPN Softmax	2916×3×2	–
50	rpnClassification cross-entropy loss with 'object' and 'background' classes	RPN Classification Output	–	–
51	fcBoxDeltas 12 fully connected layer	Fully Connected	1×1×12	Weights 12×4096 Bias 12×1
52	boxDeltas smooth-l1 loss	Box Regression Output	–	–

Figure 5. 12 The newNetwork+Faster R-CNN parameters after feature extraction layer

Furthermore, the upsampled feature map not only is helpful for the classification task of the RPN network but also significantly improves the regression of the bounding box. Figure 5.13 shows the examples of test results of newNetwork+Faster R-CNN. Compared with Figure 5.9, we find that the confidence for the hippocampal region contained in the bounding box increased from 0.99618 for AD, 0.98179 for CN and 0.98843 for MCI in VGG16+Faster R-CNN to 0.99932 for AD, 0.99955 for CN and 0.99946 for MCI in newNetwork+Faster R-CNN, respectively.

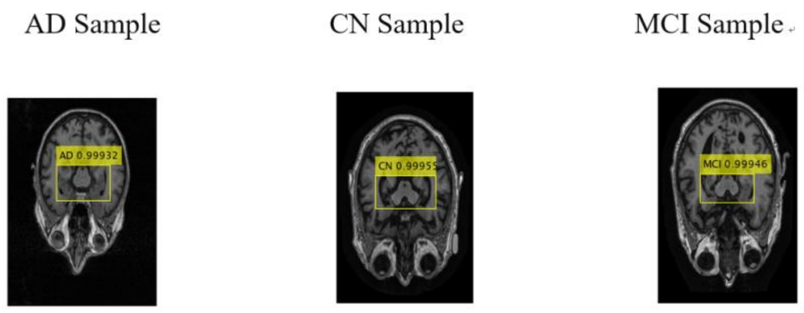


Figure 5. 13 The examples of test results of newNetwork+Faster R-CNN

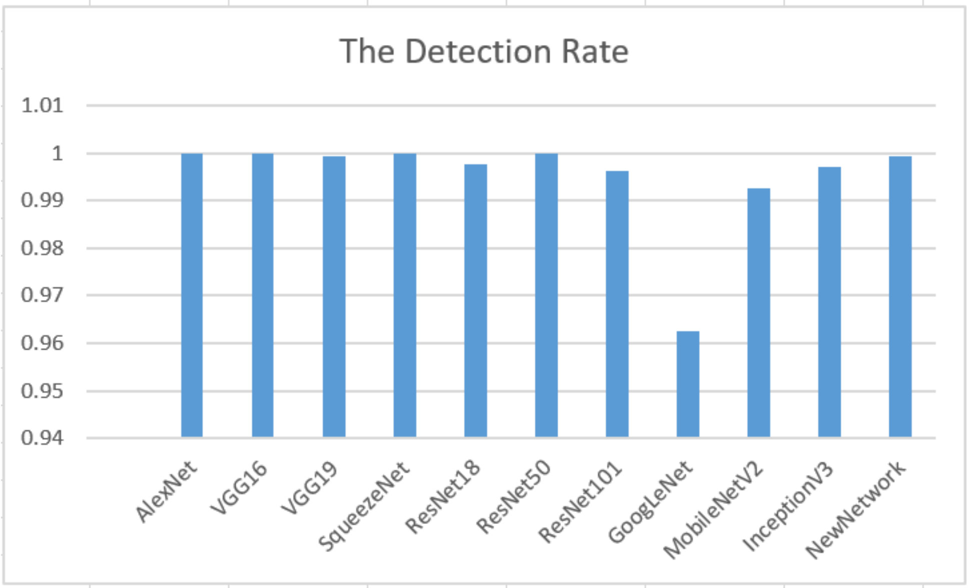


Figure 5. 14 Comparison of detection rates with different models

The object detection network faces the challenge that it cannot detect all objects in the image, the challenge in this experiment is that it cannot accurately detect the hippocampus region. The basic networks have distinct feature map extraction capabilities, which makes RPN networks detect objects with different capabilities when conducting bounding box regressions.

Figure 5.14 shows the comparison of detection rates with different models. Amongst all the models, only Faster R-CNN networks based on AlexNet, VGG16, SqueezeNet, and ResNet50 detected all images from the validation set. In the same network structure, as the network layer deepens, the detection rate decreases. For example, the detection rate of VGG19+Faster R-CNN is 99.93%, which is lower than 100% of VGG16+Faster R-CNN. Similarly, the detection rate of ResNet101+Faster R-CNN is 99.63%, which is lower than that of ResNet50+Faster R-CNN. Although newNetwork+Faster R-CNN only has one more transposed convolutional layer than VGG16+Faster R-CNN, its detection rate drops a little.

