Video Event Capturing Using RNN and CNN in Deep Learning

Keji Zheng

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

2017

School of Engineering, Computer and Mathematical Sciences

Abstract

In recent years, Deep Neural Network (DNN) has achieved remarkable progression in solving many complex problems. Recurrent Neural Network (RNN) is suitable for dealing with the problems related to time series, such as speech recognition and natural language processing. Obviously, video motion detection as an instance is also time dependent. Appearantly, video dynamic detection needs to compare the current, previous and next frames of this video. If a change occurs, it determines whether a motion or a video event has occurred or not.

In this project, video event capturing based on deep learning is implemented and our contributions effectively improve the accuracy of video dynamic detection and greatly reduce human labor compared to the traditional surveillance system. The contributions of this report are: (1) increase the correct rate to 96% compared to a Finite State Machine (FSM) based real-time video capturing. (2) By combining CNN and RNN (GRU), the training time has been greatly reduced as we expect.

Keywords: HMM, RNN, LSTM, CNN, Surveillance, Video Capturing, Event

Table of Contents

List of Figures

List of Tables

Attestation of Authorship

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

Signature: Keji Zheng Date: 23 June 2017

Acknowledgment

Fisrt of all, I would like to thank my parents for their finanical support. Owing to their unselfish and generous sponsor, I have this invaluable opportunity to complete my Master's study with the Auckland University of Technology (AUT), New Zealand.

I would also like to express my deepest gratitude to my primary supervisor Dr Wei Qi Yan. In this study, he not only provided me with professional knowledge support and careful guidance, but also helped me enrich my learning experience. I believe I could not complete my study without Dr Yan's supervision and instructions. Meanwhile, I also appreciate my secondary supervisor Dr. Parma Nand and the school administrators of AUT and China Jiliang University (CJLU) for their invaluable guidance.

Keji Zheng

Auckland, New Zealand

June 2017

Chapter 1
Introduction

This chapter is composed of five parts: the first part introduces the background and motivations, the second part includes the research question, followed by the contributions, objectives, and structure of this report.

1.1 Background and Motivation

Surveillance systems are gaining increasing usages in our community. Previously, they were only applied to sensitive premises such as airport, banks, hospitals and so on. In contrast with the developoment of technology, surveillance cameras are widely mounted in public area (Petrushin, 2005; Popoola & Wang, 2012). Moreover, traditional surveillance based manual operations has several obvious disadvantages including expensive labour costs, limited number of target objects being monitored, and high errant rates associated with manual monitoring (Remagnin, 2011). On the contrary, an intelligent surveillance system is able to resolve these problems, such as reducing costs and improving working efficiency.

Therefore, how to monitor a region in the event of an incident or accident with early alarming has become one of the hot topics in the field of video surveillance. In order to achieve better monitoring, the mastery of changes is a key factor for computing. As machine learning gains its viral popularity, research scientists in surveillance keep applying their algorithms to resolve these practical problems.

More and more surveillance systems are developing rapidly as the price of cameras continue dropping off, this causes its wide usages in our daily life, the importance of developing video surveillance systems has never been necessary like today. Surveillance cameras working in public areas mainly take into account the security needs of reducing labor; for example, such a camera is usually controlled remotely via the Internet, therefore the camera has the ability to monitor and record any abnormal behaviors.

It is said surveillance videos record events in the real world, events are able to enhance reusability and accessibility of a collection of media, an event is regarded as a function to detect abnormal human behaviors (Popoola & Wang, 2012). Therefore, in order to improve the efficiency of monitoring video and reduce our human labour, digital image processing and artificial intelligence have been widely applied to event capaturing, behavior recognition, alarm making and machine learning etc.

1.2 Research Questions

As we mentioned, this report aims to use deep learning into consideration, complete the real-time video capturing and event recording, and improve the accuracy and efficiency. Therefore, the research questions of this report are,

(1) What techniques can be implemented in real time for video capturing based on deep learning?

(2) How does deep learning based real-time video capturing improve the accuracy compared to traditional methods?

The core idea of this project is event based video capturing in surveillance using deep learning. Thus, a couple of appropriate techniques need to be chosen so as to implement the best result of video capturing and event recording. Furthermore, the methods that we adopted in this research project need to be evaluated. In terms of deep learning based video capturing, we need to train our data in order to get the best results.

1.3 Contribution

The focus of this project is on achieving real-time event capturing based on deep learning. According to the model of deep learning, we achieve high-precision and real-time event capturing which is compared with traditional methods. By the end of this report, we are able to, (1) use a simulation method to generate dataset; (2) process video in time series using RNN with GRU; (3) apply CNN to reduce the video size and extract critical information; (4) analyze the accuracy of video capturing.

Moreover, the core method of this project will be compared with a simple method based on FSM (Finite State Machine), we will collect experimental data under various conditions, analyse and justify the advantages/ disadvantages of these two methods.

1.4 Objectives of This Report

Firstly, a collection of dynamic methods in video capturing are introduced and evaluated. In addition, this project report will introduce a new dynamic method based on deep learning and compare it with the traditional ones.

Secondly, a novel framework is proposed to achieve event recording, video capturing based on dynamic detection using deep learning. Hence, the objectives of this report are divided into time series based RNN (GRU), video data reduction and feature extraction based on CNN, algorithm implementation using Python and Matlab. Finally, in this report we will introduce two methods for the purpose of comparisons through analyzing the experimental results.

1.5 Structure of This Report

The structure of this report is described as follows:

- In Chapter 2, we will conduct a literature review and discuss the relevant studies of RNN for the variants of RNN (LSTM and GRU). Meanwhile, we will compare Hidden Markov Model with the RNN model, put forward the faults of HMM in this report. In addition, the knowledge of CNN will be depicted.
- In Chapter 3, we will introduce the research methods. Experimental design and resultant comparisons will be present in this chapter.
- In Chapter 4, we will implement the proposed algorithms, collect experimental data and demonstrate the research outcomes in the form of figures and tables. Additionally, the limitations of these proposed methods will be detailed.
- In Chapter 5, we will summarize and analyze the experimental results.
- We will draw the conclusion and state our future work in Chapter 6.

Chapter 2
Literature Review

The focus of this report is on event capturing based on dynamic motion of deep learning, this chapter will introduce a plenty of traditional methods and the relevant knowledge of deep learning.

2.1 Introduction

With the increasing of global security problems, intelligent surveillance is gaining its attention from the public. The demands of surveillance for public security are soaring, such as security in banks, shopping malls, airports and markets, etc. Meanwhile, a growing number of residents are paying their close attention to privacy protection of their homes.

2.2 The Dynamic Detection

Moving object detection is the basis of target tracking and recognition. Intelligent monitoring has a broad range of real applications, these include the usages in industrial, telemedicine, military, and etc. The fast and accurate detection of moving objects has become a hotspot in the field of video surveillance (Han, 2015).

Surveillance system based on FSM is used to monitor patient movement (Noorit, 2014). The FMS-based activity capturing plays a pivotal role in the applications of human detection and recognition. The FSM has been employed to model, extract and capture the movements of any persons and identify the relevant motion characteristics. The experimental results show that FSM-based monitoring has a superior performance in video capturing and event recording.

Background subtraction based statistical model is a good method for moving target detection. When using statistical models for background training, we have to differentiate the background model from a single or multiple modes. For the single-mode background model, the colour distribution of these spots is relatively concentrated, we thus describe it by using a probability distribution model. However, for the multiple-mode background model, the colour distribution is relatively dispersed so that we use several distribution models to describe it properly. (Liu, 2012).

The motion energy method (Wildes, 1998) has been applied to various environment

which can eliminate the disturbing pixels from the complex background. Highlighting moving objects in a direction is possible, but there exist problems that cannot separate objects accurately. The neighbour frame difference algorithm and the background subtraction method (Collins & Wixson, 2000) are easy to be implemented and have high real-time capability (Anderson, 1985). However, they are affected by external conditions such as lightings, background updating, etc.

The existing motion detection methods include background subtraction and frame difference. However, there are shortages in frame difference method that are difficult to construct a background model by using background subtraction. Therefore, the test results could not reach the ideal state (Yang, 2013).

In summary, there are deficiencies in traditional methods. Hence, a new dynamic detection method based on deep learning will be implemented in this project. We will compare the new model with the traditional one under various experimental conditions.

2.3 Hidden Markov Model

The most important goal of machine learning is to estimate and speculate on unknown variables (such as category tags) of interest based on the observed evidence, such as training samples. The probabilistic model provides a descriptive framework, the learning work is attributed to the probability distribution of computational variables. In a probabilistic model, using known variables to speculate the distribution of unknown variables is called inference. The core of this model is to infer the conditional distribution of unknown variables based on observable variables. For instance, assume the variables of interest are set to Y, an observable set of variables is O, the collection of other variables is R, the generative $\mathrm{P}(Y, R, O)$, and discriminative $\mathrm{P}(Y, R|O)$. Given a set of observations of variable values, conditional probability distributions is set to $P(Y|O)$.

Hidden Markov Model, abbreviated as HMM (Baum, 1968) is one of the simplest dynamic probability models, which is mainly used in time series for data modelling and

widely applied to voice recognition and natural language processing.

Shown as Figure 1, variables in HMM are divided into two groups. The first group states variables $\{y_1, y_2 \dots y_n\}, y_i \in Y$ indicating the states of the system in the moment i. The state variable is usually assumed to be hidden and not observable, hence, the state variable is also known as the hidden variable. The second group is observational variables $\{x_1, x_2, x_3 \dots, x_n\}, x_i \in X$ representing the observed value of time i. In HMM, there are multiple state transitions $\{s_1, s_2, s_3, \dots, s_N\}$ between systems. Therefore, the value range of the observed variable Y is usually a discrete space with N possible values. An observational variable x_i is a discrete variable and may be continuity. Here we assume that the observable state is discrete, the value range X is $\{o_1, o_2, \dots, o_M\}$.

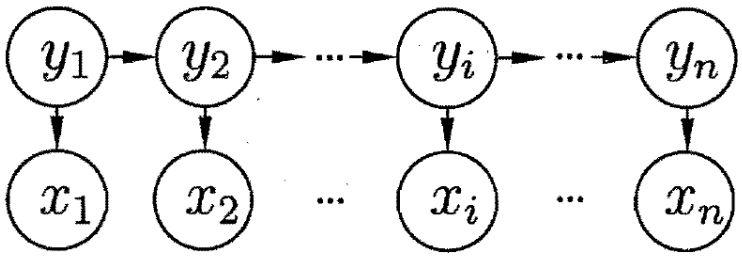

Figure 2.1. Hidden Markov Model

Figure 2.1 shows the dependency among variables. The value of observation variables depends on state variables only; x_t is determined by y_t and it is unrelated to the value of other state variables and observations. At the same time, the state at a time t depends only on the state y_{t-1} at time $t-1$ and it is unrelated to the rest of states. Based on this dependency of "Markov Chains", the joint consideration of all variables is distributed as,

$$\mathrm{P}(x_1, y_1, \dots, x_n, y_n) = \mathrm{P}(y_1)\mathrm{P}(x_1|y_1)\mathrm{P}(x_2|y_2) \dots \mathrm{P}(y_n|y_{n-1})\mathrm{P}(x_n|y_n) = \mathrm{P}(y_1)\mathrm{P}(x_1|y_1)\prod_{i=2}^{n}\mathrm{P}(y_i|y_{i-1})\mathrm{P}(x_i|y_i) \quad (2.1)$$

To determine an HMM, we also need the following three sets of parameters:

(1) **State transition probability.** The probability transition shown in different states is usually recorded as a matrix $\mathrm{A} = [a_{ij}]_{N\times N}$

$$a_{ij} = P\left(y_{t+1} = s_j \middle| y_t = s_i\right).\ 1 \le \mathrm{i,j} \le \mathrm{N} \tag{2.2}$$

(2) **Output observation probability**: The probability of each observation in the current state is usually recorded as a matrix.$\mathrm{B} = [b_{ij}]_{N\times M}$.

$$b_{ij} = P\left(x_t = o_j \middle| y_t = s_i\right), \quad 1 \le \mathrm{i} \le \mathrm{N}, 1 \le \mathrm{j} \le \mathrm{M} \tag{2.3}$$

(3) **Initial state probability**. The probability that appears in each state at the initial moment is usually recorded as a vector$\pi = (\pi_1, \pi_2, \dots, \pi_N)$.

$$\pi_i = P(y_1 = s_i),\ 1 \le \mathrm{i} \le \mathrm{N} \tag{2.4}$$

In video dynamic detection, we consider that the model has two hidden states $\mathrm{Y} = \{s_1, s_2\}$, where s_1 indicates that the scenery in the video has not changed and $s_2$represents that the scenery has changed. The observations are the images captured currently. Because of its dispersion of this image, we still consider the observational variables as discrete variables. Its pixels of the captured image are $\mathrm{w} \times \mathrm{h}$, w and h represent the width and the height of the image, respectively. Considering that the input image is an RGB image with 256 colors, the observed variable state space has a possible value of 256^{3wh} (Sotirios, 2011).

In order to apply HMM to video dynamic detection, we need to resolve the following two problems in the sequence (Eickeler, Muller, 1999)

(a) Training to get model parameters $\rho = [A, B, \pi]$. Because the artificially specified model is difficult to get the ideal effect, firstly we estimate the optimal model parameters using a set of training videos so as to achieve the maximum probability $P(x|\rho)$ on the training dataset.

(b) Given test video observation sequence $x = \{x_1, x_2, \dots, x_N\}$, we calculate the most matched status sequence $y = \{y_1, y_2, \dots, y_N\}$ so as to determine whether the video current frame is changed or unchanged according to the training model $l = \{A, B, P\}$.

There are limitations of this model (Rabiner, 1989). In the HMM, the current state is determined only by the state of last moment. In the process of video dynamic detection, the adjacent two frames may be identical in a period of time when the model is in the motion state. At this time, it is not enough to consider only the state change based on two adjacent moments. Although the HMM can be extended to the current state and the first *m* states, but under this assumption, the calculation of HMM will become more complex.

In summary, though HMM has been widely used in time series such as speech recognition and natural language processing, its computational complexity makes it unsuitable for directly applying to video dynamic detection. Currently, Deep Neural Network (DNN) has obtained very good results in learning. Among them, Recurrent Neural Network (RNN) has achieved better results in time series. Convolutional Neural Network (CNN) has also been widely used in the field of digital image processing. Therefore, this project will combine RNN and CNN in deep learning together to complete the mission of video dynamic detection.

