Traffic Sign Recognition Based on Deep Learning

Yanzhao Zhu

A project report submitted to the Auckland University of Technology

in partial fulfillment of the requirements for the degree of

Master of Computer and Information Sciences (MCIS)

2021

School of Engineering, Computer & Mathematical Sciences

Abstract

Intelligent Transportation System (ITS), including unmanned driving, has gradually matured and is about to enter the practical phase. How to eliminate the interference of various environmental factors, carry out accurate and efficient traffic sign detection and recognition, which is a key technical problem that must be solved. In addition, drivers neglect the information of the traffic instructions while driving is the main reason of many traffic accidents. Therefore, how to use computer visualization to efficiently identify traffic signs in road environment is a research topic having positive significance. However, traditional visual object recognition mainly relies on the information extracted, e.g., color, and edge, which has limitations. The convolution neural network (CNN) is designed for visual object recognition based on deep learning, which has well solved the shortcomings exisiting in traditional object recognition.

In this project, we implement an experiment to evaluate the performance of the latest version YOLOv5 based on the dataset for traffic sign recognition in New Zealand, which determines how the model for visual object recognition based on deep learning is much suitable for the traffic sign recognition through a comprehensive comparison with SSD neural network. The experiments in this project utilities our own dataset. Pertaining to the experimental results, YOLOv5 achieves 97.7% in terms of mAP@0.5 for overall classes, SSD obtains 90.14% in the same term. Meanwhile, with regard to recognition speed, YOLOv5 also outperforms SSD.

Keywords: Deep Learning, Traffic Sign Recognition, CNN, YOLOv5, SSD Neural Network

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.

Acknowledgment

First of all, I would like to sincerely appreciate my primary supervisor Wei Qi Yan. During the period of researching my project, Dr. Yan always patiently provided extremely professional advice in academic and led me to solve one problem after another. Moreover, he persisted in giving us theoretical lectures in the field of deep learning regularly and kept sharing the latest cutting-edge achievements with us. His spirit of dedication to research deeply influenced me. I believe that is what I am most worth learning and a lifelong benefit.

Secondly, I am very grateful to my family. That was them who gave me lots of encouragement to make the decision to study for this master degree after a few years of graduation from University. When I was hesitant and wanted to flinch, that was them who always stood behind me to inspire me and support me. Meanwhile, that was still them who accompanied me through this unforgettable time.

Eventually, I will thank everyone around me who loves and cares me. Due to the special situation of Covid-19, it's a tough period for everyone. Please be healthy and be stronger. There is a Chinese proverb, "Life is long, fortunately, to have you".

Yanzhao Zhu

Auckland, New Zealand

June 2021

Chapter 1
Introduction

This chapter comprises five sections. In the first section, it mainly introduces the background and motivation of this project. Then, we raise two research questions in the second section. In the following two sections, we describe contributions and objectives, respectively. The structure of this thesis is briefly listed at the end of this chapter.

1.1 Background and Motivation

In recent years, with the breakthroughs in the field of Artificial Intelligence (AI), the introduction of vehicle aided driving system has changed the previous driving mode. By acquiring real-time road condition information, the system promptly reminds the driver to make accurate operations, thereby preventing car accidents due to fatigued driving. In addition to the auxiliary driving systems, the development of autonomous vehicles also requires rapid and accurate detection of traffic signs from digital images.

Traffic signs are most important traffic information for drivers, how to identify the traffic signs quickly and accurately is very important and necessary. Traffic Sign Recognition (TSR) is to detect the location of traffic signs from digital images or video frames, gives a specific classification. First of all, the candidate area of traffic signs should be obtained from the original images, and then extract the features of the candidate area. Finally, classify and locate the extracted features to determine the traffic sign class and the location of the traffic signs.

The traditional traffic sign detection and recognition method basically take use of characteristic information such as the shape and color of traffic signs. However, the traditional TSR algorithms face some drawbacks in real-time testing, such as being easily restricted by factors, which are light, angle, obstruction, driving speed and so on. It's also very difficult to achieve multitarget detection, easy to miss detection, and slow recognition.

In the past few years, with continuous improvement of computer hardware performance, the limitation of neural networks by computing power conditions has been well alleviated, which has brought machine learning into a golden period of development. Deep learning is a type of machine learning methods proposed by Hinton and Salakhutdinov (2006). Deep learning means a deep neural network model by simulating the neural structure used by our human brain while processing information. Using this neural network model to extract the effective features in the

original road image is able to make up for the dependence on manual feature extraction in traditional TSR algorithms, obtain deeper abstract features, and improve the robustness and generalization of the algorithm (Wu, Qin, Pan, & Yuan, 2018). How to apply deep learning models to the field, develop a real-time and accurate sign detection algorithm for automatic diving is the initiative of many researchers.

This TSR project can not only avoid traffic accidents and protect drivers, but also maintains many traffic signs on modern roads efficiently and accurately, which can reduce unnecessary manpower and material resources. In addition, it also provides technical support for the unmanned and auxiliary driving. Therefore, the research work based on deep learning has far-reaching significance and critical value.

1.2 Research Questions

Traffic sign detection and recognition are important parts of unmanned driving and auxiliary driving, as well as essential elements of achieving safe driving. The TSR algorithms based on deep learning are mainly grouped into two categories. One is based on the region proposal algorithm, also known as the two-stage method. The main representative models are R-CNN (Regions with CNN features), Fast R-CNN (Fast Regions with CNN features) and Faster R-CNN (Faster Regions with CNN features). The other is based on the regression function, an end-to-end detection model, also known as the one-stage method. You Only Look Once (YOLO) and Single Shot MultiBox Detector (SSD) belong to the one-stage category.

At present, most of the research work in TSR based on deep learning algorithms is to compare one from region-proposal-based and another one from the regression-based. Still, there are few comparisons between the two algorithms, both from the regression-based. Therefore, the research questions in this report are listed as follows.

(a) How is the performance of YOLOv5 based on the dataset of New Zealand traffic signs? (b) For YOLOv5 and SSD neural network, which algorithm has a better performance for NZ-TSR?

Therefore, this report mainly studies object detection and recognition based on the dataset of New Zealand traffic signs by comparing YOLOv5 and SSD, analyses and evaluates these two models' performance.

1.3 Contributions

This report mainly investigate how to achieve an accurate and real-time TSR model based on deep learning. Our contributions come from four aspects. Firstly, we collect and pre-process sample images to compile a new and utility dataset for New Zealand traffic signs, which contains 2,182 samples in eight classes. Secondly, for the latest version, YOLOv5, we implement the experiment and evaluate TSR performance based on our own dataset. The key metrics and parameters we captured could provide a few essential references and help for subsequent research and explorations.

Thirdly, we conduct a detailed comparison of recognition performance between YOLOv5 and SSD. We also analyze and justify the advantages and disadvantages of these two models. Finally, in our experiment, we accomplish the largest network, namely YOLOv5x, training with our dataset. Hence, the well-trained model is used to recognize the NZ-TSR in digital images or videos.

1.4 Objectives of This Report

In general, the project needs to collect and generate a customized dataset representing the New Zealand traffic signs, and aims to figure out a suitable recognition model based on deep learning. More specifically, the objectives of this project are subdivided into three goals.

The first one is to investigate how the latest YOLO series algorithm performs on the NZ-TSR. Through implementing a proper experiment on Google Colab, we are required to capture various performance and obtain a well-trained model. The next is to compare and analyze two algorithms (YOLOv5 and SSD) to determine which one is suitable for NZ-TSR. Thereby, it could work out the answers for two research

questions we raised. Finally, this report will clarify and summarize experimental results so as to provide references for other research projects and discover the direction of our future work.

1.5 Structure of This Report

The structure of this report is described as follows:

- In Chapter 2, we will conduct a comprehensive literature review to introduce related work on TSR which also includes research discussion about various functional layers in CNN, a few classic CNN networks, YOLO and SSD algorithms.
- In Chapter 3, we will expound on details of the methodology adopted in this report from three aspects, the principle of experimental models, preparation and implementation of the experiment, and evaluation metrics.
- In Chapter 4, we will intensively demonstrate our experimental results, and provide the corresponding illustration as well.
- In Chapter 5, we will proceed with a comparison of TSR performance obtained by using YOLOv5 and SSD algorithms, followed by relevant analysis and discussion. Meanwhile, we will point out the limitations of this project.
- In Chapter 6, we will review the entire project, summarize our outcomes, and clarify the future work.

Chapter 2
Literature Review

With a comprehensive literature review, in this chapter, we illustrate two main categories of methods in traffic sign recognition. The follow is to elaborate on four essential layers of CNN in theory: the convolutional layer, the pooling layer, the fully connected layer, and activation functions. Next, AlexNet, VGGNet and ResNet are expounded in detail as typical CNN networks. The recent findings of YOLO and SSD are summarized concisely.

2.1 Introduction

In general, object detection is to label the presence of an interested object in the image, and its location (Shao et al., 2019). This is a simple problem of visual saliency attention mechanism for humans. However, in the field of computer vision, the way computers store images is pure data without abstract information. There is a lack of shallow image information such as texture, shape and color, and also high-level semantic information, which like positional relationships (Zhou, Zhan, & Fu, 2021). At present, there are mainly two strategies for object detection research. One is to give computers some of the object features in advance, combine with human's subjective setting, and classify the features designed by human on the image through machine learning method, so as to obtain the position and size of the object. The other is the deep learning method, in which computers extract the feature maps of visual object through a large amount of annotated data, and finally determine the location and quantity of the object through semantic information (Sun, Hongji, Nie, & He, 2019).

2.2 Traffic Sign Recognition (TSR)

TSR has always been a redhot topic in recent years. On the aspect of the purpose, traffic sign detection is to classify traffic signs and non-traffic sign areas in complex scene images, TSR is to extract the specific meaning represented by the traffic sign pattern (Sun et al., 2019). The existing TSR methods are basically grouped into two categories: One is based on traditional methods, the other is based on deep learning.

2.2.1 Based on Traditional Methods

TSR basically is grouped into the methods based on color and shape of traditional machine learning methods. The main steps of TSR methods based on the color and shape of the image are to extract the color and shape information contained in the candidate areas, capture and segment the traffic signs in the image, and then correctly classify them (Wang, 2018). Although TSR requires color and shape information

which is employed to improve the recognition accuracy. The problems of the color changes due to the illumination changes or fadin, as well as the deformation and the occlusion of traffic signs, are still unresolved yet. Therefore, this kind of TSR methods have limitations.

Traditional machine learning methods usually select specified visual features firstly, and use the appropriate features to distinguish the categories of traffic signs by using the corresponding classifiers. The specific features include Haar-like features, HOG features, SIFT features and so on (Ellahyani, Ansari, Lahmyed, & Trémeau). The neural networks, Bayesian classifier, random forest and Support Vector Machine (SVM) are employed as classifiers. As a result, the SVM is the most popular method for TSR.

Houben (2011) were the first one to use this method to extract HOG features of the interested region in images, and then sent the extracted HOG features into the pre-trained SVM classifier and obtain the recognition results. However, the performance of traditional machine learning methods depends on the selection of the specified features. Furthermore, for different classifiers, corresponding feature description information is required. Hence, traditional machine learning methods have limitations, and their real-time performance is relatively low.

2.2.2 Deep Learning Methods

Traditional TSR methods require specified pre-extracted features, which result in key feature missing, and also need high illumination and quality of digital images. Deep learning utilizes a multilayer neural network to automatically extract and learn the features of visual objects, which has good applicability for image tasks (Jing Zhang, Hui, Lu, & Zhu, 2018). Convolutional Neural Network (CNN) models are one of the most popular deep learning approaches. As opposed to traditional machine learning methods, deep learning methods automatically generate feature maps through convolutional and pooling operations based on a large number of image datasets, which doesn't need the specified features, so that it is able to achieve high recognition

accuracy.

TSR algorithms based on region proposals, also known as two-stage detection algorithm, the core idea is the selective search (Jin et al., 2020). Its advantages are good detection and positioning effects, but the cost is a large amount of computation, timing-consuming, and high-standard hardware equipment. The representative modes include R-CNN, Fast R-CNN and Faster R-CNN. Meanwhile, traffic sign recognition algorithm based on regression, also known as single-stage detection algorithm, the idea is regression method (Bangquan & Xiong, 2019). This kind of TSR algorithms completely eliminate the idea of Region Proposal Network (RPN), and directly performs regression and classification in a network. This greatly reduces the repetitive calculations in the detection process, so the detection speed has been significantly improved compared to the previous algorithms.

2.3 Neural Network Overview

Artificial neural network (ANN) was from the inspiration of the nervous system of creatures in nature. ANN plays an excellent role in object classification, speech recognition, artificial intelligence and other fields. Generally speaking, input layer, hidden layer and output layer constitute a complete neural network architecture (Yanhua, Shengyan, & Yan, 2011). Each layer is composed of multiple neurons, the neurons of the two adjacent layers are connected in the way of full connection. The ANN architecture is present in Figure 2.1.

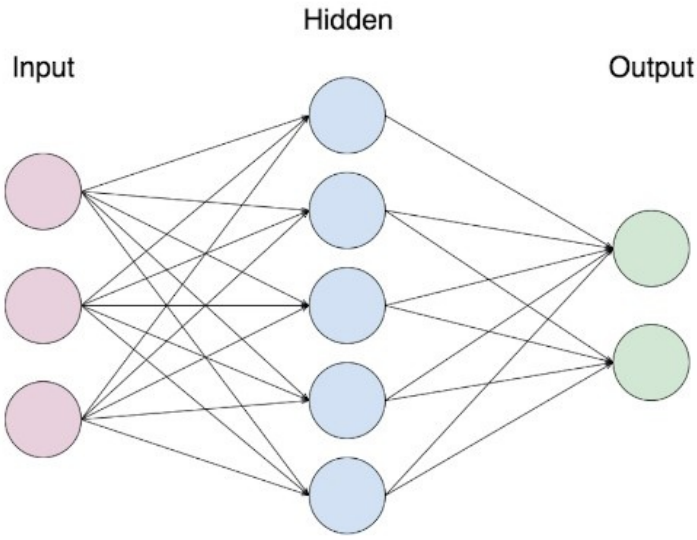


Figure 2.1 The ANN architecture

ANN was modelled based on the complex relationship between its input and output, which is a kind of operations (Zoumpourlis, Doumanoglou, Vretos, & Daras, 2017). The internal relationship of ANN is quite complicated. The mathematical model of its single neuron is shown in Figure 2.2.

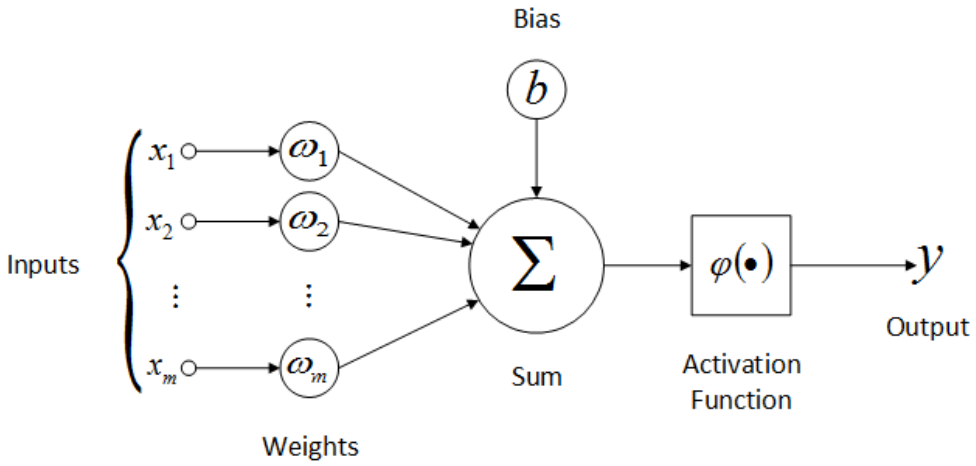


Figure 2.2 The mathematical model of the single neuron

The mathematical description is shown in Eq.(2.1),

$$y = \varphi\left(\sum_{i=1}^{n} x_i w_i + b\right) \tag{2.1}$$

Firstly, the neuron multiplies the input signal with its corresponding weight and sum them up, adds a bias to the result of the summation, finally output the result passing through the activation function.

2.4 Convolutional Neural Network (CNN)

CNN mainly operates on two-dimensional images, and spontaneously extract features from the data by constructing multiple hidden layers. Therefore, CNN is made of the input layer and the output layer as well as multiple hidden layers. In the early convolutional neural network model, the hidden layers mainly contain convolutional layer, pooling layer, fully connected layer, activation function and other functional modules. Then, batch normalization is added to the basic structure (Zhao & Zheng, 2020). Figure 2.3 displays the basic structure of the CNN net.

