Anomalies Detection and Tracking Using Siamese Neural Networks

Na An

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

2020

School of Engineering, Computer and Mathematical Sciences

Abstract

In this thesis, we detect and track anomalies on the sidewalk using deep learning. The proposed network consists of two parts: The first part is an object detection network, namely, SSD(Single Shot MultiBox Detector) is employed to detect and classify objects, then we get the abnormal targets. The second one is to find data association of objects. The proposed model is based on the single-target tracking network SiamRPN, which assists multi-target tracking through a cyclic structure. We follow Hungarian algorithm for getting the final matching results. Both the networks are trained offline, their performance is well.

The contributions of this thesis are: (1) We implement the proposed model for object recognition, classification, and tracking for multiple types of anomalies. (2) We achieve multi-target tracking by combining our object detection algorithm and single-target tracking algorithm. (3) The proposed model is a novel type of deep neural networks to achieve anomaly detection, which has not been found in previous work.

Keywords: SSD, SiamRPN, SiamFC, ResNet50, AlexNet, LSTM, Hungarian algorithm

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: Date: 15 May 2020

Acknowledgment

This research work 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) with the Auckland University of Technology (AUT) in New Zealand.

My deepest thanks are to my supervisor Dr. Wei Qi Yan who gave me a lot of suggestions and guidance. Dr. Yan explained the knowledge of deep learning at the beginning, continuously corrected my mistakes throughout the experiments, and urged me to complete the experiment with quality and quantity. Without Dr. Yan 's supervision, I can't complete this thesis. In addition, I would like thanks to our administrators in our school for their support and guidance through the MCIS in the past years.

I also want to thank my classmates Ms Y. Fu, Mr J. Lu, Mr Y. Xiang, and Mr S. Sun. They gave me invaluable suggestions through the group discussion.

Na An

Auckland, New Zealand

15 May 2020

Chapter 1
Introduction

This chapter introduces the background of anomaly detection in video surveillance and explains the reason why we launch this thesis. We review the existing work about related issues and proposed our idea to solve this issue. In addition, the main contributions of this thesis are summarized. Moreover, the structure of this paper is also stated.

1.1 Background

In intelligent monitoring systems, abnormal events are captured in real time. When abnormal events or potential threat events occur, the systems will automatically notify the security staff to respond quickly and take corresponding measures to reduce the loss caused by accidents or avoid the occurrence of accidents. This can greatly improve the level of public security, enhance the sense of security of citizens, reduce the labor intensity of security personnel, and alleviate public losses. Abnormal event detection is a key part of intelligent surveillance systems. Studying abnormal event detection technology to improve its performance and time efficiency has become an important research and practical issue.

The definition of anomaly depends on the content of the videos. Generally, small probability events in the video scene are regarded as abnormal ones. Anomalies can be grouped into global anomalies and local anomalies (Feng, Yuan, & Lu, 2016). Global anomalies mean that the group behavior of the entire scene is abnormal (Parekh, Thakore & Jaliya, 2014). This type of anomalies occur from the beginning of a frame of video sequence in terms of the entire frame of the scene, such as crowded panic running in UMN dataset (Raghavendra, et al. 2011) and the violent behavior scene in Hockey Fight dataset (Xu, et al. 2014). Local anomaly refers to that only the individual behavior in a certain area is different from the neighboring crowd or most behaviors in the entire scene, such as the behavior of cycling in the UCSD dataset (Chan, et al. 2008).

At present, anomalous event detection technology has been widely studied worldwide and primary applications in specific places. In 1997, the Defense Advanced Research Projects Agency (DARPA), Carnegie Mellon University (CMU), Massachusetts Institute of Technology (MIT), and other institutions of higher education started the cooperation (Djenouri, et al. 2019). Visual Surveillance and Monitoring (VSAM) applies computer vision and pattern recognition technology to comprehensively monitor abnormal on the battlefield (Collins, et al. 2000).

Moreover, at the beginning of the 20th century, EU countries launched the Context Aware Vision using Image-Based Recognition (CAVIAR), which mainly analyzes the behavior of customers in shopping malls, and monitors anomalies such as robbery, theft, and crowd gathering in downtown areas (Murray and Basu, 1994). Besides, international conferences on computer vision such as CVRP (IEEE Conference on Vision and Pattern Recognition) have treated abnormal event detection as an important research topic. The well-known journals such as International Journal of Computer Vision (IJCV), have published many papers related to abnormal event detection (Sultani, Chen and Shah, 2018).

Surveillance video anomaly detection is based on machine learning in an early stage. Usually requires three steps (Abdel-Aziz, et al. 2013). The first step is foreground segmentation and moving targets detection. Traditional moving target detection methods include inter-frame difference, background subtraction, and optical flow (Aslani and Mahdavi-Nasab, 2015). The frame difference method is to determine the changes of the grayscale values of the corresponding pixels between adjacent frames to detect the moving object. The background subtraction method needs to model the background so as to get the background model and compare each frame image with the background model image. Optical flow method is the popular method in moving target detection. It is defined as the apparent motion of the image brightness mode in a video frame sequence (Denman, Fookes, & Sridharan, 2009), that is, the moving speed of the point on the surface of the space object is expressed on the imaging plane of the visual sensor.

The second step is feature extraction. Feature extraction is grouped into two categories: The first one is to extract artificial design features, the other is to directly learn the original video frames to obtain deep features, both feature extraction methods are based on biological neural theory. The difference is that the features extracted by the manual design methods are imitated by the human visual framework, and the feature extraction method of deep learning focuses on learning the distribution rules of the data

itself.

Image features include texture features, colors, motion-scale invariant feature transform (SIFT), optical flow characteristics, trajectory characteristics, etc. proposed Mixtures of Dynamic Textures (MDT) was proposed to model the behavior of normal people and employed it to distinguish abnormalities in the discriminant space from normal events (Mahadevan, 2010). Also, a spatiotemporal texture model was proposed with rich crowd pattern features (Wang, et al. 2015), the extracted surveillance texture was subjected to behavior template matching in the feature space based on redundant wavelet transform to achieve anomaly detection. On the other hand, the Grey-Level Co-occurrence Matrix (GLCM) was passed from a statistical point of view which describes the spatial characteristics of abnormal events or objects such as contrast, correlation, uniformity, etc. to construct an abnormal frame that uses spatiotemporal coding to detect abnormal wandering behavior in the crowd (Rao, et al., 2015).

The rapid development of deep learning and convolutional neural networks has provided new ideas for various studies in the field of computer vision. A few methods based on deep representation are to extract high-level feature information with deep neural networks first, and traditional classification methods are offered for the classification. For example, an architecture called AMDN is split into three channels including appearance, motion, fused channel information (Xu, et.al, 2015). The sparse representation of video features was obtained by SDAE, abnormal events were detected by one-class Support Vector Machine (SVM). Moreover, a deep representation-based algorithm was proposed to extract features in an unsupervised fashion (Feng, 2016).

The third step is to implement the identification and location of abnormal events. The classification methods include supervised, semi-supervised, and unsupervised. The supervised classification methods include support vector machines (Hsu and Lin, 2002). A feature selection and support vector machine training hybrid optimization model was proposed by Miao, et al., in 2014. This model improved the accuracy of surveillance video anomaly detection by quickly obtaining the optimal features and SVM

parameters.

Semi-supervised learning makes use of normal samples for model construction because abnormal samples always deviate from the model composed of normal samples. Gaussian mixture model (Reynolds, 2009), Markov random field (Cross and Jain, 1983), a hidden Markov model (Beal, Ghahramani and Rasmussen, 2002) are typical models. crowd distribution information and crowd speed information were proposed to estimate the parameters of the Gaussian mixture model constructed by using normal behaviors and abnormal crowd behaviors (Gu, et al., 2015).

The unsupervised detection method is a typical clustering approach, relying solely on the connection between sample data to complete the clustering and modelling of normal events, the small probability or very low the event is regarded as an abnormal event. The non-negative matrix factorization learning method was proposed from the feature space to detect anomalies in the feature space (Lu, et al., 2013).

With the development of deep learning technology, in addition to deep features extraction, anomaly detection models based on deep learning algorithms, e.g., Long Short-Term Memory (LSTM), has become a hot research topic. For example, an LSTM-based algorithm was proposed by Lotter in 2016 which can detect global anomalies in surveillance videos. They explore the prediction of future frames in a video sequence as an unsupervised learning rule for learning the structure of the visual world. Each layer in the network performs local predictions, and only forwards deviations from these predictions to subsequent network layers.

1.2 Motivation

In this thesis, anomalous behavior is defined as the events associated with objects like bicycles, scooters, and motorcycles appeared on the sidewalk. As demonstrated in Figure 1.1, bikes, scooters and even motorcycles are being ridden on the sidewalk. Additionally, there are cars which are driven on the sidewalk to cut corners or avoid

traffic jams.

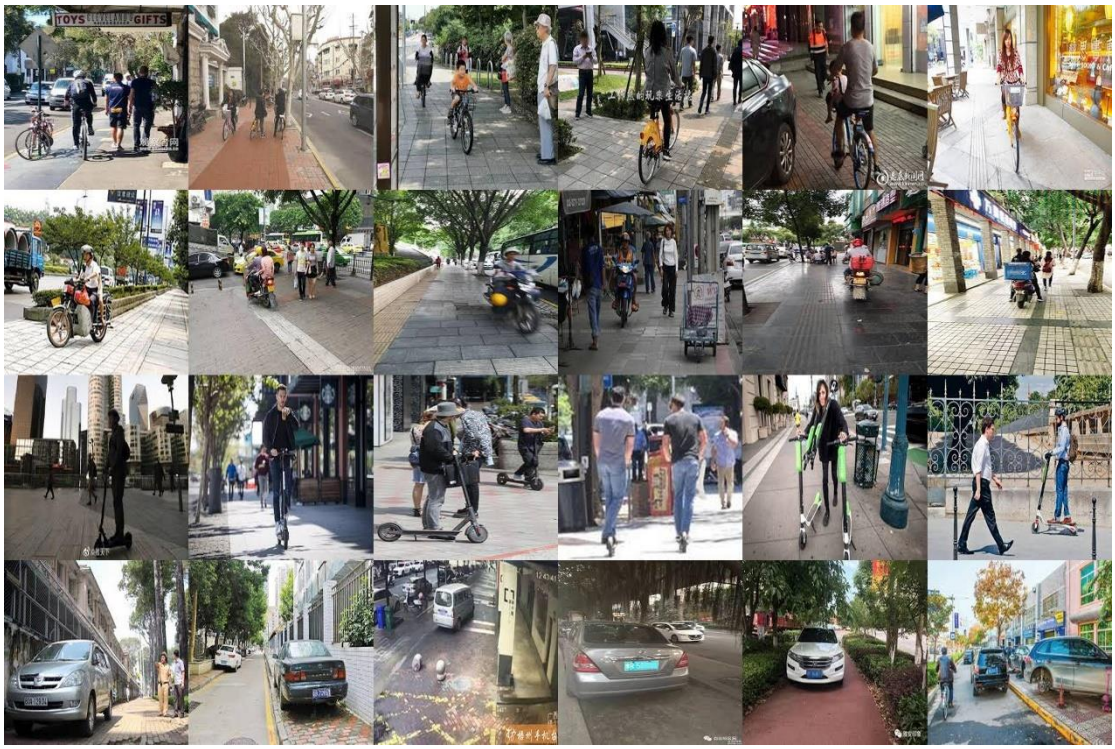

Figure 1.1 The images of abnormal events on the sidewalk

The motivation of this work is to propose a deep learning model of anomaly monitoring in video surveillance.

This experiment was inspired by the PreNet (Lotter, et al., 2015). It could detect the probability of an anomalous event occurring in a frame of video, however, it cannot locate and track anomalous objects. In this thesis, we focus on local anomaly and conduct a deep neural network for object detection and tracking. We did not treat anomaly detection as a binary classification problem. Instead, we tried to identify all objects in the video and group them into multiple categories, then tracked the abnormal objects. Accordingly, we break down this issue into two tasks: Objects detection and classification, abnormal target tracking.

Regarding objects detection and classification, SSD is the state-of-the-art algorithm in the field of object recognition and can be applied to real-time monitoring (Yadav and Binay, 2017).

On the other hand, for abnormal target tracking task, SiamRPN is a deep feature-based tracker trained offline and has achieved better performance than the most

advanced correlation tracking algorithms (Liu, et al., 2019). However, as a single target tracking algorithm, SiamRPN cannot solve the problems of the disappearance of old targets and the appearance of new targets. We solve this problem by combining the results of SSD and SiamRPN. Overall, this thesis is dedicated to detect and track abnormal objects by combining the SSD for target recognition and the SimaRPN for target tracking.

1.3 Research Questions

The research questions of this thesis are listed as:

- Can we classify and recognize anomaly and track them at the same time?
- Can deeper neural networks improve the performance of anomaly recognition and tracking?

1.4 Contributions

The main contributions of this thesis include:

- In this thesis, we propose a novel model to realize the recognition, classification and tracking of video surveillance anomalies at the same time. This model consists of two parts, namely, detection network and tracking network. Detection network is responsible for objects identification and classification. The tracking network is responsible for tracking abnormal objects.
- Existing surveillance video anomaly tracking algorithms are grouped into two categories. One category is to detect global anomalies. These algorithms treat anomaly detection as a binary classification problem that detects the anomaly and normal. The second category is to detect local anomalies, most of these algorithms track different objects in the same category, no classification is achieved. Our experiment achieved the classification of different objects.
- In recent years, single-target tracking algorithms based on deep learning have made great progress. The application of deep learning in the field of multi-target

tracking is limited, most algorithms are combined with deep features and machine learning, the application of deep learning algorithms in multi-target tracking problems is far away from being fully studied. Our work achieves multi-target tracking by combining single target tracking algorithm with an object detection algorithm together.

1.5 Structure of This Thesis

The thesis is structured as:

In Chapter 2, basic algorithms are introduced for object recognition, including R-CNN, Fast R-CNN, Faster R-CNN, YOLO and SDD. The differences between these algorithms and the shortcomings of each of them are analyzed. The multi-objects tracking algorithms are briefed, in terms of Siamese model-based network, the minimum multi-cut graph model-based network, time-domain attention model-based network, and LSTM model-based network.

In Chapter 3, the method is detailed for implementation. This includes how the data is collected, data augmentation methods, data labelling methods, algorithm design, model implementation and operating environments, as well as model evaluation methods.

In Chapter 4, we show the result of the experiments. The detection performance of each class in the SSD algorithm is compared with that of the SSD algorithm. The tracking accuracy and precision of each class are calculated, the comparison of tracking networks with different backbone.

In Chapter 5, the differences between SSD300 and SSD512 networks in objects detection are justified, the feature maps that extracted from different layers of the two algorithms are compared. Additionally, we analyze the performance of tracking networks with different backbones.

Finally, Chapter 6 sums up the process and the result of this experiment firstly, then summarizes the deficiencies of this work and visions possible future work.

Chapter 2 Literature Review

Chapter 2 introduces basic algorithms for object recognition, including R-CNN, Fast R-CNN, Faster R-CNN, YOLO, and SDD. We analyze the differences between these algorithms and the shortcomings of each. We detail the multi-target tracking algorithms, in terms of Hungarian algorithm and Kalman filter, Siamese model-based network, the minimum multi-cut graph model-based network, time-domain attention model-based network and LSTM-based network.

2.1 Introduction

In this chapter, we will introduce the related work from two fields: Object detection and multi-target tracking. Because the proposed algorithm consists of two parts, detection network and tracking network. The detection network is based on object detection and classification. The tracking network is based on multi-objects tracking.

2.2 Object Detection and Classification

Visual object detection algorithm based on deep learning is mainly grouped into two categories (Yanagisawa, Yamashita, & Watanabe, 2018). R-CNN series, in terms of R-CNN, Fast R-CNN, and Faster R-CNN, have two-stage, these algorithms need region proposals, export bounding boxes and classes. They are all region-based algorithms, the distinction is how they utilize CNN feature and how they share the calculation of ROI.

On the other hand, YOLO and SSD are the one-shot solutions that the prediction of bounding boxes is completed in one step. R-CNN-based object detection algorithms rely to regions to locate the visual objects within the image. The networks do not look at the entire image. Instead, the parts of the image having high probabilities containing the object will be scanned.

2.2.1 R-CNN

In traditional machine learning-based object detection. An exhaustive method is adopted for area selection. Using a sliding window with different sizes and aspect ratios to traverse the image has high complexity. Then, the feature extraction is performed, which includes Scale Invariant Feature Transform (SIFT) (Hsu, Lu and Pei, 2012), Histogram of Oriented Gradients (HOG) (Mizuno, et al., 2012), and other features. The robustness relies on morphological diversity, light variation diversity, and background diversity. Finally, the classifiers are employed for classification. The popular classifiers

include Support Vector Machine (SVM) (Vishwanathan and Murty, 2002), AdaBoost (Hastie, et al., 2009), etc.

The region selection strategy based on sliding windows is not applied, thus, the time complexity is high, the windows are redundant. On the other hand, the features designed are not very robust to the changes.

A method called region proposal was proposed, which finds out in advance where the objects may appear in the image. The information such as textures, edges, and colors in the image is employed to ensure that fewer windows (thousands or even hundreds) are selected. Therefore, the problem turns to find the regions that may contain objects. These frames can overlap each other and contain each other, we avoid violent enumeration all the boxes. The methods for selecting the Region Proposal include Selective Search and Edge Boxes. We thus classify these candidate regions.

For image classification, a convolutional neural network was proposed to reduce the error of the ILSVRC classification task to 15.3% at the 2012 ImageNet Large-scale Visual Recognition Challenge (ILSVRC). The top-5 error of the traditional method is 26.2%. Since then, CNNs have dominated the image classification task.

R-CNN algorithm makes use of selective search to extract 2,000 regions from the image and these regions are called region proposals (Girshick, et al., 2014). Then, there are 2,000 candidate region proposals which are warped into a square and fed into a convolutional neural network (CNN). The CNN acts as a feature extractor and the output dense layer consists of the features extracted from the image. After that, the extracted features are fed into an SVM to classify the presence of the object within that candidate region proposal.

There are two options for network architecture: The first choice is AlexNet (Yuan and Zhang, 2016); the second one is VGG16. After the test, the accuracy of Alexnet is 58.5%, and the accuracy of VGG16 is 66%. The characteristic of VGG model is to choose a relatively small convolution kernel. The accuracy of this network is high, but the calculation amount is seven times of AlexNet. The architecture of R-CNN is displayed in Figure 2.1.

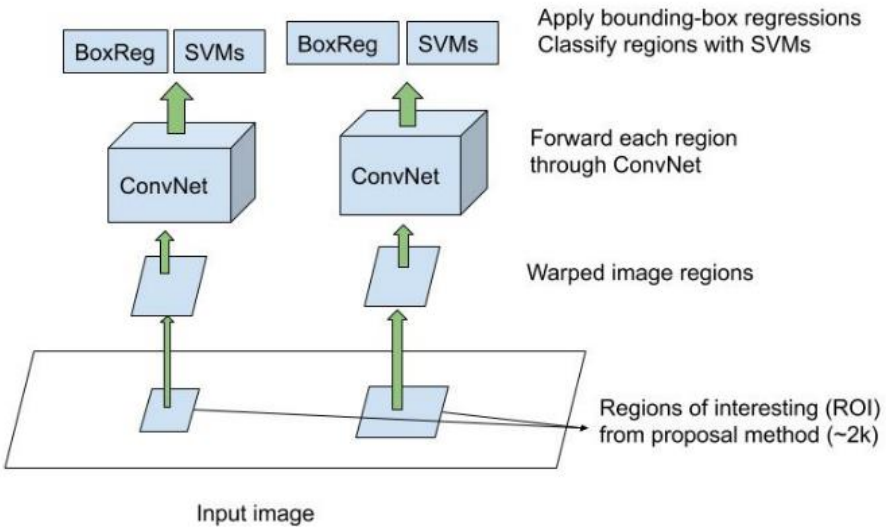

Figure 2.1 The structure of R-CNN structure

Although R-CNN is no longer as exhaustive as the traditional method, in the first step of the R-CNN process, as many as 2,000 candidate region proposals are extracted from the original picture through selective search, each of these 2,000 candidate bounding boxes needs to extract CNN feature maps and conducts SVM classification. The amount of calculation is very large, which leads to that the R-CNN detection speed is very slow, around 47 seconds for each image.

2.2.2 Fast R-CNN

Girshick et al. (2015) proposed Fast R-CNN to overcome the drawbacks of R-CNN. Firstly, the training process of R-CNN is split into multiple steps. R-CNN fine-tunes a pre-trained network first, then trains an SVM classifier for each category, and finally utilizes regressions to regression on the bounding-box. In addition, time and memory consumption are relatively large. By training SVM and regression, it is necessary to exploit the features trained by the network as input, the time consumption for saving the features and re-reading them is relatively large.

Although SPPnet (Spatial Pyramid Pooling net) algorithm (Purkait, Zhao, & Zach, 2017) has been proposed before Fast R-CNN solves the problem of repeated

convolution in R-CNN, SPPnet still has shortcomings, e.g., too many training steps, needs to train SVM classifier, needs of additional regressors saved on disk (Wang, Shrivastava & Gupta, 2017). Therefore, Fast R-CNN is equivalent to comprehensively improve the original two algorithms by not only reducing the training steps but also saving extracted features. The Fast R-CNN algorithm based on VGG16 is nearly nine times faster than R-CNN in training speed, and three times faster than SPPnet.

The approach of Fast R-CNN is similar to the R-CNN algorithm. Instead of feeding the region proposals to the CNN, it feeds the input image to the CNN to generate a convolutional feature map. From the convolutional feature map, it identifies the region of proposals and warps them into squares, then reshapes them into a fixed size by using a ROI pooling layer so that it can be fed into a fully connected layer. Then the class of the proposed region and the bounding box is predicted by using a softmax layer.

The purpose of Region of Interest (ROI) pooling is to extract feature maps with fixed sizes from the last convolution layer with different size of region proposals. It is a simplified version of SPPnet, since the input of the fully connected layer needs to be designed with the same size, it cannot directly map region proposals of different sizes to the feature map as output, thus the size conversion is required. For example, a region proposal with the size $w \times h$ is divided into a grid of size $H \times H$, this region proposal is mapped to the feature map output by using the last convolution layer. Finally, the maxima in each grid are calculated as the output of the grid, regardless of the size of the feature map before ROI pooling, the size of the feature map, obtained after ROI pooling, is $W \times H$. The structure of Fast R-CNN is presented in Figure 2.2.

Fast R-CNN mainly has three improvements: (1) Convolution is no longer performed on each region proposal, but directly on the entire image, which reduces a lot of repeated calculations. It turns out that R-CNN algorithm filters each region

proposal separately. Since there are about 2,000 region proposals in an image, the overlap rate between them must be very high, so duplicate calculations are generated. (2) ROI pooling is utilized to perform the size transformation of feature maps. Considering the input of the fully connected layer requires the same size, we cannot directly take advantage of the region proposal as an input. (3) The regressor is input into the network and has been trained together. Each class corresponds to a regressor. At the same time, the original SVM classifier is replaced by using softmax. However, region proposals become the bottlenecks in the Fast R-CNN algorithm which affects its performance.