2.4 Recurrent Neural Network

Comparing with Hidden Markov Model, RNNs represent the most general and powerful method of sequence processing that are not limited to discrete internal states but rather than continuous and distributed sequence.

Recently, Recurrent Neural Networks (RNNs) have been successfully introduced in language modelling for Automatic Speech Recognition (ASR), e.g. Mikolov et al. (2010), Mikolov et al. (2011) and Hori et al. (2014). The RNN Language Model (RNNLM) adopts the complete word history by recursively propagating the activation vector through its hidden layer with its own loops (Schwenk, 2007).

In 2012, Krizhevsky uses Deep Neural Network to reduce the errant rate of the classification to 15.3% in the ImageNet Large Scale Visual Recognition Challenge, while the traditional method is only at 26.2%. Clearly, the use of deep learning to extract image features far exceeds the traditional methodology.

Recurrent Neural Network (RNN) has been employed to model the inverse dynamics

of human body, and the RNN is one of the best ways to model dynamic networks with high nonlinearities (Narendra, 1990). The experimental data will be automatically generated by using machine learning to simulate the dynamics of human motion (Abdulrahman, 2014).

RNNs have been applied to languages models, but with the time lags increasing, the problem of gradient vanishing will be an issue (Zhang, 2016). In order to resolve this problem, special structures of RNNs are presented such as LSTM and GRU, these structures have "forget" units, given the memory cells having the ability to determine when to forget information. LSTM was first introduced to the language model in 1997 (Hochreiter & Schmidhuber, 1997). The solution based on LSTM needs a long time to be completed due to its computational complexity, hence, GRU is a variant of LSTM (Chung, 2014).

Long Short-term Memory (LSTM) is a novel efficient method that is based on gradient (Felix, 2000). LSTM aims to overcome the problem of vanishing error. Truncating the gradient where this does not work, LSTM bridges the minimal time lags in excess of 1000 discrete time steps by enforcing constant error flow through "constant error carrousels" within special units. So far, LSTM has led to successful operations involving artificial data, distributed, and real-valued problems and learned faster. LSTM also solves the complex artificial problems of long-time lag that previous algorithms never solved. Therefore, LSTM is offered in a myriad of practical applications such as speech recognition and language separation form prosody (Cummins & Schmidhuber, 1999).

The LSTM structure is proposed to resolve the problem of gradient vanishing (Pascanu, 2012). Not only can we learn the short-term dependency of time series, but also successfully get across the long-term dependency of time series. Therefore, LSTM is widely employed in RNN. The fundamental difference between LSTM and RNN units is that a long-term memory item c_{t-1} is introduced on the basis of current input x_t and the last output h_{t-1}, so that the long dependency can be stored. From the perspective of mapping, the RNN provides a mapping from x_t, h_{t-1} to h_t.

$$\text{RNN: } (x_t, h_{t-1}) \rightarrow h_t \tag{2.5}$$

While the LSTM has the mapping from x_t, h_{t-1}, c_{t-1} to h_t, c_t

$$\text{LSTM: } (x_t, h_{t-1}, c_{t-1}) \rightarrow (h_t, c_t) \tag{2.6}$$

LSTM structures take effects in various specific learning. In fact, there are more than 10,000 kinds of LSTM structures. This report takes the LSTM model used by Google Brain's Zaremba (2014) and others as an example to improve the LSTM's dropout strategy and the basic structure of LSTM. The model is also a model implemented by Basic LSTM Cell in Google's Deep Learning Frame - Tensor Flow.

An LSTM unit consists of four gate structures: Input Gate-i, Input modulation gate-g, Forget gate-f, and Output gate-o. The equations are,

$$\mathrm{i} = \mathrm{sigmoid}(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \tag{2.2}$$

$$\mathrm{f} = \mathrm{sigmoid}\left(W_{xi}x_t + W_{hf}h_{t-1} + b_f\right) \tag{2.3}$$

$$\mathrm{o} = \mathrm{sigmoid}(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \tag{2.4}$$

$$\mathrm{g} = \tanh\left(W_{xg}x_t + W_{hg}h_{t-1} + b_g\right) \tag{2.5}$$

$$c_t = f \odot c_{t-1} + \mathrm{i} \odot \mathrm{g} \tag{2.6}$$

$$h_t = o \odot \tanh\left(c_t\right) \tag{2.7}$$

The most critical part of the LSTM model is a storage unit c_t. *Forget Gate* uses the Sigmoid function as the activation function, the range of this sigmoid function is between (0, 1) which is used to represent the model's Forget Rate (Gers, 2001; Schmidhuber, 2002; Chen, 2015).

LSTM units have more than 10,000 variants, most of them are similar but differ only from selection of the activation function. There are also special variations, such as the concept of peepholes proposed by Gers (2000) etc., that is, the memory unit is also

involved in calculation of the Gate.

Gated Recurrent Unit (GRU) model based on the LSTM is a big change (Cho, 2014) as Forget gate is integrated with input gate and turns them into a single update gate. Mixed with cell states, hidden states, and other changes, the final model is simpler than the standard LSTM model.

LSTM is used to model the video (NG, 2015) which connects its output of the underlying CNN as the input at the next moment, obtains 82.6% recognition rate on the UCF database. A long-term recurrence convolution network combines CNN, reaches 82.92% accuracy rate (Donahue, 2014).

An energy decomposition model based on depth learning using GRU as the basic unit of RNN trains the model (Kim, 2016). Its model with the GRU as the basic unit is compared with the original RNN, and the accuracy is 89% ~ 98% and 81% ~ 98%. GRU is applied to question detection based on GRU to establish RNN and Bidirectional RNN (Tang, 2016). The result shows that the special advantage of GRU is that it can determine a ime scale to extract high-level contextual features which make the result better than the baseline method. Meanwhile, the RNN with GRU is applied to natural language processing and improve the effectiveness of its architecture (Zhang, 2016). The data show new models can achieve competitive performance compared with the state-of-the-art technologies.

2.5 Convolution Neural Network

The human neural system consists of billions of neurons. In computer science, Artificial Neural Network (ANN) simulates the computational model of animal nerual system, allows the computer to interact like our human brain. Fukuhima (2016) proposed a neural cognition machine that could model the central part of our visual system. In 1989, LeCun et al. applied the convolution neural network to pattern recognition using a training method based on the error gradient.

Convolution Neural Network (CNN) (Matsugu & Kaneda, 2003) is based on deep learning theory. It mainly uses weight sharing to resolve the problem of parameter expansion in the ordinary neural network. Hubert and Wessel (1962) proposed a convolutional neural network model, which reduces its complexity of the feedback neural network effectively. Now, the CNN has been developed and become an effective method.

A multi-resolution convolutional neural network is offered to extract video features by using 3D CNN (Varol et al, 2016; Karpathy, 2014). The input video was divided into two sets of data streams: low-resolution data streams and raw data streams. The streams contain convolution layers, regular layers, and extraction layers.

Senior and Beaufays (2014) also made use two convolutional neural networks for video behavior identification, which divided a video into static-frame data streams and interframe dynamic data streams. Static-frame data streams take single frame data whilst interframe dynamics of the data stream works with opticalflow data, so each data is used in the depth of convolution neural network for feature extraction.

Chapter 3
Methodology

The main content of this chapter is to clearly articulate research methods, which satisfy the objectives of this report. The chapter mainly covers the details of research methodology for video dynamic detection using deep learning which will be clearly introduced with the confident and imaginative use of the feature description methods.

3.1 Recurrent Neural Network

In recent years, Deep Neural Network (DNN) has achieved remarkable results on a great deal of complex issues. Recurrent Neural Network (RNN) is suitable for handling the time series related issues, such as speech recognition and natural language processing. The basic RNN unit is described as Figure 3.1. As we see, in an RNN unit, x_t is input and $h_t$means output. Not only x_t, but also the output of last moment h_{t-1} will be needed to calculate h_t. In Figure 4, the iterative arrows in the RNN unit indicate that the current output is stored as the next input.

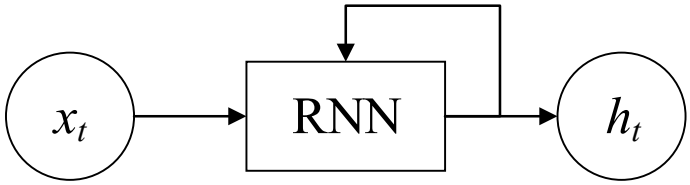

Figure 3.1 The basic unit of RNN

The RNN unit model is expressed as,

$$h_t = f(x_t, h_{t-1}) \tag{3.1}$$

This model is expanded as,

$$\begin{aligned} h_0 &= f(x_o, 0) \\ h_1 &= f(x_1, h_0) = f\big(x_1, f(x_o, 0)\big) = g(x_0, x_1) \\ &\vdots \\ h_t &= f(x_t, h_{t-1}) = g(x_0, x_1, \dots, x_t) \end{aligned} \tag{3.2}$$

We have that the RNN model interprets the state x_t at the time t into a function of all the previous input $x_0, x_1, \dots, x_t$. An RNN unit is expanded vividly in Figure 4. Clearly, $x_0, x_1, \dots, x_t$ is a function of h_t. Therefore, the RNN is regarded as the promotion of HMM, which considers not only the influence of the previous one on the current state, but also the impact of all the previous states of the current one. In the course of video dynamic detection, the RNN determines its motion state of the current frame by comparing the current frame x_t with all the previous frames $x_0, x_1, \dots, x_{t-1}$.

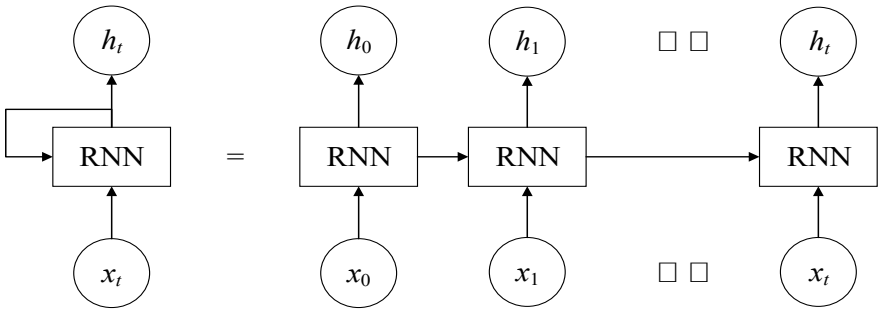

Figure 3.2 RNN unit expansion

The most commonly used RNN units are shown in the following form,

$$h_t = f(W_x x_t + W_h h_{t-1} + b), f \in \{sigmoid, tanh\} \tag{3.3}$$

Both input x_t and output h_t are vectors; W_x, W_h are the weight matrices of x_t, h_{t-1}, respectively. The activation functions are *sigmoid* functions and *tanh* functions,

$$Sigmoid(x) = \frac{1}{1+e^{-x}} \tag{3.4}$$

$$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{3.5}$$

The *sigmoid* function maps the real numbers to the range, and the *tanh* function maps the real numbers to the range; the images are shown in Figure 3.3

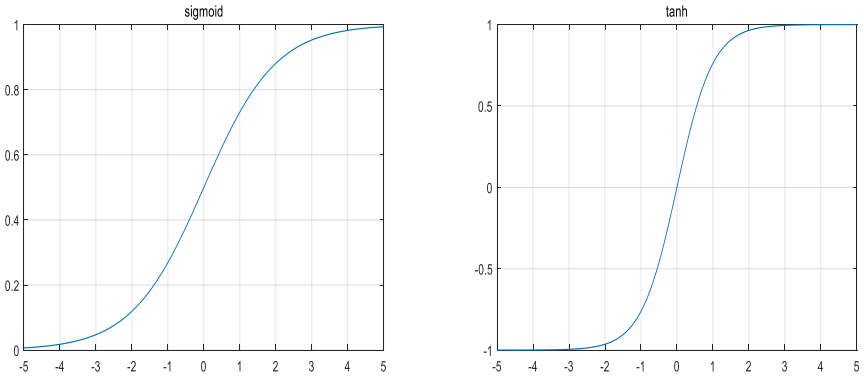

Figure 3.3 The *sigmoid* and *tanh* functions

Theoretically $h_t = f(x_t, h_{t-1}) = g(x_0, x_1, \ldots, x_t)$, the RNN tackles any length of dependency. In practice, RNN may not be able to successfully learn this knowledge.

Bingio (1994) conducted an in-depth study of this problem, because almost all the RNNs currently adopt a gradient-dependent algorithm (gradient descent method, Adam algorithm, etc.) to carry out the model training, they are dependent on the results of gradient calculation. Long dependencies of the gradient tend to reach 0 because of the limitations of traditional RNN expression that makes the training algorithm ineffective. The LSTM structure is offered to solve this problem (Hochreiter, 1997).

3.2 GRU

Zaremba (2015) has extensively tested 10,000 variants of LSTMs and concluded that GRU is the best one in most cases, initializing the offset of forget gate to a larger value for other LSTM variants is used to get a similar result as the GRU. In general, these LSTM variants have almost the same effect.

However, the GRU model has less parameters than LSTM, the model will converge earlier in the training. Also, the GRU does not require a deliberate initialization operation to achieve good results, hence we choose GRU as the basic unit of RNN.

Gated Recurrent Unit (GRU), as a relatively successful variant of LSTM, its parameters are smaller than average LSTM, and it is easier to achieve better results. Therefore, this report will adopt GRU as the basic unit of RNN. The update equations of GRU are as follows,

$$\begin{aligned} r_t &= \text{sigmoid}\left(W_r \cdot \left[h_{t-1}, x_t\right]\right) \\ z_t &= \text{sigmoid}\left(W_z \cdot \left[h_{t-1}, x_t\right]\right) \\ \tilde{h}_t &= \tanh\left(W \cdot \left[r_t * h_{t-1}, x_t\right]\right) \\ h_t &= \left(1 - z_t\right) * h_{t-1} + z_t * \tilde{h}_t \end{aligned} \tag{3.6}$$

Assume that the input and output vectors are *n*-dimensional vectors, the parameter GRU requires are $n \times 2n$ matrices, and the total number of parameters is $6n^2$. The LSTM unit in the previous section requires $8n \times n$ weights matrices and 4 *n*-dimension offset vectors, the total amount of parameters is $8n^2 + 4n$. Therefore, the number of GRU parameters is less than the ordinary LSTM model. Combined with Zaremba (2015) et al., the results

of 10,000 LSTM structures are tested. GRU, even with less parameters, can achieve better results.

3.3 Convolutional Neural Network

If a RNN is used directly in training, we consider the input vector is a 80×60 grayscale image, a GRU unit requires 1.4×10^8. While this is calculable, the training time will be very long due to the computing limitations.