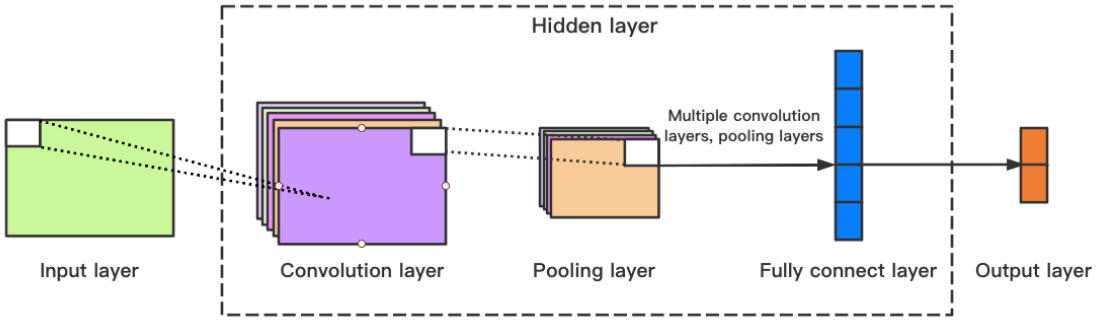


Figure 2.3 Basic structure of convolutional neural network

- Input Layer: Usually, the raw data or preprocessed data is entered into CNN. For image recognition, the input includes the two-dimensional pixel points and RGB channels of the picture.
- Hidden Layer: It makes up of multiple functional layers, convolutional layers and pooling layers extract visual features of the inputs. Then, the loss function calculates the loss error between the predicted result and the actual truth, and return the error through the feedforward mechanism. After several rounds of iterations, the error is reduced to the acceptable level so that the model is well fitted.

- Output Layer: It usually outputs the center coordinates, the length and width of the corresponding bounding box, and the classification information of the detected object.

2.4.1 Convolutional Layer

The convolutional layer is the most critical layer in CNN, the object feature map is obtained by traversing the input image through the convolution kernel. A convolution layer contains four components: Convolution kernel, receptive field, zero-padding and step size. Therefore, in the convolutional layer, the convolution kernel is used to perform a sliding window operation based on the input image to form a feature map. Corresponding multiple convolution kernels will generate multiple feature maps, and eventually they form a feature map together (Jianming Zhang, Huang, Wu, & Liu, 2017). The convolution calculation is shown as eq.(2.2).

$$a_{i,j} = f\left(\sum_{m=0}^{2}\sum_{n=0}^{2} w_{m.n}x_{i+m,j+n} + w_b\right) \tag{2.2}$$

In Eq.(2.2), $x_{i,j}$ expresses the component in the row i and the column j of the image. Each weight of the filter is numbered, then $w_{m,n}$ represents the weight of the m-th row and the n-th column, and w_b stands for the bias parameter of the filter. Each component of the feature maps is also numbered as well, so $a_{i,j}$ is employed to indicate the corresponding element in the i-th row and j-th column of the feature map, f is the chosen activation function.

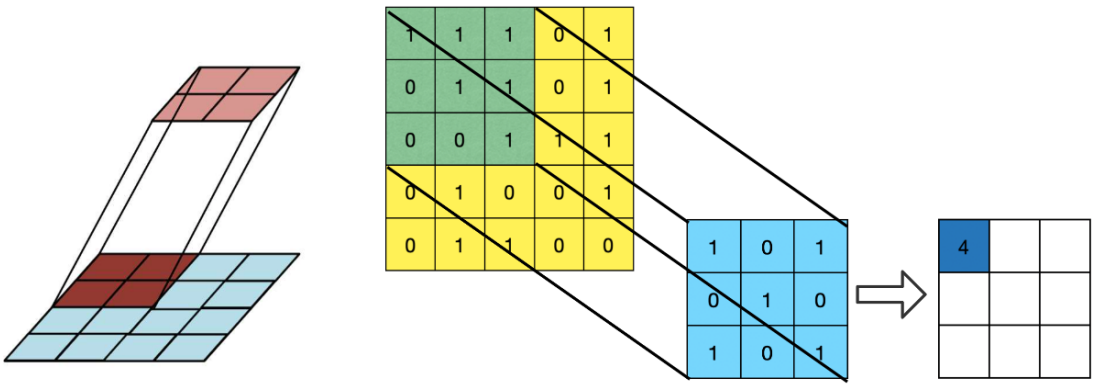


Figure 2.4 The convolution operation

In Figure 2.4, we take use of a two-dimensional convolution kernel in the 3×3 pixel size and perform a convolution operation on a rectangle image with a side length of 6.

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

$$= relu(w_{0,0}x_{0,0}+w_{0,1}x_{0,1}+w_{0,20}x_{0,2}+w_{1,0}x_{1,0}+w_{1,1}x_{1,1}+w_{1,2}x_{1,2}+w_{2,0}x_{2,0}+w_{2,1}x_{2,1}+w_{2,2}x_{2,2}+w_b) \quad (2.3)$$

The features obtained after one convolution operation is low-level features. After the low-level features are subjected to multiple convolution operations, intermediate-level features are obtained. Then intermediate-level features are convolved for many times to get high-level features. In general, we are using high-level features for the object classification and prediction. A single convolutional layer often is used to learn only the local features of the input image, but for the purpose of extracting more global features, more convolutional layers need to be added (Shustanov & Yakimov, 2017). In TSR, the first work is to identify which class the traffic sign belongs to, so it is essential to extract visual features to improve the classification accuracy.

2.4.2 Pooling Operations

The pooling operations are within a hidden layer added between successive convolutional layers, and is also called a downsampling operations. The layer is endowed with two main functions. One is to make the feature map smaller, so as to reduce the amount of parameters and the complexity of training (Chung, Kim, Kang, & Lim, 2020). The second is to effectively extract the main feature information in the feature map, thereby reduce the noise interference of other objects in the image (Hussain, Abualkibash, & Tout, 2018). The pooling operations greatly reduce the

weight parameters so that it can avoid overfitting. Usually, one or more convolutional layers are followed by a pooling layer.

The max pooling is suitable for extracting texture features, and is generally used in shallow networks to reduce useless information. On the contrary, average pooing is suitable for retaining overall features, and is generally utilized in deep networks to retain contextual information (Zhang, Wang, Lu, Wang, & Sangaiah, 2020). Max and average pooling operations are shown in Figure 2.5.

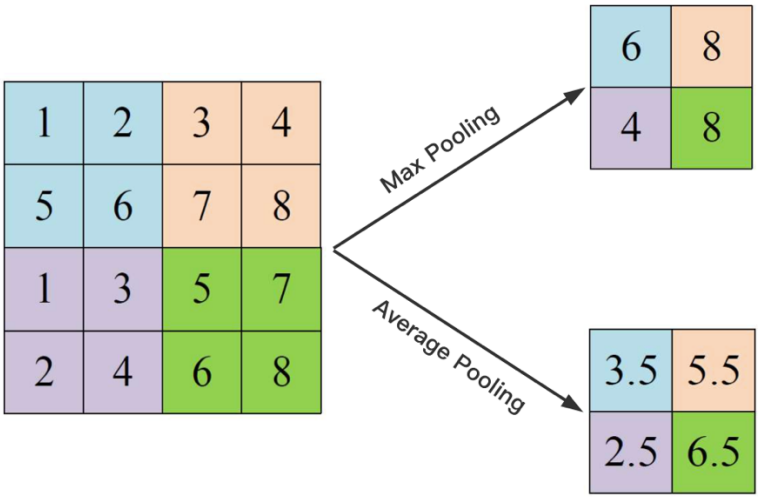


Figure 2.5 Max pooling and average pooling

2.4.3 Fully Connected Layer

The fully connected layer is usually the last one of the hidden layers in the convolutional neural network. In the fully connected layer, the neurons in the same layer are not connected, but they are connected to the neurons in the previous layers one by one. This layer aims at converting the output feature maps obtained by passing through previous convolution and pooling layers to be a one-dimensional matrix (Molchanov, Vishnyakov, Vizilter, Vishnyakova, & Knyaz, 2017). This one-dimensional matrix consisting of the feature vectors extracted after convolution and pooling operations, which facilitates subsequent regression classification operations.

The linear transformation is carried out in the fully connected layer according to the feature vector and the trained weights. Similar to the convolution layer and the

pooling layer, the fully connected layers are also combined (Luo, Yang, Tong, Wu, & Fan, 2017). However, a lot of parameters exist in the fully connected layer. The combination of multiple fully connected layers will increase the training time.

2.4.4 Activation Function

The activation function is a functional relationship between the previous layer's output and the following layer's input. With the activation function, a nonlinear mechanism is attached to the convolutional neural network (Wang, Li, Song, & Rong, 2020). Then, the entire network can propagate the activation information to the next convolutional layer. The network also has the ability to fit nonlinear functions, so that it solves the nonlinear classification problems.

The activation functions basically have into two types: A nonlinear function with the exponential shape and a piecewise linear function. The nonlinear functions with the exponential shape include sigmoid and tanh activation functions. Piecewise linear functions include ReLU and Leaky-ReLU activation functions (Croce, Andriushchenko, & Hein, 2019).

The output interval of the sigmoid function is (0, 1), so that it can be used to normalize the input or express the probabilities. But from another point of view, the sigmoid function also has its drawbacks. One of the problems is its saturation, which can easily lead to under-fitting (C. Zhang, Yue, Wang, Li, & Ding, 2020). Another problem is that the output isn't zero-centered, which causes a gradient vanishing problem. The sigmoid function curve is shown in Figure 2.6 (a). As can be seen from the curve, the function tends to be saturated as the input goes to plus or minus infinity. Equation 2-4 below represents its formula.

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{2.4}$$

Tanh function is a kind of hyperbolic functions, and is also a common activation function, which output interval is (-1, 1). As shown in Figure 2.6 (b), its curve shape is very similar to the shape of sigmoid function. The main difference is that the output

value of Tanh function is between -1 and 1, which is zero-centered. Thus it can solve the non zero-centered problem in sigmoid function. But the gradient vanishing problem still exists (J. Yang & Yang, 2018). The function formula is displayed in 2-5.

$$Tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.5}$$

ReLU (Rectified Linear Unit) function is a piecewise function, which formula indicates in Equation 2-6. The essence of this function is to keep any value greater than 0 unchanged, and return all value less than or equal to 0 to be 0. Therefore, ReLU function can ensure that its gradient is not attenuated when the input value is greater than 0, which effectively alleviates the gradient vanishing problem that happened to previous activation functions. Furthermore, the usage of ReLU function makes training no longer rely on the unsupervised pre-training method, which is very important for training deeper models. It can also proffer a reference on the later design of the activation function of the network model, effectively reducing the complexity of training (Arcos-García, Alvarez-Garcia, & Soria-Morillo, 2018). The function curve is shown in Figure 2.6(c).

$$ReLU(X) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{2.6}$$

In ReLU function, when the input value is negative, the output is always 0, which will cause the neuron to fail to update the data and the neuron stops learning. Leaky ReLU function is a variant of ReLU function. For the negative input, this function has a minor slope, which displays in Figure 2.6 (d). That change can well solve the problem that neurons stop learning when the input enters the negative interval (X. Wang, Qin, Wang, Xiang, & Chen, 2019). And its formula shows as follows.

$$LeakyReLU(x) = \begin{cases} x, & x \geq 0 \\ ax, & x < 0 \end{cases} \quad \text{and a } \in (0,1) \tag{2.7}$$

Four common activation functions' curve display in Figure 2.6.

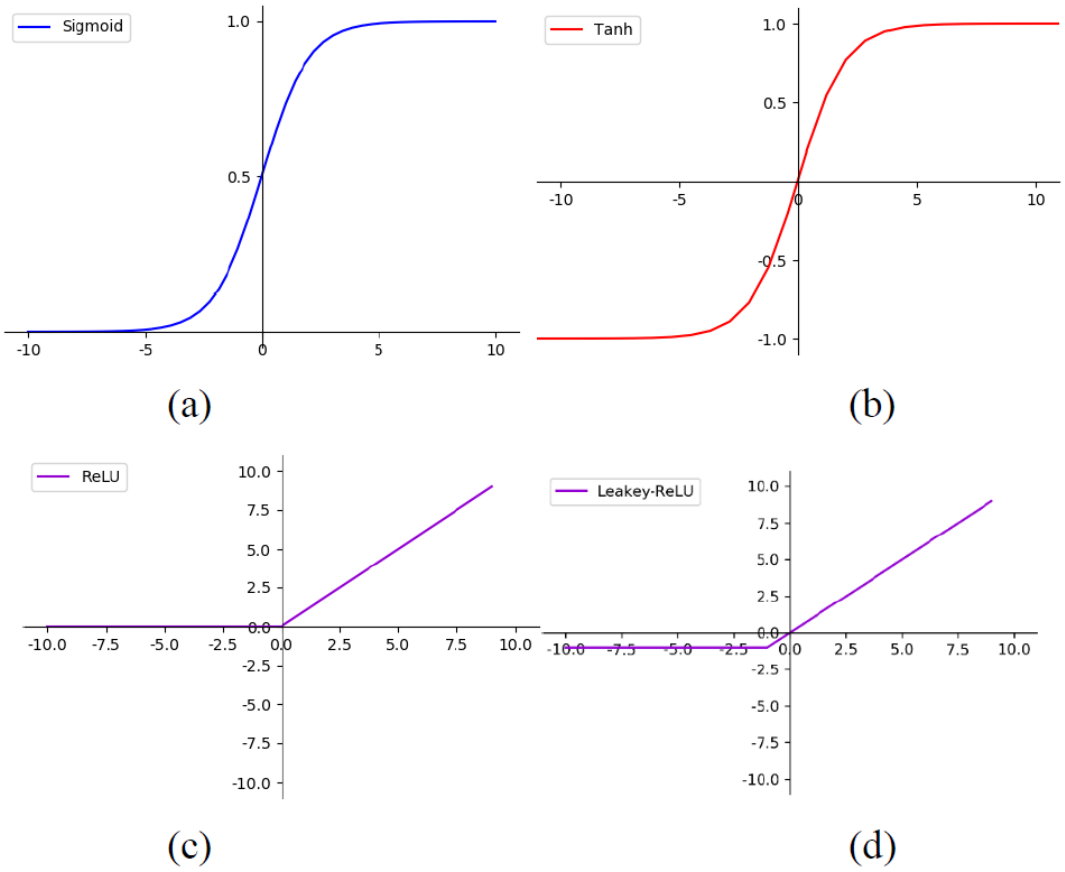


Figure 2.6 Diagram of four common activation functions

2.5 Classic Convolutional Neural Networks

2.5.1 AlexNet

AlexNet is an efficient and typical convolutional neural network structure, which is suitable for multi-GPU calculations. Meanwhile, AlexNet is also a neural network based on LeNet-5 that deepens the network structure, so as to learn more complex and high-dimensional image features. This neural network is composed of 8 layers, including 5 convolution layers, 3 fully connected layers and the softmax layer with 1000D at the end. And its detailed structure gets displayed in Figure 2.7 as follows.

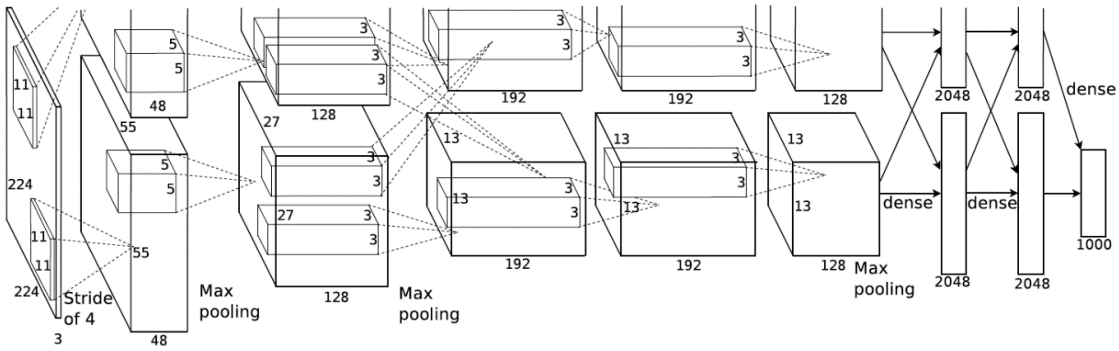


Figure 2.7 The structure of AlexNet

AlexNet has made innovations in many aspects and adopted many new technologies. First of all, on the aspect of the activation function, it uses ReLU function that is a non-linear saturation function, to replace the ordinary sigmoid function (S.-H. Wang et al., 2019). In this way, the network can iterate to the specified training error range more quickly. Secondly, the network uses two GPUs for calculations, and each GPU has its own specified calculation operation. Moreover, two GPUs can only communicate with each other at the prescribed network layer. Multiple GPUs usage does greatly improves the network's calculation efficiency (Krizhevsky, Sutskever, & Hinton, 2017). Besides that, the Dropout function starts being used in training, so a part of the neurons are randomly ignored that it could avoid overfitting the model (J. Li et al., 2019). Finally, previous convolutional neural networks generally use average pooling, while AlexNet takes the lead in adopting maximum pooling so that it effectively solves the fuzzy problem caused by average pooling.

