

# **OBJECT DETECTION IN DEEP LEARNING**

by

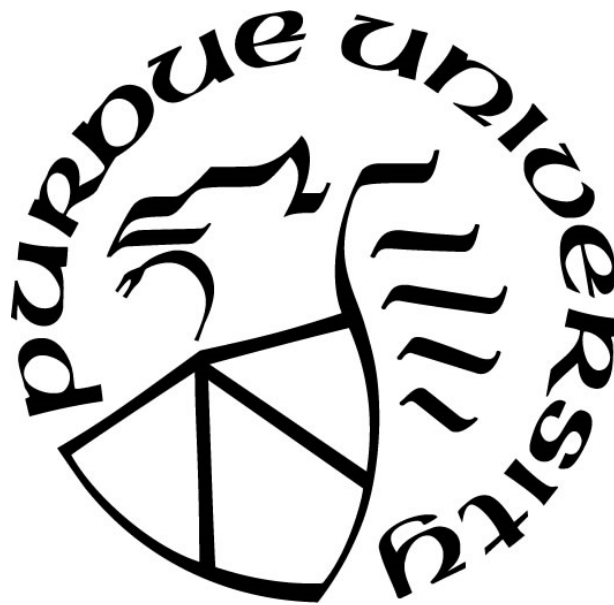
**Haoyu Shi**

**A Thesis**

*Submitted to the Faculty of Purdue University*

*In Partial Fulfillment of the Requirements for the degree of*

**Master of Science in Electrical and Computer Engineering**



Department of Electrical and Computer Engineering

Hammond, Indiana

December 2019

**THE PURDUE UNIVERSITY GRADUATE SCHOOL**  
**STATEMENT OF COMMITTEE APPROVAL**

**Dr. Lizhe Tan, Chair**

Department of Electrical and Computer Engineering

**Dr. Quamar Niyaz**

Department of Electrical and Computer Engineering

**Dr. Xiaoli Yang**

Department of Electrical and Computer Engineering

**Approved by:**

Dr. Vijay Devabhaktuni

Chair, Department of Electrical and Computer Engineering

## **ACKNOWLEDGMENTS**

First of all, I would like to express my gratitude to all professors, classmates and friends who helped and supported me for conducting this thesis research. I would like to thank my academic advisor, Professor Lizhe Tan, who gave me guidance and suggestions so that I can continuously improve and achieve my goals.

Thanks also go to my thesis committee members, Professor Quamar Niyaz and Professor Xiaoli Yang. Professor Quamar Niyaz gave me a lot of help and discussed with me academic problems and some difficulties encountered in my life. Professor Xiaoli Yang also guided me on research, which benefited me a lot.

Help and companionship between the students and friends also made me feel happy over this two-year time period. Thanks to my influential friend, Sumendra, who is elder than me and shares many experiences with me.

Finally, I would like to thank my parents for giving me life and making me happy in this world. They support me not only mentally but also financially. They encourage me to anything about my life and they are the most solid and warm power behind me.

# TABLE OF CONTENTS

LIST OF TABLES .....	7
LIST OF FIGURES .....	8
LIST OF ABBREVIATIONS .....	10
ABSTRACT.....	12
1. INTRODUCTION .....	13
1.1 The Original of Deep of Learning .....	13
1.2 Deep Learning in Computer Vision.....	13
1.2.1 Object Detection and Recognition.....	14
1.2.2 Image Semantic Segmentation .....	14
1.2.3 Track Movement.....	14
1.2.4 Visual Question Answering.....	15
1.2.5 3D Reconstruction .....	15
1.3 Objectives .....	15
1.4 Organization of Thesis.....	16
2. BACKGROUND .....	17
2.1 Traditional Machine Learning.....	17
2.1.1 Logistic Regression .....	17
2.1.2 Softmax Regression.....	18
2.2 Neural Networks .....	19
2.2.1 Introduction.....	19
2.2.2 Components .....	20
2.2.3 Mathematical Expression.....	22
2.2.4 Common Activation Functions.....	22
2.2.5 Update Neurons .....	26
2.3 Convolutional Neural Networks (CNN) Background.....	32
2.4 Convolutional Neural Networks (CNN) Definition.....	34
2.5 Convolutional Neural Networks Architecture .....	35
2.5.1 Convolutional Layer .....	35

2.5.2	Activation Layer .....	39
2.5.3	Pooling Layer.....	39
2.5.4	Fully Connected Layer .....	40
2.5.5	Batch Normalization Layer.....	40
2.6	The Neural Network Models History .....	41
2.6.1	Image Classification .....	41
2.6.2	Object Detection .....	50
3.	METHODOLOGY .....	56
3.1	Preparation of Deep Learning Environment .....	56
3.1.1	Operating System.....	56
3.1.2	NVIDIA Support.....	56
3.1.3	Tools .....	56
3.2	Motivation.....	57
3.2.1	For parking lot classification .....	57
3.2.2	For prostate detection and localization .....	58
3.3	Dataset Image Processing .....	59
3.3.1	Convolutional Neural Network for Classification .....	59
3.3.2	YOLO Network for Object Detection and Localization.....	60
3.4	Classification Network Implementation .....	61
3.4.1	Architecture .....	61
3.4.2	Loss Function.....	61
3.4.3	Evaluation Metrics in Classification Network.....	62
3.5	YOLO Model Implementation.....	63
3.5.1	IOU Introduction .....	63
3.5.2	Non-Maximum Suppression.....	64
3.5.3	Encoding Method in Pre-Processing .....	64
3.5.4	Loss Function.....	66
3.5.5	Decoding Method in Prediction.....	67
3.5.6	Evaluation of Network using mAP .....	68
3.6	Customization of YOLO Model .....	68
4.	EVALUATION AND DISCUSSION .....	69

4.1	Parking Lot Output .....	69
4.2	Prostate Output.....	71
4.3	Future Work .....	73
5.	CONCLUSION.....	75
	REFERENCES .....	76

## LIST OF TABLES

Table 2-1. The VGG network architecture [26]. Conv3-64 stands for convolutional layer with 3x3 kernel and output depth is 64. FC-4096 stands for fully connected layer with 4096 output depth. ....	44
Table 3-1. Confusion matrix .....	62
Table 3-2. The performance of two YOLO network. ....	68
Table 4-1. Precision and recall at first training.....	69
Table 4-2. Precision and recall at second training. ....	71
Table 4-3. False Positive for network that predicted 6 contained prostate images as non-contained prostate images, and location information missed.....	73

## LIST OF FIGURES

Figure 2.1. Sigmoid function. ....	18
Figure 2.2. Neural structure. ....	20
Figure 2.3. Warren McCulloch (left) and mathematician Walter Pitts (right) [9]. ....	21
Figure 2.4. The simple neuron model. ....	21
Figure 2.5. The linear function. ....	23
Figure 2.6. The sigmoid function. ....	23
Figure 2.7. The Tanh function. ....	24
Figure 2.8. The ReLU function. ....	25
Figure 2.9. The LReLU function. ....	26
Figure 2.10. The neural network structure. ....	27
Figure 2.11. The neural network parameters. ....	31
Figure 2.12. The neural network structure with 15 neurons in hidden layer [13]. ....	33
Figure 2.13. The common convolutional neural network architecture. ....	35
Figure 2.14. The input image pixels (left), the filter/kernel (middle), the output feature map (right). ....	36
Figure 2.15. The first convolutional calculation step. ....	37
Figure 2.16. The last convolutional calculation step. ....	37
Figure 2.17. The max pooling. ....	40
Figure 2.18. The Alex Network Architecture [23]. ....	42
Figure 2.19. The Inception Structure. Inception naïve module (left), Inception module with dimension reductions (right). ....	45
Figure 2.20. The Inception naïve version reduces parameters. ....	46
Figure 2.21. The inception dimension version reduces parameters. ....	47
Figure 2.22. The GoogleLeNet Architecture [27]. ....	47
Figure 2.23. The core idea about ResNet [28]. ....	48
Figure 2.24. The SENet Architecture. ....	50
Figure 2.25. The YOLO architecture [4]. ....	53
Figure 2.26. The FPN architecture. ....	54



Figure 3.1. The rotation method. ....	59
Figure 3.2. The extracted prostate images. ....	60
Figure 3.3. The customized convolutional neural network.....	61
Figure 3.4. The Convolutional Neural Network with BN layer. ....	61
Figure 3.5. The IOU calculation. ....	63
Figure 3.6 The Non-maximum example. The multiple bounding boxes (left), the optimal bounding box (right). ....	64
Figure 3.7. The Encode method.....	65
Figure 4.1 Training loss (left) and Validation dataset accuracy (right).....	69
Figure 4.2. Wrong detection images .....	70
Figure 4.3. Training loss (left) and Validation dataset accuracy (right) at second time.....	70
Figure 4.4. The first training for 50 epochs, the blue line (1) is the training data loss, the orange line (2) is the validation data loss. ....	71
Figure 4.5. The training for 5000 epochs, the blue line (1) is the training data loss, the orange line (2) is the validation data loss. ....	72

## LIST OF ABBREVIATIONS

GPU	Graphics Processing Unit
CNN	Convolutional Neural Networks
R-CNN	Region proposals & CNN
YOLO	You Only Look Once
AI	Artificial Intelligence
HOG	Histogram of Gradient
SIFT	Scale-Invariant Feature Transform
VQA	Visual Question Answering
NLP	Natural Language Processing
3D	3-Dimensional
SIMD	Single Instruction Multiple Data
MIMD	Multiple Instruction Multiple Data
MPN	McCulloch-Pitts Neuron
ReLU	Rectified Linear Unit
LReLU	Leaky ReLU
BGD	Batch Gradient Descent
SGD	Stochastic Gradient Descent
MBGD	Mini-Batch Gradient Descent
BP	BackPropagation
SIANN	Shifted-Invariant Artificial Neural Network
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
SVM	Support Vector Machine
ROI	Regions of Interest
RPN	Region Proposal Networks
R-FCN	Region-based Fully Convolutional Networks
SSD	Single Shot MultiBox Detector
FPN	Feature Pyramid Network

CE	Cross-Entropy
CUDA	Computer Unified Device Architecture
IOU	Intersection over Union
NMS	Non-Maximum Suppression
mAP	Mean Average Precision
SPP	Spatial Pyramid Pooling
FPS	Frame Per Second

## **ABSTRACT**

Through the computing advance and GPU (Graphics Processing Unit) availability for math calculation, the deep learning field becomes more popular and prevalent. Object detection with deep learning, which is the part of image processing, plays an important role in automatic vehicle drive and computer vision. Object detection includes object localization and object classification. Object localization involves that the computer looks through the image and gives the correct coordinates to localize the object. Object classification is that the computer classification targets into different categories. The traditional image object detection pipeline idea is from Fast/Faster R-CNN [32] [58]. The region proposal network generates the contained objects areas and put them into classifier. The first step is the object localization while the second step is the object classification. The time cost for this pipeline function is not efficient. Aiming to address this problem, You Only Look Once (YOLO) [4] network is born. YOLO is the single neural network end-to-end pipeline with the image processing speed being 45 frames per second in real time for network prediction. In this thesis, the convolution neural networks are introduced, including the state of art convolutional neural networks in recently years. YOLO implementation details are illustrated step by step. We adopt the YOLO network for our applications since the YOLO network has the faster convergence rate in training and provides high accuracy and it is the end to end architecture, which makes networks easy to optimize and train.

**Keywords:** Deep Learning, Object Detection, YOLO.

# **1. INTRODUCTION**

## **1.1 The Original of Deep of Learning**

Artificial intelligence (AI) [1], machine learning and deep learning are not three separate things, they are closely related each other. The artificial Intelligence is the biggest part, which contains cognitive capabilities, such as solving problems and learning from examples. AI consists of narrow AI, general AI and Strong AI. Narrow AI deals with scenarios in which the machine can do better than human in a specific task. General AI copes with scenarios like human in any intellectual task. Strong AI is in the area that the machine or computer can do much better than human in many tasks.

Machine learning is a subset of AI. The definition, from Arthur Samuel, one of the pioneers of machine learning, is “field of study that gives computers the ability to learn without being explicitly programmed.” Machine learning has two areas, that is, supervised learning which should have at least two-dimensional inputs, input and output; and unsupervised learning which only need input data. In the latter case the machine learns from the input and finds the regular pattern.

Deep learning [12] [19] is a subset of machine learning. Deep learning mainly indicates deep neural networks learning. Deep learning has input layer, hidden layer and output layer. Deep learning network is heavily used in image recognition. With a tremendous increase in computer computing power and the developed convolution neural networks, the hidden layers in deep learning network can have more depth than before. Because of this, the accuracy and speed by using deep learning can be significantly enhanced.

## **1.2 Deep Learning in Computer Vision**

Computer vision or machine vision is that computers or related devices simulate the human vision. Computer processes images or videos to obtain the information that human can easily understand and describe. Aiming to address this, scientists teach a computer or machine to learn a way to deal with human vision. At the beginning, scientists used the histogram of gradient, HOG [2] and scale-invariant feature transform SIFT [3], the traditional hand-crafted feature, with shallow-layer model

to teach the computer or machine to understand images and videos. With a rapid development of deep learning, convolutional neural network (CNN) is evolved and becomes a new representative method in computer vision. The definition of computer vision is complicated, and it is an interdisciplinary area, including artificial intelligence, digital image processing, machine learning, deep learning, pattern recognition, probabilistic graph models, scientific calculations and a series of mathematical calculations. There are many implementations which will be introduced as following.

### **1.2.1 Object Detection and Recognition**

Object detection is our main and basic research direction. Depending on the image or a frame in video, the algorithm can find common objects in it, and furthermore can give the location information about objects along with even making a category for different objects. A typical application with a detection algorithm includes face detection for security identification and vehicle plate detection for a paid parking lot.

### **1.2.2 Image Semantic Segmentation**

Image segmentation is an analysis task in image processing. The computer processes the image using a segmentation algorithm to determine a segmented area for class label, such as person, road, sky, car, flower, dog, tree or ocean. It is pixel level recognition for object detection.

### **1.2.3 Track Movement**

Track movement is also one of the basic problems in the field of computer vision. In recent years, it has been developed rapidly. The method has also been extended from the past low-depth algorithm to the deep learning algorithm, and the precision is getting higher and higher. However, the accuracy of real-time deep learning in a tracking algorithm has been difficult to improve. The tracking algorithm with very high precision runs very slow, so it is difficult to come in handy in practical applications. Visual tracking refers to detecting, extracting, recognizing and tracking moving targets in image sequences, and obtaining motion parameters of moving targets, such as position, velocity, acceleration and motion trajectory, so as to carry out the next processing and

analysis to achieve the motion. The behavior of the target is understood to complete a higher level of detection for task tracking.

Currently, tracking movement is particularly a hot research direction in athlete sports posture detection, aiming to give athletes suggestions in the training period.

#### **1.2.4 Visual Question Answering**

Visual Question Answering referred to as VQA, is a very popular direction in recent years. Generally speaking, the VQA system needs to take pictures and questions as input and combines these two parts of information to produce a human language as an output. For a specific picture, if we wonder the machine to answer a specific question about the picture in Natural Language Processing (NLP), we need to make the machine to have certain content, the meaning, the intention of the question, and related common-sense understandings. In terms of its nature, this is a multidisciplinary research issue.

#### **1.2.5 3D Reconstruction**

Vision-based three-dimensional reconstruction refers to acquiring the image data of the scene object through the camera, analyzing and processing the image, and combining computer vision knowledge to derive a three-dimensional information of an object in a real environment. The focus of 3D reconstruction technology is how to obtain the depth information of a target scene or object. Under the condition that the depth information of the scene is known, only the location and integration of the point data are needed to achieve the 3D reconstruction of the scene.

### **1.3 Objectives**

This thesis research is dedicated to investigating the development of deep learning techniques including the most popular network structures; then study the YOLO neural network architecture with modification. Finally, a new deep learning optimized architecture will be proposed, and its performance improvement will be demonstrated.

## 1.4 Organization of Thesis

This thesis is organized into five parts.

Part 1: Introduction, the artificial intelligence development is introduced, and several fields related with computer vision in deep learning are briefly described.

Part 2: Background, convolutional neural network [18] which is heavily used in deep learning field is introduced, including convolutional neural networks architecture, the difference and advantage over multilayer perceptions. Then the state-of-the-art architectures commonly used in convolutional neural networks are described. After this, the main part is the review models of object detection networks which are very widely used. These most popular models are the basic knowledge of this thesis work.

Part 3: Methodology, this is the details about thesis experiment and implementation. The first task is to prepare an environment which supports a deep learning platform and to verify codes. Then collection and collation of dataset comes to the beginning of deep learning. Due to the limited dataset, it is especially important to make data augmentation. When data is ready, the simple convolutional neural networks is implemented to achieve image classification with the cutting-edge model You Only Look Once, YOLO [4]. Based on our own dataset, the YOLO architecture details are customized to achieve our own task.

Part 4: Evaluation and Discussion part, we demonstrate our output information and discuss our designed model shortage.

Part 5: Conclusion, the summary results of this research work are presented and conclusions are drawn in this chapter.



## 2. BACKGROUND

### 2.1 Traditional Machine Learning

#### 2.1.1 Logistic Regression

When we mention about the Logistic Regression, we must review the linear regression [5]. Both of them could do prediction from the dataset. We will briefly introduce the linear regression.

The linear regression is a equation could make one or multiple variables to regress to a linear function using a least squares method. Dataset is denoted by  $\{y^1, y^2, y^3, \dots, y^n\}$  corresponding to  $\{[x_1^1, x_2^1, x_3^1, \dots, x_m^1], [x_1^2, x_2^2, x_3^2, \dots, x_m^2], \dots, [x_1^n, x_2^n, x_3^n, \dots, x_m^n]\}$ . Note that  $n$  is the number of samples,  $m$  is the number of variables, and  $y \in \{0, 1\}$ . Thus, the formula takes the form

$$y_i = \theta_0 + \theta_1 x_1^i + \dots + \theta_m x_m^i + e_i = \theta^T x^i + e_i, \quad i = 1, \dots, n \quad (1)$$

Again, notice that  $y$  is not only influenced by  $x$ , but is also influenced by a random error. This formula is the straight line in the Cartesian coordinate system. Due to this feature, the linear regression could separate one dataset into two different areas which is the idea of the logistic regression. After using the above formula, the activation function applies on the linear regression output [6]. The formula is given as following

$$h_\theta(x) = \frac{1}{1 + e^{-y}} = \frac{1}{1 + e^{-\theta^T x^i}} \quad (2)$$

where  $h_\theta(x)$  is the hypothesis, estimated probability, when  $y = 1$  on input  $X$ .

Among them

$$y = \frac{1}{1 + e^{-x}} \quad (3)$$

is designed as the sigmoid function, which is commonly used in the logistic regression.