On the other hand, because the input appears only a part of the image rather than the whole image, inputting the entire image directly will produce more redundancy. Thus, most of the image information is preserved despite the image resolution is smaller.

Convolutional Neural Network (CNN) achieves better results in image-related fields. A CNN unit typically consists of a convolutional layer and a max pool layer. The convolution layer mainly uses convolution to extract different features from the image species. The pooling layer is implemented by subsampling and further reduces the input scale of the image.

Convolution is a partial operation on the image, given a convolution matrix W, for the input image I, its output image O satisfies,

$$O(i,j) = \sum_{u,v} W(u,v) I(i+u, j+v) \tag{3.7}$$

Table 3.1 Convolution operator and convolution effects.

Description	Convolution operator	Effect
original picture	$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$	
Boundary extraction 1	$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ 1 & -2 & -1 \end{pmatrix}$	
Boundary extraction 2	$\begin{pmatrix} 0.4038 & 0.8021 & 0.4038 \\ 0.80210 & -4.8233 & 0.8021 \\ 0.4038 & 0.8021 & 0.4038 \end{pmatrix}$	

In traditional image processing, the convolution operator is assigned manually. In CNN, each element of the convolution matrix is obtained by training. Generally, for the same input image, multiple convolution operators are used to obtain the corresponding features; the convolution layer of a CNN is shown in Figure 3.4

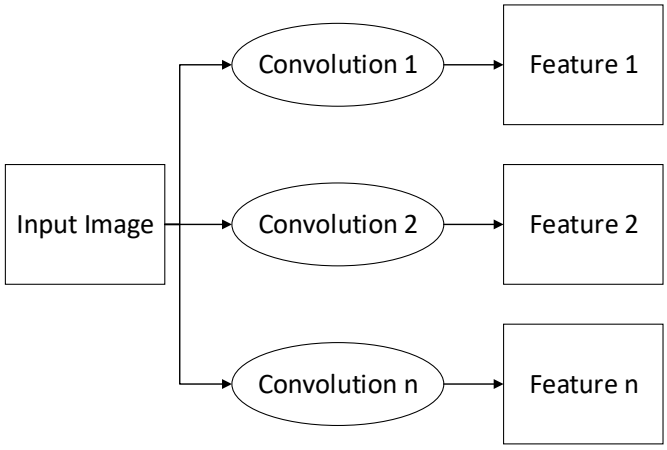

Figure 3.4 CNN convolution layer

Pooling is also a partial operation, whilst the most commonly used is the max pooling operation, of course, we also can use mean pooling in some cases. The Max Pooling is divided the input image into $m \times n$ units. The maximum value is selected to represent each cell. Figure 3.5 shows the operations of Max Pooling with 2×2 matrix. After the pooling operation, the image is condensed greatly.

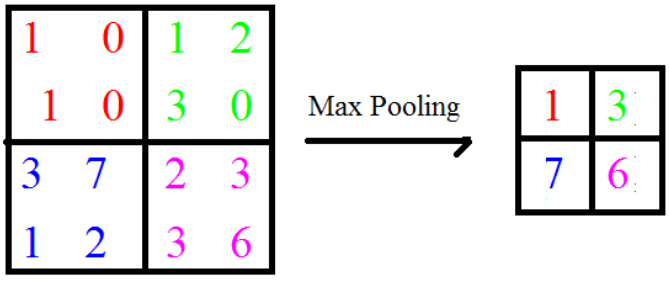

Figure 3.5 Maximum pooling

The general CNN structure is shown as Figure 3.6. A CNN first has multiple convolution layers, each of them is followed by a polling layer. After multi-layer convolution, the obtained feature is expanded into a vector, which is an input of Full Connected Feedforward Neural Network, then the final output of CNN is obtained.

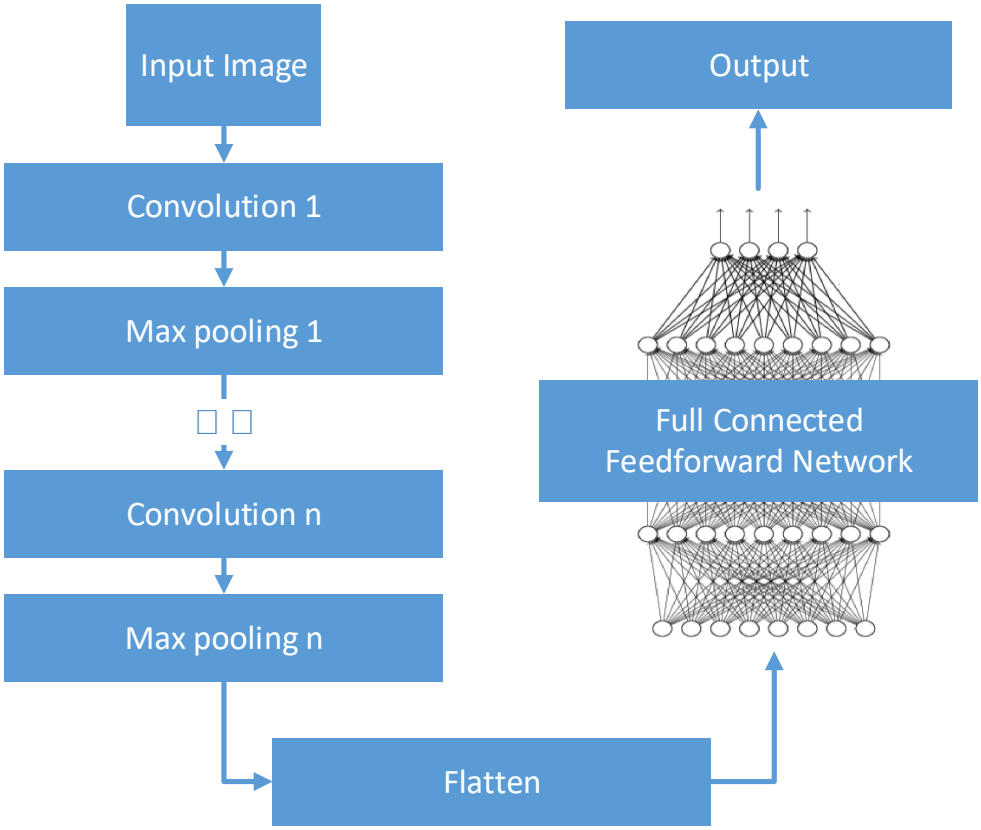

Figure 3.6 CNN diagram

In addition to a fully connected neural network, adding a dropout operation can effectively avoid the over-fitting problem of artificial neural networks. In the training process, random deletion of nodes will make the training results much stable and easy to avoid overfitting shown as Figure 3.7

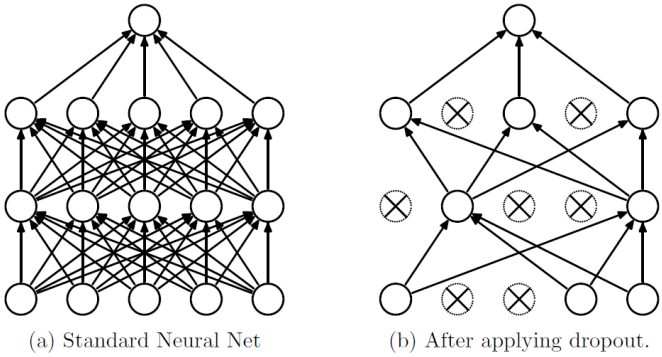

Figure 3.7 dropout diagram

In summary, because CNN can effectively reduce its size of the model parameters, it achieves better results on the image-related issues. The RNN is also suitable for dealing

with time series related issues. Hence, this report will combine CNN with RNN in one model together.

3.4 Video Dynamic Detection Model Based on Deep Neural Network

In this section, we will combine CNN and RNN together to construct a DNN so as to complete video dynamic detection. The model is shown as Figure 3.8.

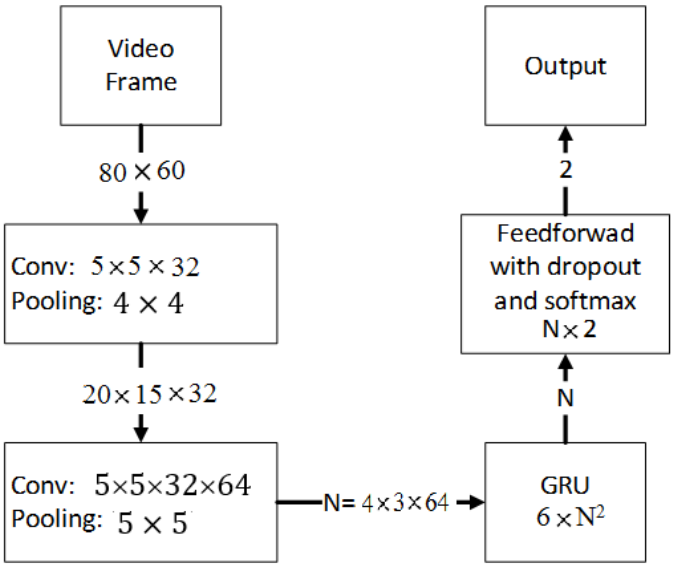

Figure 3.8 Video Dynamic Detection Based on Deep Neural Network

The model uses 80×60 grey image as input, the first convolution layer is a 5×5 convolution template, extracting 32 features, then using 4×4 max-pooling operation. The second layer is convoluted with a $5\times5\times32\times64$ convolution template and max-pooling (5×5). Eventually get $N = 4 \times 3 \times 6 = 768$ dimensional output. After the output of the CNN part is expanded into a 768-dimension eigenvector, the result of RNN part is obtained by using a GRU unit, which is still a 768-dimension eigenvector. This feature vector represents the overall difference between the characteristics of the current frame and that of all previous frames.

Finally, after a full connection layer using a dropout strategy, a two-dimensional

output is achieved. The two-dimensional output vector, after the Softmax operation, can be converted into a probability vector; each component represents the corresponding possibility of the output results. The Softmax is calculated as follows:

$$p_i = \frac{\exp(x_i)}{\sum_{i=1}^{n} \exp(x_i)} \tag{3.8}$$

The Softmax normalizes the input vector x to a probability vector p so that each component of p is positive and the sum of the components is 1. Therefore, the value of each component indicates the corresponding classification possibility of the output results. In this proposed model, its probability vector of the output is only two dimensions: the first dimension represents the probability when the current frame is static and the second dimension represents the probability when the current frame is dynamic. In the prediction, the probability of a large state is taken as the prediction result. In addition, the cross-entropy is a loss function in the model training. The cross-entropy is defined as,

$$\mathrm{L} = -\sum_{i=1}^{n} q_i log\ p_i \tag{3.9}$$

where p_i is the model prediction and q_i is the real result. If the current frame is a static frame, then $\mathrm{q} = [1,0]$. If the current frame is a dynamic frame, then $\mathrm{q} = [0,1]$. The smaller the value of the cross entropy is, the more the prediction result will be. For the DNN model in this project, the total number of parameters is 3,592,842, which is much less than that of the RNN model.

In summary, CNN is used to extract the feature of the current frame of a video, correspondingly the parametric size is reduced. Then, GRU is used as the RNN unit to obtain the difference between the current frame and the previous frame. Finally, through a fully connected neural network with dropout strategy layer, after calculating the Softmax to get the current frame, the final prediction of the current frame outputs the motion.

3.5 Training Data

When the DNN model is determined, its predictive effect depends on the training data set. Therefore, in order to achieve good results, appropriate data sets should be chosen. In the process of video dynamic detection, the dataset we need is a series of videos, each frame of them marks the motions of this video. It is hard to find such datasets and manually labeling is also a troublesome work. Hence, a simulated method is adopted to generate the training dataset in this project. In addition, the training data generated in this way can help us analyze the prediction accuracy of the proposed model in different situations. Therefore, firstly we generate the model of a single object and then the model of multiple objects will be discussed based upon the generation model of a single object.

For the model of dynamic motion detection of a single object, the following points are important:

(1) **Shape of the moving object.** In the process of video motion capturing, there are many kinds of moving objects which are represented by shapes. To simplify the problem, we generalize the shapes as a rectangle and suppose its aspect ratio is α.

(2) **Area of the moving object.** The effect of different objects on a video is different. The larger the area of the moving object is, the easier it is to be detected; the smaller the area is, the harder it is to be detected. Assuming the ratio of the region of the moving object in the screen is A, the pixel of the whole picture is S the length and width of the object are $\mathrm{l} = \sqrt{SA/\alpha}, \mathrm{w} = \alpha\sqrt{SA/\alpha}$ respectively.

(3) **Position of the moving object.** The position of the moving object can be determined by (x, y, θ) , (x, y) is the coordinates of the bottom-left corner of the object, q is the angle between the length of the region and the horizontal direction of the video.

(4) **Object motion.** The impact of the same object is also different because of the different directions in object moving. We only consider moving between the uniforms for the sake of simplicity. We set the speed as v pixels per frame, angle between the velocity direction and the horizontal direction as φ, the components

of velocity in horizontal and vertical directions are, $v_x = vcos\varphi, v_y = vsin\varphi$ respectively. The location of the object is simply denoted by (x, y).

(5) **Color of the moving object**. We consider that the moving objects only have a single color $c \in [0, 1]$.

(6) **Movement time.** We set the total length of a video as T, the time at which an object start moving is t and the duration is d, then $[t, t + d] \subset [0, T]$.

.

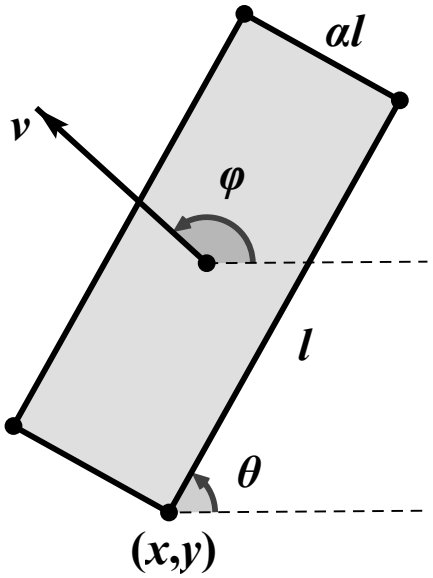

Figure 3.9 Single object motion model

The model of a single moving object is shown in Figure 3.9 A moving object is uniquely determined by the parameter $(A, \alpha, x, y, \theta, v, \varphi, c, t, d)$. Based on a single object movement model, we see that the factors that affect the video are:

(1) **Noise**. If there exist noises in a video, the results of two video shots on the same object may be different. In general, the noise may be related to brightness and edge of the scene. In this report, we assume that the noise is White Gaussian noise.