2.5.2 VGGNet

VGGNet is one of the most famous classic convolutional neural networks as yet. Since VGGNet makes improvements on the basis of AlexNet, replacing only one 7×7 convolution kernel in a convolution layer in AlexNet network structure with 2 to 4 small 3×3 convolution kernels. To be brief, VGGNet is to change one single large convolution kernel to be multiple small convolution kernels in the same convolution layer (Jun et al., 2018). Using a multi-layer small convolution kernel to replace a single

large convolution kernel not only reduces the number of parameters, but also has more non-linear convolution operations. As a consequence, multi-layer nonlinearity can increase the depth of the network to ensure the learning of more complex models, and improve the fitting expression ability.

The two most common structures for VGGNet are VGG16 and VGG19. As the name suggests, there are 16 layers in a total of convolution layers and fully connected layers in VGG16, while there are 19 layers in VGG19. It shows that there is no essential difference between the two structures. Only the latter has a deeper network depth than the former. However, compared with AlexNet, VGG uses multiple sets of small convolution layers, so its performance of convolution is more excellent (Bi, Yu, Gao, Zhou, & Yao, 2020). But meanwhile, VGGNet also has a few shortcomings. For example, this network structure consumes more computing resources, and more fully connected layers result in more memory usage (Y. Zhou, Chang, Lu, Lu, & Zhou, 2020).

2.5.3 ResNet

The proposal of VGGNet effectively proves that as the network depth increases, the network performance becomes better. Nevertheless, with the development of network model depth characteristics, researchers found that simply adding layers to increase the depth of the network will not only not improve the network's performance, but its accuracy will also drop rapidly after reaching saturation (X. Li & Cui, 2016). In the meantime, adding more useless layers to the network model with the appropriate depth will lead to higher training error. K. He, Zhang, Ren, and Sun (2016) proposed the ResNet structure. ResNet makes use of the residual structure to alleviate the problem of training degradation well. The residual structure refers to establishing an additional special connection structure that can skip one or more layers between layers to achieve a connection, namely shortcut connection. The diagram of a residual unit structure is in Figure 2.8 as follows.

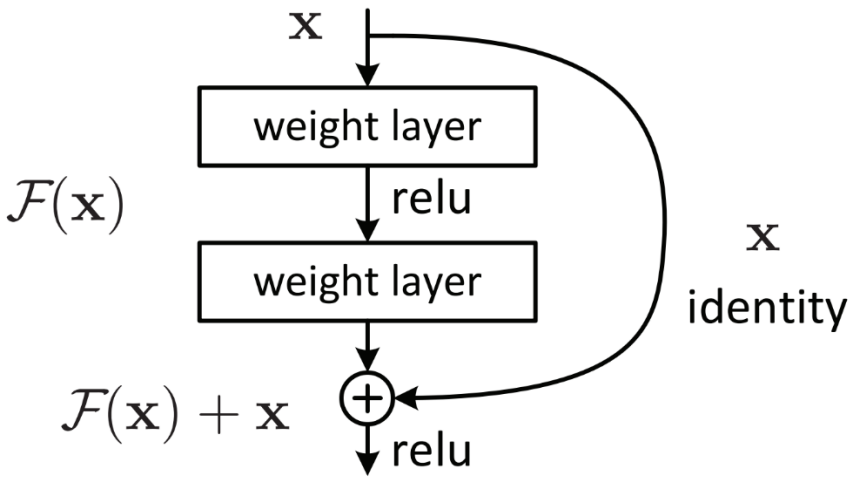


Figure 2.8 The structure of residual unit

According to the above figure showing, the output of the residual unit not only includes the output of multiple convolution layers but also includes a non-linear combination with the input element x, which is activated by ReLU function. Then ResNet is composed of multiple such residual units. As we all know, the deeper network is more complicated to train due to the gradation vanishing and gradient explosion problems. Learning shortcut connection can make it get activated from one layer, and then quickly feedback to another layer or even a deeper layer. Therefore, ResNet is more suitable for training deeper network models.

2.6 Object Detection Models

2.6.1 YOLO

Object detection consists of two tasks, which are classification and localization. Before the appearance of You Only Look Once (YOLO), these two tasks are separated in the object detection methods. So with that in mind, YOLO is a brand new object detection method proposed by Redmon, Divvala, Girshick, and Farhadi (2016). In the YOLO model, the object detection task is simply converted into a regression problem to deal with. Furthermore, YOLO is an end-to-end object detection structure that can obtain the coordinates of the predicated bounding boxes, the confidence of the target existence, and the probability of the category the target belongs to simultaneously

through one image input.

Eventually, YOLOv5 was published in June of 2020. There is few research on YOLOv5's performance in traffic sign recognition. Nevertheless, Kuznetsova, Maleva, and Soloviev (2020) proceeded an experiment of detecting apples by YOLOv5 to compare with the performance of YOLOv3. Their experimental results indicated that YOLOv5 outperformed the previous model indeed. YOLOv5 obtained 4.3% increments in the detection accuracy on the dataset of 5142 apples in 878 images. Moreover, a similar experiment was carried out for the apple picking robot by Yan, Fan, Lei, Liu, and Yang (2021). And they concluded the comparable outcomes with an improved YOLOv5s model, which were 14.95% and 4.74% rise by comparisons to YOLOv3 and YOLOv4, respectively.

2.6.2 SSD

The SSD Neural Network has already been well known (Liu et al., 2016). Meanwhile, the SSD model is already being improved and used to detect the target in various fields. Recently, the experiments are implemented based on the CTSD dataset with the improved SSD model, the results reached 94.4% for the precision and 92.6% for the recall (Huo, Zhang, Li, 2020). Besides that, a comparison of traffic sign recognition between SSD and YOLOv2 was done by Garg, Chowdhury, and More (2019). Their experiment adopted the dataset chosen from the GTSRB dataset. After an overall comparison, they clarified that SSD was 21% less accurate than YOLOv2, and the latter was 16% faster than the SSD model.

Chapter 3

Methods

The main content of this chapter is to elaborate on two models, including YOLO and SSD, theoretically used in this experiment. Afterwards, it describes the collection and pre-processing of the dataset for each model. The follows are to address the implementation our experiments and key parameters in detail. The evaluation methods of this project are introduced finally.

3.1 YOLO Series Algorithm

The YOLO series model has been updated to YOLOv5 since it was proposed. The accuracy of object detection continues to be improved, and regression is always adopted as its core idea. In this report, we also take the latest version of YOLOv5 as one of the NZ-TSR models in the experiment. Hence, the following is a detailed introduction to each version of the YOLO model.

3.1.1 YOLOv1

Consistent with other convolutional neural network detection algorithms, YOLOv1 also uses multi-layer convolution and pooling operations to extract features, and then outputs final feature maps through a fully connected layer. However, YOLOv1 subdivides the input image into many small grids, which is different from other object detection models.

First of all, YOLOv1 resizes the original input image to 448×448, and then keeps dividing it into S×S cells, usually with a size of 7×7 (Redmon et al., 2016). This step operation facilitates the division of the object's center position, and the division result should look like in Figure 3.1 below. Each cell can detect objects individually. If the center of an object falls in a cell, then that specific cell is responsible for predicting the object.

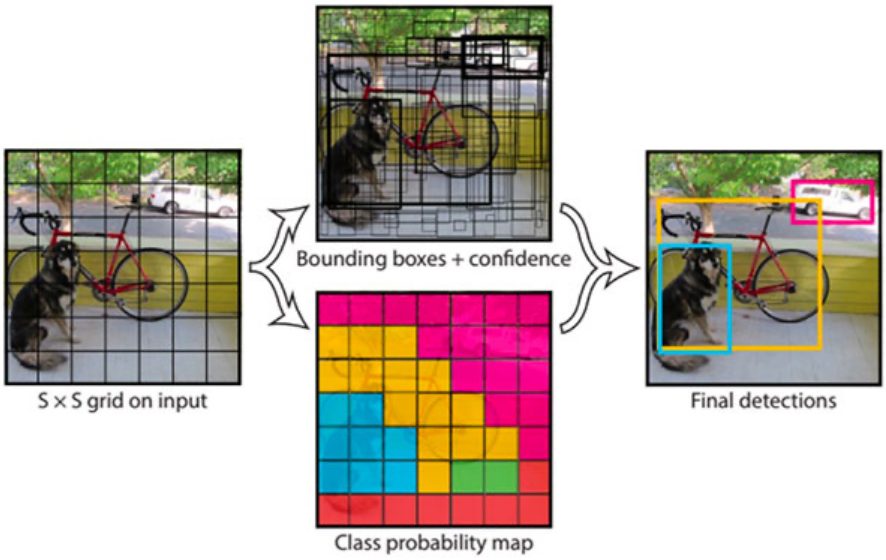


Figure 3.1 Diagram of YOLO's brief detection workflow

As shown in Figure 3.1, each cell has to predict the number of B bounding boxes' values. And the bounding box value contains coordinates (x, y), width w, height h, and a confidence score predicted for each bounding box. Hence, each cell would predict the number of B 5-dimensional matrixes. Among them, (x, y) represent the offsets of the bounding box's center position relative to the upper left corner of the current cell, and the value range is (0, 1). Where w and h are the ratios of the bounding box's width and height to the entire image, and the value range is also (0, 1). There are two parts of the calculation for the confidence score: whether an object is in the cell and the IoU (Intersection over Union) value of the prediction box and the real box. The confidence calculation formula displays as follows.

$$\boldsymbol{confidence} = \boldsymbol{Pr}(\boldsymbol{Object}) \times \boldsymbol{IoU}_{\boldsymbol{pred}}^{\boldsymbol{truth}} \quad (3\text{-}1)$$

If the cell contains an object, then *Pr(Object)* is 1, then the confidence score equals the IoU value. On the contrary, if no object exists in the cell, then the confidence score equals 0 (Jianming Zhang, Huang, Jin, & Li, 2017). Besides that, when classifying, YOLOv1 needs to predict the probability for the number of C categories in all cells. Here $Pr(Class_i|Object)$ represents the conditional probability under the confidence of the bounding box. Then the above confidence calculation

formula is converted as follows.

$$\boldsymbol{confidence} = \boldsymbol{Pr}(\boldsymbol{Class_i}|\boldsymbol{Object}) \times \boldsymbol{Pr}(\boldsymbol{Object}) \times \boldsymbol{IoU_{pred}^{truth}}$$
$$= \boldsymbol{Pr}(\boldsymbol{Class_i}) \times \boldsymbol{IoU_{pred}^{truth}} \quad (3\text{-}2)$$

Eventually, through setting a threshold, the prediction with a low confidence score would be filtered out. Then use Non-Maximum Suppression (NMS) to remove redundant ones to obtain the final prediction box. YOLOv1 network structure adopts the typical form of CNN, starting with convolution layers to extract features, and then fully connected layers to predict results (G. Li, Song, & Fu, 2018). Its interior borrows GoogLeNet, as shown in Figure 3.2 below.

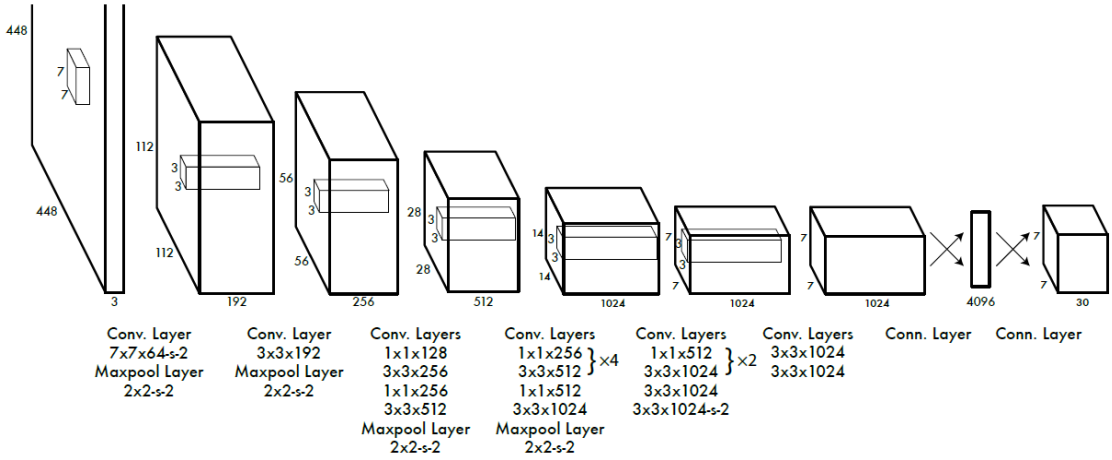


Figure 3.2 Diagram of YOLOv1 network structure

The total predicted value output by all cells is calculated like this,

$$\boldsymbol{S} \times \boldsymbol{S} \times (\boldsymbol{B} \times (\boldsymbol{4} + \boldsymbol{1}) + \boldsymbol{C}) = \boldsymbol{7} \times \boldsymbol{7} (\boldsymbol{2} \times \boldsymbol{5} + \boldsymbol{20}) \quad (3\text{-}3)$$

In Figure 3.2, the output dimension of YOLOv1 is only 30, which means that each cell can only predict two bounding boxes and only recognize 20 categories of objects. As a result, YOLOv1 has a great defect for multi-size object detection, and the limitation of 20 categories of objects is not conducive to multi-object detection.

3.1.2 YOLOv2

In order to solve the large positioning error and low recall rate problems in YOLOv1, YOLOv2 was proposed in 2017. YOLOv2 does do improvements in many aspects.

(1). Batch Normalization

YOLOv2 adds a bath normalization layer after each convolution layer to replace the dropout layer to prevent over-fitting of the model (S. Yang, Bo, Zhang, & Wang, 2020). The newly added batch normalization layer makes the model accelerate the convergence efficiency and increase the mAP by 2.4%.

(2). High-resolution classifier

Before YOLOv2, many object detectors used images smaller than 256×256. Such low-resolution detection can no longer meet the needs of high-resolution object detection. Hence, YOLOv2 starts training the model with 448×448 resolution. In the end, the mAP of high-resolution detection gets increased by 4%.

(3). Convolution with anchor boxes

In order to improve the recall value, YOLOv2 removes the fully connected layer and uses anchor boxes to predict bounding boxes. The user weighted the detection accuracy and speed, and finally determined the number of anchor boxes was 5. Then that makes the number of prediction boxes increased from 98 (7×7×2) to 845 (13×13×5) so that the accuracy of target positioning gets improved.

(4). Darknet-19

YOLOv2 proposes a classification network called Darknet-19 (C.-C. Wang, Samani, & Yang, 2019). It has 19 convolution layers and 5 max-pooling layers, as shown in Figure 3.3.

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Figure 3.3 Diagram of Darknet-19 structure

In Darknet-19, it adopts many 3×3 convolution layers, and the number of convolution kernels is doubled after the pooling operation. Then it uses 1×1 convolution layers to achieve dimensionality reduction. Thus, the model's detection accuracy is improved. Although YOLOv2 has improved the problems of inaccurate positioning and low recall in YOLOv1, there is still room for improvement in small object detection ability. The detection accuracy still needs to be enhanced.

3.1.3 YOLOv3

The problem with YOLOv2 is that the detection accuracy of small objects is not high. For example, in a feature map obtained after feature extraction of an image, the information containing small objects are lost. YOLOv3 draws on the idea of Feature Pyramid Network (FPN) and introduces the concept of multi-scale prediction. That is, the number of detection layers increases from 1 to 3 (Lawal, 2021). The resolution is 1/32, 1/16, and 1/8 of the original size. As shown in Figure 3.4, then the 3 layers correspond to feature maps at three different scales: 13×13, 26×26 and 52×52 (W.

He, Huang, Wei, Li, & Guo, 2019). The multi-scale detection strategy in YOLOv3 enables the detection layer to integrate contextual feature information, which is convenient for the model to predict objects at various scales.