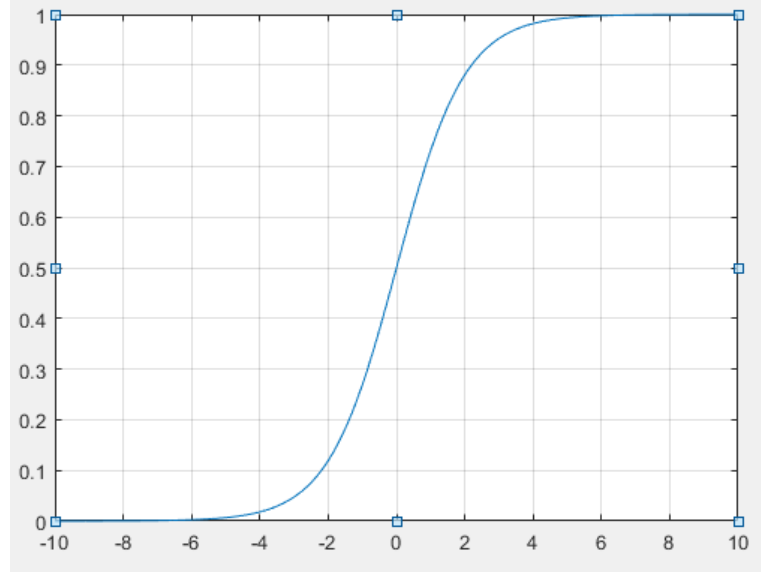


Figure 2.1. Sigmoid function.

The logistic regression contains the word ‘regression’, but it is not a regression model and it is a classification model.

### 2.1.2 Softmax Regression

The logistic regression is widely used in industrial engineering, because of its low complexity of the algorithm and easy for implementation of the features, etc. However, the logistic regression algorithm is mainly dealt with two-category (binary) problem which is not suitable for multi-classification. In this case, an algorithm capable of handling a multi-classification problem come to play. The softmax regression [7] algorithm is a generalization of the logistics regression algorithm, that is, the value of the class label  $y$  is greater than or equal to two. Assume that we have data  $\{(x^1, y^1), (x^2, y^2), \dots, (x^n, y^n)\}$  of  $n$  labeled samples.  $x^i$  is the set of  $\{x_1^i, x_2^i, \dots, x_m^i\}$ .  $y^i \in \{1, \dots, K\}$  where  $K$  is the number of classes. Suppose the function, which estimates the probability of each category for one input data  $x$ , is  $P(y^i = k|x^i; \theta)$  where  $k$  is the class and  $i$  is the one input data. Thus, the specific hypothesis function is expressed as

$$h_{\theta}(x^i) = \begin{bmatrix} P(y = 1|x^i; \theta) \\ P(y = 2|x^i; \theta) \\ \vdots \\ P(y = K|x^i; \theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^K e^{\theta_j^T x}} \begin{bmatrix} e^{\theta_1^T x^i} \\ e^{\theta_2^T x^i} \\ \vdots \\ e^{\theta_K^T x^i} \end{bmatrix} \quad (4)$$

Here  $\{\theta_1, \theta_2, \dots, \theta_K\}$  are the parameters for each category.  $\theta_j$  contain an  $m$ -dimensional vector which depends on the dimension of input  $x$ . As we can see  $P(y^i = k | x^i; \theta) = \frac{e^{\theta_k^T x^i}}{\sum_{j=1}^K e^{\theta_j^T x^i}}$ , summing each of them together leads to one. Due to this feature, we can obtain the maximum probability in these classes.

## 2.2 Neural Networks

### 2.2.1 Introduction

Neural networks [11] [12], as the name suggests, originated from the simulation of the human brain. The purpose is to understand the human brain function. Cognitive scientists and neuroscientists jointly constructed a neural network model and carried out simulation studies. This technology combined with engineering can help us build better neural network systems.

David Marr (1982) [8] believes that understanding an information processing system has three levels, collectively referred to as the level of analysis, namely:

- Computational theory: an abstract definition of the corresponding computational goals and tasks.
- Representation and algorithm: an explanation of how the input/output is represented and from the input to output.
- Hardware implementation: the actual physical implementation of the system.

It should be noted here that there can be multiple representations and algorithms for the same computational theory. For the same representation and algorithm there can be multiple hardware implementations. For example, for natural and artificial aircraft, the calculation theory can be “flying”, the algorithm is to use “aerodynamics”, and the implementation approach is “slap the wings” and the other is “launch the engine”.

The human brain can be seen as a hardware implementation of learning or pattern recognition. If we can reversely-analyze, extract the representations and algorithms used by the human brain from

this implementation, and further obtain computational theories, then we can get the computer hardware achievement that is more suitable for us.

A neural network proposes an intervening mode that introduce a small amount of local memory in the middle, using processor instructions to input different functions on the memory to implement different functions. Each of the processor corresponds to a neuron, the local parameters correspond to its prominent weights, and the entire structure is a neural network.

### 2.2.2 Components

The idea of neural network is from the biology neuron. Neuronal research has been around for a long time, and in 1904 biologists knew about the composition of neurons.

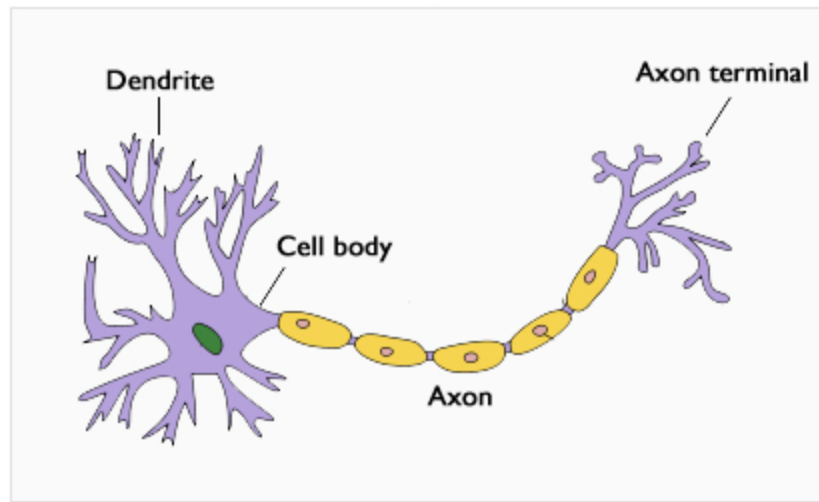


Figure 2.2. Neural structure.

A neuron usually has multiple dendrites, mainly used to receive incoming information, there is only one axon, and many axon terminals at the axon end can transmit information to other neurons. The axon terminal is connected to the dendrite of other neurons to transmit signals. In 1943, psychologist Warren McCulloch and mathematician Walter Pitts [9] referenced the structure of biological neurons and published an abstract neuron model of multilayer perception (MP). In the following, we will detail the neuron model.

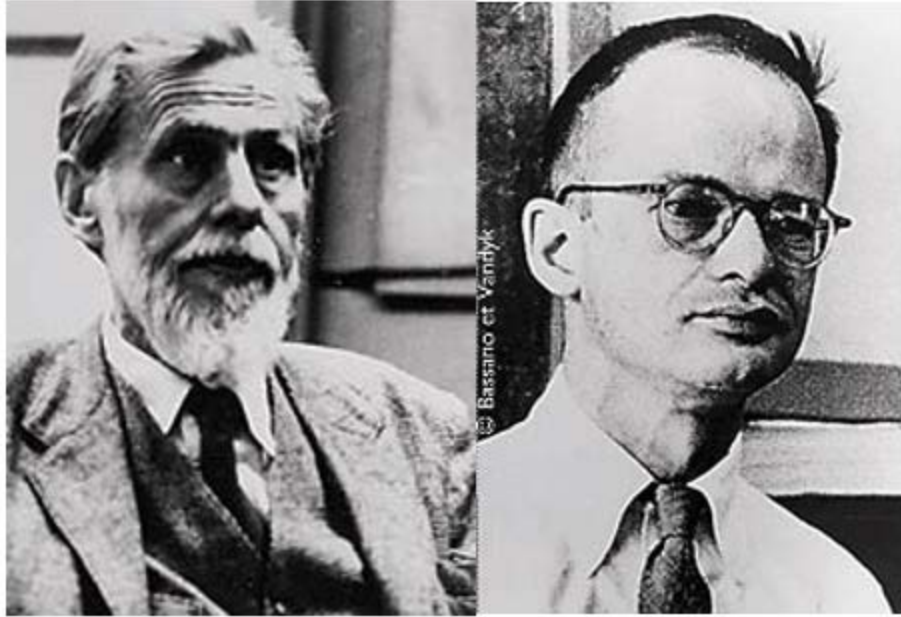


Figure 2.3. Warren McCulloch (left) and mathematician Walter Pitts (right) [9].

The neuron model [9] contains input, output and computational functions. The input can be analogized to the dendrite of the neurons, and output can be analogized to the axons of the neurons, and the calculation can be analogized to the nucleus.

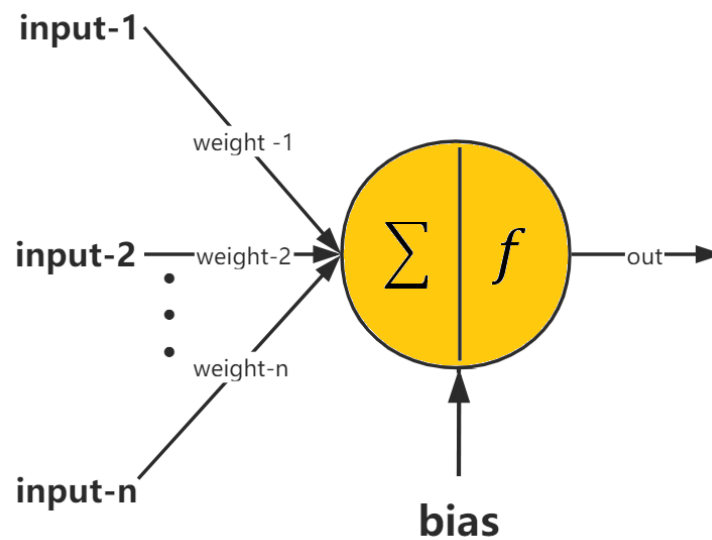


Figure 2.4. The simple neuron model.

There are many input data to the logistic unit and several outputs. Before the input data feed forward to the logistic unit, note the middle arrow line between input and logistic unit, there are not only many weights for each input data, but here is also one bias parameter for this logistic unit. In the logistic unit, the input data would sum together first and pass the activation function.

### 2.2.3 Mathematical Expression

Above model can be written as:

$$out = f(bias + in_1 * w_1 + in_2 * w_2 + \dots + in_n * w_n) = f\left(bias + \sum_{i=1}^n in_i * w_i\right) \quad (5)$$

where  $f()$  is the activation function.

### 2.2.4 Common Activation Functions

#### *Linear function*

This function is expressed as:

$$f(x) = x \quad (6)$$

A plot of the linear function is depicted as in Figure 2.5.

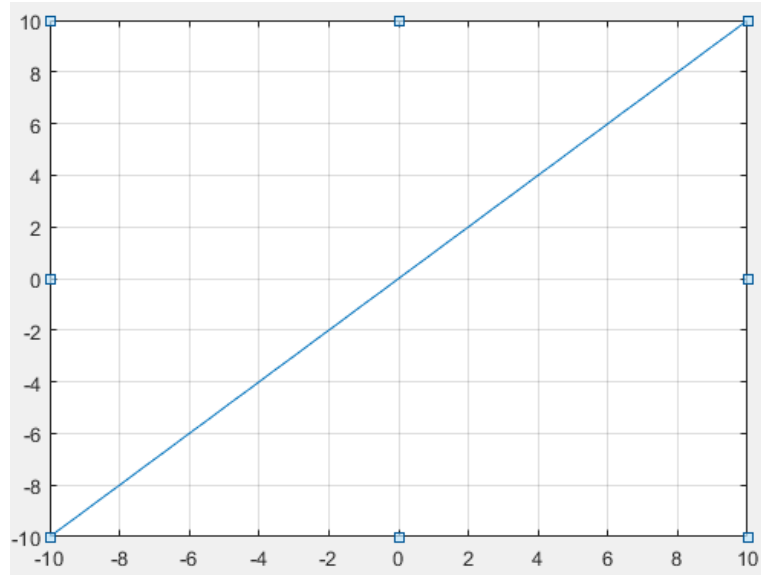


Figure 2.5. The linear function.

### *Sigmoid function*

The sigmoid function is given by:

$$f(x) = \frac{1}{1 + e^{-x}} \quad (7)$$

Figure 2.6 shows the plot of sigmoid function.

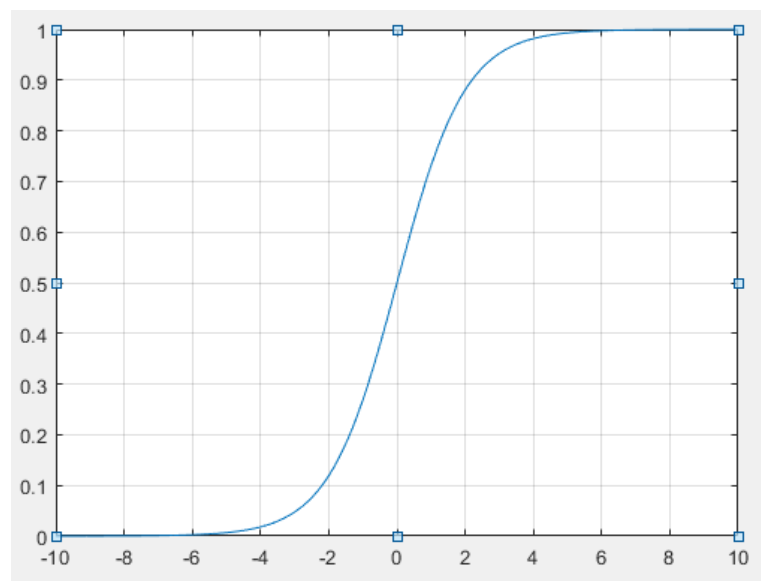


Figure 2.6. The sigmoid function.

This function could explain the neuron's firing rate, the probability, between 0 and 1.

### ***Tanh function***

This Tanh function is expressed as:

$$f(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (8)$$

The plot of Tanh function is shown in Figure 2.7

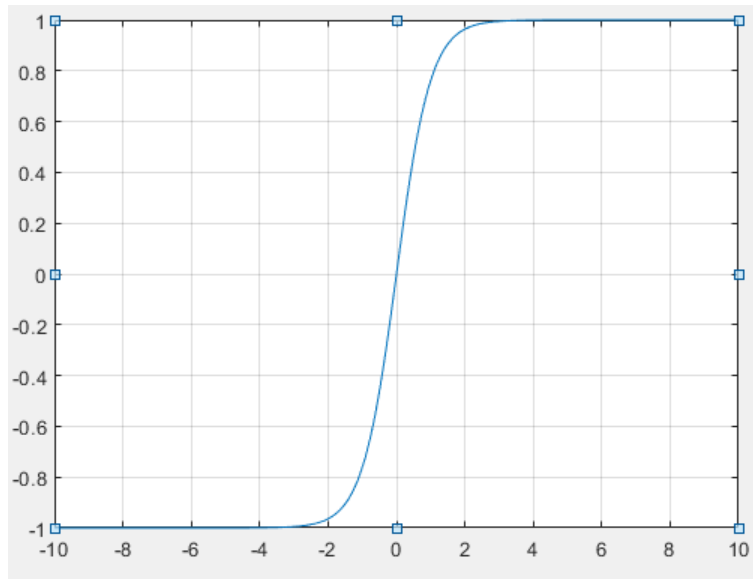


Figure 2.7. The Tanh function.

Note that the Tanh function is a rescaled sigmoid function which can be seen from the expression. Their shapes are similar, but with different scales. The Tanh function ranges from -1 to 1. Tanh is zero-centered function.

### ***Rectified linear unit (ReLU)***

The rectified linear unit (ReLU) function is given by:

$$f = \begin{cases} 0 & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases} \quad (9)$$

The plot of this function is described in Figure 2.8



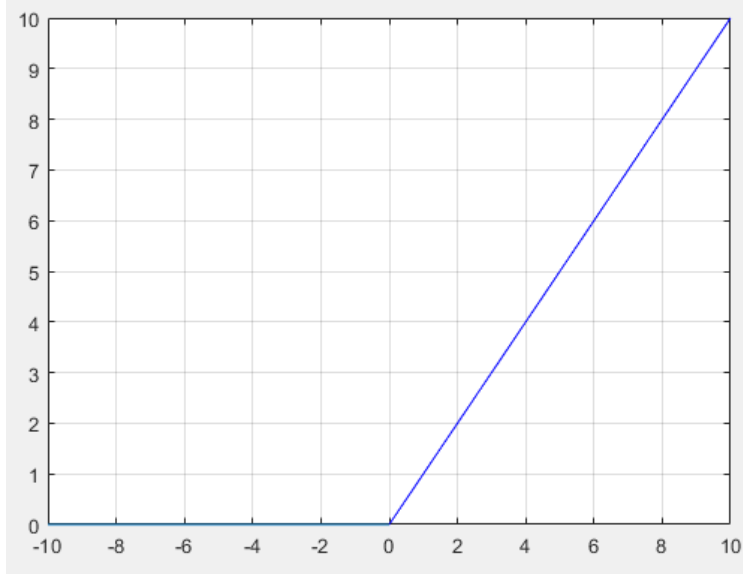


Figure 2.8. The ReLU function.

ReLU is widely used in convolutional neural networks. The complexity of the ReLU calculation is smaller than the sigmoid or tanh function, and the ReLU leads to a faster convergence rate, because of without the exponential calculation. But it is still not zero-centered.

### ***Leaky Rectified linear unit (LReLU)***

The leaky rectified linear unit (LReLU) function can be expressed below:

$$f = \begin{cases} ax & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases} = \max(ax, x) \quad (10)$$

Where parameter  $a$  is a small scalar. The plot of this function is depicted in Figure 2.9.

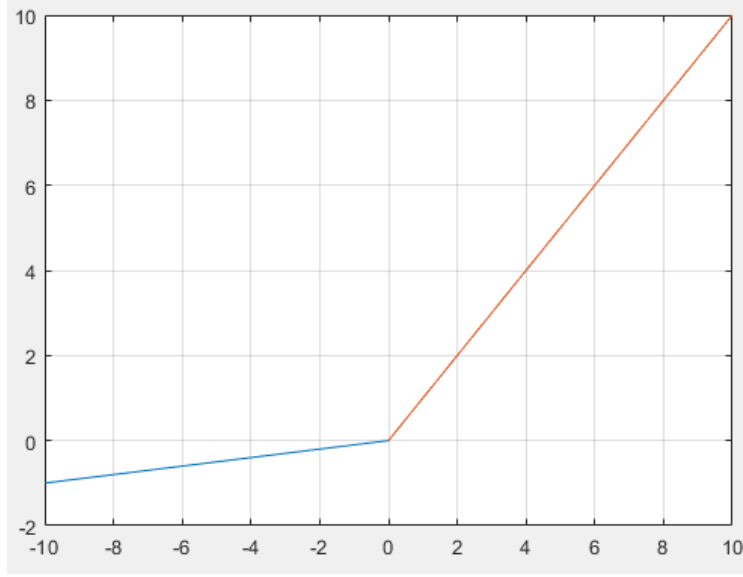


Figure 2.9. The LReLU function.

### 2.2.5 Update Neurons

When we have a neural network model, we need to train and evaluate this model. In this case, the feedforward propagation and the backward propagation are required to be applied.