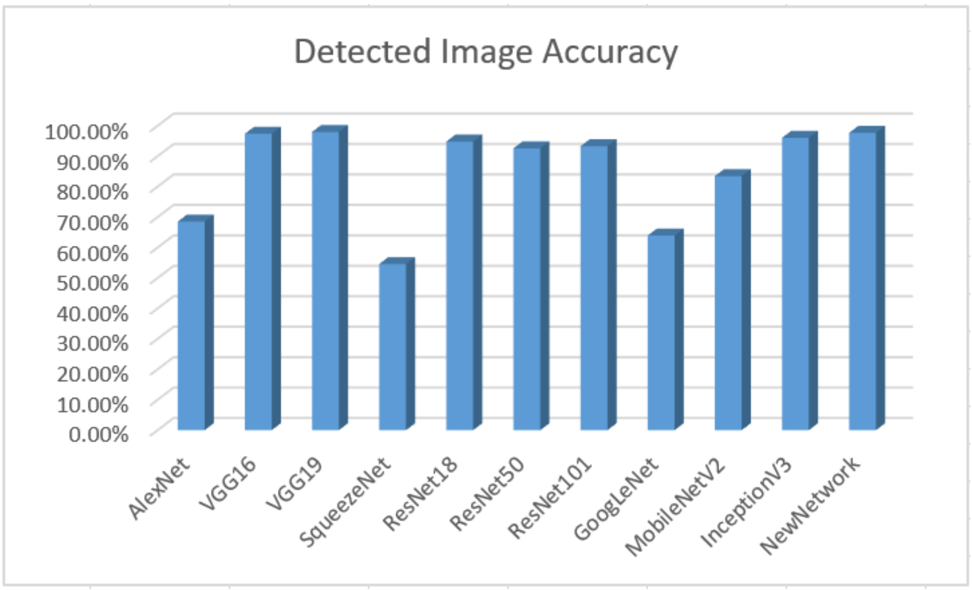


Figure 5. 15 Comparison of the accuracy of detected images with different models

The accuracy of all the detected images is calculated in this experiment. Figure 5.15 shows the comparison of the accuracy of detected images with different models. We see from the comparison that with the same basic network structure, as the network layer deepens, the accuracy of the detected image improves slightly.

For example, VGG19+Faster R-CNN with 16 convolutional layers only improves the detection accuracy of the image by 0.59% than the VGG16-Faster R-CNN with 13 convolutional layers, but the loss is the detection rate of the image. Similarly, the detection accuracy of ResNet101+Faster R-CNN is 0.72% higher than that of ResNet50+Faster R-CNN, but the detection rate of 0.37% is lost.

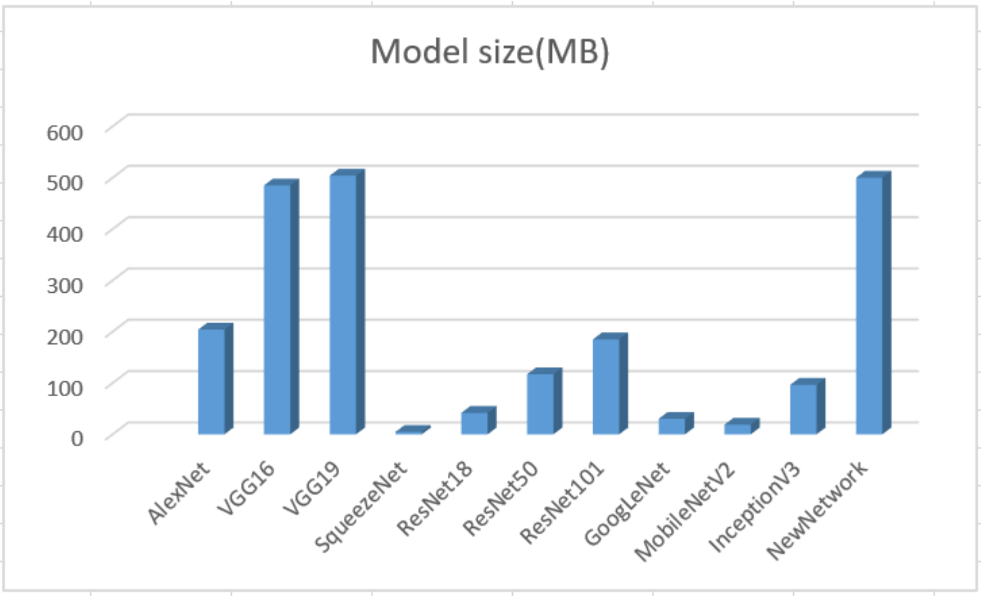


Figure 5. 16 The comparisons of sizes with different models

Figure 5.16 shows the comparison of size with different models. Due to the huge amount of basic network parameters based on the VGG structure, the three VGG-based models in this experiment have the largest size, almost all reach 500 MB. In contrast, other lightweight models are much smaller, such as SqueezeNet, GoogLeNet, and MobileNetv2. These models are easier to deploy and store, but they perform poorly on the dataset in this experiment. In addition, as the number of layers in the same network architecture deepens, the model size also increases.

