Detection and Recognition For Multiple Flames Using Deep Learning

Chen Xin

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

2018

School of Engineering, Computer and Mathematical Sciences

Abstract

Object recognition is one of the critical fields of intelligent surveillance which can be applied to tackle multiple security issues. The recognition of a flame is an extended research direction which has significant practical value for multiple applications.

The main contribution of this thesis, with the starting point of recognizing flames, is to introduce how Convolution Neural Networks (CNNs) deal with surveillance images. To achieve this, we constructed different neural network models, e.g., Single Shot MultiBox Detector (SSD) and You Only Look Once (YOLO). We enhanced the capacity of transformation of CNNs in part of convolution and RoI pooling.

In this thesis, we mainly present assorted methods in deep leaning to achieve detection and identification of various flames. First, we introduce the fundamental theories of object detection and recognition using deep learning. After that, we detail the algorithms of pattern classification and compare those algorithms for image feature extraction. Based on the theory of neural networks, we also depict these learning methods and analyze the accuracy of experimental results.

The focus of this thesis is on flame recognition using deep learning. For achieving this goal, a significant amount of measured data was used as the input data to ensure the validation of our experiments. We employ TensorFlow as the development environment with an NVIDIA GPU to train the image dataset and construct a detection model for the test dataset. Ultimately, experimental results show that the verification accuracy of our proposed algorithm is up to 98.6%.

Keywords—Flame Classification, Flame Recognition, Deep Learning, Convolutional Neural Networks (CNN), Tensorflow, Single Shot MultiBox Detector (SSD), You Only Look Once (YOLO), R-FCN

Table of Contents

Abstract I
Table of Contents II
List of Figures IV
List of Tables V
Attestation of Authorship VI
Acknowledgment VII
Chapter 1 Introduction 1
1.1 Background and Motivation 2
1.2 Research Questions 4
1.3 Contributions of This Thesis 5
1.4 Objectives of This Thesis 5
1.5 Structure of This Thesis 6
Chapter 2 Literature Review 7
2.1 Introduction 8
2.2 Flame Detection and Recognition 9
2.3 Artificial Neural Networks 13
2.4 Deep Learning 17
2.5 Convolution Neural Network (CNN) 19
2.5.1 Input layer 19
2.5.2 Convolution Layer 20
2.5.3 Sub-Sampling Layer 21
2.5.4 Output Layer 22
2.5.5 Forward Propagation 23
2.6 Applications of CNN 24
2.6.1 LeNet 24
2.6.2 VGG 25
2.6.3 GoogleNet 26
2.6.4 ResNet 27
2.6.5 DenseNet 29
2.7 Single Shot MultiBox Detector (SSD) 30
2.7.1 Default Boxes 31
2.7.2 Loss Function 31
2.8 YOLO and R-FCN 32
Chapter 3 Methodology 35
3.1 Research Designing 36
3.2 Data Collection and Manually Label Marking 37
3.3 Data Augmentation 39
3.3.1 Resizing 40
3.3.2 Random Horizontal Flipping 40
3.3.3 Random Cropping 41
3.3.4 Color Augmentation 42

3.4 Model Details 43
Chapter 4 Results 51
4.1 Experimental Parameters and Environment 52
4.2 Bounding Box Setting 53
4.3 Multiple Flame Recognition 55
4.4 Limitations of the Experiment 63
Chapter 5 Analysis and Discussions 64
5.1 MFD Models Analysis 65
5.2 Model Comparisons 71
5.2.1 Structures of Three Methods 71
5.2.2 Training and Loss Function 74
5.2.3 Activate Function 80
5.2.4 Results and Discussion 81
Chapter 6 Conclusion and Future Work 83
6.1 Conclusion 84
6.2 Future Work 85
References 86

List of Figures

Figure 2.1 Sampling process 22
Figure 2.2 The Structure of Single Shot MultiBox Detector (SSD) 30
Figure 3.1 The Flowchart of the Steps of Flame Detection and Recognition 36
Figure 3.2 Examples of Flame Categories 37
Figure 3.3 Manual Marking Label Diagram 38
Figure 3.4 Examples of Flame Resizing 40
Figure 3.5 Examples of Random Horizontal Flipping 41
Figure 3.6 Example of Random Cropping 41
Figure 3.7 Example of Color Augmentation 43
Figure 3.8 Overall Structure of Whole Network 45
Figure 3.9 Graph of Three Activation Functions 47
Figure 3.10 Overall Structure of Convolution Neural Network 47
Figure 3.11 The Three Loss Function 50
Figure 4.1 Width and High Coding Function 54
Figure 4.2 The Result of Candle Fire 55
Figure 4.3 The Result of Gas Fire 56
Figure 4.4 The Result of Alcohol Fireplace Fire 56
Figure 4.5 The Result of Fireplace Fire 57
Figure 4.6 The Result of Blowtorch Fire 57
Figure 4.7 The Result of Gas Tank Fire 58
Figure 4.8 The Result of Gas Fire and Candle Fire Recognition Together 59
Figure 4.9 Graph of Accuracy 60
Figure 4.10 Graph of Accuracy of Classification 60
Figure 4.11 Graph of Accuracy of Location 61
Figure 4.12 Graph of Loss Function 61
Figure 5.1 Training Curves of MFD_1 67
Figure 5.2 Training Curve of MFD_2 68
Figure 5.3 Training Curve of MFD_3 69
Figure 5.4 Training Curve of MFD_4 70
Figure 5.5 Accuracy of Flame Recognition From MFD_1 79
Figure 5.6 Accuracy of Flame Recognition From YOLO 79
Figure 5.7 Accuracy of Flame Recognition From R-FCN 80

List of Tables

Table 3.1 Details of MFD's CNN 47
Table 4.1 Training Interface Parameters 52
Table 4.2 Four Kinds of Box Coding 53
Table 4.3 Distribution of the Number of Different Types of Objects 62
Table 4.4 Different Types of Objects IoU Distribution 62
Table 4.5 Correct Rate Distribution of Different Types of Objects 63
Table 5.1 Correct Rate Comparison 65
Table 5.2 Details of MFD Convolutional Layers 72
Table 5.3 Details of YOLO Convolutional Layers 73
Table 5.4 Details of R-FCN Convolutional Layers 74
Table 5.5 Accuracy of Flame Types from Three Models 82

Attestation of Authorship

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

Signature:  Date: 15 July 2018

Acknowledgment

This thesis was completed as the part of the Master of Computer and Information Sciences (MCIS) course at the School of Computer and Mathematical Sciences (SCMS) in the Faculty of Design and Creative Technologies (DCT) at the Auckland University of Technology (AUT) in New Zealand. I would like to sincerely thank my parents for the financial support they provided during my entire time of academic study in Auckland.

I intend to express my most gratitude to my primary supervisor Dr Wei Qi Yan. This work will not be accomplished without his instructions and encouragement. Dr Yan always guided me to pursue the knowledge of real practice. I also hope to thank my secondary supervisor Dr Minh Nguyen for his invaluable advice and help. Also, I hope to thank school administrators for their support and guidance through the MCIS in the past years.

Chen Xin

Auckland, New Zealand

July 2018

Chapter 1
Introduction

The content of this chapter is to introduce the background and motivation of this thesis. The rationale underlying the research includes in-depth analysis of the issues. Flame detection and recognition in a visual surveillance system are proposed to be introduced in this chapter. After referral the background, this chapter also covers the details of research questions about the in-depth wide-ranging understanding of the relevant literature. Finally, the structure overview of this thesis is illustrated in the final section.

1.1 Background and Motivation

At present, computer vision has become a very popular subject for acquiring visual information. Flame identification has become one of the most important topics in the field of object recognition. It has good prospects for artificial intelligence, system monitoring and so on. Nowadays, most public places and buildings use conventional ways to present fire alarms and fire monitoring, such as smoke detectors, temperature detectors, and photodetectors. Fire smoke, temperature and light characteristics are applied to detect fire flames (Fereres & Lautenberger, 2011). However, when those detectors set indoor or outdoor which has some complex surroundings, sensor signals from those conventional detectors would have lower accuracy.

With the development of computer and image processing technology, it has been proposed to use image processing technology for flame detection. We expect a computer could detect a variety of fire flames automatically and classify the dangerous flames from the input visual data. This requires a combination of computer vision and deep learning, i.e., deep neural networks. Our human brain recognizes an object based on accumulated knowledge from our ordinary life. We usually name it as experience. When the object encountered is relatively complicated for decision making, we usualy guesses replying on our learned knowledge or experience in avoiding mistakes.

For a computer, the input data is classifed as “yes (1)” or “no (0)”. The experimental results of traditional machine learning are mostly based on the standard image dataset. The influences of lighting, shielding, and flame colors are much minor than the ones in real life. Moreover, the shape of a flame is in a continuously changing way, and the images of the same flame at different direction of viewing are also different. Therefore, the accuracy of recognition results is much lower (Albu 2009).

In this thesis, we consider the flame detection as a sub-task of object recognition. We divides the task of flame recognition into two parts: classification of a flame and location of a flame. We firstly extract high-level features from an input image through a feature

extractor, and then use the features which extracted previously as the input of the two sub-networks: classified network and identified network, respectively. The approach improves the recognition rate and accuracy (Garcia, C., & Delakis, M, 2004).

Since LeCun et al. designed and trained a convolutional neural network by using an error gradient-based algorithm (LeCun, Bottou, Bengio & Haffner, 1998). In the field of pattern recognition, outstanding performance has been achieved. Also, the convolution network has been given and proved (Chen, Wang & Jin, 2016). At present, Deep learning is successfully applied to various aspects such as document analysis (Simard, Steinkraus, & Platt, 2003), face detection (Tivive & Bouzerdoum,2003), voice detection (Abdel-Hamid, Mohamed, Jiang, Deng, Penn & Yu 2014), license plate recognition (Chen, Han, Wang, Jeng & Fan, 2006), digital handwriting recognition (Lauer, Suen & Bloch, 2007), human motion recognition in videos (Ji, Xu, Yang& Yu,2013), and human face detection (Sun, Wang, & Tang, 2013).

Convolutional neural networks have the following advantages: (a) The input image and the network topology can match well, which can avoid the complex pre-processing of the image and input the original image directly. (b) Feature extraction and pattern classification perform simultaneously, and a convolution layer is composed of a plurality of feature maps, which can be optimized through learning training. (c) The connection between neurons is not fully connected, and specific neurons in the same layer share the same connection weights. This network structure with incomplete connection and weight sharing reduces the complexity of the network model and reduces the number of weights.

In this thesis, we adopt CNN structure model to create a new neural network so as to recognize flames which let the computer learn itself to find the fire features expressed in a deep level. Also, this method can avoid the complexity and aimlessness which the traditional method has. Our algorithm for flame detection implemented in three primary methods: color-based model, a combination of RGB and HIS model and CNN. We train the image dataset with the GPU in the laptop under the Tensorflow environment. We selected NVIDIA GTX 980M GPU for our experiment. Training with a GPU is more

efficient way than that in a CPU. GPU in the vector calculation is better than the CPU, which constitutes the most in-depth study of the computation. By extracting the frames from a video which we recorded, we collect as many pictures as possible. Also, we manually marked the right regions to tell the computer where the flame pixels are.

1.2 Research Questions

As video surveillance systems are required to achieve automatic flame detection, the systems are designed for working under both indoor and outdoor conditions. However, in a real situation, the condition around an object could be the factor which affects the accuracy of detection. For example, lighting changes, shadows, and weather will cause noisy data. Due to those noisy data, the degradation of image quality brings more difficultities of object detection and recognition. Additionally, with the unfixed shape and the diversity of different sorts of flames, it is difficult to recognize a flame in the monitoring area.

As we mentioned, the goal of this thesis is to achieve flame detection and recognize the categories of multiple flames. Analysing the methods and techniques helps us understand the knowledge needed in image processing using deep Learning. Thus, this thesis focuses on creating a flame detection using deep learning that has the capability of flame detection and recognition. Hence, the research questions of the thesis are:

Question 1. *What methods or techniques can be implemented in flames detection and recognition?*

We use convolution neural network (CNN) based on deep learning to implement our system. With the model obtained after CNN operations, the program automatically analyzes and recognizes a flame and completes the positioning and recognizing the flame in a short time.

Question 2. *How to extract flame features from the videos using deep learning?*

Within flame detection, the accuracy of flame recognition plays a decisive role in fire

classification. Before deep learning, we need to label the position of all flames in the training dataset to tell CNN what the ground of truth is. After the training, manual marking will no longer be required.

Question 3. *How to carry out the task of identifying different flames using deep learning?*

We split the tasks of flame detection into two parts: classification of flames and detection of flames. We extract the feature maps from the input images. The extracted feature maps are then used as the input data for two sub-networks: detection and recognition.

1.3 Contributions of This Thesis

This section is to clarify the contributions of this thesis. In this thesis, flame recognition using deep learning is proposed and designed. Overall, the primary goal of this thesis is intended to make the following contributions:

Firstly, flame detection and recognition can adopt deep learning. Flame recognition could be applied to various platforms such as monitoring camera and mobile applications. When a flame is detected, the flame category or class will be recognized.

The second is that we can verify the flame is dangerous or not. We can integrate flame recognition with harzards analysis so as to provide much intelligent services.

1.4 Objectives of This Thesis

Frist of all, this thesis introduces flame recongition using deep learning which could detect and recognize various classes of flames. Deep learning is a new field in artificial intelligence. The principles of deep learning are demonstrated and evaluated in this thesis.

Secondary, the overall objective of this research is to develop an alogrithem of flame detection and recognition using deep learning, the main convolution neural network models we will use are: (a) Single Shot MultiBox Detector (SSD) and (b) Deconvolution

Single Shot Detection (DSSD).

Finally, in order to achieve the recognition of flames in surveillance, we will try various algorithms for our research project. Therefore, we compare experiment results from those algorithms, it is necessary to find the best one that is suitable for object detection and object classification.

1.5 Structure of This Thesis

The structure of this thesis as follows:

Chapter 2 discusses literature review of the previous work related to flame detection and recongition using deep learning. In this chapter, the fundamental knowledge of deep learning and convolution neural network (CNN) is introduced. Also, algorithms of flame detection and recognition will be introduced and compared. Additionally, the different fire characteristics with burning materials will be discussed.

In Chapter 3, multiple research methodologies will be presented, research questions of this thesis also are included. The potential problems and corresponding solutions are presented. Additionally, experiment design and implementations of the research will be presented.

In Chapter 4, we will discuss the details about the results of our experiments assisting with figures and tables. The limitations of this project are addressed as well.

Chapter 5 contains the discussion and analysis of our experimental results. By the end of this thesis, the conclusion and future work will be presented in Chapter 6.

Chapter 2
Literature Review

The primary purpose of Chapter 2 is to introduce and explain deep learning and different neural networks. We will detail our understandings of the literature. Firstly, we review the articles about classical methods that help people detect the hazardous flame. Then, we present the necessary information on Convolutional Neural Network (CNN) and its applications in real world. At last, the algorithms we will use are also introduced in this chapter.

2.1 Introduction

Conventional flame detection as one of the most important modules in security system are working for the security of buildings and public areas. Flame detection can give an early alert at the start of firing. Most of flame detection consists of built-in sensors to discover the dangers which highly depend on sensors. Generally, sensors should be installed densely to monitor a secure primise in high precision. Due to the limitations of sensors, the traditional flame detection is unrealistic to cover a large area outdoor.

With the speedy development of image and video processing techniques and artificial intelligence (AI), it has been the trend to replace the flame detection by using numberious sensors with computer vision. Flame detection and recongition which are based on image and video processing have three main phases (Chen, Wu,& Chiou, 2004): classification of the fire pixels (Toreyin, Dedeoglu & Cetin, 2005); segmentation of moving objects (Toreyin, Dedeoglu & Cetin, 2006); the last one is to anayze the region of interest (Celik, Demirel & Ozkaramanli, 2006).

Compared to the traditional monitoring technology and the image-based fire detection and recongition, image processing has many advantages which outperforms those conventional sensors, but image process is highly dependent on the features extracted by a computer algorithm. In general, the reasonable feature extraction will leave to higher recognition rate. However, the excellent quality of extraction always requires well background knowledge and pretty rich experience. For the goal, classification and detection for flames, the irregular shape and changing characteristics of the flames make it difficult to extract features reasonably and require extensive trial and error. This will increase the complexity of our algorithms, reduce the fire recognition accuracy and robustness. In this thesis, we will use image and video processing based on deep learning for flame detection and recongition. After several years' development of the convolution neural networks, deep learning has achieved marvlous achivements, such as handwritten character recognition (LeCun, Bottou, Bengio & Haffner, 1998), pedestrian detection (Szarvas, Yoshizawa, Yamamoto & Ogata,2005), face recognition (Lawrence, Giles, Tsoi

& Back,1997), etc.

2.2 Flame Detection and Recognition

The performance of flame detection primely depends on the flame classifiers, which generate seed region for classification. Thus, it requires that the classification needs to have a high accuracy and a low false alarm rate. Most of the video-based flame detection aims to provide an early fire alarming.

Traditional ultraviolet and infrared flame detection and classification have a number of disadvantages. When visual data having interference factors appears in the background, an alarm will be trigerred. At the same time, location, size and growth rate of the flame will also be wrong. Based on these questions, Healey and his team have come up with a color camera that can capture video and automatically detect the presence or absence of fire flames by analyzing the collected data. This algorithm is based on the spectral, spatial and temporal properties of fire flames (Healey, Slater, Lin, Drda & Goedeke, 1993).

In 1994, the use of image processing technology was proposed for flame detection in road tunnels. The image processing methods are firstly reviewed by using infrared or color cameras, the disadvantages of these traditional sensor detection methods are pointed out. A fire detection method was proposed that is different from the traditional infrared detectors and smoke detectors. The video images in the event of a fire are compared at the grayscale levels to reach the purpose of detecting the fire (Noda & Ueda, 1994).

Yamagishi's main idea is to capture the flickering flame with a color CCD camera and then calculate the flame profile fluctuation in the following time. The method can eliminate the interference of irrelevant information by using color information and data mining technology so as to satisfy the robustness of the inspection (Yamagishi & Yamaguchi, 1999). Foo published an article in the year 2000 which was about to use visual images for detecting fires in engine rooms of aircraft. The spectrum machine vision enables to detect hydrocarbon fires. Since luminance, color, spectrum, and flicker frequencies are unique features of a flame, these characteristics are often used in flame

detection to distinguish the fire from the background.

The fire alarm needs to have the characteristics of classified fire and rapid reaction to prevent the growth of fire spreading. Based on a group of machine vision methods for early fire detection, the appearance of flames is determined by calculating whether the pixels of an image in two adjacent video fames exceed the preset values. Specifically, the difference between histograms is also analyzed, the video frames came up with a set of statistics to reach the conclusion based on the performance of real-time data (Foo, 2000).

Tunnels are also one of the most common places that needs to take fires into special consideration. Due to the unique structure of tunnels, smoke and toxic gases after the fire often has serious effects (Koga, Inobe, Namai& Kaneko, 1997). Although designers have added safe passageways and fire suppression systems at design time, prevention of fire has always been the topest concern. Through infrared radiation sensor detection devices, flame was detected from multiple angles. When the infrared radiation value sensed by the detector exceeds a fixed threshold, the system will automatically triger an alarm, thereby eliminating the potential fire risk (Cigada, Ruggieri & Zappa, 2005).

The methods based on spectroscopy or particle sensors have the disadvantage of not being able to move the camera or detect dynamic scenes. Spectroscopy cannot be applied to visual data with the properties of time series, whereas particle sensors are far simpler than general fire detection systems (Phillips, Shah& Lobo, 2000). By creating a Gaussian smooth color histogram, it is possible to calculate the color change in the video sequence and then use the pixel motion at different time to locate the flame.

Those methods of fire prevention can be applied to small areas of space, such as inside a tunnel, but not necessarily outside a large area. Outdoor with a more complicated background is a video-based monitoring system that cannot predict where and when a fire will occur. A fire area was designed to be as accurate as possible by using selected color and framing differences (Lei, Zhang & Wang, 2012). Their results show that the method is suitable for indoor fire detection as well as outdoor fire detection. In 2003, Chen proposed an intelligent real-time fire detection method through two-stage strategy (Chen,

Kao & Chang, 2003). The first phase of color image processing can determine the existence of fire. In the second phase, it is to determine whether the number of detected flames increases over time, whether the number of fires has increased over time. When a preset threshold is exceeded, the system triggers an alarm. It is found that this method is suitable for fire prevention in various public places.

During the early years, people have been working on fire alarm devices. In 1847, a professor at the University of Maine USA, developed the world's first transmitter for urban fire alarm (Gutmacher, Hoefer & Wöllenstein, 2012). In 1890, Britain successfully developed the first sensible temperature fire detector, since then, our mankind has started a new era of automatic fire detection and alarm technology. More than one hundred and sixty years, in the tide of world science and rapid technology development, fire detection technology also got rapid development, fire detector was designed based on patterns of various flames.

We see from the previous research results and practical applications, there exist a few methods which process the image or video of flames directly. Krull proposed a kind of low-cost CCD cameras to check for fires in the cargo bay of the long-distance airplane (Krüll, Willms, Zakrzewski, Sadok, Shirer & Zeliff, 2006). This method conducts statistical features in grayscale video frames, which includes the mean pixel intensity, standard deviation and second-order moments along with other non-image process techniques. This system also provides a capability of vision which could help the pilot to check the security of the airplane, presence or absence of fire.

Chen et al. (Chen, Wu & Chiou, 2004), in 2004 used raw RGB model to develop a set of functions to classify the flame images and videos. Without using the RGB model, Toreyin (Çelik, Özkaramanlı& Demirel, 2007) combined Gaussians algorithm with RGB space to achieve the detection function. In their recent papers, they referred to the fire pixel classification method with Markov Random Field model (Celik & Demirel, 2009) to capture the motion information of flames. In 2006, Marbach et al. (Marbach, Loepfe & Brupbacher, 2006) used YUV color model as the representation of the data from flame

videos. Y means the derivative of luminance parameter, U and V are the chrominance parameters which check the fire in the candidate region or not. In addition, Marbach et al. comprised of motion in the work as well. Homg and his team (Horng, Peng & Chen, 2005) used HSI color model to segment fire area for a brighter and darker environment. They reported 96.94% detection rate, together with results including false positives and false negatives for the algorithms.

A generic color model based on normalized RGB values was proposed (Celik, Demirel, Ozkaramanli, & Uyguroglu, 2007). The reason why they used the normalized RGB is that the illumination changes in the video have severe effects on the result. The system is obtained from the statistical analysis of three planes, which are an R-G plan, an R-B plane, and a G-B plane. If a pixel is checked to exist in those three planes, the pixel will be classified as a fire pixel.

Flame detection and recognition as an active research area attracted more and more attention in the field of intelligent surveillance due to the increase of fire security problems in metropolis. Particularly, soaring requirement of active monitoring in skyscrapers, such as the Burj Khalifa Dubai, the Petronas Twin Towers in Malaysia and the Empire State Building in Manhattan. If the fire disaster happens in these buildings, it would be too late for firefighters to rescue. Hence, the research on image-based fire detection technology is relatively early, and some have been successfully applied to practical projects.