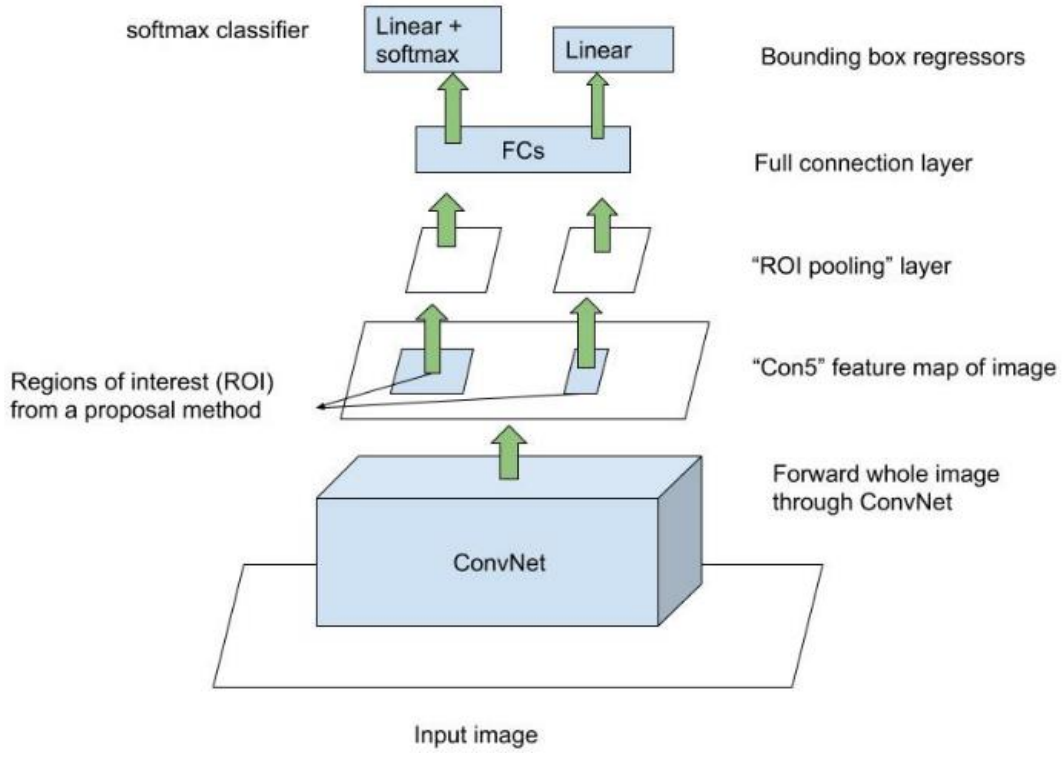

Figure 2.2 The structure of Fast R-CNN

2.2.3 Faster R-CNN

Both R-CNN and Fast R-CNN employ selective search to find out the region proposals.

Selective search is a slow and time-consuming process. Therefore, a new Faster RCNN was proposed (Girshick, et al., 2016). Faster R-CNN integrated feature extraction, proposal extraction, bounding box regression, and classification into one network. Region Proposal Network (RPN) replaces selective search for selecting candidate regions. At the same time, an anchor box is introduced to deal with the change of the target shape. The anchor is a box with a fixed position and size.

Similar to Fast R-CNN, the image is provided as an input to a convolutional network which provides a convolutional feature map. Instead of using a selective search algorithm on the feature map to identify the region proposals, a separate network is employed to predict the region proposals. The predicted region proposals are then reshaped by using a ROI pooling layer which is offered to classify the image within the proposed region and predict the bounding boxes.

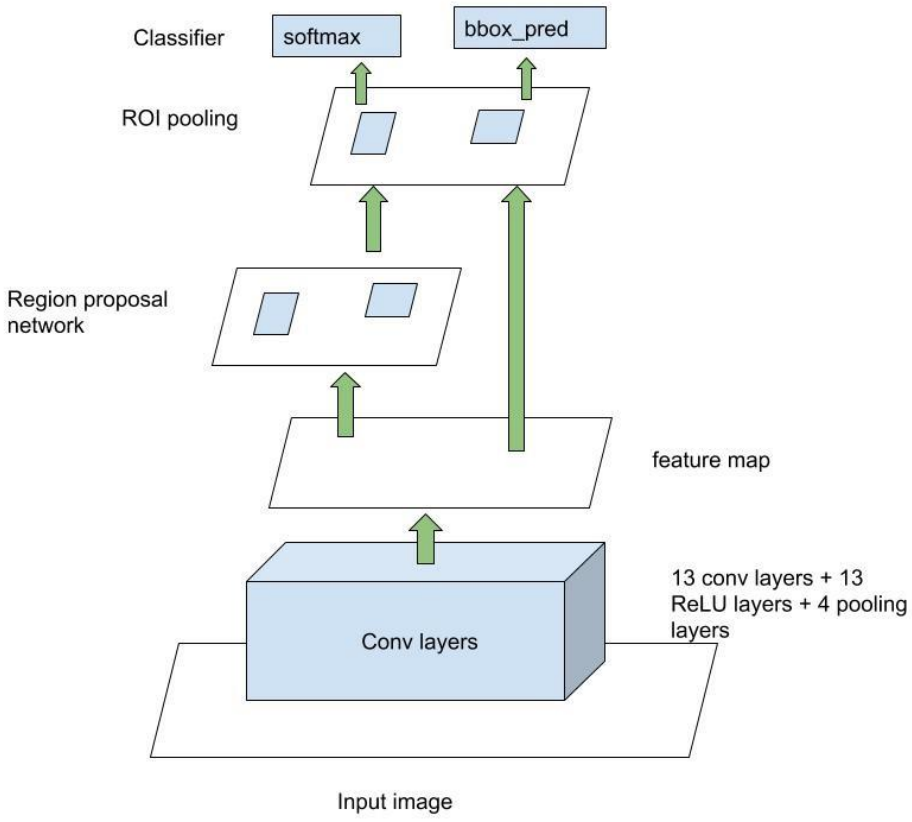

Figure 2.3 The structure of Faster R-CNN

Faster R-CNN discards the traditional sliding windows and selective search

methods exploit RPN to generate the detection frame. This is also a huge advantage of Faster R-CNN, which greatly increases the speed of detection frame generation.

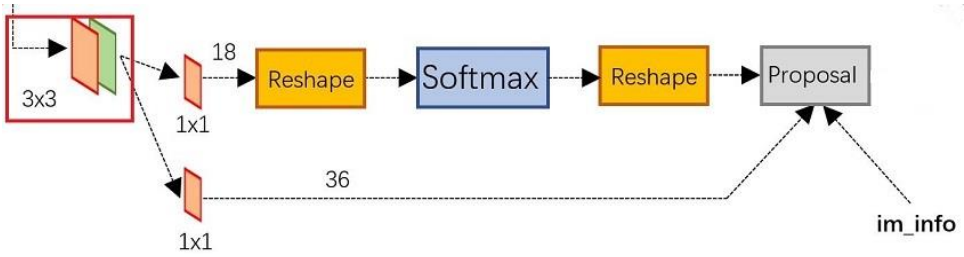

Figure 2.4 The Structure of RPN

Figure 2.4 displays the structure of the RPN network (Chen and Gupta, 2017). We think that the RPN network has two branches. The upper one obtains the foreground and background through the softmax classification anchors. The lower one is applied to calculate the regression offset of the bounding box for anchors to obtain an accurate proposal. The final proposal layer is responsible for synthesizing foreground anchors and regression offsets of the bounding box to obtain proposals, remove proposals that are too small and out of bounds. The entire network reaches the proposal layer and completes the function of target positioning.

In RPN, by using anchors to solve the problem of the indefinite length of the bounding box that is to place a fixed-size reference bounding box on the original image, rather than directly detect the position of objects. RPN selects which anchor to detect objects by determining whether each anchor contains related objects, and then adjusts the anchor to better fit the related objects.

R-CNN family of object detection methods based on region proposal, from SPP-NET and Fast R-CNN to Faster R-CNN, the process of object detection based on deep learning has become more accurate and faster.

2.2.4 YOLO

You Only Look Once (YOLO), (Redmon, et al., 2016) is a visual object detection algorithm that is different from the region-based algorithms, a single convolutional network is employed to predict the bounding boxes and the class probabilities for these boxes.

YOLO segments the input image into a $S \times S$ grid, each cell is responsible for detecting those objects whose center points fall within the grid. Each cell predicts bounding boxes and the confidence score of the bounding box. The confidence includes two aspects, one is the probability that the bounding box contains the object, the other is the accuracy of the bounding box. The first one is defined as $P_r(Object)$. If the bounding box is the background, then $P_r(Object) = 0$, otherwise $P_r(Object) = 1$.

The accuracy of the bounding box is characterized by using IOU (Jana and Biswas, 2018) of the predicted box and the ground truth. Therefore, confidence is defined as $P_r(Object) \bullet IOU_{pred}^{truth}$. The confidence of YOLO is the product of two factors, IOU_{pred}^{truth} reflects the accuracy of the prediction reflected. The size and position of the bounding box are described by four parameters x, y, w, h, where *x* and *y* are the centroid coordinates of the bounding box, *w* and *h* are the width and height of the bounding box, respectively. The predicted value of the center coordinate (x, y) is the offset value from the upper left coordinate point of each cell, the unit is related to the cell size. The parameters *w* and *h* are applied to predict values of the bounding box which are relative to the width and height of the entire picture, hence the size of the four elements should be within the interval $[0,1]$. In this way, the predicted value of each bounding box contains five parameters (x, y, w, h, c), where the first four represent the size and position of the bounding box, the last one stands for the confidence.

In additionally, regarding classification, it gives the predicted *C* class probability for each cell, which represents the probability of the bounding box predicted by this cell belonging to each class. These probability values are conditional probabilities under the confidence of each bounding box $P_r(class_i|\text{object})$. However, no matter how many bounding boxes are predicted by a cell, it only predicts a set of class probabilities. This is a disadvantage of the YOLO algorithm. In the later improved version, YOLO9000 combines the class probability as a prediction value of the bounding boxes. At the same time, class-specific confidence scores of each bounding box can be calculated:

$$P_r(class_i|\text{object}) \bullet P_r(\text{object}) \bullet IOU_{pred}^{truth} = P_r(class_i) \bullet IOU_{pred}^{truth} \quad (2.1)$$

The structure of YOLO is similar to GoogleNet (Putra et al., 2018). The difference is that YOLO utilizes a 1×1 convolution layer and a 3×3 convolution layer instead of the inception module.

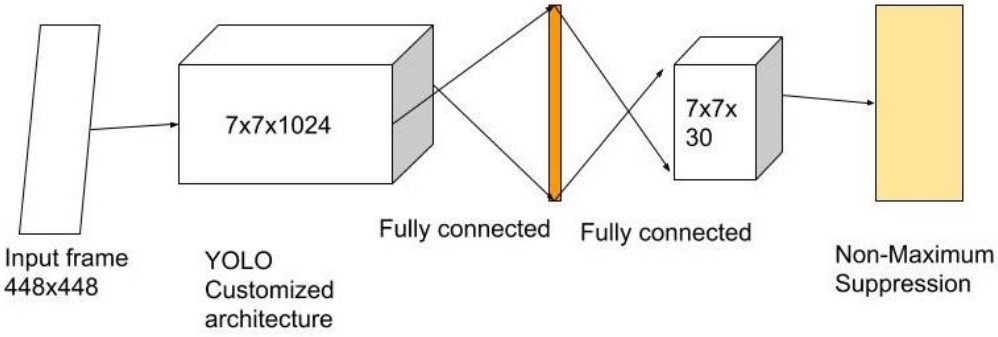

Figure 2.5 The Structure of YOLO

The entire detection network includes 24 convolutional layers and 2 fully connected layers. Features are extracted by convolution layer, the fully connected layer is utilized to predict image position and class probability values. The Leaky ReLU activation function is $\max(x, 0.1x)$, $x \in R$ for the convolutional layer and the fully connected layer. The mean square error is employed as the loss function. It consists of three parts: coordinate error, IOU error, and classification error.

$$\text{Loss} = \sum_{i=0}^{s^2} coordErr + iouErr + classErr \quad (2.2)$$

YOLO is faster than R-CNN-based algorithms, which achieves the speed of 45*fps*. In 2017, Redmon, et al. proposed YOLO9000, which improved the resolution of the training image and introduced anchor box to betterment the structure of the network. The output layer includes a convolutional operation instead of fully connected layer. Shafiee, et al. takes advantage of COCO dataset and ImageNet dataset to train the model. Compared with YOLO, YOLO9000 has greatly improved on the recognition type, accuracy, speed, and positioning accuracy.

2.2.5 SSD

In 2016, the algorithm SSD (Single Shot MultiBox Detector) was proposed to improve the precision of object detection and achieve over 74% mAP (mean Average Precision) at 59 *fps* based on standard datasets PascalVOC and COCO. Compared with YOLO, SSD takes CNN into account to perform detection directly, instead of performing detection after fully connected layers. On the other hand, there are two important changes. The first is that SSD extracts feature maps of different scales for object detection. The large-scale feature maps (the higher-level ones) are exploited to detect small objects, while small-scale feature maps are applied to detect large objects. The second is that the SSD adopts the prior boxes, the default boxes of different scales and aspect ratios. The disadvantages of the YOLO algorithm are that it is difficult to detect small targets and the positioning is not accurate (Womg, et al., 2018). However, these important improvements have assisted SSD to overcome these shortcomings.

YOLO predicts multiple bounding boxes for each grid. YOLO needs to adapt to the shape of the target during the training process due to the shape of the real target is variable. While SSD is inspired by the anchor of Faster R-CNN, SSD sets a priority boxes with different scales or aspect ratios for each gird. The predicted bounding boxes are based on these priori boxes. SSD outputs a set of independent detection values for each grid which corresponds to the bounding box.

Independent detection has two parts. The first is the confidence of each category. SSD treats the background as a special category. If the detection target has c classes, the SSD needs to predict $c + 1$ confidence values. Confidence is a score that has no goals or belongs to a background. In the prediction process, the category with the highest confidence is the category to which the bounding box belongs. The second part is the location of the bounding box, which contains four parameters $(C_x, C_y, \mathrm{W}, \mathrm{H})$, which represent the coordinates and width and height of the bounding box respectively. SSD takes advantage of VGG16 as the basic model, and then adds a convolution layer. The network structure of SSD is demonstrated in Figure 2.6.

SSD utilizes multi-scale feature maps for object detection. The input image size of the model is 300×300. Firstly, VGG16 is pretrained based on the ILSVRC CLS-

LOC dataset. Then, we convert the fully connected layers FC6 and FC7 of VGG16 to 3×3 convolution layer Conv6 and 1×1 convolution layer Conv7, the pooling layer Pool5 was changed from the original *stride* = 2 to *stride* = 1.

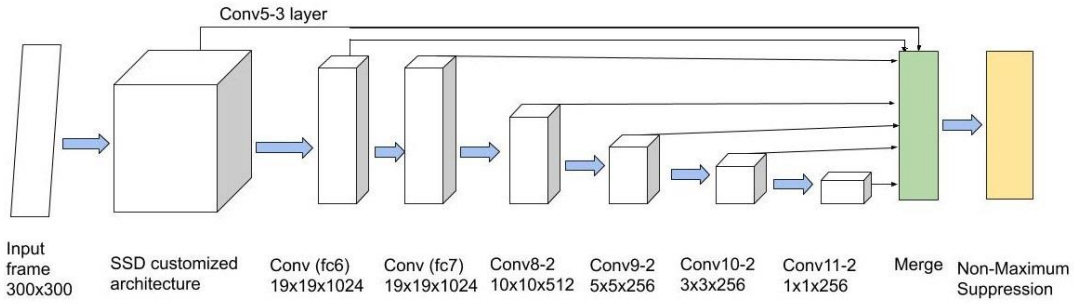

Figure 2.6 The structure of SSD

To cope with this change, Atrous algorithm is employed to exponentially expand the field of convolution without increasing the parameters and model complexity. Lastly, we remove the dropout layer and the FC8 layer and add a series of convolutional layers. The loss function is defined as the weighted sum of the location loss and the confidence loss

$$\mathrm{L}(x,c,l,g) = \frac{1}{N}\left(L_{conf}(x,c) + \alpha L_{loc}(\mathrm{x},\mathrm{l},\mathrm{g})\right), \tag{2.3}$$

where N is the number of positive samples in the prior box, $x_{ij}^{p} \in \{1,0\}$. If $\mathrm{x}_{ij}^{p} = 1$, it indicates that the i-th box matches the j-th ground truth, the class of ground truth is P, c is the confidence of category, l is the position prediction value of the corresponding bounding box of the prior frame, g is the position parameter of ground truth. For position error, Smooth L_1 loss is adopted, which is defined as

$$smooth_{L1}(x) = \begin{cases} 0.5x^2 & if|x| < 1.0 \\ |X| - 0.5 & otherwise \end{cases} \tag{2.4}$$

where $smooth_{L1}$ is the absolute value of X minus 0.5 $if|x| < 1.0$. If | X | is less than 1.0, it is $0.5x^2$.

2.3 Multi-Object Tracking

Multi-Object Tracking (MOT), as the name implies, is to track multiple targets simultaneously in a video. In multi-target tracking, there is a problem that new target entering and old target disappearing, which is the biggest difference from the single-target tracking algorithm (Ess, et al., 2010). As a result, the tracking strategy is different. In single-target tracking, a given initial box was utilized to predict the position of objects in the initial box in subsequent video frames. However, most of the multi-target tracking algorithms do not consider the initial box. Instead, a popular tracking strategy in the multi-target tracking is TBD (tracking-by-detection), or DBT (detection-based tracking). That is, object detection is carried out on each frame, and the result of target detection is employed to track the target. This step is generally called data association.

According to the sequence of trajectory generation, the multi-target tracking algorithm can be categorized into offline multi-target tracking algorithm and online multi-target tracking algorithm. The offline multi-target tracking algorithm is usually constructed as the graph model of object detection, in which the similarity or distance measurement between design and calculation is the key to determine the correctness of graph model construction. According to the current detection and observation, the online multi-target tracking algorithm calculates the matching relation with the existing trajectory, an appropriate matching measure determines the correctness of the matching. Therefore, learning the characteristics of detection results and calculating matching similarity or distance measurement are the key steps of the multi-target tracking algorithm, both offline and online.

The main task of deep learning-based multi-target tracking algorithm is to optimize the design of similarity or distance measurement between detection. According to the learning features, deep learning multi-target tracking includes deep learning based on apparent features, deep learning based on similarity measures, deep learning based on higher-order matching features, as shown in Figure 2.7.

The deep neural network could improve the multi-target tracking algorithm. For

example, the depth features obtained in image recognition or pedestrian recognition tasks can be directly replaced with the apparent features in the existing multi-target tracking algorithm framework, optical flow motion features can be learned by the deep neural network to calculate the motion correlation (Ryan, et al., 2011). A more direct way to improve the multi-target tracking algorithm is to learn the feature similarity between the detection. For example, depth network is designed to calculate the distance function of different detection. The detection distance of the same target is small while that of different target is large. The cost function for binary classification can be employed so that the detection feature matching type of the same target is 1, while the feature matching of different targets is 0. If the matching between existing trajectories is considered, the deep learning method can be applied to design and calculate the matching similarity between trajectories, which is thought as a high-order feature matching method. By using deep learning to calculate high-order feature matching, we can learn the matching similarity of multi-frame epigenetic features, we can also study the matching relevancy of motion features.

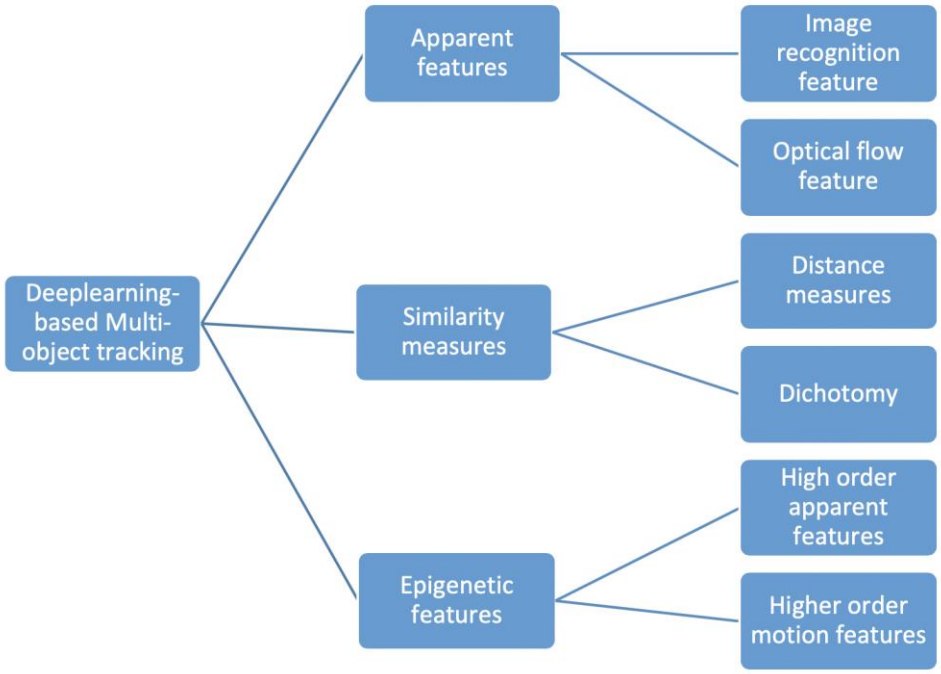

Figure 2.7 Deep-learning-based multi-object tracking

2.3.1 Hungarian Algorithm and Kalman Filter Based Tracking Algorithm

Multi-target tracking using the Hungarian algorithm and Kalman filtering is proposed by Bewley and others (2016) in their Simple Online and Realtime Tracking (SORT) algorithm. Kalman filter copes with the correlation of frame-by-frame data through the motion characteristics such as the long aspect ratio of the object, the position of the center point and speed utilizes the Hungarian algorithm for correlation measurement to obtain the correlation result. This simple algorithm achieves good performance at high frame rates.

In 2017, Wojke, et al. improved SORT algorithm by adding speed and trajectory features of the object. This algorithm DeepSort can alleviate the occlusion problem and reduce the number of id switches.

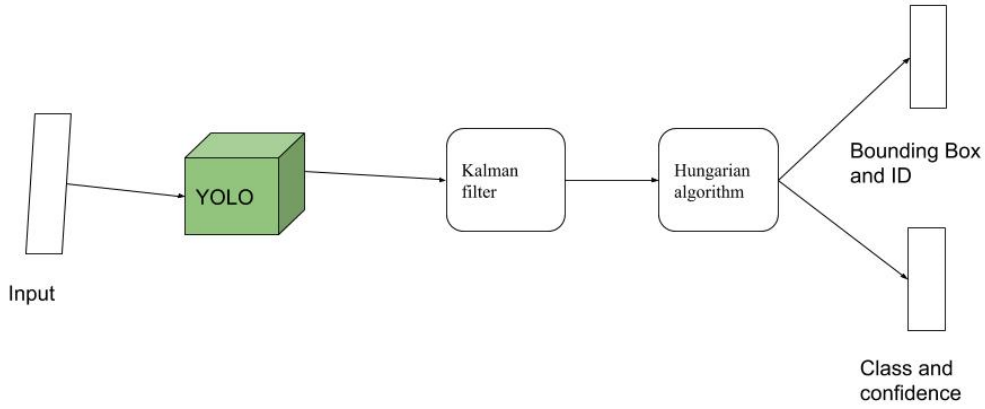

Figure 2.8 The structure of DeepSort

The structure of the DeepSort is presented in Figure 2.8. The tracking process includes the following steps:

(1) Input the first frame, initialize, and create a new tracker with the detected target and mark the ID.