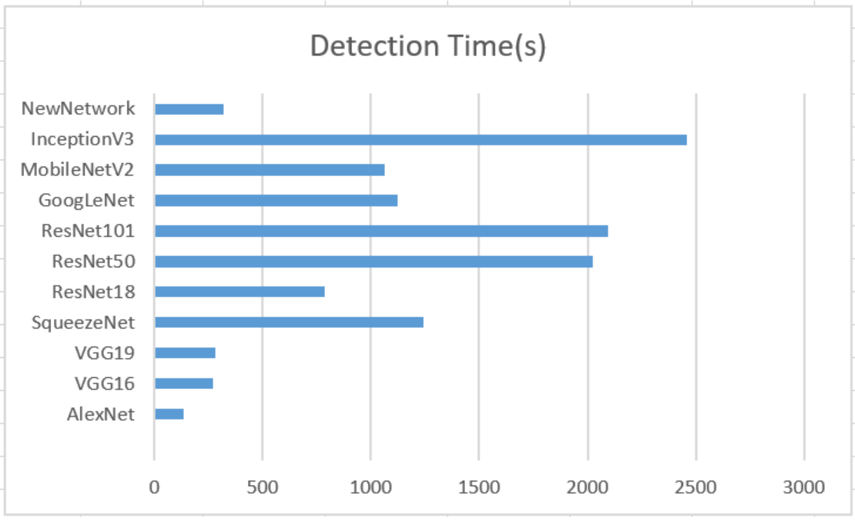


Figure 5. 17 The comparison of detection time with different models

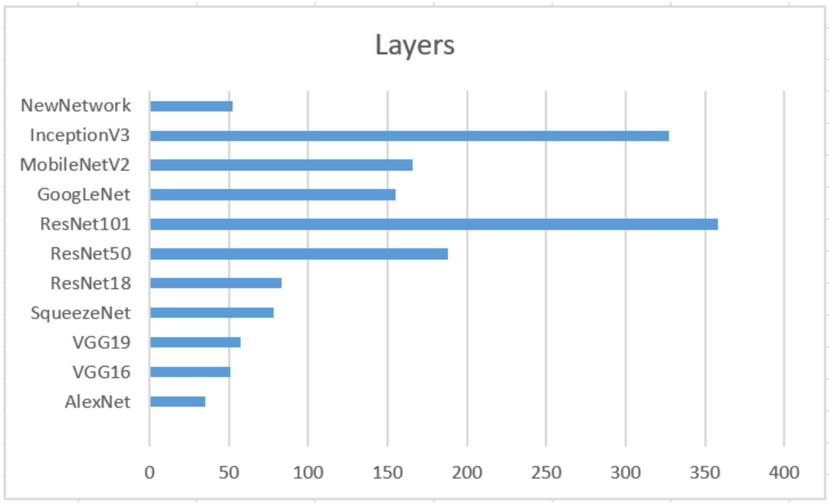


Figure 5. 18 The comparisons of layers with different models

In terms of the detection time of the validation set, it is closely related to the number of layers of the network. Figure 5.17 and Figure 5.18 show the comparisons of detection time and layers with various models. ResNet101+Faster R-CNN and Inceptionv3+Faster R-CNN have more than 300 network layers, the detection time is relatively long. In contrast, AlexNet+Faster R-CNN is the network having the lowest number of layers with only 35 and its detection time is also the shortest. In addition, the three detection networks based on the VGG structure are relatively few at the network layer, so they performed well in terms of detection time.

5.2 Discussions

AD is a progressive and degenerative neurological disease. The clinical symptom is that patients fall into dementia in their later years. Due to the increasing cost of nursing for AD, early accurate diagnosis is very important, and deep learning methods have made great contributions to the diagnosis of AD.

In this experiment, the MRI image set related to human heads downloaded from ADNI was imported as the raw data, the required data set was generated after preprocessing. According to the method described in Section 3.1, an AD detection model based on Faster R-CNN architecture was constructed and compared. MATLAB supports several basic networks for feature extraction. Our experiments show that the newNetwork+Faster R-CNN modified based on the VGG16+Faster R-CNN network can achieve 100% detection precision of AD samples and obtain 97.67% accuracy of the detected image.

Chapter 6

Conclusion and Future Work

In this project, a deep learning-based method for the early diagnosis of Alzheimer's disease is discussed, the research results and research method innovations are explained in detail. In this chapter, we also integrate and organize these conclusions into context and illustrate the limitations of this experiment, meanwhile point out our future work at the end of this thesis.

6.1 Conclusion

The purpose of this thesis is to study how to use deep learning to classify MRI images of human brain, to achieve early diagnosis and screening of Alzheimer's disease. The original MRI image downloaded from ADNI is a three-dimensional image, which is difficult to deal with traditional convolutional neural networks. Therefore, in this experiment, the 3D image was sliced to obtain 2D images.

After the slicing, the images containing hippocampus region are selected and labelled with the MATLAB image labelling tool so as to form a dataset. Faster R-CNN network for object detection was used in this experiment which can classify according to the characteristics of ROI so that it can focus on studying the atrophy of the hippocampus to diagnose Alzheimer's disease.

In our experiment, many basic networks for feature map extraction are used. From the experimental results, the newNetwork based on VGG16 adjustment has achieved good performance. It adds a transposed convolution layer after the relu5_3 layer of the original VGG16 as a new feature extraction layer for Faster R-CNN with the kernel size of 4×4, the number of filters of 512, the stride size of 4×4, and the cropping size of 1. It not only achieved relatively high image detection rate (only one image was not detected) and detection image accuracy but also obtained the highest detection precision for AD and MCI samples. In addition, confidence in the hippocampal region detected has also been improved.