CappelliniV et al. (Cappellini, Mattii& Mecocci, 1989) first proposed the flame color model of RGB in 1989. The fire pixel points are obtained based on the saturation of the red color component and the flame dynamics. The alarm would appear when the number of flame pixels is above a certain threshold.

Healey G et al. (Healey, Slater, Lin, Drda, & Goedeke, 1993) proposed in 1993 using the spectral, spatial and temproal attributes of flames to obtain the flame area. Compared to ultraviolet and infrared images, the method could reduce the false alarm rate and get more information, such as flame position, flame size, growth rate of a flame.

Neubauer A et al. (Luo, Su & Tsai, 2002) proposed a simulation technique for a fire random signal set at a standard level. Based on his theory, it was the first time that the genetic algorithm was used for fire detection. Phillips et al. (Miao & Wang, 2013) created RGB models based on color and disordered measurement techniques. The model was used to obtain the pixels where the region has a flame. It is mainly based on the density and saturation of the red color component. At the same time, the acquired flame pixels are also corrected by using the growth of dynamic features and smoke. The growth rate of flames is continuously checked and the condition is met to give an alarm.

Li Jin et al. (Li, Fong, Chow, Wong, Lu & Xu, 2004) conducted an in-depth study on the frequency of flame scintillation and used the first-order distance of the flame region as a criterion, which is less affected by the external environment and the combustion materials.

Toreyin et al. (Toreyin & Cetin, 2007) proposed to apply the wavelet domain for fire detection in 2004. In practice, the flicker frequency of fire flame is not fixed, which is reflected in pixel points. This change can be regarded as a random process. Therefore, the hidden Markov model based on flicker frequency is more effective than the traditional method to detect the flame pixel.

Celik et al. (Celik, Demirel & Ozkaramanli, 2006) proposed a method of integrating statistical color information with foreground information to detect flames. Lu, et al. proposed (Lu, Peng, Horng & Peng, 2006) in 2006 for the linear decision technology (linear discriminate technique) and logistic regression method (logistic regression), logistic regression method was used to split the flame area and construct the flame characteristic model (Andrews, Loftsgaarden & Bradshaw 2003).

2.3 Artificial Neural Networks

Artificial Neural Networks (ANN) (Hopfield, 1988) is also referred to Neural Networks (NN) (Krizhevsky, Sutskever, & Hinton, 2012). The artificial neural network is the abstraction and simulation of fundamental characteristics of human brain or biological

neural networks, the biological brain has inspired its research to a considerable extention. It is composed of a series of simple artificial neurons densely connected to each other, in which each neuron is composed of three parts: input, artificial neural cell body, and output. Each neuron has multiple real-valued inputs and produces a real-valued output.

Artificial neural cells integrate these input signals and perform threshold processing. If the integrated stimulus exceeds a certain threshold, the neurons are activated into an active state; otherwise, the neurons are inactive. Just as the human brain can achieve unremitting learning and progress by continuously adjusting the connections between neurons, the ANN can also adjust the weights on the input of connections to make the network more adaptable for the training set.

In ANN, the feature vector of a sample is considered as the input of the ANN, and the categories of the training samples (category in the classification problem) is the output of the network. In the initial stage, the network weight is initialized to a random state. When a training sample is an input to the network, the difference between the resulting network output and the training sample target output is called the training error. Next, ANN will adjust the weight value according to some mechanisms, so that the training error decreases gradually. With this training and adjustment process, the actual output of the network for training samples will be closer to our goal expectations.

- **Perceptron**

Perceptron is the most basic unit of an artificial neural network. It takes *n* real numbers as the input parameters, like (x_1, x_2, x_3, x_4, x_5), each input component corresponds to a weight w_i with an offset w_0 and calculates the linear combination of these inputs. If the result comes to be higher than 0, the output is 1. Otherwise, the output will be -1. It can be expressed as:

$$O(x_1, \dots, x_n) = \begin{cases} 1 \ if \ w_0 + w_1 x_1 + \cdots + w_n x_n > 0 \\ -1 \ Otherwise \end{cases} \quad (2.1)$$

- **Error Criteria**

Error criteria is a standard for network design. In the optimization design of various networks to achieve the best numerical approximation, it is usually necessary to determine a network function that satisfies a given error criterion so as to achieve the purpose of optimizing the design. To obtain the optimal weight vector $\vec{\boldsymbol{w}}$ in the training process, a metric standard must be specified to measure the training error of the neural network related to the training sample under this weight vector $\vec{\boldsymbol{w}}$. A common metric is the square error criterion.

$$\mathrm{E}(\vec{w}) = \frac{1}{2}\sum_{d\in D}(t_d - o_d)^2 \tag{2.2}$$

where $\boldsymbol{t_d}$ is the actual output of the nervous system for the training sample $\boldsymbol{d}$, $\boldsymbol{o_d}$ is the target output of $\boldsymbol{d}$, **D** is the set of the entire training sample. Because $\boldsymbol{o_d}$ is fixed and $\boldsymbol{t_d}$ depends on $\vec{\boldsymbol{w}}$, so that *E* is a function of $\vec{\boldsymbol{w}}$. This function represents a parabolic or hyperparabolic surface in space which is the error surface. The access to the global minimum of the error surface is the weight vector that we want to obtain.

- **Incremental Gradient Descent**

Gradient descent is the first-order optimization algorithm, also commonly referred to as steepest descent. Using the gradient descent method to find a local minimum of a function, it must iteratively search for a specified step distance point in the opposite direction of the gradient from the current point on the function. So the gradient descent method can help us to find the minimum value for the function. For the optimal solution to the *n*-dimensional problem, the gradient descent is one of the most commonly used methods. In order to quickly reach the bottom of the error surface, it should be calculated along the steepest descending direction. By calculating the gradient which is the partial derivative of E concerning each component of $\vec{\boldsymbol{w}}$, we can get the direction with the largest directional derivative:

$$\nabla\mathrm{E}(\vec{w}) = \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \dots, \frac{\partial E}{\partial w_n}\right] \tag{2.3}$$

The gradient descent training rules are:

$$\vec{w} \leftarrow \vec{w} + \Delta\vec{w}, where\ \Delta\vec{w} = -\eta\nabla \mathrm{E}(\vec{w}) \tag{2.4}$$

where $\boldsymbol{\eta > 0}$ is a normal number, namely, the learning rate, which specifies the step length of gradient descent search.

However, the gradient descent method has many problems in practice. Sometimes the convergence process is prolonged, even requiring thousands of iterations. This is because there are often many local minima on the error surface so that it will fall into a local optimum, lead to the inability and guarantee that a global minimum is found. However, the gradient descent method has many problems in practice. Sometimes, the convergence process is prolonged, even requiring thousands of iterations. An incremental gradient descent method is designed to alleviate these problems, also called stochastic gradient descent (SGD) is proposed. The standard gradient descent method computes the weight update after summing the squared errors of all training samples. The stochastic gradient computes the error increment of each sample and updates the weights to obtain an approximate gradient descent. This avoids falling into local minimal value.

- **Multi-Layer ANN**

A single neuron can only get a linear hyperplane after training, but the classification ability of the linear decision-making plane is limited, and it is incapable of many complex classification tasks. According to the fact that nerve cells in the brain are interconnected, we can also organize individual neurons together in this way to form multiple artificial neural networks. The hierarchical feedforward network is the most widely used connection mode. The output of each layer of neurons is fed forward to their next layer. Finally, the output of the entire network is obtained.

The hierarchical neural network can represent a variety of nonlinear surfaces and obtain nonlinear decision areas. In a hierarchical neural network, there is generally at least one input layer and one output layer, one or more hidden layers, so there are at least three layers. Neurons are a full connection between adjacent layers, each of the input layer neuron is connected to a unit in the hidden layer and the output of the hidden layer of

each neuron is connected to each of the output layer neurons.

- **Sigmoid Unit**

For the goal of data to apply the stochastic gradient descent algorithm in the process of training the network, the function in this unit should be a differentiable function. The Sigmoid unit is an ideal unit to satisfy these two conditions. It uses the important sigmoid function as the activation function. First, we calculate the linear combination of *n* inputs and the bias $\boldsymbol{w_0}$, and input the result to the sigmoid function. Finally, the output of the gmoid function is taken as the output $\boldsymbol{O}$ of this neuron, so as to realize the nonlinear mapping from input to output

2.4 Deep Learning

Since 2006, deep learning has become a new field of machine learning research. In the field of machine learning, deep learning as a method to model the modes like audio, image, text and so on. Deep learning aims to make machine learning closer to its original goal – artificial intelligence.

With the advent of Deep Learning, many researchers have devoted themselves to the study of principles and applications of this new technique, which are mainly reflected in the upsurges of significant conferences, research groups of universities, and enterprise applications. Hinton research group from the University of Toronto used the deep convolutional neural network to classify 1.3 million images to recognize 1000 different classes (Krizhevsky, Sutskever & Hinton,2012). Ng research team applied stochastic gradient descent to optimize for more complex structures, such as limited memory BFGS and conjugate gradient with line search, which can significantly simplify and accelerate the pre-training depth algorithm process (Le, Ngiam, Coates, Lahiri, Prochnow & Ng, 2011). Bengio research group from Montreal University employed deep learning to model statistical languages and learn the joint probability function of word sequences in a language. They suggested that the curse dimension can be informed by semantically

adjacent sentences of the model index by learning distributed representation words for each sentence training (Bengio, Ducharme, Vincent, & Jauvin, 2003). Glorot proposed a method that uses a multiscale convolutional network trained by using original pixels to extract dense feature vectors and then encodes multiple-sized regions centered on each pixel. (Glorot, Bordes & Bengio,2011).

The word ‘deep’ refers to the number of levels composed of nonlinear operations in the learned functions. In 1969, Minsky and Papert pointed out that the structure of single layer perceptron cannot be applied to implement the function of XOR, which means the perception cannot solve the linear inseparable problem (Minsky& Papert, 1969). The multilayer perceptron, i.e., deep structure can resolve the problem of linear inseparability. The deep structure combines or transforms low-level features to a higher level and learn the characteristics of the hierarchical structure. This kind of peculiar structure allows the system to automatically learning in multilevel abstraction, and the deep structure can fit complex functions. Due to the increasing importance of machine learning methods that learning data hidden in the characteristics of high capacity with the expanding of the scale of the data, more and more researchers pay attention to the basic model of deep learning as well as the application in several fields. Bengio’s theoretical results show that if high-level abstract features of data could be learned such as visual classification and language recognition, the training models often require a deeper architecture. The nonlinear operation in the deep structure can be well used as the development module of the building module which is the unsupervised learning of the single-layer model. (Bengio, 2009).

As early as 1974, Werbos proposed the Back Propagation(BP) algorithm, which solved the problem of linear inseparability of simple neural networks extended to the complex neural network model (Sapna, Tamilarasi & Kumar, 2012). Nevertheless, if we added layers in the neural network model, the effect of parameter optimization cannot be transferred to the front layer, which means that it is easy to overfit and comfortable for the model to fall into the optimal local solution.

Another rich conceptual framework is Restricted Boltzmann Machines (RBM) (Fischer & Igel, 2012), which was developed from Boltzmann Machines (BM) in 1985 (Ackley, Hinton & Sejnowski, 1987). It is a random neural network example of statistical mechanics with strong unsupervised learning ability to learn complicated rules in data. Smolensky introduced a Restricted Boltzmann Machine (RBM) and solved the problem from BM that cannot accurately calculate the distribution. Smolensky limited the original inter-layer connections of the BM. The limitation makes the different nodes in the same layer being independent on each other. The result is more natural to find its probability distribution function (Yu & Deng, 2011).

In 2006, Deep Belief network (DBN) (Sarikaya, Hinton & Deoras, 2014) was proposed, which can be regarded as a stack of Restricted Boltzmann Machines (RBM). The deep belief network is not much different from the traditional multilayer perceptron. However, it requires an unsupervised learning before supervised learning, and then takes the learned parameters as the initial value of supervised learning.

2.5 Convolution Neural Network (CNN)

In 1989, Yan Lecun et al. (LeCun, Boser & Jackel, 1989) proposed a Convolution Neural Network (CNN) structure that can be used to successfully utilize the BP to train neural networks based on previous research works. By providing constraints from the task area, generalization ability of the learning network can be significantly enhanced. How to integrate these constraints is demonstrated into the back-propagation network through the model. For the application of backpropagation networks, Yan Lecun and his team used this algorithm for handwritting recognition. Before identifying the number handwritten by a person, the minimal data pre-processing is required. However, the architecture of this network is highly constrained, it is necessary to design a network specified for the task. This technology is now applied to the U.S. Postal Service (LeCun, Boser, Denker, Henderson, Howard, Hubbard & Jackel, 1990).

2.5.1 Input layer

The input layer of the convolutional neural network can directly receive two-dimensional visual modes (e.g., two-dimensional images). The input layer can automatically extract the features of original image data, the extraction in this course does not need to be manually involved which to select or design the suitable characteristics as an input of CNN. With the input layer of CNN, this process dramatically reduces the amount of manual pre-processing and is beneficial to the most effective visual features of current classification tasks.

2.5.2 Convolution Layer

The convolutional layer is a feature extraction layer, and each convolutional layer contains multiple convolution kernels. The convolution kernel is composed of multiple neurons which are a feature extraction of the input data of the previous layer. Each convolution kernel can extract a corresponding feature. The specific extraction feature from the kernel is determined by the weight of neurons in each convolution kernel. Compared with general feedforward networks, the convolution kernels in convolutional neural networks dramatically reduce network parameters.

In order to further reduce the network parameters, convolution neural networks adopt a method, namely convolution kernel sharing. When a convolution of a convolutional layer works at different regions of the input layer, the weight of the convolution kernel is not changed. That is, a convolution kernel is only used to extract the same feature in different positions in the previous layer. Like the algorithm showed below:

$$a_j^l = f\left(b_j^l + \sum_{i \in M_j^l} a_i^{l-1} * k_{ij}^l\right) \tag{2.5}$$

The value a_j^l is a activation for the *j-th* feature mapping in the *l-th* convolution layer. The function *f* is a nonlinear activation function, genreally they can be Tanh Function and Sigmoid Function. $\boldsymbol{b_j^l}$ in this function refers to the bias of the unit $\boldsymbol{j}$ in layer l. The $\boldsymbol{M_j^l}$ is a index vector of the feature map *l* in layer (*l-1*)-*th*, and the feature map $\boldsymbol{j}$ in layer $\boldsymbol{l}$ needs to be accumulated. The signal '*****' refers to a two-dimensional convolution operation

and $\boldsymbol{k_{ij}^{l}}$ is the convolution kernel of the feature map $\boldsymbol{l}$ which works on the layer (*l-1*)-*th*, and the kernel can genereate the accumulated input part of the feature map *j* in the *l-th* layer. A convolutional layer usually consists of several feature maps, where $\boldsymbol{k_{ij}^{l}}$ is aslo known as the weight in nerural network, which has not the differenation in the same feature map. Based on the above explainnation, the convolution layers can reduce the number of free parameters.

2.5.3 Sub-Sampling Layer

The sampling layer is the feature mapping layer. Each sampling layer contains a kernel in the sampling, which consists of multiple sampling elements. The sampling kernel is only connected with the local sensing domain of the corresponding position of the previous layer network. Unlike neurons in the convolution kernel, the weight of each neuron in the sampled nucleus is fixed and does not change with the state of the network (Pinheiro & Collobert, 2014). Therefore, there is an additional layer behind each convolution layer to perform local averaging, namely, sub-sampling, to reduce the sensitivity of the output during translation and deformation.

$$a_j^l = down\left(a_j^{l-1}, \boldsymbol{N^l}\right) \tag{2.6}$$

For feature map *j* in a Sub-Sampled layer *l*, there is a function. The term $down$ refers to the function of the sub-samping process which based on the divisor $\boldsymbol{N^l}$. $\boldsymbol{N^l}$ is the window boundary size required for the Sub-Sampling in layer *l*, and then caculate the average value of the non-overlapping area in the window which in a $\boldsymbol{N^l \cdot N^l}$ size.

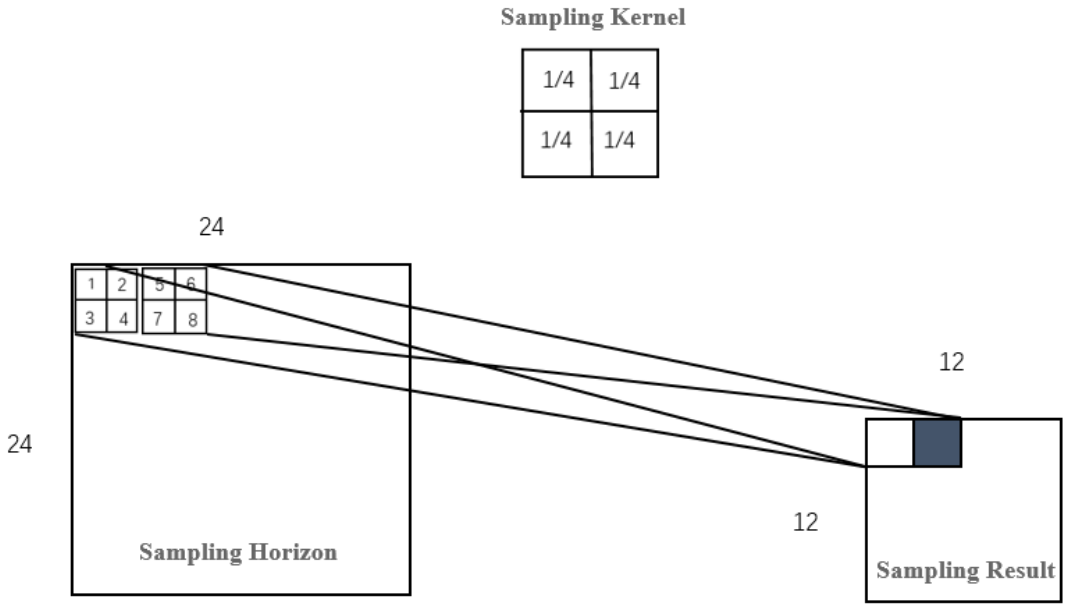


Figure 2.1 Sampling Process

2.5.4 Output Layer

The output layer of the convolutional neural network is transmitted by full links, just like feedforward neural network. The last layer can be either the convolution layer or the sampling layer. The 2D matrices are recombined into a 1D vector, which is connected to the output layer in a fully-linked manner. Assuming that the output layer of the neuron is ***C***-dimensional, then the identification of ***C***-class can be identified. Then the output layer is the production representation of the connection feature map from the previous layer:

$$0utput = f(b^0 + w^0 \boldsymbol{f_v}) \tag{2.7}$$

The $\boldsymbol{b^0}$ is the vector quantity of bais; $\boldsymbol{w^0}$ refers to the matrics of weights; $\boldsymbol{f_v}$ refers to the vector quantity of features. The parameters for the model can form into $\{\boldsymbol{k_{ij}^l}, \boldsymbol{b_j^l}, \boldsymbol{b^0}, \boldsymbol{w^0}\}$. Convolution layer and subsampling layer usually alternate layer by layer, and the number of feature map is inversely proportional to the spatial resolution of the image.

2.5.5 Forward Propagation

In the convolution layer, the feature map and convolution kernel of the upper layer form the output feature map *j* of the convolution layer. Each output feature map *j* may contain convolution with multiple input feature maps.

$$Z_j^{(l)} = \left(\sum_{i \in M_j} a_i^{l-1} * k_{ij}^{(l)}\right) + b_j^{(l)} \tag{2.8}$$

$$a_j^{(l)} = f(z_j^{(l)}) \tag{2.9}$$

where $\boldsymbol{M_j}$ represents a selection set of input feature graphs. Each output feature graph has an additional offset. However, with an output feature graph, the input feature graph is convolved with different convolution kernels.

A very successful example in the application of CNN is the LeNet-5 system proposed in 1995. On MNIST dataset, it had got an error rate of 0.9% and had already been used for handwritten bank identification. In order to get a smaller error rate, we set a threshold to control some of the metrics that must be excluded (LeCun, Jackel, Bottou, Cortes, Denker, Drucker & Vapnik, 1995).

In 2013, the CNN structure was modified into a deep convolution neural network (DCNN), which was validated on the ILSVRC-2012 dataset. Sainath hopes to use CNN for a large number of vocabulary recognition. Because CNN is an efficient network model, it performs better than DNN when processing speech signal data. Based on the two properties of spectrum variation and model spectral correlation of speech signal, after changing the convolution layer number, the number of hidden units and the pooling policy, the model final got a 4~12% relative improvement over DNNs (Sainath, Mohamed, Kingsbury & Ramabhadran,2013).

Moreover, in 2014, the multichannels deep convolutional neural networks (MC-DCNN) was proposed. Because more and more studies have been made on the time series, the recognition pattern of multivariable has begun to attract attention. In the field of health

information and bioinformatics, multivariable identification algorithm is widely used. The model got the best accuracy (94.67%) on the BID-MC which is better than all the results based on this dataset.

2.6 Applications of CNN

The convolution neural network is now a very famous network framework in the field of deep learning, especially in the field of computer vision. The purpose of CNN is to extract features from certain models and then classify, identify, predict or make decisions based on features. In this process, the most critical step is feature extraction, that is, how to extract features that can distinguish things to the most significant extent. If the extracted feature cannot be divided into different categories, then the feature extraction step will be meaningless. The realization of this great model is the iterative training of CNN. Because this network avoids the complex pre-processing of images, it can input the original images directly, so it is widely used. Starting with LeNet in the 1990s, breakthroughs were made in 2012 which is AlexNet. From GoogleNet to the latest DenseNet, the network is getting deeper, and the architecture is getting more and more complicated.

2.6.1 LeNet

LeNet was proposed by LeCun in 1998 to solve the visual task of handwritting recognition (LeCun, Bottou, Bengio & Haffner, 1998). Since then, CNN architecture has been defined: convolution layer, pooling layer, full connection layer. The LeNet used in today's deep learning frameworks is the simplified and improved LeNet-5. Slightly different from the original LeNet, the new model improves many aspects of the framework structure, such as changing the activation function to the common ReLu.

Yuan and his research team achieved the goal of offline handwritten English character recognition by modifying the model structure of LeNet-5 (Yuan, Bai, Jiao & Liu, 2012,). They modified the number of neurons in each layer of the LeNet-5 model and set special connections between partial convolution layers, which significantly improved the

accuracy of handwritten English characters. At the same time, they used error-correcting code techniques to give CNN an intensive sample learning strategy based on faulty samples. In the end, their experiments achieved 93.7% uppercase characters and 90.2% lowercase characters on the UNEPEN dataset.

When modeling long-distance dependence becomes essential, the Time Delay Neural Network (TDNN) is a better choice (Waibel,1989). Through the stacking of layers, the convolutional network can extract the local features of lower layers and extract more global features in subsequent features. This is a method of convolutional networks for visual tasks, such as the LeNet architecture (Collobert & Weston, 2008). Different languages in different countries have made character recognition a considerable task, Arabic fonts are a relatively unique and extensive typeface. Successfully and quickly recognizing handwritten characters in Arabic fonts has become a research direction for scholars. In 2009, Al-Jawfi developed a method for recognizing handwritten Arabic characters based on LeNet neural network. Al-Jawfi's method is to identify the main shape of the characters and then identify the point symbol in the text. Finally, the two parts of information are combined to achieve character recognition (Al-Jawfi, 2009).

