Special Character Recognition Using Deep Learning

Changjian Li

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

2021

School of Engineering, Computer & Mathematical Sciences

Abstract

In recent years, deep learning methods have been applied to our daily lives and various industries. Visual object detection methods are broadly employed to a consortium of tasks, including human face detection in public areas, traffic signs detection, car plate number recognition, etc. Natural Language Processing (NLP) methods are implemented for language translation, Automatic Speech Recognition (ASR), client embedding, item embedding, etc.

In this thesis, we contribute to special character recognition by using deep learning. The Adaptive Bezier Curve Network (ABCNet) is a text detection and recognition method utilized to recognize English Braille, which implements parameterized Bezier curves for detecting arbitrary-shape text in natural scenes. YOLOv5 is the second deep learning method that was implemented for Māori symbol recognition. The methods show outstanding performance in our experiments. Both methods detect and recognize visual objects with high accuracies. The results of our experiments prove deep learning methods are feasible to be implemented for detecting and classifying special characters, shortening the time cost of translation, and reducing labor costs.

Keywords: Deep learning, object detection, scene text recognition and detection, ABCNet, YOLOv5

Table of Contents

List of Figures

List of Tables

Attestation of Authorship

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

Signature: Date: 01 November 2021

Acknowledgment

First of all, I would like to thank my family for their selfless dedication. With their financial and mental support, I can focus on my academic study. Secondly, I would like to sincerely thank the Auckland University of Technology (AUT) for providing a great study environment and various study resources.

I would like to give my deepest thanks to my supervisor Wei Qi Yan. He provided me with a lot of theoretical and technical supports throughout the entire project. He also taught me how to think and study effectively and make a rational plan for the thesis. I believe I could not complete this thesis without Dr. Yan's supervision.

Changjian Li

Auckland, New Zealand

November 2021

Chapter 1
Introduction

This chapter will introduce the background information relevant to our project, which is special character detection and recognition. We will present our research questions, contributions, and the structure of this report.

1.1 Background and Motivation

From ancient times to the present, the records of what happened can be passed on to next generations, allow people to learn the experience from the past which is very important for the development of human being. Text is one of the critical media of human civilization, which is a very significant invention. It varies the methods of people delivering and exchanging information, the communications between individuals become much effective. Humans take advantage of images, murals, and symbols to describe surroundings and events in the early period. In modern times, each nation has its own official language and text. There are unique texts and symbols in special scenes employed by exceptional people, or traditional symbols and characters are still used by small crowds.

When we meet a brand-new type of language or text, translating the characters into known text relying human labour is very cumbersome, time-consuming. Using computers and deep learning for detecting and recognizing characters permits the translation process to become more efficient, and less knowledge is needed from readers (Popel et al., 2020). The motivation of this research project is to implement a type of translator by using deep learning. The special characters of braille and Māori symbols are translated into plain English, which assists people in understanding the meaning of these special characters.

Māori symbols are created by Māori tribes of the south pacific region, who are the indigenous Polynesian of New Zealand. Pre-colonial Māori had no official language. Therefore, the traditions and knowledge were recorded and passed through generations by telling descriptively and drawing. Māori symbols influence art forms in New Zealand, especially famous haka, wood carvings, pounamu carvings, and tattoos. Māori symbolism represents the culture, history, and their belief visually. The symbols recorded the past and future. The symbols are constructed by spirals, curves, supernatural deities, and natural images. For example, “Kuro” represents the silver fern

native of New Zealand, which shows in many scenarios such as large artworks and wood carving (Mead, 2016).

Braille is a formed system designed for visually impaired people to gain information through touch reading and writing. Braille system generally applies 64 kinds of braille characters for users to edit information. A braille character is represented in a braille cell, which has six flatten or raised dots, organized in the shape of a 3×2 matrix. There are two general levels of braille characters. In the first level, one character will only represent one English alphabet, number, or punctuation. And the second level, a braille character, can define a prefix, suffix, or even a word (Weygand, 2020).

In this research project, we would like to design a model using deep learning methods, which translate the braille characters into plain English and define Māori symbols into their corresponding meanings. We understand these special characters and symbols more efficiently through interpreters, with less time and labor cost for translating manually.

1.2 Research Questions

The primary purpose of this project is to detect and recognize special characters/symbols by implementing deep learning methods. Thus, the research questions of the thesis will be:

(1) Are the novel deep learning methods suitable for detecting the characters and symbols? And which should be chosen for running the experiment.

(2) Are there any modification or upgrade need to be made for the methods or the models?

(3) Can deep learning methods solve the problem with high accuracy and minor error or misjudgment?

There are various types of deep learning methods for detection tasks. We need to

select the appropriate methods for our project. Therefore, we will go through the methods in the literature review. Regarding the methods we choose, the evaluations are feasible for detecting the braille and Māori symbols, also prove how robust the methods can be when inferencing the targets.

1.3 Contributions

The focus of this project is on special character detection and recognition by using deep learning methods. We collected general braille characters and five types of Māori symbols for our datasets. We review various famous methods of deep learning, including different types of Convolutional Neural Network (CNN) models, object detection methods of Fast R-CNN series and YOLO series. We choose ABCNet for braille detection and recognition in natural scenes, YOLOv5 for Māori symbols detection. We go through the details of methods and the model structures. The modifications are made for ABCNet, and the network can implement parameterized Bezier curves to draw the prediction bounding boxes while inferencing videos and real-time detection. In this project, our contributions are:

(1) Solving the special character recognition problem by using two deep learning methods, ABCNet and YOLOv5.

(2) Two corresponding datasets are created for the project.

(3) Modifying the inference method of ABCNet, the network can draw curved prediction box for arbitrary-shape braille text when the inference resource is video or real-time webcam.

(4) We train and evaluate the methods using the datasets and prove the robustness of the methods and capability of the networks for detection and recognition.