6.2 Limitations

(1) In this experiment, the dataset used is slice images of 100 samples from each classification. The number of original samples is insufficient, which may affect the robustness of the model.

(2) The process of labelling the dataset is handled manually, which may generate the bias of the hippocampus region and affect the accuracy of the proposed model.

6.3 Future Work

We will work towards the research directions:

(1) Increasing the number of samples is needed in the dataset to improve the accuracy of the model.

(2) Only one slice image is selected for each sample, which makes the dataset more diverse, thereby improves the robustness of the model.

(3) We will continue to tune the network structure and develop new algorithms to get better performance.

References

Aël Chetelat, G., & Baron, J.-C. (2003). Early diagnosis of Alzheimer’s disease: Contribution of structural neuroimaging. *Neuroimage, 18*(2), 525-541.

Akiba, T., Suzuki, S., & Fukuda, K. (2017). Extremely large minibatch SGD: Training ResNet-50 on ImageNet in 15 minutes. *arXiv:1711.04325*, pp. 1-3.

Al-Sarayreh, M., Reis, M., Yan, W., & Klette, R. (2019) A sequential CNN approach for foreign object detection in hyperspectral images. CAIP (1) 2019: 271-283

Al-Sarayreh, M., Reis, M., Yan, W., & Klette, R. (2019) A sequential CNN approach for foreign object detection in hyperspectral images, CAIP, pp. 1-13

Alippi, C., Disabato, S., & Roveri, M. (2018). Moving convolutional neural networks to embedded systems: The AlexNet and VGG-16 case. In International Conference on Information Processing in Sensor Networks (IPSN), pp. 212-223.

Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., Asari, V. K. (2018). The history began from AlexNet: A comprehensive survey on deep learning approaches. *arXiv:1803.01164*, pp. 1-6.

Association, A. (2016). 2016 Alzheimer's disease facts and figures. *Alzheimer's & Dementia, 12*(4), 459-509.

Association, A. (2017). 2017 Alzheimer's disease facts and figures. *Alzheimer's & Dementia, 13*(4), 325-373.

Ba, J., & Frey, B. (2013). Adaptive dropout for training deep neural networks. In the Advances in Neural Information Processing Systems, pp. 3084-3092.

Batchelor, P. G., Smith, A. C., Hill, D. L. G., Hawkes, D. J., Cox, T. C. S., & Dean, A. (2002). Measures of folding applied to the development of the human fetal brain. *IEEE Transactions on Medical Imaging, 21*(8), 953-965.

Bayar, B., & Stamm, M. C. (2016). A deep learning approach to universal image manipulation detection using a new convolutional layer. In ACM Workshop on Information Hiding and Multimedia Security, pp. 5-10.

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning, 2*(1), 1-127.

Billones, C. D., Demetria, O. J. L. D., Hostallero, D. E. D., & Naval, P. C. (2016). DemNet: A convolutional neural network for the detection of Alzheimer's disease and mild cognitive impairment. *In* TENCON, pp. 3724-3727.

Braak, H., & Braak, E. (1995). Staging of Alzheimer's disease-related neurofibrillary changes. *Neurobiology of Aging, 16*(3), 271-278.

Brookmeyer, R., Johnson, E., Ziegler-Graham, K., & Arrighi, H. M. (2007). Forecasting the global burden of Alzheimer’s disease. *Alzheimer's & Dementia, 3*(3), 186-191.

Carin, L., & Pencina, M. J. (2018). On deep learning for medical image analysis. *JAMA, 320*(11), 1192-1193.

Chen, Z., Xie, Z., Zhang, W., & Xu, X. (2017). ResNet and Model Fusion for Automatic Spoofing Detection. In INTERSPEECH, pp. 102-106.

Ciernik, I. F., Dizendorf, E., Baumert, B. G., Reiner, B., Burger, C., Davis, J. B., & Von Schulthess, G. K. (2003). Radiation treatment planning with an integrated positron emission and computer tomography (PET/CT): A feasibility study. *International Journal of Radiation Oncology Biology Physics, 57*(3), 853-863.

Day, M., Horzinek, M., Schultz, R., & Squires, R. (2016). Guidelines for the vaccination of dogs and cats compiled by the Vaccination Guidelines Group (VGG) of the World Small Animal Veterinary Association (WSAVA). *Journal of Small Animal Practice, 57*, E1-E45.

de Vugt, M. E., & Verhey, F. R. (2013). The impact of early dementia diagnosis and intervention on informal caregivers. *Progress in Neurobiology, 110*, 54-62.

Domingos, P., & Pazzani, M. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning, 29*(2-3), 103-130.

Du, A., Schuff, N., Kramer, J., Ganzer, S., Zhu, X., Jagust, W., & Yaffe, K. (2004). Higher atrophy rate of entorhinal cortex than hippocampus in AD. *Neurology, 62*(3), 422-427.