#### *Feedforward Propagation*

Each layer of the Neural Network contains several neurons, and the neurons between the layers are connected by a weight matrix  $w^l$ .  $w_{ji}^{(l)}$  stands for in  $l^{\text{th}}$  layer, the weight for  $i^{\text{th}}$  node to the next layer the  $j^{\text{th}}$  node. Figure 2.10 shows a simple neural network. In the hidden layer, we note the sum function output for the all input data times weights of the  $i^{\text{th}}$  neuron in layer  $j$  as  $z_j^{(j)}$ . After this, the activation function output denotes as  $a_i^{(j)}$ . The number of layers starts at input layer.

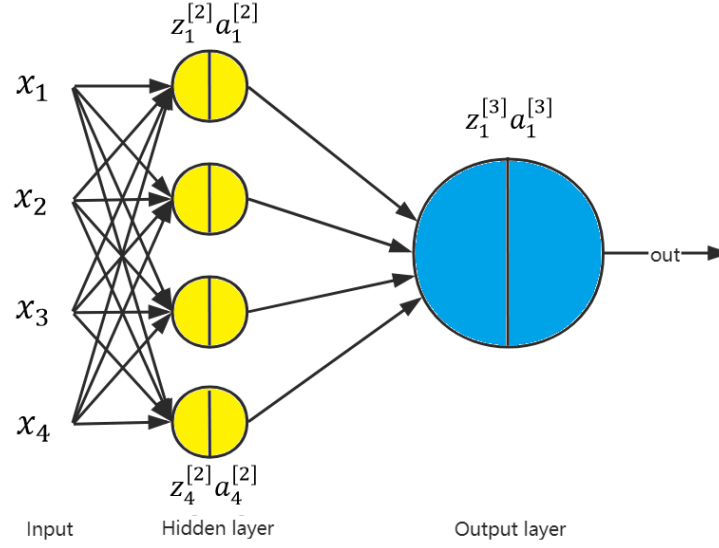


Figure 2.10. The neural network structure.

An information transfer process can be described as follows:

1. The  $j$ th layer of neurons receives the upper layer of incoming stimuli (neural impulses):

$$z^{(j)} = w^{(j-1)} a^{(j-1)} \quad (11)$$

2. After the stimulus is activated by the activation function  $g$ , an activation vector  $a^{(j)}$  is generated.  $a_i^{(j)}$  represents the activation obtained by the  $i$ th neuron in the  $j$ th layer:

$$a^{(j)} = g(z^{(j)}) \quad (12)$$

This process, because the order of occurrence constantly pass the stimulus from the previous layer to the next layer, it is called forward propagation.

The above neural network forward propagation can be written below:

$$\begin{aligned}
z_1^{(2)} &= w_{11}^{(1)} x_1 + w_{12}^{(1)} x_2 + w_{13}^{(1)} x_3 + w_{14}^{(1)} x_4 \\
a_1^{(2)} &= g(z_1^{(2)}) \\
z_2^{(2)} &= w_{21}^{(1)} x_1 + w_{22}^{(1)} x_2 + w_{23}^{(1)} x_3 + w_{24}^{(1)} x_4 \\
a_2^{(2)} &= g(z_2^{(2)}) \\
z_3^{(2)} &= w_{31}^{(1)} x_1 + w_{32}^{(1)} x_2 + w_{33}^{(1)} x_3 + w_{34}^{(1)} x_4 \\
a_3^{(2)} &= g(z_3^{(2)}) \\
z_4^{(2)} &= w_{41}^{(1)} x_1 + w_{42}^{(1)} x_2 + w_{43}^{(1)} x_3 + w_{44}^{(1)} x_4 \\
a_4^{(2)} &= g(z_4^{(2)}) \\
z_1^{(3)} &= w_{11}^{(2)} a_1^{(2)} + w_{12}^{(2)} a_2^{(2)} + w_{13}^{(2)} a_3^{(2)} + w_{14}^{(2)} a_4^{(2)} \\
h_w(x) &= a_1^{(3)} = g(z_1^{(3)})
\end{aligned} \tag{13}$$

### ***Loss/Cost Function***

In order to fit the neural network model in training, we need to have a reference to the model, which is the loss/cost function. Constantly reducing the value of the loss/cost function is the process of constantly optimizing the Neural Network model.

In linear regression, the mean variance error function is the main loss/cost function. The mathematic expression is:

$$\frac{loss}{cost} = \frac{1}{2n} \sum_{i=1}^n (h_w(x_i) - y_i)^2 \tag{14}$$

where  $x_i, y_i$  are respectively input data and label of the data.

In logistic regression, the mean variance error function is not working, because it is no longer convex function any more. It can have many local minima, which affect the gradient descent algorithm to find a global minimum. To address this, we need to reselect the loss/cost function of the logistic regression as

$$\frac{loss}{cost} = \begin{cases} -\log(h_w(x)) & \text{if } y = 1 \\ -\log(1 - h_w(x)) & \text{if } y = 0 \end{cases} \tag{15}$$

As we can see, when  $y$  is equal to 1,  $h_w(x)$  is closing to 1, the loss/cost is closing to 0. On the contrary, when  $y$  is equal to 0,  $h_w(x)$  is closing to 1, the loss/cost is closing to positive infinity. The above expression can be abbreviated and expand as  $n$  input data as follow:

$$\frac{loss}{cost} = -\frac{1}{n} \sum_{i=0}^n y_i \log(h_w(x_i)) + (1 - y_i) \log(1 - h_w(x_i)) \quad (16)$$

In the neural network, the loss/cost function can combine all layers' weights and all output units/classes. Let's define a few variables as follows:

- $n$  = the number of input data.
- $L$  = total number of layers in the Neural Network.
- $s_i$  = number of units in layer  $i$ .
- $K$  = number of output units/classes.
- $h_w(x)_k$  as being a hypothesis that results in the  $k^{th}$  output.

For the neural networks loss/cost function, we have:

$$loss = -\frac{1}{n} \sum_{i=0}^n \sum_{k=0}^K [y_i^k \log(h_w(x_i)_k) + (1 - y_i^k) \log(1 - h_w(x_i)_k)] + \frac{\lambda}{2n} \sum_{l=1}^{L-1} \sum_{i=1}^{s_i} \sum_{j=1}^{s_{j+1}} w_{j,i}^{(l)2} \quad (17)$$

### ***Gradient Descent***

Gradient descent is known as the steepest descent method. First we take partial derivative of weight to loss function, and we then calculate the difference between the weight and the opposite partial derivative to get the updated weight.

$$w^1 = w^0 - \alpha \frac{\partial loss}{\partial w} \quad (18)$$

Repeating Equation (18) can get the optimized weights, so that the loss/cost could be the minimum.

There are three method of gradient, batch gradient descent (BGD), stochastic gradient descent (SGD), mini-batch gradient descent (MBGD). BGD uses all data; SGD uses only one data; MBGD, uses a small set of data. In neural network, we use mini-batch gradient descent.

### ***Backpropagation***

The back propagation (BP) algorithm (i.e., backpropagation algorithm) [10] [12] is a learning algorithm suitable for multi-layer neural networks. It is based on the gradient descent method. The input-output relationship of BP network is essentially a mapping relationship: the function performed by a BP neural network with  $n$  input  $m$  output is a continuous mapping from  $n$ -dimensional Euclidean space. The mapping is highly nonlinear. Its information processing capability is derived from multiple recombination of simple nonlinear functions, so it has a strong ability to reproduce functions. This is the basis for the application of the BP algorithm.

The learning process of the BP algorithm assembles a forward propagation process and a backpropagation process. In the forward propagation process, the input information passes through all hidden layers and is transmitted to the output layer. If the output predicted value is not match the ground truth value, the sum of the squares of the error in the output predicted value and the ground truth value is taken as the cost function, and then the back propagation is performed, and the partial derivative of the cost function for each neuron weight is obtained layer by layer to form a target. The gradient of the function to the weight vector is used as the offset to modify and update the weight. When the error reaches the minimum value, the network learning ends.

Backpropagation is based on the gradient descent method; it is important to calculate gradient. However, there are more than two derivatives of the composition functions need to be computed in neural networks. The chain rule [16] is the best way to address this problem. Here is a brief introduction.

Assume two function,  $y = g(x), z = h(y)$ . Then to calculate the derivative of  $z$  to  $x$ , the calculation process is given as

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x} \quad (19)$$

which will be applied to the neural networks

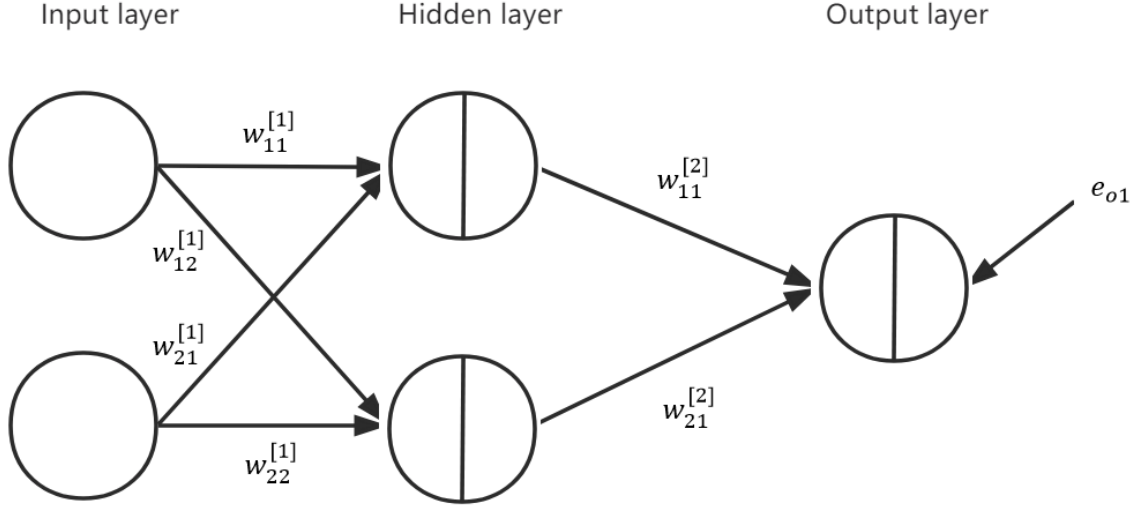


Figure 2.11. The neural network parameters.

For a network depicted in Figure 2.11, the parameters have the same definition as previous ones. After forward propagation, each unit will have their own output.

$$\begin{aligned}
 e_{o1} &= \frac{1}{2} (a_1^{(3)} - y_1)^2 \\
 a_1^{(3)} &= \text{sigmoid}(z_1^{(3)}) \\
 z_1^{(3)} &= (w_{11}^{(2)} \cdot a_1^{(2)} + w_{12}^{(2)} \cdot a_2^{(2)} + b_1^{(3)})
 \end{aligned} \tag{20}$$

$b_1^{(3)}$  stands for bias for layer 2 to layer 3.

Depending on the chain rule, we can calculate the error partial derivative over  $w_{11}^{(2)}$ .

$$\frac{\partial e_{o1}}{\partial w_{11}^{(2)}} = \frac{\partial e_{o1}}{\partial a_1^{(3)}} \cdot \frac{\partial a_1^{(3)}}{\partial z_1^{(3)}} \cdot \frac{\partial z_1^{(3)}}{\partial w_{11}^{(2)}} \tag{21}$$

According to Equation (21), we can get the final equation as follow.

$$\frac{\partial e_{o1}}{\partial w_{11}^{(2)}} = (a_1^{(3)} - y_1) \cdot a_1^{(3)} \cdot (1 - a_1^{(3)}) \cdot a_1^{(2)} \tag{22}$$

The partial derivative to  $w_{12}^{(2)}$  has the same process as above.

$$\frac{\partial e_{o1}}{\partial w_{12}^{(2)}} = (a_1^{(3)} - y_1) \cdot a_1^{(3)} \cdot (1 - a_1^{(3)}) \cdot a_2^{(2)} \quad (23)$$

Also, the same process is used for the  $w_{11}^{(1)}$ ,  $w_{12}^{(1)}$ ,  $w_{21}^{(1)}$  and  $w_{22}^{(1)}$ . We could update our weights based on the gradient descent, that is.

$$w_{ji}^{(l)} = w_{ji}^{(l)} - \alpha \frac{\partial loss}{\partial w_{ji}^{(l)}} \quad (24)$$

The whole process is the main idea for backpropagation algorithm. And this is main method to update in neural networks.

### 2.3 Convolutional Neural Networks (CNN) Background

Traditional neural networks will generate numerous weights. Since there are too many parameters, the amount of backpropagation computation load is huge. Thus, the traditional neural network is not recommended from the perspective of computing resources and tuning. If we apply images in traditional neural networks, we will generate a large number of weights.



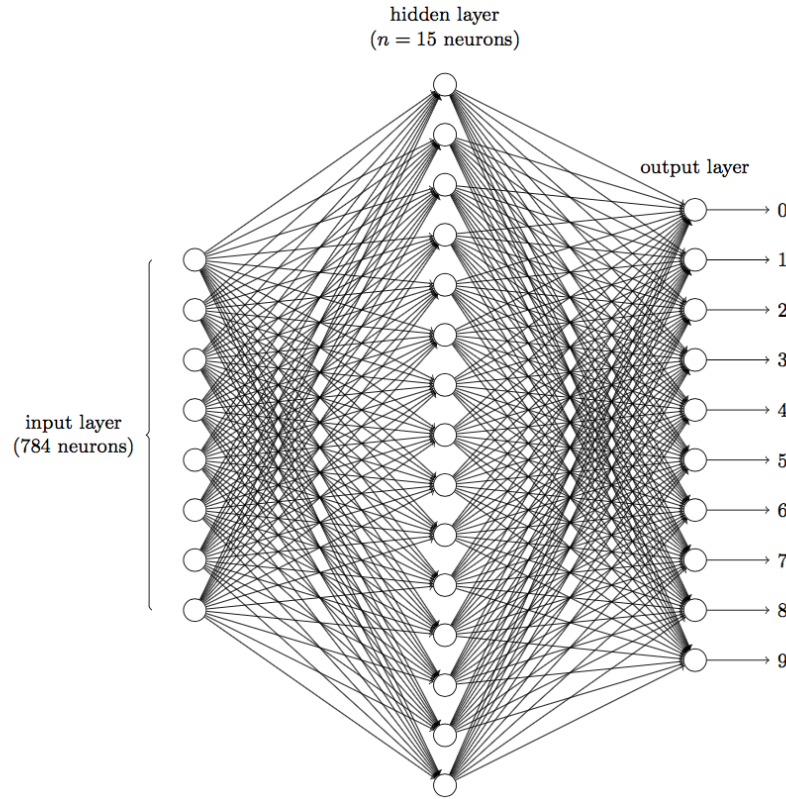


Figure 2.12. The neural network structure with 15 neurons in hidden layer [13].

If we have a traditional neural network is shown in Figure 2.12, we assume the input image is a binary image with a size of 28 by 28. The input layer is  $28 * 28 = 784$  neurons. There are 15 neurons, 10 neurons respectively in hidden layer and output layer. So from simple calculation, we need the number of parameters (weights and bias):  $784 * 15 * 10 + 15 + 10 = 117625$ . This neural network only has one hidden layer. We increase several hidden layers, the number of parameters will exponentially increase. There are three parts need to be solved. The first one, local connections, this is the easiest to think of. Each neuron is no longer connected to all neurons in the upper layer, but only to a small number of neurons. This reduces many parameters in neural networks. The second is weight sharing. A set of connections can share the same weight, rather than having a different weight for each connection, which reduces many parameters as well. The third is down sampling input information. To address this problem, the convolutional neural networks (CNN) is developed. Similar to the neural network model, CNN's design inspiration comes from the study of nerve cells.

The Nobel Prize in Medicine in 1981 was awarded to David Hubel [17], Torsten Wiesel, and Roger Sperry. Their main contribution is to discover that the information processing of the human visual system is hierarchical.

Convolutional neural networks could work with many sophisticated problems, such as in image recognition, speaking recognition, and so on. This thesis work only focusses on image recognition problems. The following part mainly devotes to convolutional neural network.

## **2.4 Convolutional Neural Networks (CNN) Definition**

Convolutional neural networks [13] are a class of feedforward neural network with convolutional computation and deep structure. It is one of the representative algorithms for deep learning. The convolutional neural network has representation learning ability and can perform shift-invariant classification of input information according to its hierarchical structure. Therefore, it is also called “Shifted-Invariant Artificial Neural Network, SIANN.”

The study of convolutional neural networks began in the 1980s and 1990s. The time delay network and LeNet-5 [20] were the earliest convolutional neural networks. After the 21<sup>st</sup> Century, with the introduction of deep learning theory and the improvement of numerical computing equipment, convolutional neural networks have been rapidly developed and applied to computer vision, natural language processing and other fields.

The convolutional neural network constructs the visual perception mechanism of the creature, which can perform supervised learning and unsupervised learning. The convolutional kernel parameter sharing and the sparseness of the inter-layer connection in hidden layer enable the convolutional neural network to minimize computational computations for grid-like topology features such as pixel and audio. The convolutional neural networks have a stable effect and have no additional feature engineering requirements for data.

## 2.5 Convolutional Neural Networks Architecture

A convolutional neural network is contributed by input layer, multiple hidden layers and output layer. A series of convolutional layers constitute in hidden layer. Mathematically, convolutional layer computing is technically a sliding dot product or cross-correlation. After convolutional computing, an activation function is applied in the same fashion as the traditional neural network. The ReLU function is a common activation function used in the convolution layer in CNN. In order to down sample each convolution layer input, there is a special convolutional layer which is called pooling layer. After series of the convolutional layer and pooling layer pairs, the final fully connected layer ends the CNN structure as described in Figure 2.13.

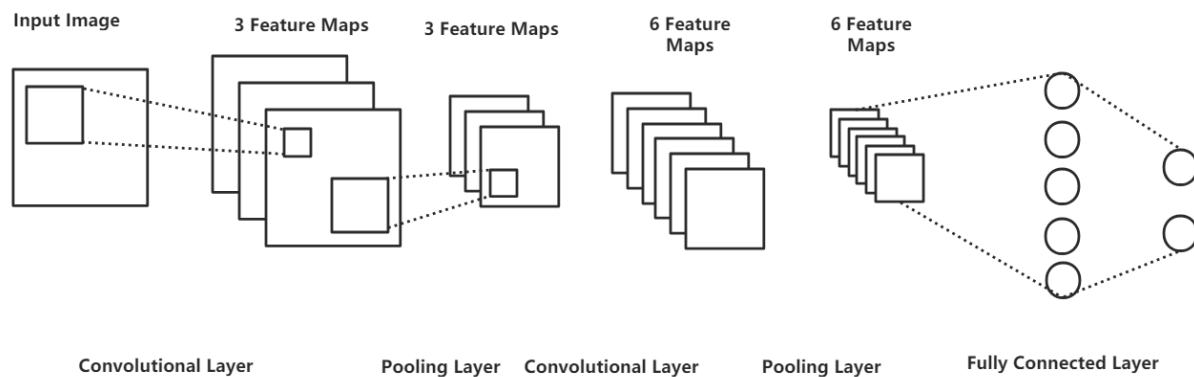


Figure 2.13. The common convolutional neural network architecture.

### 2.5.1 Convolutional Layer

The most important part in CNN is the convolutional layer [13] [18]. The name of CNN is derived from the use of convolution operations. The purpose of convolution is mainly to extract the features of the picture. Convolution operations preserve the spatial relationship between pixels.