1.4 Objectives of This Report

In this thesis, our objective is to solve the problem of translating special characters. We find feasible deep learning methods as the solutions. While the methods are selected, we prepare for the relevant data and train the network properly. After the training process is

accomplished, the networks need to be evaluated through the data not included in the dataset. We must check whether the networks can infer the targets if the input resource is the image, video, and real-time webcam. The real-time detection is what we desire that the networks can achieve, which showcases that the networks have the ability to apply mobile detection.

1.5 Structure of This Report

The structure of the thesis is described as follows:

- In Chapter 2, we will expound on the related deep learning methods, the various types of CNN models are mentioned. We will also introduce R-CNN and YOLO's object detection methods and discuss the advantages and disadvantages of the methods.
- In Chapter 3, the deep learning methods are chosen to run the experiment of the project. The details of the methods, including the network structures, data preprocessing methods, and the loss function are explained in the methods. We describe the details of the datasets for the training and validation processes. Also, the modifications to the models will be mentioned in this chapter
- In Chapter 4, we will show the results of our experiment. The evaluation metrics, time consumption, loss curves will be mentioned. The modification and inference results will be represented.
- In Chapter 5, we will discuss the performance of the network based on the results in Chapter 4. The unsatisfactory results will also be shown, and the reasons will be discussed. After analyzing the results of the modes, we will examine the limitations of our work at the end of this chapter.
- In Chapter 6, we draw our conclusion for our project, then envision the possible improvements and operations in the future based on the conclusion and the previous work.

Chapter 2
Literature Review

This chapter focuses on the literature of machine learning and deep learning methods relevant to this project, including convolutional neural networks and object detection methods.

2.1 Introduction

Computer vision is one of the essential fields in deep learning. It plays a vital role in image searching, instant translation, self-driving vehicles, robot application, and navigation, etc. such as identifying people, objects, car license plates in images, videos, or live cams (Khalil & Mouftah, 2021; Vaswani et al., 2018; Lin et al., 2015). Visual object detection and instance translation using deep learning can support us to understand special characters and symbols. This project will implement special character recognition based on object detection/recognition techniques and natural scenes text recognition. This chapter will introduce the relevant fundamental concepts and knowledge, including Convolutional Neural Network, object detection/recognition, Recurrent Neural Network, and Scene Text Recognition (STR).

2.2 Convolutional Neural Network

2.2.1 Simulate Human Brain Functionality

The concept of neural network was first introduced as a biological aspect, and the neural networks in artificial intelligence mimic the constructures of biological neurons. Related work that influenced the origin of the Convolutional Neural Network (CNN) is relevant to the visual cortex. One of the notable achievements was the project made in 1968 (Hubel & Wiesel, 1968). They anesthetized a cat, connected its optic nerves to an oscilloscope using electrodes, showed different images to the cat, and observed the brain waves (Hubel & Wiesel, 1962). It is shown:

- The lateral geniculate nucleus (LGN) responds to tiny spots of light but not diffuse reflection.

- When the visual cortexes receive information from LGN, they no longer respond to aperture but the bright and dark lines.

- A simple type of cortical cells in the visual cortex only responds when specific areas are stimulated at particular angles.

- Other cells (including complex cortical cells) will only respond to the light in specific directions but not sensitive to area changes, and the lights can move on the screen without causing cell inactivation. This can be explained as a group of simple cortical cells converging on complex cells.

According to the phenomena, in each step, the inputs of interneurons are converged and give one output. The information is selectively erased in each step. Thus, the brain can act like a filter on different levels and distinguish some complex features. The first concept is the multilayer structure. The experiment shows that the retina and LGN constitute the first layer for receiving the visual signal and making preliminary processing in response to light spots. Then, the simple cells start responding to the lines, and more complex cells do further processes. The second concept is filtering, that different cells respond to specific inputs and filter out other information. The third one is local connectivity, which means each neuron will not respond to the whole image. It only focuses on the area that next to it. The last concept is translation invariance.

In 1980, Fukushima et al. (1982) introduced a pattern recognition mechanism called Neocognitron to help people understand how our brain works by building a pattern recognition network simulating the human brain. This model has been regarded as the prototypical model of Convolutional Neural Network (CNN). Two types of cells were designed based on a biological neural network called simple cells (S-cells) and complex cells (C-cells), the cells will be grouped as planes, and a layer inside the network is composited by the planes. Neocongnitron contributed to simulating the structure of neuroscience using the computer, with the step-by-step filter utilized CNN nowadays, take advantage of ReLU as a nonlinear function, average pooling for downsampling, guarantee the translation invariance of the network and sparse interaction (Schmidhuber, 2015).

Although Neocognitron is very fancy, it is built based on unsupervised learning WTA (Winner Take All), which has limited practicability. From 1989 to 1990, LeCun et al. (1989) applied backpropagation to supervised learning in deep networks, similar to Neocognitron. The difference is when generating each feature map, only a single neuron is applied to every receptive field (Gilbert & Wiesel, 1992; Luo et al., 2016). The entire convolution operation is equivalent to scanning the input with a small convolution kernel and then following with a squashing function. This operation permitted weight sharing, which reduced the number of free variables, the risk of overfitting, and improved generalization ability. It speeds up the training process while reducing the parameters.

The model cresceptron (Weng et al. 1992) contributed two tricks that were designed for training models. The first trick is data augmentation, which applies multiple transformation methods to original training data, including translation, rotation, rescaling, etc. This trick augments the size of the training dataset and improves the robustness of the algorithms, reducing the risk of overfitting. The second trick is max pooling (Weng et al., 1992; Nagi et al., 2011).