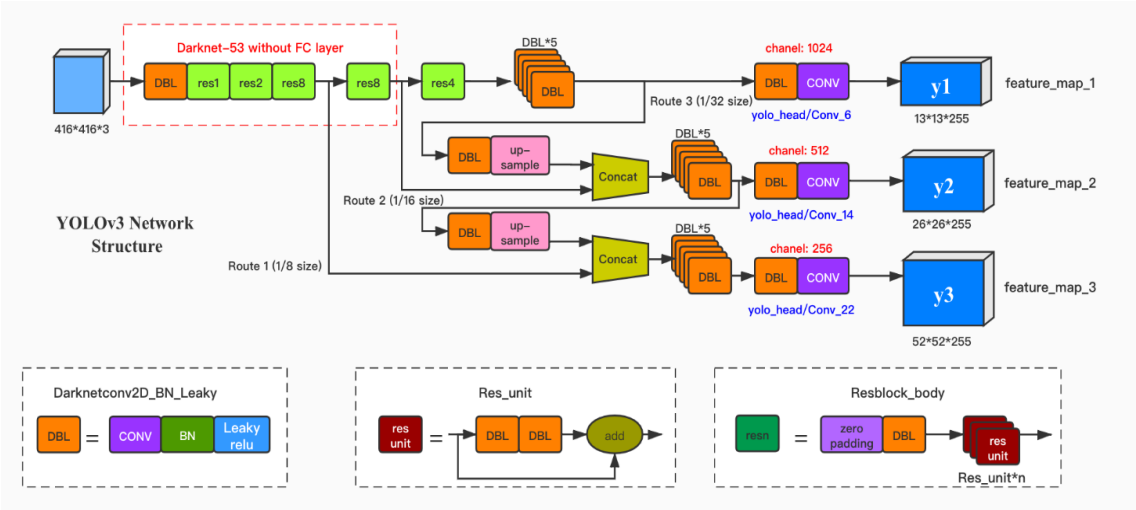


Figure 3.4 Diagram of YOLOv3 network structure

The backbone network of YOLOv3 uses Darknet-53, which is deeper and more robust, as the feature extraction network, and contains 53 convolutional layers (M. Zhang, Zhao, Li, Wang, & Zhang, 2019). As displayed in Figure 3.5, Darknet-53 draws on the practice of the residual network (ResNet) and sets up residual components between convolution layers. Each residual component is composed of two convolutional layers and a fast connection (Fang et al., 2021). It ensures that the network still maintains the characteristics of the training object when it is deep. And it also makes the loss function can still converge and avoids the gradient vanishing problem.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 3.5 Diagram of Darknet-53 structure

YOLOv3 deepens the network structure to improve accuracy at the expense of network speed. The problem of false detection and omission of overlapping objects still exists in YOLOv3. Therefore, it still needs to seek a more effective detection model.

3.1.4 YOLOv4

The purpose of proposing YOLOv4 is to speed up the training speed and optimize the neural network so that the model can be trained and tested by using conventional GPUs. Similar to YOLOv3, YOLOv4 mainly adopts the CSPDarknet-53 as its backbone network. And it also uses ResNet to extract deep features, and eventually obtain the object's classification and position through a multi-scale prediction layer (Gong, Ergu, Cai, & Ma, 2020). In addition, YOLOv4 chiefly makes the following four improvements.

(1). Input layer improvements

YOLOv4 starts using the Mosaic method to enhance the input data, enriching the dataset by selecting four images for random cropping, random splicing, and random arrangement. The benefit brought by such a splicing operation is that many small objects are added so that the robustness of the network for small object recognition has been improved.

(2). Backbone network improvements

CSPDarknet-53 changes the activation function in the first convolution layer in the original Darknet-53 from ReLU to Mish function (Misra, 2019). Compared to ReLU function, Mish function is smoother, and there is no un-differentiable point in the whole process. The Mish function graph in Figure 3.6 shows that it is similar to ReLU function in the positive interval, but it still has a gradient in the negative interval. And that gradient only disappears when it tends to negative infinity. The model gets better feature transferability and thus has better generalization ability. Mish function equation displays in 3-4.

$$\boldsymbol{Mish} = \boldsymbol{x} \times \boldsymbol{tanh}\left(\boldsymbol{ln}(\mathbf{1} + \boldsymbol{e}^{\boldsymbol{x}})\right) \quad (3\text{-}4)$$

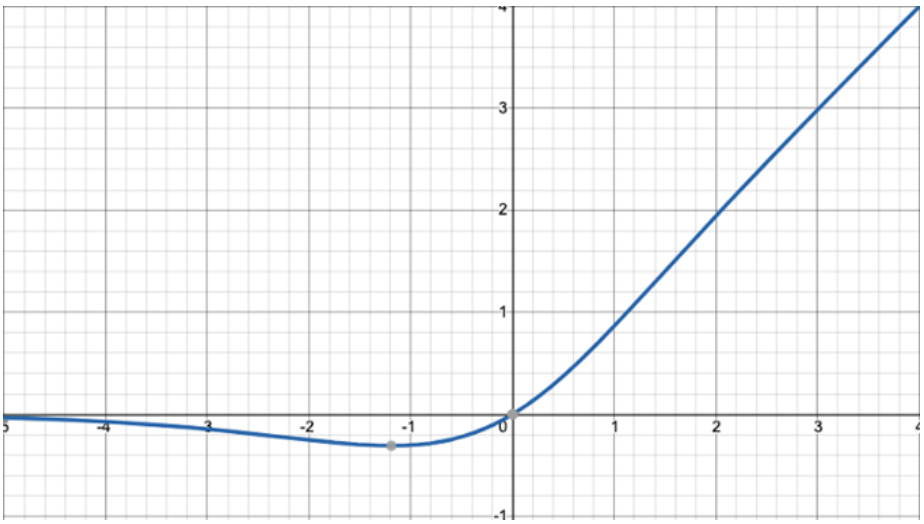


Figure 3.6 Mish function graph

(3). Neck network improvements

YOLOv4 adds Spatial Pyramid Pooling (SPP) module and FPN+PAN (Path Aggregation Network) between the backbone and the final output layer.

(4). Prediction Layer

During training, it adopts CLOU_Loss as the loss function and DIOU_nms to filter prediction boxes. The benefit is that the three factors of overlap area, center point distance, and aspect ratio are considered.

3.1.5 YOLOv5

The structure of YOLOv5 algorithm is very similar to that of YOLOv4. The entire network model can also be divided into four parts: input, backbone, neck and the prediction layer. Figure 3.7 represents the detailed network structure of YOLOv5.

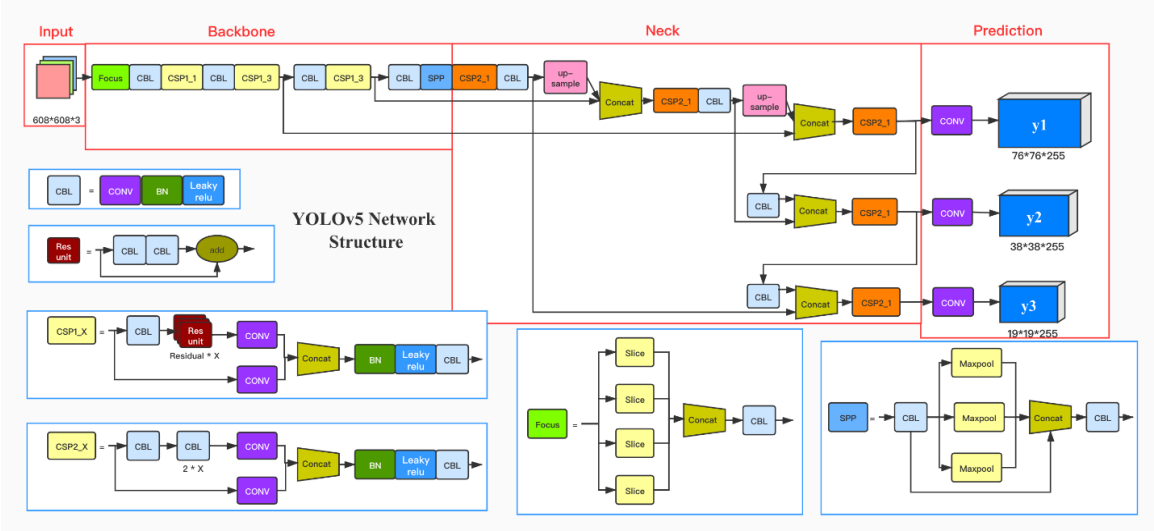


Figure 3.7 Diagram of YOLOv5 network structure

In the input part, YOLOv5 and YOLOv4 both use the Mosaic method to enhance the input data. Mosaic method is very effective for small object detection. Its detailed operations have been introduced in the part of YOLOv4 above. The algorithm needs to change the size of the input image to a fixed size then input it into the detection model for training. As shown in Figure 3.7, the standard size of the image is 608×608×3. In addition, the network trains based on the initial anchor box to obtain the prediction box, compares it with the real annotated box, and updates the network model parameters iteratively through the reverse update of the difference.

The backbone part contains Focus module and CSP module (Shi, Hu, Lei, & Xu, 2021). Focus module does not exist in the previous YOLOv3 and YOLOv4 algorithms. And its key step is to compress the height and width of the input image through the slicing operation. Then the images are spliced to realize the integration of the image width and height dimensional information into the channel dimensions so that the number of input image channels is increased. The structure of Focus module also gets displayed in Figure 3.7. On the aspect of CSP module, two CSP module structures are designed in YOLOv5, which are CSP1_X and CSP2_X (Lian, Yin, Li, Wang, & Zhou, 2021). Among them, CSP1_X module is mainly used in the backbone network, and the CSP2_X is mainly used in the neck network. Both two CSP module structures are included in above Figure 3.7 as well.

The neck part in YOLOv5, similar to that in YOLOv4, also adopts FPN+PAN structure. Feature Pyramid Network (FPN) is from top to bottom and uses up-sampling to transfer and fusion information to obtain predicted feature maps. In contrast, PAN (Path Aggregation Network) is a feature pyramid from bottom to top. Compared with ordinary convolution operations in YOLOv4, YOLOv5 adopts CSP2_X module mentioned above to strengthen the ability of the network feature integration in its neck part.

In the prediction part, unlike YOLOv4, YOLOv5 uses GIoU_Loss as the loss function, which effectively solves the problem when the bounding boxes do not coincide (S. Li et al., 2021). And GIoU calculation equation shows as below.

$$\boldsymbol{GIoU} = \boldsymbol{IoU} - \frac{|\boldsymbol{C} - (\boldsymbol{A} \cup \boldsymbol{B})|}{|\boldsymbol{C}|} \tag{3-5}$$

In the above 3-5 equation, we find the smallest box C for two arbitrary boxes A and B, including A and B. After that, calculate the ratio of the area of C that does not cover A and B to the total area of C, then subtract this ratio from the IoU value of A and B. GIoU can be treated as a distance. So GIoU_Loss can be as follows.

$$\boldsymbol{GIoU\_Loss} = \mathbf{1} - \boldsymbol{GIoU} = \mathbf{1} - \left(\boldsymbol{IoU} - \frac{|\boldsymbol{C} - (\boldsymbol{A} \cup \boldsymbol{B})|}{|\boldsymbol{C}|} \right) \tag{3-6}$$

Besides that, YOLOv5 has four models, which are YOLOv5s, YOLOv5m, YOLOv5l and YOLOv5x, according to the network depth and the feature graph width, and they are shown in Figure 3.8. Among them, YOLOv5s has the smallest network and the fastest training speed, while the lowest Average Precision (AP) value. However, YOLOv5s is suitable for the situation where the detection is based on the big object and the pursuit of speed. On this basis, the other three networks have been continuously deepened and widened, and AP value has also been continuously improved, but the speed consumption has also been increased. In this report, we take the largest network YOLOv5x as a model in the experiment.

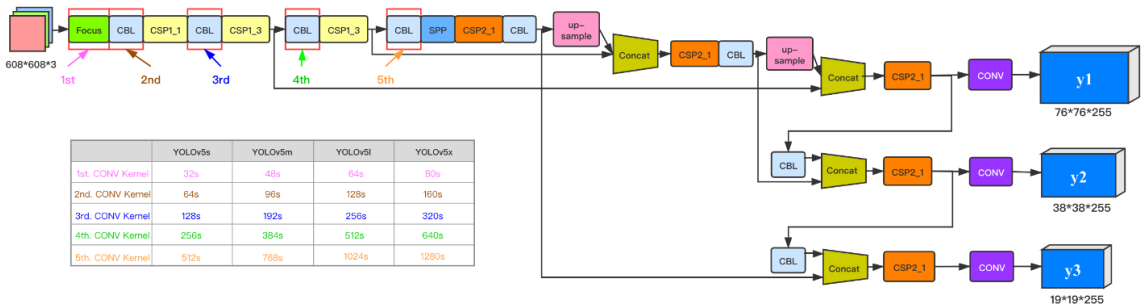


Figure 3.8 YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x's network depth

3.2 SSD Neural Network

SSD is short for Single Shot MultiBox Detector. As the name suggests, Single Shot explains that the SSD model is a one-stage detection method. The user only needs to input an image, and then it passes through the convolutional neural network to directly calculate the classification and location of the object in the image. And MultiBox shows that the SSD model uses different size detection boxes when extracting object features and generates various feature maps that strengthen the network's feature extraction ability. The network structure of the SSD model mainly contains two parts, as shown in the following Figure 3.9.

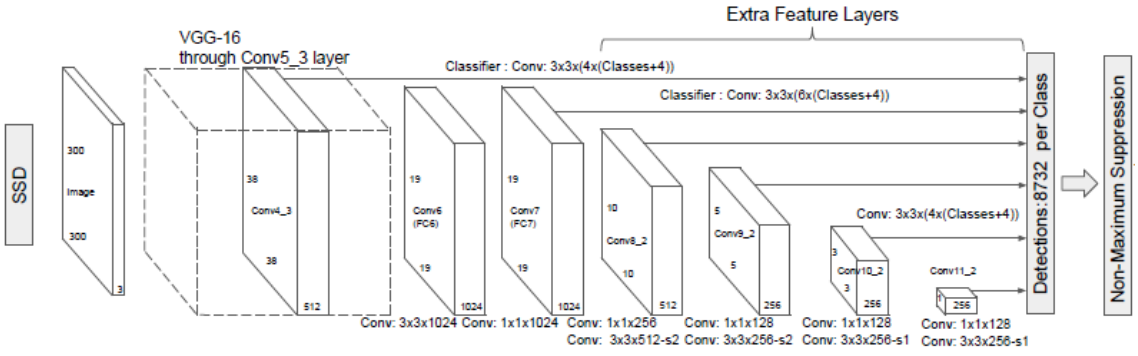


Figure 3.9 Diagram of SSD network structure

The first part is the basic feature extraction network, which uses the VGG-16 network without the dropout layer, FC8 and softmax classification layers. It replaces the fully connected layers FC6 and FC7 of the ordinary VGG network with the convolution layers Conv6 and Conv7 (Liu et al., 2016). The other part is a multi-scale

feature detection network, which is a convolutional neural network with a feature pyramid structure. In the second part, four convolution layers of Conv8, Conv9, Conv10, Conv11 have been newly added. Each convolution layer uses a 1×1 convolution kernel for dimensionality reduction and then uses a 3×3 convolution kernel for feature extraction (Yao et al., 2019). Then it forms a multi-scale feature detection network with the previous Conv4 and Conv7 layers, and performs feature detection under different scale conditions on the extracted feature maps.

The SSD model extracts 6 sets of feature maps with different scales from Conv4_3, Conv7, Conv8_2, Conv9_2, Conv10_2, Conv11_2. And then sends these 6 sets of feature maps to perform 3×3 convolution operations and generates different scales prediction boxes and its corresponding classification probability (Kang, 2020).

Table 3.1 The total number of bounding boxes calculation in SSD model

Conv4_3	38×38×4	5776
Conv7	19×19×4	2166
Conv8_2	10×10×6	600
Conv9_2	5×5×6	150
Conv10_2	3×3×4	36
Conv11_2	1×1×4	4
Total	-	8732

As calculated in the above table, multiplying by 4 or 6 is because the number of detection boxes that the feature map is divided into in each convolution layer is 4 or 6. The SSD model finally has 8732 bounding boxes outputs. Compared to YOLOv1, there are 7×7 grids, and each grid only has 2 bounding boxes, then 98 (7×7×2) bounding boxes in total. The SSD model has more bounding boxes, making the network learn more features, thereby improving the detection accuracy.

The SSD model is set with a default detection box. In the training phase, the IoU value of the default detection box and the bounding box in the ground truth determines the positive sample of the negative sample. The positive sample means that

it contains the target. Otherwise, the negative sample is the background. And through the non-maximum suppression (NMS) algorithm removes the redundant overlapping detection boxes of the same target (Y. Yu, Han, & Du, 2020). In the loss calculation phase, it encodes the default box and ground truth box first and then performs the loss calculation between them and the prediction box. In the testing phase, the SSD model detects each default box, and the default box with a confidence score greater than the threshold indicates that it contains the target so that the prediction box is obtained finally.