In the convolutional layer, by forward propagation, the input image dot product with filter (kernel), and the value of each filter (kernel) is updated by backpropagation.

We first use a simple example to explain how to calculate the convolution and then abstract some important concepts and calculation methods for the convolutional layer.

Suppose there is a 5x5 image, convolving with a 3x3 filter, and we want to get a 3x3 feature map, as shown below:

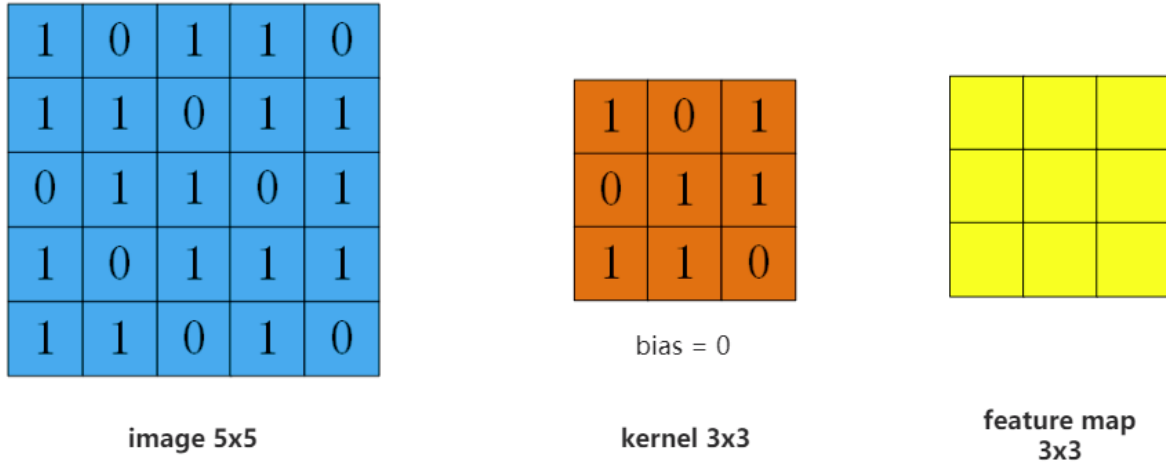


Figure 2.14. The input image pixels (left), the filter/kernel (middle), the output feature map (right).

In order to clearly describe the convolution calculation process, we first encode each pixel of the image, and use  $x_{i,j}$  to represent the  $i^{\text{th}}$  row and the  $j^{\text{th}}$  column element of the image. Each weight of the filter encoded by  $w_{i,j}$  is used to represent the weight of the  $i^{\text{th}}$  row and the  $j^{\text{th}}$  column, and  $w_b$  is used to represent the bias term of the filter. Use  $a_{i,j}$  to represent the  $i^{\text{th}}$  row and the  $j^{\text{th}}$  column of the feature map and  $f$  to indicate the activation function. Then, convolution can be calculated below:

$$a_{i,j} = f\left(\sum_{m=0}^2 \sum_{n=0}^2 w_{m,n} x_{i+m,j+n} + w_b\right) \quad (25)$$

For example, for the  $a_{0,0}$  element in the upper left corner of the feature map, the convolution calculation method is:

$$\begin{aligned} a_{i,j} &= f\left(\sum_{m=0}^2 \sum_{n=0}^2 w_{m,n} x_{i+m,j+n} + w_b\right) \\ &= \text{ReLU}(1 + 0 + 1 + 0 + 1 + 0 + 0 + 1 + 0) \\ &= \text{ReLU}(4) = 4 \end{aligned} \quad (26)$$

The calculation results are shown in the figure below:

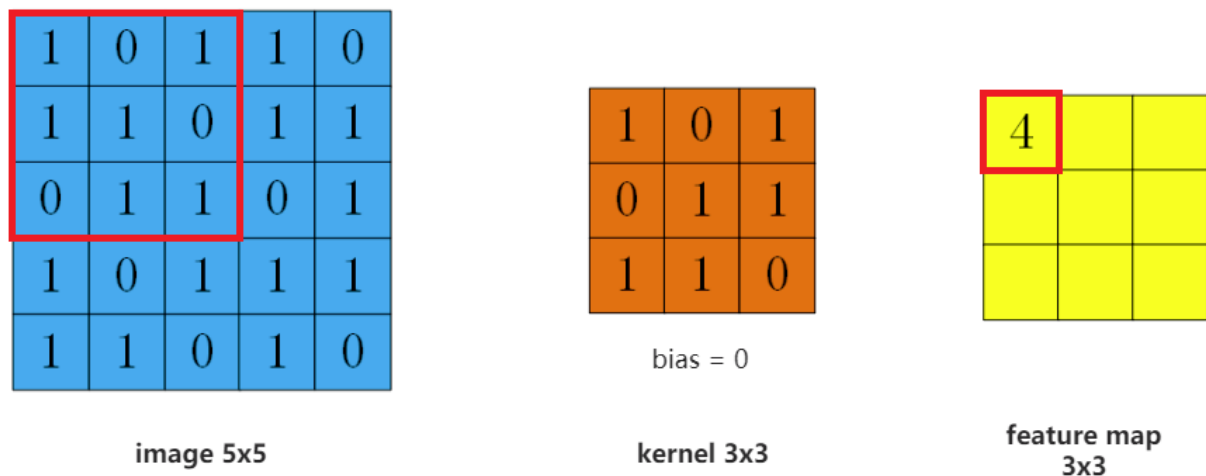


Figure 2.15. The first convolutional calculation step.

The rest of feature map results could be calculated in the same way.

In the above calculation process, the stride is 1. The stride can be set to number greater than one. For example, when the stride is 2, the feature map is calculated as follows:

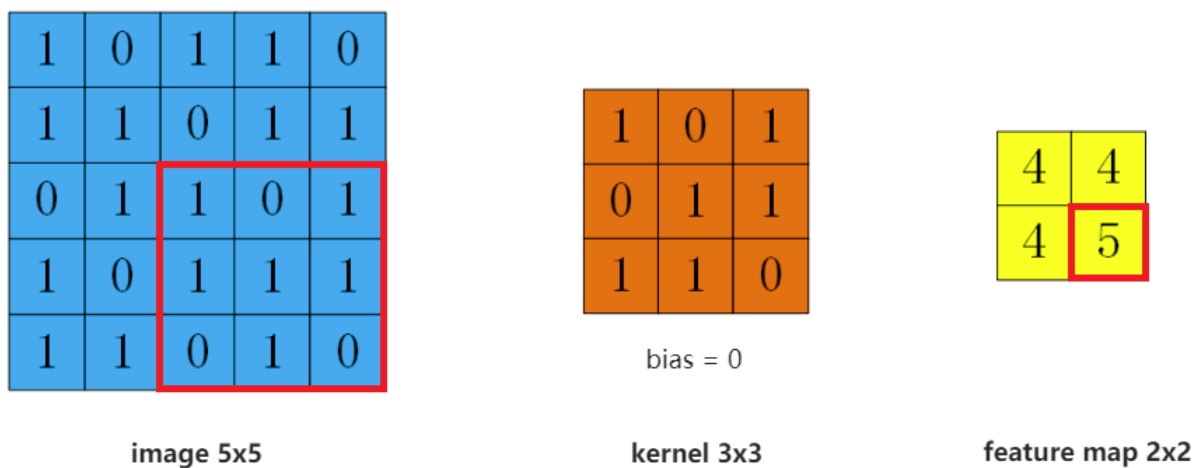


Figure 2.16. The last convolutional calculation step.

We noticed that when the stride is set to 2, the feature map size becomes 2x2. This means that the image size, stride, and the size of the feature map after convolution are related. In fact, they satisfy the following relationship:

$$W_{feature\ map} = \frac{W_{input\ image} - F + 2P}{S} + 1 \quad (27)$$

$$H_{feature\ map} = \frac{H_{input\ image} - F + 2P}{S} + 1 \quad (28)$$

where  $F$  is the width of the filter and  $P$  is the number of zero-padding. Note that zero-padding pads a few laps 0s around the original image. If the value is 1, then we make one lap zero padding.  $S$  is the number of the strides.

So far, the calculation method of convolutional layer with depth 1 is described. . If the image depth before convolution is  $D$  which is larger than 1, then the depth of the corresponding filter must also be  $D$ . We can extent Equation (25) as following:

$$a_{i,j} = f \left( \sum_{d=0}^{D-1} \sum_{m=0}^{F-1} \sum_{n=0}^{F-1} w_{d,m,n} x_{d,i+m,j+n} + w_b \right) \quad (29)$$

where  $D$  is the depth;  $F$  is the size of the filter (width or height, the two are the same);  $w_{d,m,n}$  represents the weight of the  $m^{th}$  row and the  $n^{th}$  column of the  $d^{th}$  layer of the filter;  $a_{i,j}$  represents the weight of the  $i^{th}$  row and the  $j^{th}$  column of the feature map.

As we mentioned before, there can be multiple filters per convolutional layer. After each filter and the original image are convolved, we can get a feature map. Therefore, the depth (number) of the feature map and the number of filters in the convolutional layer after convolution are the same. There are two characteristics, local connectivity and parameter sharing, in convolutional layer.

### ***Local connectivity***

Comparing with the traditional neural network, each neuron in the adjacent layer connects with others. Because of the connection method, the number of weights in the traditional neural network is huge so a dramatically computing power. However, the convolutional neural network addresses weights challenge and decrease the number of weights significantly. To do so, the filter connecting with the upper layer has the local connection, which will isolate the number of weights with input data size.

### ***Parameter sharing***

One filter will spread out to all input images, which means that the whole image uses the same filter size and weights. In comparison with full connection, the number of weights is tiny. Whole image shares the same weights to get the output feature maps.

### **2.5.2 Activation Layer**

The activation layer mainly performs a nonlinear mapping on the output of the convolutional layer because the calculation of the convolutional layer is still a linear calculation. The activation function increases the ability of generalization. The ReLU function is generally used. Notice that the ReLU function only returns  $\max(0, x)$ , or simply removes the negative weights. The associated layer is also called ReLU layer.

The convolutional layer and the activation function are usually combined together as a convolutional block.

### **2.5.3 Pooling Layer**

When the input passes through the convolutional layer, if the receptive field is relatively small, the length of the stride is relatively small, and the obtained map is still relatively large. The dimensioning operation can be performed on each feature map through the pooling layer, and the depth of output is still the same, and it is still the number of feature maps.

The pooling layer also has a “pooling” to scan the feature map matrix and calculate the matrix values in the pooling receptive field. There are generally two ways to calculate:

- Max pooling: take the maximum value in the pooling receptive field matrix.
- Average pooling: take the average value in the pooling receptive field matrix.

The scanning step stride, which is also involved in the scanning process, scans in the same way as convolutional layer, scanning from left to right, then moving down the step stride and scanning from left to right again until the end of image. The process is shown in Figure 2.17.:

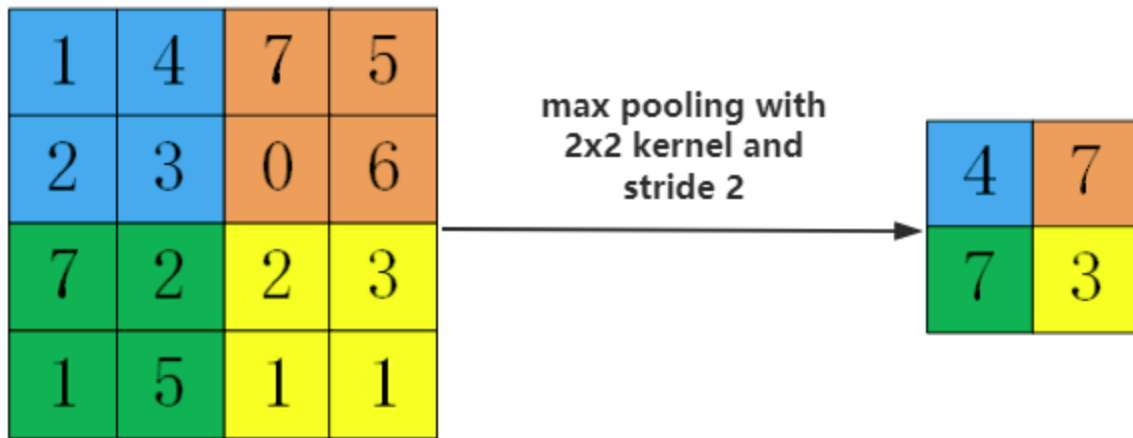


Figure 2.17. The max pooling.

The Figure 2.17 shows the max pooling operation with the filter size as 2x2 and stride as 2, respectively. As can be seen, the input size of 4x4 is divided by 4 parts, and each part is presented by its largest number. So the output is 2x2 feature map.

#### 2.5.4 Fully Connected Layer

The fully connected layer mainly re-fitting features to reduce the loss of feature information. The output layer is mainly prepared for the output of the final target result.

#### 2.5.5 Batch Normalization Layer

We know that once the network is trained, the parameters will be updated. In addition to the input layer data, because the input layer data have artificially normalized. The data distribution of each layer of the rest network is always changing, because during training, the update of the previous layer training parameters will lead to changes in the distribution of the data of the later layers. Take the second layer of the network as an example, the second layer input of the network is calculated by the parameters and input of the first layer, and the parameters of the first layer are always changing throughout the training process, so it will inevitably cause a change in the distribution of the input data for each subsequent layer. We refer to the change in data distribution during the



training process of the network at middle layer: “Internal Covariate Shift” [21]. To address this problem, the birth of Batch Normalization [22] [23] evolved.

Just like the activation layer, the convolutional layer, the fully connected layer, and the pooling layer, Batch Normalization (BN) layer is also a layer of the network. In the previous section, we mentioned that in addition to the input and the output layer, the other layers update the parameters during the training, which causes changes in the distribution of input data in the later layer. BN layer normalized the output of the upper layer network to a mean of 0 and a variance of 1, and then acts as an input to the underlying network, so that BN layer solves the aforementioned “Internal Covariate Shift” problem.

## **2.6 The Neural Network Models History**

### **2.6.1 Image Classification**

#### ***AlexNet***

AlexNet [23] is not the first convolutional neural network architecture, but it is the most influential one. In 2012, Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton from the University of Toronto created a “big and deep convolutional neural network” that won ILSVRC (ImageNet Large Scale Visual Recognition Challenge).

In the paper, Hinton et al. proposed a new network architecture called AlexNet. Compared with the existing CNN at the time, AlexNet has a streamlined layout consisting of five convolutional layers with overlapping pooling layers and dropout layers [24] [25], and three fully connected layers, which can classify 1000 targets.

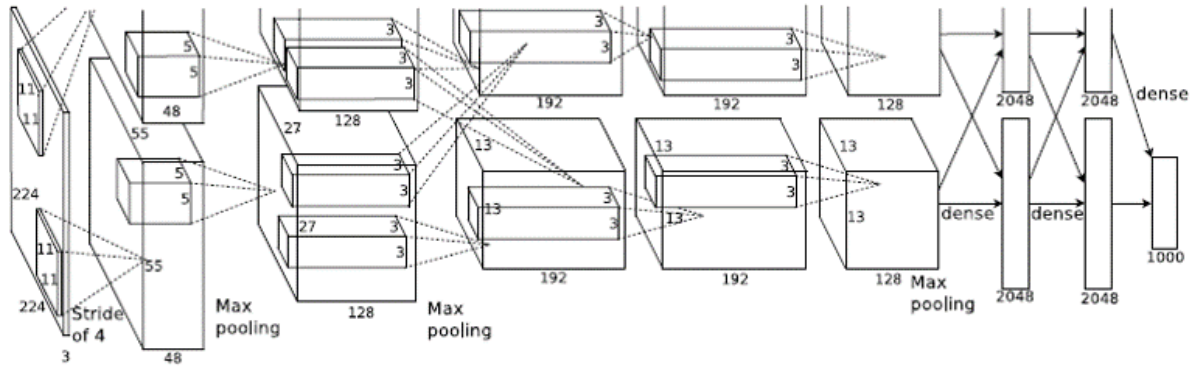


Figure 2.18. The Alex Network Architecture [23].

It is the first time the ReLU activation function is implemented in AlexNet. The two new layers were introduced, overlapping pooling layer and dropout layer. Comparing with the common CNN which uses average pooling, the AlexNet uses maximum pooling to avoid the average pooling fuzzification effect. AlexNet proposes that the stride size is smaller than the size of the pooling kernel, so that there will be overlap and coverage between the output of the pooling layer, which improves the richness of features and reduces the loss of information. Dropout layer is new idea from Hinton to be able to prevent over-fitting of neural networks. A regularization method is used to prevent over-fitting of the model as compared to the general linear model, and in the neural network, dropout is implemented by modifying the structure of the neural network itself.

## VGG

After the AlexNet, there so many CNN models arising. The new powerful and influential model is VGG [26]. VGG was proposed by Oxford's 'V'isual 'G'eometry 'G'roup. The neural network model is related work at ILSVRC 2014. The main work is to prove that increasing the depth of the network can actually affect the final performance of the network. VGG has two structures, VGG16 and VGG19. There is no essential difference between the two, but the network depth is different. One improvement of VGG16 over AlexNet is the replacement of larger convolution kernels (11x11, 7x7, 5x5) in AlexNet with several consecutive 3x3 convolution kernels. For a given receptive field (the local size of the input image associated with the output), the use of stacked small convolution kernels is superior to the use of large convolution kernels, because multiple

layers of nonlinear layers can increase network depth to ensure more complex models learned, and the cost is relatively small (less parameters).

In simple terms, in VGG, three 3x3 convolution kernels are used instead of 7x7 convolution kernels, and two 3x3 convolution kernels are used instead of 5x5 convolution kernels. The main purpose of this is to ensure the same under the condition of the perceptual field, the depth of the network is improved, and the effect of the neural network is improved to some extent.

VGG has 16 or 19 hidden layers which contain 13 convolutional layers or 16 convolutional layers and 3 fully connection layers. The architecture is shown as follow:

Table 2-1. The VGG network architecture [26]. Conv3-64 stands for convolutional layer with 3x3 kernel and output depth is 64. FC-4096 stands for fully connected layer with 4096 output depth.

ConvNet Configuration					
A	A-LARN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19weight layers
Input(224 x 224 RGB image)					
Conv3-64	Conv3-64 LRN	Conv3-64 Conv3-64	Conv3-64 Conv3-64	Conv3-64 Conv3-64	Conv3-64 Conv3-64
Maxpool					
Conv3-128	Conv3-128	Conv3-128 Conv3-128	Conv3-128 Conv3-128	Conv3-128 Conv3-128	Conv3-128 Conv3-128
Maxpool					
Conv3-256 Conv3-256	Conv3-256 Conv3-256	Conv3-256 Conv3-256	Conv3-256 Conv3-256 Conv1-256	Conv3-256 Conv3-256 Conv3-256	Conv3-256 Conv3-256 Conv3-256 Conv3-256
Maxpool					
Conv3-512 Conv3-512	Conv3-512 Conv3-512	Conv3-512 Conv3-512	Conv3-512 Conv3-512 Conv1-512	Conv3-512 Conv3-512 Conv3-512	Conv3-512 Conv3-512 Conv3-512 Conv3-512
Maxpool					
FC-4096					
FC-4096					
FC-1000					
Soft-max					

## GoogleLeNet

GoogleLeNet [27] is the winner of the 2014 ILSVRC14 [55] competition in ImageNet, the same as the VGG network, the VGG network is the runner-up of the competition that year. But in fact, the TOP-5 error rate of the two networks is not much different. The network structure of GoogleLeNet is more complicated than VGG. It is a 22-layer network, and proposes a structure of inception, which is a great progress and inspiration for ResNet.