2.2.2 Modern CNN

LeCun et al. (1998) suggested the first complete modern CNN called LeNet-5. The structure of LeNet-5 contains all types of essential layers compared to previous models. The network layers have been deepened to 7 layers, with two layers of convolutional layers and pooling layers. Between the second layer (subsampling layer S2) and the third layer (convolutional layer C3), the number of feature maps is increased from 6 to 16. Instead of applying a full connection between S2 and C3 layer, different feature maps will have 3 to 5 different S2 outputs (in different ranges) as inputs. This gives two benefits: Reducing the connection numbers by not using the full connection and breaking down the symmetry between different feature maps, which is conducive to the robustness of its representation. LeNet-5 takes use of Tanh as its activation function,

which is shown as eq.(2.1) :

$$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{2.1}$$

where a symmetric activation function triggers faster converge. ReLU has replaced in LeNet-5 because the model was not too deep, the problem of gradient vanishing can be ignored, the nature of accelerating convergence is much critical to be considered. Another alternation is at the output layer, RBF layer replaced the original fully connected layer as shown in eq.(2.2)

$$y_i = \sum_j (x_j - w_{ij})^2 \tag{2.2}$$

where y_i is regarded as a penalty term. From the perspective of probability theory, the output of RBF is thought as a Gaussian distribution that has not been regularized with negative log-likelihood. The benefit of using RBF is that the user can set the parameters of RBF (Er et al., 2002).

In 2012, the emergence of AlexNet marked the rise of deep learning. The structure of AlexNet has eight layers, including five convolutional layers and three pooling layers. AlexNet is similar to LeNet-5, but more convolutional layers and larger parameter space are exploited for fitting dataset ImageNet, which is a boundary between shallow neural network and deep neural network (Krizhevsky et al., 2012). After each convolutional layer, there is a ReLU function as shown in eq.(2.3),

$$ReLU(x) = \max(0, x). \tag{2.3}$$

If the input is a positive number, gradient saturation will not occur, solving the problem of vanishing gradient, which Sigmoid cannot handle, giving faster convergence. But there are some disadvantages of the ReLU function. One typical issue is when the input is negative, the gradient will be 0 when proceeding with backpropagation, and this is called Dead ReLU (Lu et al., 2019). Data augmentation was also applied while building AlexNet. The technique Dropout is used to selectively ignore the nodes in the backpropagation process during training, preventing overfitting (Gal & Ghahramani,

2016).

In 2015, the proposal of ResNet was a milestone in the history of CNN, which won the champion of ILSVRC 2015 with 3.6% error rate. Relying upon the experience, the depth of a neural network is highly relevant to its performance. Theoretically, when the layers are added into a network, it can execute more complex feature extraction, better results can be achieved when the model is deeper. Unfortunately, when more layers are added to the network, the accuracy appears to be saturated or even decreased.

Given a hypothesis that a shallow neural network has reached its best configuration, we add extra layers to form a deep neural network. The parameters inside the extra layers are set to be always identically equal. The extra layers will not affect input and output, which gives the deeper network the same error rate. But during training, the parameters in additional layers cannot be identically equal, which conducts a worse performance called degradation. The experimental result (He et al. 2016) shows that the 56-layer network has a higher error rate in training and testing than the 20-layer network. It is not a problem of overfitting because both training error and testing error of 56-layer network are higher. What happens is called vanishing/exploding gradients. The loss function is shown as eq. (2.4):

$$Loss = F(X, W) \tag{2.4}$$

$$\frac{\partial Loss}{\partial X} = \frac{\partial F(X,W)}{\partial X} \tag{2.5}$$

$$Loss = F_n(X_n, W_n), L_n = F_{n-1}(X_{n-1}, W_{n-1}), \dots L_2 = F_1(X_1, W_1) \tag{2.6}$$

$$\frac{\partial Loss}{\partial X_i} = \frac{\partial F_n(X_n,W_n)}{\partial X_n} * \dots * \frac{\partial F(X_{i+1},W_{i+1})}{\partial X_i} \tag{2.7}$$

Its gradient after backpropagation is calculated by eq. (2.5). Extending to multi-layer neural networks with the same principle, the loss function can be represented as eq. (2.6), where *n* represents the number of layers. According to the chain rule, the gradient in layer *i* can be calculated by using eq. (2.7) (Hecht-Nielsen, 1992).

The initial value of W usually is set as 0. If $W_1 \times W_2 \times ... \times W_n$ is getting smaller, it generates a slower update of W, which is called vanishing gradient problem. If W in each layer has an enormous value which is bigger than 1.0, it will induce gradient exploding. The problem of gradient vanishing and exploding problems are remedied using methods such as batch normalization, which allows the network depth to become 10 times deeper with less increased error rate. However, the degradation problem might occur.

ResNet has the residual blocks for solving the issues if the network gets deeper. The residual function $H(X) = F(X) + X$ replaces the standard output, which means eq. (2.7) has changed into eq. (2.8):

$$\frac{\partial X_{i+1}}{\partial X_i} = \frac{\partial X_i + \partial F(X_i, W_i)}{\partial X_i} = 1 + \frac{\partial F(X_i, W_i)}{\partial X_i} \tag{2.8}$$

The "shortcut connections" does not introduce additional parameters and computation under identity mapping. A residual block has two layers, where X represents the input of the residual block, $F(x)$ is the output after the first layer of linear change and activation function. Before the second activation function, $F(x)$ is added with input X. For redundancy layers that can satisfy identity mapping, we only need $F(x) = 0$, which is easier than $H(x) = x$ (Naranjo-Alcazar et al., 2019).

2.3 Object detection

The watershed of the development process of object detection is separated into conventional object detection and object detection based on deep learning. Girshick et al. introduced R-CNN, proposed the first object detection method using deep learning techniques. In modern times, object detection methods are grouped into two categories, two-stage detection and one-stage detection. Two-stage detection defines the detection as "from coarse to fine", and one-stage detection describes it as "one step in place" (Wang et al., 2013; Zou et al., 2019).

2.3.1 R-CNN

R-CNN and CNN methods are applied to object detection, which are the main research ideas in visual object detection. Compared to traditional methods such as Viola-Jones face detection using a sliding window to determine all possible areas, R-CNN operates Selective Search to pre-extract regions that are more likely to be visual objects and only extract features from these regions using CNN (Viola & Jones, 2001; Uijlings et al., 2013). R-CNN includes four steps: Generating regions, feature extraction, classification, and localization refinement (Girshick et al., 2014).