(2) Input the next frame, the state prediction and covariance prediction are generated according to the previous frame box by using Kalman filter, find all target state predictions of the tracker and the IOU of the box detected in this frame. The best match is obtained by using the Hungarian algorithm, the

matching pairs whose matching value is less than the IOU threshold are removed.

(3) Utilize the detection box of the matched target in this frame to update the Kalman tracker, calculate the Kalman gain, state update and covariance update, output the state update value as the tracking box for this frame. Re-initialize the tracker for targets that are not matched in this frame.

Kalman filtering takes an 8D feature vector $(\mathrm{u}, \mathrm{v}, \mathrm{r}, \mathrm{h}, x^*, y^*, r^*, h^*)$ to present the state of the trajectory. These 8D vector variables represent the position, aspect ratio, height of the center of the bounding box, the corresponding speed information in the image coordinates. A Kalman filter is applied to predict the updated trajectory. The Kalman filter takes advantage of a uniform velocity model and a linear observation model.

In addition, Kalman filtering also handles with the problem of new target generation and the disappearance of old targets, there is a threshold a for each trajectory to record the time from the last successful match of the trajectory to the current time. If the value is greater than the threshold, the track is considered to be terminated. On the other hand, the track is considered to a new trajectory. However, a new trajectory may be generated for object detection that does not match successfully. The newly generated trajectory in this case is marked with the status 'tentative', and it is observed whether the consecutive matches are successful in the next consecutive frames. If matched, it is considered to be a new trajectory, marked as "confirmed", otherwise, it is considered as a false trajectory, the status is marked as 'deleted'.

Hungarian algorithm combines target motion and surface feature information for object matching, which applies Mahalanobis distances to evaluate the predicted Kalman state and the new state is

$$d^{(1)}(\mathrm{i}, \mathrm{j}) = \left(d_j - y_i\right)^T S_i^{-1} (d_j - y_i) \tag{2.5}$$

where $d(i, j)$ represents the degree of motion matching between the j-th detection and the i-th trajectory, S_i^{-1} is the covariance matrix of the observation space at the current time predicted by the Kalman filter, y_i is the prediction of the trajectory at the current

time Observation, d_j denotes the state of the *j*-th detection (u, v, r, h).

If the target motion uncertainty is low, Mahalanobis distances is a good correlation metric. In practice, if the camera moves, it will cause a large number of Mahalanobis distances that cannot be matched, which will invalidate this metric. Therefore, for each d_j of the detected bounding box, we calculate a feature descriptor $r_j, |r_j| = 1$, create a gallery to store the latest $L_k = 100$ trajectory descriptor, $R_k = \left\{r_k^{(i)}\right\}_{k=1}^{L_k}$, the minimum cosine of the *i*-th trajectory and the *j*-th trajectory is the second measure.

$$d^{(2)}(\mathrm{i}, \mathrm{j}) = \min\left\{1 - r_j^T r_k^{(i)} \mid r_k^{(i)} \in R_i\right\} \tag{2.6}$$

Then, we merge these two distances

$$c_{i,j} = \gamma d^{(1)}(\mathrm{i}, \mathrm{j}) + (1 - \gamma) d^{(2)}(\mathrm{i}, \mathrm{j}) \tag{2.7}$$

where γ is a hyperparameter to adjust the weight of different distances. The selection of γ depends on the specific data set. For example, for a data set with a large motion range, the weight of the Mahalanobis distance need to be small.

2.3.2 Siamese Network-Based Multi-Target Tracking Algorithm

This algorithm is based on Siamese symmetric convolution network, which takes two image blocks of the same size as the input, the output is the discrimination of whether the two image blocks belong to the same target (Guo, et al., 2017).

The Siamese symmetric network was proposed to learn the similarity of the epigenetic features (Lealtaixe, 2016), the similarity between the fusion motion and the epigenetic features was obtained by fusing the motion features of the classifier based on the gradient descent algorithm. After using the Siamese network, a fully connected network was connected as the output of apparent features and 6D motion context features in terms of relative change in size, changes in position and speed. Through the classical gradient descent algorithm, the multi-target tracking results are obtained by linear programming optimization.

Siamese-RPN was proposed in 2018. The algorithm is split into two parts: the

Siamese feature extraction network and region proposal network. During the training, the algorithm can be trained through densely labelled and sparsely labelled datasets. Compared with the existing methods, the sparsely labelled dataset greatly increases the source of training data, so that the deep neural network can be trained more fully, the coordinate regression in the region proposal network can make the tracking box more accurate and eliminate multi-scale time. The training structure of Siamese-RPN is displayed in Figure 2.9.

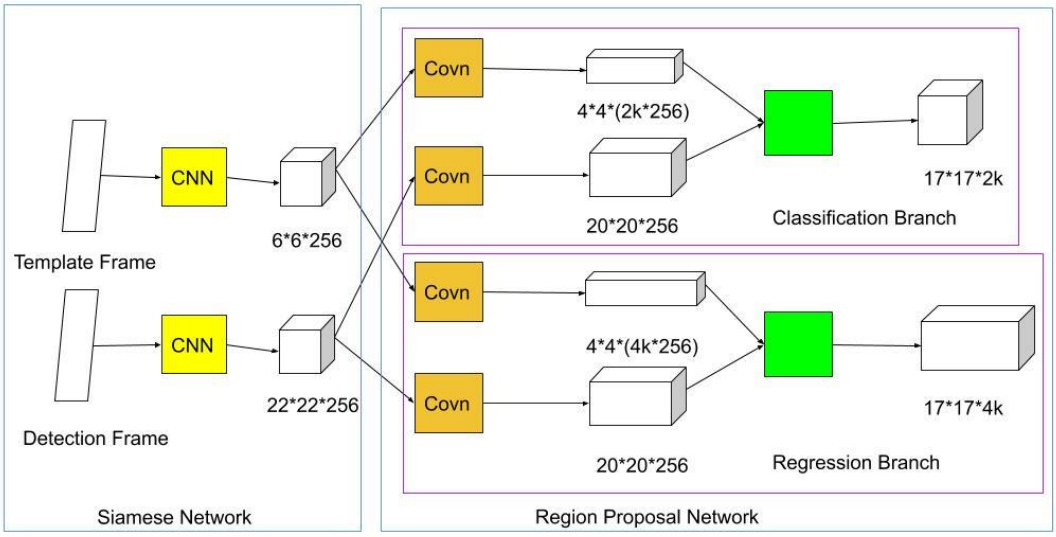

Figure 2.9 Training structure of SiameseRPN

The left part in Figure 2.9 is a Siamese network. The network structure and parameters of the upper and lower branches are the same. The search area of the frame to be detected is larger than the area of the template frame. The right part is RPN, which is divided into two branches. The upper branch is the classification branch. The features of the template frame and the detection frame after passing through the Siamese network are passed through a convolution layer. K is the number of anchor boxes since it is grouped into two categories, so it is $2k$. The bottom part in Figure 2.9 shows the bounding box regression branch, as there are four quantities $[x, y, w, h]$, it is $4k$.

The tracking part is different from the training structure. The two branches of the network were dismantled during the test. The template branch only propagated forward in the template frame, only the detection branch was performed in each frame. The

template frame calculates and retains the two feature maps, the detection branch only needs to retain these two features, no further forward propagation of the template frame is required. The structure is demonstrated in Figure 2.10.

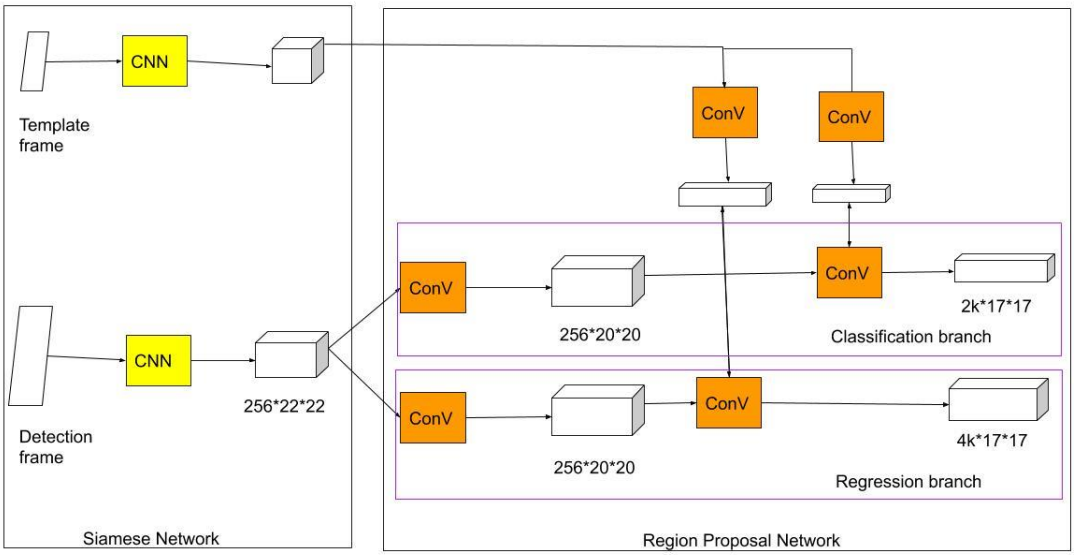

Figure 2.10 Tracking structure of SiameseRPN

To ensure the accuracy of the template, the first frame is handled as the template during the test to prevent the error from being continuously accumulated. In the case that only the first frame is needed as a template, the network is split into two small subnetworks according to the input image during tracking. The template branch extracts two features based on the template image in the first frame, only does the detection branch.

As presented in Figure 2.10, the template branch only inputs the template frame in the first frame to obtain two feature vectors. According to the description in the training framework, the tracker converts these two feature vectors into two convolutional layers without Bias, that is, the two features connected by thick lines and double arrows in Figure 2.9. After the transformation, the detection branch becomes a simple detection network. After passing through the feature extraction network, it goes through two convolutional layers to obtain the final classification results and regression results.

According to the output of the network, all the boxes predicted by the network and the corresponding scores can be obtained. After the suppression of the Gaussian window and the suppression of the shape, a weighted score can be acquired, from which

the corresponding box with the highest score is selected, which is employed as the target position for the final network prediction.

2.3.3 Minimum Multi-Cut Graph Model-Based Multi-Target Tracking Algorithm

Tang et al. (2016) claimed that the detection match between two frames is not the best model representation. Due to the existence of any inaccuracies in detection, the detection and matching relationships between images and within images are considered at the same time, the corresponding graph model has a wider representation capability than the graph model that only checks the detection and matching between frames. Tang, et al. took advantage of deep matching as matching feature. They proposed the least cost poly cut graph model based on the lifted edge.

The basic idea is to extend the constraint conditions of the original multi-cut formula to group the connections of nodes in the graph into the regular edge and promoted edge. The regular edge records the short-term matching state, the promoted edge records, and the matching relationship between long-term similarity detection.

Similar to the minimum cost flow model for multi-target tracking algorithms, this model takes into account intra-frame matching that is modelled as a minimal multi-cut problem for graphs,

$$\min_{x\in\{0.1\}^E} \sum_{e\in E} c_e\, x_e \tag{2.8}$$

where c_e represents the cost of each edge, it is calculated using the similarity between the detections, $x = 0$ means nodes belong to the same object, vice versa. The constraints of this binary linear programming problem indicate that for any existing loop, if there is a connection $x = 0$, then all other paths on this loop are $x = 0$. That is, for the zero loops in the optimization result, they are all in the same object. Hence, $x = 1$ represents the segmentation of different object, so this problem is transformed into the minimum multi-cutting problem of graphs. KLJ algorithm is utilized to solve the problem of the minimum cost multi-cut.

Deep matching features, calculated by the deep learning algorithm framework, are employed as the matching feature. There are give-dimensional features based on the deep matching feature.

$$f_1^{(e)} = \mathrm{MI}/\mathrm{MU} \tag{2.9a}$$

$$f_2^{(e)} = \min\{\delta_v, \delta_w\} \tag{2.9b}$$

$$f_3^{(e)} = f_1^{(e)} f_2^{(e)} \tag{2.9c}$$

$$f_4^{(e)} = (f_1^{(e)})^2 \tag{2.9d}$$

$$f_5^{(e)} = (f_2^{(e)})^2 \tag{2.9e}$$

where *MI* and *MU* represent the intersection size and union size of matching points in the rectangular frame, δ_v and δ_w represent the degree of detection trust. Using these five-dimensional features, we learn a logistic regression classifier and get the probability pe that is the same goal.

In order to connect long-distance matches and enhance the occlusion processing ability, while avoiding apparently similar but connected between different object detections, a lifted-edge multi-cut graph model with minimum cost is proposed. The basic idea is to extend the constraints of the original multi-cut and split the connection of nodes in the graph into regular edges and lifted edges. The short-term matching status, the lifted edges record the matching relationship between long-term similarity detection. Additionally, two additional constraints for lifting edges have been added, the lifting edges are correctly matched, there should have support for regular matching on regular edges. In edge cutting, there should also be supported for continuous cut edges on regular edges.

2.3.4 Time-domain Attention Model for Multi-target Tracking Algorithm

If the movements of the two objects interact, the occluded object cannot be distinguished correctly, this results in tracking drift. A statistical analysis of the drift of

the tracking algorithm is conducted in the pedestrian multi-target tracking problem and found that when different pedestrians interact, mutual occlusion is an important reason for the drift of the tracking algorithm (Chu, 2017). To solve this problem, a spatial-temporal concern model (STAM) was proposed for learning occlusion and identifying possible interference targets as shown in Figure 2.11.

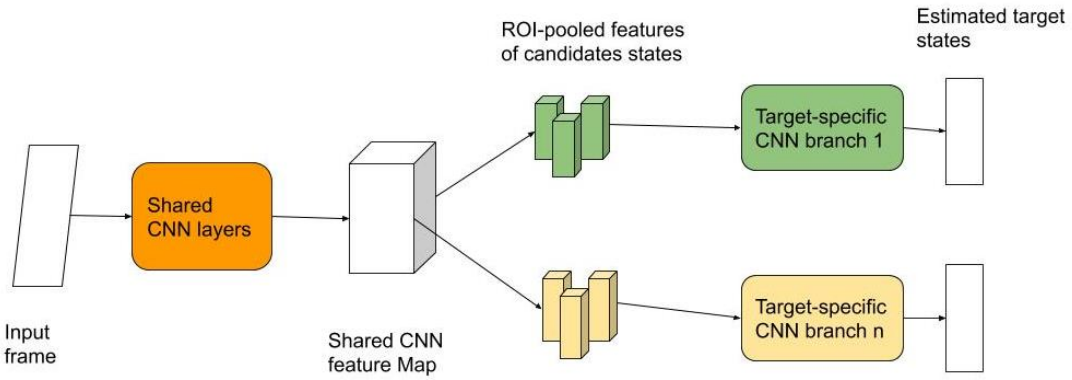

Figure 2.11 The STAM model framework for occlusion discrimination

The spatial attention model is employed to generate feature weights when occlusions occur. After the candidate detection features are weighted, they are selected by the classifier to obtain the estimated target tracking results. The time attention model (Zhu, et al., 2018) weights historical samples and current samples to obtain a weighted loss function. Update the target model online.

In this model, each target manages and updates its spatial-temporal concern model and feature model independently. We select candidate detection for tracking. Therefore, this method is an extension of single-target tracking algorithm in multi-target tracking. To distinguish between different targets, the key steps are how to model the occlusion state and distinguish between different targets that are close. The spatial attention model is utilized to analyze the occlusion state at each moment. The spatial attention model mainly split into three parts. The first step is to learn the visibility map of features

$$\mathrm{V}\left(x_t^j\right) = f_{vis}\left(\emptyset_{roi}\left(x_t^j\right); w_{vis}^i\right), \mathrm{V}(x_t^j) \in R^{w*h} \tag{2.10}$$

where f_{vis} is a network operation of a convolutional layer and a fully connected layer, w_{vis}^i is the parameter to be learned.

The second step is to calculate the spatial attention map based on the feature visible map

$$\varphi(x_{t,j}^{i}) = f_{att}(V(x_{t,j}^{i});\ w_{att}^{i}), \varphi(x_{t,j}^{i}) \in R^{w \times h} \tag{2.11}$$

where f_{att} is a locally connected convolution and scoring operation, w_{att}^{i} is the learned parameter.

The third step weights the original feature map according to the spatial attention map:

$$\emptyset_{att}(x_{t,j}^{i}) = \emptyset_{roi}(x_{t,j}^{i})\varphi(x_{t,j}^{i}), \emptyset_{att}(x_{t,j}^{i}), \emptyset_{roi}(x_{t,j}^{i}) \in R^{w \times h \times c}, \varphi(x_{t,j}^{i}) \in R^{w \times h} \tag{2.12}$$

Next, the generated weighted feature map is fed into convolution and fully connected network to generate a binary classifier to determine whether it is the target itself. Finally, the obtained classification score matrix is utilized to select the best tracking result.

2.3.5 Long Short-term Memory Network-Based Multi-target Tracking Algorithm

The historical trajectory information is important to judge the target state in multi-target tracking. It is feasible to design a network structure that can remember the historical information and learn to match the similarity measure based on historical information (Zhang, et al., 2018). A feature fusion algorithm is proposed based on the long short-term memory network (LSTM) to learn the matching similarity between trajectory history information and current detection (Sadeghian, 2017). LSTM is handled as the historical information model of the apparent model, the motion model and the interaction model in the multi-target tracking algorithm of the loop network distinguish the fusion apparent motion interaction. Figure 2.12 shows the structure of this model.

Three aspects of feature calculation match the trajectory of historical information with detection: Apparent features, motion features, and interaction mode features. The fusion of these three features is calculated hierarchically (Xingjian, et al., 2015).

In the underlying feature matching calculations, all three features utilize the LSTM model. For the apparent features, we borrow the idea from Zhao and other's work (2017)

using the VGG-16 convolutional network to generate a 500-dimensional feature and input this feature to the LSTM. The output features of the network, for the current detection, we calculate the features of the same dimension, connect these two features and calculate the 500D features through the fully linked network layer, learn the classifier according to the matches, and pre-train the network.

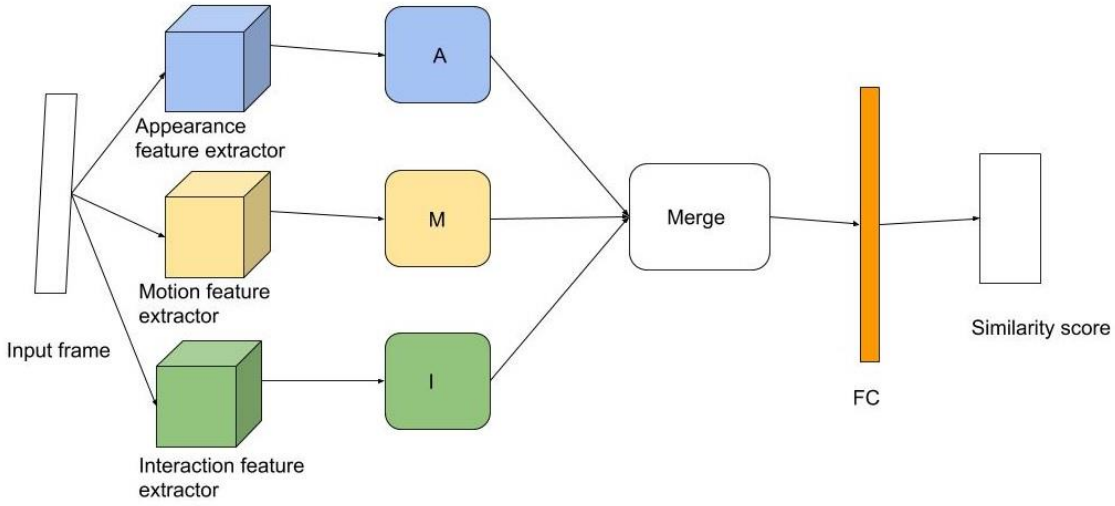

Figure 2.12 The LSTM-based tracking model

Pertaining to motion features, we take the relative displacement as the basic input feature, input the LSTM model to calculate the output and the relative displacement at the next moment. The feature is calculated through the fully connected network, similar to the 500-dimensional feature calculated from the apparent feature and train the network using a binary matching classifier.

For interactive features, we take the relative position map occupied by other objects in the rectangular area around the center position of the target as the input feature of the LSTM model, calculate the output feature. We detect an object at time $t + 1$, calculate a similar relative position map as the feature through a fully connected network. Similar to the motion model, we calculate 500-dimensional feature through a fully connected network and conduct the same classification training.

When all three features are calculated, we splice them into complete features, feed them into the upper layer of the LSTM network, the output vectors are fully connected. The correct match is 1, otherwise, it is 0. For the final network structure, fine-tuning is

required to optimize overall network performance. The final classification score is regarded as the similarity to detect the matching calculation with the trajectory target. The final tracking framework is calculated using online detection and trajectory matching methods.

Just using the basic LSTM model is not the best solution for apparent characteristics. After analyzing the design of various gate functions in LSTM, Kim et al. proposed an epigenetic learning network model based on bilinear LSTM. The hidden layer feature and the input of LSTM are employed as a feature to learn the matching classifier. In the results, the apparent properties of bilinear LSTM were the best, the optimum historical correlation length was 40.

Chapter 3
Methodology

Chapter 3 elaborates the method of implementation including data collection, our algorithms for data augmentation, data labelling as well as deep learning algorithms, models and operating environments, and model evaluations.

3.1 Data Collecting

In this thesis, our focus is on detecting and tracking the abnormal events happened in the sidewalk. We define the objects of abnormal events as scooters, bicycles, or cars riding on the sidewalk. Therefore, our dataset needs to contain five or more object classes in terms of scooters, bicycles, cars, buses and pedestrians.

Our dataset for object detection was taken on Queen Street, Auckland, New Zealand by using our iPhones. We took a total of one-hour video footages, picked up 3,000 photos from the video, approximately 600 photos for each class. Figure 3.1 demonstrates samples of our dataset.