(2) **The number of background objects**. Assuming there are, M background objects, the model of a single object can be used to generate M moving objects with a velocity of 0 as the background object. The set of parameters is denoted as S_M.

(3) **The number of moving objects.** In the same video screen, there may be multiple

moving objects. Assuming N moving objects, the set of their parameters is denoted as M_N.

(4) **Video duration.** In addition, we need to specify the video duration T.

To sum up, a video is uniquely identified by the quadruple (T, σ, S_M, M_N). We use the above method to generate multiple videos according to different parameters. Part of the generated dataset should be employed for training the model and the rest for the validation. Since the parameters of this model are known, we therefore analyze the accuracy changes of our proposed model under different parameter conditions.

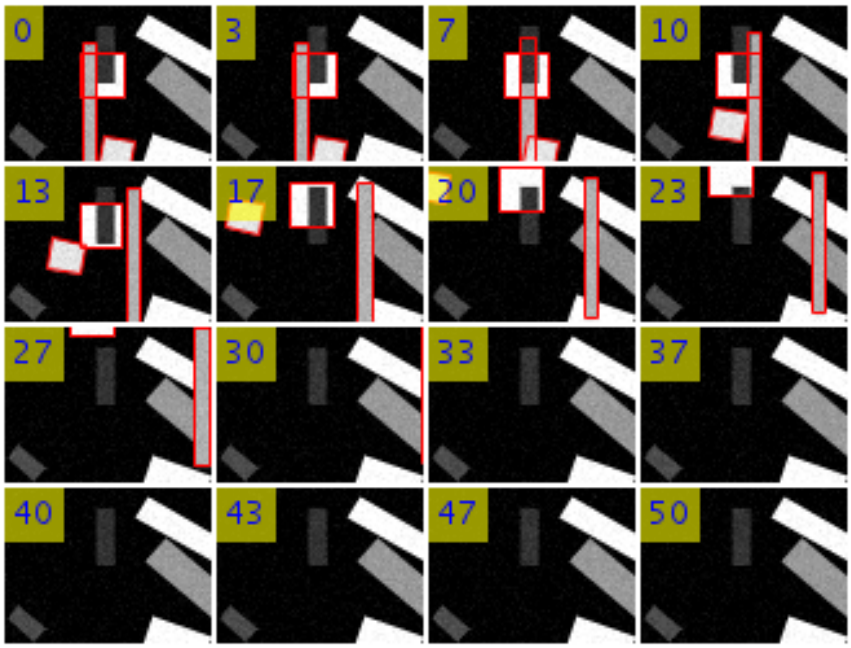

Figure 3.10 Training video examples

Using the method described, we generate complex scenes to simulate the real video. As shown in Figure 3.10, there are three moving objects marked with a red frame in the scene, the thumbnails mark the current frame number at the top left corner. Figure 13 depicts a scene with three moving objects: the bright white rectangle moving upward vertically (object 1), the long grey rectangular block moving horizontally to the right (object 2), the tilted rectangular block moving to the up-left direction (object 3). This shows that our model has the ability to describe multiple objects of different movements. Object 1 is located at the lowest position and

occluded by a dark grey rectangle at the upper part of this image. Object 2 is located on the top layer of the object 1. Object 3 is located at the uppermost layer which shows that our model is able to tackle the actual relationship of mutual occlusion in the video. In addition to the three motional rectangles, five rectangular blocks with different colors are introduced in the video.

3.6 Program Implementation

Because of its excellent performance in video capturing, image processing, and data analysis, MATLAB is employed to achieve our goal of this project. Although Python can also fulfil this mission, it has to rely on multiple third-party libraries and its documentation is not perfect. Additionally, we are not familiar with the Python language and MATLAB also has the interface to call Python, so we chose mixed programming model of Python and Matlab to achieve our program. Our programming framework is shown as Figure 3.11.

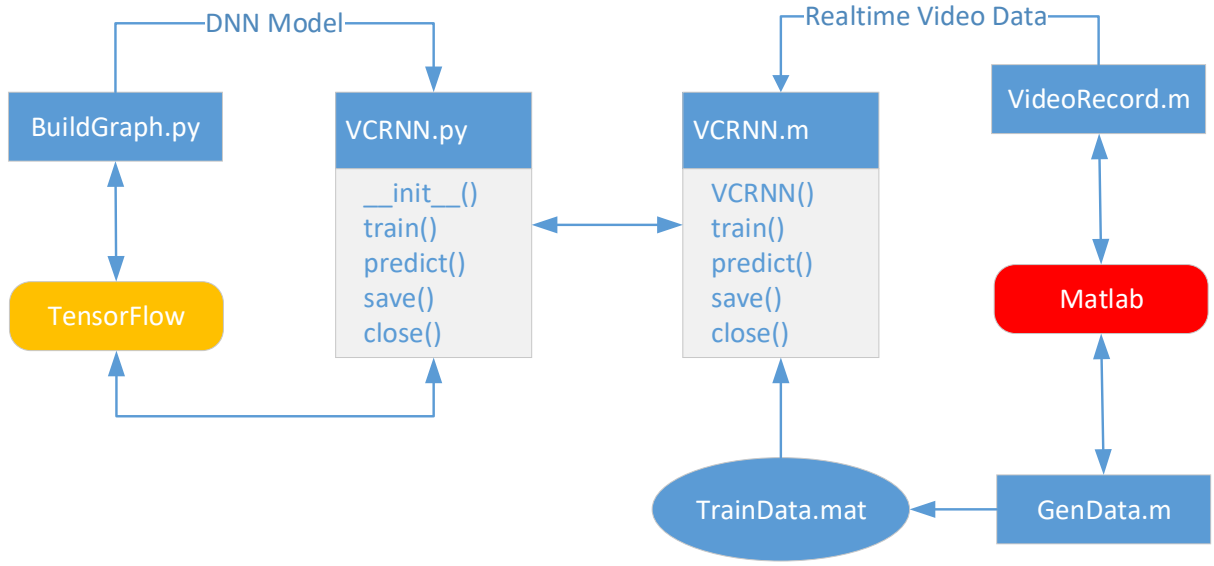

Figure 3.11 Program framework

Although MATLAB can call the functions of Python, it has some limitations and does not support all Python syntax. In addition, MATLAB and Python data transmission has limitations, the data representations are also different. Therefore, in Python and MATLAB, there are two interface files VCRNN.py and VCRNN.m, respectively, to achieve the call the functions of MATLAB by using TensorFlow DNN model. These two

files achieve a function to manage the structure, training, prediction, preservation, and other functions of DNN model. The Python interface is responsible for packaging the TensorFlow model, provided a convenient calling interface; the MATLAB interface is responsible for calling Python interface and mutual data transformation between MATLAB and Python. In Python, VCRNN.py is an interface, the specific DNN model is implemented in BuildGraph.py.

In MATLAB, GenData.m is responsible for generating training data and saving it to TrainData.mat. When training the model, the relevant data is read from TrainData.mat and passed by VCRNN.m to Python for training. VideoRecord.m is responsible for implementing a video recording and dynamic monitoring of the demo, and then transmits the captured real-time video frame to the VCRNN.m and calls TensorFlow DNN model to determine the status of the video frame.

In this Chapter, we will describe how to use TensorFlow to implement our DNN model, and describe what the difference is between the actual implementation of the DNN model and the theoretical model.

In addition, using the DNN model of TensorFlow, visualization can be easily realized through TensorBoard. For example, the visualization results are shown as Figure 3.12. The Conv1 and Conv2 in the figure represent Convolution Layer 1 and Convolution Layer 2, respectively. GRU represents our RNN layer, the “forward” represents the fully connected network layer with dropout strategy, the “train” refers to the relevant model for training; the “valid” indicates the relevant validations; the “save” means the relevant saving operation of the model. “rnn” is the specific implementation of GRHC unit and it is automatically created by TensorFlow. The composition and function of the relevant structure is illustrated as well.

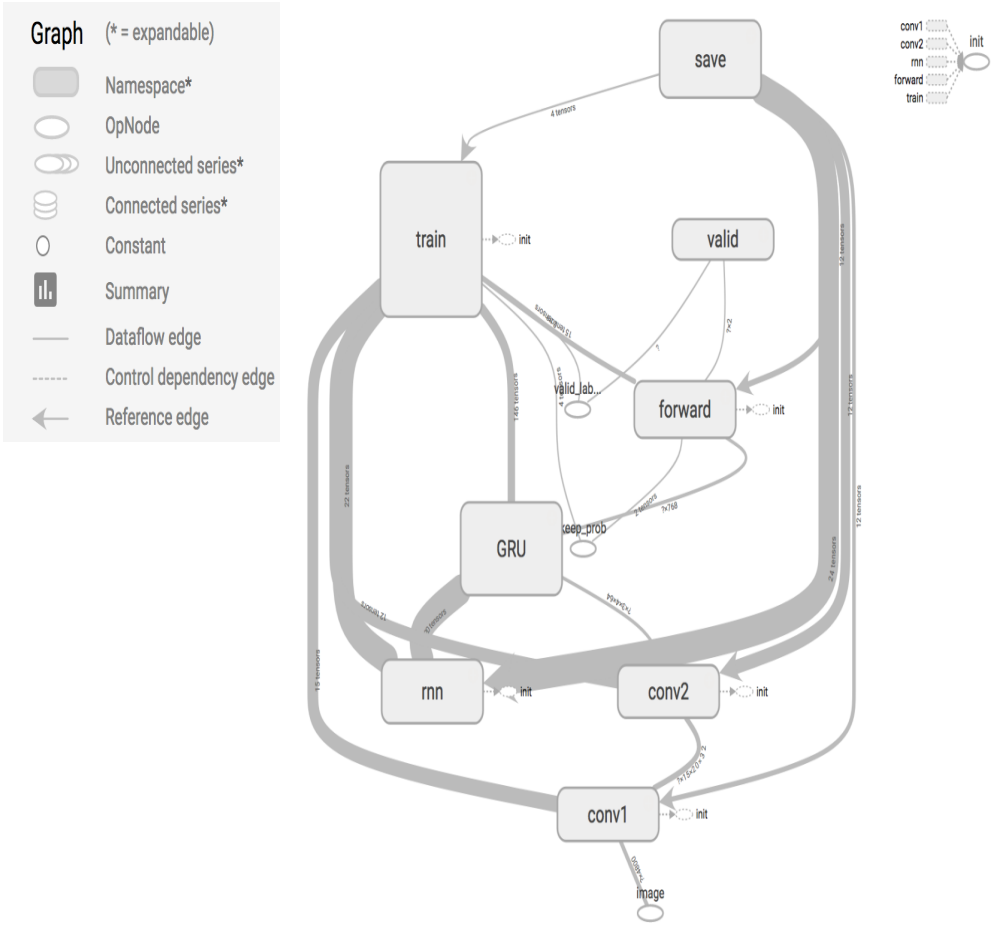

Figure 3.12 DNN model of TensorBoard visualization

In the first convolution layer, we define a weight variable W_1 whose size is [5, 5, 1, 32], representing a four-dimensional tensor consisting of 25 (=5×5) convolution templates. After the input data is changed from [batch_size, 4800] to [batch_size, 60, 80, 1], it is convoluted with convolution template W_1, and a tensor of [batch_size, 60,80,32] is obtained. After the 32 bits are extracted by convolution, the input is added to a bias b_1 and then output by the activation function ReLU. The output is max pooling according to the 4 × 4 grids, and its size of the output is changed to [batch_size, 15, 20, 32]. The mathematical expression of ReLU is,

$$ReLU(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{3.10}$$

The reason why the ReLU function rather than *Sigmoid* or *tanh* function is used as the activation function is that the ReLU function does not have the problem of gradient which can guarantee the effective training of the model.

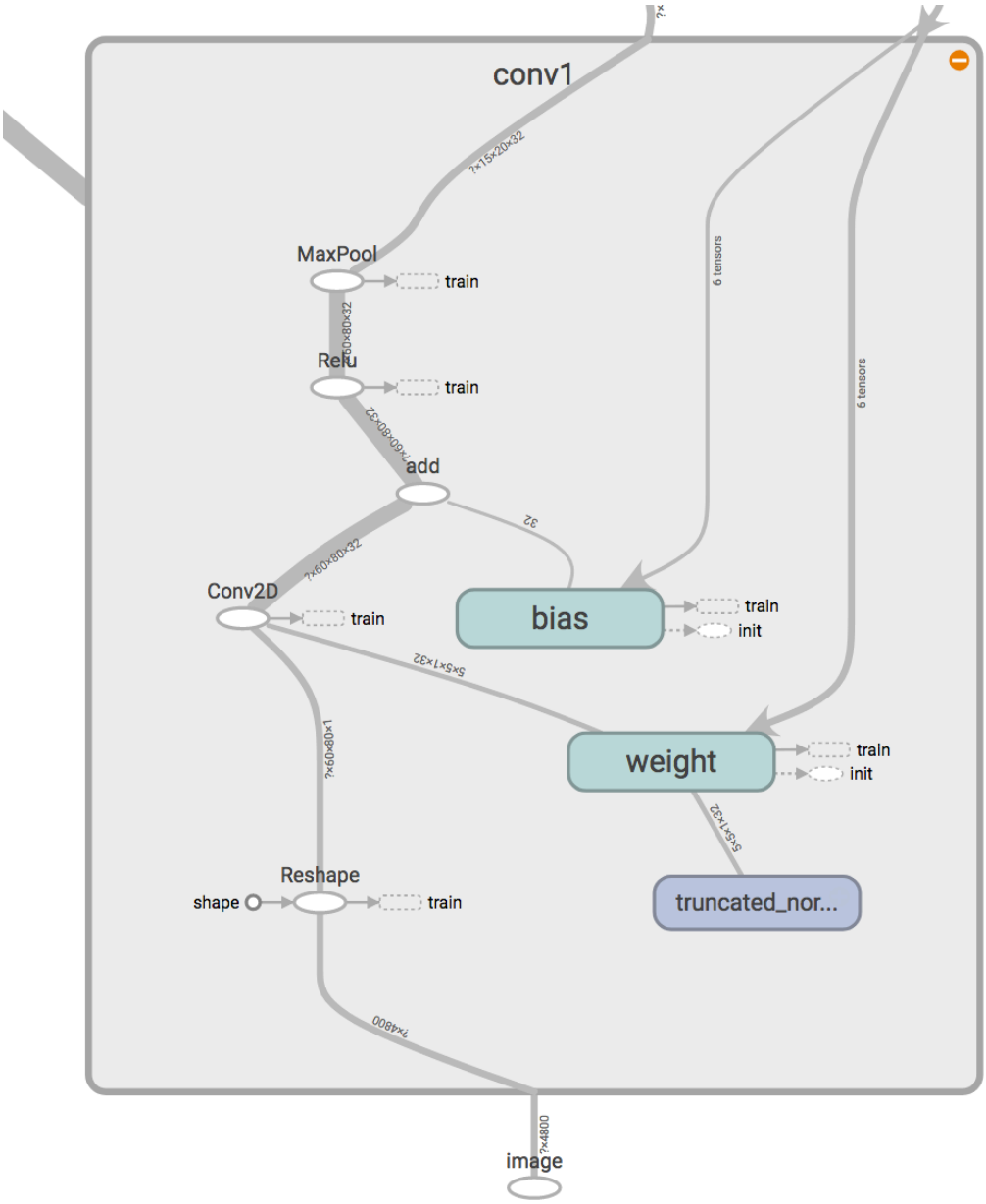

Figure 3.13 First convolution layer

After Convolution Layer 1 is developed, the result in Figure 3.13 is obtained. In Figure 3.13, the image node is used as input; the shape is changed through the reshape node, and then convoluted by Conv2D and convolution template weight. The result is added by the node and the bias. After the ReLU node is used as the activation function, the maximum pooling is carried out, and its output of the convolution layer 1 is finally obtained. TensorBoard is used to display of the convolution layer 1 network structure vividly.