In the first step, the network receives an image as input by using the Selective Search method to generate 1,000~2,000 candidate regions called Region Proposal/Region of Interest (RoI). The regions are in rectangular shapes with different sizes and ratios. The regions will be reshaped for fitting CNN, two methods are considered: Anisotropic scaling and isotropic scaling. In feature extraction, R-CNN takes advantage of AlexNet as its backbone. The AlexNet will be pre-trained into a classifier that can classify 1,000 categories. The pre-trained CNN will be finetuned for classifying 21 classes (20 types of objects + background).

Regarding pattern classification, R-CNN refers to SVMs as its classifiers (21 SVMs included) (Cortes & Vapnik, 1995). The output features from CNN will be graded by SVMs, then through Non-maximum Suppression (NMS) for eliminating overlapped regions (Bodla et al., 2017). For localization, each class will be refined with its bounding box by using a linear regressor. Because R-CNN must go through thousands of regions, the computation cost is prohibitively expensive and time-consuming.

2.3.2 Fast R-CNN & Faster R-CNN

To reduce the computational time, Girshick et al. (2015) came up with a new plan and proposed Fast R-CNN. Instead of extracting features from each region, Fast R-CNN adopts the whole image as the input of CNN. Fast R-CNN achieves most end-to-end

training processes (except RoI). The features are temporarily stored inside graphic memory, do not need extra storage. SVM classification and bounding box regressor are combined with CNN while training. Two layers replace the softmax layer. One responds softmax for regions classification (including backgrounds), another is for finetuning the bounding boxes (Grave et al., 2017).

With regard to feature extraction, the procedures such as convolution, pooling, and activation function (ReLU) do not require fixed-size inputs. RoI pooling layer is added to the network. It maps inputs in different sizes onto fixed-scale vectors. RoI Pooling divides the regions evenly into $M \times N$ blocks, then performs max pooling to each block. The regions are transformed into the same size and ready to be passed through to the next layer. Although inputs in different sizes will generate different sizes of feature maps, the RoI pooling layer extracts a fixed-dimensional feature representation for every region, which feature maps can go through softmax.

Fast R-CNN still applies selective search to generate RoI, which is time-consuming to provide every RoI. Therefore, in Faster R-CNN, a neural network is proposed for extracting edges, which means, generating RoI, feature extraction, classification, and localization are all unified into a deep neural network (Ren et al., 2015). The structure of Faster R-CNN is regarded as a Region Proposal Network (RPN) with Fast R-CNN. The RPN is employed to replace the selective search method (Li et al., 2018).

2.3.3 YOLO

The R-CNN series have high accuracies in detection tasks. However, due to characteristic of the two-stage network structure, the detection speed cannot meet the real-time requirement. It is indispensable to design a faster network for solving real-time tasks.

A one-stage detection method called YOLO (You Only Look Once) has been proposed (Redmon et al., 2016). It has a faster detection speed of 45 FPS, published in CVPR 2016, and has attracted wide attention. The main idea of YOLO is to turn

detection tasks into a regression problem. A whole image is the input of the network, getting the location and classes of the bounding box. In Faster R-CNN, an RPN structure is employed for getting RoI of the targets, this method has high accuracy, but the extra effort is needed to train the RPN. YOLO splits the image into 7×7 grids, these grids are regarded as the RoI, and RPN is not required. Each grid is responsible for predicting multiple bounding boxes with the confidence value. The corresponding data of a bounding box contains the information of its location and the confidence score. The data is shown as

$$\mathbf{V} = (x, y, w, h, c) \tag{2.9}$$

where (x, y) represents the coordinates of the central point of the box, w shows the width, h is the height of the box, c denotes the confidence score.

The first version of YOLO, namely, YOLOv1 improves the speed of object detection. It permits the pipeline of the network to become straightforward. But it has a disadvantage which cannot proceed with multitarget detection properly. Also, the accuracy of localization is poor, the recall of the method is low. Therefore, YOLO9000 is introduced to eliminate the problem in the previous version of YOLO (Redmon & Farhadi, 2017). More versions of the YOLO series were introduced for visual object detection. The novel detection method YOLOv5 gains very efficient inference speed while ensuring the accuracy of detection. This method is one of the candidates' deep learning methods to be chosen for running this project's experiment (Jocher et al., 2020).

Chapter 3
Methodology

The main content of this chapter is to clearly demonstrate the deep learning that we select for this project based on the reviewed methods. We articulate the details of ABCNet and YOLOv5, which will be implemented for the experiments of this thesis.

3.1 ABCNet For Braille Recognition

We execute E2E (end-to-end) scene text recognition (spotting) for braille characters. ABCNet as a novel solution is regarded as a relatively suitable method (Wang et al., 2011; Li et al., 2017). ABCNet (Liu et al. 2020) is an E2E model for text spotting. The structure of ABCNet has three arts, including an FPN-ResNet-based detection framework, Bezier Align, and a lightweight recognition branch. ABCNet is the first method that implements parameterized Bezier curve for scene text detection and recognition. It detects arbitrary-shape text in natural scenes adaptively, feasible for curve text detection.

3.1.1 FPN-ResNet Detection Branch

Inspired by FCOS, ABCNet takes use of the structure of FPN with ResNet as its anchor-free detection branch (Tian et al., 2019; He et al., 2019). The popular object detection methods, including R-CNN series models, SSD, YOLO series, and anchor-based detection methods, implement pre-set bounding boxes and scan the entire image to search targets (Liu et al., 2016). These methods have the following disadvantages:

- The bounding boxes are in various sizes, ratios, and usage counts. These hyperparameters have outstanding outcomes based on the detection results.
- The bounding boxes need to be adjusted during different tasks because the sizes of the targets in each task are different, and small targets are very hard to detect.
- To improve the score of recall, the network will generate a huge number of bounding boxes, most of the boxes are negative samples, which lead to the unbalance between positive samples and negative samples, affect the final results.