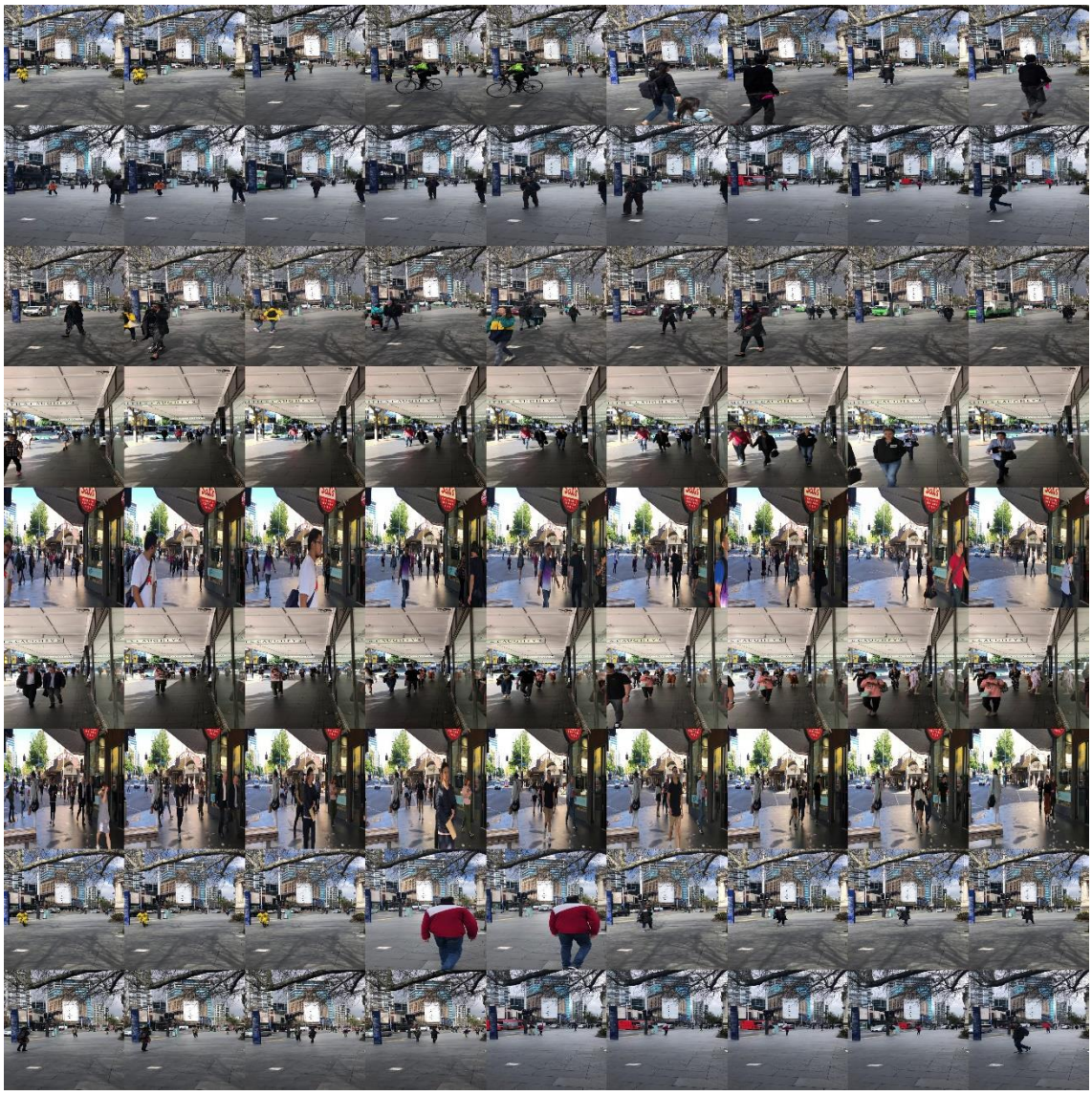

Figure 3.1 The selected frames from our dataset

For object tracking, we utilize our dataset and MOT16 dataset to train the model. Our dataset comprises of 15 short image sequences showing various objects in challenging backgrounds. The MOT16 data set is employed to measure the multi-target detection and tracking method standards of the multi-target tracking MOT Challenge series proposed in 2016. The dataset consists of 14 videos, these videos automatically were selected to feature objects in natural settings without editing or post-processing, with a recording quality often akin to that of a hand-held cell phone camera. Some of the frames of our training data are displayed in Figure 3.2.

Figure 3.2 The selected frames from the MOT16 dataset

3.2 Data Labeling

All images are labelled according to the format of VOC 2007. The PASCAL Visual Object Challenge (The PASCAL VOC) is a world-class computer vision challenge. VOC standard dataset is widely utilized in computer vision models such as classification, location, detection, segmentation, and motion recognition, especially the object detection models, such as the famous R-CNN series, followed by YOLO, SSD, etc.

In this project, we adopt three important folders of VOC2007 dataset: Annotations, ImageSets, and JPEGImages. Annotations folder is employed to store files in .xml format, which is the label corresponding to the image. Each .xml file corresponds to a picture in the JPEGImages folder. The structure of the data folder is presented in Figure 3.3.

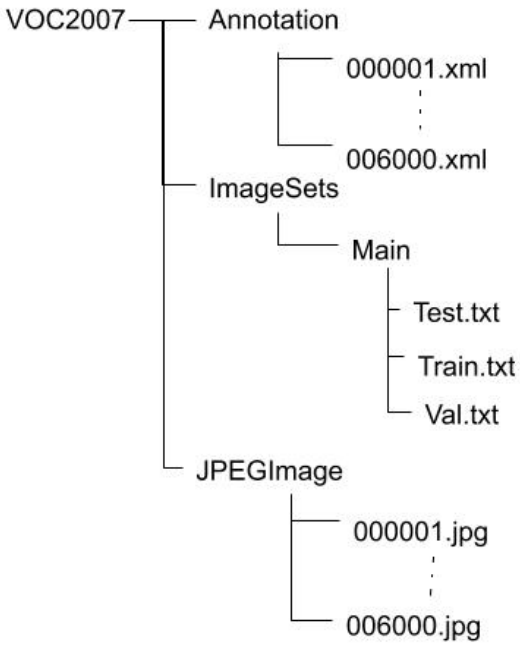

Figure 3.3 The structure of COV2007 dataset

All images are labelled by using LableIMG tool manually and saved to .xml file. The label file is showed in Figure 3.4.

```
<?xml version="1.0"?>
<annotation>
    <folder>JPEGImages</folder>
    <filename>107.jpg</filename>
    <path>D:\120\VOC2007\JPEGImages\107.jpg</path>
  - <source>
        <database>Unknown</database>
    </source>
  - <size>
        <width>1080</width>
        <height>1920</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
  - <object>
        <name>scooter</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
      - <bndbox>
            <xmin>465</xmin>
            <ymin>910</ymin>
            <xmax>530</xmax>
            <ymax>1114</ymax>
        </bndbox>
    </object>
</annotation>
```

Figure 3.4 The labeled file

3.3 Data Augmentation

Usually, we need to input enough data to avoid overfitting. But we can't collect sufficient data in practice. Data augmentation is to generate more data sets from our existing limited data to improve the accuracy and generalization ability of the network (Lemley, Bazrafkan and Corcoran, 2017). There are two categories of image data enhancement technical in deep learning, Data augmentation method based on artificial experience and data enhancement method based on machine learning (Perez and Wang, 2017).

The experience-based data augmentation includes geometric transformation, Affine transformation, noise injection and random erasing, so on. Geometric transformation carries out geometric transformation based on the original image data, changes the position of the image pixel and ensure that the features remain unchanged (Wu, et al., 2018).

Affine transformation is a linear transformation from two-dimensional coordinates

to two-dimensional coordinates. We implement the transformations using the 2 ×3 matrix, a linear transformation of two-dimensional coordinates using a matrix is expressed as

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}. \tag{3.1}$$

If we define $\mathbf{R} = \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix}$, $\mathrm{t} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$, $\mathrm{T} = [R \quad t]$, **R** is a linear transformation matrix. Therefore, the transformation is a linear transformation plus translation. The transformation contains image processing methods, including translation, rotation, scaling, and flipping. They correspond to different transformation matrices. In this thesis, we randomly rotate the image by a given angle and change the orientation of the image as shown in Figure 3.5.

Noise injection is other data augmentation technical, which randomly perturbs the RGB of each pixel of the image. The popularly noise modes are salt and pepper noises and Gaussian noises.

Figure 3.5 The affine transformation

In this thesis, we take advantage of the Box-Muller algorithm to generate Gaussian noises. The Box-Muller method is based on two sets of independent random numbers *U* and *V,* which are uniformly distributed on (0,1]. *U* and *V* are applied to generate two sets of independent standard normal distribution random variables *X* and *Y*,

$$\mathrm{X} = \sqrt{-2\ln U}\ \cos 2\pi V \tag{3.2a}$$

$$\mathrm{Y} = \sqrt{-2\ln U}\ \sin 2\pi V \tag{3.2b}$$

Equations (3.2a) and (3.2b) were proposed because the Chi-square distribution of two degrees of freedom can be easily generated by an exponential random variable. Therefore, the random variable *V* is employed to select an angle that surrounds the circle uniformly, the exponential distribution is applied to select the radius and then transformed into (normally distributed) *x* and *y* coordinates.

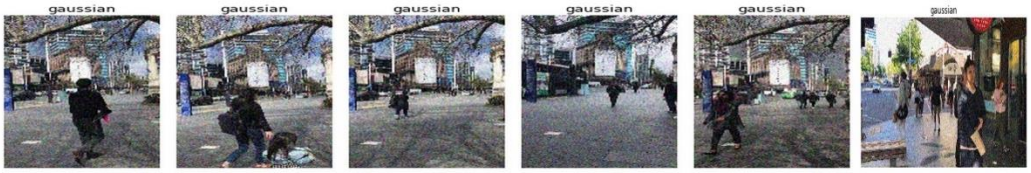

Figure 3.6 Gaussian noises

Random erasing is a data enhancement technique developed (Zhong, et al., 2017). Inspired by the dropout mechanism, they randomly selected a part of the image and deleted this part. This technology can improve the performance of the model when an object is partially occluded. Besides, it can also ensure that the network pays attention to the entire image without Just part of it. The procedure of random erasing algorithm is

Initialization: $p_1 \leftarrow$ Rand (0, 1).
if $p_1 \geq p$ **then**
  $I^* \leftarrow I$;
  return I^*.
else
  while *True* **do**
    $S_e \leftarrow$ Rand $(s_l, s_h) \times S$;
    $r_e \leftarrow$ Rand (r_1, r_2);
    $H_e \leftarrow \sqrt{S_e \times r_e}$, $W_e \leftarrow \sqrt{\frac{S_e}{r_e}}$;
    $x_e \leftarrow$ Rand $(0, W)$, $y_e \leftarrow$ Rand $(0, H)$;
    if $x_e + W_e \leq W$ *and* $y_e + H_e \leq H$ **then**
      $I_e \leftarrow (x_e, y_e, x_e + W_e, y_e + H_e)$;
      $I(I_e) \leftarrow$ Rand (0, 255);
      $I^* \leftarrow I$;
      return I^*.
    end
  end
end

Figure 3.7 The random erasing algorithm

In this algorithm, the input image is *I*, the probability of erasing is *p*, the erasing region with a ratio ranging from S_l to S_h, the aspect ratio ranges from r_1 to r_2. The first step is to determine whether an image needs to be erased according to the probability *p*, p_1= Rand (0,1). If $p_1 > p$, the image is not processed, otherwise it needs to be erased.

According to the input image *I*, the length *H* and width *W* of the input image can be obtained, and the area *S* can be acquired. Obtained the area S_e according to

Rand$(S_l,\ S_h)$ ×S, the length and width of the erased region are obtained by using eq.(3.3).

$$r_e = Rand(r_1, r_2), H_e = \sqrt{S_e * r_e}, W_e = \sqrt{\frac{S_e}{r_e}} \tag{3.3}$$

Following Rand (0, *w*) and Rand (0, *H*), the coordinates of the erased $x_e\ and\ y_e$ in the original image are obtained. If $x_e + W_e > W$ or $y_e + H_e > H$, the algorithm is repeated until $x_e + W_e \leq W$ or $y_e + H_e \leq H$ is satisfied. In this work, we set $P = 0.5$, $S_l = 0.02, S_h = 0.4, r_1 = \frac{1}{r_2} = 0.3$. The frame is deonstrated in Figure 3.8.

Figure 3.8 Region random erasing

These data augmentation techniques are related to all offline data augmentation, the generated images need to be re-marked due to the position of the object in some samples has been shifted. It is necessary to remove those low-quality samples, such as the frames whose objects are randomly erased to ensure the quality of the training data set.

3.4 Algorithm Design

This thesis aimed to propose an object detection, classification, and tracking framework to detect and tracking anomaly. We consider anomaly detection as two tasks of object recognition and tracking thus the network consists of a convolutional neural network (SSD) to detect the anomaly and a tracking network for anomaly tracking.

The model consists of SSD and modified SiamRPN neural networks. The raw frames were input in SSD to detect the objects, the detected objects were cropped from the original images and feed into a Siamese-RPN to feature extraction and calculate the similarity between objects. The obtained similarity between the template target and the detected target is fed to the Hungarian algorithm to obtain the best match. When the similarity score is higher than the threshold, the tracking is successful, otherwise, it is

not successful. The structure of the model is displayed in Figure 3.9.

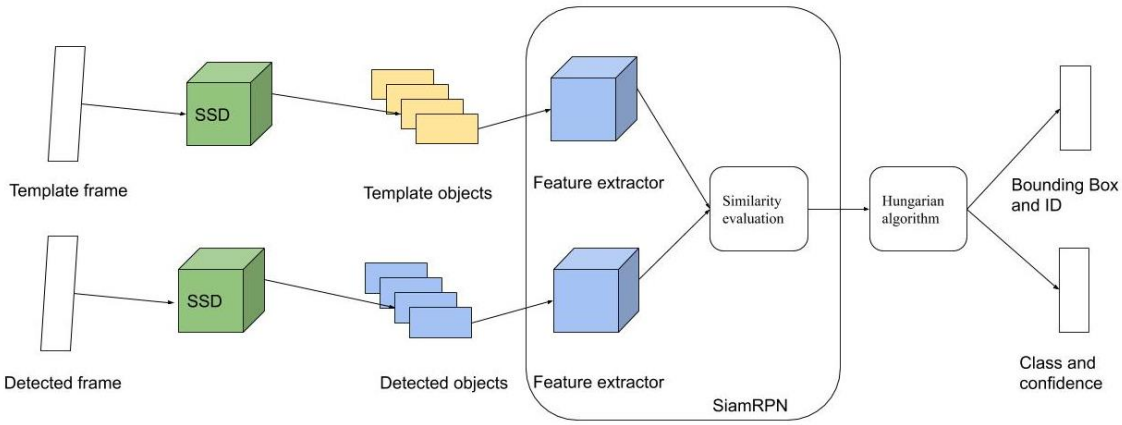

Figure 3.9 The structure of SSD + SiamRPN + Hungarian algorithm

The design of this network structure is inspired by the MBMD network (Zhang, et al. 2018) which took the first place in 2018. The MBMD framework is mainly composed of a regression network based on offline training and a verification network based on classification updated online, as shown in Figure 3.10.

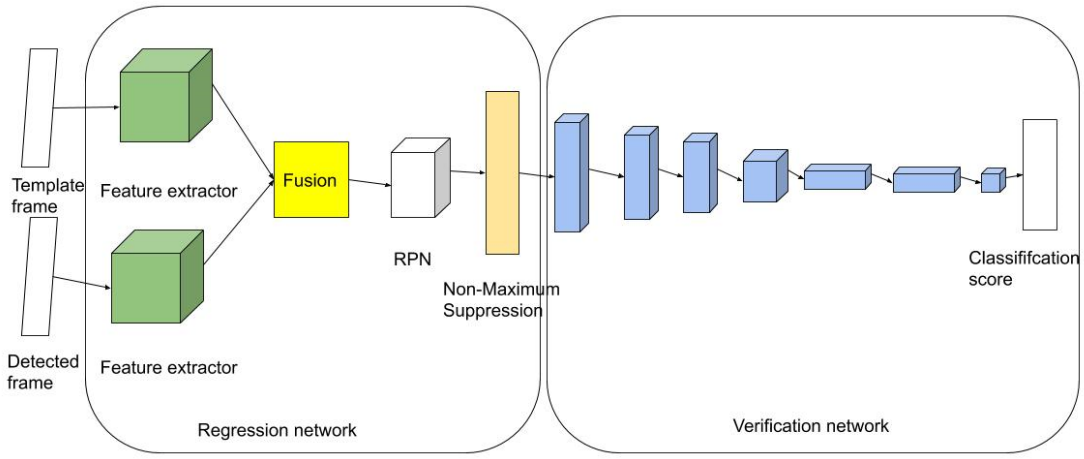

Figure 3.10 The structure of MBMD model

The regression network has two inputs, namely, the local search region and the target image block. The target position of the previous frame in the local search area is the center, and the four times size region is cropped in the current frame. In each frame, the regression network proposes the candidate frames similar to the target in the search area, and each candidate frame has a score describing the similarity.

The verification network learns a classifier online. It first checks whether the

candidate frame that most resembles the object is the target. If so, the target frame is the current frame tracking result. If the most similar candidate frame is divided into backgrounds by using the verification network, the verification network will select one of the candidate frames as the current frame tracking result. If neither network can find a candidate box that is similar to the target or is classified as the foreground, then the tracker will search the whole image, that is, starting from the upper left corner of the picture, a local search area is cut out, and the full image is searched. The step size is half of the target length, the vertical step size is half of the target width.

The matching network can locate similar objects in the area through offline training. SSD and MobileNets are utilized as the feature extraction algorithm. The input of the upper branch is a local search area, and output feature maps of two sizes (19 ×19, 10×10). The two scales feature maps are employed to deal with dramatic changes in target size. The lower branch inputs the target which will be tracked in the first frame and outputs a feature vector. The feature maps obtained by the fusion of the feature maps are input to the subsequent candidate RPN, the RPN module outputs the feature maps encoding the candidate frame information, which is sent to the non-maximum suppression to get the final candidate box.

Our model also take advantage of two algorithms to solve the problems of classification and data association. Both the detection network and the tracking network are offline pre-trained model. The detection network is SSD, the raw images are fed in SSD to detect and classify objects first, then, we get the coordinate of the targets in six classes, as well as the bounding box. After that, the results are fed into a tracking network for objects tracking.

The proposed tracking network is modified by using SiamRPN subnetwork. Multi-target tracking is different from single-object tracking. Single-object tracking gives the initial object and its bounding box first and tracks the objects in subsequent frames (Huang et al., 2019). Meanwhile, multi-target tracking performs object detection in each frame firstly. The second step is called data association, the proposed model utilizes the deep learning-based algorithm for feature extraction and calculates the

similarity score of the features of the tracked object. While in multi-objects tracking, there is a problem that the template object has multiple targets with similar features in the tracked image, therefore, the similarity scores are fed to the Hungarian algorithm to obtain the final matching result.

Tracking objects are regarded as learning problems. The most typical algorithm for similarity learning in deep convolutional networks is the Siamese algorithm. Therefore, we choose the Siamese region proposal network (SiamRPN) as the tracking network. However, SiamRPN is an algorithm for single-object tracking, which consists of a Siamese subnetwork for feature extraction and RPN network for region propose and classification (Cui, Tian and Yin, 2019).

A region proposal subnetwork includes the classification branch and regression branch for prediction. In the inference phase, SiamRPN can pre-compute the template branch of the Siamese subnetwork and formulate the correlation layers as trivial convolution layers to perform online tracking (Melekhov, Kannala and Rahtu, 2016).

The key structure of the proposed tracking network is RPN. RPN was proposed (Ren, 2015) as a part of Faster R-CNN, RPN generates the proposal for object detection. RPN has a specialized and unique architecture as presented in Figure 3.11.

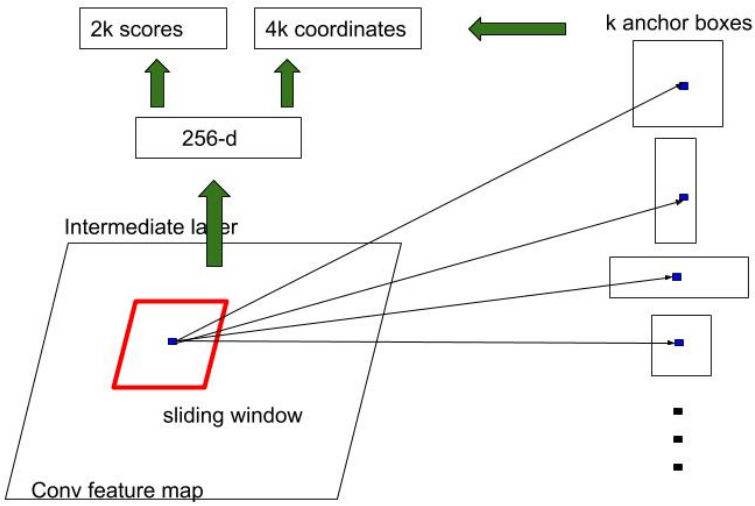

Figure 3.11 The region proposal network

RPN has a classifier and a regressor. An anchor is the central point of the sliding window. The classifier determines the probability of a proposal having the target object.

Suppose there are a total of k anchors, each anchor needs to be divided into foreground and background, thus $class = 2k$. Each anchor is given by four parameters x, y, w, and h, correspondingly, $reg = 4k$.

Regression is applied to find the coordinates of the region proposals. For any images, the scale is the size of the image, the aspect ratio is the width and height of this image. The developers chose three scales and three aspect-ratio. Therefore, nine proposals are possible for each pixel, this is how the value of k is decided, k is the number of anchors. For the whole image, the number of anchors is $W \times H \times K$. The anchors are the highest intersection-over-union overlap with a ground truth box.

We got the objects and their coordinate from SSD, we cropped the template objects from the original template frame. The detected objects were cut out from the detected frame. The template objects and detected objects are fed into a classification network. The classification network of RPN in this thesis is demonstrated in Figure 3.12.

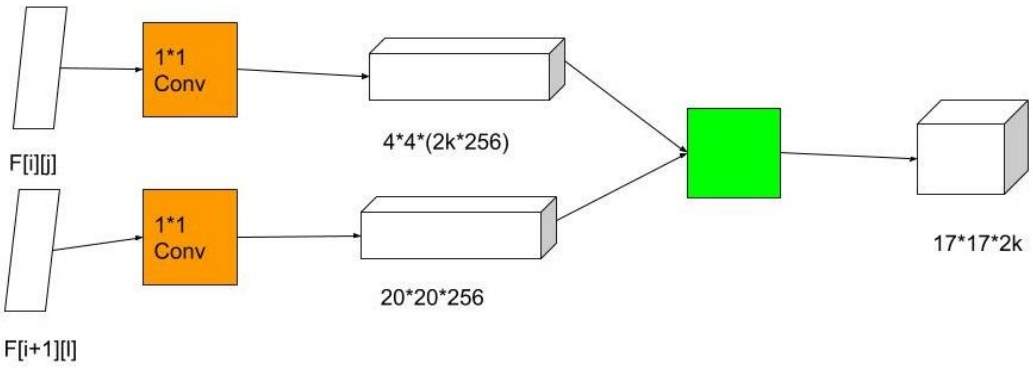

Figure 3.12 The classification network of RPN

We assume that the i-th frame has j objects, the (i+1)-th frame has l objects. F[i][j] is the j-th object of i-th frame which corresponds the template frame in original RPN. F[i + 1][l] is the l-th object of (i+1)-th frame that corresponds to the detection frame in original RPN. The detail of the classification branch of RPN works is shown in Figure 3.13.