Poultney (Poultney, Chopra& Cun, 2007) described in his article an unsupervised method for learning sparse and overcomplete feature data. The model uses a linear encoder and a linear decoder to transform a code vector into a quasi-binary sparse code vector. Using unsupervised learning to initialize the weights of deep networks can significantly improve the performance of deep networks before using back-propagation for supervised learning (Hinton, Osindero & Teh, 2006). During the experiments, the proposed method is used to initialize the first layer of the convolutional network, while imitating LeNet architecture to conduct training experiments on data samples from MNIST (Deng, 2012). It shows that the error rate of the unsupervised method is slightly lower than the best report result in the MNIST dataset.

2.6.2 VGG

The Visual Geometry Group proposed VGG-Nets at Oxford University. VGG can be seen as a deeper version of AlexNet (Yu, Yang, Bai, Xiao, Yao& Rui, 2016). To solve the initialization (weight initialization) and other issues, VGG uses a pre-training approach which is often seen in the traditional neural network. The training method is to gradually deepen the training on the stable network by training a part of the small network first, and then ensuring that this part of the network is stable.

VGGNet is effective for object recognition in static state images. However, it does not perform well for directly identifying scene data in ImageNet datasets. Therefore, by increasing the number of GPU and increasing the convolution network layer in VGG architecture, i.e., VGGNet-11, VGGNet-13, and VGGNet-16, it can achieve the goal of functional training of astronomical data.

2.6.3 GoogleNet

GoogLeNet (Szegedy, Liu, Jia, Sermanet, Reed, Anguelov, ... & Rabinovich, 2015) is a network structure starting from LeNet-5. It is a convolutional neural network with a standard stacked convolutional layer and one or more fully connected layers. GoogLeNet through the existence of the inception structure in the model, the possibility of controlling the computing resources in the network is well utilized. Moreover, increasing the width and depth of the network without growing the computational workload. At the same time, GoogLeNet uses the Hepin principle and multiscale processing to optimize the network quality.

GoogLeNet refers to Serre's idea of using some fixed Gabor filters for multiscale processing which learns all the filters in the inception structure and reset the structure into a 22-layer network (Serre, Wolf, Bileschi, Riesenhuber & Poggio, 2007). The usage of 1×1 convolution kernel to increase the depth of the network. Hence, GoogLeNet also used a 1×1 convolution kernel to perform dimensionality reduction and network size limitation. GoogLeNet refers to R-CNN's (Girshick, Donahue, Darrell & Malik, 2014) practice of dividing the entire inspection task into two sub-problems. That initially used the

underlying features such as color, texture, etc. to extract proposals that are not related to the category, and then put these proposals into the CNN to train the category information. GoogLeNet also learns from this approach and has improved both phases. The first phase uses multiple border prediction, and the second phase uses a better CNN network structure.

With a relatively deep network structure and robust identification capabilities, GoogleNet has attracted more and more people's attention. GoogleNet's neural networks have been used to classify and recognize Chinese characters. (Liu, Yin, Wang & Wang 2013)(Liu, Jaeger & Nakagawa 2004)(Long & Jin, 2008) (Liu, Yin, Wang & Wang, 2010). Despite this, HCCR is still a severe problem to solve because of its large vocabulary (Yin, Wang, Zhang & Liu, 2013), handwriting styles that vary widely (Wu, Fan, He, Sun & Naoi 2014,), too many similar and confusing characters (Jin, Huang, Yin & He, 2000), and so on. When neural networks have deep enough convolutional layers can not only improve the performance of the model, but also reduce the design of related parameters (Zhong, Jin & Xie, 2015).

In 2016, GoogleNet was used to overcome the shortcomings of traditional scene recognition methods. For example, the recognition model has a high-performance recognition effect outdoors, but the performance of the space is reduced when it is transferred to indoor (Lazebnik, Schmid & Ponce, 2006) (Sivic & Zisserman, 2003), the algorithms based on the popular feature package model may become insufficient when the scene begins to become very complex and there are intra-class changes (Lowe, 2004). Therefore, GoogleNet has been split into three parts: the output features from each of the three parts are applied to scene recognition, which leads to the proposed GoogLeNet-based multistage feature fusion. Experimental results show that the model is superior to the existing CNN model for scene recognition and 92.90% accuracy is identified in the dataset Scene15 (Tang, Wang & Kwong, 2017).

2.6.4 ResNet

ResNet was introduced in 2015 (He, Zhang, Ren & Sun, 2016), this method has sufficient evidence to demonstrate the advantage of using signal additive fusion for image recognition, especially for object detection. As the network goes deeper, the decreaseing in the accuracy of the training set will appear. We confirm that is not due to overfit overfitting, because the overfitting case will let the result of training set be very accurate. A brand new deep residual network was proposed for this problem, which allows the network to deepen the model framework as much as possible by adding convolutional layers. ResNet provides two options for the inverse problem of network depth and accuracy: identity mapping and residual mapping. If the network has reached its optimum, the network is deepened continously, the residual mapping will be pushed to 0, and the identity mapping will be maintained. By this method, the network can always be in an optimal state, and the performance will not decrease any more.

The introduction of residual connections in traditional image recognition architecture can achieve very good performance at low computational cost (Russakovsky et al. 2015). But for such a change, a question was asked: what are the advantages of combining an initial architecture with a residual connection? They argue that residual connections are not necessary for highly trained convolution models. Combined with empirical evidence and experimental results, the existence of residual connections accelerated the training of initial network, and the performance was superior to that of the general network (Szegedy, Ioffe, Vanhoucke & Alemi, 2017). 3.08% top-5 error performance was achieved in ImageNet Large Scale Visual Recognition Competition (ILSVRC).

ResNets outperforms previous models in various tasks such as object detection (Dai, He & Sun, 2016) and semantic image segmentation (Chen, Papandreou, Kokkinos, Murphy & Yuille, 2018). To resolve the problem of vanishing gradient during training convolutional network, simply increasing the depth may not be the best way to improve performance, especially when a series of restrictions arise (He, Zhang, Ren & Sun, 2016). In this regard, Wu et al. constructed a new shallower residual network structure. Through the study of these problems, they conducted experiments in the original model to observe some basic behaviors, then made new expositions on the deconstruction attempts in the

residual network. During the test, they tried to use this shallow residual network for image classification. By developing semantic segmentation methods, this performance can be applied to other areas such as PASCAL VOC, PASCAL Context, and Cityscapes.

2.6.5 DenseNet

DenseNet (Dense Convolutional Network) is a newly proposed convolutional neural network with dense connections (Huang, Liu, Weinberger & van der Maaten, 2017). It is a completely new structure which is mainly compared with ResNet and Inception Networks. In this network, there is a direct connection between any two layers, that is, the input of each layer of the network is the union of the outputs of all previous layers, and the characteristic maps, learned by the layer, are also directly transmitted to the network. All subsequent layers are used as input. We know that ResNet solves the problem of gradient disappearance when the network depth is increased by adding the residual networks. The better results and fewer parameters were achived through the ultimate use of features while reducing the disappearance of the gradient problem.

Convolutional neural networks can be extended to perform object detection by iterating over dense or selected suggested object regions. However, the time-consuming operation of this type of operation will be very slow due to the scaling of the area to be checked for each image. Convolutional layer of DenseNet was used to calculate dense multiscale features. They discussed the use of DenseNet which effectively supports the classification methods proposed for multiple possible regions on the image and the key issues that support each region’s data center and different aspect ratios.

In the ImageNet ILSVRC, a 161-layer DenseNet achieved the top single-crop error of 22.2%. It is reasonable to expect a bigger network to be performed better. However, for most existing DenseNet implementations, the model size is currently limited by GPU memory. A naïve DenseNet implementation may require a large amount of GPU memory: pre-activation of batch normalization and continuous convolution operations may result in feature maps that grow in quadratic curve with network depth if the calculation method

is inappropriate (He, Zhang, Ren & Sun, 2016). A new strategy significantly reduces DenseNet's training time. By introducing shared memory allocation, all layers will use this allocation to store intermediate images. Subsequent layers overwrite previous intermediate image results. This kind of operation can train the pictures with the minimum memory cost. The memory consumption of the function will decrease from the quadratic to linear. On ImageNet, this model achieves a single-crop top-1 error of 20.26%, which (to the best of our knowledge) is the state-of-the-art result.

2.7 Single Shot MultiBox Detector (SSD)

Single Shot MultiBox Detector (SSD) combines the regression ideas in YOLO and the anchor mechanism in Faster-RCNN. It uses the multiscale regional features of all positions in the whole graph to perform regression, which not only maintains the characteristics of YOLO speed but also guarantees that window prediction is as same as that of Faster-RCNN. The core of SSD is to use the convolution kernel on the feature map to predict the category and coordinate offset of a series of Default Bounding Boxes. To improve detection accuracy, SSD predicts on different scales of feature maps.

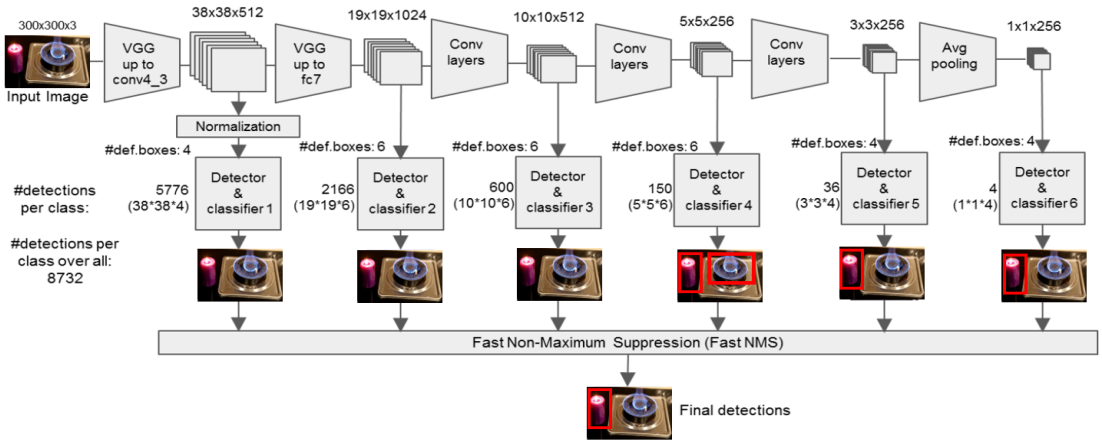


Figure 2.2 The Structure of Single Shot MultiBox Detector (SSD)

The Single Shot MultiBox Detector (SSD) adopts the basic network structure of VGG16 with the first five layers and then uses the Astrous algorithm to convert the Fc6 and Fc7 layers into two convolutional layers. In addition, SSD adopted 3 convolutional layers (Conv 8, Conv 9 ,Conv 10) and an average pool layer. The size of feature map from

these extra convolutional changes relatively large, which allows the detection of objects at different scales, for the feature map in lower-level, the receptive field is relatively small, and for the feature map in the high-level, the receptive field is relatively large. When the convolution operation is performed on different feature maps, multiscale objectives can be detected.

2.7.1 Default Boxes

Single Shot MultiBox Detector (SSD) associates default boxes with feature maps at the top of the network (Liu, Anguelov, Erhan, Szegedy, Reed, Fu & Berg, 2016). For each feature map, the SSD predicts the offsets related to the shape of the default boxes. According to different scales and ratios, default boxes are generated, which are similar to the anchor in Faster R-CNN. The newly added feature map of each convolution layer will be operated by a small convolution kernel to obtain 21 confidence (20 Categories and 1 Background) and four offsets (Shape Offsets) about the object categories. If the size of feature map is $m \times n$ with p channels, each feature point on feature map corresponds to K default boxes, and the number of object categories is C. Then results in a total of $K \times (C+4)$ filters which are around each location in the feature map. The output in total is in size of $(m \times n) \times K \times (C+4)$ for an $m \times n$ feature map.

2.7.2 Loss Function

The loss function is divided into two parts: calculating the score (confidence) of corresponding default box and target category, corresponding regression results (location regression). For each bounding box to predict the target category, we have confidence (c_1, c_2 ,…, c_p) and box offset $\Delta(c_x, c_y, w, h)$. The term x_{ij}^p refers to the correct probability of category p between the *i-th* bounding box and *j-th* ground turth box. If the bounding box matches with the ground truth box, then the $x_{ij}^p = 1$. Otherwise, the $x_{ij}^p = 0$. The total loss function is the weighted sum of the loss function of the target classification and the offset loss of the regression position:

$$L(x, c, l, g) = \frac{1}{N}\left(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)\right) \quad (2.10)$$

where the term *N* refers to numbers of the bounding boxes which match the ground truth boxes; L_{conf} is the Softmax loss; c refers to the confidence for each categories; L_{loc} is the smooth loss function L_1, which is used to for the certer coordinate, weight and highth of the regression box; *l* is the bounding box; *g* is the ground truth box; α is the weight value in the network.

2.8 YOLO and R-FCN

YOLO is one of the great object classification approaches which is entirely different from other approaches, such as R-CNN, Fast R-CNN, and Faster R-CNN. YOLO's full name is You Only Look Once which also illustrate how this method process images. When the input image enters the neural network of YOLO, the image only be operated once by YOLO, which is not a conventional classifier like R-CNN. The network in YOLO divides the input image into a grid of 13×13, and each grid will predict five bounding boxes which maybe intercept the grids. These bounding boxes elaborate the rectangle that encloses a target, that means those bounding boxes will not only detect whether there is a target but also describe what kind the object is. The correct predictions are based on the confidence scores that tell users how good it is that the bounding box encloses a target. The confidence scores reflect if the shape of the box is good or not instead of what kind of targets are in the box. For each bounding box, the grid also predicts a class, it gives a probability distribution over all the possible classes.

YOLO trained the PASCAL VOC dataset which including 20 different classes such as a bicycle, boat, car and so on. When the bounding box gives out the confidence score, YOLO will combine the score with the class prediction together to display the percentage of this bounding box contains a particular target. When each cell of the image predicts five bounding boxes, it will end up with hundreds of bounding boxes in total. The reason why YOLO is so powerful and fast is that the neural network in YOLO processes hundreds separated bounding boxes at the same time and does not run the image a second

time.

YOLO solves the object detection problem as a regression problem. The model is based on a single End-To-End network, and the output from the input of the original image to the position and category of the object is completed. After an input image passes an inference, the position of all the objects in the image and its corresponding category with confidence probability can be obtained. The YOLO network draws on the GoogLeNet classification network structure. The difference is that instead of using the Inception Module, YOLO uses a simple 1×1 convolutional layer (The existence of 1×1 convolution layers here is for cross-channel information integration) add with a 3×3 convolutional layer. On the other hand, YOLO uses the full graph as the Context information, thereby reducing the recognition of background errors as object error rates, which leads to that YOLO has strong generalization ability.

R-FCN is a new target detection structure proposed by Jifeng Dai of Microsoft Research Asia in 2016. It modified the traditional Faster R-CNN structure by moving the convolutions operation to the front of RoI layer and used a position-sensitive feature map to evaluate the probability of each category. This approach can keep R-CNN achieved higher accuracy of position and greatly increased the speed of the detection.

The fundamental problem that R-FCN has to solve is the one of slow detection speed of Faster R-CNN. The slow speed is because different proposals do not share the structure behind the RoI layer. Just imagine that if there are 300 proposals, then the fully-connected network after the ROI must be calculated 300 times whose consuming is too scary. Therefore, the structure was moved that follows the RoI to increase the speed, but the RoIs would cause the translation variability problem after the step Conv5. The variability of the structure must be strengthened by other methods. Therefore, a position-sensitive score map was added.

The concept of position-sensitive score map first came from another article of example segmentation by Jifeng Dai as well. Each output pixel in the InstanceFCN is used as a classifier to determine the relative position of the target object. The redder color

of the score map after the convolutional layer operation represents the higher confidence score. The position-sensitive score maps were drawn based on this idea, but its purpose is not to segment the target instances. R-FCN combines the score maps within the RoI region to obtain voting scores belonging to a class which uses for the following classification.

There are two concepts in the paper, translation invariance, and translation variance. Translation invariance refers to the classification of an object using a basic classification structure such as ResNet or Inception, no matter how the object is distorted or translated, the object is ultimately identified by this class. The deeper the level of the network is, the more apparent this characteristic will be. Translation variability is detected for the target. For example, if a cat moves from the left side of the picture to the right side, the change in the detected cat's coordinates is called translational variability. When the convolutional network becomes deeper, the feature map of the last convolution layer becomes smaller. At the same time, the small offset of the input will not be perceived on the small feature map which comes from operations after numbers of pooling layer. The gap between the results obtained with different network architecture designs proves the importance of translation variability for target detection.

Chapter 3
Methodology

In this chapter, we focus on articulating research methods and echoing the objectives of this thesis. The details of research methodology for flame detection and recognition are mainly mentioned and introduced by using the feature description methods in this chapter.

3.1 Research Designing

To achieve the detection and recognition of the flame is the primary purpose of this thesis. As shown in Figure 3.1, the flowchart of flame detection and recognition for each step is pointed out through three modules as the structure of this research.

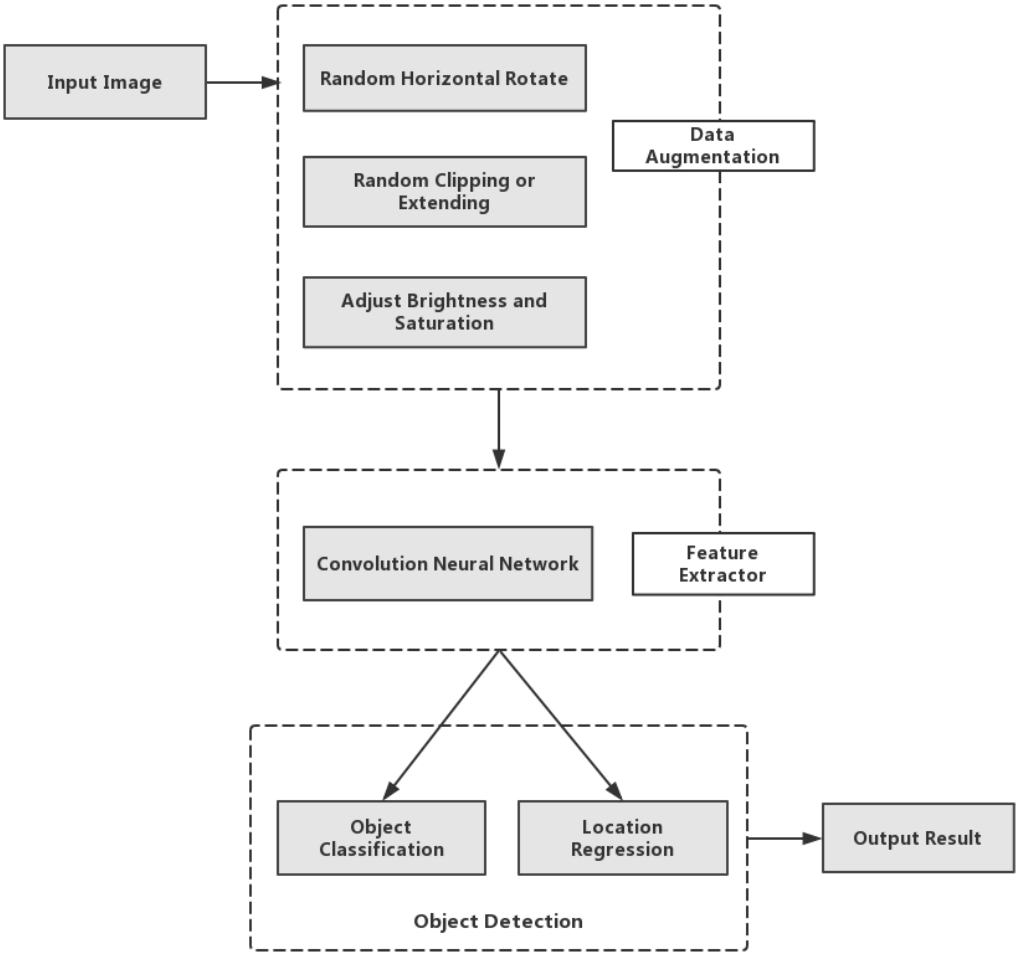


Figure 3.1 The Flowchart of the Steps of Flame Detection and Recognition

The first module is thought as the Data Augmentation which mainly was designed to enhance the dataset by changing the size and shape of the image. The second module is the Feature Extractor which was designed as the leading part in this structure. We consider the complexity of our problems is lower than that of ordinary objects, and the limitation of our computer hardware, we design an eight layers CNN as the feature extractor. The

following module is the part of Object Detection, the detection task would be split into Object Classification and Location Regression.

3.2 Data Collection and Manually Label Marking

The most critical step for deep learning is to collect the appropriate datasets. The flame classification is a segmentation task which does not have an authoritative database from the Internet. Therefore, we decided to gather fire images by using video collection and manually labelling. We enrich the flame categories in our dataset as far as possible, and finally gather six different kinds of flames: Campfire, Candle Fire, Gas Fire, Blowtorch Fire, Gas Tank Fire, and Alcohol Fireplace.

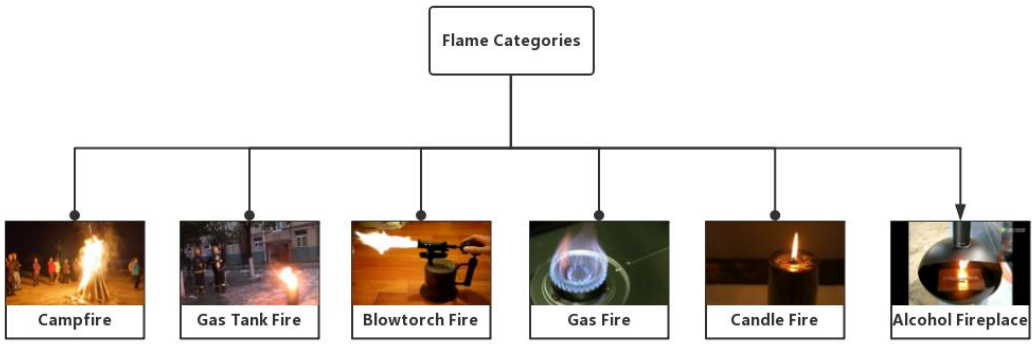


Figure 3.2 Examples of Flame Categories

For those flames that we find in our home or we can approach, such as candlelight and gas, we use the method of direct shooting by using a camera. We shoot the flames at various positions, angles, and distances to enrich the information of the flames so that we can get a better model. At the same time, we except that the shape of a flame in our dataset is diversable. In the selection of training data for flame detection, we require the flames in a video to have rich morphological and positional information to train a better model. There are serval points we focus on what can improve the quality of our dataset:

- The small movement of the camera lens can avoid the phenomenon of dynamic collapse, so it ensures that the picture of the video is clear.
- Do not let fire always appear in the same parts of the picture. The probability of

the flame occurs in each region should be equal.

- The size of a flame in the picture to be enhanced which needs to record the flames from different distances.

After completing the video capture, we need to sparse the video into image dataset. We select 100 frames for each type of flame at regular intervals as training images, which means we pick 100 images for each flame category from those videos. Additional, we set the image size uniformly to 960×520. Therefore, for the above five categories, we collected a total of 500 training images.