It is well known that the most straightforward way to improve deep neural networks is to increase the size of the network, including increasing depth and width. Increasing the depth is to increase the number of layers of the neural network. Increasing the width is to increase the number of units per layer. However, increasing the depth and width brings two problems. 1. Because the parameters become more and more, the network is easier to overfitting, which reduces the accuracy. 2. The amount of calculation will be greatly increased, and some parameters may be trained to eventually be 0, which wastes computing resources when computing resources are limited. So the ultimate goal is to reduce parameters, reduce calculations, increase depth, and increase width.

The solution is to replace the full connection with the sparse connection structure, but the existing calculation method is very inefficient for non-uniform sparse calculations, so the GoogleLeNet proposes the improvement way, inception model.

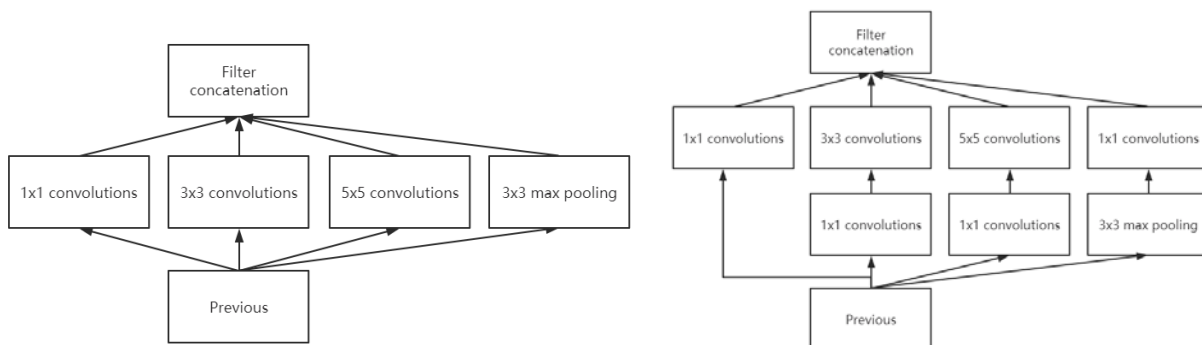


Figure 2.19. The Inception Structure. Inception naïve module (left), Inception module with dimension reductions (right).

Figure 2.19 describes two proposed inception models. Model (b) is an improved version of model (a). As shown in the figure, model (a) is a dense component to approximate and replace an optimal local sparse structure. This may be due to the fact that this connected model has a summary of the information brought by the four-connection method, and the reduced parameters of the connection with the direct 3x3 convolution layers. Why is the parameter reduced? Give an example of the figure below:

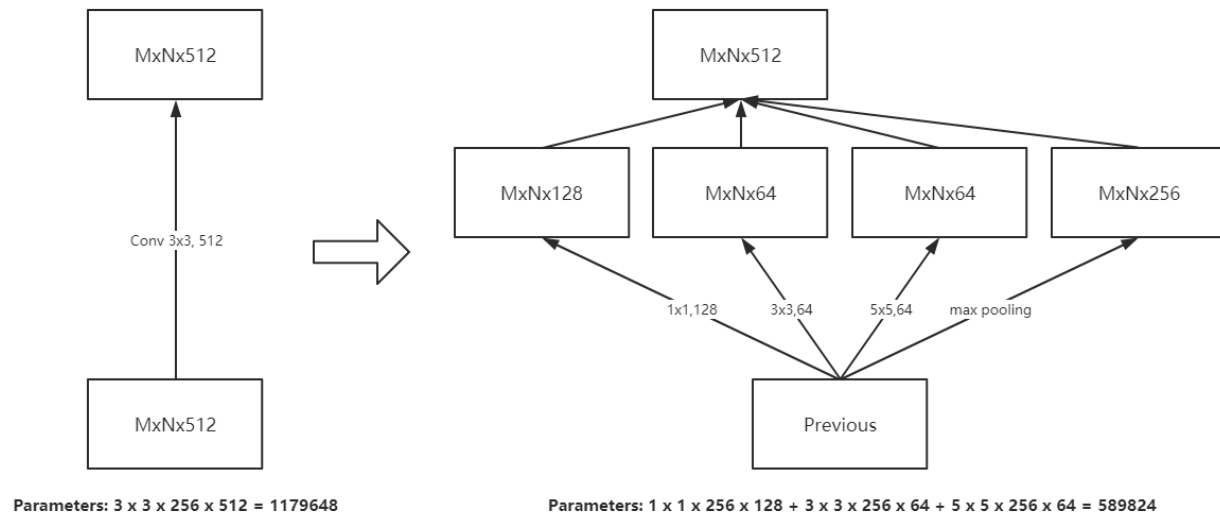


Figure 2.20. The Inception naïve version reduces parameters.

We noticed the parameter quantity is reduced, but the reduction is not too much, and if the input data channel value is relatively large, the parameter quantity will still be relatively large, so the improved model (b) is proposed. Model (b) relative to the model (a) uses a 1x1 convolutional layer for dimensionality reduction before using a 3x3 or a 5x5 convolution connection, which reduces the parameter amount again, and also increases the network non-linear activation after 1x1 convolutional layer. The amount of parameters changes as shown below.

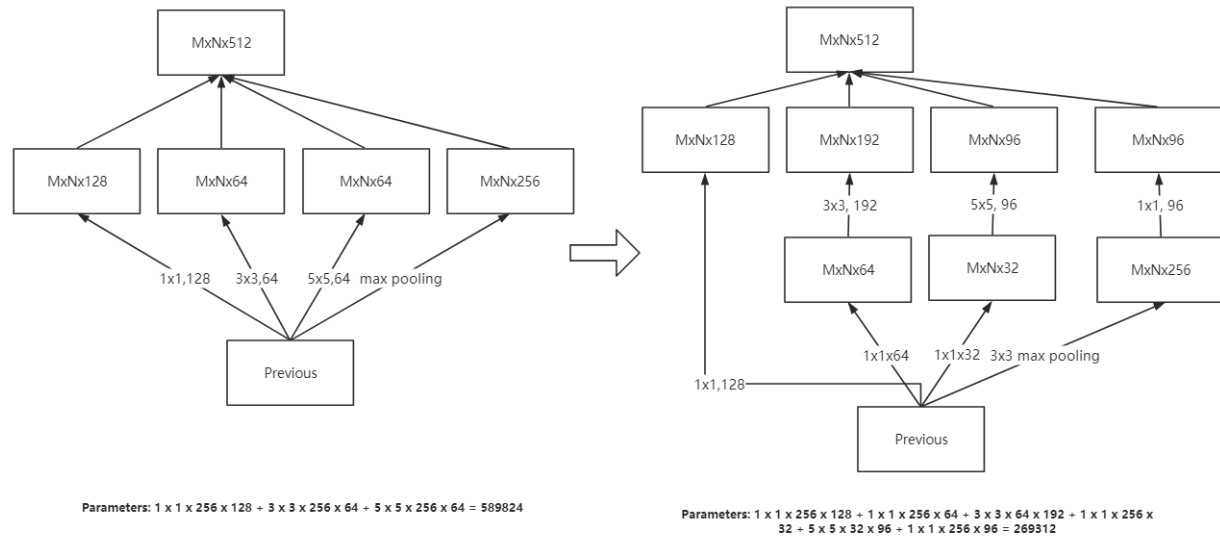


Figure 2.21. The inception dimension version reduces parameters

The GoogleLeNet structure is as follow:

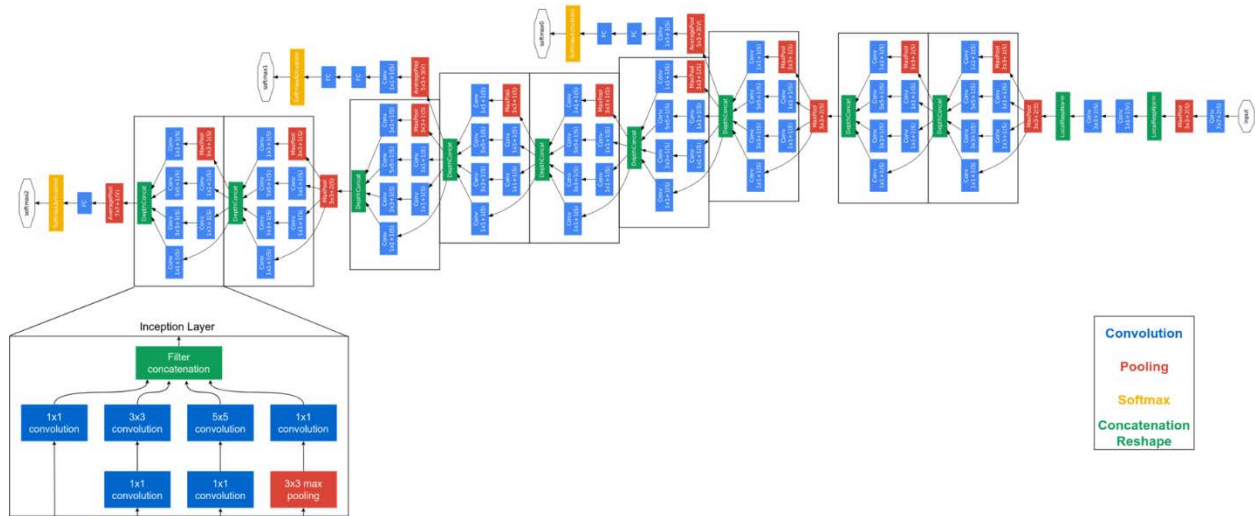


Figure 2.22. The GoogleLeNet Architecture [27].

## ResNet

Traditionally, the deeper the network structure of deep learning, the better the effect and ability of the network. Therefore, in the subsequent network design, designers always make the network design as complicated and deep as possible, but the results are often disappointing, and the network is in trouble. One of the reasons for this problem is that the gradient will disappear. But now the

shallow network can't significantly improve the recognition effect of the network, so the problem to be solved now is how to solve the problem of gradient disappearing while the network becomes deep.

The idea of ResNet [28] could track back to Highway Network [28], allowing the original input to be passed directly to the later layers, the basic ResNet structure as shown in the following Figure 2.23.

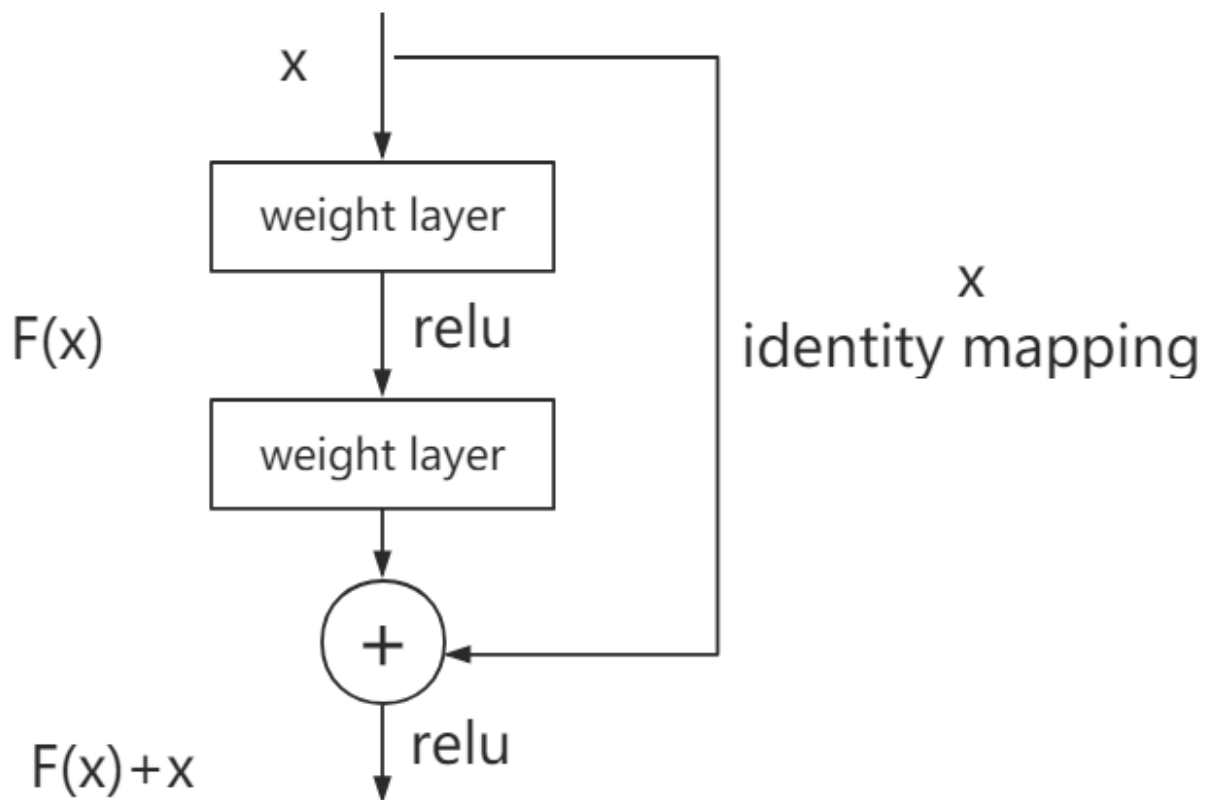


Figure 2.23. The core idea about ResNet [28].

By superimposing layers of  $y=x$  (called identity mappings) on a shallow network, we can increase the network with depth without causing degradation of performance. This reflects that multi-layer nonlinear networks cannot approach an identity mapping network. However, not degrading is not our goal, we hope to have a better performance network. ResNet learns the residual function  $F(x)$



$= H(x) - x$ , where  $F(x) = 0$ , then the identity map mentioned above. In fact, ResNet is a "shortcut connections" in the special case of connections under the identity mapping, it does not introduce additional parameters and computational complexity. If the optimization objective function is to approximate an identity map instead of a zero map, then learning to find the perturbation of the identity map is easier than re-learning a mapping function. The residual function generally has a small response fluctuation, indicating that the identity mapping is a reasonable preprocessing.

### ***SENet***

Squeeze-and-Excitation Networks (SENet) [30] is the new network architecture proposed by the Momenta Huijie team (WMW). Using SENet, the winner of the imageNet 2017 competition Image Classification mission, reduced top-5 error to 2.251% on the imageNet dataset, the original best score was 2.991%.

The method is to insert the SENet block into the existing multiple classification networks. This method has achieved good results. The motivation is to explicitly model the interdependencies between feature channels. In addition, the author did not introduce a new spatial dimension to fuse between feature channels but adopted a new "feature recalibration" strategy. Specifically, it is to learn the importance of each feature channel automatically by learning, and then to enhance the useful features according to this importance and suppress the features that are not useful for the current process.

The SE block is embedded in some original classification networks and inevitably adds some parameters and calculations, but it is still acceptable in the relative of effects. The Squeeze-and-Excitation (SE) block is not an integrated network structure, which could integrate with other architype network to enhance the performance.

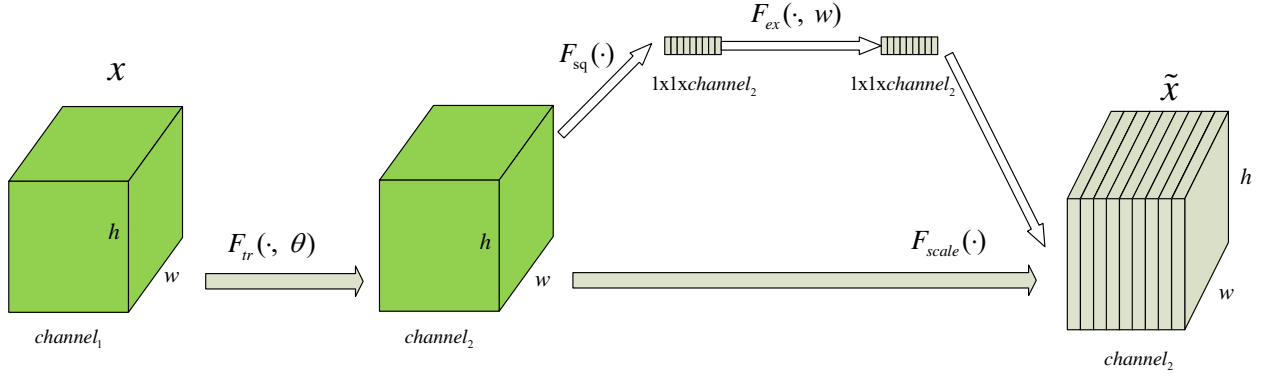


Figure 2.24. The SENet Architecture.

Figure 2.25 is a schematic diagram of the SE module. Given an input  $x$ , the input channel is  $channel_1$ , after a convolutional layer  $F_{tr}(\cdot, \theta)$ , the new channel number will be  $channel_2$ . Unlike the traditional convolutional neural network, the following three functions  $F_{sq}(\cdot)$ ,  $F_{ex}(\cdot, w)$  and  $F_{scale}(\cdot)$  could modify and update the final output layer  $\tilde{x}$ .

The  $F_{sq}(\cdot)$  function will compress the previous layer 3-dimensional data into two-dimensional data. Those data contain the global information. The second function  $F_{ex}(\cdot, w)$  is an activation function, which will reflect the correlation between following layer and previous layer. Finally, a scale operation reweights the feature map weights as input and produces a collection of upper layer.

## 2.6.2 Object Detection

### *R-CNN*

The full name of R-CNN [31] is region-convolutional neural network, which is arguably the first algorithm to successfully apply deep learning to target detection. Traditional target detection methods are mostly based on image recognition. Generally, we can use the exhaustive method to select the area frames that all objects may appear on the picture, extract the features from these area frames and classify them by image recognition method, and obtain all the areas with successful classification, and pass non-maximum value suppression output results.

R-CNN follows the idea of traditional target detection, and also uses the extraction frame to perform target detection on each frame extraction feature, image classification, and non-maximum value suppression. In the step of extracting features, traditional features (such as SIFT [3], HOG [2] features, etc.) are replaced by features extracted by deep convolutional networks.

For an input image, R-CNN generates approximately 2000 candidate regions based on the selective search method, and then each candidate region is resized to a fixed sized and sent to a CNN model, and finally a feature vector is obtained. This feature vector is then fed into a multi-class SVM [1] classifier to predict the probability values of the objects contained in the candidate region that belong to each class. Each category trains an SVM [14] classifier to infer the probability value of the category from the feature vector. In order to improve the positioning accuracy, R-CNN finally trained a bounding box regression model and corrected the exact position of the frame through the border regression model.

### ***Fast R-CNN***

In the R-CNN [31] network structure, since the fully connected layer of the convolutional neural network has limitations on the input image size, the image of all candidate regions must undergo deformation transformation before being subjected to feature extraction by the convolutional neural network model. However, the original image information cannot be completely preserved regardless of whether the crop or the warp is used. The spatial pyramid pooling Layer [32] [33] proposed by He Kaiming et al. effectively solves a limit on the size of the input image in the traditional convolutional neural network.