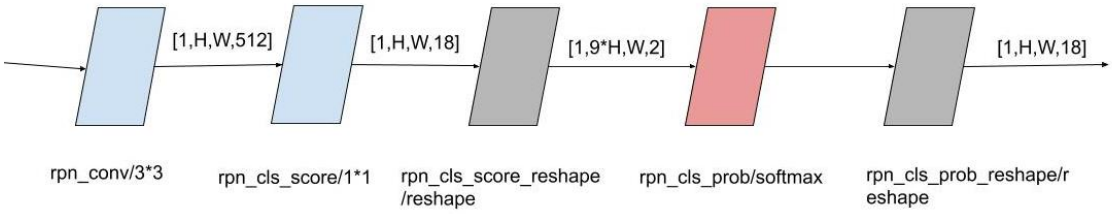

Figure 3.13 The process of RPN classification

The RPN classification is based on binary classification. Firstly, we divide the image into $K \times H \times W$ regions, namely, anchors, $K = 9$, H is the height of the feature map, W is the width on the feature map. We need to determine whether the anchor is foreground or background by comparing the overlap between these anchors and ground truth, then label each anchor with the foreground or background. After that, RPN can be trained to have the ability to recognize foreground and background for any input.

An image with the size $M \times N$ is fed into the Faster RCNN network, the resolution of this image becomes $(M / 16) \times (N / 16)$ before fed in the RPN network. We define $W = M / 16$ and $H = N / 16$. The image size $W \times H$ is fed into a 1×1 convolution first. Then, a reshape layer is followed by using 1×1 convolution, we see that the output is $9 \times H \times W \times 2$, which corresponds the possibilities of $9 \times H \times W$ anchors being classified as foreground and background. The outputs are the annotations of each anchor label, we compare it with the binary classification probability to get the classified loss. A feature map has $9 \times H \times W$ anchors, each point corresponds to nine anchors. These nice anchors have three aspect ratios of 1: 1, 1: 2, and 2: 1, and each aspect ratio has three sizes.

However, in actual application, $H \times W \times 9$ anchors are too much to label labels. There are rules to remove anchors with poor performance:

- Anchors covering the boundary of the feature map do not participate in training. The anchor at the border between foreground and background does not participate in training.
- These junctions serve as neither foreground nor background, in case of misclassification. IOU is the ratio of the overlapping area of anchor and ground truth to the total coverage area of the two (Bochinski, et al. 2018) as displayed in Figure 3.14.

- The number of samples in a batch during training is 256, corresponding to 256 anchors of the same image, the number of foregrounds cannot exceed half. If it exceeds, the number 128 is randomly selected as the foreground, and the background has similar filtering rules.

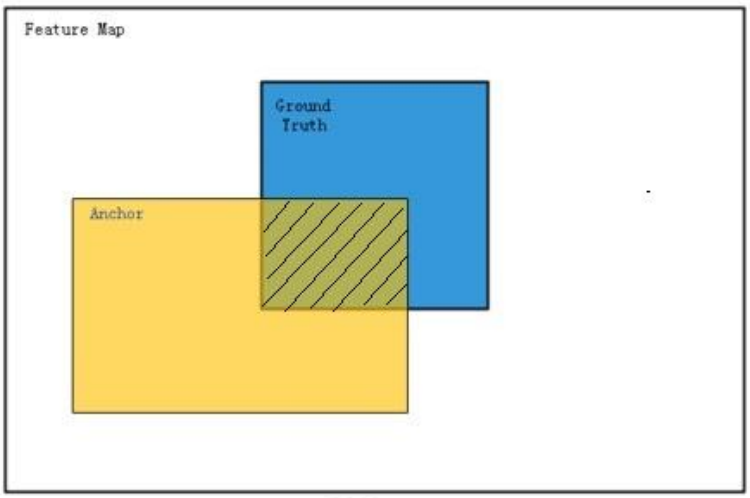

Figure 3.14 The definition of IOU

RPN is an algorithm that needs to be trained (Dong and Shen, 2018), the loss function is defined as

$$\mathrm{L}(\{p_i\},\{t_i\}) = \left(\frac{1}{N_{cls}}\right)\cdot\sum L_{cls}(p_i,p_i^*) + \left(\frac{\lambda}{N_{reg}}\right)\cdot\sum p_i\cdot L_{reg}(t_i\cdot t_i^*) \quad (3.4)$$

where *i* is the index of anchor, *p* is the probability of being an object or not, *t* is the vector of 4 parameterized coordinates of the predicted bounding box, * represents ground truth box, *L* for *cls* represents logarithmic loss over two classes, *p** with regression term in the loss function ensures that if and only if the object is identified as yes, only regression will count, otherwise *p** will be zero, so the regression term will become zero in the loss function. N_{cls} and N_{reg} are the normalizations. Λ = 10 by default is assigned to scale classifier and regressor on the same level.

The goal of RPN is to improve the computing speed and ensure the acquisition of the object position. However, during the experiment, we found that the bounding boxes predicted by RPN are not as accurate as the bounding boxes predicted by SSD. Figure 3.13 shows the bounding box for a bicycle and pedestrian predicted by using RPN and

SSD300. The blue bounding box is the result of the SSD prediction.

Figure 3.15 The comparisons of the bounding boxes of bicycle and pedestrian predicted by using SSD300 and RPN

As presented in Figure 3.15, when SiamRPN predicts the bounding box of the object, it depends on the initial coordinate of the object in the first frame. Although the size of the bounding box was adjusted while tracking objects are in subsequent frames, it was not accurate enough. The following factors may lead to this result, firstly, the bicycle characteristics given in the first frame are not obvious enough. Secondly, the speed of bicycle movements is fast. Therefore, we selected pedestrians in the same frames as a tracking object for comparison.

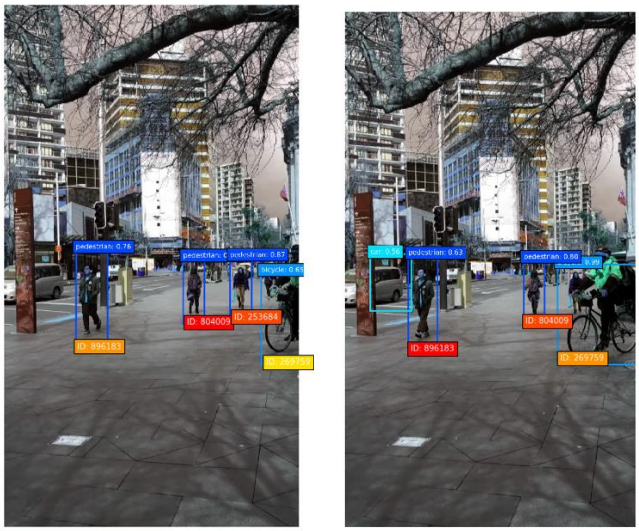

Figure 3.16 Video frames for object tracking

The blue bounding box is output by using SSD300, the orange bounding box is predicted by RPN. The results predicted by the two algorithms are similar. It is learned that if the object characteristics of the first frame are not obvious, or if the object moves

quickly resulting in a large change in position, the prediction of RPN is not as accurate as SSD. In this experiment, we applied SSD to predict the position of the object, then take advantages of SiamRPN to determine the similarity of the object in different frames. The video frames are presented in Figure 3.16.

Overall, we see in our dataset that SSD has better object positioning capabilities. Therefore, we employ the bounding box predicted by using SSD instead of the coordinate got by regression branch. At the same time, multi-target tracking is achieved using a loop design architecture.

3.5 Model Implementation

3.5.1 Detection Network

In order to save our time on manually labelling samples for network training, we apply transfer learning in the network to object detection. The focus of transfer learning is on storing knowledge while solving one problem and applying it to a different application. Transfer learning can be grouped into the sample-based transfer, feature-based transfer, model-based transfer, and relationship-based transfer.

Sample-based transfer learning completes knowledge transfer by using the calibrated samples in the source domain. Feature-based transfer fulfils the transfer by mapping the source and target domains to the same space and minimizing the distance between the source and target domains (Bengio, 2012). Model-based transfer combines the source and target domain models with samples to adjust the model's parameters. Relation-based transfer carried out the transfer of knowledge by learning the relationship between concepts in the source domain and then analogizing it to the target domain.

In this experiment, we transfer the learned model parameters to a new model in to help the new model training. We share the learned parameters or the knowledge learned by using the model to the new model to speed up and optimize the learning efficiency.

In most of the cases, models having been pre-trained can more or less improve generalization capabilities than the train-from-scratch model (Mishkin and Matas, 2015). Moreover, Yosinski explained the details on how transferable features are applied in deep neural networks. Deep Neural Network (DNN) is a hierarchical feature representation of data obtained through pretraining and takes advantage of high-level semantic classification. The bottom layer of the model is low-level semantic features such as edge information, color information, etc. The characteristics are constant in most of the classification tasks, while the difference is the high-level features, which also explains that the new datasets are usually exploited to update the last few layers of AlexNet and GoogLeNet weights to achieve a simple transfer. Figure 3.18 shows the example of transfer learning with CNN.

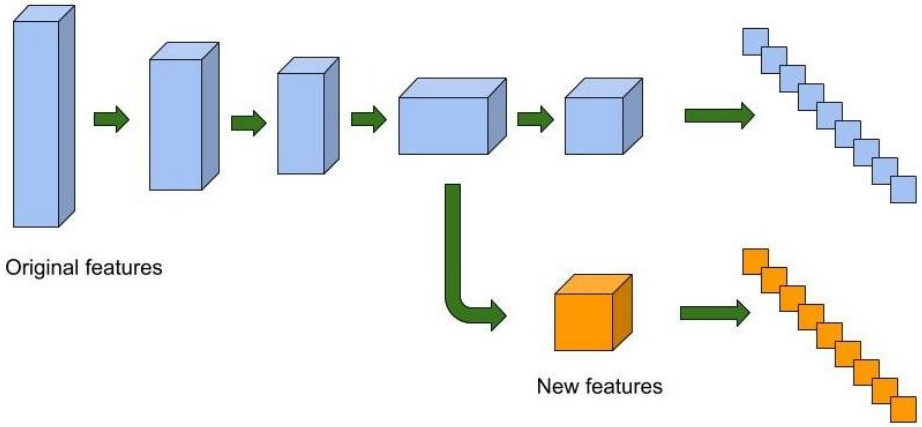

Figure 3.17 Transfer learning for CNN

In this experiment, we need to identify and classify six targets, motorcycles, scooters, bicycles, buses, cars and pedestrians. The MS COCO dataset has these visual objects, we download a fully trained original SSD model that has been trained based on MS COCO dataset first. Then fine-tuning is applied to the model based on our dataset.

SSD was trained based on MS COCO which can predict 80 different classes, but the dataset only includes six classes. The weight tensors of the classification layers of

the MS COCO model have not the right shape for this model, that is supposed to learn only six classes. It is the first time that we simply map the six classes in our dataset to six out of 80 that the MS COCO model predicts. The class IDs in our dataset show that only 6 out of every 80 neurons predict the class for a given box, the other 74 are not trained due to the gradient for them always is zero. After all, these labels will not appear in our dataset.

MS COCO dataset contains the six classes that we interest. When we sample the weight tensors of the classification layers, we pick exactly those elements from the tensor that is responsible for the classification of the same classes. We pick other elements randomly for the classes that are not contained in MS COCO. All the classification layers and the source weights perform the following steps:

- Get the kernel and bias tensors from the source weights file.
- Compute the sub-sampling indices for the last component.
- Overwrite the corresponding kernel and bias tensors in the destination weights file with our newly created sub-sampled kernel and bias tensors.

3.5.2 The Network for Object Tracking

The network is based on SimaRPN, which is a deep feature-based offline trained network. Compared with the filtering-based algorithm, the performance of SiamRPN can be improved by using the larger-scale data set, e.g., MOT16 dataset. Therefore, we employ MOT16 dataset to train the model. If IOU is greater than 0.6, it is the foreground; if it is less than 0.3, it is the background.

In inference phase, we modified the matching logic of template frames and detection frames due to the SimaRPN is a single target tracking network, in multitarget tracking task, the network needs to solve the problems of automatic initialization problem when new targets appear and automatic deletion problem when old targets disappear.

ALGORITHM : Tracking Algorithm

```
Initialization :     templist = extend2dimension(ssdResults)
For    each file in image file list, do
      If file is the first frame, do
                  Init each bounding box with box ID and count
      Else do
                  for each object in object list of pre-frame, do
                        feature(object) = GetFeature(object)
                        score(object) = GetScore(feature(object))
                        if score(object) < threshold
                              templist.append(object)
                        end
                  for each template object in templist, do
                        feature(object) = GetFeature(object)
                        score(object) = GetScore(feature(object))
                        scorelist.append(score(object))
                        if score(object) < threshold
                              template object count + 1
                        end
                  for each template object in templist, do
                        if template object count > threshold
                              remove object from template list
                        end
                        trackers = MatchID(scoreMatrix)
                        for each tracker in trackers, do
                              tracker.updateID()
                              tracker.updatePos()
                         end
       end
```

In the pseudocode, we get the classification result and bounding box from SSD, we need to convert SSD output into SimaRPN input. Then, we initialize the first frame of the video and save the objects that appeared in the first frame as a template list. Next, we obtain the targets appeared in *f*-th frame, and match each object with *f*-1 frame, respectively. If the target does not match in *f*-1 frame, it is considered a new object. We assign it a new ID and save it to the template list. After that, each target in the template list is matched in the current frame. If it is matched, the matched target is assigned as the same ID. If it is not matched, the number of disappearance times increases by 1.

The fourth step is to remove the objects in the template list that do not appear for three consecutive frames.

3.6 Evaluation Methods

3.6.1 Object Detection and Classification

When we consider applying deep learning to practical problems, we hope that the models are fast and accurate, take up less memory. For most common problems solved by using deep learning, there are usually multiple models available. Each model has its benefits and behaviors. Each model evaluates performance on the "verification / test" dataset, the performance is measured by various statistical metrics such as accuracy, precision, and recall.

Accuracy describes how accurate the model is, that is, how many the true examples are among the predictions that are positive examples. The accuracy is calculated as

$$\text{Accuracy} = \frac{TP+TN}{TP+NP+TN+FN} = \frac{TP+TN}{All\ detections} \tag{3.5}$$

where true positives (*TP*) is the number of positive examples that are correctly classified, the number of instances (samples) that are positive and classified by using the classifier as positive examples. True negative (*TN*) is the number of negative samples correctly predicted. False negatives (*FN*) is the number of negative cases that were incorrectly classified into negative class, that is, the number of instances is positive but grouped into negative class by the classifier.

Precision is the proportion of positive examples that are classified as positive, we precision as

$$\text{Precision} = \frac{TP}{FP+TP} = \frac{TP}{All\ detections} \tag{3.6}$$

where false positives (*FP*) is the number of cases that were incorrectly classified into

positive class, that is, the number of instances is negative but is classified as a positive class by the classifier.

Recall describes how complete the model is, namely, how many the true samples are predicted by using our model as a positive example. We describe the recall as

$$\text{Recall} = \frac{TP}{FN+TP} = \frac{TP}{All\ ground\ truths} \tag{3.7}$$

In computer vision, *mAP* is often utilized to evaluate the performance of an object detection system. For example, in the PASCAL VOC competition and MS COCO competition, MAP can be defined as equation 3.8.

$$mAP = \frac{\sum_{q=1}^{Q} AveP(q)}{Q} \tag{3.8}$$

where Q is the number of class, AP is the area covered by the Precision-Recall Curve (PRC), which is adopted to measure the quality of the model on each specific category. *mAP* reflects both the accuracy of the classification and the accuracy of the predicted location of the bounding box.

The number of bounding boxes predicted in object detection is often not fixed, but ground truth bounding boxes are fixed. We denote all the predicted boxes as Detection Targets (DT). Each DT contains its position coordinates and classification score. We sort the DTs according to the classification score from large to small. On the other hand, we denote all Ground Truth as GTs. Intersection Over Union (IOU) is handled as the threshold to mark whether the prediction box is correct or not, due to detection model outputs multiple prediction boxes that usually far exceeds the number of real boxes.

For each DT in the DTs, IOU and GTs are calculated. If the maximum IOU value is greater than the threshold, then the detection is considered successful. As *TP* and *GT* with the maximum IOU value are considered to be a successful match, they are removed from the GTs. If the maximum IOU is less than the threshold value, it means that the DT fails to match all GTs, which is an error Inspection, naturally counted as FP. There

will be multiple detection results DT matching with GT, the highest score (not the highest IOU value, but the classification score of the DT) is considered, the remaining detection results are considered *FP*, after traversing all the DTs, we get *TP* and *FP*, at this time, if there are remaining in the GTs due to all the matches are removed, it is considered FN.

In addition, PRC is utilized to evaluate the performance of the detection network. The PRC curve is obtained by using recall and precision. Precision describes the ability of a model to predict positive categories, meanwhile, recall depicts the ability of a model to correctly predict positive categories.

PRC employs recall as the abscissa and precision as the ordinate. Both precision and recall take advantage of the true positive as the numerator, precision is utilized of the true positive and false positive predicted by the classifier as the denominator. Meanwhile, recall takes use of the true positive and false negative as the denominator. The area under PRC curve is average precision (AP).

3.6.2 Object Tracking

For target tracking, the goal of this model is to find all targets in time (Bashir and Porikli, 2006). The target position should be consistent with the real target position as possible. Each target should be assigned a unique ID, this ID remains the same throughout the video sequence or the scene.

Multiple object tracking accuracies (MOTA) and multiple objects tracking precision (MOTP) jointly measure the algorithmic ability to continuously track objects (Chandan, Jain & Jain, 2018), a number of objects are accurately judged in consecutive frames, the positions are accurately determined to achieve uninterrupted continuously tracking.

MOTA is reflected in the accuracy of determining the number of targets and the relevant attributes of the object. It is employed to count the erratic accumulation in

tracking, including *FP*, *FN*, etc.

$$\text{MOTA} = 1 - \frac{\sum_t (m_t + fp_t + mme_t)}{\sum_t gt} \tag{3.9}$$

where m_t is the number of missing objects, object O_j should be matched, fp_t is the number of false positives (*FN*), *FN* refers to that there is no matching objects in the position h_j of *t*-th frame. mme_t is the number of times that ID switching occurs in the tracking target in the *t*-th frame.

MOTP is reflected in the accuracy of determining the target position, which measure the accuracy of determining the target position

$$\text{MOTP} = \frac{\sum_{i,t} d_t^i}{\sum_t c_t} m \tag{3.10}$$

where c_t indicates the number of matches between the object o_i and the hypothesis h_j in *t*-th frame, d_t^i indicates the distance between the object o_i and its paired hypothetical position in *t*-th frame.

Suppose there are targets in the image for each frame. The assumption of tracker output in this frame is $\{h_1, h_2 \ \ldots\ldots h_n\}$. The nearest neighbor method is adopted to match the hypothesis with the smallest distance to the corresponding target between the target and the hypothesis. The distance is calculated using Euclidean distance, the threshold *T* was set as the distance between the centers of the hypothesis and the target when they overlap least. The matching in (*t*+2)-th frame is an invalid match and the object o_j is a missing object.

We evaluate the consistency of object tracking as introduced (Manohar, et al. 2006). The optimal matching sequence is handled to count the entire video frames to construct a matching sequence between the hypothetical position and the target. $M_t = \{(o_i, j_i)\}$, M_t represents the matching sequence established up to frame t. At frame *t* + 1, if the matching object of o_i is h_j, which is different from the matching record in M_t, a mismatching is recorded, and (o_i, h_k) is updated to M_t. We continue to match the

subsequent frames.

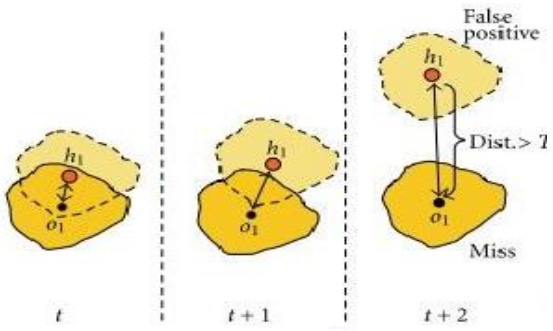

Figure 3.18 The example of the invalid match

For the problem of disappearing old objects in continuous tracking and reappearing after a few frames. Firstly, at frame t + 1, if the old object disappears and the currently tracked object does not have a hypothetical position that can be matched, the target is missed. Secondly, at frame t + 2, a new object appears and there are two hypothetical positions, two valid matches $(o_1,\ h_1)$, $(o_1,\ h_2)$ and $d_{11} > d_{12}$, we choose $(o_1,\ h_1)$ that already exists to reduce the error rate and the number of corresponding transitions.

The analysis of the tracking algorithm is for video sequences, assuming that the detected video has a total of N frames. A tracking match sequence should be created from the first frame, count the three errors, and maintain the match sequence at the flowing frames. The pseudo-code in Figure 3.19 illustrates the process.

Initialization: init $M_0 = \{.\}$
For t in frames:
    State $\leftarrow M_{t-1}(o_i, h_j)$
    If $d_{ij} <$ T
        $M_t \leftarrow (o_i, o_j)$
    Else
        $M_t \leftarrow (o_i, o_p)$
        MissMatch +1
    MOTP(o_i)
    MOTA(o_i)

Figure 3.19 The pseudocode for calculating MOTP and MOTA

MOTA and MOTP together measure the ability of an algorithm to continuously track a target. In continuous frames, the number of targets can be accurately tracked and their positions can be precisely defined, to achieve continuous tracking without interruption.

3.7 Experimental Environments

In our experiment, the model is implemented by using Python. The software dependencies are shown in Table 3.1.

Table 3.1 Software dependencies

Dependencies	Version
Keras	2.4
OpenCV	3.2
TensorFlow-GPU	1.10
Python	3.7

Execution environment:

Table 3.2 Execution environment

Software/Hardware	Version
NVIDIA GeForce	MX150
Anaconda	3.2
CUDA	10.0
CUDNN	7.4

Chapter 4

Results

Chapter 4 illustrates the results of our experiments, including detection accuracy, tracking accuracy, and tracking precision. We compare the accuracy of object detection by using SSD300 and SSD512. At the same time, the results of original SiamRPN and SSD for object tracking are also compared. The performance by using different backbone networks including AlexNet, ResNet50 and SiamFC based SiamRPN, SiamRPN++ and LSTM are evaluated.

4.1 The Results of SSD

In this thesis, we utilize SSD300 and SSD512 networks as a detector to observe the results of using neural networks with different depths. We compare the classification performance from aspects of accuracy, precision, and recall. Meanwhile, the tracking performance is revealed through mAP. Additionally, the performance from both the training set and the validation set could measure whether the model is overfitting.

Figure 4.1 shows the results of object classification and the bounding boxes detected by using SSD300. Almost all pedestrians and bicycle have been identified. Also, the cars that are far away from the camera, the bus occluded by another bus in front, were identified.