The anchor-free methods do not implement pre-defined bounding boxes for the tasks by applying regression methods to generate the boxes (Zhu et al., 2019). YOLOv1 is

one of the famous methods of anchor-free detection. ABCNet takes an example by using the architecture of FCOS and FPN with ResNet as its backbone and detection branch to detect different sizes of features on different feature maps.

In YOLO, if a grid contains the central point of the target, then it is marked as a positive sample. Otherwise, it will be labeled as negative samples. This method brings a lousy impact on recall estimation. The networks of ABCNet and FCOS are treated as a position(pixel) as the positive sample as long as it is within the ground truth bounding box (Long et al., 2015). Instead of only focusing on how to predict the central point of a target, the network is learning how to predict the distance to the top, bottom, left, and right (t, b, l, r) of the ground truth bounding box of each position.

We assume F_i is the *i-th* feature map on the backbone, s represents the stride of the feature map. The ground truth of bounding boxes of the input image is defined as $B_i = (x_0^i, y_0^i, x_1^i, y_1^i, c^i)$, x_0 and y_0 represent the top-left coordinates of the bounding box, x_1 and y_1 display the bottom-right coordinates of the bounding box, c shows the class of the target. Each pixel (x, y) on the feature map F_i is mapped to the original input image as eq. (3.1)

$$\boldsymbol{p} = (\left\lfloor \frac{s}{2} \right\rfloor + xs, \left\lfloor \frac{s}{2} \right\rfloor + ys) \tag{3.1}$$

Regression operation is applied to each position, if a position (x, y) is within the ground truth, then we regard it as the positive sample c^*, it will be regarded as a negative sample $(c^* = 0)$, the network can make full use of foreground samples by this method. The network defines a 4D vector t^* as the regression target

$$t^* = (left^*, top^*, right^*, bottom^*) \tag{3.2}$$

where the 4-tuple t^* represents the distances between the position and the four boundaries of the ground truth. Apparently, the values are non-negative and need an exponential function to give correct outputs of these four values, as shown in eq. (3.3)

$$left^* = x - x_0^{(i)}, top^* = y - y_0^{(i)}, right^* = x_1^{(i)} - x,\ bottom^* = y_1^{(i)} - y \tag{3.3}$$

FPN structure in ABCNet backbone is employed for dealing with the issues while detecting overlapping objects. The FPN lets the network make predictions on various sizes of feature maps, feature maps in different sizes are offered to predict objects in different scales (Lin et al., 2017). For example, a feature map with *stride*=8 is accommodated for predicting small objects, *stride*=16 for the visual objects in medium sizes, *stride*=32 for big objects. In ABCNet, visual features are extracted on three scales, as $\frac{1}{4}$, $\frac{1}{16}$, and $\frac{1}{32}$

There are five feature maps implemented $[p_3, p_4, p_5, p_6, p_7]$, with six thresholds: $[m_2, m_3, m_4, m_5, m_6, m_7] = [0,64,128,256,512,\infty]$. The feature map p_i corresponds to interval (m_{i-1}, m_i). For the employment of the thresholds, the network will traverse all positions on every feature map. For position (x, y) on the feature map p_i, firstly, we calculate the values of (l^*, t^*, r^*, b^*), the maximum value $m = max(l^*, t^*, r^*, b^*)$, then we estimate whether m is within the threshold, which means if it satisfies $m_{i-1} < m < m_i$, the maximum value m will be grouped into either positive or negative samples (Kannadaguli, 2020).

Regarding objects overlapping during object detecting, while performing the detection tasks, if there are two bounding boxes of different sizes which overlap with each other, the maximum value M of all m calculated in the smaller box will not exceed the boundaries of the box, which means $M \leq \max(b_h, b_w)$. Therefore, the pixel positions in the small box will be mostly mapped to a smaller threshold, which permits the positions relate to a feature map with a smaller stride. Similarly, most of the positive samples that in the large bounding box will be mapped to a larger threshold. By the solution above, the backbone network of ABCNet can perform multiscale prediction and solve the problem of objects overlapping.

The network suppresses the low-quality detection of bounding boxes that are far away from the center of the target (Lee & Park, 2020). There is still a gap between anchor-based methods and anchor-free methods after applying multiscale detection. The reason is that object detection is expanded to the entire ground truth boxes instead of the

target center, which might generate a number of prediction boxes that the centers are far away from the target center. Therefore, the method is introduced to the network by using Binary Cross-Entropy (BCE) for optimization to constrain the number of generated boxes (De Boer, 2005). The equation of Center-ness is shown as eq. (3.4):

$$centerness^* = \sqrt{\frac{\min(l^*, r^*)}{\max(l^*, r^*)} + \frac{\min(t^*, b^*)}{\max(t^*, b^*)}} \tag{3.4}$$

where the value of Center-ness is within the interval [0,1].

3.1.2 Bezier Curve Detection

The most novel feature of ABCNet is to execute Bezier curves and form the border of the bounding boxes of text. Instead of the rectangular boxes applied to current Scene Text Recognition (STR) methods such as CharNet, FOTS, CRAFT, and Attention OCR (Liu et al., 2018; Liu et al., 2018; Baek et al., 2019; Zhang et al., 2019). This permits that the network is able to predict arbitrary-shaped text in natural scenes.