In any object detection model, the loss function is the compulsory component. The loss function of the SSD model consists of two parts, namely the localization loss (L_{loc}) and the confidence loss (L_{conf}) (Hao, Yingkun, & Yi, 2019). The entire loss function is the weighted sum of the localization loss and the confidence loss, as shown in 3-7.

$$\boldsymbol{L(x,c,l,g)} \;=\; \frac{\boldsymbol{1}}{\boldsymbol{N}}\Big(\boldsymbol{L_{conf}(x,c)} \;+\; \boldsymbol{\alpha L_{loc}(x,l,g)}\Big) \tag{3-7}$$

In equation 3-7, N represents the number of positive instances in the prediction box, that is, the number of prediction boxes that can be successfully matched with the ground truth labelled box. c is the confidence of the predicted classification, l is the prediction box by the model, g is the ground truth labelled box, and α is the weight coefficient of the localization loss and the confidence loss.

The confidence loss function (L_{conf}) adopts the Softmax Loss (Chen, Huang, Zhou, & Tan, 2018), the input is the confidence of each classification c, then L_{conf} calculation equation is as follows.

$$\boldsymbol{L_{conf}(x,c)} \;=\; -\sum_{\boldsymbol{i\in Pos}}^{\boldsymbol{N}} \boldsymbol{x_{ij}^{p} log\big(\hat{c}_i^{p}\big)} - \sum_{\boldsymbol{i\in Neg}} \boldsymbol{log\big(\hat{c}_i^{0}\big)} \tag{3-8}$$

$$\hat{C}_i^p = \frac{exp(c_i^p)}{\sum_p exp(c_i^p)} \tag{3-9}$$

The localization loss function (L_{loc}) adopts the Smooth L1 Loss (G. Yu, Fan, Zhou, Wu, & Zhu, 2020) to be the parameters of the prediction box (l) and the ground truth labelled box (g). And it also includes the center coordinate position (x, y), the width w and the height h. So the equation displays as below.

$$smooth_{L1}(x) = \begin{cases} 0.5x^2, & |x| < 1 \\ |x| - 0.5, & |x| \geq 1 \end{cases} \tag{3-10}$$

$$L_{loc}(x, l, g) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx, cy, w, h\}} x_{ij}^k smooth_{L1}(l_i^m - g_i^m) \tag{3-11}$$

In the above equation 3-11, g_i^m is the offset of the ground truth labelled box relative to the default detection box, and l_i^m is the prediction box output by the model. Therefore, the prediction box output by the SSD model is not the direct coordinates of the prediction box but the offset of the prediction box relative to the detection box.

3.3 Dataset Collection

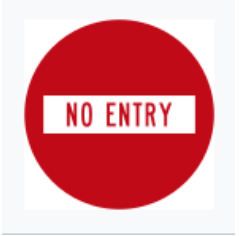
At present, there are many public traffic sign datasets from different countries. Representatives are German Traffic Sign Detection Benchmark (GTSDB), Belgium TS and Tsinghua-Tencent 100K (TT100K). And these famous datasets have been widely used in numerous previous researches. However, in this report, we decide to collect the traffic sign images in New Zealand as the dataset to implement the traffic sign recognition experiment.

Traffic signs can vividly express their meaning through a few words and symbols, and instruct pedestrians and vehicles to complete the traffic behaviours correctly. In

this report, we select 8 classifications of traffic signs with high awareness and important safety significance in New Zealand. Because of a low density of traffic signs on the road of New Zealand, we collected traffic sign images by cameras instead of driving videos. I invited my friends for help, and we split into two groups. Both two groups walked on the street in Auckland and used the phone's camera to take images. One group used Huawei P40, and the other group used Samsung Galaxy Note8.

Eventually, our dataset is composed of 2182 traffic sign images, namely NZTS-2K. Among them, No U-turn (271 images), Road bump (329 images), Road works (294 images), Watch for children crossing (176 images), Crosswalk ahead (313 images), Give way (317 images), Stop (286 images) and No entry (196 images), which are shown in Table 3.2.

Table 3.2 Dataset summarization of NZST-2K

No U-turn		271	Road bump		329
Road works		294	Watch for children crossing		176
Crosswalk ahead		313	Give way		317
Stop		286	No entry		196

3.4 Dataset Preprocessing

For the raw data in our dataset, a few images were in landscape view. First of all, we had to use a software tool to convert any images that are in landscape view to be portrait. In this project, we used the software called JPEG Autorotate to rotate images for us. After that, due to the ultra-high definition images leading to too long training time, we resized the image in the same proportion of its width and height. So we normalized all images in our dataset to be 1128×2016 and 1536×2048 and formatted them to be *.jpg* extension. So far, all images' preprocessing tasks had been done. Next, we labelled images based on different models' requirements, obtained corresponding annotation files, and split them into training, validation, and test datasets.

3.4.1 Dataset Preprocessing for YOLOv5 Experiment

As preparation of the image's annotation file for the model training in YOLOv5, it requires a *.txt* file contains the label information. In this report, we used a labelling tool, namely *Labellmg*. Especially, we needed to change the format to be *YOLO* since the default is *PascalVOC*.

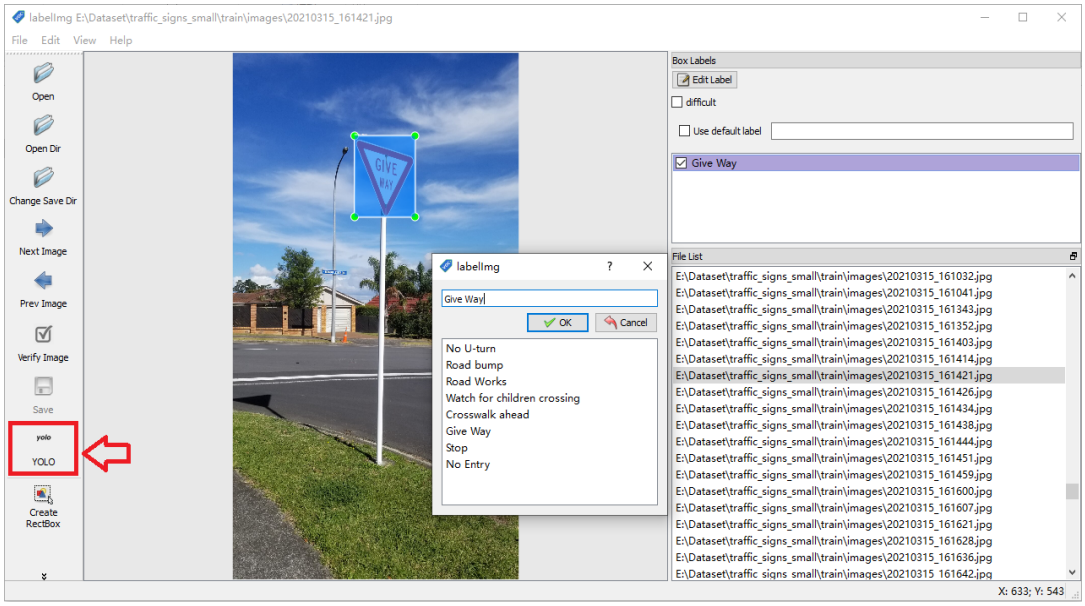


Figure 3.10 Labellmg GUI for labelling an image in the YOLOv5 dataset

As shown in Figure 3.10, we created a rectangle box to cover the object in the

image. Each label comprises five parameters: the index of classification, the center point coordinates (x, y), the width and height. Once all images' labelling done, we divided all images in our dataset into training and test dataset in the proportion of eight and two. And put images and corresponding annotation files into the */images* and */labels* folders respectively, as the structure in Figure 3.11.

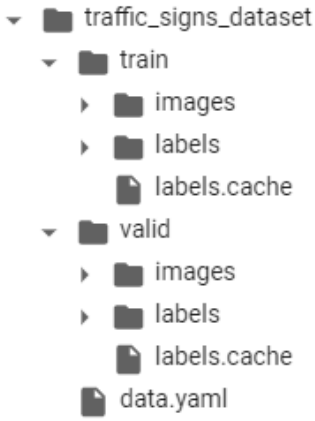


Figure 3.11 YOLOv5 dataset structure

3.4.2 Dataset Preprocessing for SSD Experiment

Compared to YOLOv5, the dataset used for training in the SSD model requires VOC2007 format. Hence, in *Labellmg*, we used the default format option, as shown in Figure 3.12.

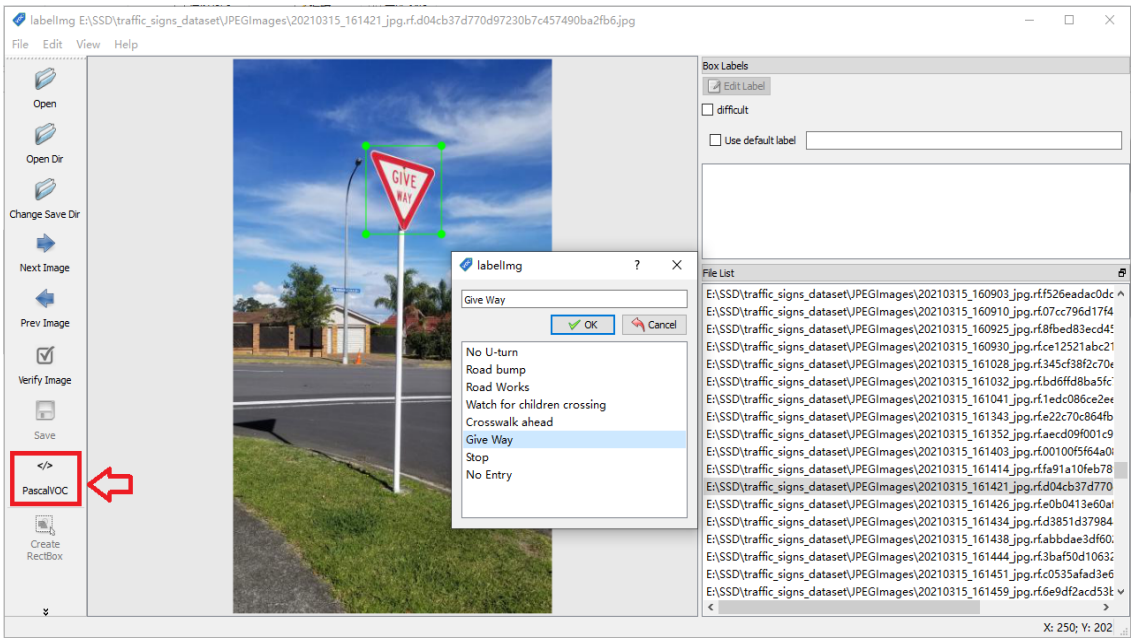


Figure 3.12 Labellmg GUI for labelling an image in the SSD dataset

After saving, the image label information would be generated and stored in the *.xml* file in the specified folder. The particular *.xml* also contains the label's classification, coordinates, width and height, which likes displayed in Figure 3.13.

```
<annotation>
    <folder></folder>
    <filename>20210315_161421_jpg.rf.d04cb37d770d97230b7c457490ba2fb6.jpg</filename>
    <path>20210315_161421_jpg.rf.d04cb37d770d97230b7c457490ba2fb6.jpg</path>
    <source>
        <database>Unkown</database>
    </source>
    <size>
        <width>384</width>
        <height>640</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>Give Way</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <occluded>0</occluded>
        <bndbox>
            <xmin>165</xmin>
            <xmax>246</xmax>
            <ymin>103</ymin>
            <ymax>201</ymax>
        </bndbox>
    </object>
</annotation>
```

Figure 3.13 An example of the annotation file in the SSD dataset

The formal VOC2007 dataset contains *Annotations* folder, *ImageSets* folder, *JPEGImages* folder, *SegmentationClass* folder and *SegmentationObject* folder. Since this report is related to TSR, the last two folders are no longer required. Finally, our SSD dataset structure is shown in Figure 3.14.

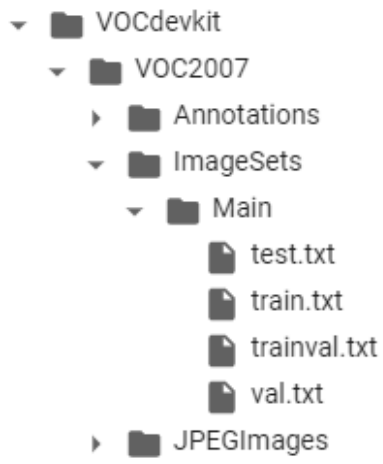


Figure 3.14 SSD dataset structure

Among them, the *Main* fold has four *.txt* files, which specifies corresponding

images' file names in test, training, validation, and training-validation datasets. For the dataset division, it needs to set up the specific proportion.

```
[ ]  import os
     import random

     trainval_percent = 0.8
     train_percent = 0.8
     xmlfilepath = 'Annotations/'
     txtsavepath = 'ImageSets/Main'
     total_xml = os.listdir(xmlfilepath)
```

Figure 3.15 Code of SSD dataset division

In Figure 3.15, we see in this report that the number of the training-validation dataset is 80%, the number of the test dataset is 20%, and training's and validation's are 64% and 16%, respectively.

3.5 Experiment Implementation

To implement YOLOv5 and SSD's experiment with our dataset, in this report, we utilized the Google Colab platform with powerful GPU support. Here, we list a few key hardware configurations and parameters about our experiment environment.

Table 3.3 Experiment's key configurations and environment parameters

Operation System	Ubuntu 18.04.5 LTS
GPU	Tesla P100-PCIE-16GB
RAM	26GB
Hard Disk Drive	108GB
Programming Language	Python 3.7.10
CUDA	Version 11.0.228
PyTorch	Version 1.8.1

3.5.1 YOLOv5 Experiment Implementation

Once the experiment environment was already set up, we needed to mount our

Google Drive to Colab to access the prepared dataset. And we chose the largest network that was YOLOv5x for training in the experiment. The training command including key parameters shows in Figure 3.16.

Figure 3.16 YOLOv5 training command and parameters

In the above figure, we set up the image size to 640×640 pixels and the batch size to 16, and trained 200 epochs. Meanwhile, it required to invoke the correct dataset configuration file, which was */traffic_signs_small/data.yaml* in our case. After the training was done, various results graphs were generated and stored in the */runs/trains* folder. And we obtained the *best.pt* file, which was the trained model and round 167MB.

3.5.2 SSD Experiment Implementation

Firstly, the same as YOLOv5, we mounted the drive to access the prepared dataset stored in Google Drive. Then in the *train.py* file, the batch size was also set up to 16, and the variables of *Init_Epoch*, *Freeze_Epoch* and *Unfreeze_Epoch* were token as 0, 100 and 200, respectively. After a long time model training, the output file with the *.pth* extension would be generated, which was the final model of our dataset. Then the last step was to fill in the correct model file path and pre-designed parameters shown in Figure 3.17 to finish the experiment.

```
_defaults = {
    "model_path"        : 'logs/Epoch200-Total_Loss1.4382-Val_Loss1.2025.pth',
    "classes_path"      : 'model_data/voc_classes.txt',
    "input_shape"       : (300, 300, 3),
    "confidence"        : 0.5,
    "nms_iou"           : 0.45,
    "cuda"              : True
}
```

Figure 3.17 Experiment parameters of SSD

3.6 Evaluation Methods

For deep learning algorithms, usually one or two evaluation indicators cannot evaluate the model's performance. Instead, multiple evaluation indicators need to be comprehensively considered. Therefore, in this report, we use the detection indicators, such as IoU, Precision, Recall, F1-score and mAP, to evaluate YOLOv5 and SSD models' performance on our dataset. Furthermore, we also consider the indicator of detection speed. Because for the actual traffic sign detection, whether it can meet the real-time detection requirements is critical.

(1). Intersection Over Union (IoU)

IoU evaluates the degree of overlap between the prediction box and the ground truth box. And its calculation equation is represented in 3-12. If the IoU value is greater than or equals the threshold, which is 0.5 here, we declare it as a positive sample (Jin et al., 2020).

$$\boldsymbol{IoU} = \frac{\boldsymbol{Prediction} \cap \boldsymbol{GroundTruth}}{\boldsymbol{Prediction} \cup \boldsymbol{GroundTruth}} \geq \mathbf{0.5} \tag{3-12}$$

(2). Precision and Recall

Precision is the ratio of the number of correctly detected objects to the total number of detected objects, and it examines the accuracy of the detection model. However, recall refers to the ratio of the number of correctly detected objects to the number of real objects, and it examines the recall level of the detection model (Yuan, Xiong, & Wang, 2016). And Precision and Recall represent in Equation 3-13 and 3-14, respectively.

$$\boldsymbol{Precision} = \frac{\boldsymbol{TP}}{\boldsymbol{TP} + \boldsymbol{FP}} \tag{3-13}$$

$$\boldsymbol{Recall} = \frac{\boldsymbol{TP}}{\boldsymbol{TP} + \boldsymbol{FN}} \tag{3-14}$$

In the above equations, TP represents the number of positive samples correctly predicted to be positive samples. FP represents the number of negative samples that are incorrectly predicted to be positive samples. And FN represents the number of positive samples that are incorrectly predicted to be negative samples (Dewi, Chen, & Tai, 2020).