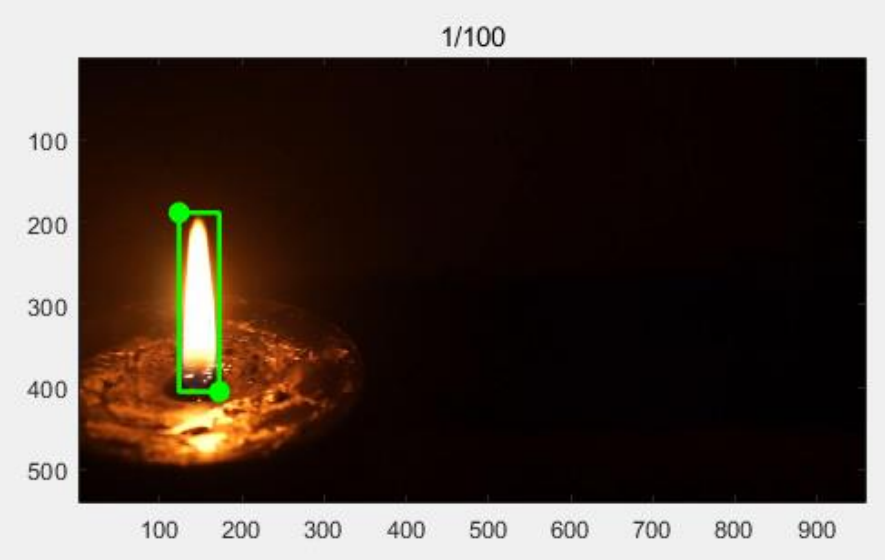


Figure 3.3 Manual Marking Label Diagram

After extracting the training image, we need to acquire the ground truth box and the coordinate parameters corresponding to the box. Because we collect the dataset by recording videos, we do not have the coordinate parameters which correspond to the ground truth box. To solve this problem, we used MATLAB to create a graphical user interface to label the flame position manually. The software interface is shown in the Figure 3.3 above. A ground truth box can be determined by clicking on the position of the upper-left corner and lower-right corner of the object. After manually marking the flame position, this procedure will generate new text files consisting of the fire type and four vertices coordinate parameters of the ground truth box: X_{min}, X_{max}, Y_{min}, and Y_{max}. Each text file displays four coordinates that built-in the size of 960×520 as the image data. The

coordinate of the top-left corner is represented as (X_{min}, Y_{max}), and the right point is represented as (X_{max}, Y_{max}). Moreover, the bottom corners are represented as (X_{min}, Y_{min}), (X_{max}, Y_{min}) respectively from left to right.

3.3 Data Augmentation

One barrier to resolve problems with deep learning is the amount of data required to train the model. The demand for big data is because the participation of a large number of parameters in the model will lead to better results. There is a nearly linear relationship between the amount of data required for deep learning and the size of the model resulting from training. The model should be large enough to capture the relationships in the data (such as the texture and shape in the image, etc.), the specific details of the problem (such as the number of categories). Early layers in the model capture low-level relationships (such as edges and patterns, etc.) between different parts of the inputs. The following layers capture information that helps make the final decision, which usually helps to distinguish between the desired outputs. Therefore, if the complexity of the problem is high (such as image classification), the number of parameters and the amount of data required are also vast.

The data used in this study is all from our video shots with a camera, and then through our program compiled in Matlab. The program divides the number of video frames into equal parts and intercepts the corresponding frames as the input pictures. Since we set the video frame size to 100 in the program, we got 600 photos corresponding to the six types of flames we choose.

Successful neural networks have millions of data, 600 images as input data is a quite small volume for training flame recognition model. Having a large number of flame image data parameters provides the same proportion of instance data for our deep learning model so that we will obtain excellent performance for flame recognition. We employ Data Augmentation to generate more training images by using the original images. Specific methods we used in this thesis are resizing; random horizontal flipping; random

cropping; color augmentation.

3.3.1 Resizing

An excellent neural network can recognize objects of different sizes. Different sizes of the same object can affect the accuracy of deep learning. For a flame, we want to identify its aspect ratio changes without affecting recognition. Data enhancement is also the preparation of data for entering the neural network while modifying data, we resize the image to the scale that can be entered into the convolution layer. We refer to the neural network model structure of SSD and YOLO, where we have to resize the original image size from 960×520 to 448×44.

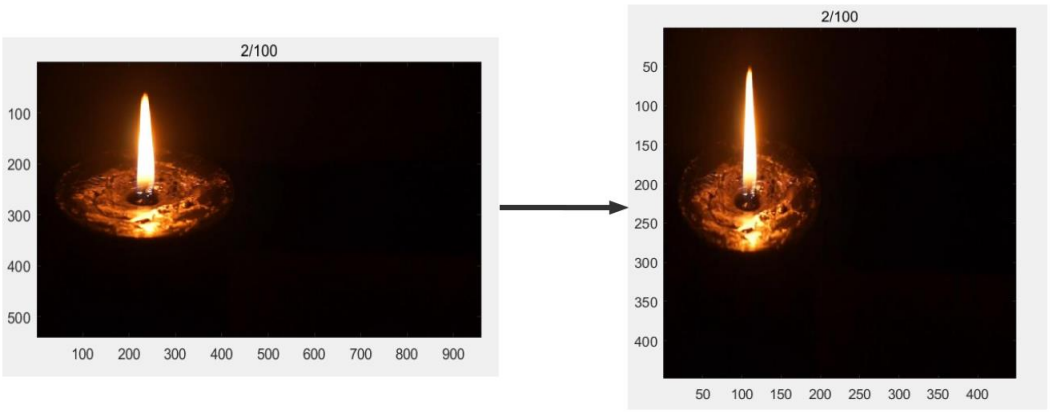


Figure 3.4 Examples of Flame Resizing

3.3.2 Random Horizontal Flipping

Flipping images is another way to enhance data, usually by flipping them horizontally and vertically. For our goal, in most cases, the fire captured by the camera is burning upwards, unless the camera is flipped because it is not fixed. Figure 3.5 is an example of a rollover image. After resizing the original picture, we flip the image data horizontally with a certain probability.

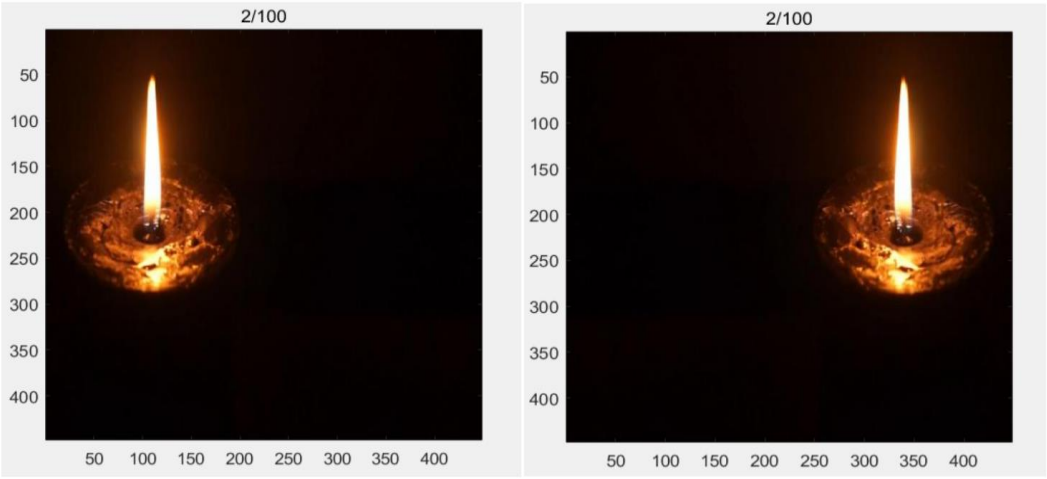


Figure 3.5 Examples of Random Horizontal Flipping

3.3.3 Random Cropping

Unlike scaling, we randomly sample a portion of the original image. Then, we adjust this part of the image to be the same size as the original image. The popular name for this method is Random Cropping. Here's our example of picture cropping. We can find that cutting is a kind of different from scaling technology. Also, we can adjust the ratio of the cropped image to the original image according to our expectations. Hence, in this research, we set the cropping ratio to [0.9, 1].

Figure 3.6 starts from the left side: the original image after resizing, a square part is cut out from the upper left corner, and then a part cut out from the lower right corner. All the images cropped are trimmed to the original image size (448×448).

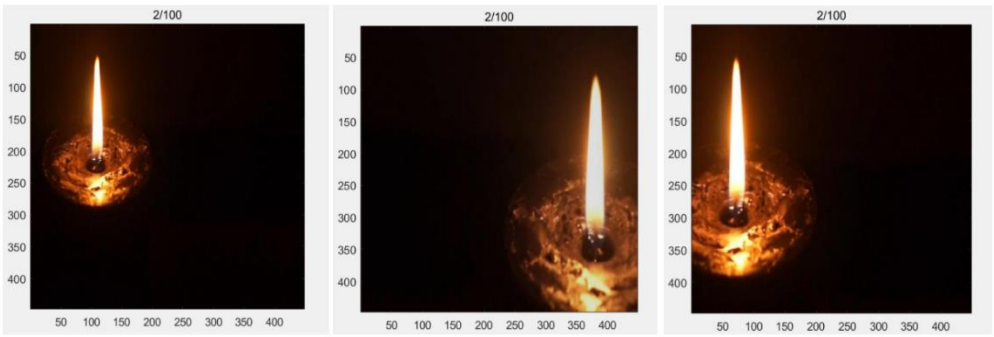


Figure 3.6 Example of Random Cropping

3.3.4 Color Augmentation

In general, the collected color image is represented by RGB space when performing image processing. R, G, B respectively represent red, green, and blue color channels. A pixel is associated with a 3D vector. Each component of the vector represents the brightness of the color corresponding to red, green, and blue (Kumar & Verma, 2010). In addition to RGB space, YIQ space, YUV space, HSI space, HSV space and CMYK space are commonly used to represent color images (Chen, Shi & Xuan, 2007). Both the RGB and CMY color models are hardware-oriented, and the HSV (Hue, Saturation, Value) color model is user-oriented. Therefore, this paper used HSV space to represent the image. The HSV color space is also commonly referred to as the HSB color space, which is consistent with human subjective feelings. H represents the Hue of the pixel, S stands for the Saturation of the pixel, and V refers to the Value, i.e., the brightness of the pixel (Vitabile, Pollaccia, Pilato & Sorbello, 2001). In this thesis, we enhance the color of images in the HSV color space. The first step we need to do is to transfer the RGB space into HSV space. We change the S and V luminance components of the saturation and keep the Hue unchanged:

$$R' = R/255,\ G' = G/255,\ B' = B/255 \tag{3.1}$$

$$C_{max} = max(R', G', B'),\ C_{min} = min(R', G', B'),\quad \Delta = C_{max} - C_{min} \tag{3.2}$$

Saturation is calculated by

$$S = \begin{cases} 0, & C_{max} = 0 \\ \frac{\Delta}{C_{max}}, & Cmax \neq 0 \end{cases} \tag{3.3}$$

It is calculated by

$$V = C_{max} \tag{3.4}$$

We multiply that in interval (0.8,1.2) on the S and V components of each pixel to increase the illumination variation.

$$S' = \alpha \cdot S, V' = \alpha \cdot V, \ \alpha \in [0.8, 1.2] \tag{3.5}$$

The result is shown in Fig. 3.7, the left picture is the original image, following two images with saturation and brightness changes.

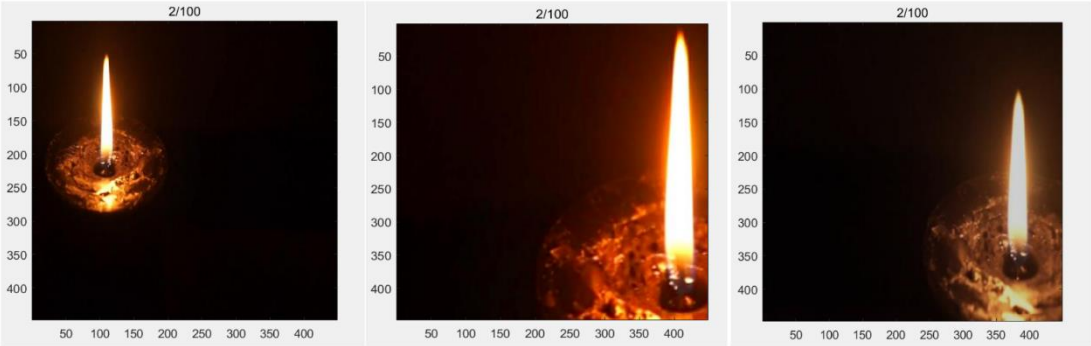


Figure 3.7 Example of Color Augmentation

We repeat the above three data enhancement methods, the final image is scaled to the input size of CNN that we designed, i.e., 448×448, the final training dataset is obtained. For each original training image, we generated 12 enhanced training samples, 10 of which were training samples, the remaining 2 were verification samples. As a result, we finally got 5,000 training samples and 1,000 valid samples.

3.4 Model Details

In this thesis, we design a new convolution network model based on the SSD concept, we named it as Multiple Fire Detection (MFD). The SSD is an approach which is based on the feedforward convolutional neural network. SSD provides bounding boxes which are in a fixed-size collection and marks the object class instance existing in those bounding boxes. Then, SSD will produce a non-maximum suppression step to generate the final result.

The flame detection can be considered as a sub-task of flame recognition. We divide the task of flame recognition into two parts: flame classification and flame locating. We firstly extract high-level abstract features from input images, and then take the features extracted as the input of the two sub-networks.

In this thesis, we select Tensorflow as our machine learning framework. TensorFlow is the second generation of artificial intelligence learning framework developed by Google based on DistBelief. It is a system that transmits complex data structures to the artificial intelligence neural network for analysis and processing (Abadi, Barham, Chen, Chen, Davis & Kudlur, 2016).

Feature extraction is an essential part of image processing. Most models use deep learning, the feature extractor is a CNN model, the standard extractors used in other projects are VGG, ResNet, InceptionNet and so on. Considered the massive scale of the above network model, the complexity of flame identification is also lower than that of the general object recognition, so we have designed a 8 layers CNN as the MFD feature extractor.

The following pictures are the whole structure of MFD from the Tensorflow, our framework mainly consists of two types of layers: convolutional layers and fully connected layers.

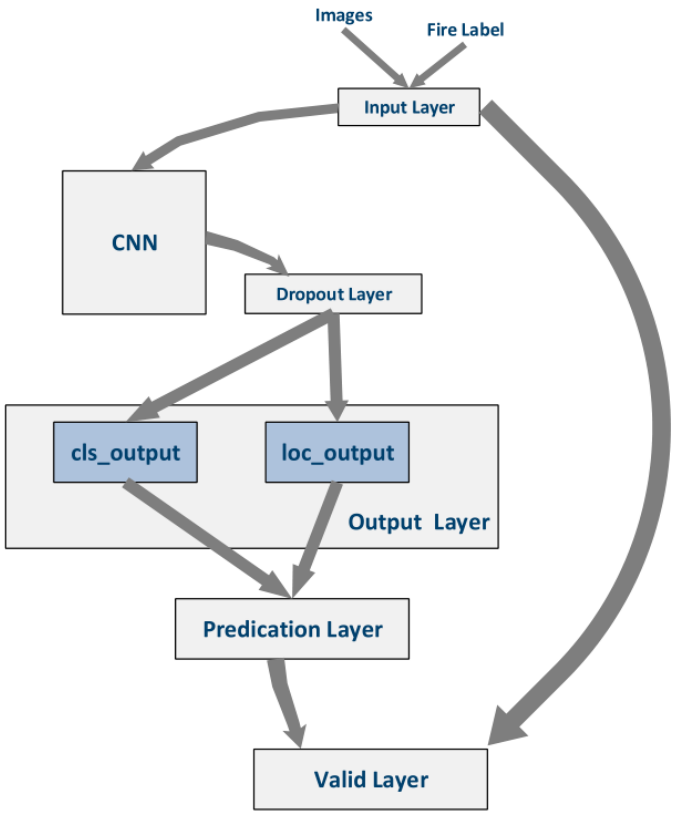


Figure 3.8 Overall Structure of Whole Network

- **Input Layer**

This layer is designed to preprocess the image data. In data preprocessing, the first step is data normalization. Although there are a series of possible methods, this step is usually based on the specific circumstances of the data and selected. The standard methods for feature normalization include the following: Simple Scaling (Anderson, Thouless, Abrahams & Fisher, 1980), Sample-by-sample Average Reduction and Feature Standardization. The subsequent processing of data preprocessing is vital because many default parameters assume that the data has been scaled to a reasonable interval.

In our model, we adopt the Simple Scaling. Our goal is to rescale the values of each dimension of the data (these dimensions may be independent of each other). The subsequent processing of data preprocessing is vital because many default parameters assume that the data has been scaled to a reasonable interval. In our model, we use Simple Scaling, our goal is to re-adjust the values of each dimension of the data (these dimensions may be independent on each other). Our image is stored as a matrix of three color components, where each pixel value is an integer in the interval [0,255]. As the network model is a continuous model, we need to convert the discrete color values to continuous floating-point values for storage so that the final data vector falls within the interval $[-1, 1]$. At the same time, the ground truth coordinate of each training sample is also encoded at this level during training.

- **Convolution Layer**

The CNN part of our MFD model consists of eight convolutional layers, each of which has a convolution operation and a maximum pooling layer. For the convolutional layer, its basic structure is:

$$y = \sigma(W \cdot x + b) \tag{3.6}$$

After entering specific parameters and offsets:

$$
\begin{aligned}
y^{(1)} &= \sigma(W^{(1)} \cdot x^{(0)} + b^{(1)}) \\
y^{(2)} &= \sigma(W^{(2)} \cdot x^{(1)} + b^{(2)}) \\
&\vdots \\
y^{(l)} &= \sigma(W^{(l)} \cdot x^{(l-1)} + b^{(l)})
\end{aligned} \tag{3.7}
$$

From the equation, *x* denotes the input layer; *y* refers to the output layer; *W* stands for the convolution template; *b* represents the basis and l is the number of layer. The dimension of *x* in our model is [*N, M, N, C*]. The *N* is the batch size in training, which is the number of the input samples in only one step of training. $M \times N$ indicates the size of the input feature, *C* indicates the number of input features. The dimension of the convolutional template *W* is [*m, n, C, D*], $m \times n$ means the size of the convolution template, *C* is the number of input features, *D* is the number of output features. Moreover, σ is called an activation function. When the image process after the convolutional layer, the dimension of the output *y* goes to be [*N, M, N, D*].

The nature of convolution is a linear operation. After convolution, the result needs to be non-linearized through a nonlinear activation function, so it is imperative to choose the appropriate activation function. There are also many choices, like Sigmoid, ReLU, and LeakyReLU:

$$\text{Sigmoid}(x) = \frac{1}{1+e^{-x}} \tag{3.8}$$

$$\text{ReLU}(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{3.9}$$

$$\text{LeakyReLU}(x) = \phi(x) = \begin{cases} x & x \geq 0 \\ 0.1 \cdot x & x < 0 \end{cases} \tag{3.10}$$

From these equations, we compare them and find the most suitable one. The Sigmoid is the oldest activation function. However, when CNN goes to a deeper layer, the problem of gradient vanishing is easy to occur, which making the model difficult to train. At present, fewer models have chosen sigmoid function as the activation function. The emergence of ReLU function solves the problem of gradient vanishing. However, when input parameters are negative, the partial gradients of the ReLU function always give the number 0, and that property may lead to some parameters not being trained. Therefore, we chose LeakyReLU as the activation function. Fig. 3.9. shows the difference

between those functions.

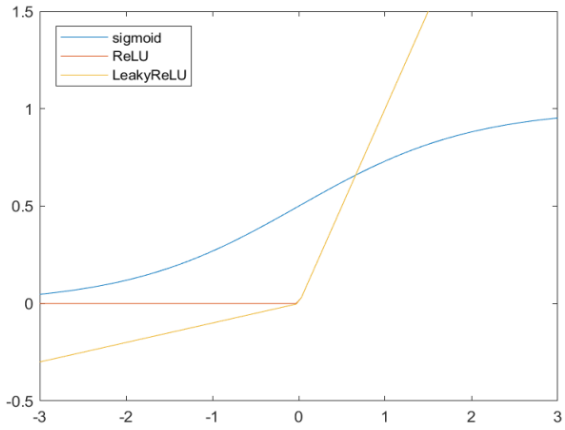


Figure 3.9 Graph of Three Activation Functions

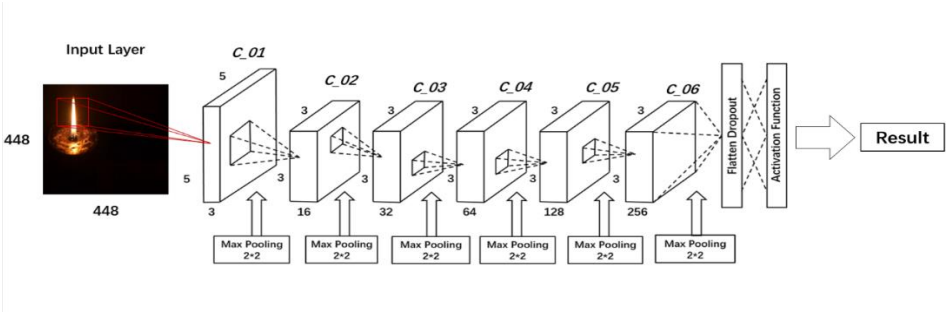


Figure 3.10 Overall Structure of Convolution Neural Network

Table 3.1 Details of MFD's CNN

Layer	Size of Output	Structure parameter
Input Layer	[*Batch_Size*, 448,448,3]	Normalize the pixel value to [-1,1]
*Con*_01	[*Batch_Size*, 448,448,3]	conv [5,5,3,16] max pooling 2×2
*Con*_02	[*Batch_Size*, 224,224,16]	conv [3,3,16,32] max pooling 2×2
*Con*_03	[*Batch_Size*, 112,112,32]	conv [3,3,32,64] max pooling 2×2
*Con*_04	[*Batch_Size*,56 ,56,64]	conv [3,3,64,128] max pooling 2×2
*Con*_05	[*Batch_Size*,28,28,128]	conv [3,3,128,256] max pooling 2×2
*Con*_06	[*Batch_Size*,14,14,256]	conv [3,3,256,512] max pooling 2×2
*Con*_07	[*Batch_Size*,7,7,512]	conv [3,3,512,1024] max pooling 2×2
*Con*_08	[*Batch_Size*,14,14,256]	conv [3,3,256,512] max pooling 2×2

After the convolution layer, the max pooling layer always followed. In each maxpooling with the size $m \times n$, the output data is the most significant value from each grid in that max pooling. If the input data feature is in the size of [N, M, N, C], then the size of the output data will become [N, $\lceil M / m \rceil$, $\lceil N / n \rceil$, C], $\lceil \cdot \rceil$ represents the ceiling function. The structure of fully connect layer is presented as:

$$Y = \text{maxpool}(\sigma(x \cdot W + b)) \tag{3.11}$$

where x in that equation is in the size of [B, M]; the weights are at [M, N]; the output Y is in size [B, N]; we set the B as the *Batch_Size* in training. Based on the above basic structure, the parameter settings of the model are shown in Table 3.1.