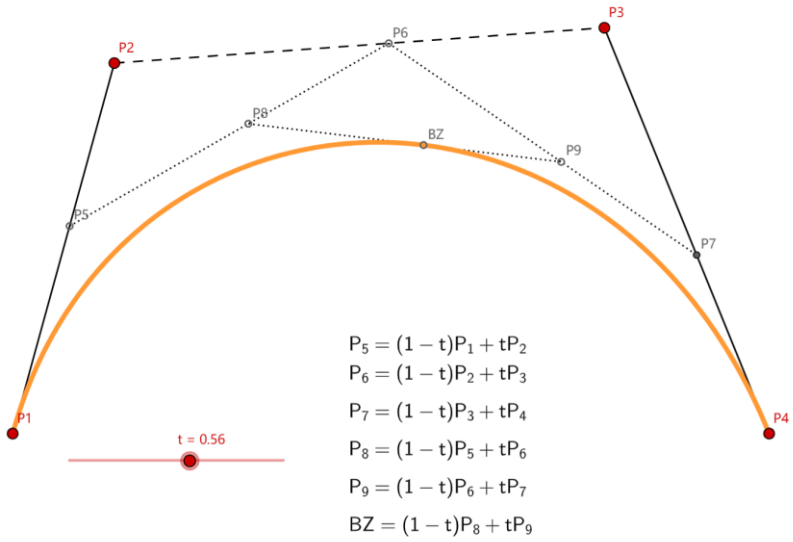

Figure 3.1: An example of Bezier curve

The Bezier curve was named in 1972 after the French engineer Pierre Bezier (Hazewinkel, 1997). It has been implemented in various image processing fields, including graphic design, animation, trajectory calculation, etc. For example, computer-

aided design software includes the function of drawing curves by using Bezier curve method. The design of fonts adopted in modern computers is also controlled by piece-wise Bezier curves. Compared with other spline-based methods, the Bezier curve is a type of curve represented by connected vectors, shown in Figure 3.1 (Choi et al., 2008).

Bezier curve is interactive, the shape of the curve is modified by changing the vectors. While drawing the curve, a polygon is shown through multiple vectors in advance to represent the trend and direction of the curve, like the initial outlines. A Bezier curve is defined as the product of vectors and a primary function, as shown in eq. (3.5)

$$V(t) = \sum_{i=0}^{n} f_{i,n}(t) A_i \tag{3.5}$$

where A_i represents vector, $f_{i,n}$ shows a primary function, which is expressed by using eq.(3.6)

$$f_{i,n}(t) = \begin{cases} 1 & i = 0 \\ \frac{(-t)^i}{(i-1)!} \frac{d^{i-1}}{dt^{i-1}} \frac{(1-t)^{n-1}-1}{t} \end{cases} \tag{3.6}$$

This primary function is equivalent to a polynomial of $n-1$ for the parameter $t \in [0,1]$. Through the equation, we calculate the curve by using the connected vectors. A Bezier curve is defined based on a set of points by Bernstein polynomials (Lorentz, 2013; Oruç, 2003). For the vectors to generate the curve, regarding control points, which are presented as control points $P_0, P_1, P_2 \ldots P_n$, we define a control point $P(t)$ as shown in eq. (3.7)

$$P(t) = \sum_{i=0}^{n} P_i B_i^n(t), \tag{3.7}$$

where $B_i^n(t)$ represents the Bernstein polynomial of the i-th control points:

$$B_i^n(t) = C_n^i t^i (1-t)^{n-i} \tag{3.8}$$

The C_n^i represents the combinatorial number,

$$C_n^i = \frac{n!}{i!(n-i)!} \tag{3.9}$$

ABCNet implements cubic Bezier curves for generating the bounding boxes during prediction, which is flexible enough to sufficiently describe most of the arbitrary-shaped scene text as shown in eq. (3.10)

$$B(t) = P_0(1-t)^3 + 3P_1t(1-t)^2 + 3P_2t^2(1-t) + P_3t^3 \tag{3.10}$$

The cubic Bezier curve is responsible for generating upper and bottom curve boundaries of the bounding boxes, other two are straight lines. Each curve has four control points and eight control points in total for a text box. Based on the cubic Bezier curve, the task of finding the arbitrary-shaped scene text is simplified into the task of finding a bounding box that contains eight control points. Since the detection branch of ABCNet only needs to predict a few more coordinates than other networks without adding additional layers, the overall inference speed is not significantly affected.

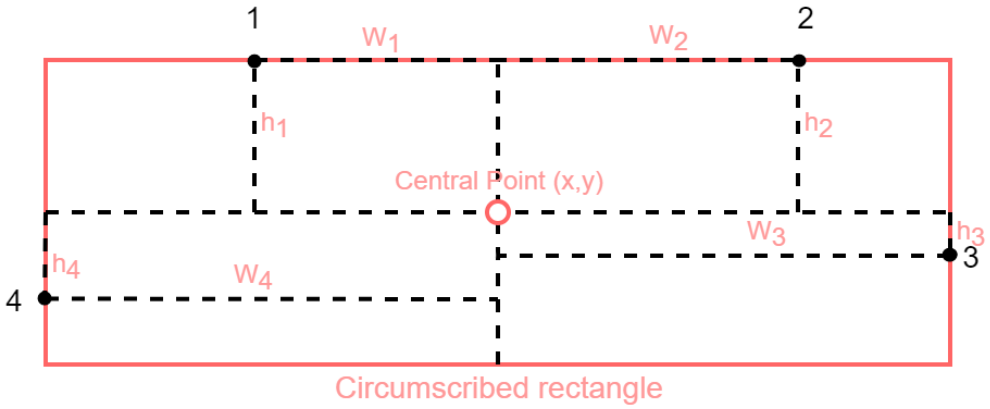

Figure 3.2: The four control points relate to one Bezier curve

Regarding the network to learn the coordinates of the control points, a regression method is utilized for regressing the targets based on the ground truth information. For one Bezier curve in a bounding box, the coordinates of four control points are reformatted into $(x, y, w_1, h_1, w_2, h_2, w_3, h_3, w_4, h_4)$, where (x, y) represents the

coordinates of the central point relevant to the minimum circumscribed horizontal rectangle, w_i and h_i show the relative coordinates of i-th control point with respect to the central point. The coordinates of four control points are described as $(x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4) = (x + w_1, y + h_1, x + w_2, y + h_2, x + w_3, y + h_3, x + w_4, y + h_4)$, w_i and h_i are negative. The predicted coordinates P^* are expressed as eq.(3.11).