Duara, R., Grady, C., Haxby, J., Sundaram, M., Cutler, N., Heston, L., & Rapoport, S. (1986). Positron emission tomography in Alzheimer's disease. *Neurology, 36*(7), 879-879.

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research, 12*(Jul), 2121-2159.

Dutta, A., & Zisserman, A. (2019). The VGG image annotator (VIA). *arXiv:1904.10699*.

Fairuz, S., Habaebi, M. H., & Elsheikh, E. M. A. (2018). Finger vein identification based on transfer learning of AlexNet. *In* International Conference on Computer and Communication Engineering (ICCCE), pp. 465-459.

Fenster, A., Downey, D. B., & Cardinal, H. N. (2001). Three-dimensional ultrasound imaging. *Physics in Medicine & Biology, 46*(5), R67.

Ferreri, F., Pauri, F., Pasqualetti, P., Fini, R., Dal Forno, G., & Rossini, P. M. (2003). Motor cortex excitability in Alzheimer's disease: A transcranial magnetic stimulation study. *Annals of Neurology: Official Journal of the American Neurological Association and the Child Neurology Society, 53*(1), 102-108.

Gal, Y., Hron, J., & Kendall, A. (2017). Concrete dropout. In the Advances in Neural

Information Processing Systems, pp. 3581-3590.

Gambhir, S. S. (2002). Molecular imaging of cancer with positron emission tomography. *Nature Reviews Cancer, 2*(9), 683-693.

Gao, H., Yuan, H., Wang, Z., & Ji, S. (2019). Pixel Transposed Convolutional Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1-3.

Gauthier, S., Reisberg, B., Zaudig, M., Petersen, R. C., Ritchie, K., Broich, K., & Chertkow, H. (2006). Mild cognitive impairment. *The Lancet, 367*(9518), 1262-1270.

Ge, R., & Shen, H. (2013). Researching on Feature Extraction of Brain CT Image. *International Journal of Signal Processing, Image Processing and Pattern Recognition, 6*(5), 39-48.

Geekiyanage, H., Jicha, G. A., Nelson, P. T., & Chan, C. (2012). Blood serum miRNA: non-invasive biomarkers for Alzheimer's disease. *Experimental Neurology, 235*(2), 491-496.

Giedd, J. N. (2004). Structural magnetic resonance imaging of the adolescent brain. *Annals of the New York Academy of Sciences, 1021*(1), 77-85.

Girshick, R. (2015). Fast R-CNN. In IEEE International Conference on Computer Vision, pp. 1440-1448.

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In IEEE Conference on Computer Vision and Pattern Recognition, pp. 580-587.

Giusti, A., Cireşan, D. C., Masci, J., Gambardella, L. M., & Schmidhuber, J. (2013). Fast image scanning with deep max-pooling convolutional neural networks. *In* IEEE International Conference on Image Processing, pp. 4034-4038.

Gottapu, R. D., & Dagli, C. H. (2018). DenseNet for anatomical brain segmentation. *Procedia Computer Science, 140*, 179-185.

Greenspan, H., Van Ginneken, B., & Summers, R. M. (2016). Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique. *IEEE Transactions on Medical Imaging, 35*(5), 1153-1159.

Hampel, H., Prvulovic, D., Teipel, S., Jessen, F., Luckhaus, C., Frölich, L., & Bertram, L. (2011). The future of Alzheimer's disease: The next 10 years. *Progress in Neurobiology, 95*(4), 718-728.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In IEEE Conference on Computer Vision and Pattern Recognition, pp. 770-778.

Hong-meng, L., Di, Z., & Xue-bin, C. (2017). Deep learning for early diagnosis of Alzheimer’s disease based on intensive AlexNet. *Computer Science, 44*(6), 50-59.

Hu, K., Zhang, Z., Niu, X., Zhang, Y., Cao, C., Xiao, F., & Gao, X. (2018). Retinal vessel segmentation of color fundus images using multiscale convolutional neural network with an improved cross-entropy loss function. *Neurocomputing, 309*, 179-191.

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In IEEE Conference on Computer Vision and Pattern Recognition, pp. 4700-4708.

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. *arXiv:1602.07360*, pp. 1-10.

Islam, J., & Zhang, Y. (2017). A novel deep learning based multi-class classification method for Alzheimer’s disease detection using brain MRI data. *In* International Conference on Brain Informatics, pp. 213-222.

Jensen, J. A., Nikolov, S. I., Gammelmark, K. L., & Pedersen, M. H. (2006). Synthetic aperture ultrasound imaging. *Ultrasonics, 44*.

Ji, H., Liu, Z., Yan, W. Q., & Klette, R. (2019). Early diagnosis of Alzheimer's disease using deep learning. In International Conference on Control and Computer Vision, pp. 87-91.

Ji, H., Liu, Z., Yan, W. Q., & Klette, R. (2019). Early diagnosis of Alzheimer's disease based on selective kernel network with spatial attention. In ACPR (2) 2019: 503-515

Jiang, H., & Learned-Miller, E. (2017). Face detection with the Faster R-CNN. *In* IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017), pp. 650-657.