We use the GRU unit to construct the RNN, the Tf.contrib.rnn.GRUCell is employed to declare a GRU unit. First, we expand the output of convolution layer into a Batch GRU

vector, and then use the feed function to expand *Num* Steps to get the output of RNN.

```
gru_size = 4 * 3 * 64
num_steps = 5
with tf.name_scope('GRU'):
    cnn_output = tf.reshape(pool2, [-1, gru_size], name='cnn_output')
    gru_cell = tf.contrib.rnn.GRUCell(gru_size)
    (gru_output, output_state) = feed_rnn(
        gru_cell, cnn_output, num_steps, gru_size)
```

Amongst them, the actual RNN model needs to be carried out according to the concrete steps. If we expand any length, we cannot compute the relative gradient of the RNN unit, so that the model cannot be optimized. The specific implementation of the feed function is as follows,

```
def feed_rnn(cell, x, num_steps, rnn_size):
    with tf.name_scope('expand'):
        x_pre = [tf.reshape(x[0, :], [1, rnn_size])
                 for k in range(num_steps - 1)]
        x = tf.concat([*x_pre, x], axis=0)
        input_ = []
        for k in range(num_steps):
            if k < num_steps - 1:
                input_.append(x[k:(-num_steps + k + 1), :])
            else:
                input_.append(x[k:, :])
    out, state = tf.contrib.rnn.static_rnn(cell, input_, dtype=tf.float32)
    return out, state
```

To invoke the Tensorflow tf.contrib.rnn.static_rnn function, we need to convert the original input *x* into a list with *Num* Steps elements. The simplified RNN model is,

$$y = f(x_{t-num_{steps}+1}, \dots, x_{t-1}, x_t) \tag{3.11}$$

To make the dimensions of the output and the number of dimensions of the output equal, we fill in a number of x_0 to populate the subscript less than or equal to zero.

Finally, we take its output of the RNN as input of the fully-connected neural network layer, and the matrix multiplication by the weight matrix, plus the bias vector $\mathbf{B}_3$ to get the final output. In addition, by using Tf.nn.dropout function, a Dropout operation can be added to avoid the problem of overfitting.

After defining the DNN model, we also need to specify the loss function and training

algorithm to train the model. We use the cross entropy as the loss function of the model and the ADAM algorithm as the optimization algorithm.

3.7 Evaluation Methods

In this project, the evaluation was fulfilled by comparing different methods in video capturing. Before the experiment, we first need to generate the experimental data. In this Section, we will describe how to generate these experimental data, and indicate the range of parameters.

First of all, we capture a unified video length $T = 50$ indicating a total of 51 frames starting from frame 0. If the FPS equals 10, the video duration is 5 seconds. In order to analyze the standard deviation s between the proposed algorithm and the given noise, the relationship between the number of background objects M and the number of moving objects N, and s={0,0.125,0.025}, M={0,1,2,3,4,5}, N={1,2,3,4,5}. For the Cartesian Product, it is a total of 3×6×5 =90 sets of parameters. Each set of parameters generates 10 random samples, a total of 900 test videos and 45,900 images. Other parameters are randomly generated within the specified range; the specific values are as shown as Table 2.

Table 3.2 Parameter range of values

Parameters	**Ranges**
Area ratio A	(0.01,0.1)
Width to length ratio	(0.1,1)
Coordinate (x,y)	(0,80) (0,60)
Tilt angle q	(0, p/2)
Speed size v	(1,10)
Speed direction angle j	(0, 2p)

Color c	(0.1,1)

Before we test our algorithm, a benchmark algorithm, Simple Threshold Method (STM), is introduced. At time t, a frame of video S_t is a $h \times w$ matrix, the matrix element is between {0,1}. The STM decision method, given a threshold δ, calculates the difference between two adjacent video frames,

$$\Delta S_t = \frac{1}{hw}\sum_{i=1}^{h}\sum_{j=1}^{w}(S_t(i,j) - S_{t-1}(i,j))^2 \tag{3.12}$$

Where $\Delta S_t > \delta$, the video frame is dynamic; when $\Delta S_t < \delta$, the video frame is static. The 0-1 delta function δ (·) is shown as,

$$f(t,\delta) = \begin{cases} 0, & \Delta St < \delta \\ 1, & \Delta St \geq \delta \end{cases} \tag{3.13}$$

We calculate the prediction accuracy of the model by using,

$$\text{acc} = \frac{1}{T+1}\sum_{t=0}^{T} I(f(t;\delta) = y_t) \tag{3.14}$$

Using the simple threshold method to test data generated in the above test, we get the relationship between the threshold and the correct rate as shown as Figure 3.14. When the threshold is 7×10^{-4}, the prediction accuracy is about 91.9%, which reaches its maxima.

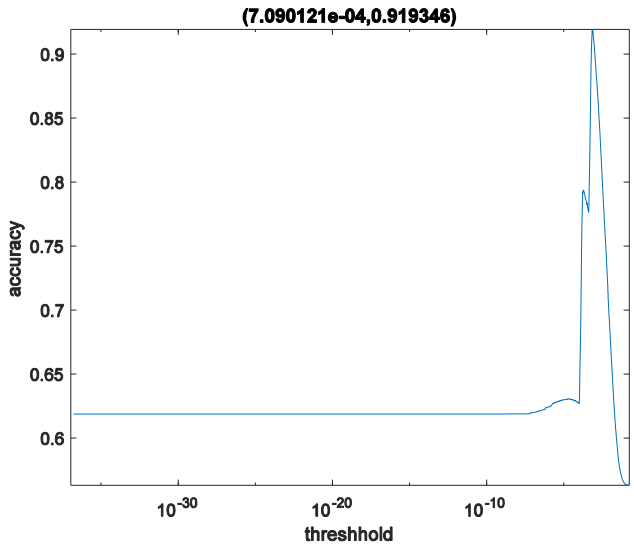

Figure 3.14 The relationship between threshold and accuracy

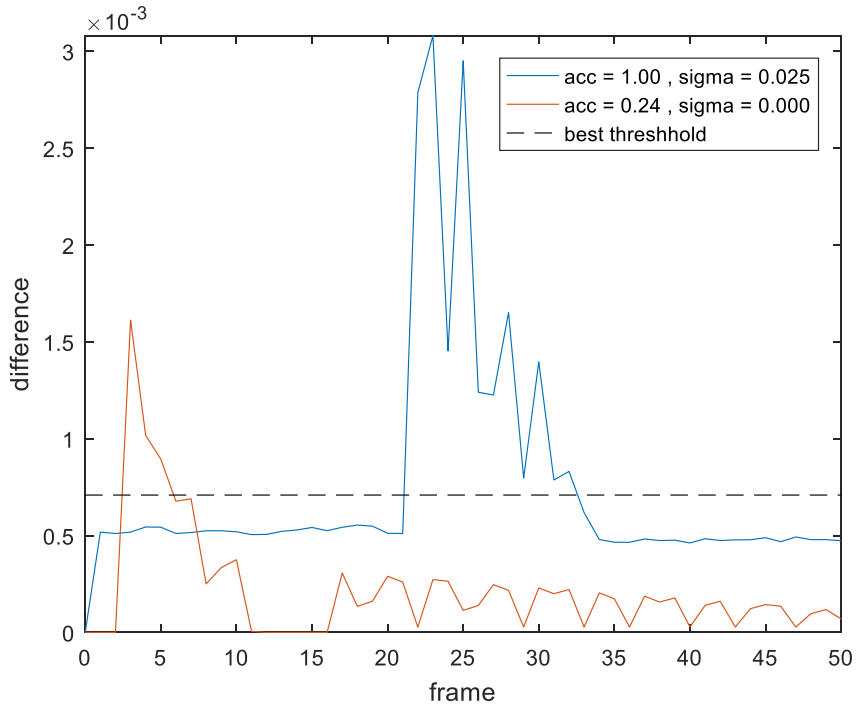

Figure 3.15 Comparison of the maximum correct rate and the minimum correct rate of the video

Comparing the video prediction accuracy between the highest and the lowest, we get the results shown as Figure 3.15. The figure shows time-varying DS_t curves of two videos. Because the background noise is large, when the video that the blue line represents is in a static state, the difference is 0.5×10^{-3}. When it starts to move, the difference exceeds our optimal threshold 0.7×10^{-3}; hence, its predictive accuracy is 100%. While the noise of the video that the red line represents is 0, the actual moving time is between the 4 ~ 11 frame and 18 ~ 51 frame. But because the changes are smaller when it moves and most of them are lower than the overall optimal threshold, the accuracy is very low.

Therefore, we see that the main reason for the limited effect of the simple threshold method is that the noise of the different videos may be different, make it impossible to correctly judge all the videos with a simple threshold. In addition, the difference within the range of different video frames is one of the reasons for the above problems.

Chapter 4
Results

The main content of this chapter is to collect video data and demonstrate the experimental results. In the end, this chapter will also discuss the limitations of the project.

4.1 Data Collection and Experimental Environment

In the previous section, we pointed out that the difference between noise and pixel intensity limits the simple threshold method. In the DNN model, we use the convolution for image denoising in CNN. In RNN, we achieve more results of calculation than the direct reduction. Hence, the actual effect of the DNN model is verified. After training the model, our accuracy rate reaches 96.64%, which is better than the STM method. The accuracy distributions of the two methods are shown in Table 4.1.

Table 4.1 STM and DNN accuracy rate distribution table

Accuracy	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
STM	0.0067	0.0056	0.0200	0.0167	0.0267	0.0511	0.0722	0.8011
DNN	0.0000	0.0011	0.0011	0.0067	0.0089	0.0089	0.0367	0.9367

The k-th data in Table 4.1 indicates the ratio within the correct rate in the range of $(a_{k-1}, a_k]$, and the first column of data indicates the proportion of the samples with the correct rate in the range of $[0, a_1]$.

The DNN correct rate is less than STM in the range of [0, 0.9] and the correct rate is greater than the STM in the range of (0.9, 1.0). Therefore, DNN not only improves the prediction accuracy as a whole, but also reduces the possibility of the emergence of low-accuracy results.

In addition, comparing with the results of the STM and DNN, we get the results shown as in Figure 4.1. The difference between the correction rate of DNN and STM is plotted. For most cases, the correction rate of DNN is greater than that of STM. The increase of correction rate is between [0.0.7] and, for some samples, the correct rate of DNN is lower than that of STM. Most of the declining ranges are around 0.02, and only a few examples have dropped by 0.1. Therefore, the DNN algorithm improved the accuracy rate of most samples greatly with the exception of a small number of samples.

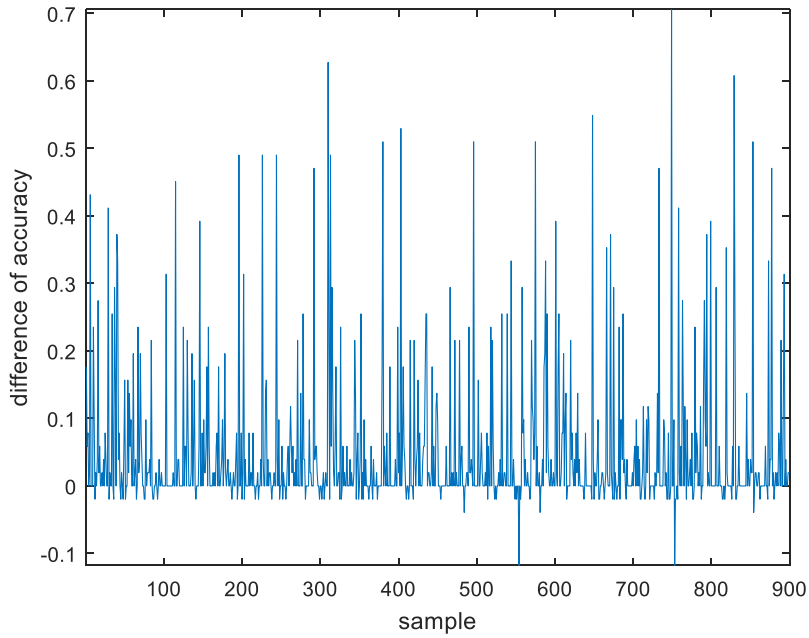

Figure 4.1 Correct rate of DNN and STM paired comparison diagram

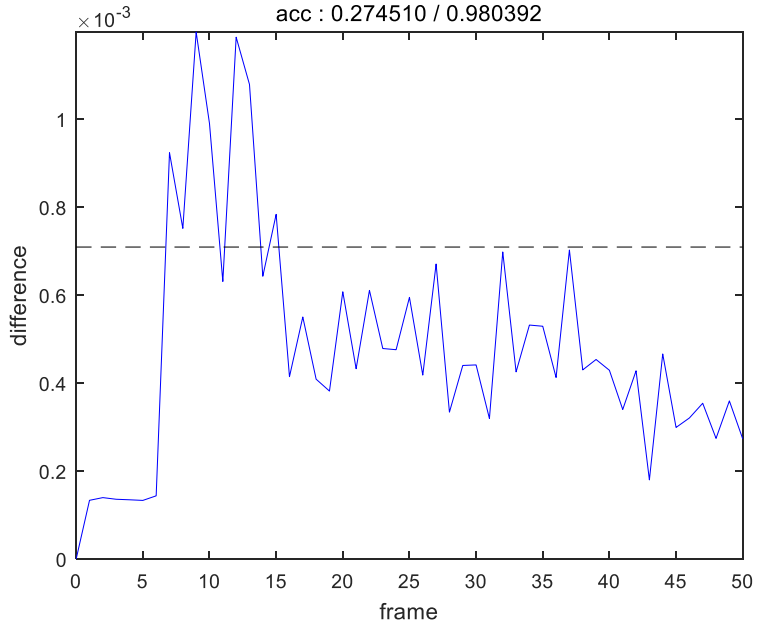

Figure 4.2 The difference curve of the sample with the correct rate raising most

Figure 4.3 shows the most accurate samples. We see that there are two moving objects, and there is not a big difference between the colors and difference between backgrounds of two moving objects. Figure 4.2 shows the difference curve of the samples, and it shows that the change between the samples is small, results in a lower accuracy:

27.45%. Meanwhile, the DNN is less affected by the colors of the moving object and it extracts the video motion information correctly as the correct rate reaches 98.04%.