$$P^* = (p_x^*, p_y^*, p_{w1}^*, p_{h1}^*, p_{w2}^*, p_{h2}^*, p_{w3}^*, p_{h3}^*, p_{w4}^*, p_{h4}^*) \tag{3.11}$$

The coordinates of ground truth are

$$P = (p_x, p_y, p_{w1}, p_{h1}, p_{w2}, p_{h2}, p_{w3}, p_{h3}, p_{w4}, p_{h4}) \tag{3.12}$$

Based on the information of the given coordinates, we calculate the minima X_{min} and maxima X_{max} of the circumscribed rectangle, its width $w_{chr} = X_{max} - X_{min}$and its height $h_{chr} = Y_{max} - Y_{min}$. The parameterizations of the coordinates are adopted as eq. (3.13-3.17):

$$d_X = \frac{p_X^* - p_X}{w_{chr}}, d_Y = \frac{p_Y^* - p_Y}{h_{chr}} \tag{3.13}$$

$$d_{w1} = \frac{p_{w1}^* - p_{w1}}{w_{chr}}, d_{h1} = \frac{p_{h1}^* - p_{h1}}{h_{chr}} \tag{3.14}$$

$$d_{w2} = \frac{p_{w2}^* - p_{w2}}{w_{chr}}, d_{h2} = \frac{p_{h2}^* - p_{h2}}{h_{chr}} \tag{3.15}$$

$$d_{w3} = \frac{p_{w3}^* - p_{w3}}{w_{chr}}, d_{h3} = \frac{p_{h3}^* - p_{h3}}{h_{chr}} \tag{3.16}$$

$$d_{w4} = \frac{p_{w4}^* - p_{w4}}{w_{chr}}, d_{h4} = \frac{p_{h4}^* - p_{h4}}{h_{chr}} \tag{3.17}$$

For each control point, the coordinates are calculated as eq. (3.18)

$$\Delta x = P_{ix} - x_{min}, \Delta y = P_{iy} - y_{min} \tag{3.18}$$

where x_{min} and y_{min} are representing the minimum value of x and y of the four vertices. There are only one extra convolutional layer and sixteen outputted channels

required for learning the values of Δx and Δy, which costs negligible computation overhead (Liu & Jin, 2017).

3.1.3 Bezier Ground Truth

The Bezier ground truth needs to be labeled based on the polygon annotations, such as dataset CTW1500 and Total-Text (Liu et al., 2017; Ch'ng & Chan, 2017). The general ground truth and annotations are presented in polygon shapes. The network has to generate the control points of Bezier curves for the bounding boxes based on the polygon ground truth for training.

Figure 3.3: Comparison of polygon ground truth and Bezier ground truth

Figure 3.3 shows the transformation from general polygon ground truth into Bezier ground truth. On the left side of the image, the polygon annotations are with 14 annotation points in total. Taken the upper curved boundary as an example, the annotated points of polygon ground truth are $[p_0, p_1, p_2, p_3, p_4, p_5, p_6]$. The control points of the Bezier curve are $[b_0, b_1, b_2, b_3]$, which are the values we need to calculate. We wish the generated Bezier curve could pass through these annotated points p_i as many as possible, which means all the p_i could meet the parameterized Bezier curve equation, and the coordinates can satisfy the matrix in the equation below:

$$\begin{bmatrix} B_{0,3(t_0)} & \cdots & B_{3,3(t_0)} \\ B_{0,3(t_1)} & \cdots & B_{3,3(t_1)} \\ \vdots & \ddots & \vdots \\ B_{0,3(t_m)} & \cdots & B_{3,3(t_m)} \end{bmatrix} \begin{bmatrix} b_{x_1} & b_{y_1} \\ b_{x_2} & b_{y_2} \\ b_{x_3} & b_{y_3} \\ b_{x_4} & b_{y_4} \end{bmatrix} = \begin{bmatrix} p_{x_1} & p_{y_1} \\ p_{x_2} & p_{y_2} \\ \vdots & \vdots \\ p_{x_m} & p_{y_m} \end{bmatrix} \tag{3.19}$$

where t is calculated using the ratio between the cumulative length of polyline segments and the perimeters of the polylines of seven annotated points based on the curved boundary. The number of b_i usually less than p_i, we find the best values of each b_i and the Bezier curve using the standard least square method. The result of the Bezier ground truth is shown on the right side of Figure 3.3. Compared to the original polygon-shaped ground truth, the Bezier ground truth fits the actual text better.

3.1.4 BezierAlign

After generating the Bezier ground truth annotations, the alignment method called BezierAlign is implemented for feature alignment and sampling, which is an extended method from ROI Align. For the traditional ROI Pooling and ROI Align method, the sampling grids are usually rectangular-shaped or quadrilateral-shaped, which will contain too much background information into the sampling results and bring negative affect to the training process (He et al., 2017; Sun et al., 2018).

For ABCNet, which is handling the arbitrary-shaped scene text, the general alignment methods are not applicable. For BezierAlign, as a point at any position inside the feature map, we firstly calculate the ratio $t \in [0,1]$ between g_{iw} and w_{out} that is shown in Figure 3.4.

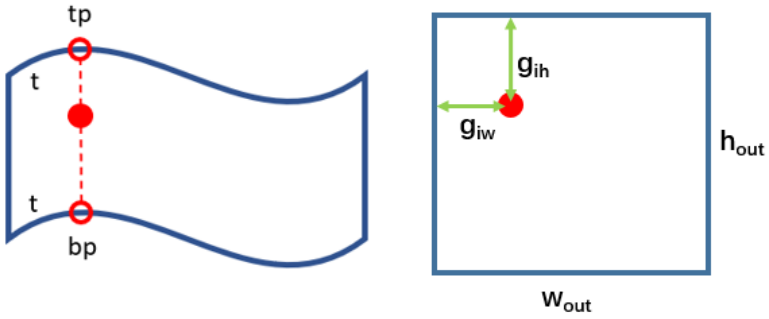

Figure 3.4: BezierAlign

$$t = \frac{g_{iw}}{w_{out}} \tag{3.20}$$

where g_{iw} represents the distance from the observed point to the left boundary of the

feature map, w_{out} shows the width of the feature map.