The problem of image distortion is solved, but other problems of R-CNN, such as cumbersome training steps and large disk space overhead, still need to be resolved. In order to solve the problem in which R-CNN training is slow and the space required for training is large, the original author of R-CNN, Ross, Girshick, has improved R-CNN [31] and proposed Fast R-CNN [32], which absorbs the characteristics of SPP-Net [33], the speed of target detection is greatly improved. The Fast R-CNN architecture is shown as follow.

## ***R-FCN***

According to the previous introduction, we already know that the calculation of the fully connection layer is much larger than that of the convolutional layer. In the previous image detection network, all of them have several fully connection layers. For example, in Fast R-CNN [32], there are 300 full connection layers, and the resource of calculation can be imagined to be huge.

Observed in ResNet [28] and GoogleLeNet [27], there is only one fully connection layer in the last layer, unlike VGG has two fully connection layers. So it is natural to think about that we can remove the fully connection layer behind ROI Pooling, RPN (Region Proposal Network) share all pervious convolutional output, ROI Pooling directly followed by one fully connection layer. In fact, the effect is not acceptable. Because classification requires features to have translation invariance, detection requires translation variance which is an accurate response to the translation of the target. Most of the current CNN can do a good job of classification, but it is not good for detection. SPP [33], Fast R-CNN [32] and other methods are convolution before ROI Pooling, which is translation invariant, but once inserted into ROI Pooling, the subsequent network structure no longer has translation invariance. Therefore, the concept of position sensitive score map that R-FCN [34] [35] wants to propose is to integrate the position information of the target into the ROI Pooling.

## ***YOLO***

The above object detection networks are two-stage network, the first stage use a selective search like the object proposal method to generate the area which probably has an object and the second stage is to classify the proposal area and to localize the object. Because of two-stage network, it is hard to debug and optimize the network. Meanwhile, training and testing time become lengthy.

When humans observe the process of recognizing pictures and locating objects, humans can always make a general classification of pictures at a glance. That is to say, the recognition speed of neural networks should be real time. A network like a two-stage network doesn't do very well, so someone thinks about how to turn the network into a one-stage. Joseph Redmon proposed the

new network architecture in 2015 and named it YOLO [4], You Only Look Once. Compared to the architecture of R-CNN family, YOLO provides another way to turn the problem of object detection into a regression problem. Given an input image, it is possible to directly find bounding boxes of the target and classification category of the target at multiple locations of the image.

YOLO is a convolutional neural network which combines the prediction of multiple bounding box locations and categories into single stage. YOLO does not choose the sliding window or the method of extracting the proposal to train the network, but directly selects the whole picture training model. The architecture is shown as follow.

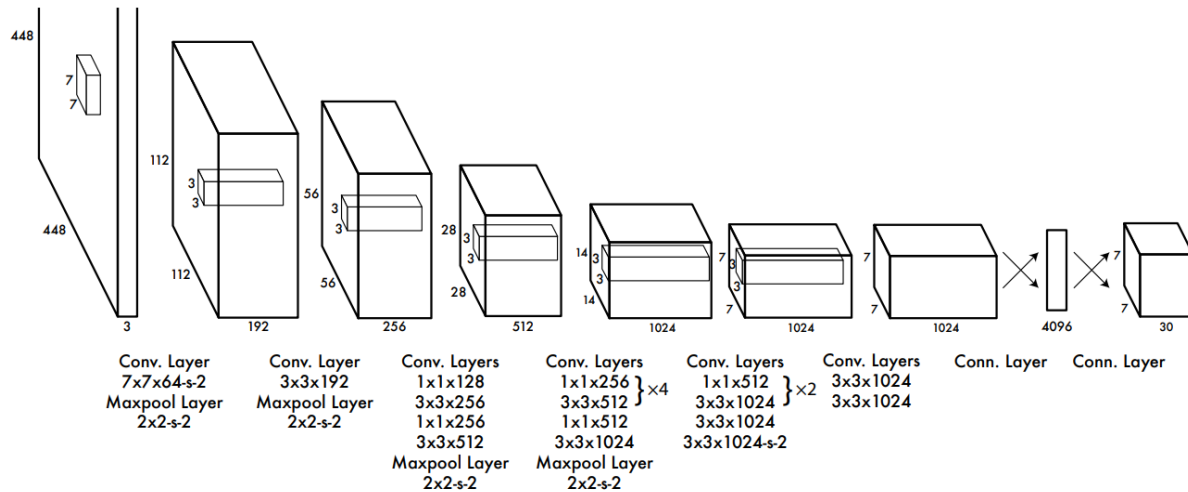


Figure 2.25. The YOLO architecture [4].

## SSD

SSD [37] full name: Single Shot MultiBox Detector, published in 2016. Some researchers consider that SSD is a combination of faster R-CNN and YOLO. First of all, the contribution of SSD is that it takes advantage of the multi-layer network features, not just FC7. This is mainly because SSD still borrows the mechanism of detection into regression. It is like the faster RCNN because it borrows the mechanism of the anchor, but its anchor is not a fine adjustment of every position. It is the same as YOLO to draw the grid, and then create an anchor on the grid. Due to the use of multi-layer features, the anchor scale is different in each layer, so more hyper-parameters are generated, which increases the training difficulty.

SSD is faster than Faster R-CNN and has higher accuracy than YOLO. (At the same time, the accuracy is also high. The results are very good under different test sets).

### ***FPN***

FPN [38], Feature Pyramid Network, is an improvement on the object detection method using convolutional neural networks. By extracting multi-scale feature information for fusion, the accuracy of object detection is improved, especially in the detection of small objects.

The idea of FPN is relatively simple. It mainly uses the feature pyramid to scale the features of different levels, and then integrates the information, so that the lower-level information can be extracted and be more completely reflected, which is more detailed information than the top-level features. The top-level features may neglect some information of small objects in the process of continuous convolution pooling. This method can widely be applied to the detection of small target objects.

As shown Figure 2.31, the high-level features of low-resolution and high-semantic information and the low-level features of high-resolution and low-semantic information are connected from the top to the bottom, so that the features at all scales have rich semantic information.

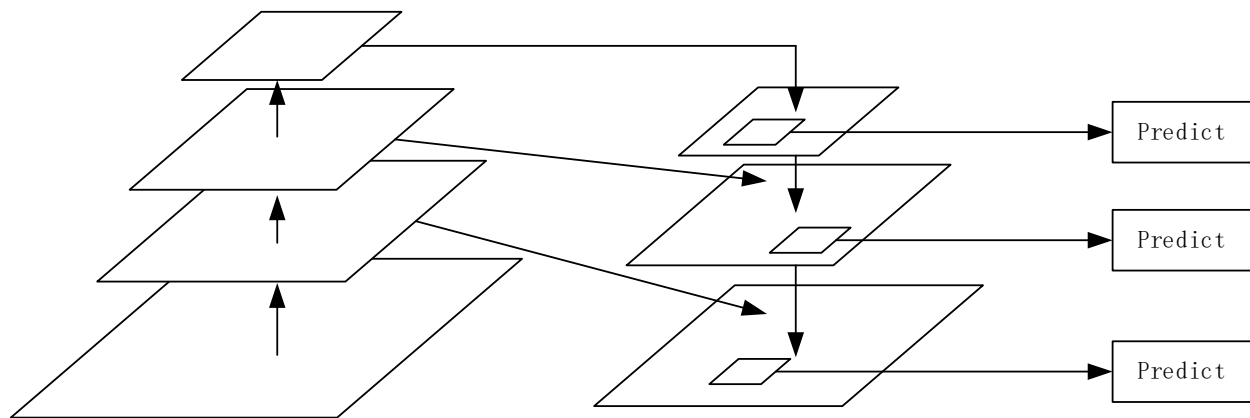


Figure 2.26. The FPN architecture.

The algorithm of FPN can be divided into three parts: a bottom-up convolutional neural network (left), a top-down process (right), and a side connection between features and features.

The bottom-up part is actually the forward process of convolutional neural networks. In the forward process, the size of the feature map changes after passing through some layers, but does not change when passing through other layers, and the layer that does not change the size of the feature map is classified into one stage. Therefore, the characteristics of each extraction are the output of the last layer of each stage, so that it can form a feature pyramid.

The top-down process is performed using upsampling. Upsampling always uses the interpolation method. That is, the original image is enlarged by inserting a new element between pixels by using an appropriate interpolation algorithm based on the original image pixels. As note that, the feature map after upsampling has the same size as the feature map of the next layer.

### ***RetinaNet***

RetinaNet [39] is just a combination of the original FPN network and FCN network, so it has no special innovation in the object detection framework. The biggest innovation in RetinaNet comes from the introduction of Focal loss and the successful application of the single-stage target detection network RetinaNet (essentially ResNet + FPN + FCN). Focal loss is an improved cross-entropy (CE) loss, which successfully solves focal loss by multiplying the original CE loss by an exponential formula that weakens the easy-to-detect object to the model training. In the object detection, the positive and negative sample areas are extremely unbalanced, and the object detection loss is easily affected by large batches of negative samples. This problem is the biggest cause of the single-stage target detection framework (such as the SSD/Yolo series) and the two-stage target detection framework (such as Faster-RCNN/R-FCN).

### **3. METHODOLOGY**

#### **3.1 Preparation of Deep Learning Environment**

##### **3.1.1 Operating System**

Most deep learning tools only work with Linux operating system. Ubuntu is our first choice. Ubuntu is an open-source operating system and it has the biggest community to maintain and update. Our workstation Ubuntu version is 16.04, which is the long-term support version before 26<sup>th</sup> April 2018.

##### **3.1.2 NVIDIA Support**

Since Hinton invent AlexNet and use GPU to help calculating parameters, the crazy of deep learning has begun. GPU computing makes large-scale computing neural network possible. People can use the CUDA [40] (Compute Unified Device Architecture) introduced in 2007 to control the GPU for parallel computing. In our workstation, we use 2 NVIDIA graphic cards, GTX 1080Ti. The version of CUDA is 9.0, which is powerful and supporting well.

##### **3.1.3 Tools**

There are so many deep learning tools. Because the deep learning is the most popular field in recent years, deep learning frameworks are getting more diverse and powerful. Most of this framework based on Python language. The following tools are popular which will be brief introduced respectively.

##### ***Tensorflow***

On November 9<sup>th</sup>, 2015, Google released the deep learning framework, Tensorflow, and announced an open source. In March 2016, Google's AI program, AlphaGO, beat the human player, Shishi Li with a big score of 4 over 1, which caused widespread concern in the world, causing a wave of artificial intelligence. Tensorflow [41] [42] is the most popular framework with the most comprehensive API and powerful community. Tensorboard can visually view the network



structure, which helps researchers understand the network, debug and optimize the network more easily.

### ***Pytorch***

Pytorch [43] is very popular among academic researchers and is a relatively new deep learning framework. Pytorch is a Torch-based Python machine learning library. It was developed in 2016 by Facebook's Artificial Intelligence Research Group. In this thesis, the Pytorch 0.4 is used.

### ***Keras***

Keras [46] is a high-level neural network API for building and testing neural networks. It is written in Python and uses Tensorflow, Theano [44], and CNTK [45] as back-end. Strictly speaking, Keras is not called a deep learning framework, it is more like a deep learning interface. It is built on a third-party framework. Keras' over-encapsulation has made it less flexible and has also caused it to run slower. But Keras is the easiest to learn, and it's very convenient to use Keras to verify some simple models.

### ***Caffe***

Caffe [47], the full name of Convolutional Architecture for Fast Feature Embedding, is a deep learning framework that combines expressiveness, speed and modularity. Developed by the Berkeley Artificial Intelligence Research Group and the Berkeley Center for Visual and Learning. Although its kernel is written in C++, Caffe has Python and MATLAB related interfaces. Caffe is widely used in research institutes.

## **3.2 Motivation**

### **3.2.1 For parking lot classification**

Parking lot detection is ubiquitous. In a big city, especially in a busy and crowded parking lot, for a driver, when he drives into a parking lot, he has to drive around the parking lot by himself, find the parking space subjectively, or find some information board prompts how many vacancies are currently in the parking lot, but it cannot provide accurate information about the parking space. In

order to solve this problem, the current technology is to install a sensor in each parking space, and the sensor will transmit the parking space information in real time, with or without a vehicle. The problem with sensors is that they cannot be visualized or used on a large scale. If it is used on a large scale, the sensor's investment in money in the early stage will be very high. This thesis proposes an idea to use the security cameras around the parking lot to obtain the location information of the parking lot, use the convolutional neural network to make judgments on the parking lot information, and finally return to the parking lot with or without the vehicle information. This method is not only convenient and efficient, but also widely used, because the camera is easy to set up, and it can not only provide vehicle information detection, but also improve surrounding security.

This simple convolutional network is used for a parking lot cars detection project. At the beginning, we download dataset from the open-source parking lot dataset, and then we collect data from our university parking lots. The first thing to do is to mark every parking space in the parking lot and crop the image of the parking space, meanwhile, label each parking space containing the car or not. These labeled parking space consists of the training and testing dataset.

### **3.2.2 For prostate detection and localization**

The detection and analysis of medical images is becoming more and more popular in the field of computer vision. It takes a lot of money and time to develop a professional physician capable of analysis and diagnosis, but with the development of deep learning, these problems can be solved by computers. This work addresses the problem of prostate location. The prostate is a place where inflammation often occurs. For doctors, it is very important to be able to accurately locate the prostate on the image and analyze the lesions. YOLO [4] [49] network is a fast and efficient target positioning network. By modifying the network structure and changing the output, the purpose of prostate identification and positioning is achieved.

### 3.3 Dataset Image Processing

#### 3.3.1 Convolutional Neural Network for Classification

##### *Dataset source*

The parking lot dataset source is named as PKLot Dataset which is from Universidade Federal do Paraná (UFPR) [48]. The dataset has 12,417 images of parking lots and 695,899 images of parking spaces segmented from them.

##### *Crop images*

Except the downloading dataset, we have our own parking lots as well. For those parking lots, we should create the own segmented parking spaces dataset. The segmented parking spaces extracted from each whole parking lot, there are areas that are not the vertical rectangle like Figure 3.1. The skew adjustment is used, which is different from the method proposed in [48]. We rotate any slop image to 90° degree as shown in Figure 3.1.

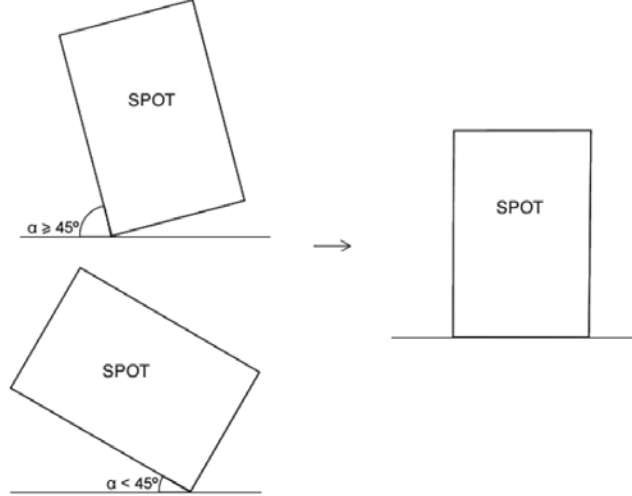


Figure 3.1. The rotation method.

### 3.3.2 YOLO Network for Object Detection and Localization

#### *Prostate image dataset source*

The dataset is from MICCAI Grand Challenge [50]. The dataset name is PROMISE12. There are 50 training cases available, which include a transversal T2-weighted MR image of the prostate. For each of the cases a reference segmentation is also included.

#### *Pre-processing images*

The dataset image format is test-based tagged file format for medical images. The dataset is 3-dimensional image with a raw file, which contains the location information. In order to use the dataset, the first step is to convert the mdh/raw file into a common jpg/txt file [51], which stores the image information and the bounding box information corresponding to the image.

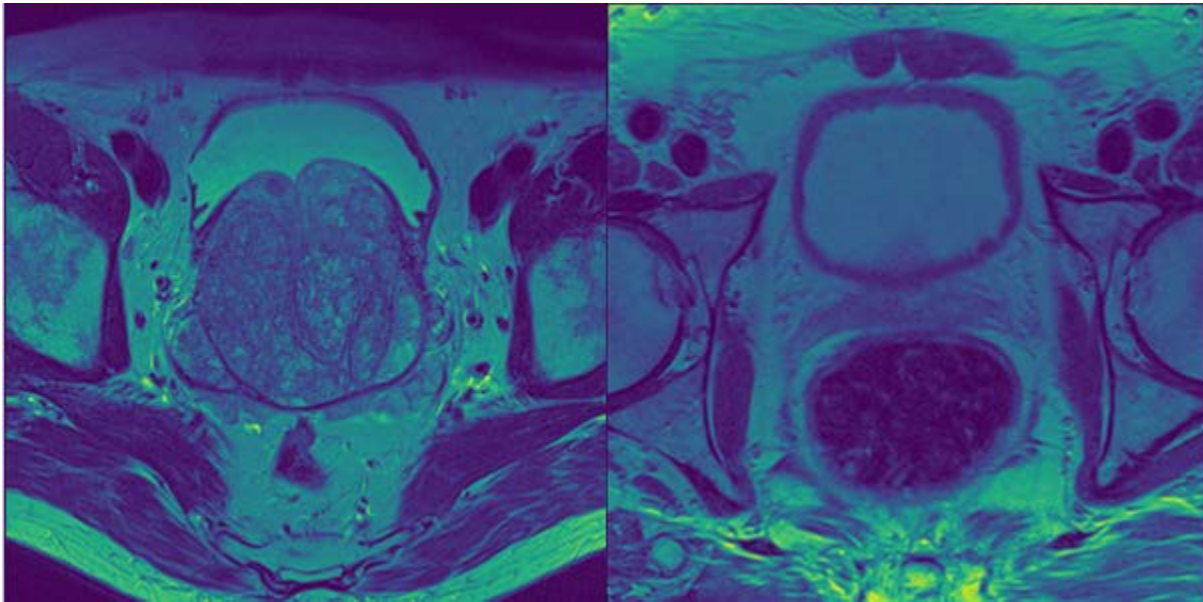


Figure 3.2. The extracted prostate images.

### 3.4 Classification Network Implementation

#### 3.4.1 Architecture

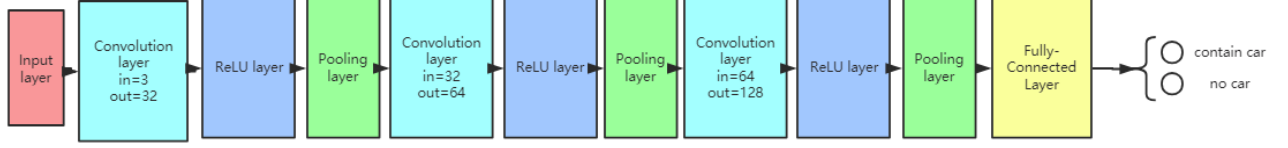


Figure 3.3. The customized convolutional neural network.

The overview about classification convolutional neural network architecture is shown in Figure 3.3. It has 3 convolutional blocks, which consists of a convolutional layer, a ReLU layer and a pooling layer. Before the output layer, there is a fully connected layer, which makes classification function.