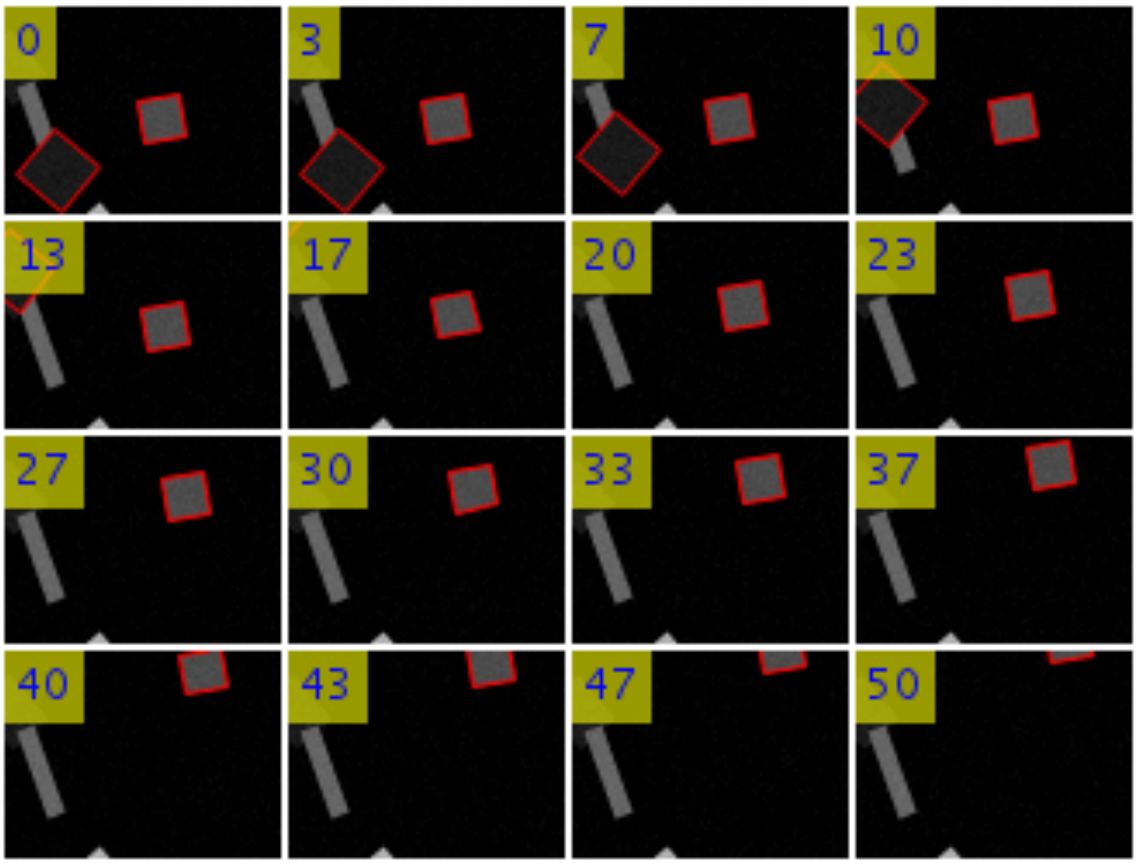

Figure 4.3 Most accurate samples

Figure 4.5 shows the samples of the lowest correct rate, with only one moving object in the sample and the duration is shorter. Figure 4.4 shows the differential curve of the sample. It shows that the samples have three peaks after a short movement and since the peak is less than the optimal threshold of STM, the predictive accuracy of STM is 100%. DNN sees the three small peaks as motion and the total length is five frames, so the correction rate decreased around 10%. After the verification, because of the boundary smoothing algorithm of MATLAB in drawing rectangle and the impact of random noise, there are three boundaries offset of 1 pixel, which rises to 3 peaks in the differential curve and an increase in DNN mis-judgement rate.

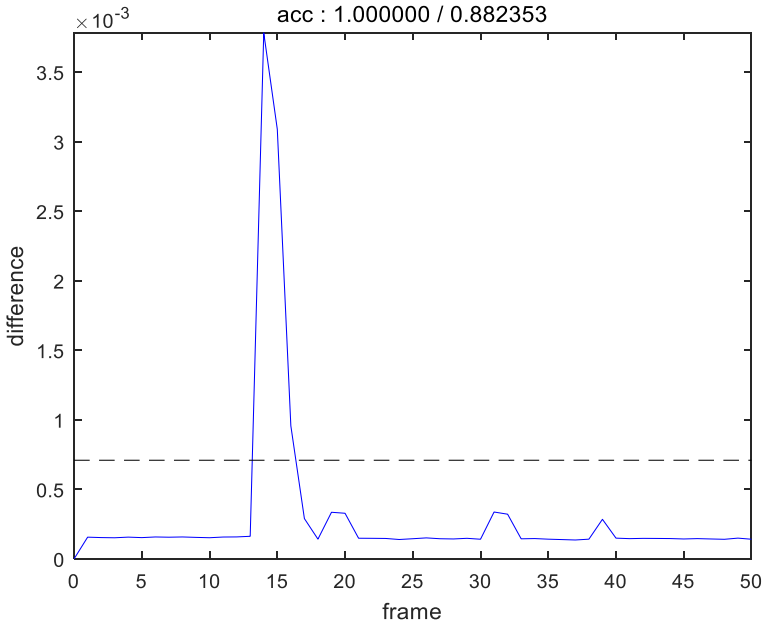

Figure 4.4 Differential curve of the sample

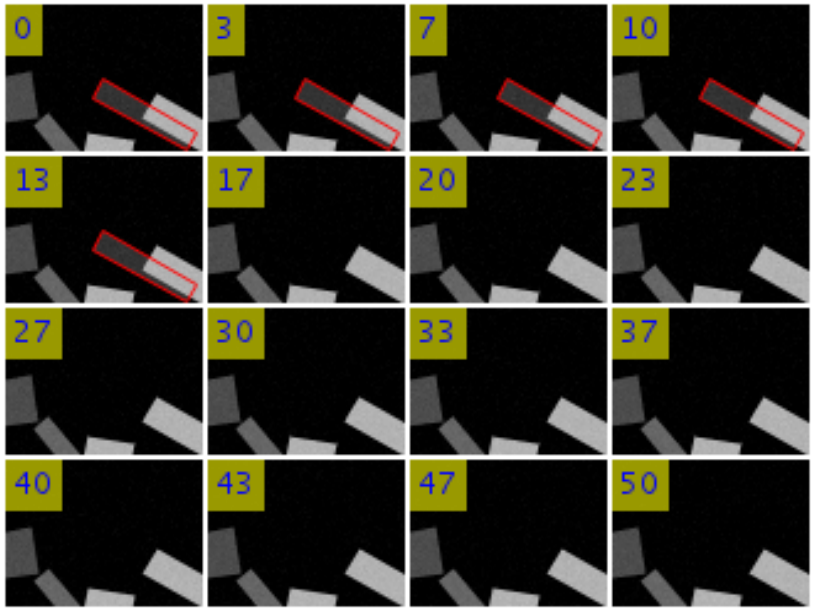

Figure 4.5 The samples of the lowest correct rate

Now, we analyze the difference of correction rate between STM and DNN algorithms under different parameters. First, we analyze the effect of noises. As we see from Table 4.2, under different variances, the part of accuracy rate of DNN algorithm under 0.9 is less than that of STM, meanwhile the proportion of the correction rate above 0.9 is more than that of STM. With the increase of noises, both STM and DNN algorithms are achieving satisfactory results. The proportion of the accuracy below 0.9 decreases and the proportion of the accuracy above 0.9 increases. DNN is less affected by the noises

than that of STM.

Table 4.2 Effect of noises on accuracy rate

Accuracy	$s = 0$		$s = 0.0125$		$s = 0.025$	
	STM	DNN	STM	DNN	STM	DNN
0.5	0.0500	0.0033	0.0333	0.0033	0.0133	0.0000
0.6	0.0233	0.0067	0.0133	0.0100	0.0133	0.0033
0.7	0.0300	0.0100	0.0300	0.0100	0.0200	0.0067
0.8	0.0733	0.0133	0.0567	0.0100	0.0233	0.0033
0.9	0.0833	0.0400	0.0967	0.0400	0.0367	0.0300
1.0	0.7400	0.9267	0.7700	0.9267	0.8933	0.9567

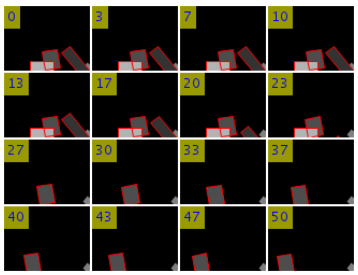

Figure 4.6 $s = 0$

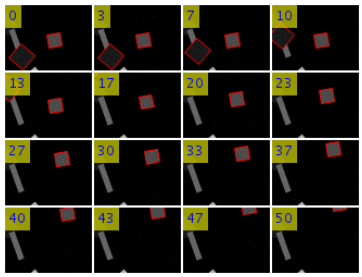

Figure 4.7 $s = 0.0125$

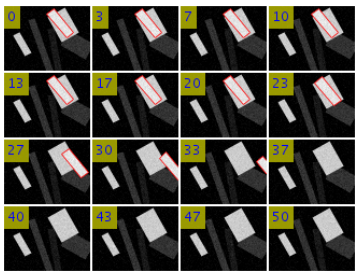

Figure 4.8 $s = 0.025$

Figures 4.6 ~ 4.8 show the samples with the biggest improvement in accuracy under three different noise backgrounds. When s=0, there is no noise in the picture; when s =0.0125, we have to enlarge the picture to see the noise; and when s=0.025, each color block has a lot of visible noises.

In Figure 4.9, we have the differential curves in three cases. In these cases, there is an amount of exercise time and the difference is below the optimal threshold, which results in poor STM effects, but DNN can acquire better results.

Then, the influence of the numbers of moving objects are analyzed and the results are shown in Table 4.3. Overall, the DNN algorithm always satisfies different numbers of moving objects. The proportion of the correction rate below 0.9 is less than that of STM, and the proportion of the correction rate above 0.9 is greater than that of STM. With the increase of moving objects, the correction rate of both DNN and STM algorithms decreases gradually, but the descent rate of STM is larger and the decrease of DNN is smaller.

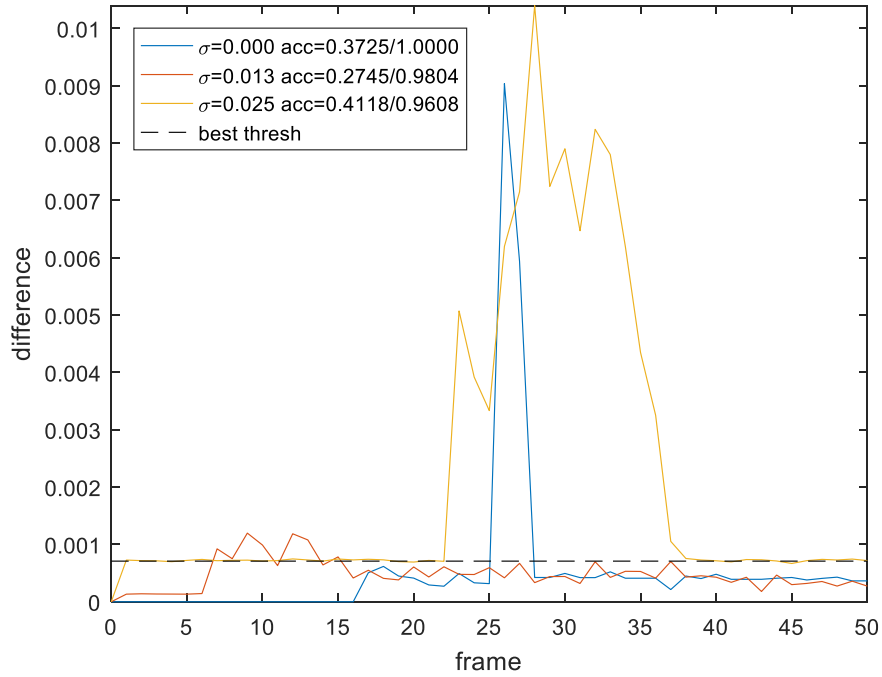

Figure 4.9 Difference curves under different noises

Table 4.3 The influence of the number of moving objects on the accuracy

Accuracy	1		2		3		4		5	
	STM	DNN	STM	DNN	STM	DNN	STM	DNN	STM	DNN
0.5	0.03	0.00	0.05	0.00	0.04	0.01	0.02	0.00	0.02	0.00
0.6	0.01	0.01	0.02	0.01	0.02	0.01	0.02	0.01	0.01	0.01
0.7	0.02	0.01	0.01	0.02	0.04	0.00	0.03	0.01	0.03	0.01
0.8	0.04	0.01	0.03	0.01	0.07	0.02	0.07	0.01	0.05	0.00
0.9	0.02	0.02	0.07	0.03	0.07	0.04	0.07	0.04	0.13	0.04
1.0	0.88	0.96	0.82	0.93	0.75	0.93	0.78	0.92	0.77	0.94