Johnston, D., & Amaral, D. G. (2004). Hippocampus, In G. M. Shepherd (Ed.), The synaptic organization of the brain (pp. 455–498). Oxford University Press.

Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A convolutional neural network for modelling sentences. *arXiv:1404.2188*, pp. 1-9.

Karlawish, J., Jack Jr, C. R., Rocca, W. A., Snyder, H. M., & Carrillo, M. C. (2017). Alzheimer's disease: The next frontier—Special Report 2017. *Alzheimer's & Dementia, 13*(4), 374-380.

Killiany, R., Hyman, B., Gomez-Isla, T. a., Moss, M., Kikinis, R., Jolesz, F., & Albert, M. (2002). MRI measures of entorhinal cortex vs hippocampus in preclinical AD. *Neurology, 58*(8), 1188-1196.

Kingma, D. P., & Ba, J. (2014). ADAM: A method for stochastic optimization. *arXiv:1412.6980*, pp. 1-10.

Klein, S., Pluim, J. P., Staring, M., & Viergever, M. A. (2009). Adaptive stochastic gradient descent optimisation for image registration. *International Journal of Computer Vision, 81*(3), 227.

Kohavi, R., & Wolpert, D. H. (1996). Bias plus variance decomposition for zero-one loss functions. In ICML, pp. 275-283.

Kokkinos, I. (2017). UberNet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In IEEE Conference on Computer Vision and Pattern Recognition, pp. 6129-6238.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In the Advances in Neural Information Processing Systems, pp. 1097-1105.

Krupa, D. J., Wiest, M. C., Shuler, M. G., Laubach, M., & Nicolelis, M. A. (2004). Layer-specific somatosensory cortical activation during active tactile discrimination. *Science, 304*(5679), 1989-1992.

Le Roux, N., & Bengio, Y. (2008). Representational power of restricted Boltzmann machines and deep belief networks. *Neural Computation, 20*(6), 1631-1649.

LeCun, Y., & Bengio, Y. (1995). Convolutional networks for images, speech, and time series. *The Handbook of Brain Theory and Neural Networks, 3361*(10), 1995.

Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., & Wang, Z. (2017). Photo-realistic single image super-resolution using a generative adversarial network. In IEEE Conference on Computer Vision and Pattern Recognition, pp. 4681-4690.

Lee, J.-G., Jun, S., Cho, Y.-W., Lee, H., Kim, G. B., Seo, J. B., & Kim, N. (2017). Deep learning in medical imaging: General overview. *Korean Journal of Radiology, 18*(4), 570-584.

Leifer, B. P. (2003). Early diagnosis of Alzheimer's disease: Clinical and economic benefits. *Journal of the American Geriatrics Society, 51*(5s2), S281-S288.

Liu,X., Neuyen, M., & Yan, W. (2019) Vehicle-related scene understanding using deep learning. In ACPR Workshops, pp. 61-73

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). Ssd: Single shot multibox detector. In European Conference on Computer Vision, pp. 21-37.

Liu, Z., Yan, W. Q., & Yang, M. L. (2018). Image denoising based on a CNN model. In International Conference on Control, Automation and Robotics (ICCAR), pp. 389-393.

Loshchilov, I., & Hutter, F. (2016). SGDR: Stochastic gradient descent with warm restarts. *arXiv:1608.03983*, pp. 1-10.

Lu, J., Yan, W., & Nguyen, M. (2018) Human behaviour recognition using deep learning. IEEE AVSS, pp.1-6

Luo, X., Zang, X., Yang, L., Huang, J., Liang, F., Rodriguez-Canales, J., & Xiao, G. (2017). Comprehensive computational pathological image analysis predicts lung cancer prognosis. *Journal of Thoracic Oncology, 12*(3), 501-509.

Marchesini, S., He, H., Chapman, H. N., Hau-Riege, S. P., Noy, A., Howells, M. R., & Spence, J. C. (2003). X-ray image reconstruction from a diffraction pattern alone. *Physical Review B, 68*(14), 140101.

Miller, D. D., & Brown, E. W. (2018). Artificial intelligence in medical practice: The question to the answer? *The American Journal of Medicine, 131*(2), 129-133.

Minhas, R. A., Javed, A., Irtaza, A., Mahmood, M. T., & Joo, Y. B. (2019). Shot classification of field sports videos using alexnet convolutional neural network. *Applied Sciences, 9*(3), 483.

Mohamed, A. A., Berg, W. A., Peng, H., Luo, Y., Jankowitz, R. C., & Wu, S. (2018). A deep learning method for classifying mammographic breast density categories. *Medical Physics, 45*(1), 314-321.

Muhammad, N. A., Ab Nasir, A., Ibrahim, Z., & Sabri, N. (2018). Evaluation of CNN, Alexnet and GoogleNet for Fruit Recognition. *Indonesian Journal of Electrical Engineering and Computer Science, 12*(2), 468-475.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*: MIT press, pp. 1-5.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In International Conference on Machine Learning (ICML-10), pp. 807-814.

Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., & Ng, A. Y. (2011). Multimodal deep learning, In ICML, pp. 1-7.