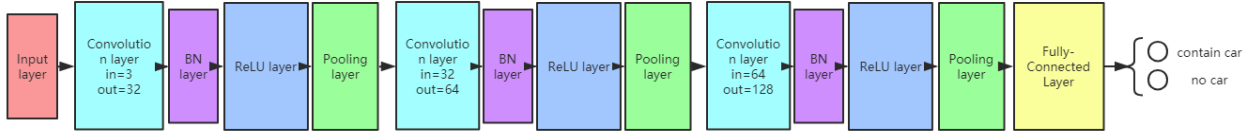


Figure 3.4. The Convolutional Neural Network with BN layer.

Figure 3.4 shows the new architecture with batch normalization layer.

#### 3.4.2 Loss Function

For this CNN, we use the cross-entropy to calculate the loss value. The cross-entropy formula as follow.

$$cross\ entropy = - \sum_{k=1}^N p_k * \log(q_k) \quad (30)$$

$$Softmax = \frac{\exp(x[class])}{\sum_{j=0}^N \exp(x[j])} \quad (31)$$

where  $p_k$  represents the true/target value, which is a one hot form in this formula;  $q_k$  is the predicted value, assuming that it is already the result after passing Softmax function. N denotes the number of the classes. When we remove the  $p_k$  component, the expression change to as follow.

$$loss = -\log\left(\frac{\exp(x[class])}{\sum_{j=0}^N \exp(x[j])}\right) = -x[class] + \log\left(\sum_{j=0}^N \exp(x[j])\right) \quad (32)$$

### 3.4.3 Evaluation Metrics in Classification Network

The simplest and most intuitive evaluation metrics is accuracy. For the test dataset with N samples, the network output contains M correct prediction samples, simple division is performed to obtain the accuracy as criterion. M contains the true positive predictions and true negative predictions.

$$Accuracy = \frac{M}{N} \quad (33)$$

For many times, there is no problem with accuracy as the evaluation criteria. But it is unable to evaluate if the network is good or not. There is an example, the network can identify whether an email is spam or not. There are 100 test data, 98 of them are spam and 2 of them are non-spam. Assuming the network always predicts an incoming email as spam, then the network's accuracy is 98%, and the number shows it is the pretty good network, but it doesn't match our actual needs. In order to evaluate the network more efficiency, the confusion matrix [52], is introduced. Figure 3.5 shows the confusion matrix.

Table 3-1. Confusion matrix

		<b>Predicted</b>	
		<b>Negative</b>	<b>Positive</b>
<b>Actual</b>	<b>Negative</b>	True Negative	False Positive
	<b>Positive</b>	False Negative	True Positive

Using this confusion matrix, we could calculate two metrics to evaluate the network, which are precision and recall.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (34)$$

$$Precision = \frac{TP}{TP + FP} \quad (35)$$

$$Recall = \frac{TP}{TP + FN} \quad (36)$$

Note that precision is the proportion of correct prediction positive samples to the total predicted positive samples. Recall is the proportion of correct prediction positive samples to the all samples in actual positive samples.

### 3.5 YOLO Model Implementation

#### 3.5.1 IOU Introduction

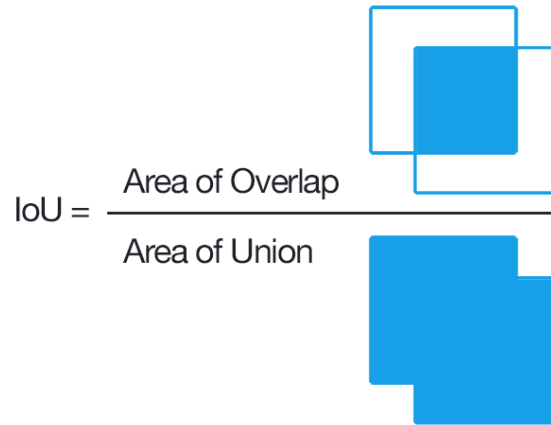


Figure 3.5. The IOU calculation.

In the evaluation system of object detection, there is a parameter called IOU [56], Intersection over union, which is simply the overlap ratio between the two boxes. In Figure 3.7, the first step is to calculate the area of two boxes overlap, then calculating the area of two boxes union. Dividing the first calculated area by the second calculated area leads to the IOU value.

### 3.5.2 Non-Maximum Suppression



Figure 3.6 The Non-maximum example. The multiple bounding boxes (left), the optimal bounding box (right).

In object detection, a large number of candidate bounding boxes are often processed by using non-maximum suppression algorithm (NMS) [57] to remove redundant candidate bounding boxes and obtain an optimal detection bounding box to clearly show the object. The essential idea is to search for local maximum value and suppress non-maximum values. Figure 3.7 is for the detection about horse. In the left, that is the image before the NMS. After the network prediction, there are several bounding boxes produced, with several coordinates and confidence value. Above figure shows one object detection with the different confidence values for each predicted bounding box, first the NMS will sort this bounding boxes depending on the confidence value from big to small, then NMS calculates the IOU with remaining bounding box using the bounding box with the maximum confidence value. If some IOU of the remaining bounding boxes are greater than the threshold, these bounding boxes are discarded. Immediately after, NMS selects next bounding box with the maximum confidence value in the remaining bounding boxes to do the same IOU calculation and comparison operation until there is no remaining bounding box left.

### 3.5.3 Encoding Method in Pre-Processing

YOLO converts object recognition and object localization into a regression problem. Since it is a regression problem, there must be a one-to-one correspondence between input and output. The label information only needs to convert the character information into the corresponding digital number to achieve the encoding requirement, but for the position information, it cannot be simply



processed. At this point, it is necessary to perform certain encoding processing on the input information, so that the input and output are related. YOLO is encoded by dividing an image into  $S \times S$  grid cells. If the center of an object grounding truth box falls in the grid, the grid is responsible for prediction of this object. Each grid has to predict  $B$  bounding boxes. In addition to regress its own position, each box is also accompanied by a prediction of a confidence value. This value represents whether the predicted bounding box contains an object. If it contains an object, the confidence value is equal to the IOU of the ground truth box and the bounding box of this object. If not, the confidence should be 0.

$$Confidence = Pr(Object) * IOU_{pred}^{truth} \quad (37)$$

Each bounding box has 5 predictions:  $x$ ,  $y$ ,  $w$ ,  $h$  and confidence. The  $(x, y)$  represents the center of the bounding box related to the relative grid cell. And  $w$ ,  $h$  represent width and height of the bounding box to the whole image.

The location information encoding method of YOLO is the similar to that of the faster R-CNN [58], mainly based on the offset of the position information in the image, so that the width and height information can be distributed between 0 and 1.

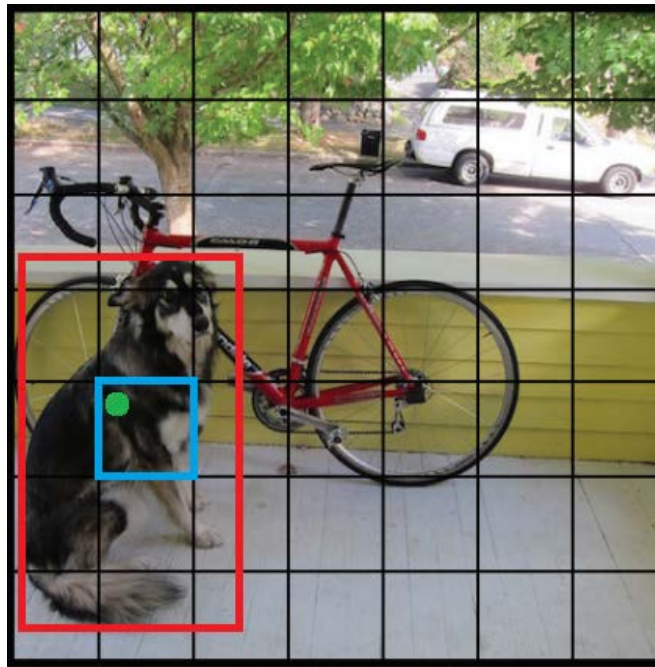


Figure 3.7. The Encode method.

As shown in Figure 3.8, the image is divided into 7 x 7 grid cells, and we set the cell in the upper left corner is (0, 0), increasing from left to right and top to bottom. The red box is the grounding truth box, and the green dot represents the center of the red box. As mentioned before, the grid cell where the green point is located is responsible for predicting the red ground truth box, thus predicting whether there is a dog or where the dog is. We set the width and height of the whole image to be  $w_i, h_i$ , the red ground truth box is  $w_g, h_g$ , and the coordinates of the center of the red ground truth box are  $(x_g, y_g)$ .

In order to make the width and height distributed between 0 and 1, we set  $w_{go} = \frac{w_g}{w_i}, h_{go} = \frac{h_g}{h_i}$ .

But for (x, y) encoding, we need to use the offset of the grid cell. In Figure 3.8, the grid cell is located in column 1 and row 4. Assume  $(x_{go}, y_{go})$  represents the offset which is the center of ground truth box relate to the grid cell.

$$x_{go} = \text{ceil}\left(\frac{x_g}{w_i} * S\right) - \text{column} \quad (38)$$

$$y_{go} = \text{ceil}\left(\frac{y_g}{h_i} * S\right) - \text{row} \quad (39)$$

### 3.5.4 Loss Function

The loss function is written below:

$$\begin{aligned} \text{loss} = & \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{I}_{ij}^{obj} \left[ (x_p - x_{go})^2 + (y_p - y_{go})^2 \right] \\ & + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{I}_{ij}^{obj} \left[ (\sqrt{w_p} - \sqrt{w_{go}})^2 + (\sqrt{h_p} - \sqrt{h_{go}})^2 \right] \\ & + \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{I}_{ij}^{obj} \left[ (C_i^p - C_i^g)^2 \right] + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{I}_{ij}^{noobj} \left[ (C_i^p - C_i^g)^2 \right] \\ & + \sum_{i=0}^{S^2} \mathbb{I}_{ij}^{obj} \sum_{c \in \text{classes}} \left[ (P_i^p(c) - P_i^g(c))^2 \right] \end{aligned} \quad (40)$$

$(x_p, y_p)$  stands for the center of prediction bounding box offset to the grid cell, and the  $(x_{go}, y_{go})$  stands for the center of ground truth box offset to the grid cell.  $\|_{ij}^{obj}$  means the grid cell contain the object the  $j^{\text{th}}$  bounding box response to this object, calculate error using this bounding box coordinates and confidence score.  $C_i^p, C_i^g$  respectively represents confidence score in prediction and ground truth.  $P_i^p(c), P_i^g(c)$  represent the classes possibility for each class in prediction and ground truth. The above loss function combines the localization error and classification error. The weight of these two errors should not occupy the same weight in the expression. Because in the image, the number of ground truth boxes is much less than the number of prediction bounding boxes, this unbalance output would make the loss function unstable. The Joseph Redmon [4] adds two super-parameters in this function  $\lambda_{coord}$  and  $\lambda_{noobj}$ , respectively set the value 5 and 0.5. These two super-parameters mean that when the grid cell contains an object, the loss function is more sensitive in the classification error, if not contains, the loss function should merely ignore the classification error. Because in the image, the number of ground truth boxes is much less than the number of prediction bounding boxes, this unbalance output would make the loss function unstable. To make more stable for this formula, we use the square root of the bounding box and ground truth box width and height. The sum-squared error will be stable and small after the square root of width and height.

### 3.5.5 Decoding Method in Prediction

The process of decoding is the inverse of encoding. Decoding is to display the bounding box relative to the entire image when the network makes prediction, rather than the bounding box relative to the grid cell. According to the previous coding process,  $(x_g, y_g)$  relative to the whole image can be derived from the following formula.

$$x_g = \frac{x_{go} + column}{S} * w_i \quad (41)$$

$$y_g = \frac{y_{go} + row}{S} * h_i \quad (42)$$

### 3.5.6 Evaluation of Network using mAP

Localization is no longer easy and clear to evaluate the quality of the network compared to classification. In this case, some new metrics for object detection is needed. The establishment of these standards is based on different online competitions, such as PASCAL VOC Challenge [55]. COCO Detection Challenge [59], Google Open Images Dataset V4 Competition, ImageNet Object Localization Challenge [60]. The evaluation mAP is the widely and commonly used in objects detection network, which stands for mean Average Precision. The generally understood about mAP [53] [54] is the area under the precision-recall curve. The ordinate is the precision, and the abscissa is the recall, as the recall rate increases, the precision decreases.

### 3.6 Customization of YOLO Model

The original YOLO backbone network is DarkNet [4], which is the network based on C. In this thesis, we use ResNet50 to replace the DarkNet, because the problem in this thesis is one object detection, therefore, the network structure does not need to be too complicated. In the original paper [4], the end of DarkNet is connected with 2 fully connected layers. We remove the fully connected layer and use the customized convolutional layer for the final output.

After we change the backbone network, we use the original YOLO and our new architecture to evaluate mAP. The performance is better than the original one.

Table 3-2. The performance of two YOLO network.

Network architecture	Map for voc2012	FPS
YOLO with ResNet	68.32%	58
Original YOLO	62.13%	42

As can be seen in Table 3-1, the mAP score is almost 6% increased, and the FPS is faster than the original YOLO architecture.

## 4. EVALUATION AND DISCUSSION

### 4.1 Parking Lot Output

The training and validation results for parking lot are displayed in Figure 4.1.

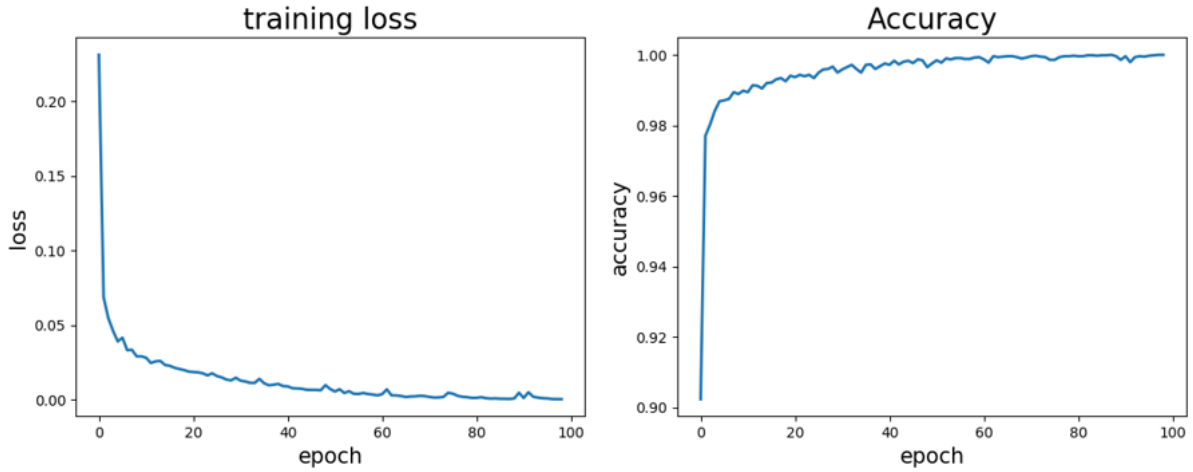


Figure 4.1 Training loss (left) and Validation dataset accuracy (right).

In Figure 4.1, we can see the loss value drops dramatically down before 20 epochs, and slowly decreases down to 0 at 100 epochs. The accuracy depends on the 2000 validation images, the performance of validation dataset relates to the training loss, the accuracy is getting higher while the training loss is dropping down. The final accuracy is almost 100%. The fault detection parking space is shown as follow. We are using 500 images to test our model. The precision and recall as shown in Table 4-1.

Table 4-1. Precision and recall at first training.

Precision	Recall
0.9492	0.972

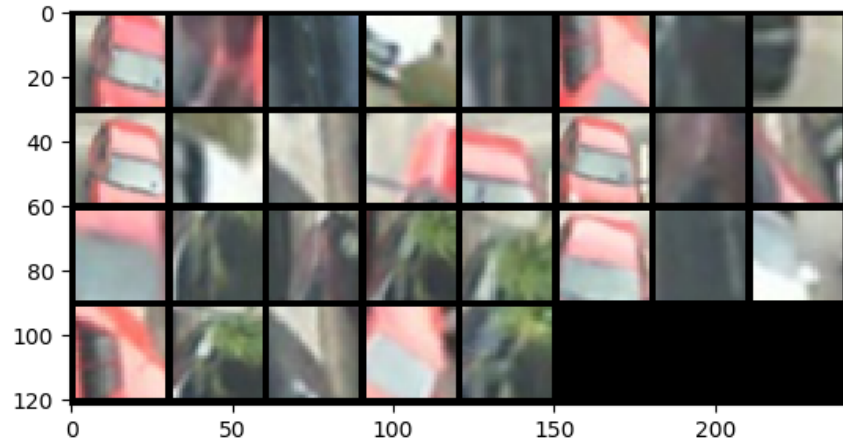


Figure 4.2. Wrong detection images

As we can see, the wrong detections are almost all red cars, we check our training dataset, we calculate the red cars only obtain 7% of all training dataset, then we use data augmentation method, rotate, crop or flip to get more red cars images. After that, we re-train our model, and test our model again.

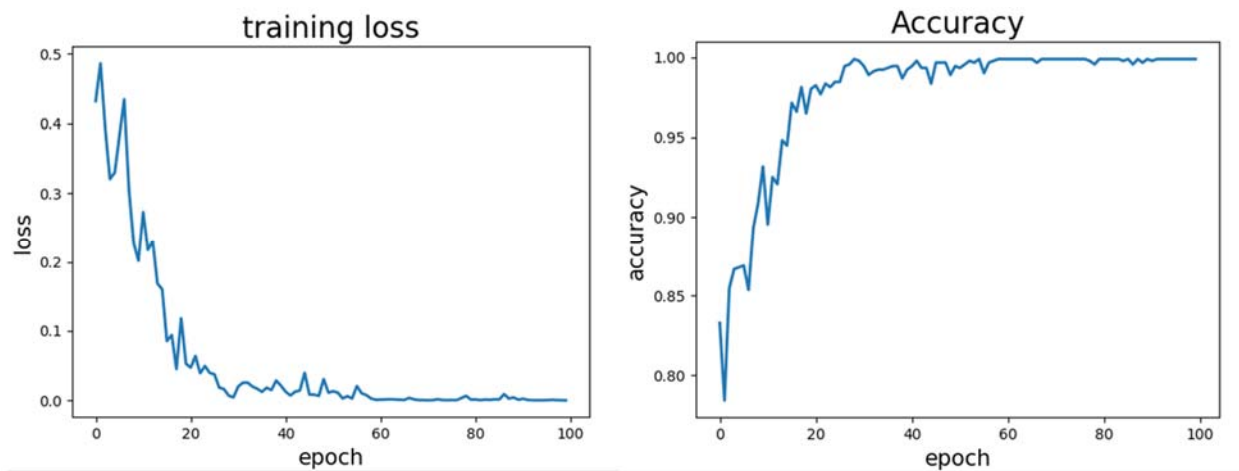


Figure 4.3. Training loss (left) and Validation dataset accuracy (right) at second time.

For this time, we increase the learning rate at beginning, set value as 0.01 and set 0.001 at 50 epochs. The training loss is approximately 0 at final epoch, and the accuracy using test dataset is 100%. Recall is equal to 1 and precision is 0.98.

Table 4-2. Precision and recall at second training.

Precision	Recall
0.98	1

## 4.2 Prostate Output

Figure 4.4 shows the training and validation results for the application of prostate detection.