- **Dropout Layer**

To avoid the problem of overfitting in this model, we add a *Flatten_Dropout* layer so that only a part of the nodes are trained during training. Dropout is an entirely different technique than regularization. Dropout does not modify the cost function but modifies the depth network itself (Nielsen, 2015). Our model is used to make predictions, the dropout is a way to ensure that our model remains robust in the absence of a single clue. It is similar to L1 and L2 regularization, which tends to reduce the weights and make more robust under the situation that ignores some connections in the network. For our MFD model, we set the dropout rate to be 0.5 in the training time. The reason is that when the dropout is 0.5, the random number of generated network structure is the maximum.

- **Output From CNN and Loss Function**

For an n-class classification problem, we generally set a probability vector $p = (p_1, p_2, \ldots, p_n)$ as the output data. The probability vector $p_i \in [0, 1]$ which indicates the probability that the input image belongs to the *i-th* category. We also set the label $q = (q_1, q_2, \ldots, q_n)$ as the label of training sample. Therefore, if the training sample belongs to *i-th* classification, then the q_i will be 1 (q_i =1), and q_j will be 0 (q_j =0) where q_i is not equal to q_j. The general object classification problem uses the Cross-Entropy as the Loss Function:

$$L_{cls} = \text{crossentropy}(q_k|p_k) = E_k[-\log\ p_k] = -\sum_{k=1}^{n} q_k \log p_k \tag{3.12}$$

For the aim to meet $pi \in [0, 1]$, $\sum_{k=1}^{n} p_k = 1$, we normalize the output $f_{loc} = (y_1, y_2, \dots, y_n)$ from Deep Neural Network (DNN) by using SoftMax function. We take the result as the confidence of this class:

$$p_i = \frac{e^{(y_i)}}{\sum_{k=1}^{n} e^{(y_k)}} \tag{3.13}$$

In another hand, we regard the positioning problem of the flame as a general regression problem. We set up the output of location regression as f_{loc}, and set up b as the location of the bounding boxes of the training sample. Also, we add a loss function ℓ_{loc} for the error between predicted location and excepted location. The predicted location f_{loc} and excepted location b are also suitable for the box encoding in coding part. Then, the loss function of positioning problem is:

$$L_{loc} = \ell_{loc}(f_{loc} - b) \tag{3.14}$$

For the Loss Function of Regression Problem, there are many functions that we can select, and the three loss functions are the most common ones in machine learning:

$$L_1(x) = |x| \tag{3.15}$$

$$L_2(x) = x^2 \tag{3.16}$$

$$SmoothL_1(x) = \begin{cases} |x| - 0.5 & |x| > 1 \\ 0.5x^2 & |x| \le 1 \end{cases} \tag{3.17}$$

We compare them in the Matlab, and the graph shows blew, and we select *SmoothL1* function as the loss function ℓ_{loc}.

Finally, the loss of function in total is a linear weighting of the classification problem loss function L_{cls} and the location problem at any time function L_{loc}. Minimizing the total loss function, we achieve a flame-recognized DNN by training.

$$L_{total} = L_{loc} + L_{cls} \tag{3.18}$$

We finally get the loss function

$$L_{total} = SmoothL_1\left(\frac{e^{(y_i)}}{\sum_{k=1}^{n} e^{(y_k)}} - b\right) + (-\sum_{k=1}^{n} q_k \log p_k) \tag{3.19}$$

- **Learning Rate**

Learning rate is a crucial hyper-parameter, which controls the convergent speed. Most optimization algorithms (such as SGD, RMSprop, and Adam) include it. The smaller the learning rate, the slower we fall along the loss gradient. In the long run, this choice may be useful because that can avoid missing any optimal local solution, but it also means that we have to spend more time for the converge, especially if we are at the highest point of the curve.

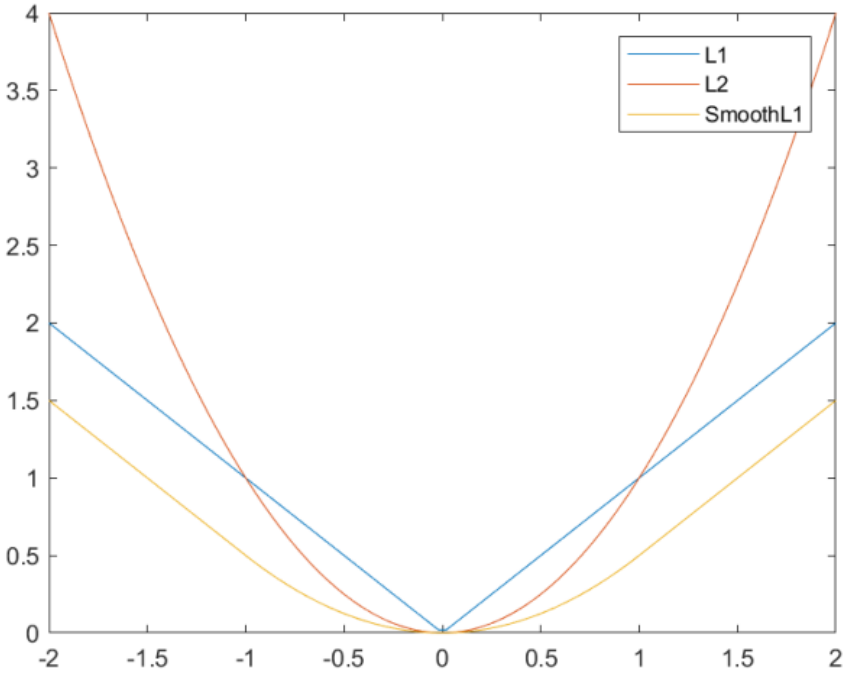


Figure 3.11 The Three Loss Function

The learning rate directly affects how quickly our model can converge to a local minimum (i.e., achieve the best accuracy). In general, the higher the learning rate, the faster the neural network can learn. If the learning rate is too small, the convergence of the network model will be too slow, but if it is too large and exceeds the limit, the model will converge correctly. The rate will have greater fluctuations, and the accuracy rate will generally be lower. Here, we set the number of training rounds to *n*; we take the *k*-round learning rate as λ_k where we set the term γ_{max} as -3 and γ_{min} as -5,

$$\lambda_k = \exp\left(\gamma_{max} - \frac{k-1}{n-1}(\gamma_{max} - \gamma_{min})\right)^{ln10} \tag{3.20}$$

Chapter 4

Results

The main content of this chapter is to introduce the schema of methods and implementation of flame detection and recognition. The experimental environment will be articulated in this chapter. Also, each type of flame in our dataset will be detailed, the results of flame detection and recognition will be clarified. Moreover, the results and findings will be evaluated as well as the limitations of this thesis will be pointed out at the end of this chapter.

4.1 Experimental Parameters and Environment

This thesis goal is to implement flame detection and recognization by using a surveillance camera to prevent fire disaster. We design the interface using Matlab and train the model with Python under the TensorFlow framework.

In the process of model training, some important parameters need to be set: batch size, number of the epoch, learning rate and weight of location loss α in the above model. Table 4.1 shows the parameters.

Table 4.1 Training Interface Parameters

Parameters	Settings
acfunc	Activation function, here is the LeakyReLU.
box_encode	Box encoding, optional 1, 2, 3, 4, default is 1.
loc_weight	Positioning loss function weight, default is 1.0
num_epoch	Training rounds, the default is 50
min_batch	Batch size, the default is 50
max_lr_exp	That is γ_{max}, the default is -3
min_lr_exp	That is γ_{min}, the default is -5
log_period	The number of steps for calculating the relevant indicators of training data and verification data. Default is 20
save_period	The number of steps to save a model, the default 10000, if more than the total number of training steps, then the training is completed and then saved.
outdir	Indicating the name of the saved model. It is used to mark the differentiated model. If it is not specified, it is marked by the start time of the training by default.

First, batch size represents the number of samples passed to the model for training in one step of training. This number is mainly limited to the memory of the machine. If the graphics card is used for acceleration, it will be limited by the size of the memory. For example, if we take a batch size of 50, the memory usage is about 3G, because TensorFlow will occupy almost all of the video memory by default, we cannot know how much memory it needs. Training on the NVIDIA GTX 980M 8G graphics card consumes approximately 9G of memory. If the memory is insufficient, we need to reduce the batch size.

Then, epoch represents the number of rounds of training. In one round of training, a training sample will be traversed. Only when the number of training rounds is large enough, the model can be adequately trained to get better accuracy. We take a training round of 100. In the case of a batch size 50, we will perform 10,000 steps of training.

4.2 Bounding Box Setting

After setting the training parameter and training interface, we still need design different algorithms for bounding box regression so that we could figure out the best flame recognition model. The Bounding Box Regressors are essential because the initial region initiatives might not entirely coincide with the region that is indicated by using the learned features of the Convolutional Neural Network. It is a kind of a refinement step. Based on the weights of the classifier (e.g., Neural Networks, SVMs), the region requests are regressed. The function used for regression is the one obtained at the end of the final pooling layer. This kind of regression gives a better estimate of the object position than our simple Proposal Generators since it is based on the features generated by the Deep Neural Networks.

For the positioning problem of flames, we treat it as a general regression problem. As we mentioned in the previous chapter, the predicted location f_{loc} and excepted location b in location loss fucntion are also suitable for the box encoding in coding part. Hence, we attempt four different parameters to adopt for our model.

Table 4.2 Four Kinds of Box Coding

Model	Box Encoding
MFD_1	$\{x_{min}, x_{max}, y_{min}, y_{max}\}$
MFD_2	$\{x_c/w, y_c/h, w_b/w, h_b/h\}$
MFD_3	$\{x_c/w, y_c/h, \sqrt{w_b/w}, \sqrt{h_b/h}\}$
MFD_4	$\{x_c/w, y_c/h, \log(w_b/w), \log(h_b/h)\}$

Inbox encoding, we design the coding as $[x_{min}, x_{max}, y_{min}, y_{max}]$. w and h denote the width and height of the input image, (x_{min}, x_{max}) and (y_{min}, y_{max}) are the coordinate of upper left corner and the lower right corner of the bounding box respectively. We also

indicate the coordinates of the center of the bounding box (x_c, y_c) as:

$$x_c = \frac{x_{min}+x_{max}}{2},\ y_c = \frac{y_{min}+y_{max}}{2} \tag{4.1}$$

where w_b, h_b respectively indicate the width and height of the bounding box, which satisfy

$$w_b = x_{max} - x_{min},\ h_b = y_{max} - x_{min} \tag{4.2}$$

MFD_2 is equivalent to a linear transformation of MFD_1, the difference is that MFD_2 normalizes the value between [0, 1]. MFD_3 and MFD_4 aim to improve the positioning accuracy of the model for small objects. Because MFD_1 and MFD_2 are an absolute measure of positioning errors, the deviation between large and small objects is treated equally. For example, a 10-pixel deviation may be trivial for a large object, but it can be quite large for a small object. As shown in Figure 4.1, the width and height of MFD_3 and MFD_4 enable a small amount of deviation, when width and height are minimal to bring about a large change in coding value, thus improve the positioning accuracy of small objects.

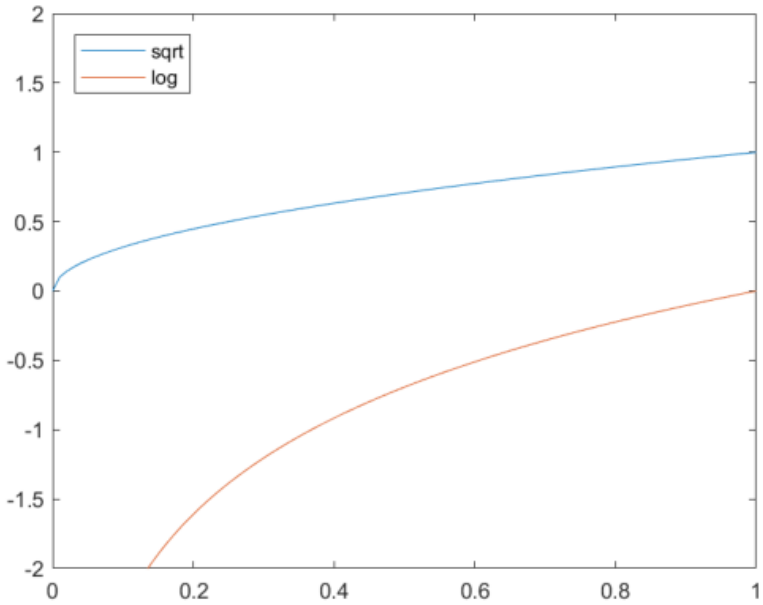


Figure 4.1 Width and High Coding Function

The blue curve represents bounding box No.3 and the orange curve represents code bounding box No.4. After comparison, it is not difficult to find that when the width and

height are made small, a small deviation will bring about a large change in the code value, thereby improve the positioning accuracy of small objects.

4.3 Multiple Flame Recognition

The classification is the most important step in our experiment. It directly influences the results of the recognition. In this thesis, we investigate six different flame categories including Campfire, Candle Fire, Gas Fire, Blowtorch Fire, Gas Tank Fire, and Alcohol Fireplace. When a flame appears through the camera, our model will automatically detect the region and recognize one of these five different flames. The system will restrain the flame in a bounding box with name of the flame and the accuracy for classification. Convolutional neural network has a unique way to convert the color layer of a picture to a digital matrix. Therefore, the result from the CNN still seems as good as that of object detection which uses color model, the accuracy of the bounding box locating is pretty good. The background shows the similar color to the flame is not appeared in those bounding boxes. The results reached the highest accuracy (98.6%).

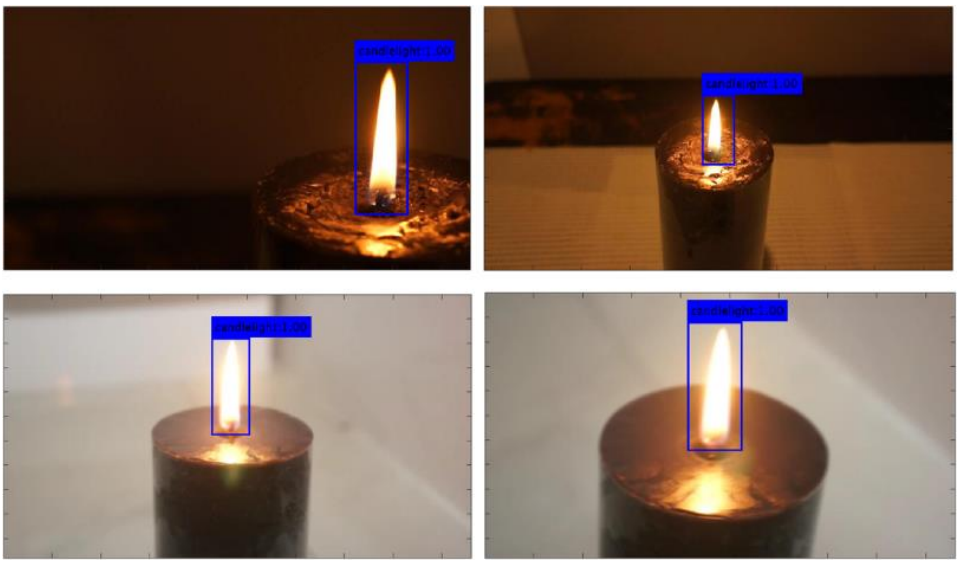


Figure 4.2 The Result of Candle Fire

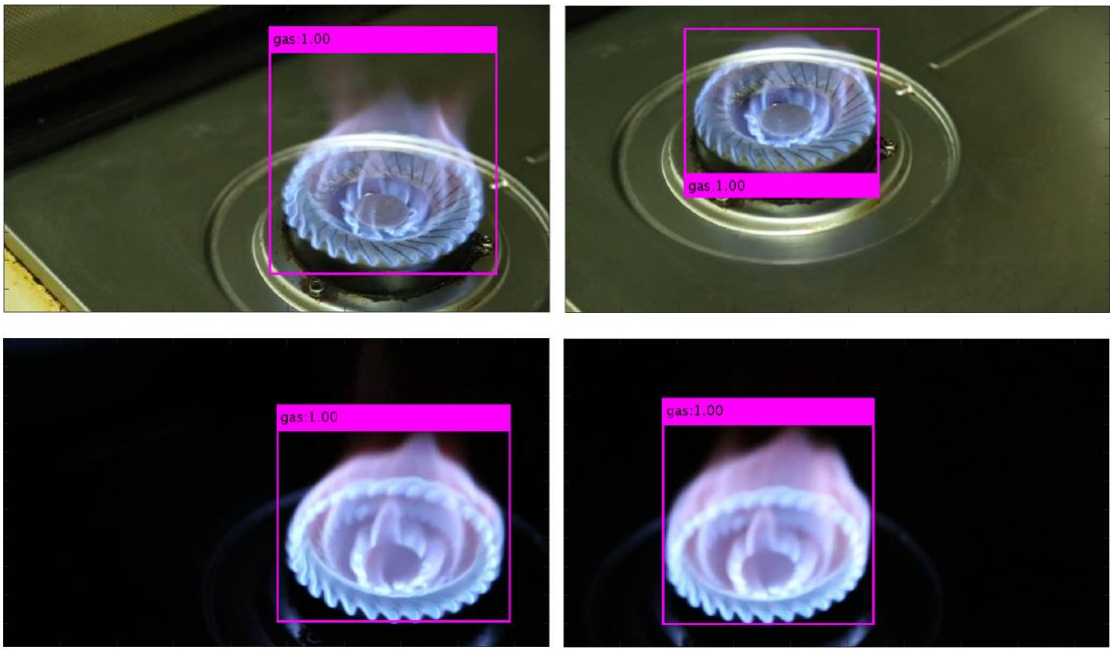


Figure 4.3 The Result of Gas Fire

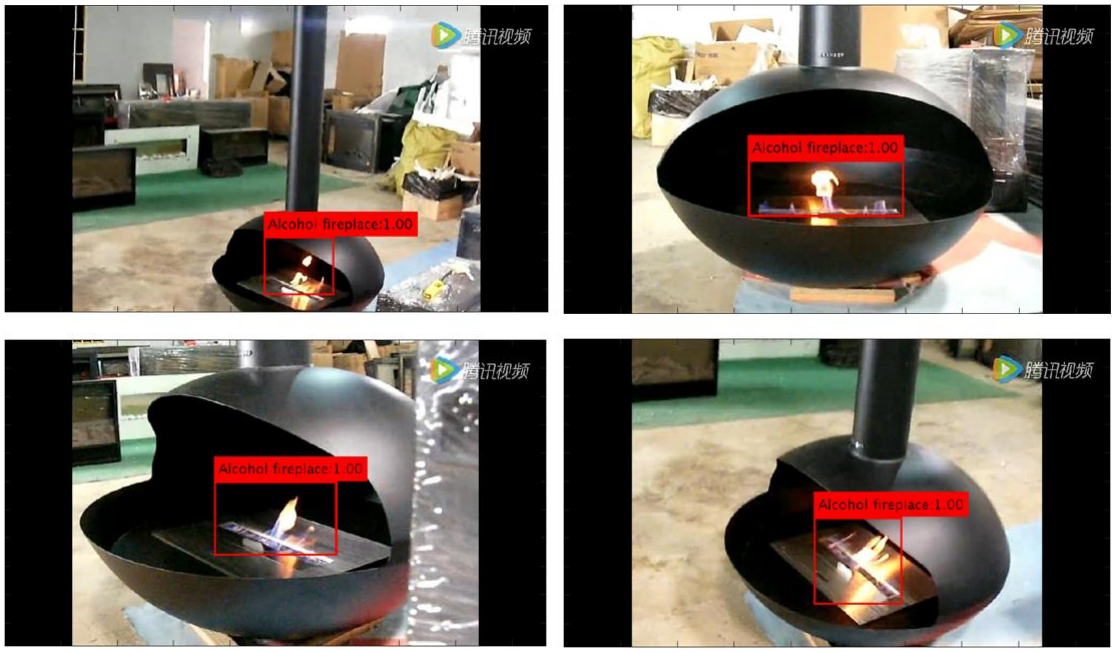


Figure 4.4 The Result of Alcohol Fireplace Fire

Figure 4.5 The Result of Fireplace Fire

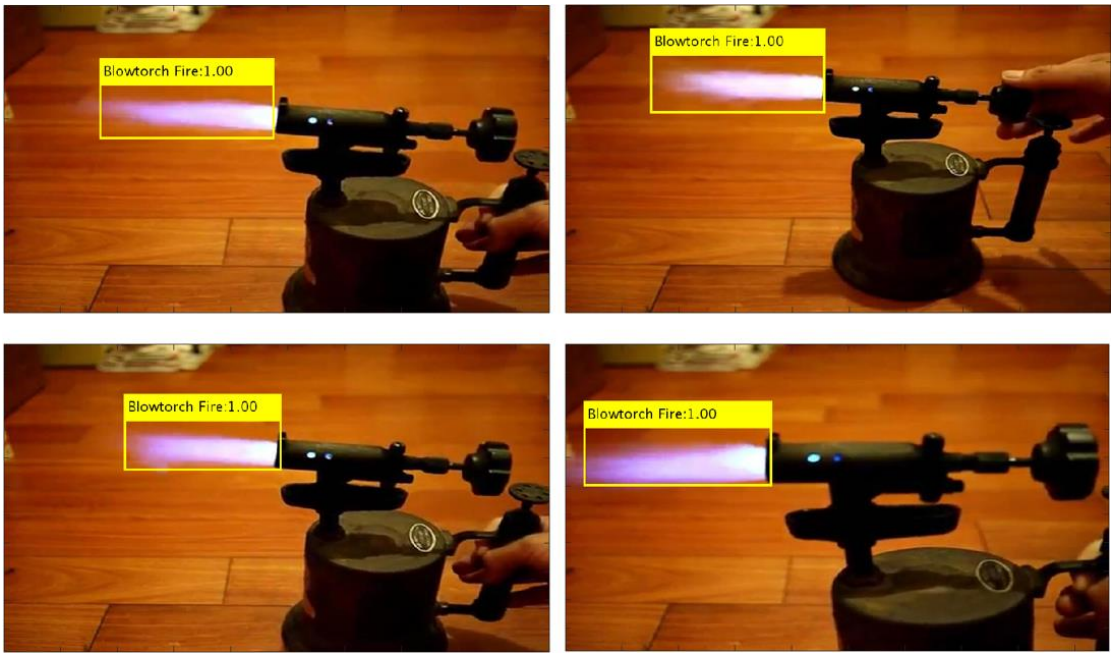
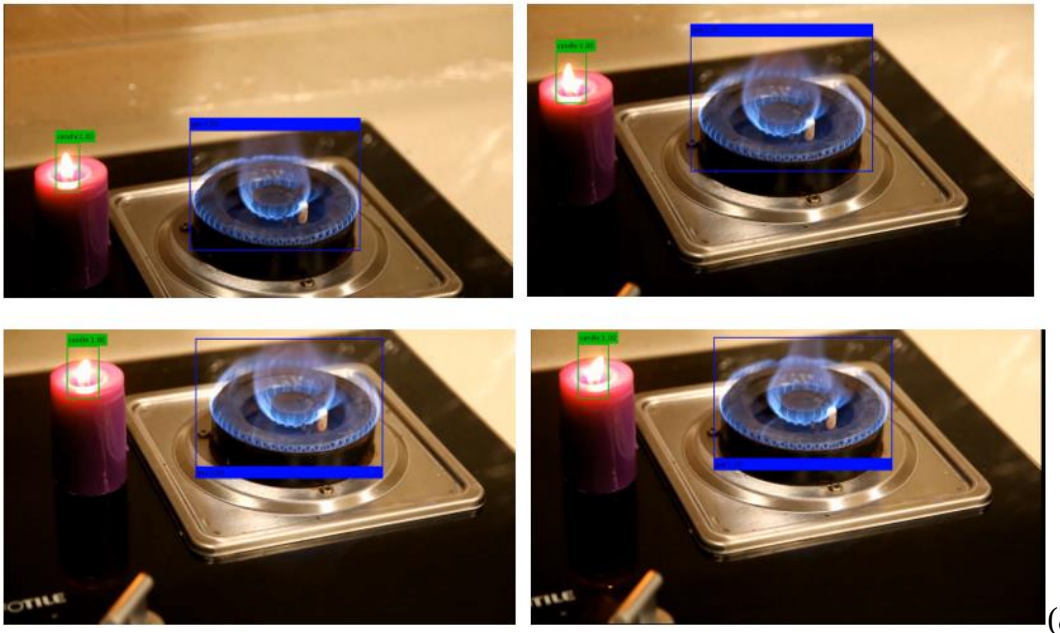