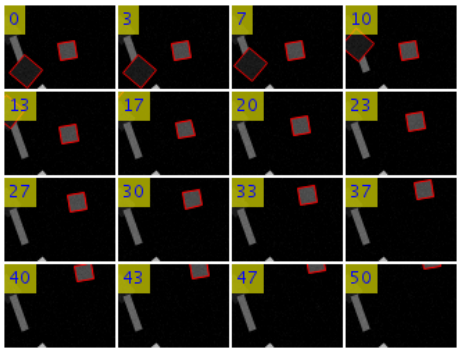

Figure 4.10

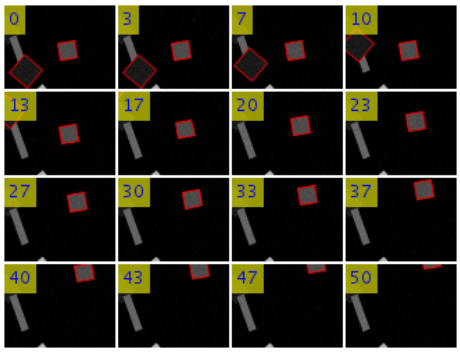

Figure 4.11

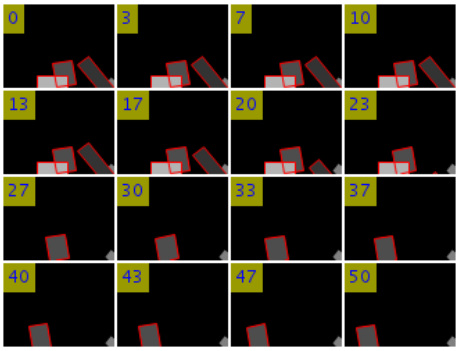

Figure 4.12

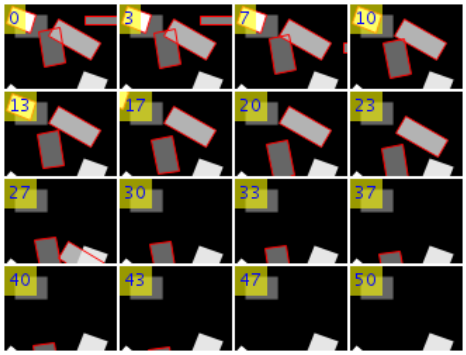

Figure 4.13

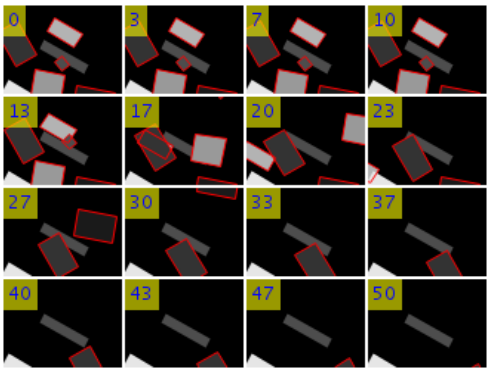

Figure 4.14

Figures 4.10~4.14 show the samples with the biggest improvement in accuracy with a number of moving objects. Figure 4.15 shows the differential curves for these samples. Since most of the differential values of these samples are below the optimal threshold, the STM accuracy is low. The accuracy of STM cannot be improved by simply adjusting the thresholds because even though the correction rate of these samples may be increased by the adjustment, the overall correctness is reduced. However, DNN can overcome this problem and it has better results in different situations.

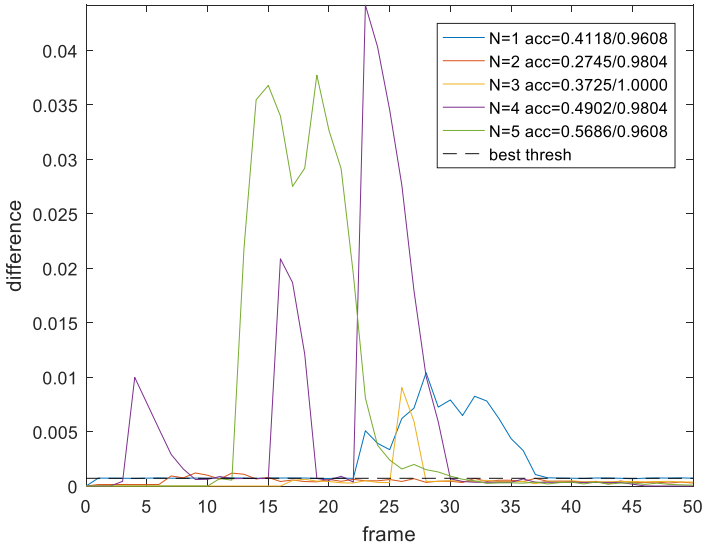

Figure 4.15 The difference curves of the samples with the highest accuracy in the case of different moving objects

Then, the impact of the numbers of background objects on the accuracy is investigated. The results are shown in Table 4.4. Overall, the DNN algorithm is satisfactory even under the condition of many different background objects. The proportion of the correction rate below 0.9 is less than that of the STM, and the proportion of the correction rate above 0.9 is greater than that of STM. With the increase of the numbers of the background objects, the accuracy rate of STM and DNN algorithms remains essentially unchanged. So, the accuracy rate of the STM algorithm and the DNN algorithm is independent on the number of background objects.

Table 4.4 The influence of the number of background objects on the accuracy

Accuracy		0.5	0.6	0.7	0.8	0.9	1
0	STM	0.03	0.02	0.01	0.05	0.07	0.81
	DNN	0.01	0.00	0.01	0.00	0.04	0.94
1	STM	0.04	0.00	0.03	0.06	0.05	0.83
	DNN	0.00	0.01	0.00	0.01	0.04	0.95
2	STM	0.01	0.05	0.02	0.03	0.06	0.83
	DNN	0.00	0.01	0.01	0.02	0.01	0.95
3	STM	0.04	0.02	0.04	0.06	0.10	0.74
	DNN	0.00	0.00	0.02	0.01	0.04	0.93
4	STM	0.03	0.01	0.01	0.05	0.07	0.83
	DNN	0.01	0.01	0.01	0.00	0.04	0.93
5	STM	0.04	0.01	0.05	0.05	0.09	0.77
	DNN	0.00	0.02	0.00	0.01	0.05	0.92

Figures 4.16-4.22 shows the samples of the biggest improvement in accuracy with several background objects. Figure 4.23 shows the corresponding differential curves of the samples.

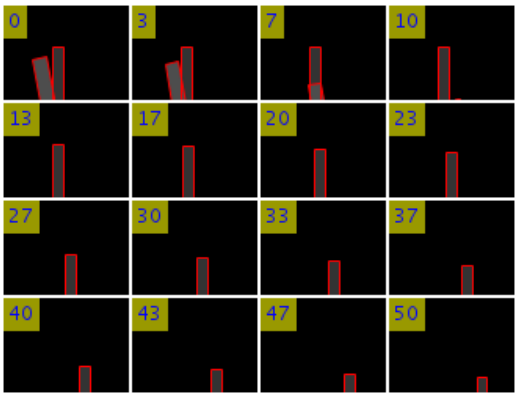

Figure 4.16

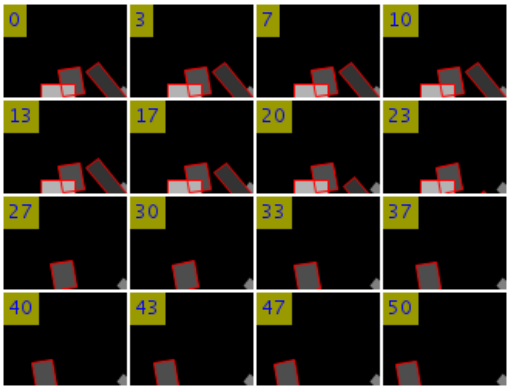

Figure 4.17

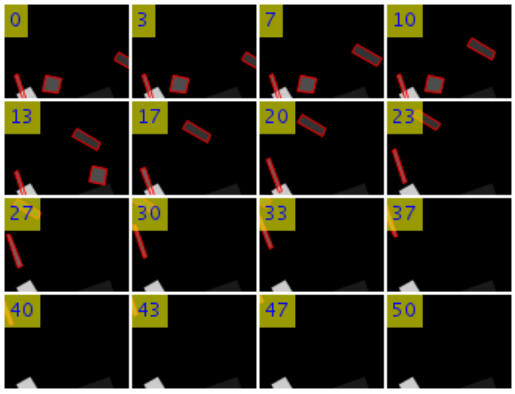

Figure 4.18

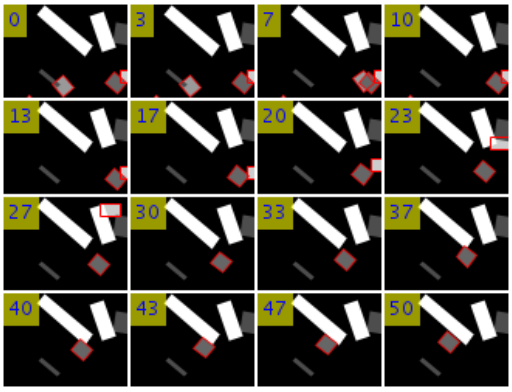

Figure 4.19

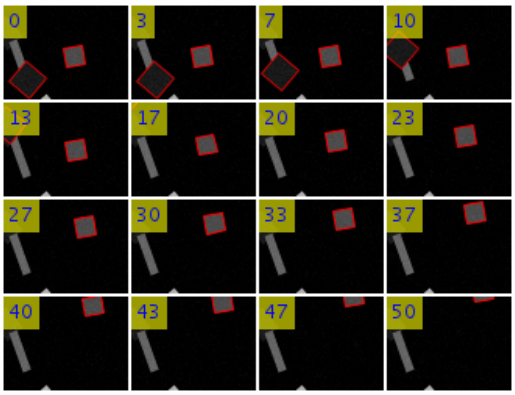

Figure 4.20

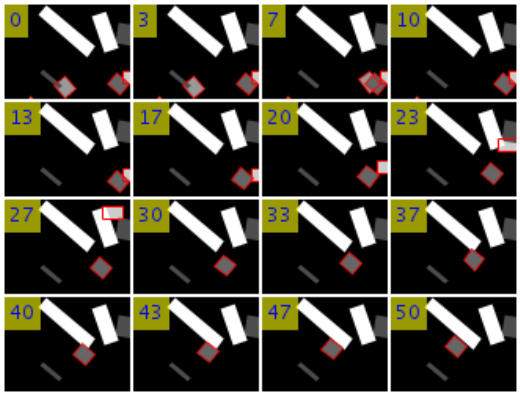

Figure 4.21

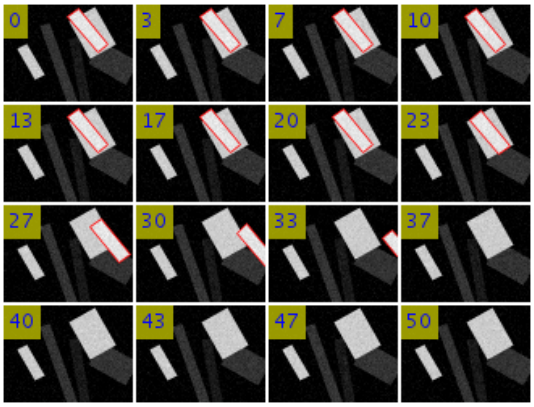

Figure 4.22

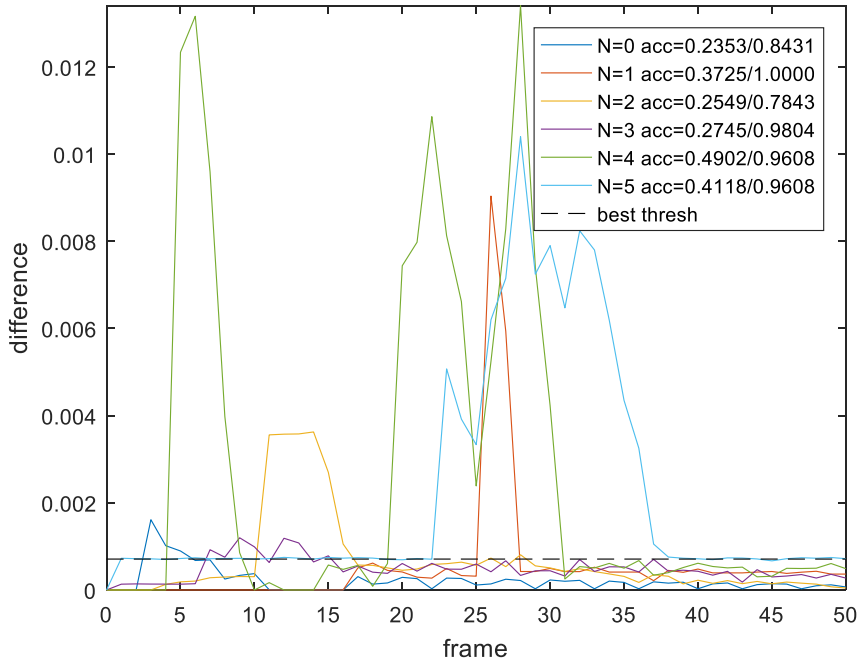

Figure 4.23 The difference curve of the sample with the highest correct rate in the case of different quantity background objects

In addition, the color of moving objects is also one of the important factors affecting the accuracy rate. We analyze the samples with only one moving object and get the results shown as Table 4.5. The DNN algorithm is always satisfactory under the condition of a number of different background objects, the ratio of the correction rates below 0.9 is less than that of STM, and the correction rate above 0.9 is greater than that of STM. With the increase of the brightness of moving objects, the correction rate of STM and DNN algorithm is significantly increased. Therefore, the greater the difference between the moving object and the background is, the higher the correction rate of the STM and DNN

algorithm is. When *c* is above 0.3, the correct rate above 0.9 of DNN is 100%.

Table 4.5. Effect of color of moving object on correct rate

Accuracy		0.5	0.6	0.7	0.8	0.9	1.0
0.1	STM	0.10	0.10	0.05	0.15	0.15	0.45
	DNN	0.00	0.05	0.05	0.00	0.10	0.80
0.2	STM	0.07	0.00	0.00	0.27	0.00	0.67
	DNN	0.00	0.00	0.00	0.00	0.13	0.87
0.3	STM	0.00	0.00	0.00	0.00	0.00	1.00
	DNN	0.00	0.00	0.00	0.00	0.00	1.00
0.4	STM	0.06	0.00	0.00	0.00	0.00	0.94
	DNN	0.00	0.00	0.00	0.00	0.00	1.00
0.5	STM	0.00	0.00	0.00	0.00	0.00	1.00
	DNN	0.00	0.00	0.00	0.00	0.00	1.00
0.6	STM	0.00	0.00	0.10	0.00	0.00	0.90
	DNN	0.00	0.00	0.00	0.05	0.00	0.95
0.7	STM	0.00	0.00	0.00	0.00	0.00	1.00
	DNN	0.00	0.00	0.00	0.00	0.00	1.00
0.8	STM	0.00	0.00	0.04	0.00	0.00	0.96
	DNN	0.00	0.00	0.00	0.00	0.00	1.00
0.9	STM	0.04	0.00	0.00	0.00	0.00	0.96
	DNN	0.00	0.00	0.00	0.00	0.00	1.00
1	STM	0.00	0.00	0.00	0.00	0.00	1.00
	DNN	0.00	0.00	0.00	0.00	0.00	1.00

4.2 Limitations of the Research

(1) Because the data set is simulated, it may be limited in generalization, and the use of real dataset may make the model more accurate.