Figure 4.4. The first training for 50 epochs, the blue line (1) is the training data loss, the orange line (2) is the validation data loss.

As shown in Figure 4.4, the training loss trending decreases dramatically at beginning several epochs, because the backbone network is pre-trained, which load the weights from the ResNet-50 all layers except the fully connected layer. After first 20 epochs training, the loss is almost at the minimum. This proves that our network works, which is learning from the training data to predict whether the prostate is contained and the location of the prostate.

We try to use this model to test the 778 prostate images. The accuracy for prediction of contain prostate images is only 80.2%. This result is far lower than our expectation. The average IOU of all correct prediction is 0.736, which is not reasonable.

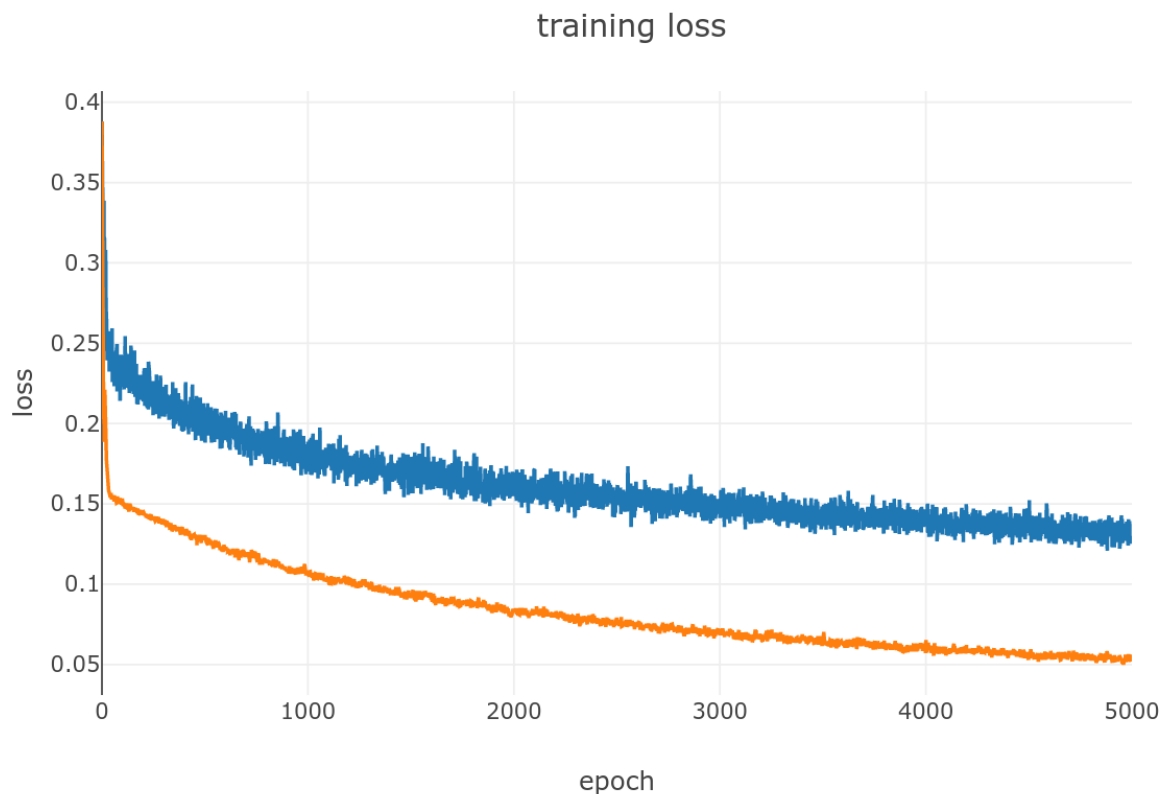


Figure 4.5. The training for 5000 epochs, the blue line (1) is the training data loss, the orange line (2) is the validation data loss.

We continue to train our model for 5000 epochs, the training loss trend is decreasing. The accuracy for prediction of containing prostate images reaches 99.2%. This result is much better than above network. The average IOU of all correct predictions is 0.884.



Table 4-3. False Positive for network that predicted 6 contained prostate images as non-contained prostate images, and location information missed.

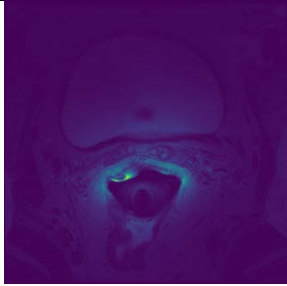
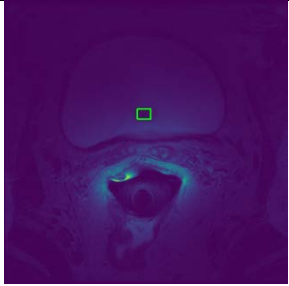
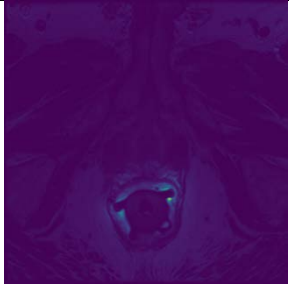
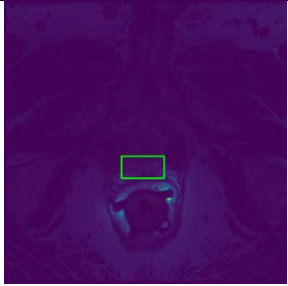
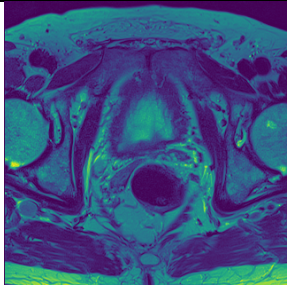
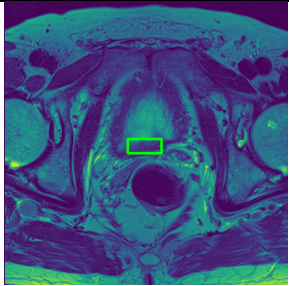
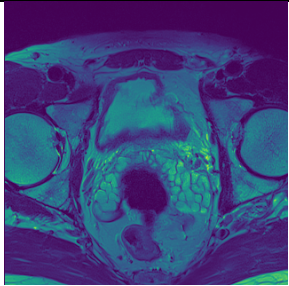
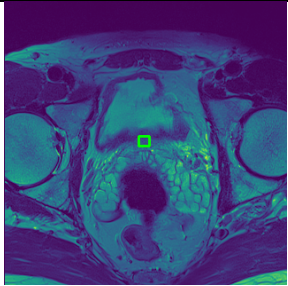
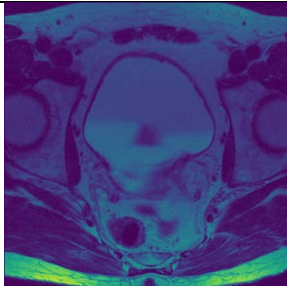
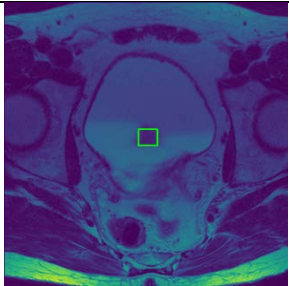
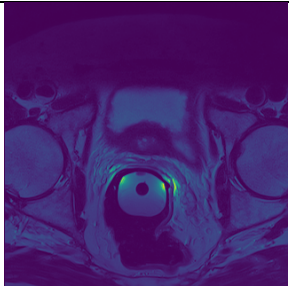
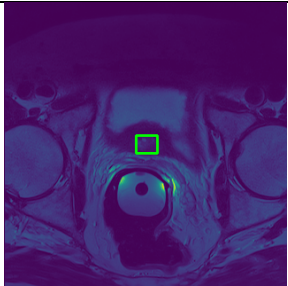
Prediction	Ground truth	Prediction	Ground Truth
			
Prediction	Ground truth	Prediction	Ground truth
			
Prediction	Ground truth	Prediction	Ground truth
			

Table 4-3 shows the false positive prediction images. The prediction images should note the image with prostate, but they actually do not. After reviewing predecessors work, we concluded that this problem can happen because of the YOLO method shortage. The YOLO sets a 7 x 7 grids in the image, thus it is hard to detect the small objects. In our model, we set the grid size to 14.

### 4.3 Future Work

Regarding the classification of parking lots with or without vehicles, the corresponding parking lot will be optimized successively.

1. Collect information about specific parking lots, such as campus parking lots.

2. Division and numbering of parking lot information
3. Make a real-time detection system and develop a mobile application

In the detection and location of the prostate, we found that it is difficult to detect the prostate in a small area. The accuracy of neural networks for detecting and locating small targets is very low. Sometimes when the detection area is small, the accuracy of the network is almost zero. This problem has not been effectively solved. The network structure can also be updated later. For setting the grid size, we can increase the value as much as possible. But the problem is that the training time and detection time will increase. We can also investigate a new network structure, YOLO-v3.

## 5. CONCLUSION

In chapter 1 (Introduction), we briefly introduce the concept of deep learning and the implementation in computer vision field. The most important part is the object detection and recognition, which are the main topic in this thesis.

In chapter 2 (Background), we review the traditional machine learning methods and the neural network. The new concept of neural networks influences the trend of machine learning and the future. Comparing with traditional machine learning, the problem targeted is often linear correlation. The neural network uses an activation function to transform linear correlations into nonlinear correlations, so we review some common activation functions in machine learning and deep learning. After we review the updating parameter method of neural network, the first step is forward propagation, which makes the input pass through the entire network. The second step uses loss function to calculate the error between output and true value. Next, using the gradient descend method and back propagation updates all weights in each neuron. The next part is the concept of convolutional neural network, which is the main topic of the thesis. We review the background and the definition about CNN, and then introduce the architecture of CNN. As the final part, we review the state-of-the-art CNN models.

In chapter 3 (Methodology), at beginning we prepare the working environment and choose the advanced deep learning tools. For deep learning, dataset is the most important part. We collect parking lot dataset and prostates dataset, one is used for object classification, and the another one is used for object localization. Then we introduce our classification network architecture and localization network architecture.

In chapter 4 (Evaluation and Discussion), we review and evaluate our model and verify performance, and discuss some problems and improvements in our models.

## REFERENCES

- [1] “Artificial neural network.” [Online]. Available: [https://en.wikipedia.org/wiki/Artificial\\_neural\\_network](https://en.wikipedia.org/wiki/Artificial_neural_network).
- [2] Navneet Dalal, Bill Triggs. “*Histograms of Oriented Gradients for Human Detection.*” International Conference on Computer Vision & Pattern Recognition (CVPR '05), Jun 2005, San Diego, United States. pp.886–893, ff10.1109/CVPR.2005.177.
- [3] Lowe, D.G. “*Distinctive Image Features from Scale-Invariant Keypoints.*” International Journal of Computer Vision (2004) 60: 91.
- [4] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi. “*You Only Look Once: Unified, Real-Time Object Detection.*” The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779-788.
- [5] George A. F. Seber, Alan J. Lee. “*Linear Regression Analysis.*” Jan 20, 2012.
- [6] David G. Kleinbaum, Mitchel Klein. “*Logistic Regression: A Self-Learning Text.*” Jun 14, 2010.
- [7] Richard S. Sutton and Andrew G. Barto. “*Reinforcement Learning: An Introduction.*” Chapter 2.3 Softmax Action Selection.
- [8] David Marr. “*Marr's Three Levels: A Re-evaluation.*” Published in Minds and Machines, May 1991.
- [9] Warren McCulloch and mathematician Walter Pitts. “*A logical calculus of the ideas immanent in nervous activity.*” Published in: Bulletin of Mathematical Biophysics, Vol. 5, 1943, p. 115-133.
- [10] Li J., Cheng J., Shi J., Huang F. (2012) “*Brief Introduction of Back Propagation (BP) Neural Network Algorithm and Its Improvement.*” In: Jin D., Lin S. (eds) Advances in Computer Science and Information Engineering. Advances in Intelligent and Soft Computing, vol 169. Springer, Berlin, Heidelberg.
- [11] Anderson, J. A., 1995, “*Introduction to Neural Networks*” (Cambridge, MA:MIT Press).
- [12] Nielsen M A. “*Neural networks and deep learning*” [M]. 2015. <http://neuralnetworksanddeeplearning.com/>
- [13] Tianyi Liu, Shuangfang Fang, Yuehui Zhao, Peng Wang, Jun Zhang. “*Implementation of Training Convolutional Neural Networks.*” arXiv: 1506.01195.

- [14] D. Meyer, “*Support Vector Machines*,” R News, 2001.
- [15] A. Bhaya, “*Neural networks, in Decision Sciences: Theory and Practice*,” 2016.
- [16] Paul Dawkins, “Paul's Online Notes.” [Online]. Available: <http://tutorial.math.lamar.edu/Classes/CalcI/ChainRule.aspx>
- [17] Giovanni Berlucchi (2006) Revisiting the 1981 Nobel Prize to Roger Sperry, David Hubel, and Torsten Wiesel on the Occasion of the Centennial of the Prize to Golgi and Cajal, Journal of the History of the Neurosciences.
- [18] Wikipedia, “*Convolutional Neural Network*.”
- [19] H. Il Suk, “*An Introduction to Neural Networks and Deep Learning*,” Deep Learning for Medical Image Analysis, 2017.
- [20] LeCun, Yann. “*LeNet-5, convolutional neural networks*”. Retrieved 16 November 2013.
- [21] Normalization, Batch. “*Accelerating deep network training by reducing internal covariate shift*.” CoRR.–2015.
- [22] Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton. “*Layer normalization*.” arXiv preprint arXiv:1607.06450 (2016).
- [23] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. “*Imagenet classification with deep convolutional neural networks*.” Advances in neural information processing systems. 2012.
- [24] Bouthillier, Xavier, et al. “*Dropout as data augmentation*.” arXiv preprint arXiv: 1506.08700 (2015).
- [25] Srivastava, Nitish, et al. “*Dropout: a simple way to prevent neural networks from overfitting*.” The journal of machine learning research 15.1 (2014): 1929-1958.
- [26] Simonyan, Karen, and Andrew Zisserman. “*Very deep convolutional networks for large-scale image recognition*.” arXiv preprint arXiv: 1409.1556 (2014).
- [27] Szegedy, Christian, et al. “*Going deeper with convolutions*.” Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.
- [28] He, Kaiming, et al. “*Deep residual learning for image recognition*.” Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
- [29] Short-Term Load Forecasting based on ResNet and LSTM - Scientific Figure on ResearchGate. Available from: [https://www.researchgate.net/figure/The-structure-of-ResNet-12\\_fig1\\_329954455](https://www.researchgate.net/figure/The-structure-of-ResNet-12_fig1_329954455)

- [30] Hu, Jie, Li Shen, and Gang Sun. “*Squeeze-and-excitation networks.*” Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.
- [31] Girshick, Ross, et al. “*Rich feature hierarchies for accurate object detection and semantic segmentation.*” Proceedings of the IEEE conference on computer vision and pattern recognition. 2014.
- [32] Girshick, Ross. “*Fast R-CNN.*” Proceedings of the IEEE international conference on computer vision. 2015.
- [33] He, Kaiming, et al. “*Spatial pyramid pooling in deep convolutional networks for visual recognition.*” IEEE transactions on pattern analysis and machine intelligence 37.9 (2015): 1904-1916.
- [34] Dai, Jifeng, et al. “*Instance-sensitive fully convolutional networks.*” European Conference on Computer Vision. Springer, Cham, 2016.
- [35] Dai, Jifeng, et al. “*R-FCN: Object detection via region-based fully convolutional networks.*” Advances in neural information processing systems. 2016.
- [36] Redmon, Joseph, et al. “*You only look once: Unified, real-time object detection.*” Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
- [37] Liu, Wei, et al. “*SSD: Single shot multibox detector.*” European conference on computer vision. Springer, Cham, 2016.
- [38] Lin, Tsung-Yi, et al. “*Feature pyramid networks for object detection.*” Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
- [39] Lin, Tsung-Yi, et al. “*Focal loss for dense object detection.*” Proceedings of the IEEE international conference on computer vision. 2017.
- [40] Kirk, David. “*NVIDIA CUDA software and GPU parallel computing architecture.*” ISMM. Vol. 7. 2007.
- [41] Abadi, Martín, et al. “*Tensorflow: A system for large-scale machine learning.*” 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16). 2016.
- [42] Abadi, Martín, et al. “*Tensorflow: Large-scale machine learning on heterogeneous distributed systems.*” arXiv preprint arXiv:1603.04467 (2016).
- [43] Ketkar, Nikhil. “*Introduction to Pytorch.*” Deep learning with python. Apress, Berkeley, CA, 2017. 195-208.

- [44] Team, The Theano Development, et al. “*Theano: A Python framework for fast computation of mathematical expressions.*” arXiv preprint arXiv:1605.02688 (2016).
- [45] Seide, Frank, and Amit Agarwal. “*CNTK: Microsoft's open-source deep-learning toolkit.*” Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2016.
- [46] Ketkar, Nikhil. “*Introduction to Keras.*” Deep Learning with Python. Apress, Berkeley, CA, 2017. 97-111.
- [47] Vision, Berkeley, and Learning Center. “*Caffe.*” (2013).
- [48] De Almeida, Paulo RL, et al. “*PKLot—A robust dataset for parking lot classification.*” Expert Systems with Applications 42.11 (2015): 4937-4949.
- [49] Laroca, Rayson, et al. “*A robust real-time automatic license plate recognition based on the YOLO detector.*” 2018 International Joint Conference on Neural Networks (IJCNN). IEEE, 2018.
- [50] Litjens, Geert, et al. “*Evaluation of prostate segmentation algorithms for MRI: the PROMISE12 challenge.*” Medical image analysis 18.2 (2014): 359-373.
- [51] Pan, Huitong, et al. “*Prostate Segmentation from 3D MRI Using a Two-Stage Model and Variable-Input Based Uncertainty Measure.*” arXiv preprint arXiv:1903.02500 (2019).
- [52] D. M. W. POWERS, “*Evaluation: From Precision, Recall and F-Measure To Roc, Informedness, Markedness & Correlation,*” J. Mach. Learn. Technol., 2011.
- [53] Henderson, Paul, and Vittorio Ferrari. “*End-to-end training of object class detectors for mean average derivative precision.*” Asian Conference on Computer Vision. Springer, Cham, 2016.
- [54] Davis, Jesse, and Mark Goadrich. “*The relationship between Precision-Recall and ROC curves.*” Proceedings of the 23rd international conference on Machine learning. ACM, 2006.
- [55] Everingham, Mark, et al. “*The pascal visual object classes (voc) challenge.*” International journal of computer vision 88.2 (2010): 303-338.
- [56] Rezatofighi, Hamid, et al. “*Generalized intersection over union: A metric and a loss for bounding box regression.*” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019.

- [57] Hosang, Jan, Rodrigo Benenson, and Bernt Schiele. "*Learning non-maximum suppression.*" Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.
- [58] Ren, Shaoqing, et al. "*Faster R-CNN: Towards real-time object detection with region proposal networks.*" Advances in neural information processing systems. 2015.
- [59] Lin, Tsung-Yi, et al. "*Microsoft coco: Common objects in context.*" European conference on computer vision. Springer, Cham, 2014.
- [60] Russakovsky, Olga, et al. "*Imagenet large scale visual recognition challenge.*" International journal of computer vision 115.3 (2015): 211-252.