Ogawa, S., Lee, T.-M., Kay, A. R., & Tank, D. W. (1990). Brain magnetic resonance imaging with contrast dependent on blood oxygenation. In *Proceedings of the National Academy of Sciences, 87*(24), 9868-9872.

Ortiz, A., Munilla, J., Gorriz, J. M., & Ramirez, J. (2016). Ensembles of deep learning architectures for the early diagnosis of the Alzheimer's disease. *International Journal of Neural Systems, 26*(07), 1650025.

Pan, C., Li, X., & Yan, W. (2018) A learning-based positive feedback approach in salient object detection. IEEE IVCNZ, pp.1-6

Patino-Saucedo, A., Rostro-Gonzalez, H., & Conradt, J. (2018). Tropical fruits classification using an AlexNet-type convolutional neural network and image augmentation. *In* International Conference on Neural Information Processing, pp. 371-379.

Pazo-Alvarez, P., Amenedo, E., & Cadaveira, F. (2004). Automatic detection of motion direction changes in the human brain. *European Journal of Neuroscience, 19*(7), 1978-1986.

Pennanen, C., Kivipelto, M., Tuomainen, S., Hartikainen, P., Hänninen, T., Laakso, M. P., & Helkala, E.-L. (2004). Hippocampus and entorhinal cortex in mild cognitive impairment and early AD. *Neurobiology of Aging, 25*(3), 303-310.

Poldrack, R. A. (2007). Region of interest analysis for fMRI. *Social Cognitive and Affective Neuroscience, 2*(1), 67-70.

Querbes, O., Aubry, F., Pariente, J., Lotterie, J.-A., Démonet, J.-F., Duret, V., & Celsis, P. (2009). Early diagnosis of Alzheimer's disease using cortical thickness: Impact of cognitive reserve. *Brain, 132*(8), 2036-2047.

Razzak, M. I., Naz, S., & Zaib, A. (2018). Deep learning for medical image processing: Overview, challenges and the future. In *Classification in BioApps* (pp. 323-350): Springer.

Reddi, S. J., Kale, S., & Kumar, S. (2019). On the convergence of adam and beyond. *arXiv:1904.09237*, pp. 1-3.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, real-time object detection. In IEEE Conference on Computer Vision and Pattern Recognition, pp. 779-788.

Reitz, C., Brayne, C., & Mayeux, R. (2011). Epidemiology of Alzheimer disease. *Nature Reviews Neurology, 7*(3), 137.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In the Advances in Neural Information Processing Systems, pp. 91-99.

Ren, Y., Nguyen, M., & Yan, W. (2018) Real-time recognition of series seven New

Zealand banknotes. IJDCF 10(3): 50-65.

Ruiz, A., Sertel, O., Ujaldon, M., Catalyurek, U., Saltz, J., & Gurcan, M. (2007). Pathological image analysis using the GPU: Stroma classification for neuroblastoma. In IEEE International Conference on Bioinformatics and Biomedicine (BIBM 2007), pp. 78-88.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *arXiv:1801.04381*, pp. 1-5.

Schladt, T. D., Schneider, K., Schild, H., & Tremel, W. (2011). Synthesis and bio-functionalization of magnetic nanoparticles for medical diagnosis and treatment. *Dalton Transactions, 40*(24), 6315-6343.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks, 61*, 85-117.

Selkoe, D. J. (2001). Alzheimer's disease: genes, proteins, and therapy. *Physiological Reviews, 81*(2), 741-766.

Sengupta, A., Ye, Y., Wang, R., Liu, C., & Roy, K. (2019). Going deeper in spiking neural networks: VGG and residual architectures. *Frontiers in Neuroscience, 13*.

Senohradski, K., Karovic, B., & Miric, D. (2001). Computer tomography in the diagnosis and therapy of acetabular fractures. *Srpski Arhiv Za Celokupno Lekarstvo, 129*(7-8), 194-198.

Shamir, L., Ling, S. M., Scott Jr, W. W., Bos, A., Orlov, N., Macura, T. J., & Goldberg, I. G. (2008). Knee X-ray image analysis method for automated detection of osteoarthritis. *IEEE Transactions on Biomedical Engineering, 56*(2), 407-415.

Shen, D., Xin, C. Nguyen, M., & Yan, W. (2018) Flame detection using deep learning, IEEE ICCAR.

Shen, D., Wu, G., & Suk, H.-I. (2017). Deep learning in medical image analysis. *Annual Review of Biomedical Engineering, 19*, 221-248.

Shen, Y., & Yan, W. (2018) Blind spot monitoring using deep learning. IEEE IVCNZ,pp. 1-5

Shen, Y., Harris, N. C., Skirlo, S., Prabhu, M., Baehr-Jones, T., Hochberg, & M., Englund, D. (2017). Deep learning with coherent nanophotonic circuits. *Nature Photonics, 11*(7), 441.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, pp. 1-5.

Song, C., He, L., Yan, W., & Nand, P. (2019) An improved selective facial extraction model for age estimation. In IEEE IVCNZ, pp. 1-6

Su, Y., Sun, W., Liu, J., Zhai, G., & Jing, P. (2019). Photo-realistic image bit-depth enhancement via residual transposed convolutional neural network. *Neurocomputing, 347*, 200-211.