(3). F1 Score

Precision and recall are often contradictory. If simply use precision or recall, it cannot accurately evaluate the detection performance. Therefore, the F1 score is introduced as an integrated performance indicator of Precision and Recall (D. Li, Zhao, Chen, & Zhang, 2018). Its value range is between 0 and 1. The larger the value is, the better the detection performance would be. The equation is shown in 3-15.

$$\boldsymbol{F1} = \frac{\boldsymbol{2} \times \boldsymbol{Precision} \times \boldsymbol{Recall}}{\boldsymbol{Precision} + \boldsymbol{Recall}} \tag{3-15}$$

(4). Mean Average Precision (mAP)

Another way to comprehensively consider precision and recall is the Precision and Recall (PR) curve. While the average precision (AP) refers to the area under the PR curve. And the mean average precision (mAP) refers to the mean of multiple categories' AP values (Arcos-Garcia, Alvarez-Garcia, & Soria-Morillo, 2018). Hence, mAP is the most important performance indicator in object detection. The larger the mAP value, the better the detection performance of the model. AP and mAP calculation equations are displayed in Equation 3-16 and 3-17 below.

$$\boldsymbol{AP} = \int_{\boldsymbol{0}}^{\boldsymbol{1}} \boldsymbol{p}(\boldsymbol{r})\boldsymbol{dr} \tag{3-16}$$

$$\boldsymbol{mAP} = \frac{\boldsymbol{1}}{\boldsymbol{N}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} \boldsymbol{AP_i} \tag{3-17}$$

Chapter 4

Our Results

This chapter is mainly to demonstrate our experimental results of YOLOv5 and SSD, respectively. In addition, the relevant illustration matched with the corresponding evaluation methods is expounded.

4.1 Data Description

By removing duplicated and poor-quality images, our final dataset (New Zealand Traffic Signs) consists of 2,182 images with 8 classes, namely, “No U-turn”, “Road bump”, “Road works”, “Watch for children crossing”, “Crosswalk ahead”, “Give way”, “Stop” and “No entry”. For the pixel size of images in the dataset, two categories are 1128×2016 and 1536×2048. That’s generated by using two different data collection equipment. Figure 4.1 represents the distribution of 8 classes in our dataset.

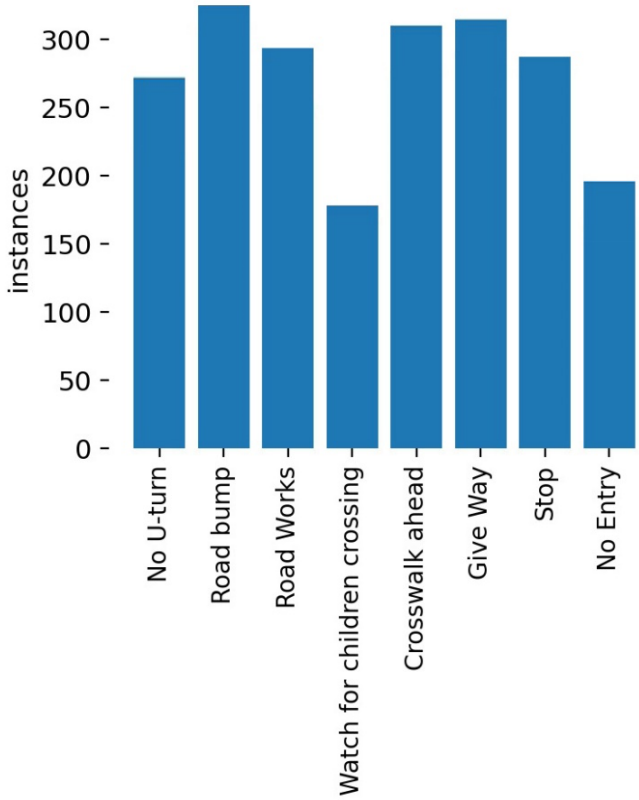


Figure 4.1 The histogram of eight classes distribution in our dataset

Due to the particularity of traffic signs, a large number of traffic signs in the same class cannot appear in the same road section, the traffic signs with special functions are even rarer. As a result, there is a big difference in the number of different classes of traffic signs in the collected dataset, which leads to a shortage of the overall samples and serious problem of imbalanced samples between different classes. However, from Figure 4.1, we clearly see that the number of instances of each class in our dataset is sufficient and basically balanced, no significant difference exists. The two classes with

the fewest instances are “Watch for children crossing” and “No entry”, but both of them are over 150 instances. In consequence, the reliability of the dataset is guaranteed that it is conducive to the diversity of the dataset. The model after training can maintain consistent detection accuracy for different classes of traffic signs, and has better generalization ability and adapt to a variety of environments.

Furthermore, another important type of charts about the dataset is the label distribution map, which reflects the label position in the dataset. Firstly, it’s the heatmap of labels position as shown in Figure 4.2.

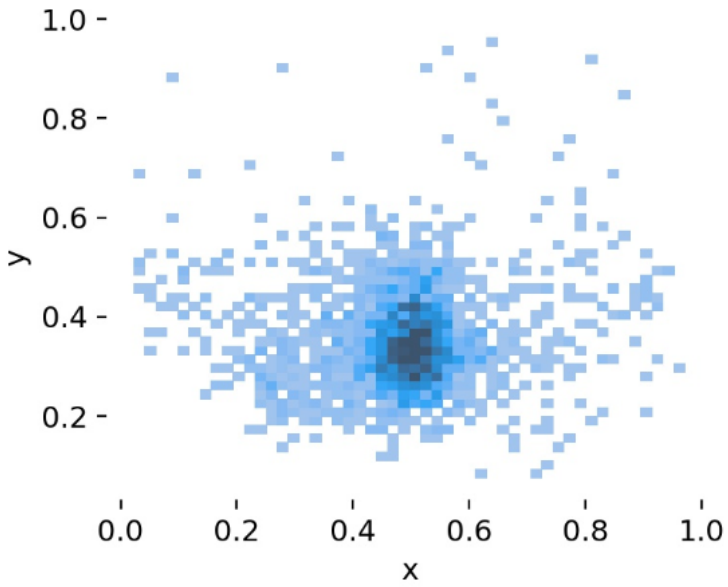


Figure 4.2 The heatmap of label position in our dataset

In Figure 4.2, each point represents a traffic-sign sample in the dataset, the *x*-axis and *y*-axis show the coordinates of a traffic-sign labelled in the corresponding image. We notice the coordinates are scale values that are normalized by the sigmoid function to be in the range (0, 1). The benefit is that it easily obtains the position of the label in the corresponding image even if the image size might be various. In Figure 4.2, the labels in our dataset are mainly concentrated in the middle of images. There are also a few scattered in the corners of images. Secondly, Figure 4.3 reflects the labels in images which is called the heatmap of labels.

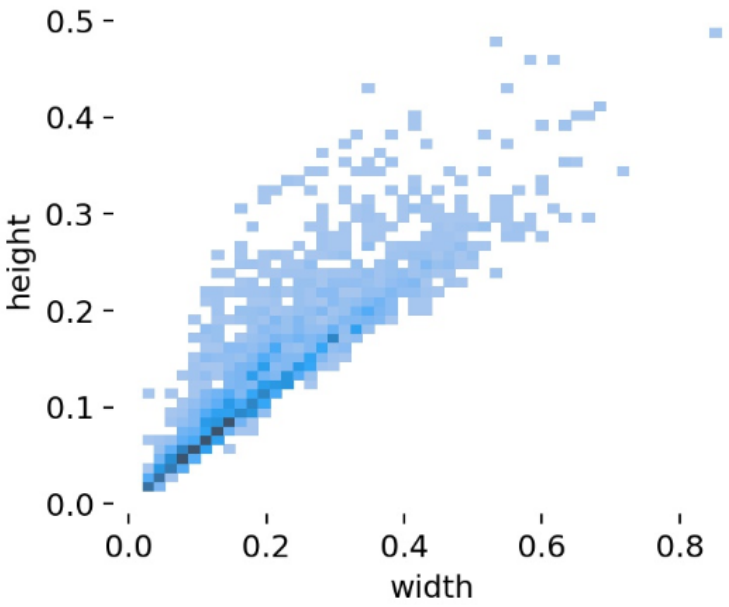


Figure 4.3 The heatmap of label dimension in our dataset

The width and height of each label is plotted as a point, which are also normalized by the sigmoid function within the range (0, 1). Obviously, like general object detection, the object to be detected occupies most area of the image, the structure is clear, and it is easy to be detected. For traffic-sign detection, if the less pixel information is acquired, the fewer features will be extracted. According to Figure 4.3, in our dataset, the scales of width and height are around 0.15 and 0.1. The label size is roughly in the resolution 210 × 210 from the original image with the resolution 1536 × 2048. Meanwhile, we observe that there are also a few small proportion and big proportion instances in our dataset.

Therefore, in summary, our dataset takes into account the diversification regardless of the position of the label or the size of the label. This is conducive to our trained model to improve the accuracy of traffic signs recognition of various sizes and in various environments.

4.2 Experiment Results of YOLOv5

In this experiment, we used YOLOv5x in YOLOv5 net as one of the models to proceed NZ-TSR with our dataset on the Google Colab platform. YOLOv5 model has an

excellent visualization function in the result, at first, we visually display its final recognition results for our dataset, as shown in Figure 4.4.

Figure 4.4 TSR results by using YOLOv5

First of all, in a complex environment, YOLOv5 is capable of detecting the traffic signs in images keenly. The detected traffic signs are further recognized to the specific class correctly. The accuracy of “No U-turn” and “Road bump” recognition could reach 100%, and others are all above 90% as well.

Furthermore, we also obtain the experiment results from the perspective of digitized charts. The confusion matrix is to summarize the true classification labelled in the dataset and the classification predicated by the model in a matrix form. As shown in Figure 4.5, we observe the final prediction results intuitively that YOLOv5 could almost achieve the perfect classification prediction of all 8 classes of traffic signs in our dataset.

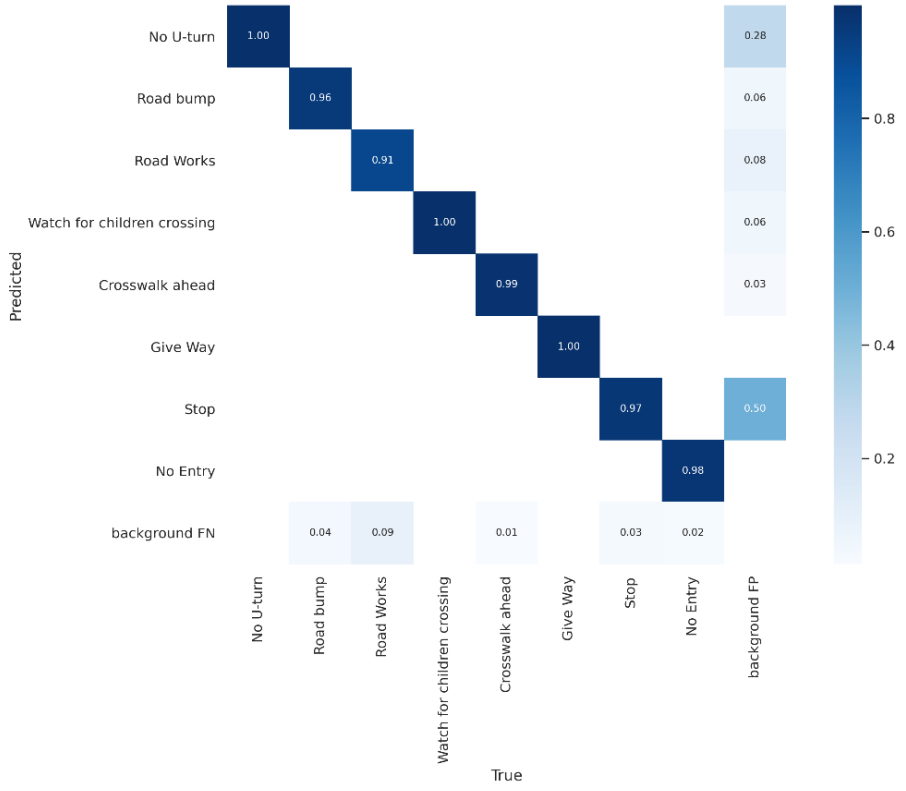


Figure 4.5 The confusion matrix using YOLOv5 based on our dataset

After running 200 epochs, we acquire the final results in 9.814 hours training time. We evaluate the YOLOv5 performance by using four main measurements, which are Precision, Recall, Mean Average Precision with IoU=0.5 (mAP@0.5), and Mean Average Precision with IoU from 0.5 to 0.95, step 0.05 (mAP@.5:.95). All of

them are summarized in Table 4.1.

Table 4.1 YOLOv5 experiment results summary with our dataset

Classes	Precision	Recall	mAP@0.5	mAP@.5:.95
No U-turn	0.937	1	0.995	0.9
Road bump	1	0.903	0.965	0.822
Road works	0.979	0.897	0.928	0.781
Watch for children crossing	0.997	1	0.995	0.904
Crosswalk ahead	1	0.979	0.986	0.93
Give way	1	1	0.995	0.905
Stop	0.984	0.938	0.975	0.834
No entry	1	0.938	0.976	0.827

According to the above table, we see the precisions of “Road bump”, “Crosswalk”, “Give way” and “No entry” achieve the value of 1 dramatically. Even if the lowest precision obtained by “No U-turn” is also 0.937. In terms of Recall, the values for all 8 classes indicate the excellent TSR performance of YOLOv5 in our dataset as well. In order to demonstrate the relations among Precision, Recall, Figure 4.6 is extremely desired here.

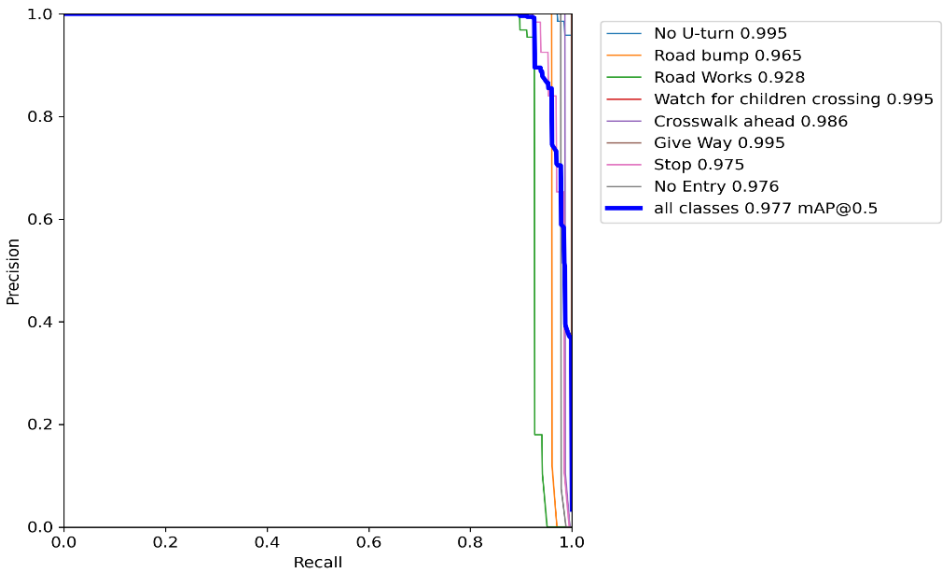


Figure 4.6 Precision-recall curves based on YOLOv5

From Figure 4.6, if the recall is lower than 0.8, the final recognition accuracy of 8 classes in our dataset is all close to 100%, which almost gets the completely accurate prediction. Overall, the mean average precision of “No U-turn”, “Watch for children crossing” and “Give way” are as high as 99.5%. Meanwhile, all classes are as high as 97.7%. In the end, all specific performance metrics of YOLOv5 in our dataset are shown in Figure 4.7.