(2) We only achieved a dynamic video capturing, the functionality is not strong enough, dynamic object tracking and event recognition should be considered in the project.

(3) The reinforcement learning algorithms require a lot of data that is difficult to be implemented.

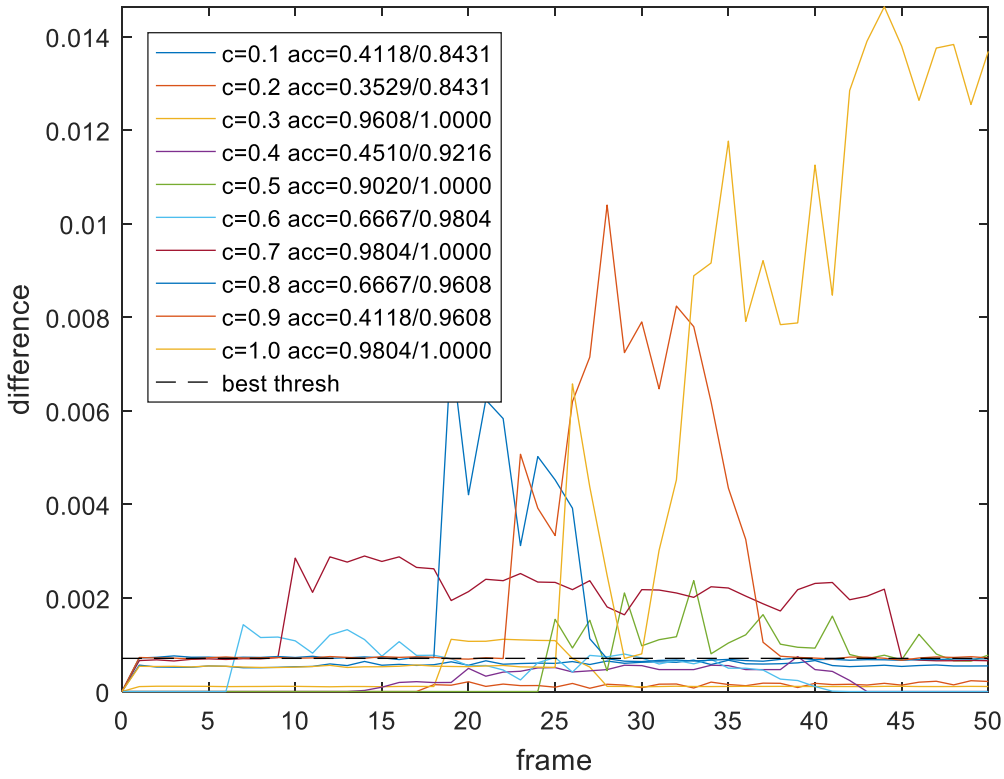

Figure 4.24 The difference curve of the sample with the correct rate of the moving object of different color

Chapter 5
Analysis and Discussions

In this chapter, the experimental results are analyzed and compared. Comparisons of results under different experimental conditions will be mentioned.

5.1 Analysis

In summary, the overall accuracy rate of DNN algorithm reaches 96.64%, which is better than STM algorithm's 91.93%. The DNN algorithm increases the proportion of high-accuracy samples and reduces the proportion of low-accuracy samples. The DNN algorithm greatly improved the accuracy rate of most samples; only a small number of correction rate decreased slightly. Additionally, the DNN algorithm is more stable for the prediction of the videos of different parameters. The results of the analysis of different parameters are summarized as: the greater the difference between adjacent video frames is, the higher the accuracy rate of STM and DNN algorithm will be.

5.2 Discussions

In the experimental part, we compared DNN and STM two algorithms, and analyzed the performance of the two algorithms under different conditions. First, under different noisy conditions, through the data analysis we drw the conclusion: with the increase of noise, both STM and DNN algorithm are able to achieve satisfactory results, but DNN is less affected by noises than STM. Then, the impact of the numbers of background objects on the accuracy is probed. The results show with the increase of moving objects, the correction rate of both DNN and STM algorithm decreases gradually, but the descent rate of STM is larger and the decrease of DNN is smaller. The problem of STM cannot be improved by simply adjusting the thresholds because the correction rate of these samples may be increased by adjusting the threshold, but the overall correctness is reduced. However, DNN overcomes this problem and it has better results in different situations. Then, we run an analysis by the number of background objects, and we find that the accuracy rate of the STM algorithm and the DNN algorithm is independent on the number of background objects. Finally, the colors of moving objects are considered in our project, and a conclusion was drawn that with the increase of the brightness of moving objects, the correction rate of STM and DNN algorithm is significantly increased.

Chapter 6
Conclusion and Future Work

In this chapter, we will summarize the subject and method of this project, and propose new research direction according to the result and insufficiency of the experiment, preparing for the future work.

6.1 Conclusion

The purpose of this report is to propose a new dynamic capture method based on deep learning as we combine CNN and RNN to implement our model. In the report, we also show the contrast of dynamic motion capture based on deep learning and traditional methods, and successfully verify that dynamic motion capture based on deep learning has a great deal of advantages in different conditions.

With the RNN processing, we fully consider the continuity of time series. In addition, we have combined CNN and RNN together to greatly reduce the size of the video data and the time of model training. The results show that the accuracy of dynamic capture based on deep earning reaches 96.64%. The DNN algorithm is more stable for dynamic video capturing and event recording. Finally, dynamic capture based on deep learning is less disturbed by external factors.

6.2 Future Work

We will collect a large number of real videos for the model training which will be more effective to help us improve the accuracy of the model.

We will not only focus on the dynamic motion capturing for videos, but also recognize the action of the objects.

The real-time tracking of moving objects will be considered in our future work.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., & Ghemawat, S. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Abdulrahman, A., & Iqbal, K. (2014). Capturing Human Body Dynamics Using RNN Based on Persistent Excitation Data Generator. In *International Symposium on Computer-Based Medical Systems (CBMS),* (pp. 221-226). IEEE.

Anderson, C. H., Burt, P. J., & Van Der Wal, G. S. (1985). Change detection and tracking using pyramid transform techniques. In *Cambridge Symposium* (pp. 72-78). International Society for Optics and Photonics.

Baum, L. E., & Sell, G. (1968). Growth transformations for functions on manifolds. *Pacific Journal of Mathematics*, *27*(2), 211-227.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, *5*(2), 157-166.

Chatzis, S. P., & Kosmopoulos, D. I. (2011). A variational Bayesian methodology for hidden Markov models utilizing Student's-t mixtures. *Pattern Recognition*, *44*(2), 295-306.

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). IEEE.

Collins, R. T., Lipton, A. J., Kanade, T., Fujiyoshi, H., Duggins, D., Tsin, Y., & Wixson, L. (2000). A system for video surveillance and monitoring.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for

statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Eickeler, S., & Muller, S. (1999). Content-based video indexing of TV broadcast news using hidden Markov models. In *IEEE International Conference on Acoustics, Speech, and Signal Processing.*(Vol. 6, pp. 2997-3000). IEEE

F. Cummins, F. A. Gers, and J. Schmidhuber. (1999) Language identification from prosody without explicit features. In Proceedings of EUROSPEECH'99, volume 1, pages 371–374, 1999

Fu, R., Zhang, Z. and Li, L. (2016). Using LSTM and GRU neural network methods for traffic flow prediction. *Youth Academic Annual Conference of Chinese Association of Automation (YAC)*.

Gers, F. A., & Schmidhuber, J. (2000). Recurrent nets that time and count. In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on* (Vol. 3, pp. 189-194). IEEE.

Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. *Neural computation*, *12*(10), 2451-2471.

Gers, F. A., Schraudolph, N. N., & Schmidhuber, J. (2002). Learning precise timing with LSTM recurrent networks. *Journal of machine learning research*, *3*(Aug), 115-143.

Gers, F. A., & Schmidhuber, E. (2001). LSTM recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks*, *12*(6), 1333-1340.

Gers, F. A., Schraudolph, N. N., & Schmidhuber, J. (2002). Learning precise timing with LSTM recurrent networks. *Journal of machine learning research*, *3*, 115-143.

Haritaoglu, I., Harwood, D., & Davis, L. S. (2000). W/sup 4: real-time surveillance of people and their activities. *IEEE Transactions on pattern analysis and machine intelligence*, *22*(8), 809-830.

Han, X., Gao, Y., Lu, Z., Zhang, Z., & Niu, D. (2015). Research on Moving Object Detection Algorithm Based on Improved Three Frame Difference Method and Optical Flow. In *International Conference on Instrumentation and Measurement, Computer, Communication and Control (IMCCC)* (pp. 580-584). IEEE.

Heikkila, M., & Pietikainen, M. (2006). A texture-based method for modeling the background and detecting moving objects. *IEEE transactions on pattern analysis and machine intelligence*, *28*(4), 657-662.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.

Hori, T., Kubo, Y., & Nakamura, A. (2014). Real-time one-pass decoding with recurrent neural network language model for speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 6364-6368). IEEE.

Hubel, D. H., & Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, *160*(1), 106-154.

Katagiri, S., & Lee, C. H. (1993). A new hybrid algorithm for speech recognition based on HMM segmentation and learning vector quantization. *IEEE Transactions on speech and audio processing*, *1*(4), 421-430.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., & Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (pp. 1725-1732).

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).

Kim, J., & Kim, H. (2016). Classification performance using gated recurrent unit recurrent neural network on energy disaggregation. In *International Conference on Machine Learning and Cybernetics (ICMLC)* (pp. 105-110). IEEE.

LeCun, Y., & Ranzato, M. (2013, June). Deep learning tutorial. In *Tutorials in International Conference on Machine Learning (ICML'13)*.

LeCun, Y. (1989). Generalization and network design strategies. *Connectionism in perspective*, 143-155.

Liu, H., & Hou, X. (2012, August). Moving detection research of background frame difference based on Gaussian model. In *International Conference on Computer Science & Service System (CSSS),* (pp. 258-261). IEEE.

Matsugu, M., Mori, K., Mitari, Y., & Kaneda, Y. (2003). Subject independent facial expression recognition with robust face detection using a convolutional neural network. *Neural Networks*, *16*(5), 555-559.

Maryam, K., & Reza, K. M. (2012). An analytical framework for event mining in video data. *Artificial Intelligence Review, 41*(3), pp. 401-413.

Mikolov, T., Kombrink, S., Burget, L., Černocký, J., & Khudanpur, S. (2011). Extensions of recurrent neural network language model. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* (pp. 5528-5531). IEEE.

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., & Khudanpur, S. (2010). Recurrent neural network based language model. In *Interspeech* (Vol. 2, p. 3).

Narendra, K. S., & Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks. *IEEE Transactions on neural networks*, *1*(1), 4-27.

Noorit, N., & Suvonvorn, N. (2014). Human activity recognition from basic actions using

finite state machine. In *Proceedings of the First International Conference on Advanced Data and Information Engineering (DaEng-2013)* (pp. 379-386). Springer.

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. *ICML (3)*, *28*, 1310-1318.

Petrushin, V. A. (2005). Mining rare and frequent events in multi-camera surveillance video using self-organizing maps. *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pp. 794-800.

Popoola, O. P., & Wang, K. (2012). Video-based abnormal human behavior recognition—A review. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *42*(6), 865-878.

Rabiner, L., & Juang, B. (1986). An introduction to hidden Markov models. IEEE *ASSP magazine*, *3*(1), 4-16.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*(2), 257-286.

Remagnino, P., Monekosso, D. N., & Jain, L. C. (Eds.). (2011). *Innovations in Defence Support Systems-3: Intelligent Paradigms in Security* (Vol. 336). Springer Science & Business Media.

Sak, H., Senior, A., & Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Sak, H., Senior, A., & Beaufays, F. (2014). Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128*.

Schwenk, H. (2007). Continuous space language models. *Computer Speech & Language*, *21*(3), 492-518.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(1), 1929-1958.

Tang, Y., Huang, Y., Wu, Z., Meng, H., Xu, M., & Cai, L. (2016). Question detection from acoustic features using recurrent neural network with gated recurrent unit. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 6125-6129). IEEE.

Trinh, H., Fan, Q., Jiyan, P., Gabbur, P., Miyazawa, S., & Pankanti, S. (2011). Detecting human activities in retail surveillance using hierarchical finite state machine. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 1337-1340). IEEE.

Sak, H., Senior, A., & Beaufays, F. (2014). Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128*.

Sisson, J. H., Stoner, J. A., Ammons, B. A., & Wyatt, T. A. (2003). All-digital image capture and whole-field analysis of ciliary beat frequency. *Journal of microscopy*, *211*(2), 103-111.

Wildes, R. P. (1998). A measure of motion salience for surveillance applications. In *International Conference on Image Processing* (pp. 183-187). IEEE.

Varol, G., Laptev, I., & Schmid, C. (2016). Long-term temporal convolutions for action recognition. *arXiv preprint arXiv:1604.04494*.

Xingjian, S. H. I., Chen, Z., Wang, H., Yeung, D. Y., Wong, W. K., & Woo, W. C. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems* (pp. 802-810).

Yue-Hei Ng, J., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., & Toderici, G. (2015). Beyond short snippets: Deep networks for video classification. In

Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4694-4702).

Xiaoyang, Y., Yang, Y., Shuchun, Y., Yang, S., Huimin, Y., & Xifeng, L. (2013). A novel motion object detection method based on improved frame difference and improved Gaussian mixture model. In *International Conference on Measurement, Information and Control* (Vol. 1, pp. 309-313). IEEE.

Zaremba, W., Sutskever, I., & Vinyals, O. (2014). Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Zaremba, W. (2015). An empirical exploration of recurrent network architectures.

Zhang, Y., Er, M. J., Venkatesan, R., Wang, N., & Pratama, M. (2016). Sentiment classification using Comprehensive Attention Recurrent models. In *International Joint Conference on Neural Networks (IJCNN)* (pp. 1562-1569). IEEE.