Sun, C., Shrivastava, A., Singh, S., & Gupta, A. (2017). Revisiting unreasonable effectiveness of data in deep learning era. In IEEE International Conference on Computer Vision, pp. 843-852.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In IEEE Conference on Computer Vision and Pattern Recognition, pp. 2818-2826.

Tang, Z., Jiang, W., Zhang, Z., Zhao, M., Zhang, L., & Wang, M. (2019). DenseNet with Up-Sampling block for recognizing texts in images. *Neural Computing and Applications*, 1-9.

Tejani-Butt, S. M., Yang, J., & Pawlyk, A. C. (1995). Altered serotonin transporter sites in Alzheimer's disease raphe and hippocampus. *Neuroreport, 6*(8), 1207-1210.

Ullah, A., Xie, H., Farooq, M. O., & Sun, Z. (2018). Pedestrian detection in infrared images using Fast R-CNN. In International Conference on Image Processing Theory, Tools and Applications (IPTA), pp. 1-6.

Wang, J., Bacic, B., & Yan, W. (2018) An effective method for plate number recognition. Multimedia Tools and Applications, 77 (2), pp.1679-1692

Wang, X., Shrivastava, A., & Gupta, A. (2017). A Fast R-CNN: Hard positive generation via adversary for object detection. In IEEE Conference on Computer Vision and Pattern Recognition, pp. 2606-2615.

Wang, X., & Yan, W (2020) Human gait recognition based on frame-by-frame gait energy images and convolutional long short-term memory. Int. J. Neural Syst. 30(1): 1950027:1-1950027:12

Wang, X., & Yan, W (2020) Cross-view gait recognition through ensemble learning. Neural Computing and Applications 32(11): 7275-7287

Wang, X., & Yan, W. (2019) Multi-perspective gait recognition based on ensemble learning. Springer Neural Computing and Applications.

Winslow, B. T., Onysko, M. K., Stob, C. M., & Hazlewood, K. A. (2011). Treatment of Alzheimer disease. *American Family Physician, 83*(12), 1403.

Wu, M., & Chen, L. (2015). Image recognition based on deep learning. In Chinese Automation Congress (CAC), pp. 542-546.

Wu, R., Yan, S., Shan, Y., Dang, Q., & Sun, G. (2015). Deep image: Scaling up image recognition. *arXiv:1501.02876, 7*(8).

Wu, Z., Shen, C., & Van Den Hengel, A. (2019). Wider or deeper: Revisiting the ResNet model for visual recognition. *Pattern Recognition, 90*, 119-133.

Xu, Q., Zhang, M., Gu, Z., & Pan, G. (2019). Overfitting remedy by sparsifying regularization on fully-connected layers of CNNs. *Neurocomputing, 328*, 69-74.

Xu, Y., Jack, C., O'brien, P., Kokmen, E., Smith, G., Ivnik, R., & Petersen, R. C. (2000). Usefulness of MRI measures of entorhinal cortex versus hippocampus in AD. *Neurology, 54*(9), 1760-1767.

Xu, Y., Mo, T., Feng, Q., Zhong, P., Lai, M., Eric, I., & Chang, C. (2014). Deep learning of feature representation with multiple instance learning for medical image analysis. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1626-1630.

Yan, L., Yoshua, B., & Geoffrey, H. (2015). Deep learning. *Nature, 521*(7553), 436-444.

Yan, W. (2019) Introduction to Intelligent Surveillance - Surveillance Data Capture, Transmission, and Analytics, Third Edition. Springer.

Yu, D., Wang, H., Chen, P., & Wei, Z. (2014). Mixed pooling for convolutional neural networks. In International Conference on Rough Sets and Knowledge Technology, pp. 364-375.

Zeiler, M. D. (2012). Adadelta: An adaptive learning rate method. *arXiv:1212.5701.*

Zeiler, M. D., Ranzato, M., Monga, R., Mao, M., Yang, K., Le, Q. V., & Dean, J. (2013). On rectified linear units for speech processing. In IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 3517-3521.

Zhang, C.-L., Luo, J.-H., Wei, X.-S., & Wu, J. (2017). In defense of fully connected layers in visual representation transfer. In Pacific Rim Conference on Multimedia, pp. 807-817.

Zhang, J., Lu, C., Li, X., Kim, H.-J., & Wang, J. (2019). A full convolutional network based on DenseNet for remote sensing scene classification. *Math. Biosci. Eng, 16*(5), 3345-3367.

Zhang, L., Lin, L., Liang, X., & He, K. (2016). Is Faster R-CNN doing well for pedestrian detection? In European Conference on Computer Vision, pp. 443-457.

Zhang, Q., & Yan, W. (2018) Currency detection and recognition based on deep learning. IEEE AVSS, pp. 1-6

Zheng, K., Yan, W., & Nand, P. (2018) Video dynamics detection using deep neural networks. IEEE Trans. Emerging Topics in Comput. Intellig. 2(3): 224-234

Zhu, Y., & Newsam, S. (2017). Densenet for dense flow. *In* IEEE International Conference on Image Processing (ICIP), pp. 790-794.