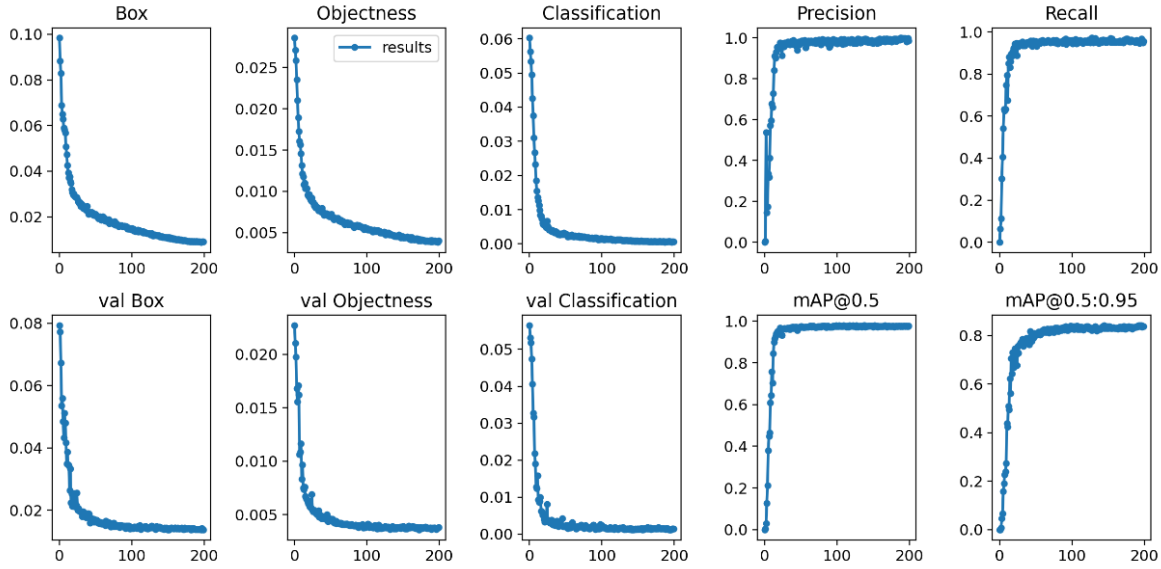
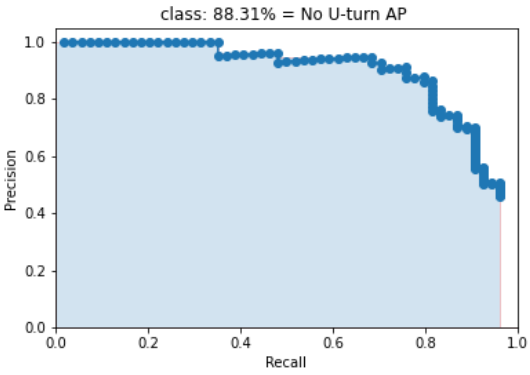


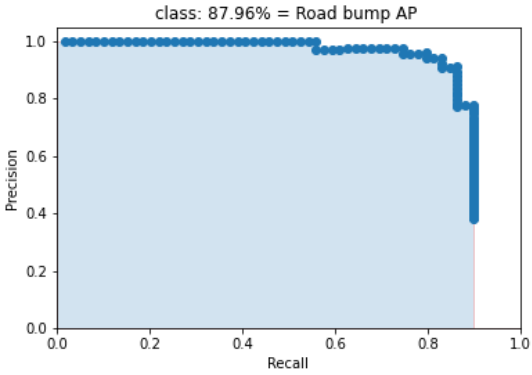
Figure 4.7 All specific performance metrics of YOLOv5 in our dataset

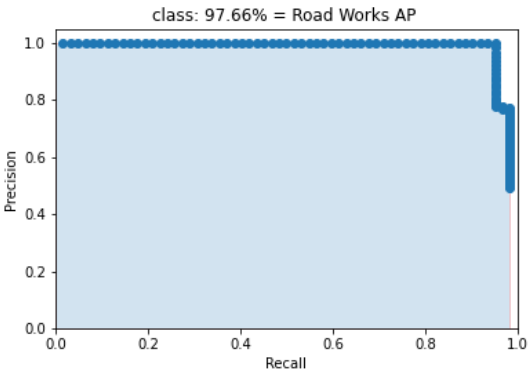
In Figure 4.7, the first column is the mean of the box loss function on the training dataset and the validation dataset respectively. The smaller the value is, the more accurate the box of the detected label is. Similarly, the second and third columns are the mean values of the loss functions of the training dataset and the validation dataset for the object detection and classification respectively. The smaller the value is, the better the recognition effect of the model is. The last two columns are performance indicators of precision, recall, mAP@0.5 and mAP@.5:.95, which have been discussed in Table 4.1 and will not be repeated here. Therefore, from all experimental results of YOLOv5, the model achieves a very precise TSR results by using our dataset.

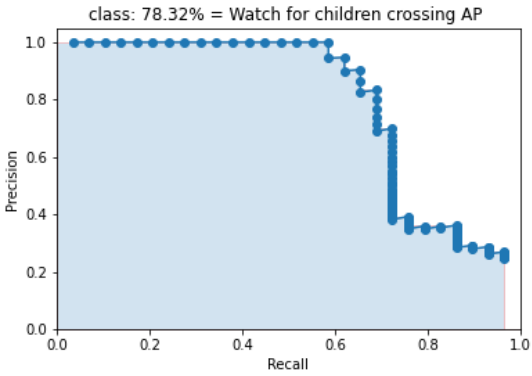
4.3 Experiment Results of SSD Neural Network

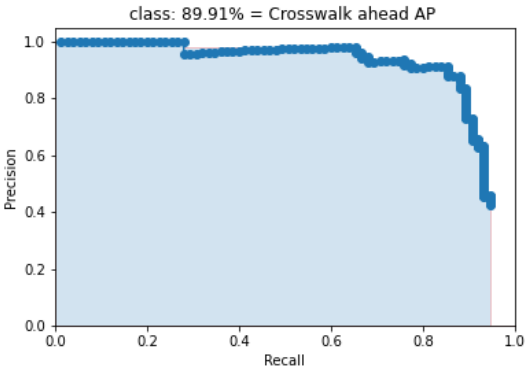
Usually, the most convenient and direct way to evaluate the experiment results is accuracy, in this report, we demonstrate the precision in the experiment results of SSD neural network with our dataset.

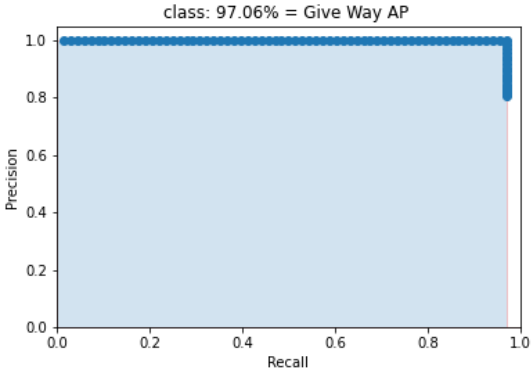












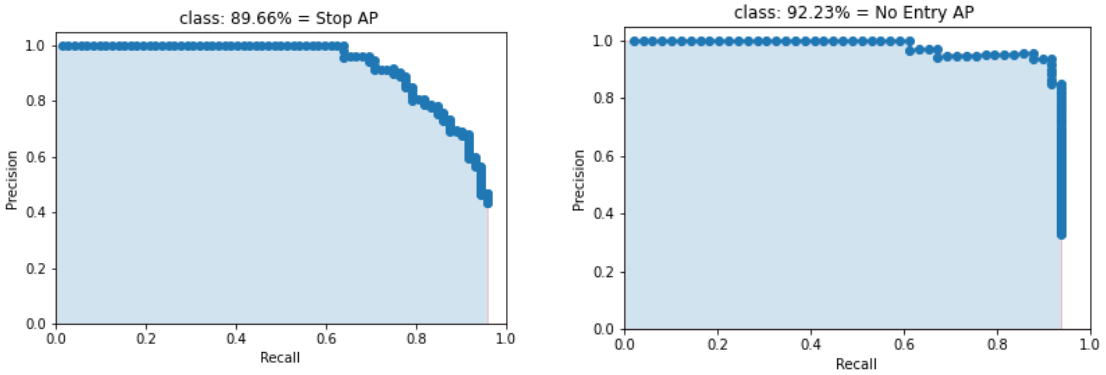
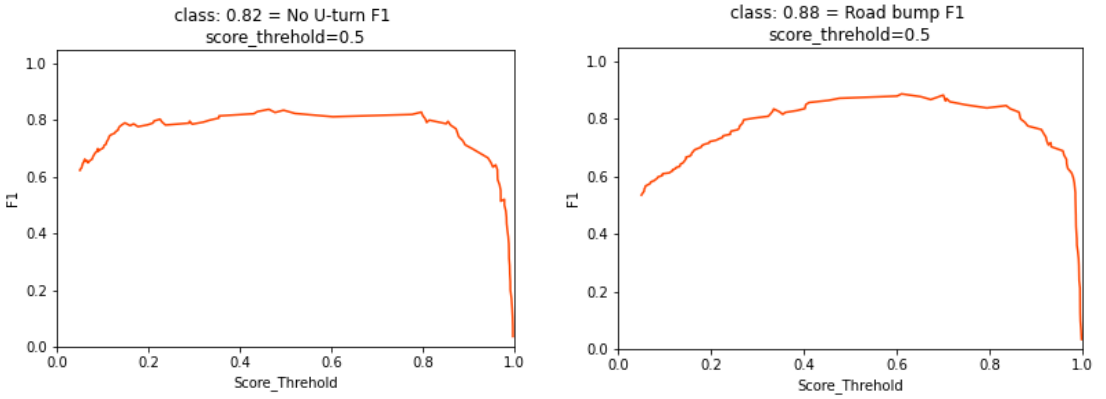


Figure 4.8 Precision vs Recall for 8 classes in SSD experiment with our dataset

From Figure 4.8, we see the prediction results of SSD neural network for 8 classes of TSR by using our dataset. Overall, the accuracy of recognition and classification of most classes reach nearly 90%. In particular, TSR for "Give way" has the best result, the final average precision is as high as 97.06%. However, the average precision of "Watch for children crossing" is quite low, only 78.32%.

In previous chapter, precision and recall are a pair of contradictory metrics. Hence, this report needs to introduce another evaluation criteria, namely F1-measure, also known as F-score. Based on the formula of F1 value, it is not difficult to see that F1 combines the results of precision and recall. When F1 value is higher, the experimental model is more ideal. The following graphs specifically show the experiment results of F1 values for 8 classes of traffic signs in the SSD neural network when score_threhold = 0.5.



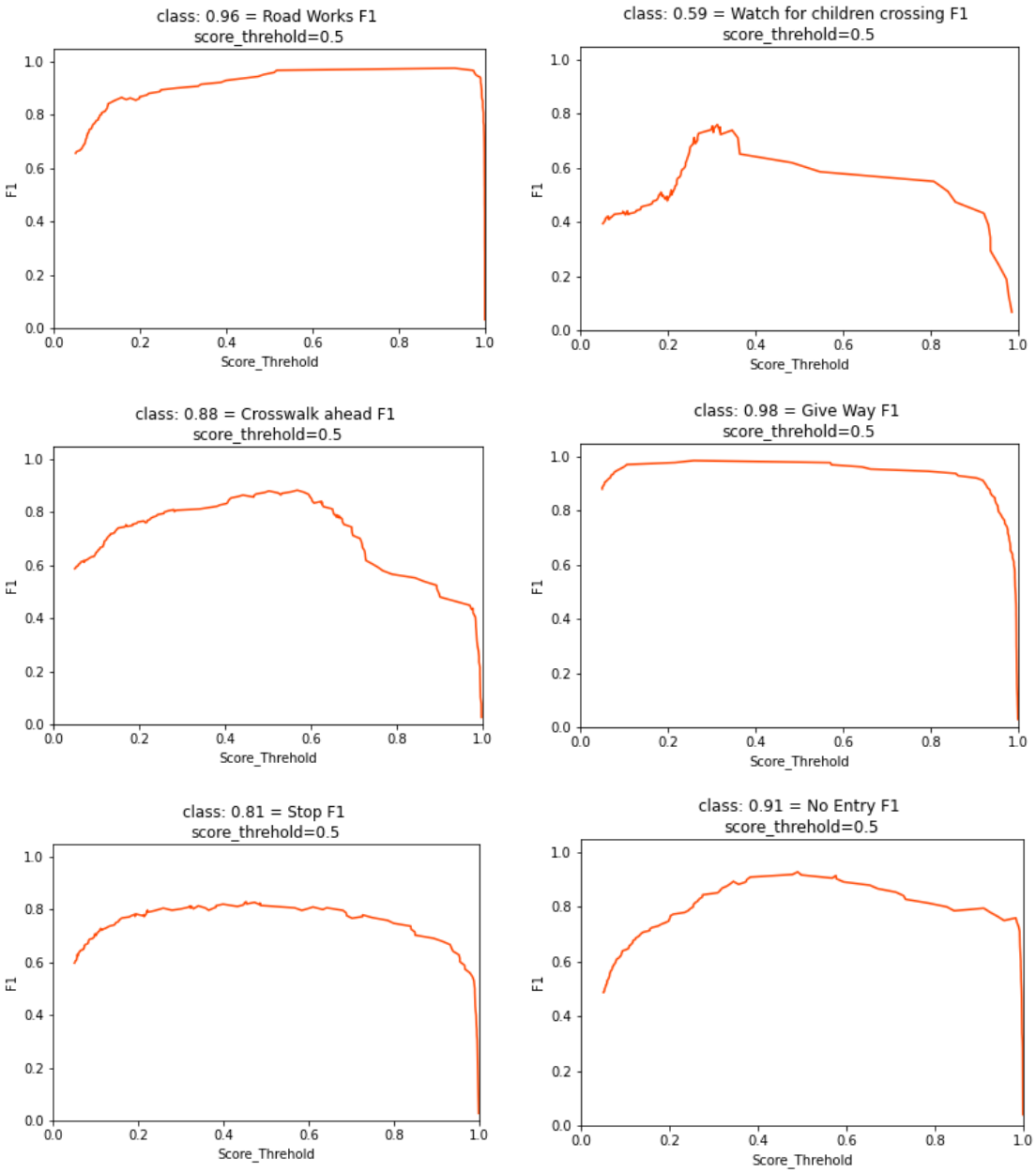


Figure 4.9 F1 for 8 classes in SSD experiment with our dataset

In Figure 4.9, F1 values of traffic-sign recognition for 8 classes in SSD neural network experiment get displayed clearly. Except for “Watch for children crossing”, all other classes’ F1 values are above 0.8, which is quite impressive. On the contrary, F1 of “Watch for children crossing” is only 0.59. The reason is highly possible that the number of instances of that specific class is lower than others. Finally, we summarize various performance of SSD neural network experiment with our dataset and display them in Table 4.2.

Table 4.2 SSD neural network experimental results with our dataset

Classes	Precision	Recall	F1	mAP@0.5
No U-turn	0.875	0.778	0.82	0.881
Road bump	0.895	0.864	0.88	0.88
Road works	0.967	0.952	0.96	0.977
Watch for children crossing	1	0.414	0.59	0.783
Crosswalk ahead	0.88	0.88	0.88	0.899
Give way	1	0.956	0.98	0.971
Stop	0.912	0.722	0.81	0.897
No entry	0.935	0.878	0.91	0.922
* Above experiment results are based on score_threhold = 0.5				

All results illustrate the model has a relatively good outcome based on in our dataset apart from “Watching for children crossing”. In the SSD neural network, the accuracy of the class prediction is proportional to the number of instances in each class. In other words, for SSD neural network, the more instances the dataset contains, the more accurate prediction will be.

Chapter 5
Analysis and Discussions

In this chapter, our focus is on the comparison between YOLOv5 and SSD based on our experimental results. The substantial analysis and discussion will be carried out according to the comparison result. At last, the limitations of this project will be explicated as well.

5.1 Analysis and Discussion

According to the experiment results shown in the previous chapter, we conduct a comprehensive comparison of the key performance metrics of YOLOv5 and SSD neural network in our dataset. The detailed results are displayed in Table 5.1.

Table 5.1 The mean average precision of YOLOv5 and SSD summary

Class	YOLOv5	SSD neural network
No U-turn	0.995	0.883
Road bump	0.965	0.880
Road works	0.928	0.977
Watch for children crossing	0.995	0.783
Crosswalk ahead	0.986	0.899
Give way	0.995	0.971
Stop	0.975	0.897
No entry	0.976	0.922
Overall	0.977	0.901

In Table 5.1, we see intuitively that the mean average precision of all 8 classes obtained by YOLOv5 and SSD neural network is 0.977 and 0.901 respectively. From the perspective of each class accuracy, the experiment results of YOLOv5 are all better than SSD's except for "Road works", the performance of YOLOv5 for "No U-turn", "Watch for children crossing" and "Give way" is remarkable, which has reached 0.995, close to 100% detection. For the SSD experiment, the highest recognition accuracy is 0.977 obtained in "Road works". However, in "Watch for children crossing" with a small number of samples, the accuracy goal is only 0.78. In the end, on the aspect of TSR accuracy in our dataset, both YOLOv5 and SSD neural networks showed good capabilities, but YOLOv5 performed a bit better.

In the field of traffic-sign recognition, the recognition accuracy of the model is important, but the recognition speed is also an indispensable metric. Therefore, in the experiment of our dataset, we compare the efficiency of two models based on YOLOv5 and SSD neural network, and display the comparison result in Table 5.2.