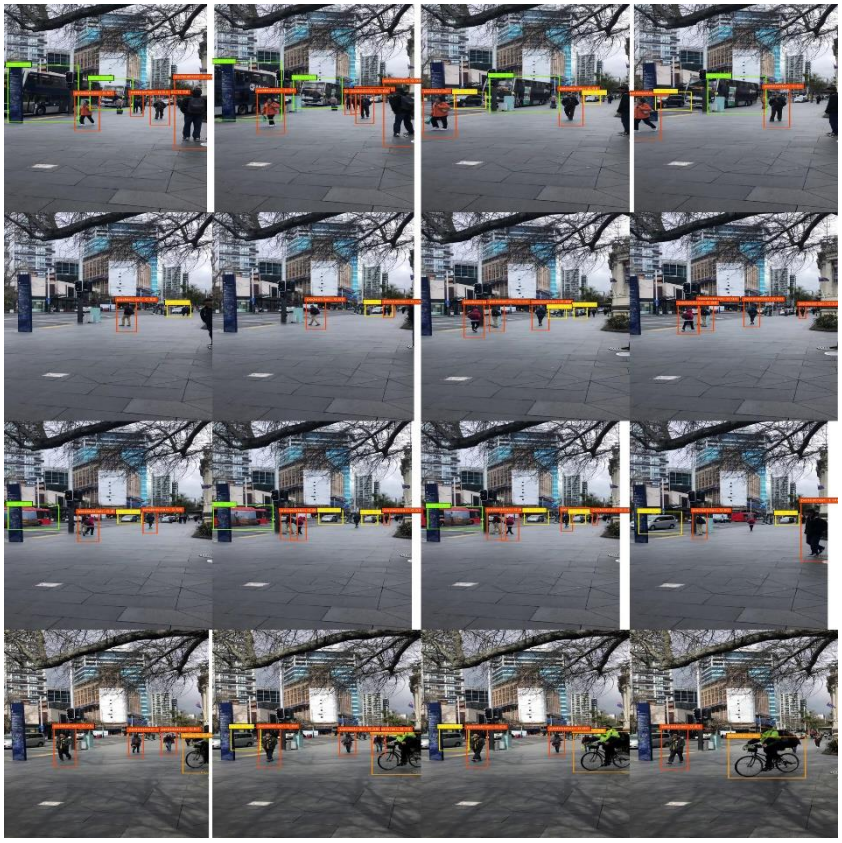

Figure 4.1 The selected video frames with bounding boxes

Table 4.1 demonstrates the classification accuracy, precision, and recall of the five classes after trained SSD300 network. We set the learning rate to 0.001, momentum to 0.9, weight decay to 0.0005, and batch size to 4. We train the model based on the image resolution 1080×1920.

Table 4.1 The performance of SSD300 network as a classifier

Classes	Accuracy		Precision		Recall	
	training	validation	training	validation	training	validation
Pedestrian	0.836	0.813	0.832	0.816	0.805	0.796
Bus	0.807	0.802	0.797	0.785	0.793	0.781
Bicycles	0.828	0.815	0.816	0.803	0.808	0.794
Scooters	0.643	0.634	0.631	0.622	0.601	0.588
Car	0.703	0.701	0.696	0.690	0.701	0.697
Average	0.777	0.765	0.759	0.747	0.752	0.739

In Table 4.1, the accuracy of pedestrian detection achieves the highest accuracy of the five classes (0.836), meanwhile, scooter detection has the lowest accuracy (0.643). All classes have achieved acceptable results that the average accuracy of the model based on our collected datasets is 0.777.

In the training process, the accuracy rate based on the training data set is close to the accuracy rate on the verification set. This indicates that the model has no overfitting. Figure 4.2 demonstrates the Precision and Recall Curve (PRC) of each class in the test dataset, the area under the PRC curve is Average Precision (AP).

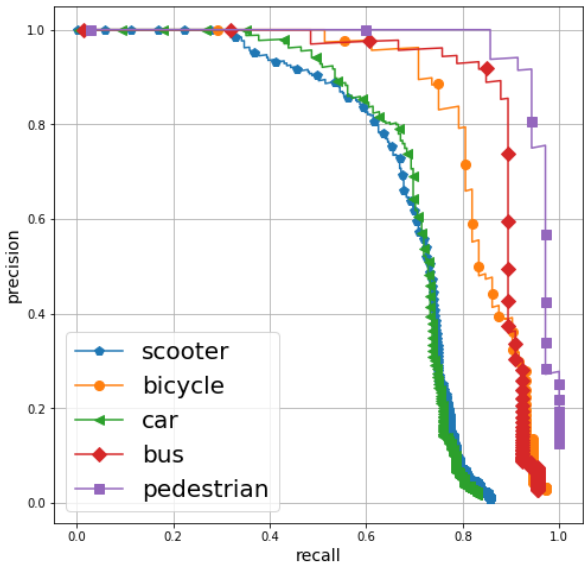

Figure 4.2 The PR curve of SSD300 as a classifier

The detection network works well in pedestrian class. Both the classification and

the location of the bounding box are accurate. The AP of pedestrians is 0.886. Pedestrians occupy the highest proportion of training data due to a picture often contains multiple pedestrians, therefore the performance of classification is also the best. The AP of the bus detection is 0.84 which is slightly lower than that of pedestrian detection. Compared with pedestrian detection, the training data for bus detection is relatively small, but buses occupy a large region in an image and have obvious visual features, the recognition result is good. The AP of car detection is 0.68, compared with buses, visual characteristics of cars are not so obvious, especially in the case of occlusion because cars are more easily obscured.

The Scooter samples have the lowest AP (0.655) amongst the five classes. The visual characteristics of the scooter are not obvious due to the region occupied by the scooter in the picture is relatively small, especially when a person riding the scooter appears with a frontal posture in the camera, it is easy to be misidentified as a pedestrian.

We manage to compare the accuracy of object detection by using SSD512 and our dataset. Table 4.2 shows the classification accuracy, precision, and recall of the five classes after training SSD512. We set the learning rate to 0.001, momentum to 0.9, weight decay to 0.0005, and batch size to 4. The model is trained based on the image size of 1080 ×1920.

Table 4.2 The performance of SSD512 network as a classifier

Classes	Accuracy		Precision		Recall	
	training	validation	training	validation	training	validation
Pedestrian	0.857	0.844	0.844	0.831	0.835	0.817
Bus	0.841	0.829	0.826	0.820	0.812	0.804
Bicycles	0.831	0.812	0.811	0.791	0.819	0.811
Scooters	0.655	0.639	0.601	0.599	0.606	0.591
Car	0.757	0.742	0.761	0.753	0.754	0.736
Average	0.796	0.775	0.768	0.758	0.765	0.751

We compared the classification performance of SSD300 and SSD512 in object detection. The average accuracy of SSD512 is 0.796, that of SSD300 is 0.777. Pedestrian detection accuracy is still the highest one (0.857), scooter detection is still

the lowest one with the value 0.655.

There is no significant difference between the accuracy of the training dataset and the accuracy of the verification dataset, which indicates no overfitting occlusion during the model training process. Figure 4.3 demonstrates the PR curve of each class.

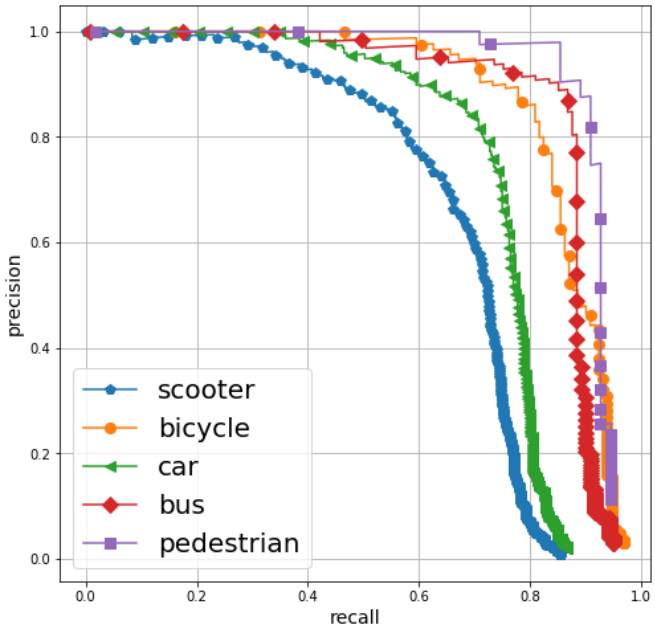

Figure 4.3 The PR curve of SSD512 as a classifier

The accuracy of SSD300 network and SSD512 network are almost the same, 0.836 and 0.857, respectively. The AP of pedestrian detection is the highest one among the five classes (0.892). The AP of the bus detection has been improved by 1.5% in SSD512, from 0.827 to 0.841. The AP of car detection is also improved from 0.68 to 0.78 by using SSD512. By contrast, the AP of bicycle detection has not been significantly improved. Similarly, the accuracy of bicycle detection in both networks is similar, 0.828 and 0.831 respectively. The accuracy of scooter detection has been improved by 1% in SSD512 network from 0.643 to 0.655.

Overall, in our experiment, the average accuracy of the detection network is improved by 2% in SSD512 network. In Table 4.3, we compare the average accuracy and speed of the two networks. The result shows that SSD512 has 2% higher accuracy, while the speed dropped 50%.

Table 4.3 The comparisons of mAPs and speed of SSD300 and SSD512 networks

Methods	mAPs	FPS	Batch sizes
SSD300	0.777	43	4
SSD512	0.796	20	4

4.2 The Results of SSD + Siamese Network + Hungarian Algorithm

Figure 4.4 illustrates the tracking results in our test video. During the tracking, we assign the same ID to the same object in different frames. The label on the top of the bounding box is the class of object, the label at the bottom of the bounding box is the object ID. The demo videos could be found on https://youtu.be/PHogpgKtDXY.

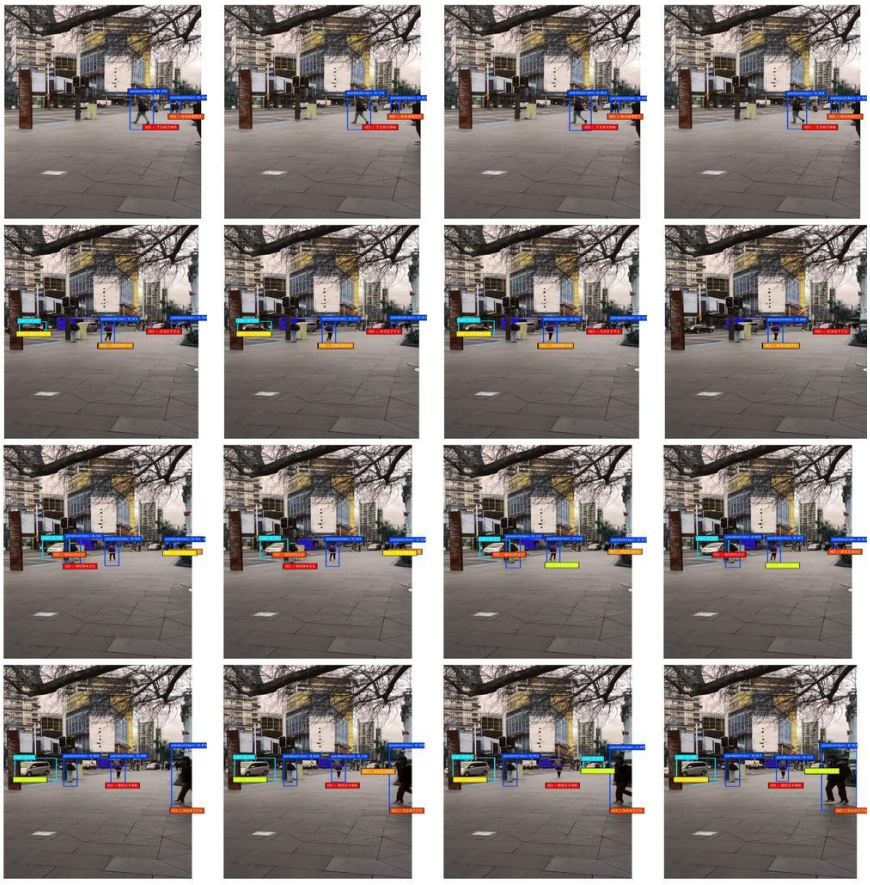

Figure 4.4 The selected results of SSD + SiamRPN + Hungarian algorithm

4.2.1 AlexNet as Backbone Network

We evaluate our model based on MOT16 challenging benchmark (Milan, et al. 2016). The MOT16 dataset was proposed in 2016 to measure the multi-target trackers. MOT16 consists of a total of 14 video sequences, all of which are still or moving cameras in an unconstrained environment. The detection results are already given in the data. Therefore, the results only focus on tracking performance. The evaluation is carried out according to the following metrics:

- Multi-Object Tracking Accuracy (MOTA): Summary of overall tracking accuracy in terms of false positives, false negatives, and identity switches.
- Multi-Object Tracking Precision (MOTP): Summary of overall tracking precision in terms of bounding box overlap between ground-truth and reported location.
- Mostly Tracked (MT): The percentage of ground-truth tracks that have the same label for at least 80% of their life span
- Mostly Lost (ML): The percentage of ground-truth tracks that are tracked for at most 20% of their life span
- Identity Switches (IDs): The number of times when the ID assigned by Ground Truth has changed
- Fragmentation (FM): The number of times that a track is interrupted by a missing detection

Table 4.4 presents the details of the tracker performance based on MOT16 dataset. The average accuracy of MOT dataset is 72.7%. Compared with the benchmark of MOT challenge, this result is acceptable.

Table 4.4 Tracking performance of AlexNet-Based SiamRPN

	MOTA(%)	MOTP(%)	MT	ML	FP	FN	IDs	FM
MOT16-02	17.8	74.1	7	27	501	13059	2005	189
MOT16-05	20.6	70.5	12	30	256	4092	965	154
MOT16-09	26.5	71.2	11	7	252	2054	274	114
MOT16-11	24.3	75.0	11	19	329	4687	559	174

Because MOT16 already gives the coordinates of detected objects, the results only reflect the performance of trackers. To verify the performance of the model in practical applications, precision plot and success plot are employed to evaluate the model on our collected test dataset. The precision plot and success plot were introduced in 2015 (Wen, et al., 2015). The overlap score (OS) is the bounding box, obtained by the tracking algorithm and the box is given by ground-truth.

For the single-target tracking algorithm, if the OS of a frame is greater than a given threshold (usually 0.5), the frame is considered successful. In this experiment, the successful object detection is defined as an object if the OS of this object is greater than the given threshold. The percentage of the total number of successful objects to the total number of objects is the successful rate. The value of OS ranges from 0 to 1, and a curve is drawn.

The precision plot reflects the deviation of object positioning. Location error is the distance between the center point of the bounding box estimated by the tracking algorithm and the center point of the ground-truth of objects. In the single-target tracking algorithm, the coordinates represent the percentage of video frames whose location error is less than a given threshold to the total video frames. In this experiment, the ordinate represents the ratio of the number of objects whose location error is less than a given threshold to the total objects. The success plot and precision plot for each class are displayed in Figure 4.5.

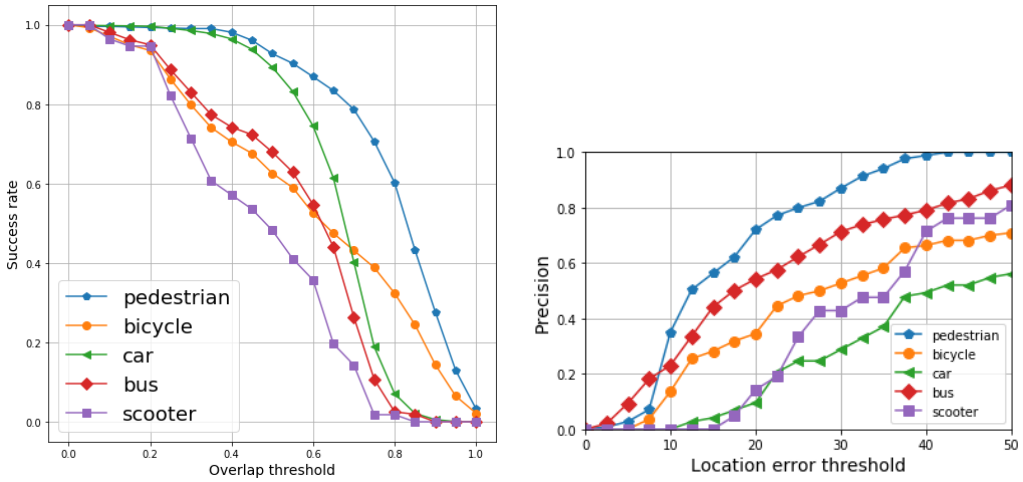

Figure 4.5 Success plot and precision plot of SSD300 + AlexNet-based SimaRPN

Pedestrian tacking has the highest success rate among the five classes if the overlap threshold is 0.5. The success rate of other classes is acceptable. However, as the overlap is worth increasing, the success rate of these bus and car tracking drops dramatically. In contrast, the tracking success rate of bicycle tracking is relatively stable. The scooter has the lowest success rate (0.5).

The location error of pedestrian tracking is the lowest one among the five classes that 70% pedestrian location error is around 15 pixels. Conversely, car detection has the highest location error. The location error of bus and bicycle is similar. Scooter detection has the lowest success rate, but the position deviation is acceptable. On the other hand, though the success rate of car and bus tracking is not much different, the deviation of car positions is much greater than that of buses.

The success plot and precision plot have been redefined in this work by reason of these two curves were originally utilized to measure the performance of single object tracking. On the other hand, all objects in our own datasets are manually labelled. Compared with the standard dataset, the quality of these labels will be affected by various reasons, such as the bounding box of the ground truth is too large or not large enough. These issues also affect the final measurement results.

4.2.2 SimaFC as Backbone Network

Compared with the AlexNet-based network, the performance of the pedestrian and bus tracking has been significantly improved. The performance of other trackers for car, scooter and bicycle tracking is similar to the Alexnet-based network. In general, the tracking performance does not benefit from the deeper neural networks. The performance of SimaFC-based tracking network is presented in Table 4.5.

Table 4.5 Tracking performance of SimaFC

	MOTA(%)	MOTP(%)	MT	ML	FP	FN	IDs	FM
MOT16-02	21.2	72.3	5	19	602	18306	2112	209
MOT16-05	19.5	74.1	8	11	279	4431	1045	186
MOT16-09	20.6	72.5	11	9	302	2025	398	214
MOT16-11	17.3	73.6	9	12	365	4672	609	173

The results of SiamFC-based tracking network are similar to AlexNet-based tracking network. The average MOTA of SiamFC-based tracking network is 19.1% while 22.3% in AlexNet-based tracking. The average accuracy of the two networks are 73.1% and 73.2%. Figure 4.6 shows the differences of success rate and location error between different classes.

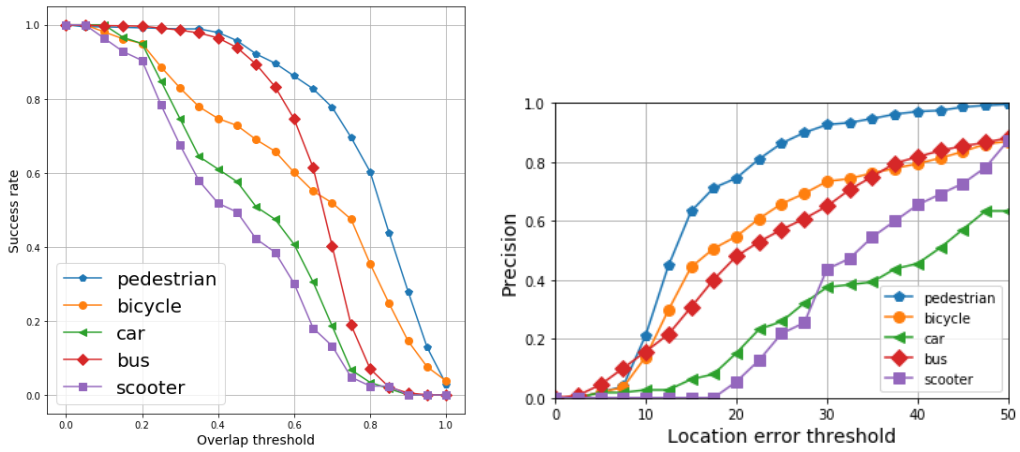

Figure 4.6 Success plot and precision plot of SiamFC

The tracking success rate of these five classes is not significantly different from that of the Alexnet-based network. Pedestrian tracking has the highest success rate 0.9 and the scooter has the lowest one 0.5 when the overlap threshold is 0.5. The success rate of the other three classes is between 0.55 and 0.8 which are close to that of Alexnet-based and Resnet-based network.

The location error of 70% of the pedestrians and bicycle is around 12 pixels which closer to the results in AlexNet-based network. While the average location error of 70% bus and bicycle are approximately 28 pixels. On the other hand, the category car has the largest position deviation. Compared with the class of bus, the position deviation of cars is large, though buses are larger than cars. This may due to the higher detection accuracy of buses in the detection network.

4.2.3 ResNet as Backbone Network

The backbone of the tracking network is AlexNet. In this experiment, we also compared the tracking network with other backbones in terms of ResNet50 and SiamFC to observe the impact of different CNNs on tracking accuracy.

Original SiamRPN is improved based on SiamFC. RPN module in Faster R-CNN is introduced to allow the tracker to return to the position and shape. It saves testing time and further improves the performance, but the backbone network still retains the original AlexNet. The algorithms make use of ResNet as the backbone since it solves the problem of gradient vanishing. Table 4.5 illustrates the tracking performance of ResNet50-based based on MOT16 dataset.

Table 4.6 Tracking performance of ResNet50-based SiamRPN

	MOTA(%)	MOTP(%)	MT	ML	FP	FN	IDs	FM
MOT16-02	23.9	72.5	9	21	552	12651	1985	154
MOT16-05	21.1	74.7	12	28	237	3862	963	161
MOT16-09	26.4	71.8	7	5	209	1988	379	120
MOT16-11	19.3	73.9	15	19	281	4326	495	149

The average accuracy based on MOTA dataset is 73.2% which is similar to AlexNet-based tracking network.

Additionally, we also evaluated the success rate and position deviation based on our test data set. Figure 4.6 shows the success plot and precision plot of each class.

If the overlapping threshold is set to 0.5, the success rate of pedestrian is 0.7. Similar to the results of AlexNet-based network, pedestrian has the highest accuracy. The success rates of bicycle, car and bus tracking are similar (83%), scooter has the lowest success rate (0.57). Compared with the AlexNet-based network, the success rate of these five classes have not been improved significantly.