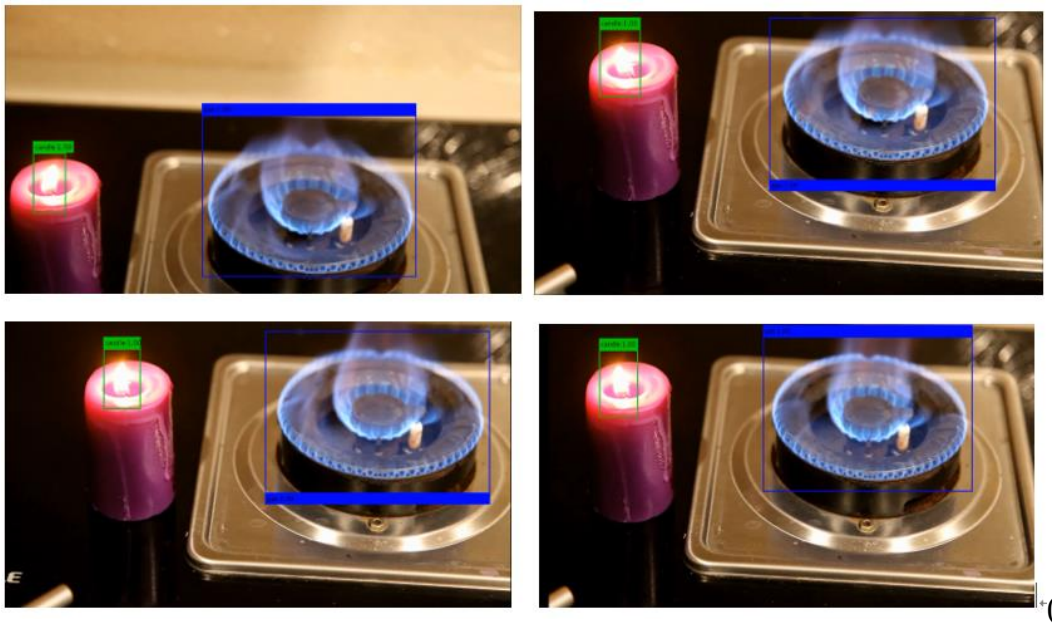
Figure 4.6 The Result of Blowtorch Fire

Figure 4.7 The Result of Gas Tank Fire

On the other hand, we attempted to simultaneously present two different kinds of flames from the same camera in an attempt to identify more than one flame at the same time. Here we chose gas fires and candle fires, and collected data as many angles as possible during shooting. Multiple angle shooting can simulate the flame image captured by using the camera in a real-world. After data enhancement through the same method which we introduced in Chapter 3, we used the self-designed MFD convolution neural network to train this set of data. Finally, we got a model with an accuracy of 93.6%. The following pictures are from two groups of our experiments.



(a)



(b)

Figure 4.8 The Result of Gas Fire and Candle Fire Recognition Together

The convolutional neural network has an entirely different way to process those images. Before running the neural networks, we do not need to extract features from the image manually. Therefore, compared to the traditional object recognition technology, CNN has a significant performance. From Figure 4.8, we see that the centers of the flame are almost the centers of the bounding box. The flames of different burning material are all sitting in the bounding box, except the detection result of a candle flame. When the

background is bright, the bounding box shows that the candle flame is not that accurate. Fortunately, from the last image of Figure 4.2, we find that CNN does not recognize the reflection of candle-light on the wall as a real flame.

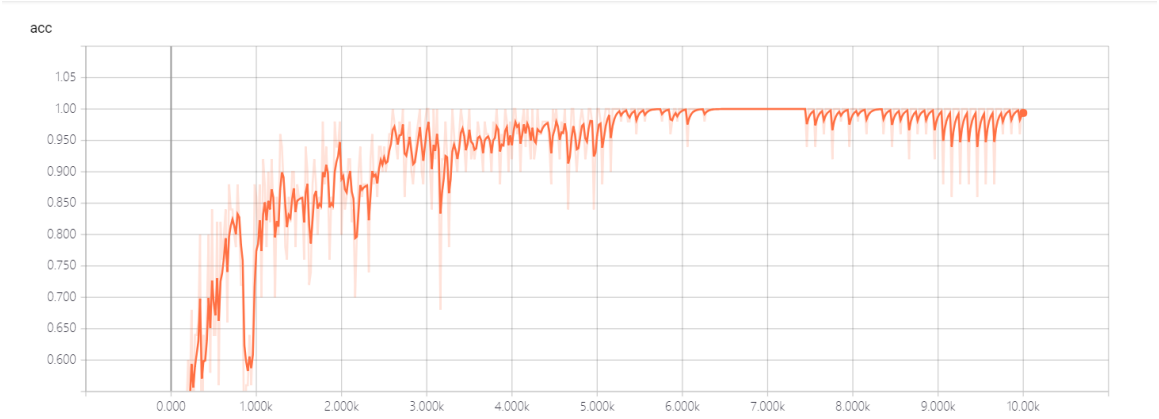


Figure 4.9 Graph of Accuracy

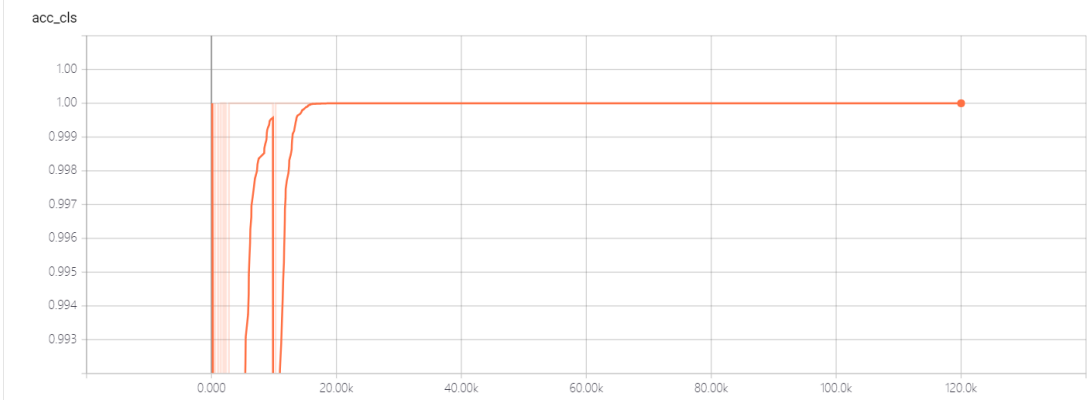


Figure 4.10 Graph of Accuracy of Classification

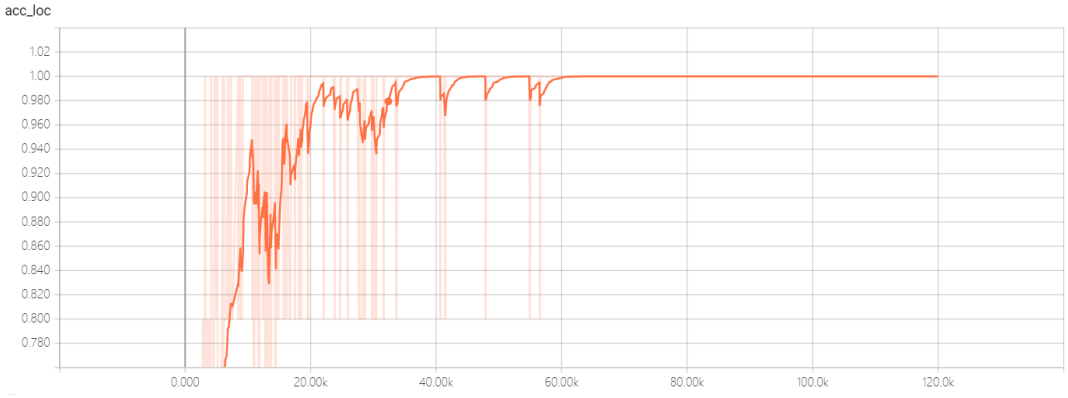


Figure 4.11 Graph of Accuracy of Location

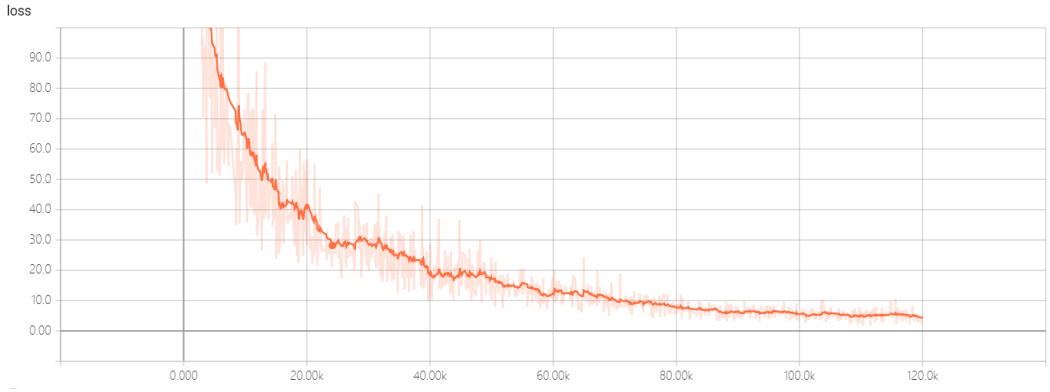


Figure 4.12 Graph of Loss Function

In Fig.4.10, after 10000 times of training, the accuracy of the final training reaches 100%. However, training began around the 7400th round, and the accuracy of the results varies from 96% to 100%. In general, the results show outstanding performance. Fig. 4.11 shows the accuracy of the classification. During the 10000 times training, the classification accuracy raises from 60% to 98% in the first 1000 times training and maintained at around 100% in the remaining time. Fig. 4.12 shows the accuracy of the bounding box location. It is similar to the accuracy of the result. Fig. 4.13 is the graph of the loss function, in the late period of training, it is close to 0. In short, the SSD gives us an excellent performance and a high accuracy.

After the best results, we will perform a more detailed analysis of the classification results for the optimal model. First, we classify six types of flame samples according to the area ratio of the bounding box that occupies the entire image. We divide it into three categories: small-area objects, medium-size objects, and large-area objects. The sample

size of various types of objects is shown in Table 4.3. Overall, the small-area objects are the most, the medium-sized objects are the second, and the large-area objects are the least. The candlelight and Blowtorch Fire have a small area of objects.

Table 4.3 Distribution of the Number of Different Types of Objects

Type	(0.00, 0.33]	(0.33, 0.67]	(0.67, 1.00]	Total
Alcohol Fireplace	1018	158	24	1200
Gas Tank Fire	1104	94	2	1200
Blowtorch Fire	1060	140	0	1200
Campfire	1200	0	0	1200
Candlelight	1200	0	0	1200
Gas	530	655	15	1200
Total	6112	1047	41	7200

If the positioning is greater than 0.5 by using IoU, the IoU is calculated as

$$\mathrm{IoU}(A,B) = \frac{S(A \cap B)}{S(A \cup B)} = \frac{S_{AB}}{S_A + S_B - S_{AB}} \tag{4.3}$$

Among them, S_A, S_B, S_{AB} are the area of *A*, *B*, and intersection of *AB*, respectively. The IoU value is closer to 1, the positioning accuracy will be higher. Because the accuracy rate does not intuitively reflect the accuracy of the positioning, we use the average IoU to compare different classes.

The positioning accuracy of the type object. The positioning accuracy of candle light is the lowest, and the average IoU is only minimal, the mean IoU of the fire, gas, and explosion is massive. In addition, according to the area ratio, the average IoU of the small area object is small, the medium area object and the large area object have considerable positioning accuracy.

Table 4.4 Different Types of Objects IoU Distribution

Type	**(0.00, 0.33]**	**(0.33, 0.67]**	**(0.67, 1.00]**	**Total**
Alcohol Fireplace	0.0.8985	0.9300	0.9071	0.9028
Gas Tank Fire	0.9146	0.0.9339	0.8711	0.9160
Blowtorch Fire	0.9226	0.9435	0	0.9251
Campfire	0.9494	0	0	0.9494
Candlelight	0.8834	0	0	0.8834
Gas	0.9475	0.9603	0.9716	0.9548
Total	0.9169	0.9511	0.9290	0.9219

Let's look at the detection accuracy of different types of objects. From Table 4.5, the classification accuracy of campfire and gas is the highest, because they have the smallest number of objects in a small area. The candle light has only a small area object, making it the lowest classification accuracy.

Table 4.5 Correct Rate Distribution of Different Types of Objects

Type	(0.00, 0.33]	(0.33, 0.67]	(0.67, 1.00]	Total
Alcohol Fireplace	0.9674	0.9773	0.9167	0.9675
Gas Tank Fire	0.9482	0.9694	1	0.9515
Blowtorch Fire	0.8772	1	0	0.9375
Camp fire	1	0	0	1
Candle light	0.9528	0	0	0.9528
Gas	1	1	1	1
Total	0.9936	0.9971	0.9512	0.9869

4.4 Limitations of the Experiment

The MFD model is successfully implemented for flame detection and recognition. However, there are still some limitations in our experiments that should be improved in future.

Frist, the test videos only have two types of flames; secondly, we only selected those types of fire which have entirely different color or shape. In future, we will try to recognize two flames that have a similar shape. Finally, only six types of fire were taken into account for flame detection in this thesis; in future, we will add more different kinds of flames into consideration.

Chapter 5
Analysis and Discussions

In this chapter, our goal is to analyze in detail the different effects of the different parameter settings and the corresponding accuracy differences. On the other hand, we have tried existing object detection method based on convolutional neural networks such as YOLO and R-FCN. We compared the differences in these methods and pointed out that at the same time we tested six different flame datasets and compared the results.

We have illustrated the immediate results of multiple flame recognition by training dataset with convolution neural network in the previous chapter. We display the result that the six categories of flames were recognized as accurate, and we also showed the successful demos which have two different types of fire at the same time. In this chapter, the resultant analysis based on the different parameters will be detailed. Moreover, we compare two existing object detection methods (YOLO and R-FCN) as well in this section by using the same dataset, and we will display the accuracy between these models.

5.1 MFD Models Analysis

As we mentioned in Chapter 4, we designed four kinds of bounding box algorithms which are different. Here we rename the model number 1, 2, 3 and 4 as MFD_1, MFD_2, MFD_3 and MFD_4. We use the NVIDIA GTX 950M graphics card for training, which requires 9 hours to train a model due to hardware capabilities and data set size limitations. After 36 hours of training, we finally obtained a comparison of the accuracy of the four models using different box encodings, as shown in Table 5.1.

Table 5.1 Correct Rate Comparison

Model	**Mean**	**Train**	**Valid**
MFD_1	0.993	1.000	0.957
MFD_2	0.826	0.861	0.654
MFD_3	0.805	0.844	0.610
MFD_4	0.866	0.901	0.692

We find that the highest correct rate was obtained with the model MFD_1, and the verification accuracy rate reaches 95.7%. The effect was satisfactory. However, the accuracy of models using other codes is less than 66%.

Now, we will analyze the training curves of the four models, as shown in Fig. 5.1~ 5.4. For better performance trends, we perform smoothing on all data and take logarithms of the loss function values.

The comparisons between the two are generally used to observe whether the model has overfitting and underfitting. If the model converges, there is no difference between the

two, indicating that there is no overfitting phenomenon. If there is a vast difference between the two, or if the loss function value of the verification data does not decrease, the overfitting phenomenon occurs. If the loss function of the training data does not converge, there is a downward trend, it indicates that the model has a phenomenon of underfitting, and it also needs to increase the number of training rounds to continue training. Among the four groups of images, we find that only the curve of model MFD_1 shows convergence, and the other three models are underfitting. We finally chose the model of code one as the optimal model, with an average accuracy of 98.2%.

1) The first group represents the model MFD_1, which achieves the highest accuracy for flame recognition. Fig.5.1(b) shows that after 120,000 steps of training, the average accuracy of the training phase is nearly 100%, and the accuracy of the verification part is also maintained at about 95%. Both the accuracy of the training phase and the accuracy of the verification phase started at a steep rise around 15,000 steps. Fig.5.1(a) reflects the values change in loss function throughout the training process. Both loss functions exhibit a very smooth convergence process. The loss function in the training time finally converged to around 0.6, while the loss function in the validation phase was finally kept at 1.5. The difference between these two values is not significant, indicating that there is not overfitting phenomenon. The image in Fig.5.1(c) represents the changes that the loss function shows in the process of locating and classifying objects. The lower right picture shows that the accuracy of classification and determination is 100%.

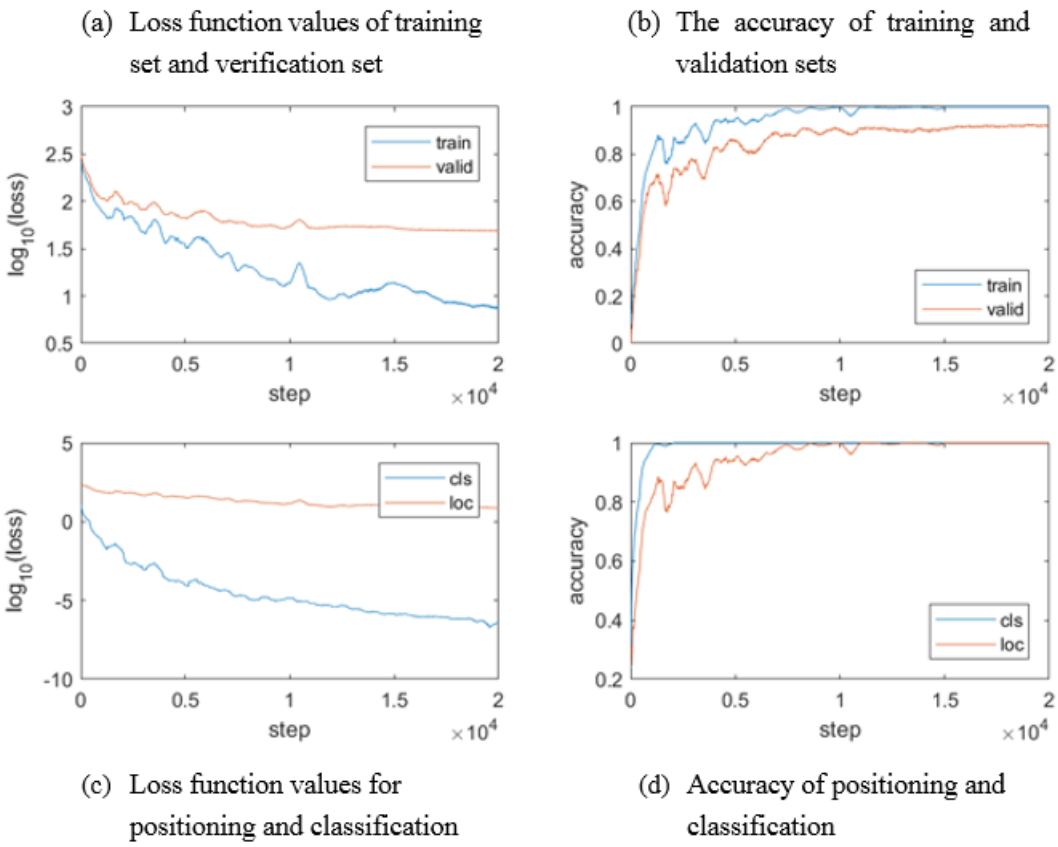


Figure 5.1 Training Curves of MFD_1

2) The second group graph represents the model MFD_2. The loss function of both training and validation decrease during the whole testing, and the loss function value of training decline much more than that of validation. Overall, they are in a convergence trend, but there is two slight growth in validation around 7500 steps and 17500 steps respectively. Regarding accuracy part, both the training accuracy rate and validation accuracy rate have fluctuant reached the highest value with 85% and 64% respectively at the last step. They also both display a drop at around 7500 steps which are connected with the increase in loss function. Also, the curve represents location declines stably in the loss function while the classification curve contains four growths between 7400 steps and 17500 steps.

Fig.5.2(a) shows the location accuracy and classification accuracy of MFD_2. We witness an increasing trend in location curve with slight decrease around 7500 steps, whereas classification accuracy reaches the peak point 100% at around 2000 step. Comparing MFD_1, the accuracy of the training phase reached the highest value at

the end, and it was not as smooth as in model MFD_1. Also, the accuracy of the validation reached only half of model MFD_1. Although the flame classification results of MFD_2 is as good as that in MFD_1, the location of the multiple flame result is very terrible.