Table 5.2 The traffic-sign recognition efficiency of YOLOv5 and SSD summary

	YOLOv5	SSD neural network
The number of images (frame)	450	450
Total time cost (second)	15	129
Efficiency (frame per second)	30	3.49

According to the recognition time cost by two models in the above table, with the same number of images in the test dataset, YOLOv5 used only 15s, while SSD neural network needed 129s. The speed of TSR by using YOLOv5 is 30 fps (frame per second), nearly 10 times faster than SSD's, which is 3.49 fps. In other words, on the aspect of TSR by using our dataset, YOLOv5 performed better than SSD as well.

5.2 Limitations of This Project

Although we have spent a lot of time and efforts on this project, the shortcomings and limitations still exist objectively in our project. These limitations are mainly reflected in three aspects. First of all, the dataset used in this experiment does not include all categories of traffic signs in New Zealand. We collected only eight classes that we thought were representative and had important safety significance. Nevertheless, this practice itself has subjective view. Compared to widely used public datasets such as GTSDB, Belgium TS and TT-100K, the number of instances in each class in our dataset is still too few. The traffic sign images in our dataset were captured under a limited condition. It lacks a balanced number of samples to truly simulate different illumination intensity, different distances, omni-directional angles, and coverage or damage cases.

Secondly, even though the final TSR accuracies obtained in our experiment are impressive and inspiring. The way to overcome this limitation is to explore combinations of parameters for better performance, and compare two recognition algorithms when they have as much better performance as possible.

Finally, we only compared the performance of YOLOv5 and SSD neural network based on our NZ-TSR dataset. However, except for those two algorithms used in this project, we have explored more object recognition algorithms with great performance, such as Faster R-CNN, Mark R-CNN and Siamese neural network. Therefore, the outcome of this project cannot answer the question of which algorithm has the best performance in TSR until more results of comparisons are given.

Chapter 6

Conclusion and Future Work

In this chapter, we will make a conclusion for the entire project based on our experimental results. Meanwhile, the perspective of future research will be visioned.

6.1 Conclusion

This project aims to prob the accuracy and efficiency of the object recognition model based on deep learning based on the dataset of New Zealand traffic signs. Hence, in this report, we select the latest version of the YOLO series of algorithms, namely YOLOv5, to evaluate its performance. Besides, in this report, we also identify which model is more suitable for our classification between YOLOv5 and SSD net, which both belong to one-stage object recognition methods in deep learning.

In this experiment, we adopt a customized dataset of New Zealand traffic signs collected by ourselves since no such existing dataset can best fit for our experiments. The dataset contains 2,182 traffic sign images having eight classes, which are captured manually on the Auckland streets. Then, we implemented a well-designed experiment on the Google Colab platform with powerful GPU having very strong computational capability. In addition, we also analyze and compare two models by using key evaluation metrics.

In the end, from the experimental results, we achieve impressive and inspiring performance from YOLOv5 based on our own dataset. Overall, its accuracy reaches 97.7% for all classes, the mean average precision in each class is over 90%. Relatively speaking, SSD neural network obtains 90.14% on the accuracy in general. But for the class with small samples, it only has 78.32% correctness. Therefore, we conclude that YOLOv5 performs better than SSD neural network in terms of recognition accuracy. Especially in the small samples group, YOLOv5 is able to more accurately detect and recognize the target than the latter. Furthermore, from the perspective of recognition speed, YOLOv5 is faster than SSD neural network with 30 fps (frames per second), SSD only has 3.49 fps. Consequently, we determine that the performance of YOLOv5 is superior to SSD net no matter on the aspect of the recognition accuracy or the recognition efficiency. We believe that YOLOv5 is much suitable for the traffic sign recognition in the real-time traffic environment.

6.2 Future Work

For the achievements obtained and the shortcomings exposed in this project, we define that we will carry out much work in future from two aspects. Firstly, we will keep extending our datasets to cover all categories of traffic signs in New Zealand. Meanwhile, more samples will be added to each category so that the expanded dataset has outstanding discriminative and covering. Secondly, we should expand the experimental model to include the newly developed models for object recognition, such as Mark R-CNN, Siamese neural network, etc. In addition, more professional evaluation measurements will be adopted to assess the performance of our models in the future experiment.

References

Arcos-Garcia, A., Alvarez-Garcia, J. A., & Soria-Morillo, L. M. (2018). Evaluation of deep neural networks for traffic sign detection systems. *Neurocomputing, 316*, 332-344.

Arcos-García, Á., Alvarez-Garcia, J. A., & Soria-Morillo, L. M. (2018). Deep neural network for traffic sign recognition systems: An analysis of spatial transformers and stochastic optimisation methods. *Neural Networks, 99*, 158-165.

Bangquan, X., & Xiong, W. X. (2019). Real-time embedded traffic sign recognition using efficient convolutional neural network. *IEEE Access, 7*, 53330-53346.

Bi, Z., Yu, L., Gao, H., Zhou, P., & Yao, H. (2020). Improved VGG model-based efficient traffic sign recognition for safe driving in 5G scenarios. *International Journal of Machine Learning and Cybernetics*, 1-12.

Chen, Q., Huang, N., Zhou, J., & Tan, Z. (2018). An SSD algorithm based on vehicle counting method. Chinese Control Conference (CCC)

Chung, J. H., Kim, D. W., Kang, T. K., & Lim, M. T. (2020). Traffic sign recognition in harsh environment using attention based convolutional pooling neural network. *Neural Processing Letters, 51*(3), 2551-2573.

Croce, F., Andriushchenko, M., & Hein, M. (2019). Provable robustness of relu networks via maximization of linear regions. International Conference on Artificial Intelligence and Statistics

Dewi, C., Chen, R.-C., & Tai, S.-K. (2020). Evaluation of robust spatial pyramid pooling based on convolutional neural network for traffic sign recognition system. *Electronics, 9*(6), 889.

Ellahyani, A., Ansari, M., Lahmyed, R., & Trémeau, A. A new traffic sign recognition method for intelligent vehicles. *Journal of the Optical Society of America*.

Fang, M.-t., Chen, Z.-j., Przystupa, K., Li, T., Majka, M., & Kochan, O. (2021). Examination of abnormal behavior detection based on improved YOLOv3. Electronics, 10, 197

Garg, P., Chowdhury, D. R., & More, V. N. (2019). Traffic sign recognition and classification using YOLOv2, Faster RCNN and SSD. International Conference on Computing, Communication and Networking Technologies (ICCCNT)

Gong, B., Ergu, D., Cai, Y., & Ma, B. (2020). A method for wheat head detection based on YOLOv4. Research Square.

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

Hao, G., Yingkun, Y., & Yi, Q. (2019). General Target Detection Method Based on Improved SSD. IEEE Joint International Information Technology and Artificial Intelligence Conference (ITAIC)

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

He, W., Huang, Z., Wei, Z., Li, C., & Guo, B. (2019). Tf-yolo: An improved incremental network for real-time object detection. *Applied Sciences, 9*(16), 3225.

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science, 313*(5786), 504-507.

Houben, S. (2011). A single target voting scheme for traffic sign detection. IEEE Intelligent Vehicles Symposium (IV)

Huo, A., Zhang, W., & Li, Y. (2020). Traffic sign recognition based on improved SSD model. International Conference on Computer Network, Electronic and Automation (ICCNEA)

Hussain, S., Abualkibash, M., & Tout, S. (2018). A survey of traffic sign recognition systems based on convolutional neural networks. IEEE International Conference on Electro/Information Technology (EIT)

Jin, Y., Fu, Y., Wang, W., Guo, J., Ren, C., & Xiang, X. (2020). Multi-feature fusion and enhancement single shot detector for traffic sign recognition. *IEEE Access, 8*, 38931-38940.

Jun, H., Shuai, L., Jinming, S., Yue, L., Jingwei, W., & Peng, J. (2018). Facial expression recognition based on VGGNet convolutional neural network. Chinese Automation Congress (CAC)

Kang, Y. (2020). Research on SSD base network. IOP Conference Series: Materials Science and Engineering

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM, 60*(6), 84-90.

Kuznetsova, A., Maleva, T., & Soloviev, V. (2020). Detecting apples in orchards using YOLOv3 and YOLOv5 in general and close-up images. International Symposium on Neural Networks

Lawal, M. O. (2021). Tomato detection based on modified YOLOv3 framework. *Scientific Reports, 11*(1), 1-11.

Li, D., Zhao, D., Chen, Y., & Zhang, Q. (2018). Deepsign: Deep learning based traffic sign recognition. International Joint Conference on Neural Networks (IJCNN)

Li, G., Song, Z., & Fu, Q. (2018). A new method of image detection for small datasets

under the framework of YOLO network. Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)

Li, J., Yuan, Z., Li, Z., Ren, A., Ding, C., Draper, J., . . . Wang, Y. (2019). Normalization and dropout for stochastic computing-based deep convolutional neural networks. *Integration, 65*, 395-403.

Li, P., Nguyen, M., Yan, W. (2018) Rotation correction for license plate recognition.International Conference on Control, Automation and Robotics.

Li, S., Gu, X., Xu, X., Xu, D., Zhang, T., Liu, Z., & Dong, Q. (2021). Detection of concealed cracks from ground penetrating radar images based on deep learning algorithm. *Construction and Building Materials, 273*, 121949.

Li, X., & Cui, Z. (2016). Deep residual networks for plankton classification. OCEANS 2016 MTS/IEEE Monterey

Lian, J., Yin, Y., Li, L., Wang, Z., & Zhou, Y. (2021). Small object detection in traffic scenes based on attention feature fusion. *Sensors, 21*(9), 3031.

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

Liu, X., Yan, W. (2020) Vehicle-related scene segmentation using CapsNets. IVCNZ.

Liu, X., Nguyen, M., Yan, W. (2019) Vehicle-related scene understanding using deep learning. Asian Conference on Pattern Recognition 1 (1), 1-12，45 2019

Luo, H., Yang, Y., Tong, B., Wu, F., & Fan, B. (2017). Traffic sign recognition using a multi-task convolutional neural network. *IEEE Transactions on Intelligent Transportation Systems, 19*(4), 1100-1111.

Misra, D. (2019). Mish: A self regularized non-monotonic activation function. *arXiv preprint arXiv:1908.08681*.

Molchanov, V., Vishnyakov, B., Vizilter, Y., Vishnyakova, O., & Knyaz, V. (2017). Pedestrian detection in video surveillance using fully convolutional YOLO neural network. Automated Visual Inspection and Machine Vision II

Nguyen, M., Yan, W., Gong, R., Delmas, P. (2015) Toward a real-time belief propagation stereo reconstruction for computers, robots, and beyond. IVCNZ

Pan, C., Yan, W. (2019) A learning-based positive feedback in salient object detection. IVCNZ

Pan, C. Yan, W. (2020) Object detection based on saturation of visual perception. Multimedia Tools and Applications 79 (27-28), 19925-19944

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

Shao, F., Wang, X., Meng, F., Zhu, J., Wang, D., & Dai, J. (2019). Improved faster R-CNN traffic sign detection based on a second region of interest and highly possible regions proposal network. *Sensors, 19*(10), 2288.

Shen, D., Xin, C., Nguyen, M., Yan, W. (2018) Flame detection using deep learning. International Conference on Control, Automation and Robotics (ICCAR)

Shen, Y., Yan, W. (2018) Blind spot monitoring using deep learning. IVCNZ

Shi, X., Hu, J., Lei, X., & Xu, S. (2021). Detection of flying birds in airport monitoring based on improved YOLOv5. International Conference on Intelligent Computing and Signal Processing (ICSP)

Shustanov, A., & Yakimov, P. (2017). CNN design for real-time traffic sign recognition. *Procedia engineering, 201*, 718-725.

Sun, W., Hongji, D., Nie, S., & He, X. (2019). Traffic sign recognition method integrating multi-layer features and kernel extreme learning machine classifier.

Comput. Mater. Continua, 60(1), 147-161.

Wang, C.-C., Samani, H., & Yang, C.-Y. (2019). Object detection with deep learning for underwater environment. International Conference on Information Technology Research (ICITR)

Wang, C. (2018). Research and application of traffic sign detection and recognition based on deep learning. International Conference on Robots & Intelligent System (ICRIS)

Wang, J., Ngueyn, M., Yan, W. (2017) A framework of event-driven traffic ticketing system. International Journal of Digital Crime and Forensics (IJDCF) 9 (1), 39-50

Wang, J., Yan, W. (2016) BP-neural network for plate number recognition. International Journal of Digital Crime and Forensics (IJDCF) 8 (3), 34-45

Wang, S.-H., Xie, S., Chen, X., Guttery, D. S., Tang, C., Sun, J., & Zhang, Y.-D. (2019). Alcoholism identification based on an AlexNet transfer learning model. *Frontiers in psychiatry, 10*, 205.

Wang, X., Qin, Y., Wang, Y., Xiang, S., & Chen, H. (2019). ReLTanh: An activation function with vanishing gradient resistance for SAE-based DNNs and its application to rotating machinery fault diagnosis. *Neurocomputing, 363*, 88-98.

Wang, Y., Li, Y., Song, Y., & Rong, X. (2020). The influence of the activation function in a convolution neural network model of facial expression recognition. *Applied Sciences, 10*(5), 1897.

Wu, Y., Qin, X., Pan, Y., & Yuan, C. (2018). Convolution neural network based transfer learning for classification of flowers. IEEE International Conference on Signal and Image Processing (ICSIP)

Yan, B., Fan, P., Lei, X., Liu, Z., & Yang, F. (2021). A real-time apple targets detection method for picking robot based on improved YOLOv5. *Remote Sensing, 13*(9), 1619.

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

Yang, J., & Yang, G. (2018). Modified convolutional neural network based on dropout and the stochastic gradient descent optimizer. *Algorithms, 11*(3), 28.

Yang, S., Bo, C., Zhang, J., & Wang, M. (2020). Vehicle logo detection based on modified YOLOv2. International Conference on Robotic Sensor Networks

Yanhua, J., Shengyan, Z., & Yan, J. (2011). Traffic sign recognition using ridge regression and OTSU method. Intelligent Vehicles Symposium (IV)

Yao, Y., Yang, Y., Su, X., Zhao, Y., Feng, A., Huang, Y., & Pu, H. (2019). Optimization of the bounding box regression process of SSD model. International Conference on Computer Engineering, Information Science & Application Technology (ICCIA 2019)

Yu, G., Fan, H., Zhou, H., Wu, T., & Zhu, H. (2020). Vehicle target detection method based on improved SSD model. *Journal of Artificial Intelligence, 2*(3), 125.

Yu, Y., Han, X., & Du, L. (2020). Target part detection based on improved SSD algorithm. Journal of Physics: Conference Series

Yuan, Y., Xiong, Z., & Wang, Q. (2016). An incremental framework for video-based traffic sign detection, tracking, and recognition. *IEEE Transactions on Intelligent Transportation Systems, 18*(7), 1918-1929.

Zhang, C., Yue, X., Wang, R., Li, N., & Ding, Y. (2020). Study on traffic sign recognition by optimized LeNet-5 algorithm. *International Journal of Pattern Recognition and Artificial Intelligence, 34*(01), 2055003.

Zhang, J., Huang, M., Jin, X., & Li, X. (2017). A real-time chinese traffic sign detection algorithm based on modified YOLOv2. *Algorithms, 10*(4), 127.

Zhang, J., Huang, Q., Wu, H., & Liu, Y. (2017). A shallow network with combined pooling for fast traffic sign recognition. *Information, 8*(2), 45.

Zhang, J., Hui, L., Lu, J., & Zhu, Y. (2018). Attention-based neural network for traffic sign detection. International Conference on Pattern Recognition (ICPR)

Zhang, J., Wang, W., Lu, C., Wang, J., & Sangaiah, A. K. (2020). Lightweight deep network for traffic sign classification. *Annals of Telecommunications, 75*(7), 369-379.

Zhang, M., Zhao, W., Li, H., Wang, F., & Zhang, S. (2019). Research on the application of deep learning YOLOv3 in aerial patrol inspection of optical cable lines. Journal of Physics: Conference Series

Zhao, C., & Zheng, W. (2020). Fast traffic sign recognition algorithm based on multi-scale convolutional neural network. International Conference on Advanced Cloud and Big Data (CBD)

Zheng, K., Yan, W., Nand, P. (2017) Video dynamics detection using deep neural networks. IEEE Transactions on Emerging Topics in Computational Intelligence

Zhou, K., Zhan, Y., & Fu, D. (2021). Learning region-based attention network for traffic sign recognition. *Sensors, 21*(3), 686.

Zhou, Y., Chang, H., Lu, Y., Lu, X., & Zhou, R. (2020). Improving the performance of VGG through different granularity feature combinations. *IEEE Access, 9*, 26208-26220.

Zoumpourlis, G., Doumanoglou, A., Vretos, N., & Daras, P. (2017). Non-linear convolution filters for CNN-based learning. IEEE International Conference on