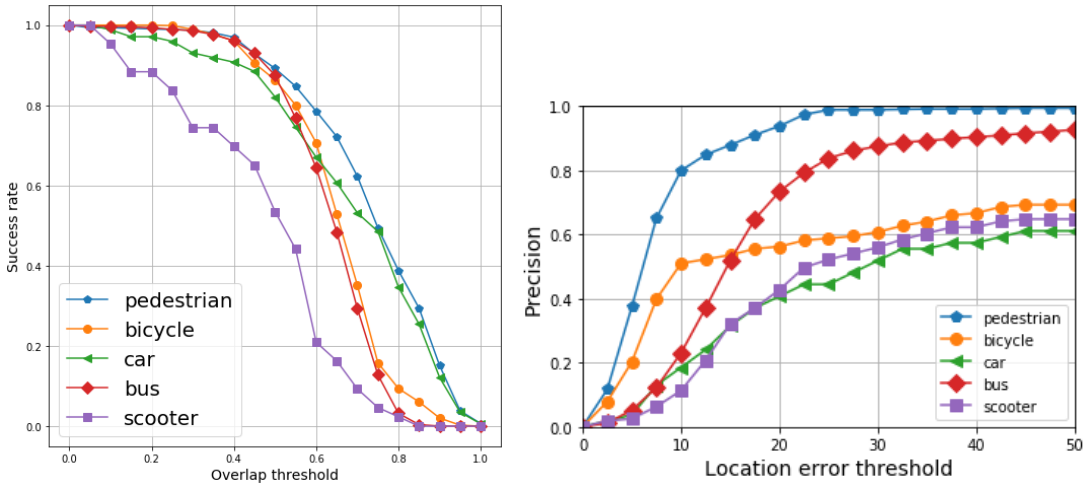

Figure 4.7 Success plot and precision plot of ResNet-based SiamRPN

The precisions of pedestrian and bus tracking are significantly improved, the location errors of 80% pedestrian and bus tracking are 10 pixels and 20 pixels which are better than previous networks. The location errors of 60% other three classes are around 30 pixels.

In our experiments, the performance of the tracking networks of different backbone networks has not changed much. This is inconsistent with our hypothesis that the deeper the neural network, the better the tracking performance.

4.2.4 SiamRPN++ as Backbone Network

SiamRPN has two branches: Classification branch and regression branch. The classification outputs the similarity between the template area and the detected area. The regression branch exports the location of the detected object. The classification is regarded for the match of the object as a dichotomy, if the feature of the detected region matches the feature of the template region, it is judged as the foreground. Otherwise, the judgment is the background.

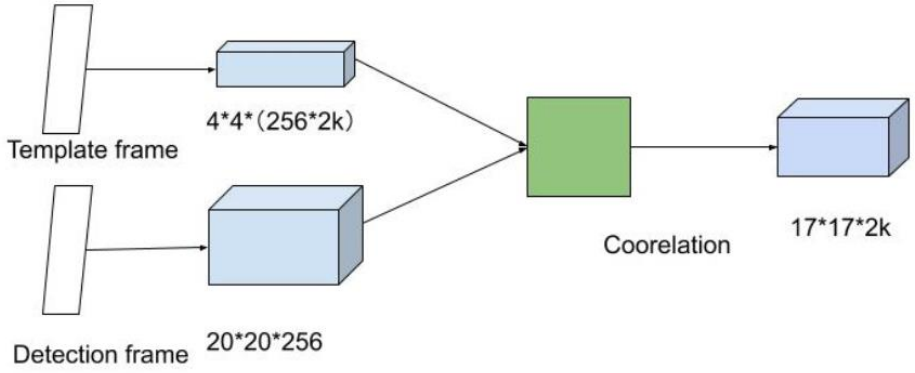

Figure 4.8 Correlation process

The correlation method is to treat the correlation layer as a convolutional layer, the features are extracted from the template branch as the convolution kernel, the features are extracted from the detection branch as the input of the convolution layer, so that the outputting channels became 2K by changing the shape of the convolution kernel. The specific method has two different convolutional layers. The convolutional layer of the template branch is responsible for dimensionality, the number of channels is increased to $256\times2k$. The detection branch also adds a convolutional layer, but the number of channels is constant. Then correlation operation is performed to get the final classification result.

The correlation is obtained by calculating the similarity of each position in the form of a sliding window. The similarity is defined as

$$f(z,x) = \emptyset(z) * \emptyset(x) + b \tag{4.1}$$

where z is a template image, x is a detection image, b_1 represents the value of each

position in the score chart. Equation (5.1) treats $\varphi(z)$ as a convolution kernel and performs convolution on $\varphi(x)$. During tracking, the response score map is calculated by the search image centered on the previous frame of the target position, the position with the largest score is collaborated with the step size to obtain the current target position.

Therefore, the network needs to satisfy strict translation invariance firstly. However, padding will destroy this nature. Secondly, the network needs to have symmetry, that is, if the images are searched, the output result should be unchanged.

For deeper networks, to ensure that the network has an appropriate resolution, almost all modern network backbones have a padding structure. ResNet does not have strict translation invariance (Bhat, et al., 2019), the introduction of padding makes the response of the network output having different perceptions for different positions. The training in this step makes that the network learns how to distinguish the regression objects, which limits the application of deep networks in the field of tracking.

Additionally, since SiamRPN supervises the regression offset and pre-background score, it no longer has symmetry. Therefore, in the improvement of SiamRPN, it is necessary to introduce asymmetric components.

An algorithm SiamRPN++ was proposed (Li, et al., 2019) which solves the problems by using optimized ResNet50. The stride of the last two blocks of ResNet50 was removed, and dilated convolution was added. One was to increase the receptive field, the other was to train the parameters. Moreover, multilayer fusion is used. The output of the last three blocks of the network is selected for fusion. The fusion method adopts the method of directly doing linear weighting.

Table 4.7 Tracking performance of SiamRPN++

	MOTA(%)	MOTP(%)	MT	ML	FP	FN	IDs	FM
MOT16-02	29.1	74.1	12	30	482	12964	1287	146
MOT16-05	25.8	73.6	11	2	236	3906	849	198
MOT16-09	28.2	75.1	21	23	205	2004	557	66
MOT16-11	21.5	72.9	15	29	327	4168	769	125

In the experiments, the performance of SiamRPN ++ in VOT data set is 4% higher

than that of SiamRPN. We replaced the SiamRPN with the SiamRPN++, the results are shown in Table 4.7.

Compared with SiamRPN, the MOTP of SiamRPN ++ is not significantly improved, while average MOTA is increased by 5%, which means that there are more matching objects. The success plot and precision of each class are displayed in Figure 4.9.

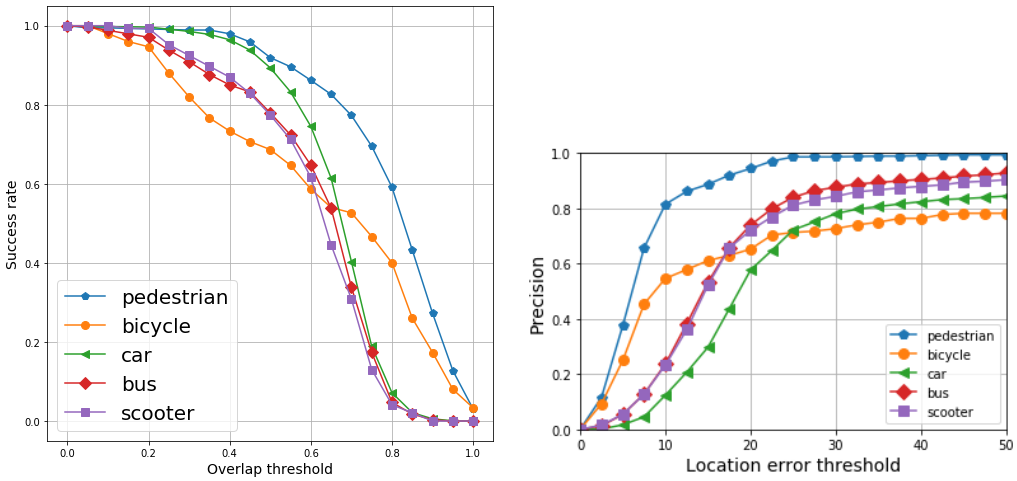

Figure 4.9 Success plot and precision plot of SiamRPN++

The average success rate of the five classes is improved significantly. The successful rate of the pedestrian is 0.9 if the overlapping threshold is set to 0.5. The successful rate of the car is close to the pedestrian, the area under the success rate is not significantly increase compared to SimaRPN algorithm. The successful rate of bus and scooter is similar, the results have been improved compared with the previous network. However, if the overlapping threshold becomes larger, the accuracy rate drops sharply.

The location error of 80% pedestrian tracking is 10 pixels. The location error of 70% cars is 28 pixels, the location error has not been improved much yet. The location error of 80% buses is 28 pixels. Both the success rate and the location error are improved, compared with the result to SiamRPN algorithm, the location error of 78% objects is around 20 pixels. The location error of 65% objects between the prediction box and the ground truth box is 18 pixels. Overall, compared to the tracking result of SiamRPN, the average tracking accuracy and positioning error of SiamRPN ++ have been improved.

4.3 SSD + SiamRPN++ + LSTM

SiamRPN ++ resolves the problem of loss of translation and symmetry in the SiamRPN network, improves the ability of correlation operations to calculate similarity. Considering that the positions of the objects in the front and back frames in the video sequence are related, Long Short Term Memory networks (LSTM) algorithm is utilized to predict the trajectory of the object to improve the accuracy of tracking. LSTM is a special RNN that can learn and store long-term dependencies (Hochreiter & Schmidhuber, 1997). The main purpose is to solve the problem of gradient disappearance and gradient explosion during long sequence training.

LSTM is employed for location prediction in objects tracking by learning the implicit motion model. An attention model is adopted to predict the bounding box of objects from the original image, which means that the LSTM learns both the features of objects and the position prediction (Kahou, 2017) (Zhou, Wan and Xiao, 2016). Ning et al. combined with the detection and tracking, fed the feature of objects and prediction results of detection into LSTM for further prediction, though the accuracy of tracking is improved, it can only track categories that have been trained in the detection model. Gordon proposed a model which takes advantage of two LSTM networks, the first LSTM learns the motion feature of the object, the second LSTM is responsible for regression, which is to output the diagonal coordinates of the target frame.

In this experiment, we take advantage of LSTM to predict the trajectory of an object in the template branch. Associated with data, the object which is closest to the predicted position is fed into SiamRPN++ for feature extraction and measure the feature similarity. In this way, the matching performance is improved. Also, we combined the results of LSTM prediction to update the bounding box of the template branch object.

Table 4.8 shows the tracking results on MOT16, we see a significant increase in the number of mostly tracked objects and a decrease of mostly lost objects. Then, we measure the performance of detection and tracking networks as a whole. The success plot and precision plot for each class are demonstrated in Figure 4.10.

Table 4.8 Tracking performance of SSD + SiamRPN++ and LSTM

	MOTA(%)	MOTP(%)	MT	ML	FP	FN	IDs	FM
MOT16-02	32.6	74.3	5	19	602	8306	705	309
MOT16-05	27.5	70.1	8	11	279	4431	410	132
MOT16-09	28.2	73.5	11	9	302	1036	323	185
MOT16-11	31.3	72.6	9	12	365	2493	581	208

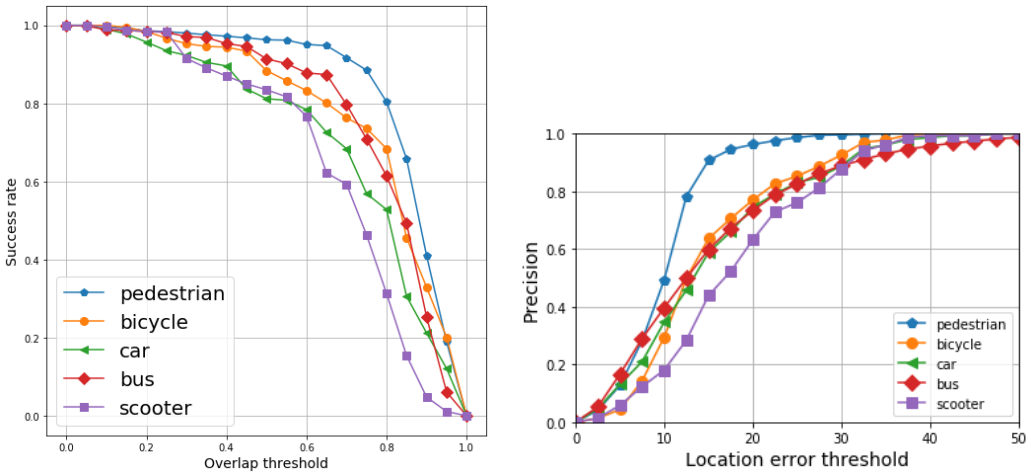

Figure 4.10 Success plot and precision plot of SSD + SiamRPN++ + LSTM

Pedestrian has the highest success rate. Meanwhile, the success rate of scooter is the lowest one of five classes, if the overlapping is 0.6. Overall, the success rates of the five classes are higher than 0.8.

We see that the position deviation of 60% of tracked objects is less than 20 pixels. 80% location error of pedestrians is around 12 pixels. The location error of car, bus and bicycle are similar, around 20 pixels when precision is 0.8. Meanwhile, the location error of scooter is slightly more than others that 25 pixels of the precision reaches to 0.8.

In general, the tracking performance of SSD + SiamRPN++ + LSTM model works well, the MOTA reaches 30.9% on the public data set MOT16. It also performs well on our own test dataset. If the ratio of the detection bounding box to the real bounding box is 0.6, 80%, the detected objects are successfully tracked. At the same time, LSTM

predicts the position of the detected object based on the timing characteristics to help improve the coordinates of the detected object, thereby reducing the location error of the tracked objects.

Overall, the successful rate of object detection by using SSD that was trained based on our own dataset is up to 77.7%, which is close to the results of the original results. The result of the tracking network based on MOT16 is also acceptable. The tracking performance on our own data set has been greatly improved due to the merged results of detection and tracking network.

Chapter 5

Analysis and Discussions

In this chapter, we discuss the results of the detection network by analyzing the differences from feature extraction between SSD300 and SSD512 networks. The tracking performance of the networks is also analyzed, we compared the tracking performance of network of the different backbones and the tracking results of five class of anomaly. Also, the results of the experiments are observed too.

5.1 Analysis and Discussions

5.1.1 Detection Network

In our experiments, we trained SSD on our own dataset, the average detection accuracy of the detection network on our dataset is 77.7%, which is closer to the result of Liu et al.'s work (81.6%) in their paper. The detection accuracy of SSD512 network is 1.5% higher than that of SSD300.

Compared with other single-stage detectors, the biggest bright spot of SSD is that it tends to utilize shallow layers to detect small targets and deep layers to detect large targets. This is the reason why shallow neural networks have more detailed information and are much effective for small targets, while deep neurons have larger receptive fields, abstract semantic information is much effective for large targets. The network structures of SSD300 and SSD512 networks are similar. SSD512 network has one more convolution block than that of SSD300.

SSD300 adopts a pyramid structure, which employs multiple feature maps to simultaneously perform classification and position regression, then merge these results. Meanwhile, SSD512 takes use of feature maps to classification and position regression, then combines the results. Figure 5.1 illustrates the feature maps of different layers of SSD300, we find the differences between these two algorithms in feature maps.

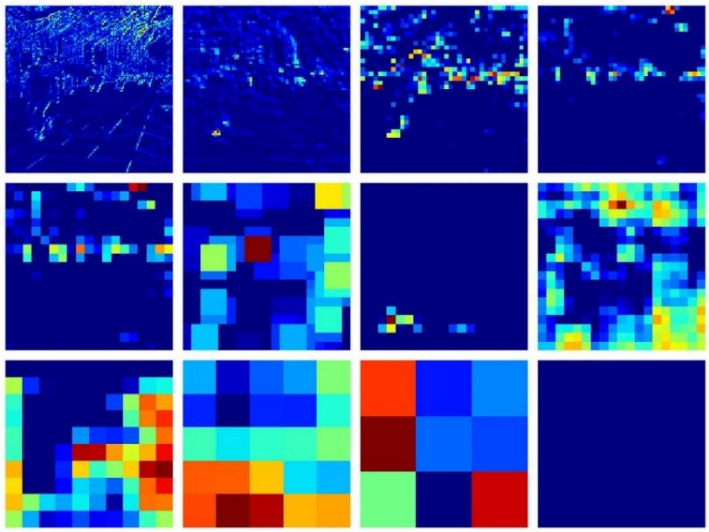

Figure 5.1 Feature map of SSD300 network

These feature maps come from pool1, pool2, pool3, conv4-3, cf6, cf7, conv6-2, conv7-2, conv8-2 and conv9-2 layers. As displayed in Figure 5.1, after the first convolutional network block, the network learns the features of the picture details such as points, lines, textures, etc. After the second convolutional block, the network learns larger-scale texture features. In the third and fourth convolutional network blocks, the network learns features in terms of position and color. We see from the feature maps, the resolution of the feature map is reduced layer by layer. As the SSD perception field becomes larger, the extracted features become more and more abstract.

The last layer of feature map becomes 1×1, it is too abstract to be interpreted. These multiple feature maps with different resolutions are employed for object detection. Low-level feature maps have learned fine surface features, and high-level feature maps have learned abstract features.

The input frame of SSD512 network is 512×512 which is greater than SSD300 (300×300), SSD512 has one more feature map of conv10-2 layer. Figure 5.2 shows the feature maps from layer pool1, pool2, pool3, conv4-3, pool5, fc6, fc7, conv6-2, conv7-2, conv8-2, conv9-2 and conv10-2.

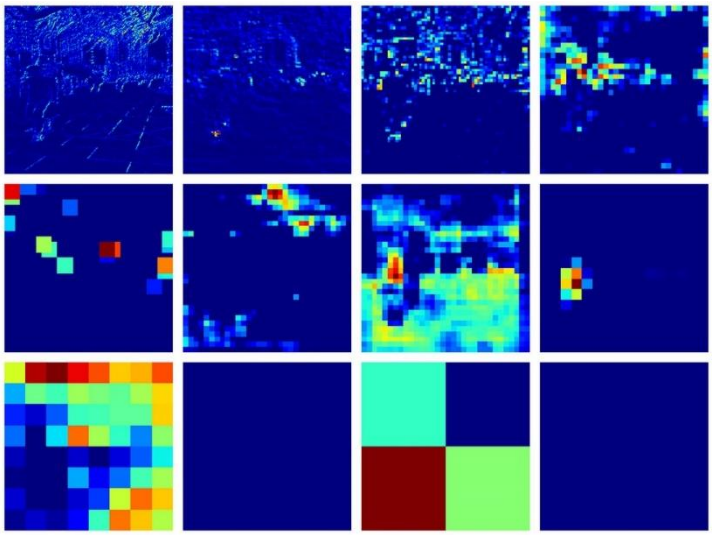

Figure 5.2 Feature map of SSD512 network

Compared with SSD300, the feature map of the first convolution block of SSD512 contains more line and texture features. The feature map of the conv4-3 layer in SSD512 contains more details than the feature map of the conv4-3 layer of SSD300.

We see the outline of the pedestrian in front.

The feature map of the SSD512 intermediate layer has a higher resolution than the SSD300 intermediate layer. Feature map of conv6-2 shows the feature of pedestrian that bigger than other objects.

Although SSD512 adopts one more layer feature map, compared with SSD300, the high-level feature map of SSD512 does not contain more information. Regarding the extra feature map that from conv10-2 layer of SSD512, it does not provide more information, meanwhile, the low-level feature map of SSD512 contains more information. This may result in the higher performance of SSD512.

5.1.2 Tracking Network

In the experiment, we make use of different deep networks as the backbone of SiamRPN as well as SimaRPN++ for feature extraction, Hungarian algorithm and LSTM to get the best matching objects. Table 5.1 compares the tracking performance on MOT16 dataset.

Table 5.1 Comparisons of various trackers

Algorithms	MOTA(%)	MOTP(%)	IDs
SiamRPN(Alexnet) + Hungarian algorithm	21.3	72.7	950
SiamRPN(Resnet) + Hungarian algorithm	22.6	73.2	945
SiamFC + Hungarian algorithm	21.1	73.9	1041
SiamRPN++(Resnet) + Hungarian algorithm	26.1	73.1	865
SiamRPN++(Resnet) + LSTM	30.9	72.6	504

The performance of SiamRPN has not significantly improved by using deeper neural network. While the SiamRPN++ and Hungarian algorithm improve the tracking performance by overcoming the barriers that the neural network loses translation equivalence in ResNet due to padding. If a neural network has padding and the objects are located in the center of the image for training, then when the neural network predicts

the target on the detection image, it will learn the prediction preference for the center of the image due to the distribution characteristics of the target in the training sample. No matter where the target moves to the image, the network will only predict the location of the central area. This is the reason why tracking performance does not increase but decreases after deepening the network with ResNet.

SiamRPN ++ takes advantages of the data of positive samples uniformly distributed at various positions in the image, which allows the network to learn a preference for each position. Hence, the tracking performance of the network will not decrease as the network deepens.

SiamRPN++ and LSTM algorithms have the highest MOTA, we see a significant decrease in ID switch. By combining the results of LSTM trajectory prediction and the results of SiamRPN feature similarity evaluation, the tracking accuracy is improved.

Overall, the model works well on our test dataset. We see that pedestrian has the highest tracking accuracy and scooter has the lowest tracking accuracy. These two types of objects also have the highest and lowest detection accuracy in the detection network respectively. The performance of the same tracking network based on SSD512 is a bit higher than that of tracking network based on SSD300. However, compared with other classes, the tracking success rate and positioning have not improved much during the experiment, especially the location error is still large. Thus, we compared the results of the car and the results of the bus.

As presented in Figure 5.3, in these two frames, a car is split into two parts by street traffic signs. If the network classifier finds that cars are blocked by street signs, the accuracy is low. The position of the car is different between the two frames, the predicted bounding boxes in these two frames are different from the ground truth bounding boxes. This leads to a large position error of the car.

Compared with cars, though buses are also divided into two parts by street signs, the accuracy of recognition is not affected.

Figure 5.3 Tracking result of cars and buses

The reason why the recognition algorithm is not accurate is that the data of car in the training model is not continuous video frames. It was selected from the videos which have not occlusion objects in order to ensure that the training is effective. Meanwhile, the test dataset is the video footage in the real world. It is inevitable that there are the images in which the car was blocked, these images have not been learned by the network.

5.2 Limitations of Research

In this experiment, we identify anomalies on the sidewalk and track them. We take a lot of videos and pick up frames as the dataset for detection network training. Additionally, we apply public datasets to train the tracking network. We train different neural networks as the backbone of the model. However, there are some limitations.

(1) Due to time constraints, it is impossible to study all algorithms in the related fields. There may be better solutions for abnormal object detection and tracking, while we did not found them.

(2) The quality of the training data is not high enough. Due to the frequency of bicycles and scooters appear on the sidewalk is low, there are not enough pictures taken. The characteristics of the target in some frames are not obvious caused by shooting angle.

(3) The data to measure the tracking network is manually annotated, which may lead to inaccurate performance evaluation.

(4) The network can’t identify enough types of anomalies due to there is not enough data. For example, other categories such as motorcycles and tricycles are not recognized.

Chapter 6
Conclusion and Future Work

This chapter outlines the implementation and training methods of the proposed model, preparation of experimental data, and experimental results. In addition, the shortcomings of this experiment and future work are discussed.