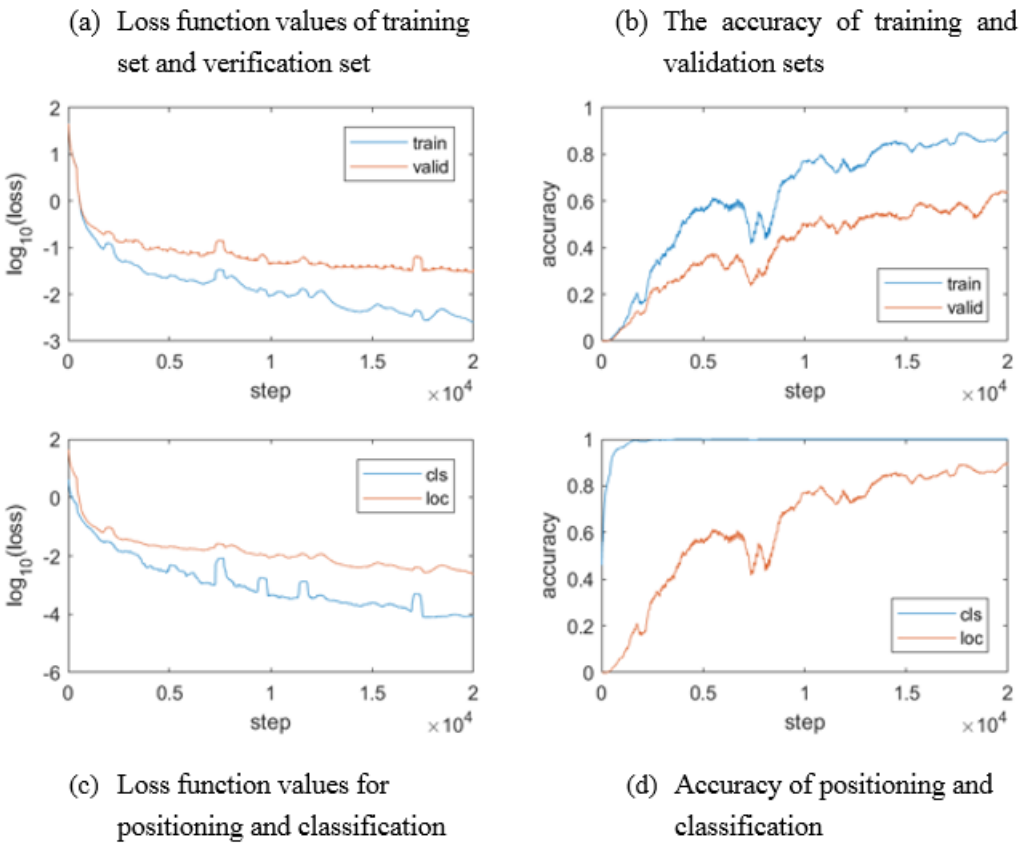


Figure 5.2 Training Curve of MFD_2

3) In the MFD_3, with the increase of the step, the loss functions of both training phase and validation phase show a trend of convergence during the whole period. However, from the upper left image, we can find both of them increase suddenly at around 6000 steps while the accuracy of them decline correspondingly. For the value of the loss function in the validation phase, it should be a steady downward trend, but the value of the loss function raised by one instead of declining. That means the whole model was overfitted during those stages.

Fig 5.2(b) shows the tendency of accuracy keep growing during the period, and the total tendency is similar to that in MFD_2. As for classification loss function and location loss function value, they show a similar situation like the loss function of

training and validation. The accuracy of location keeps raising and reach the highest point 90% at 2000 steps while classification's rate peak to 100% significantly early than location's rate at around 2500. In general, MFD_2 shows an underfitting phenomenon both in the training phase and the validation phase. Therefore, it is necessary to increase the number of training epochs for MFD_2 to continue training in order to obtain better results.

Figure 5.3 Training Curve of MFD_3.

4) The loss function of validation performs very well in the model MFD_4 which shown in Fig 5.4(a), the smooth polyline shows that MFD_4 does not exhibit underfitting or overfitting during the validation phase. Unfortunately, the loss function in the training phase is well-running until about 10,000 steps, and the value of loss function shows a decreasing trend. After 15,000 steps, the function value rose back to about two but then continued to decrease until the end of the training. That means that after 10,000 steps, the model training has under-fitting. From the lower left subgraph, we can find the loss functions for classification and location have the same situation as that of

training and validation. For the accuracy of the model, the accuracy changes are very different from the other three models. Between the 10000 steps and 15000 steps where the loss function value starts to decrease, both the training accuracy and the subsequent verification accuracy show an upward trend, when the value of the loss function rises, the accuracy decreases. The performance of this model is precisely the opposite of the that of the first three models. The classification accuracy rate is always higher than the accuracy rate of location, which the same phenomenon was shown in the four groups of models.

Figure 5.4 Training Curve of MFD_4

Overall, the values of the loss function for the validation phase and model location are always higher, while the accuracies of the training phase and model classification are always higher. In detail, all aspects of MFD_1 are the best among the four models. There are no underfitting or overfitting phenomena that determine the quality of the model. At the same time, it is not only the training phase and validation phase, but also the model positioning and model classification achieve the highest precision and the best

performance. Although the value of the positioning loss function of the MFD_1 is the highest of the four models, the performance of the whole process is very stable, and no significant changes occurred.

5.2 Model Comparisons

5.2.1 Structures of Three Methods

We have already introduced two object recognition YOLO and R-FCN, and now we attempt to put the structure of MFD with that of these two methods together to explore the differences between these methods.

- **MFD**

Our convolutional neural network has a total of 8 convolution layers and a max pooling layer followed by each convolution layer. After the Conv_8, we expand the CNN output and add the dropout operation to overcome the overfitting problem. The dropout strategy refers to output the data after randomly disabling some nodes. This can increase the fault-tolerance of the model, and the temporarily loose efficacy of some nodes will not affect the overall result of the model. After the Dropout layer, it is followed by a fully connected layer for object location and object recognition. The full connection layer contains nonlinear activation functions and loss functions.

From the Table 5.2 displayed above, we see that the size of the first convolution kernel is in size 5×5, the other convolution kernels are in size 3×3. The most massive pooled layer is unified with a 2×2 kernel. Our input image is in the size of 448×448 with three color channels, and finally, the output is in the size of 7×7.

Table 5.2 Details of MFD Convolutional Layers

Layer	Size	Output
Input Process		(448, 448, 3)
Con_1	5 × 5	
Max Pooling	2 × 2	(244, 244, 16)
Con_2	3 × 3	
Max Pooling	2 × 2	(112, 112, 32)
Con_3	3 × 3	
Max Pooling	2 × 2	(56, 56, 64)
Con_4	3 × 3	
Max Pooling	2 × 2	(28, 28, 128)
Con_5	3 × 3	
Max Pooling	2 × 2	(14, 14, 256)
Con_6	3 × 3	
Max Pooling	2 × 2	(7, 7, 512)
Con_7	3 × 3	
Max Pooling	2 × 2	(14, 14, 256)
Con_8	3 × 3	
Max Pooling	2 × 2	(7, 7, 512)
Dropout		
Fully Connect Layers		

- **YOLO**

The architecture of YOLO is also simple, it is just a convolution neural network which has nine convolution layers and six max-pooling layers. This neural network uses the standard layer types, except the last convolution layer has a 1×1 size convolution kernel, all the rest convolution kernel is in 3×3 size. Moreover, the pooling kernel is the same size as the kernel in our MFD model. However, there is no fully connected layer in YOLO. The last convolution layer with the 1×1 kernel aims to reduce the shape 13×13×125. So the image reshapes into the same size of the grid that the image divided into at the beginning of image processing. The final shape of the image ends up with 125 channels, the number refers to that each grid predicts five bounding boxes and each bounding box describes 25 classes from the dataset.

Table 5.3 Details of YOLO Convolutional Layers

Layer	Kernel	Output Shape
Input		(416, 416, 3)
Convolution	3 x 3	(416, 416, 16)
Max Pooling	2 x 2	(208, 208, 16)
Convolution	3 x 3	(208, 208, 32)
Max Pooling	2 x 2	(104, 104, 32)
Convolution	3 x 3	(104, 104, 64)
Max Pooling	2 x 2	(52, 52, 64)
Convolution	3 x 3	(52, 52, 128)
Max Pooling	2 x 2	(26, 26, 128)
Convolution	3 x 3	(26, 26, 256)
Max Pooling	2 x 2	(13, 13, 256)
Convolution	3 x 3	(13, 13, 512)
Max Pooling	2 x 2	(13, 13, 512)
Convolution	3 x 3	(13, 13, 1024)
Convolution	3 x 3	(13, 13, 1024)
Convolution	1 x 1	(13, 13, 125)

- **R-FCN**

The object detection task needs to define the specific location of the object , therefore neural network requires the translation transformation feature. The R-FCN (Region-based Fully Convolutional Network) is a framework based on a shared full convolutional structure for object detection just like FCN (Dai, Li, He & Sun, 2016). FCN does not achieve the balance between positional variability and positional variability when categorizing tasks. Therefore, in the R-CFN framework, position-sensitive score maps wer created to control the information of these score maps.

The network of this algorithm is mainly based on ResNet-101. ResNet-101 contains 100 convolutional layers, a mean downsampling layer, and a 1000-category fully connected layer. First, input the image processed through a full convolutional network ResNet-101. A particular convolutional layer was added to generate a position-sensitive score map. On the other hand, the output of this full convolutional network serves as the input to the RPN (Ren, He, Girshick & Sun, 2015) network which designed in advance, and the RPN network finally generates RoIs. The input of pooling layer will be the

previous position-sensitive score map and RoIs, and final pooling layer will output categories information. Also, the regression part and the classification part are parallel.

Table 5.4 Details of R-FCN Convolutional Layers

Layer	Kernel	Out shape
Input		
Conv1	7×7	(112, 112, 64)
Max Pooling	3×3	
Conv2_x	$\begin{pmatrix}1\times1\\3\times3\\1\times1\end{pmatrix}\times3$	(56, 56, 64)
Conv3_x	$\begin{pmatrix}1\times1\\3\times3\\1\times1\end{pmatrix}\times4$	(28, 28, 256)
Conv4_x	$\begin{pmatrix}1\times1\\3\times3\\1\times1\end{pmatrix}\times23$	(14, 14, 512)
Conv5_x	$\begin{pmatrix}1\times1\\3\times3\\1\times1\end{pmatrix}\times3$	(7, 7, 1024)
RPN	3 × 3	(1, 1024, 38, 63)
ROI pooling	7× 7	

Comparing these three kinds of object recognition patterns, we easily find that the R-FCN model framework is the most complex one. It is based on ResNet-101 plus position-sensitive score map and RoIs pooling layer to achieve the purpose of accurately identifying objects so that the framework increases steps for the image to operate in the network. On the other hand, though YOLO is very simple because of his structural design, compared with our own designed MFD, there are still three more convolutional layers than ours. The convolutional operations in our MFD network structure are simple, but it still exhibits high accuracy for detecting and recognizing the flame.

5.2.2 Training and Loss Function

Different object recognition frameworks will encounter bounding box regression problems when performing object recognition or detection. Usually, before the picture enters the convolution operation in neural networks, we will give the ground truth of the target to be identified in the picture. After several convolution operations and related processes, the neural network will give a Region Proposal. In the widespread sense,

Region Proposal is what the neural network considers to be a target object in the area. Since Region Proposal is not 100% correct, the target window generated by the model and the original Ground Truth box will have an overlap rate. We call this overlap rate as intersection over union. The regional proposals frame is not positioned correctly, so this map corresponds to the incorrect detection of the target object. If we fine-tune the region proposal box so that the trimmed box is closer to the Ground Truth box, this positioning will be more accurate.

- **MFD**

In our MFD model, we chose four coordinates to determine the ground truth of the flame, we got four coordinates as $[x_{min}, x_{max}, y_{min}, y_{max}]$. For the positioning problem of the flame, we treat it as a general regression problem. We set the output of Location Regression as f_{loc}, and the position of the bounding box of the training sample as *b*. We then use the loss function to calculate the difference between the two, and finally get our location loss function. Our design for the bounding box can be represented by four coordinates; *w* and *h* denote the width and height of the input image, (x_{min}, x_{max}) and (y_{min}, y_{max}) are the points of upper left corner and the lower right corner of the bounding box, respectively. We also indicate the coordinates of the center of the bounding box (x_c, y_c) as:

$$x_c = \frac{x_{min}+x_{max}}{2},\ y_c = \frac{y_{min}+y_{max}}{2} \tag{5.1}$$

And w_b, h_b respectively indicate the width and height of the bounding box, which satisfy:

$$w_b = x_{max} - x_{min},\ h_b = y_{max} - x_{min} \tag{5.2}$$

For the regression problem of flame positioning, we first need to obtain the f_{loc} output by location regression. At the same time, for the classification problem, we also set a probability vector $p = (p_1, p_2, \ldots, p_n)$ as the output of classification probability. The probability p_i indicates the possibility that the input image belongs to the *i-th* category.

For the training sample we also set *i* kinds of categories. If the training tag q_n belongs to the *n-th* category, then we got q_n=1. We used the Cross-Entropy as the Loss Function to solve object classification problem:

$$L_{cls} = -\sum_{k=1}^{n} q_k \log p_k \tag{5.3}$$

Finally, we get the final loss function. It is a linear function with a location loss function and a classification loss function:

$$L_{total} = \ell_{loc}(f_{loc} - b) + L_{cls} \tag{5.4}$$

- **YOLO**

YOLO is using another method to solve this regression problem. As we mentioned earlier, in the grid of *S*×*S* divided by YOLO, each grid will predict *B* bounding boxes. In addition to return each bounding box to its position, each bounding box also comes with a predictive confidence score. This confidence score represents the confidence that the predicted box contains the object and how accurate this box predicts the double information. The value is calculated as:

$$p_r(Object) * IOU_{pred}^{truth} \tag{5.5}$$

If there is an object in a grid cell, the $p_r(Object)$ will be 1; otherwise, it will take 0. The second item is the IoU value between the predicted bounding box and the actual ground truth box. Each bounding box predicts a total of five values which are *x, y, w, h* and confidence. Each grid also predicts a category information, which is denoted as class *C*. Then each grid in *S*×*S* not only predicts *B* bounding boxes, but also to predict *C* categories. At the end. the output will be a tensor value which in shap of *S*×*S*×(5×*B*+*C*).

In test, the class-specific confidence score of each bounding box is obtained by multiplying the predicted class information of each grid and the confidence information predicted by using the bounding box:

$$p_r(Class_i) * IOU_{pred}^{truth} = p_r(Class_i|Object) * p_r(Object) * IOU_{pred}^{truth} \tag{5.6}$$

The $p_r(Class_i|Object)$ on the right side of equation (5.6) is the category information predicted by each grid. The $p_r(Object) * IOU_{pred}^{truth}$ refers to the confidence of each bounding box's prediction. This encodes the probability that the predicted box belongs to a certain class, as well as the accuracy of the box. After getting the class-specific confidence score of each box, we set a threshold to filter out the low score boxes and get the final testing result.

The design goal of the loss function is to achieve a good balance of coordinates, confidence and category. Merely using Sum-Squared Error Loss will have the following drawbacks: 1) The 8-dimensional localization error and the 20-dimensional classification error should have precedence; 2) If there is no object in a grid, the confidence of the boxes in these grids will be pushed to 0. Compared to fewer grids with the object, this approach is overpowering which will cause the network to not stable or even divergent. So the loss function is as follows:

$$
\begin{aligned}
Loss_{YOLO} = \lambda\text{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \\
+ \lambda\text{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \\
+ \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} (c_i - \hat{c}_i)^2 \\
+ \lambda\text{noobj} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{noobj} (c_i - \hat{c}_i)^2 \\
+ \sum_{i=0}^{s^2} 1_{ij}^{obj} \sum_{j=0}^{B} (p_i(c) - \hat{p}_i(c))^2 \qquad (5.7)
\end{aligned}
$$

- **R-FCN**

The network of this algorithm is mainly based on ResNet-101. ResNet-101 contains 100 convolutional layers, a mean downsampling layer, and a 1000-category fully connected layer. The last convolutional output of ResNet-101 is 2048-dimensional, where

the author added a 1024-dimensional 1×1 convolutional layer for dimensionality reduction. On the one hand, a $k^2 \times (C+1)$-dimensional convolutional layer is added to generate the score maps. These score maps are mainly used to generate the object's category. On the other hand, the sum was added to perform the bounding box regression. The Fast-RCNN is similar to the bounding box regression convolutional layer with a dimension of $4 \times k^2$. This layer is in parallel with the convolutional layer generated the score maps. In addition to this main network, the algorithm also introduces RPN networks to generate RoIs. The generated RoIs are pooled with the score maps generated by using the classified convolutional layers and finally the probability that each RoI belongs to (a total of $C + 1$ classes). In addition, this RoI will be pooled with the output of the regression convolutional layer to obtain the four coordinates of each RoI. The loss function is as same as Fast-RCNN.

The output of target positioning is similar to the classification output process, except that the dimension is no longer $k^2 \times (C+1)$ but $k^2 \times 4$, representing 4 offset coordinates of 9 small regions [*dx,dy,dw,dh*]. For the $w \times h$ RoI region, the size of each box is about $w/k \times h/k$. The last convolution layer consists of k^2 score maps. For the box of the *i-th* row and *j-th* column ($0 \leq i, j \leq k-1$), the calculation formula of the score map is:

$$k(i_J i|\Theta) = \sum_{(x,y)\in(i,j)} z_{i,j,c}(x + x_0, y + y_0|\Theta)/n \tag{5.8}$$

The end-to-end training of R-FCN structure is elementary by using pre-calculated region proposals. Next, the definition of our loss function consists of two parts: cross-entropy loss and box regression loss

$$L(s, t_{x,y,w,h}) = -log(s_{c^*}) + \lambda[c^* > 0]L_{reg}(t, t^*) \tag{5.9}$$

In summary, we see that YOLO with a relatively simple frame structure has a more complex method for calculating loss function. Because of the special model structure of R-FCN, it is necessary to calculate the confidence of position-sensitive score maps first; then through the pooling operation, and finally perform the loss function calculation. The following three charts show the loss functions for the three models, respectively.

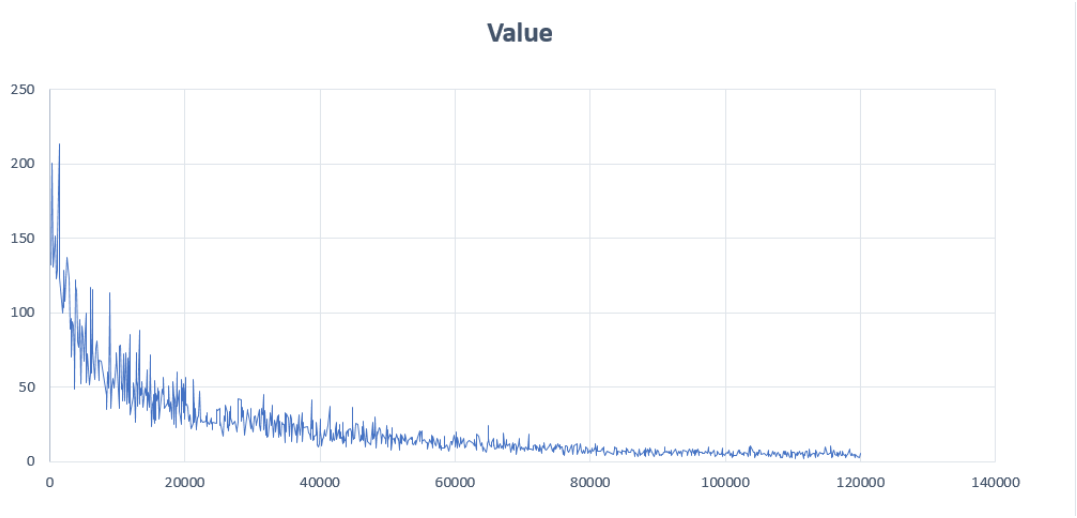


Figure 5.5 Accuracy of Flame Recognition From MFD_1

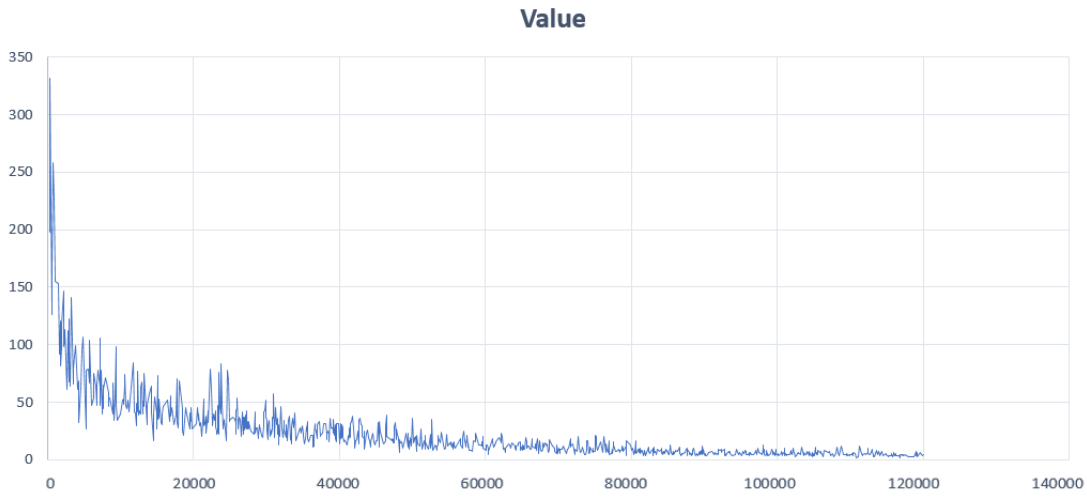


Figure 5.6 Accuracy of Flame Recognition From YOLO

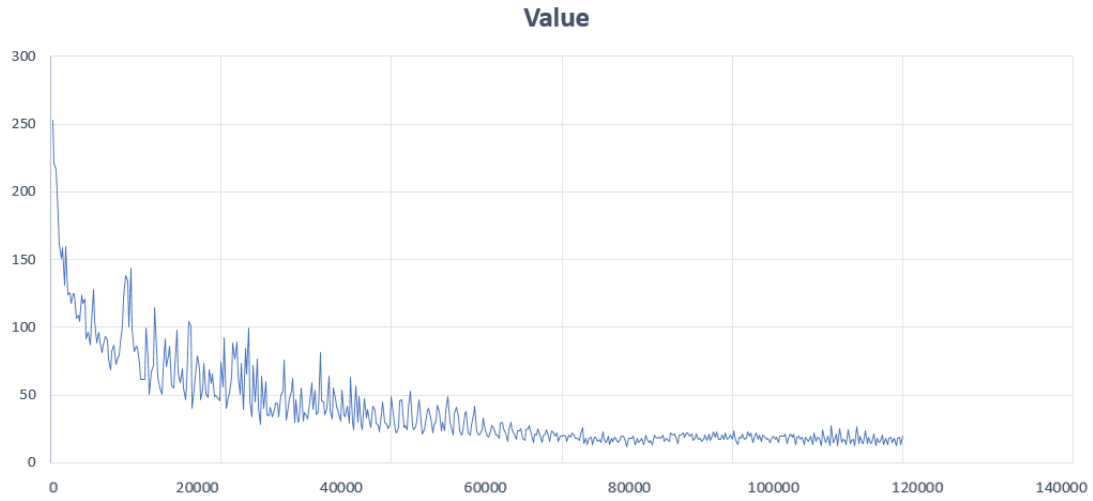


Figure 5.7 Accuracy of Flame Recognition From R-FCN

From the results, the loss functions of these three models have been well calculated in training. The loss functions of MFD and YOLO all showed significant changes at the beginning of the training, and the RFCN was steadily declined from the beginning. All three models reach their minimum after the same time. The difference is that the loss functions of MFD and YOLO eventually stay below 10, while the RFCN loss function converges to 25.

5.2.3 Activate Function

- **MFD and YOLO**

The nature of this convolution is a linear operation. In order to allow the neural network to fit the nonlinear system, the result needs to be nonlinearized through a nonlinear activation function after the convolution operation. So we added Leaky ReLU as an activation function after eight convolutional layers.

$$\text{LeakyReLU}(x) = \phi(x) = \begin{cases} x & x \geq 0 \\ 0.1 \cdot x & x < 0 \end{cases} \tag{5.10}$$

Compared to Sigmoid function, it avoids the problem of gradient varnashing during training. Compared to ReLU, it avoids the problem of insufficient node training. We chose Leaky ReLU as the activation function.