6.1 Conclusion

In this experiment, we detect anomalies on the sidewalk. We define anomaly as moving bicycles, scooters and motorcycles appeared on the sidewalk. We propose a network to identify and classify objects on the sidewalk. We track the anomaly objects in the video. Firstly, we develop SSD network as the detection network to detect the objects and classify them. Then, the detected objects and associated classes are fed into the modified SiamRPN network, the features of the tracked target in the template frame and the detection frame are extracted, respectively. Finally, we assess the similarity of these features to determine whether it is the same object.

In the process of model training, transfer learning is employed to save the time of manually labelling samples, so that the model could migrate from the existing marked data to the unmarked data. The data set for network training was taken in Queen street, Auckland, New Zealand. Since there is not enough data in the abnormal category, we adopts the Affine transformation, Gaussian noise injection and random erasing for data augmentation to avoid overfitting.

The model was developed by using Python 3.7. We compare the performance of SSD300 and SSD512 networks to recognize and classify objects. Our tracking network is based on SiamRPN, we assume that our model can track targets like our human beings, that is, extracting the key features after seeing the target and recognize them in the subsequent frames. Thus, we trained deep neural network to recognize similar objects.

The results of object detection and classification in our dataset are close to other related work. The performance of SiamRPN-based tracking network is well. We have replaced neural networks with different depths for feature extraction and similarity discrimination. But we can’t see a big difference in performance. Thus, we employ SiamRPN++ instead of SiamRPN, the tracking performance is improved by 5%. However, in the experiment, we found that in the process of data association, there are

cases where the similarity scores of a template object and multiple tracked objects all exceed the threshold, which results in matching errors. The reason why we obtained this result is that in multiple target tracking, there are multiple targets with similar characteristics. As a single target tracking algorithm, SiamRPN cannot distinguish multiple objects with similar apparent characteristics in the same frame. Therefore, we offer the Hungarian algorithm after the SiamRPN to get the best match of objects. Additionally, considering that the positions of the objects in the video sequence are related, we take advantage of the LSTM to predict the trajectory of the object and improve the accuracy of tracking. The results show significantly decrease in ID switch and 4% improvement of tracking accuracy. Overall, the main contributions of this thesis are:

- We implement the proposed model for object recognition, classification, and tracking for multiple types of anomalies.
- We achieve multi-target tracking by combining object detection algorithms and single-target tracking algorithms.
- The proposed model is a novel deep learning-based model to achieve anomaly detection and tracking, which has not been done in previous work.

6.2 Future Work

Our future work may include:

(1) The results of object detection and classification in this experiment are good, but the performance of object tracking is not ideal. Tracking objects by using data association may not be the best solution. In the future, LSTM can be employed to predict the trajectory of objects, thereby improves the accuracy of object tracking.

(2) The types of anomalies monitored by the network are limited. In future, more data can be trained to implement the detection of more abnormal categories.

(3) The network proposed in this experiment cannot tracking objects across cameras due to the similarity of the same target in different backgrounds predicted by SiamRPN is not accurate enough. Meanwhile, for the actual application of video surveillance, anomaly detection is usually multiple camera-based, for example, tracking the same suspicious target in different cameras. In the future, the algorithm can be optimized to achieve multi-target tracking by using these cameras.

References

Abdel-Aziz, A. S., Hassanien, A. E., Azar, A. T., & Hanafi, S. E. O. (2013). Machine learning techniques for anomalies detection and classification. In *International Conference on Security of Information and Communication Networks*, 219-229

Al-Sarayreh, M., Reis, M., Yan, W., Klette, R. (2019) A sequential CNN approach for foreign object detection in hyperspectral images. In *CAIP'19* (1): 271-283

Aslani, S., & Mahdavi-Nasab, H. (2013). Optical flow based moving object detection and tracking for traffic surveillance. *International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering*, *7*(9), 1252-1256.

Bashir, F., & Porikli, F. (2006). Performance evaluation of object detection and tracking systems. In *IEEE International Workshop on PETS* , 7-14.

Beal, M. J., Ghahramani, Z., & Rasmussen, C. E. (2002). The infinite hidden Markov model. In *Advances in Neural Information Processing Systems,* 577-584.

Bengio, Y. (2012). Deep learning of representations for unsupervised and transfer learning. In *ICML Workshop on Unsupervised and Transfer Learning,* 17-36.

Bewley, A., Ge, Z., Ott, L., Ramos, F., & Upcroft, B. (2016). Simple online and realtime tracking. In *IEEE International Conference on Image Processing (ICIP),* 3464-3468.

Bhat, G., Danelljan, M., Gool, L. V., & Timofte, R. (2019). Learning discriminative model prediction for tracking. In *IEEE International Conference on Computer Vision,* 6182-6191.

Bochinski, E., Senst, T., & Sikora, T. (2018). Extending IOU based multi-object tracking by visual information. In *IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS),* 1-6.

Chan, A., & Vasconcelos, N. (2008). UCDS pedestrian dataset. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, *30*(5), 909-926.

Chandan, G., Jain, A., & Jain, H. (2018). Real time object detection and tracking using deep learning and OpenCV. In *International Conference on Inventive Research in Computing Applications (ICIRCA)*, 1305-1308.

Chen, X., & Gupta, A. (2017). An implementation of Faster R-CNN with study for region sampling. *arXiv:1702.02138*.

Cho, H., Seo, Y. W., Kumar, B. V., & Rajkumar, R. R. (2014). A multi-sensor fusion system for moving object detection and tracking in urban driving environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, 1836-1843.

Chu, Q., Ouyang, W., Li, H., Wang, X., Liu, B., & Yu, N. (2017). Online multi-object tracking using CNN-based single object tracker with spatial-temporal attention mechanism. In *IEEE International Conference on Computer Vision*, 4836-4845.

Collins, R. T., Lipton, A. J., Kanade, T., Fujiyoshi, H., Duggins, D., Tsin, Y., ... & Wixson, L. (2000). A system for video surveillance and monitoring. *VSAM Final Report*, 1-68.

Cross, G. R., & Jain, A. K. (1983). Markov random field texture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (1), 25-39.

Cui, S., Tian, S., & Yin, X. (2019). Combined correlation filters with Siamese region proposal network for visual tracking. In *International Conference on Neural Information Processing*, 128-138

Denman, S., Fookes, C., & Sridharan, S. (2009). Improved simultaneous computation of motion detection and optical flow for object tracking. *Digital Image Computing: Techniques and Applications* IEEE. 175-182

Djenouri, Y., Belhadi, A., Lin, J. C. W., Djenouri, D., & Cano, A. (2019). A survey on urban traffic anomalies detection algorithms. *IEEE Access*, *7*, 12192-12205.

Dong, X., & Shen, J. (2018). Triplet loss in Siamese network for object tracking. In *European Conference on Computer Vision (ECCV)*, 459-474.

Ess, A., Schindler, K., Leibe, B., & Van Gool, L. (2010). Object detection and tracking for autonomous navigation in dynamic environments. *The International Journal of Robotics Research*, *29*(14), 1707-1725.

Feng, Y., Yuan, Y., & Lu, X. (2016). Deep representation for abnormal event detection in crowded scenes. In *ACM International Conference on Multimedia*, 591-595.

Feng, Y., Yuan, Y., & Lu, X. (2017). Learning deep event models for crowd anomaly detection. *Neurocomputing*, *219*, 548-556.

Fu, Z., Hu, W., & Tan, T. (2005). Similarity based vehicle trajectory clustering and anomaly detection. In *IEEE International Conference on Image Processing 2005* (Vol. 2, pp. II-602).

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*, 580-587.

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

Gordon, D., Farhadi, A., & Fox, D. (2018). Re^3: Real-time recurrent regression networks for visual tracking of generic objects. *IEEE Robotics and Automation Letters*, *3*(2), 788-795.

Gu, X., Cui, J., & Zhu, Q. (2014). Abnormal crowd behavior detection by using the particle entropy. *Optik*, *125*(14), 3428-3433.

Guo, Q., Feng, W., Zhou, C., Huang, R., Wan, L., & Wang, S. (2017). Learning dynamic Siamese network for visual object tracking. In *IEEE International Conference on Computer Vision*, 1763-1771.

Gu, Q., Yang, J., Yan, W., Li, Y., Klette, R. (2017) Local Fast R-CNN flow for object-centric

event recognition in complex traffic scenes. In *PSIVT'17 Workshops*: 439-452.

Gu, Q., Yang, J., Yan, W., Klette, R. (2017) Integrated multiscale event verification in an augmented foreground motion space. In *PSIVT'17*: 488-500.

Gu, Q., Yang, J., Kong, L., Yan, W., Klette, R. (2017) Embedded and real-time vehicle detection system under challenging on-road scenes. SPIE Optical Engineering (SCIE), 56(6): 063102.

Hastie, T., Rosset, S., Zhu, J., & Zou, H. (2009). Multi-class AdaBoost. *Statistics and Its Interface*, *2*(3), 349-360.

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. In *IEEE International Conference on Computer Vision*, 2961-2969.

Hinton, G. E., Krizhevsky, A., & Sutskever, I. (2012). Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, *25*, 1106-1114.

Hochreiter, S., & Schmidhuber, J. (1997). LSTM can solve hard long time lag problems. In *Advances in Neural Information Processing Systems*, 73-479.

Hsu, C. W., & Lin, C. J. (2002). A comparison of methods for multiclass support vector machines. *IEEE transactions on Neural Networks*, *13*(2), 415-425.

Hsu, C. Y., Lu, C. S., & Pei, S. C. (2012). Image feature extraction in encrypted domain with privacy-preserving SIFT. *IEEE Transactions on Image Processing*, *21*(11), 4593-4607.

Huang, Z., Zhan, J., Zhao, H., Lin, K., Zheng, P., & Lv, J. (2019). Real-time visual tracking base on SiamRPN with generalized intersection over union. In *International Conference on Brain Inspired Cognitive Systems*, 6-105

Kahou, S. E., Michalski, V., Memisevic, R., Pal, C., & Vincent, P. (2017). RATM: Recurrent attentive tracking model. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 1613-1622.

Kim, C., Li, F., Ciptadi, A., & Rehg, J. M. (2015). Multiple hypothesis tracking revisited. In *IEEE International Conference on Computer Vision*, 4696-4704.

Kim, H., Lee, Y., Yim, B., Park, E., & Kim, H. (2016). On-road object detection using deep neural network. In *IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)* (pp. 1-4).

Ji, H., Liu, Z., Yan, W., Klette, R. (2019) Early diagnosis of Alzheimer's disease based on selective kernel network with spatial attention. In *ACPR'19* (2): 503-515.

Jana, A. P., & Biswas, A. (2018). YOLO based detection and classification of objects in video records. In *IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, 2448-2452.

Lemley, J., Bazrafkan, S., & Corcoran, P. (2017). Smart augmentation learning an optimal data augmentation strategy. *IEEE Access*, *5*, 5858-5869.

Li, B., Yan, J., Wu, W., Zhu, Z., & Hu, X. (2018). High performance visual tracking with

siamese region proposal network. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 8971-8980).

Li, B., Wu, W., Wang, Q., Zhang, F., Xing, J., & Yan, J. (2019). SiamRPN++: Evolution of Siamese visual tracking with very deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4282-4291).

Li, D., Wang, X., & Yu, Y. (2019). Siamese visual tracking with deep features and robust feature fusion. In *IEEE International Conference on Consumer Electronics-Asia*, 16-34.

Li, F., Zhang, Y., Yan, W., Klette, R. (2016) Adaptive and compressive target tracking based on feature points matching. In *IEEE ICPR'16*, Mexico.

Li, W., Mahadevan, V., & Vasconcelos, N. (2013). Anomaly detection and localization in crowded scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *36*(1), 18-32.

Liu, C., Yan, W. (2020) Gait recognition using deep learning. In IGI Global Handbook of Research on Multimedia Cyber Security, pages 214-226.

Liu, M., Lei, Q., Yu, L., Gao, Y., & Zhang, X. (2019). Visual object tracking via an improved lightweight Siamese network. In *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)* (pp. 284-294).

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*, 21-37.

Liu, X., Neuyen, N., Yan, W. (2019) Vehicle-related scene understanding using deep learning. In *ACPR'19 Workshops*: 61-73.

Liu, Z., Yan, W., Yang, B. (2018) Image denoising based on a CNN model. In *ICCAR'18*, pp. 389-393.

Lotter, W., Kreiman, G., & Cox, D. (2015). Unsupervised learning of visual structure using predictive generative networks. *arXiv:1511.06380*.

Lu, C., Shi, J., & Jia, J. (2013). Abnormal event detection at 150 fps in MATLAB. In *IEEE International Conference on Computer Vision* (pp. 2720-2727).

Lu, J., Shen, J., Yan, W., Bacic, B. (2017) An empirical study for human behavior analysis. IJDCF 9(3): 11-27.

Lu, J., Yan, W., Nguyen, M. (2018) Human behavior recognition using deep learning. In *IEEE AVSS'18*: 1-6.

Lu, J., Nguyen, M., Yan, W. (2020) Comprehensive evaluations of deep learning models for human behavior recognition. IGI Global Handbook of Research on Multimedia Cyber Security, pages 176-189.

Mahadevan, V., Li, W., Bhalodia, V., & Vasconcelos, N. (2010). Anomaly detection in crowded scenes. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1975-1981).

Manohar, V., Soundararajan, P., Raju, H., Goldgof, D., Kasturi, R., & Garofolo, J. (2006).

Performance evaluation of object detection and tracking in video. In *Asian Conference on Computer Vision* (pp. 151-161).

Melekhov, I., Kannala, J., & Rahtu, E. (2016). Siamese network features for image matching. In *International Conference on Pattern Recognition (ICPR)* (pp. 378-383).

Miao, Y., & Song, J. (2014). Abnormal event detection based on SVM in video surveillance. In *IEEE Workshop on Advanced Research and Technology in Industry Applications (WARTIA)* (pp. 1379-1383).

Milan, A., Leal-Taixé, L., Reid, I., Roth, S., & Schindler, K. (2016). MOT16: A benchmark for multi-object tracking. *arXiv:1603.00831*.

Mizuno, K., Terachi, Y., Takagi, K., Izumi, S., Kawaguchi, H., & Yoshimoto, M. (2012). Architectural study of HOG feature extraction processor for real-time object detection. In *IEEE Workshop on Signal Processing Systems* (pp. 197-202).

Murray, D., & Basu, A. (1994). Motion tracking with an active camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *16*(5), (pp. 449-459).

Ning, C., Zhou, H., Song, Y., & Tang, J. (2017). Inception single shot multi-box detector for object detection. In *IEEE International Conference on Multimedia & Expo Workshops (ICMEW)* (pp. 549-554).

Ning, G., Zhang, Z., Huang, C., Ren, X., Wang, H., Cai, C., & He, Z. (2017). Spatially supervised recurrent convolutional neural networks for visual object tracking. In *IEEE International Symposium on Circuits and Systems (ISCAS)* (pp. 1-4).

Pan, C., Li, X., Yan, W. (2018) A learning-based positive feedback approach in salient object detection. In *IVCNZ'18*: 1-6.

Pan, C., Yan, W. (2020) Object detection based on saturation of visual perception. Multimedia Tools and Applications (https://doi.org/10.1007/s11042-020-08866-x).

Pan, S. J., & Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345-1359.

Parekh, H. S., Thakore, D. G., & Jaliya, U. K. (2014). A survey on object detection and tracking methods. *International Journal of Innovative Research in Computer and Communication Engineering*, *2*(2), 2970-2978.

Perez, L., & Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. *arXiv:1712.04621*.

Purkait, P., Zhao, C., & Zach, C. (2017). SPP-Net: Deep absolute pose regression with synthetic views. *arXiv:1712.03452*.

Putra, M. H., Yussof, Z. M., Lim, K. C., & Salim, S. I. (2018). Convolutional neural network for person and car detection using YOLO framework. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, *10*(1-7), 67-71.

Raghavendra R, Del Bue A, Cristani M, (2016). Optimizing interaction force for global anomaly detection in crowded scenes. *IEEE International Conference on Computer*

Vision Workshops (ICCV Workshops). 136-143.

Rao, A. S., Gubbi, J., Rajasegarar, S., Marusic, S., & Palaniswami, M. (2014). Detection of anomalous crowd behavior using hyperspherical clustering. In *International Conference on Digital Image Computing: Techniques and Applications (DICTA)* (pp. 1-8).

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

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).

Ryan, D., Denman, S., Fookes, C., & Sridharan, S. (2011). Textures of optical flow for real-time anomaly detection in crowds. In *IEEE International Conference on Advanced Video and Signal-Based Surveillance* (pp. 230-235).

Sadeghian, A., Alahi, A., & Savarese, S. (2017). Tracking the untrackable: Learning to track multiple cues with long-term dependencies. In *IEEE International Conference on Computer Vision* (pp. 300-311).

Shafiee, M. J., Chywl, B., Li, F., & Wong, A. (2017). Fast YOLO: A fast you only look once system for real-time embedded object detection in video. *arXiv:1709.05943*.

Shen, C., Jin, Z., Zhao, Y., Fu, Z., Jiang, R., Chen, Y., & Hua, X. S. (2017). Deep Siamese network with multi-level similarity perception for person re-identification. In *ACM International Conference on Multimedia* (pp. 1942-1950).

Shen, Y., Yan, W. (2018) Blind spot monitoring using deep learning. In *IVCNZ'*18: 1-5.

Song, C., He, L., Yan,W., Nand, P. (2019) An improved selective facial extraction model for age estimation. In *IVCNZ'19*: 1-6

Sultani, W., Chen, C., & Shah, M. (2018). Real-world anomaly detection in surveillance videos. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 6479-6488).

Tang, S., Andres, B., Andriluka, M., & Schiele, B. (2016). Multi-person tracking by multicut and deep matching. In *European Conference on Computer Vision* (pp. 100-111).

Tang, S., Andriluka, M., Andres, B., & Schiele, B. (2017). Multiple people tracking by lifted multicut and person re-identification. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3539-3548).

Tian, Y., Yang, G., Wang, Z., Wang, H., Li, E., & Liang, Z. (2019). Apple detection during different growth stages in orchards using the improved YOLOv3 model. *Computers and Electronics in Agriculture*, *157*, 417-426.

Varior, R. R., Shuai, B., Lu, J., Xu, D., & Wang, G. (2016). A Siamese long short-term memory architecture for human re-identification. In *European Conference on Computer Vision* (pp. 135-153).

Vishwanathan, S. V. M., & Murty, M. N. (2002). SSVM: A simple SVM algorithm. In *International Joint Conference on Neural Networks (Cat. No. 02CH37290)* (Vol. 3, pp. 2393-2398).

Wang, J., Yan, W. (2016) BP-neural network for plate number recognition. IJDCF 8(3): 34-45

Wang, J. Ngueyn, M., Yan, W. (2017) A framework of event-driven traffic ticketing system. IJDCF 9(1): 39-50.

Wang, J., Bacic, B., Yan, W. (2018) An effective method for plate number recognition. Multim. Tools Appl. 77(2): 1679-1692.

Wang, J., & Xu, Z. (2015). Crowd anomaly detection for automated video surveillance. In *International Conference on Imaging for Crime Prevention and Detection.* London, UK, (pp. 1-6)

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

Wang, X., Feng, S., Yan, W. (2019) Human gait recognition based on SAHMM. IEEE/ACM Transactions on Biology and Bioinformatics.

Wang, X., Zhang, J., Yan, W. (2019) Gait recognition using multichannel convolution neural networks. Neural Computing and Applications (https://doi.org/10.1007/s00521-019-04524-y)

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

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

Wen, L., Du, D., Cai, Z., Lei, Z., Chang, M. C., Qi, H., ... & Lyu, S. (2015). UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking. *arXiv:1511.04136*.

Wojke, N., Bewley, A., & Paulus, D. (2017). Simple online and real time tracking with a deep association metric. In *IEEE International Conference on Image Processing (ICIP)* (pp. 3645-3649).

Womg, A., Shafiee, M. J., Li, F., & Chwyl, B. (2018). Tiny SSD: A tiny single-shot detection deep convolutional neural network for real-time embedded object detection. In *15th Conference on Computer and Robot Vision (CRV)* (pp. 95-101).

Wong, S. C., Gatt, A., Stamatescu, V., & McDonnell, M. D. (2016). Understanding data augmentation for classification: when to warp? In *International Conference on Digital Image Computing: Techniques and Applications (DICTA)* (pp. 1-6).

Wu, E., Wu, K., Cox, D., & Lotter, W. (2018). Conditional infilling GANs for data augmentation in mammogram classification. In *Image Analysis for Moving Organ, Breast, and Thoracic Images* (pp. 98-106).

Wu, Y., Lim, J., & Yang, M. H. (2015). Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(9), 1834-1848.

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).

Xu, L., Gong, C., Yang, J., Wu, Q., & Yao, L. (2014). Violent video detection based on MoSIFT feature and sparse coding. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 3538-3542).

Yadav, N., & Binay, U. (2017). Comparative study of object detection algorithms. *International Research Journal of Engineering and Technology (IRJET)*, *4*(11), 586-591.

Yan, W. (2019) Introduction to Intelligent Surveillance - Surveillance Data Capture, Transmission, and Analytics (Third Edition), Springer, pp. 1-212.

Yanagisawa, H., Yamashita, T., & Watanabe, H. (2018). A study on object detection method from manga images using CNN. In *International Workshop on Advanced Image Technology (IWAIT)* (pp.1-4).

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems* (pp. 3320-3328).

Yuan, J., Guo, H., Jin, Z., Jin, H., Zhang, X., & Luo, J. (2017). One-shot learning for fine-grained relation extraction via convolutional Siamese neural network. In *IEEE International Conference on Big Data (Big Data)* (pp. 2194-2199).

Yuan, Z. W., & Zhang, J. (2016). Feature extraction and image retrieval based on AlexNet. In *International Conference on Digital Image Processing (ICDIP 2016)* (Vol. 10033, p. 100330E).

Zhan, Y., Fu, K., Yan, M., Sun, X., Wang, H., & Qiu, X. (2017). Change detection based on deep siamese convolutional network for optical aerial images. *IEEE Geoscience and Remote Sensing Letters*, *14*(10), 1845-1849.

Zhang, Y., Wang, D., Wang, L., Qi, J., & Lu, H. (2018). Learning regression and verification networks for long-term visual tracking. *arXiv:1809.04320*.

Zhang, Z., Qiao, S., Xie, C., Shen, W., Wang, B., & Yuille, A. L. (2018). Single-shot object detection with enriched semantics. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 5813-5821).

Zhao, Z., Chen, W., Wu, X., Chen, P. C., & Liu, J. (2017). LSTM network: A deep learning approach for short-term traffic forecast. *IET Intelligent Transport Systems*, *11*(2), (pp. 68-75).

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

Zhong, Z., Zheng, L., Kang, G., Li, S., & Yang, Y. (2017). Random erasing data augmentation.

arXiv:1708.04896.

Zhou, X., Wan, X., & Xiao, J. (2016). Attention-based LSTM network for cross-lingual sentiment classification. In *Conference on Empirical Methods in Natural Language Processing* (pp. 247-256).

Zhu, J., Yang, H., Liu, N., Kim, M., Zhang, W., & Yang, M. H. (2018). Online multi-object tracking with dual matching attention networks. In *European Conference on Computer Vision (ECCV)* (pp. 366-382).