Later in the classification operation layer, since this part of the structure is a fully connected neural network layer, we also need to add an activation function to the output of this neural network layer, where we have also chosen LeakyReLU.

YOLO contains a detection network with 24 convolutional layers which are the first 20 convolutional layers, and an average-pooling layer and full-connection layers. For such a network structure, YOLO uses a linear activation function at the last layer, while other layers use the same leaky ReLU activation function as MFD.

- **R-FCN**

As mentioned earlier, the whole R-FCN convolution layer is divided into several large convolution layer groups. After each set of convolution layers, RFCN chose the ReLU formula as an activation function.

$$\mathrm{ReLU}(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{5.11}$$

However, ReLU also has shortcomings. ReLU forced sparse processing to reduce the sufficient capacity of the model. (There are too many features masked, resulting in that the model can not learn the effective features.) When the input value of this activation function is less than 0, the gradient value of the activation function is set to zero. Any data may no longer activate this zeroed neuron, we call it neurosis. As a variant of ReLU, LeakyReLU has the advantage of not being overfitted, the calculation is simple and efficient, and faster than other activation function bracelets.

5.2.4 Results and Discussion

After having been tested on these model frames with the same dataset, we have obtained the results in the following table. We see that in these results, our MFD_1 got the best performance regarding overall accuracy. Since our tests were performed on an NVIDIA GTX 950M GPU, and MFD_1 performed well, we tried to conduct the same training on the CPU. The accuracy of these results displayed is 93.5% which also performed well, but the time used on CPU is almost doubled on the GPU, which significantly reduces the

training efficiency. YOLO and R-FCN achieved 93.1% and 95.4% accuracy on the training results respectively, though they did not exceed our designed MFD_1 model. Regarding time-consuming, we see from the table that using the same hardware and dataset to train the model, YOLO only spent 20 hours, which is the fastest of all models. From the perspective of individual flame types, the accuracies of Campfire, Gas Stove Fire and Gas Tank Fire are the highest on MFD_1, and the Candle light performs at the best on the MFD_2. At the same time, R-FCN has the best performance in the identification of Blowtorch Fire and Alcohol Fireplace flames.

Table 5.5 Accuracy of Flame Types from Three Models

Name	Time	Hardware	Campfire	Candle Fire	Gas Oven Fire	Blowtorch Fire	Gas Tank Fire	Alcohol Fireplace	Total Accuracy
MFD_1	23hours	GPU	**100**	90.2	**100**	93.7	**95.1**	95.3	**98.6**
MFD_1	48hours	CPU	98.7	**95.3**	97.9	83.5	89.3	92.4	93.5
MFD_2	24hours	GPU	89.8	80.7	88.3	79.8	81.5	82.6	82.6
MFD_3	25hours	GPU	81.9	76.3	78.6	69.9	72.3	77.2	80.5
MFD_4	25hours	GPU	87.3	81.5	84.9	77.4	78.7	83.9	86.6
YOLO	**20hours**	GPU	95	88.9	94.9	87.3	89.6	89.3	93.1
R-FCN	23hours	GPU	97.3	91.9	89.7	**95.5**	92.6	**95.6**	95.4

In general, the MFD_1 model performs best in these object recognition frameworks. The excellent performance is not only for the part of accuracy but also in recognition of a single category. Whether it is a large-area object or a small-area object, the MFD model can handle perfectly. In comparison, YOLO does not perform well for tiny groups because it predicts only two boxes in a grid and only falls into one category. For the test images, the new and unusual aspect ratio for the same class of objects appears, and other situations are. Due to the problem of loss function, positioning error is the main reason that affects the detection efficiency. Especially in the treatment of large and small objects, it needs to be strengthened. Because of the complex structure, RFCN performs better than YOLO in flame categories recognition.

Chapter 6
Conclusion and Future Work

In this thesis, an in-depth articulation of the flame detection is detailed. We demonstrated the innovation in research outcomes. In the final stage, we conducted experiments to test the performance of our developed system when the system is running in different environments. In this chapter, we will summarize the entire research project and look ahead the future work

6.1 Conclusion

With the rapid increase in urban population and the rapid development of urbanization, especially for those highly-rise buildings and large space places, the prevention and detection of fire become more and more urgent. The traditional fire detection technology cannot effectively play the role of existing fire detection equipment in places with abundant space and outdoor environment. Under this circumstance, fire alarming technology based on digital image processing as a new type of useful fire detection technology, will surely attract more and more attention from people, and it will undoubtedly have a wide spectrum of applications.

Based on the image processing, this thesis uses the convolutional neural network in deep learning to train a useful model so as to identify and detect the flame automatically. This thesis introduces the different parts of convolutional neural networks and its functions, and also describes the current use of this network. The neural network extracts high-level flame abstraction features from image data.

In the experimental stage, we refer to the existing object recognition algorithm and design a relatively simplified neural network model to achieve our goal of identification. We collected six different types of flame datasets to enrich the experimental dataset as much as possible. In the end, we got a 96% recognition accuracy. Also, we analyzed the flame area size and its corresponding accuracy changes, from which we found that when the flame occupies a large area of the image, the recognition accuracy will be significantly improved.

At the end of the thesis, the algorithm is carefully tested and analyzed. We also tried to apply other existing recognition algorithms to our dataset and compare the results with our model results. The experiments prove that the flame detection algorithm presented in this thesis has a faster response speed and higher anti-interference, accuracy, and practicality.

6.2 Future Work

Our future work includes,

(1) We will work for detection and recognition of multiple flames, more complex flames such as forest fire, building fire and so on will be taken into consideration.

(2) Due to the complex and diverse environment in real scenarios, there will be more disturbances. In this thesis, there is no interference and the testing environment is relatively simple. In future, we will study the dynamic characteristics of video sources such as incandescent lamps, fluorescent lamps, and fires.

(3) In future, we will present fire information timely. Based on the information we set in advance, if a flame is identified, whether the flame will be at a risk of causing a big fire will be alarmed.

(4) In future, other feature extraction techniques should be taken into account to investigate flame categories because the patterns of different fires need accurate and precise description.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., & Kudlur, M. (2016). TensorFlow: A System for Large-Scale Machine Learning. In OSDI (Vol. 16, pp. 265-283).

Abdel-Hamid, O., Mohamed, A. R., Jiang, H., Deng, L., Penn, G., & Yu, D. (2014). Convolutional neural networks for speech recognition. IEEE/ACM Transactions on audio, speech, and language processing, 22(10), 1533-1545.

Abdel-Rahman, E. M., Mutanga, O., Adam, E., & Ismail, R. (2014). Detecting Sirex noctilio grey-attacked and lightning-struck pine trees using airborne hyperspectral data, random forest and support vector machines classifiers. ISPRS Journal of Photogrammetry and Remote Sensing, (88), 48-59.

Ackley, D. H., Hinton, G. E., & Sejnowski, T. J. (1987). A learning algorithm for Boltzmann machines. In Readings in Computer Vision (pp. 522-533).

Albu, R. D. (2009). Human face recognition using convolutional neural networks. Journal of Electrical and Electronics Engineering, (2), 110.

Al-Jawfi, R. (2009). Handwriting Arabic character recognition LeNet using neural network. Int. Arab J. Inf. Technol., 6(3), 304-309.

Anderson, P. W., Thouless, D. J., Abrahams, E., & Fisher, D. S. (1980). New method for a scaling theory of localization. Physical Review B, 22(8), 3519.

Andrews, P. L., Loftsgaarden, D. O., & Bradshaw, L. S. (2003). Evaluation of fire danger rating indexes using logistic regression and percentile analysis. International Journal of Wildland Fire, 12(2), 213-226.

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

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. Journal of machine learning research, 3, 1137-1155.

Cappellini, V., Mattii, L., & Mecocci, A. (1989). An intelligent system for automatic fire detection in forests. In Recent issues in pattern analysis and recognition (pp. 351-364).

Celik, T., & Demirel, H. (2009). Fire detection in video sequences using a generic color model. Fire Safety Journal, 44(2), 147-158.

Celik, T., Demirel, H., & Ozkaramanli, H. (2006). Automatic fire detection in video sequences. In Signal Processing Conference (pp. 1-5).

Celik, T., Demirel, H., Ozkaramanli, H., & Uyguroglu, M. (2007). Fire detection using statistical color model in video sequences. Journal of Visual Communication and Image Representation, 18(2), 176-185.

Çelik, T., Özkaramanlı, H., & Demirel, H. (2007). Fire and smoke detection without sensors: Image processing-based approach. In Signal Processing Conference (pp. 1794-1798).

Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2018). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. IEEE Transactions on Pattern Analysis and Machine Intelligence, 40(4), 834-848.

Chen, S., Wang, H., Xu, F., & Jin, Y. Q. (2016). Target classification using the deep convolutional networks for SAR images. IEEE Transactions on Geoscience and Remote Sensing, 54(8), 4806-4817.

Chen, T. H., Kao, C. L., & Chang, S. M. (2003). An intelligent real-time fire-detection method based on video processing. In International Carnahan Conference on Security Technology (pp. 104-111).

Chen, T. H., Wu, P. H., & Chiou, Y. C. (2004). An early fire-detection method based on image processing. In International Conference on Image Processing (Vol. 3, pp. 1707-1710).

Chen, W., Shi, Y. Q., & Xuan, G. (2007). Identifying computer graphics using HSV color model and statistical moments of characteristic functions. In IEEE International Conference on Multimedia and Expo (pp. 1123-1126).

Chen, Y. N., Han, C. C., Wang, C. T., Jeng, B. S., & Fan, K. C. (2006). The application of a convolution neural network on face and license plate detection. In International Conference on Pattern Recognition (Vol. 3, pp. 552-555).

Cigada, A., Ruggieri, D., & Zappa, E. (2005). Road and railway tunnel fire hazard: a new measurement method for risk assessment and improvement of transit safety. In Measurement Systems for Homeland Security, Contraband Detection and Personal Safety Workshop (pp. 89-94).

Collobert, R., & Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In International Conference on Machine Learning (pp. 160-167).

Dai, J., He, K., & Sun, J. (2016). Instance-aware semantic segmentation via multi-task network cascades. In IEEE Conference on Computer Vision and Pattern Recognition (pp. 3150-3158).

Dai, J., Li, Y., He, K., & Sun, J. (2016). R-FCN: Object detection via region-based fully convolutional networks. In Advances in neural information processing systems (pp. 379-387).

Deng, L. (2012). The MNIST database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine, 29(6), 141-142.

Deng, L., Li, J., Huang, J. T., Yao, K., Yu, D., Seide, F., & Gong, Y. (2013). Recent advances in deep learning for speech research at Microsoft. In IEEE International

Conference on Acoustics, Speech and Signal Processing (pp. 8604-8608).

Fereres, S., Lautenberger, C., Fernandez-Pello, C., Urban, D., & Ruff, G. (2011). Mass flux at ignition in reduced pressure environments. Combustion and flame, 158(7), 1301-1306

Fischer, A., & Igel, C. (2012). An introduction to restricted Boltzmann machines. In Iberoamerican Congress on Pattern Recognition (pp. 14-36).

Foo, S. Y. (2000). A machine vision approach to detect and categorize hydrocarbon fires in aircraft dry bays and engine compartments. IEEE Transactions on Industry Applications, 36(2), 459-466.

Garcia, C., & Delakis, M. (2004). Convolutional face finder: A neural architecture for fast and robust face detection. IEEE Transactions on PAMI, 26(11), 1408-1423.

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

Glorot, X., Bordes, A., & Bengio, Y. (2011). Domain adaptation for large-scale sentiment classification: A deep learning approach. In International Conference on Machine Learning (ICML-11) (pp. 513-520).

Gutmacher, D., Hoefer, U., & Wöllenstein, J. (2012). Gas sensor technologies for fire detection. Sensors and Actuators B: Chemical, 175, 40-45.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In IEEE CVPR (pp. 770-778).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Identity mappings in deep residual networks. In European Conference on Computer Vision (pp. 630-645).

Healey, G., Slater, D., Lin, T., Drda, B., & Goedeke, A. D. (1993). A system for real-time fire detection. In Computer Vision and Pattern Recognition (pp. 605-606).

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A. R., Jaitly, N., & Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal processing magazine, 29(6), 82-97.

Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. Neural Computation, 18(7), 1527-1554.

Hopfield, J. J. (1988). Artificial neural networks. IEEE Circuits and Devices Magazine, 4(5), 3-10.

Horng, W. B., Peng, J. W., & Chen, C. Y. (2005). A new image-based real-time flame detection method using color analysis. In Networking, Sensing and Control (pp. 100-105).

Huang, G., Liu, Z., Weinberger, K. Q., & van der Maaten, L. (2017). Densely connected convolutional networks. In IEEE CVPR (Vol. 1, No. 2, p. 3).

Ji, S., Xu, W., Yang, M., & Yu, K. (2013). 3D convolutional neural networks for human action recognition. IEEE Transactions on PAMI, 35(1), 221-231.

Jiao, Y., Weir, J., & Yan, W. (2011). Flame Detection in Surveillance. Journal of Multimedia, 6(1): 22-32.

Jin, L., Huang, J., Yin, J., & He, Q. (2000). Deformation transformation for handwritten Chinese character shape correction. In Advances in Multimodal Interfaces—ICMI 2000 (pp. 450-457).

Koga, K., Inobe, T., Namai, T., & Kaneko, Y. (1997). Integrated traffic flow monitoring system in a large-scale tunnel. In IEEE Conference on Intelligent Transportation System (pp. 165-170).

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In Advances in neural information processing

systems (pp. 1097-1105).

Krüll, W., Willms, I., Zakrzewski, R. R., Sadok, M., Shirer, J., & Zeliff, B. (2006). Design and test methods for a video-based cargo fire verification system for commercial aircraft. Fire Safety Journal, 41(4), 290-300.

Kumar, T., & Verma, K. (2010). A Theory Based on Conversion of RGB image to Gray image. International Journal of Computer Applications, 7(2), 7-10.

Lauer, F., Suen, C. Y., & Bloch, G. (2007). A trainable feature extractor for handwritten digit recognition. Pattern Recognition, 40(6), 1816-1824.

Lawrence, S., Giles, C. L., Tsoi, A. C., & Back, A. D. (1997). Face recognition: A convolutional neural-network approach. IEEE Transactions on Neural Networks, 8(1), 98-113.

Lazebnik, S., Schmid, C., & Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In IEEE CVPR (Vol. 2, pp. 2169-2178).

Le, Q. V., Ngiam, J., Coates, A., Lahiri, A., Prochnow, B., & Ng, A. Y. (2011). On optimization methods for deep learning. In International Conference on International Conference on Machine Learning (pp. 265-272).

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436.

LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., & Jackel, L. D. (1990). Handwritten digit recognition with a back-propagation network. In Advances in neural information processing systems (pp. 396-404).

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4), 541-551.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied

to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

LeCun, Y., Jackel, L. D., Bottou, L., Cortes, C., Denker, J. S., Drucker, H., & Vapnik, V. (1995). Learning algorithms for classification: A comparison on handwritten digit recognition. Neural networks: the statistical mechanics perspective, 261, 276.

Lei, B., Zhang, Z., & Wang, C. (2012). Video fire detection based on three-state Markov modal and fractal dimension calculation. In Optoelectronic Imaging and Multimedia Technology II (Vol. 8558, p. 85580B).

Li, J., Fong, N. K., Chow, W. K., Wong, L. T., Lu, P., & Xu, D. G. (2004). The motion analysis of fire video images based on moment features and flicker frequency. Journal of Marine Science and Application, 3(1), 81-86.

Liu, C. L., Jaeger, S., & Nakagawa, M. (2004). Online recognition of Chinese characters: the state-of-the-art. IEEE Transactions on PAMI, 26(2), 198-213.

Liu, C. L., Yin, F., Wang, D. H., & Wang, Q. F. (2010). Chinese handwriting recognition contest 2010. In Chinese Conference on Pattern Recognition (CCPR) (pp. 1-5).

Liu, C. L., Yin, F., Wang, D. H., & Wang, Q. F. (2013). Online and offline handwritten Chinese character recognition: benchmarking on new databases. Pattern Recognition, 46(1), 155-162.

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

Long, T., & Jin, L. (2008). Building compact MQDF classifier for large character set recognition by subspace distribution sharing. Pattern Recognition, 41(9), 2916-2925.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60(2), 91-110.

Lu, T. F., Peng, C. Y., Horng, W. B., & Peng, J. W. (2006, August). Flame feature model development and its application to flame detection. In International Conference on Innovative Computing, Information and Control (Vol. 1, pp. 158-161).

Luo, R. C., Su, K. L., & Tsai, K. H. (2002). Fire detection and isolation for intelligent building system using adaptive sensory fusion method. In IEEE International Conference on Robotics and Automation (Vol. 2, pp. 1777-1781).

Marbach, G., Loepfe, M., & Brupbacher, T. (2006). An image processing technique for fire detection in video images. Fire Safety Journal, 41(4), 285-289.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 5(4), 115-133.

Miao, L., & Wang, A. (2013). Video flame detection algorithm based on region growing. In International Congress on Image and Signal Processing (CISP) (Vol. 2, pp. 1014-1018).

Minsky, M., & Papert, S. (1969). The perceptron: Principles of computational geometry. MIT press. McCulloch, WS and Pitts, W., A Logical Calculus of the Ideas Imminent in Nervous Activity, Bulletin of Mathematical Biophysics, 5, 115-133.

Noda, S., & Ueda, K. (1994). Fire detection in tunnels using an image processing method. In Vehicle Navigation and Information Systems Conference (pp. 57-62).

Phillips, W., Shah, M., & Lobo, N. D. V. (2000). Flame recognition in video. In IEEE Workshop on Applications of Computer Vision (pp. 224-229).

Pinheiro, P. H., & Collobert, R. (2014). Recurrent convolutional neural networks for scene labeling. International Conference on Machine Learning (ICML) (No. EPFL-CONF-199822).

Poultney, C., Chopra, S., & Cun, Y. L. (2007). Efficient learning of sparse representations with an energy-based model. In Advances in neural information processing

systems (pp. 1137-1144).

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems (pp. 91-99).

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., & Berg, A. C. (2015). Imagenet large scale visual recognition challenge. International Journal of Computer Vision, 115(3), 211-252.

Sainath, T. N., Mohamed, A. R., Kingsbury, B., & Ramabhadran, B. (2013). Deep convolutional neural networks for LVCSR. In IEEE International Conference on ICASSP (pp. 8614-8618).

Sapna, S., Tamilarasi, A., & Kumar, M. P. (2012). Backpropagation learning algorithm based on Levenberg Marquardt Algorithm. Computer Science Informatoin Technol (CS and IT), 2, 393-398.

Sarikaya, R., Hinton, G. E., & Deoras, A. (2014). Application of deep belief networks for natural language understanding. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 22(4), 778-784.

Serre, T., Wolf, L., Bileschi, S., Riesenhuber, M., & Poggio, T. (2007). Robust object recognition with cortex-like mechanisms. IEEE Transactions on PAMI, 29(3), 411-426.

Shen, D., Chen, X., Nguyen, M., & Yan, W. Q. (2018). Flame detection using deep learning. In 4th International Conference on Control, Automation and Robotics (ICCAR)(pp. 416-420).

Simard, P. Y., Steinkraus, D., & Platt, J. C. (2003). Best practices for convolutional neural networks applied to visual document analysis. In ICDAR (Vol. 3, pp. 958-962).

Sivic, J., & Zisserman, A. (2003). Video Google: A text retrieval approach to object

matching in videos. In ICCV2003 (p. 1470).

Sun, Y., Wang, X., & Tang, X. (2013). Deep convolutional network cascade for facial point detection. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013 (pp. 3476-3483).

Szarvas, M., Yoshizawa, A., Yamamoto, M., & Ogata, J. (2005, June). Pedestrian detection with convolutional neural networks. In Intelligent vehicles symposium (pp. 224-229).

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017, February). Inception-v4, Inception-ResNet and the impact of residual connections on learning. In AAAI (Vol. 4, p. 12).

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

Tang, P., Wang, H., & Kwong, S. (2017). G-MS2F: GoogLeNet based multi-stage feature fusion of deep CNN for scene recognition. Neurocomputing, 225, 188-197.

Tivive, F. H. C., & Bouzerdoum, A. (2003). A new class of convolutional neural networks (SICoNNets) and their application of face detection. In International Joint Conference on Neural Networks (Vol. 3, pp. 2157-2162).

Toreyin, B. U., & Cetin, A. E. (2007, June). Online detection of fire in video. In IEEE Conference on Computer Vision and Pattern Recognition (pp. 1-5).

Toreyin, B. U., Dedeoglu, Y., & Cetin, A. E. (2005). Flame detection in video using hidden markov models. In IEEE International Conference on Image Processing (Vol. 2, pp. II-1230).

Töreyin, B. U., Dedeoğlu, Y., Güdükbay, U., & Cetin, A. E. (2006). Computer vision-based method for real-time fire and flame detection. Pattern recognition letters,

27(1), 49-58.

Veit, A., Wilber, M. J., & Belongie, S. (2016). Residual networks behave like ensembles of relatively shallow networks. In Advances in Neural Information Processing Systems (pp. 550-558).

Vitabile, S., Pollaccia, G., Pilato, G., & Sorbello, F. (2001, September). Road signs recognition using a dynamic pixel aggregation technique in the HSV color space. In International Conference on Image Analysis and Processing (pp. 572-577).

Waibel, A. (1989). Modular construction of time-delay neural networks for speech recognition. Neural Computation, 1(1), 39-46.

Wu, C., Fan, W., He, Y., Sun, J., & Naoi, S. (2014, September). Handwritten character recognition by alternately trained relaxation convolutional neural network. In International Conference on Frontiers in Handwriting Recognition (ICFHR) (pp. 291-296).

Yamagishi, H., & Yamaguchi, J. (1999). Fire flame detection algorithm using a color camera. In International Symposium on Micromechatronics and Human Science (pp. 255-260).

Yan, W. (2017) Introduction to Intelligent Surveillance: Surveillance Data Capture, Transmission, and Analytics. Springer.

Yin, F., Wang, Q. F., Zhang, X. Y., & Liu, C. L. (2013, August). ICDAR 2013 Chinese handwriting recognition competition. In International Conference on Document Analysis and Recognition (ICDAR) (pp. 1464-1470).

Yu, D., & Deng, L. (2011). Deep learning and its applications to signal and information processing. IEEE Signal Processing Magazine, 28(1), 145-154.

Yuan, A., Bai, G., Jiao, L., & Liu, Y. (2012). Offline handwritten English character recognition based on convolutional neural network. In International Workshop on

Document Analysis Systems (DAS) (pp. 125-129).

Zheng, Y., Liu, Q., Chen, E., Ge, Y., & Zhao, J. L. (2014, June). Time series classification using multi-channels deep convolutional neural networks. In International Conference on Web-Age Information Management (pp. 298-310).

Zhong, Z., Jin, L., & Xie, Z. (2015). High performance offline handwritten Chinese character recognition using GoogLeNet and directional feature maps. In International Conference on Document Analysis and Recognition (ICDAR) (pp. 846-850).