Once t is calculated, the point of the upper Bezier curve tp and lower Bezier curve bp are rendered by using the value of t and the Bezier equation. Based on the values of tp and bp, we get all the positions of the points on the line segment between tp and bp by applying linear interpolation

$$op = bp \cdot \frac{g_{ih}}{h_{out}} + tp \cdot (1 - \frac{g_{ih}}{h_{out}}). \quad (3.21)$$

3.1.5 Light Weight Recognition Head

Layers (CNN-RNN)	Parameters (kernel size, stride)	Output Size (n,c,h,w)
Conv ×4	(3, (1,1))	$(n, 256, h, w)$
Conv ×4	(3, (2,1))	$(n, 256, h, w)$
Average pooling	-	$(n, 256, 1, w)$
Channels-permute	-	$(w, n, 256)$
Bi-LSTM	-	$(w, n, 512)$
Fully connected layer	-	(w, n, n_{class})

Table 3.1: the structure of the recognition head of ABCNet

The recognition branch of ABCNet is a lightweight recognition head, including six convolutional layers, LSTM, CTC layer, and one fully connected layer as its main components (Shi et al., 2016; Hochreiter & Schimidhuber, 1997; Graves et al., 2006). For each convolutional layer, the padding is set to 1, n represents the size of each batch, c shows the size of the channel. The width and height of outputted feature map are shown by using w and h. The number of classes n_{class} are set to 97, which include lower and upper English alphabets, general symbols, and digits.

In Natural Language Processing (NLP), the task of sentiment classification is to

classify the emotional tendency of a given text, which is regarded as a class of classification tasks (Chowdhury, 2003). The general method for handling emotion classification is to express the word or phrase first, then combine the expression of words in the sentence by an appropriate combination method. Finally, the presentation of the sentence is employed to classify the sentiment.

Long short-term memory (LSTM) is a type of recurrent neural network (RNNs) (Mikolov et al., 2010). Due to its characteristics, it is very suitable for modeling time-series data, such as text (phrases, sentences, etc.). In order to form a phrase or a sentence, the easiest way is to add the words together by summing the expression of the words or taking their average value. Unfortunately, this method cannot be considered for the positions of words inside the phrase. For example, the phrase "I do not think this is good," where the word "not" is placed before the words "think this is good" and brings the sentiment into a derogatory term. By implementing LSTM, we capture long-distance dependencies because LSTM can selectively learn and forget the information through training.

The components of LSTM include input word X_t at time t, cell state C_t, hidden state h_t, temporal cell state $\tilde{C}_t$, forget gate f_t, input gate i_t, and output gate o_t. The process of LSTM emphasizes forgetting additional information of the cell and remembering new information, letting the essential and valuable messages can be passed down, and redundant data is abandoned, and the h_t will be given in each time step. The operation of forgetting, remembering, and outputting corresponds to the prior hidden state h_{t-1} with the present output X_t to calculate the values of f_t, i_t, and o_t. All three gates apply the Sigmoid function, where the activation function is Tanh.

The first step for LSTM is to decide what information should be abandoned selectively. Forget gate f_t is composed of the Sigmoid function layer. The inputs are h_{t-1} and X_t, the output of each node in cell C_{t-1} will be restricted in [0,1]. The output '1' represents fully reserved, '0' stands for forgetting the information. The forget stage is described by the following equation:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3.20}$$

The next step will decide what information will be retained in the neural cells, containing two parts. Firstly, there will be an input gate layer structured by sigmoid function responds to the value for altering (Sak et al., 2014). Then, a Tanh function layer will generate a new candidate value C_t, which will be added into the cell state. These two steps will be combined for altering the state value:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{3.22}$$

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{3.23}$$

Based on the steps above, we alter the prior cell state C_{t-1} into the new cell state C_t:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{3.24}$$

where $i_t * \tilde{C}_t$ represents the new candidate value. It is measured by how much we want to update the value of each state.

Finally, there is the output state. The output is based on the states of the cells, with a filter involved. The sigmoid layer correspondingly decides which part of the cell state need to be outputted (Malhotra et al., 2015), then take the cell state through the Tanh function layer, multiply it by using the output of the sigmoid threshold, to output the desired result:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{3.25}$$

$$h_t = o_t * \tanh(C_t) \tag{3.26}$$

The backpropagation (BP) of LSTM updates the parameters iteratively by using gradient descent, which is the same as the BP in RNN (Gonzalez & Yu, 2018). In RNN, we adopt a hidden state h_t and gradient $\delta^{(t)}$ for the BP. There are two h_t and C_t:

$$\delta_h^{(t)} = \frac{\partial L}{\partial h^t} \tag{3.27}$$

$$\delta_C^{(t)} = \frac{\partial L}{\partial C^t} \tag{3.28}$$

for a better derivation, the loss function $L(t)$ is split into two parts, which include loss $l(t)$ at time t, and loss $L(t+1)$ after time t, as shown in eq.(3.29)

$$L(t) = \begin{cases} l(t) + L(t+1), if\ t < \tau \\ l(t) \qquad\qquad\quad , if\ t = \tau \end{cases}. \tag{3.29}$$

The full BP equation of LSTM is

$$\frac{\partial L}{\partial W_f} = \sum_{t=1}^{\tau} [\delta_C^{(t)} \odot C^{(t-1)} \odot f^t \odot (1 - f^t)](h^{(t-1)})^T \tag{3.29}$$

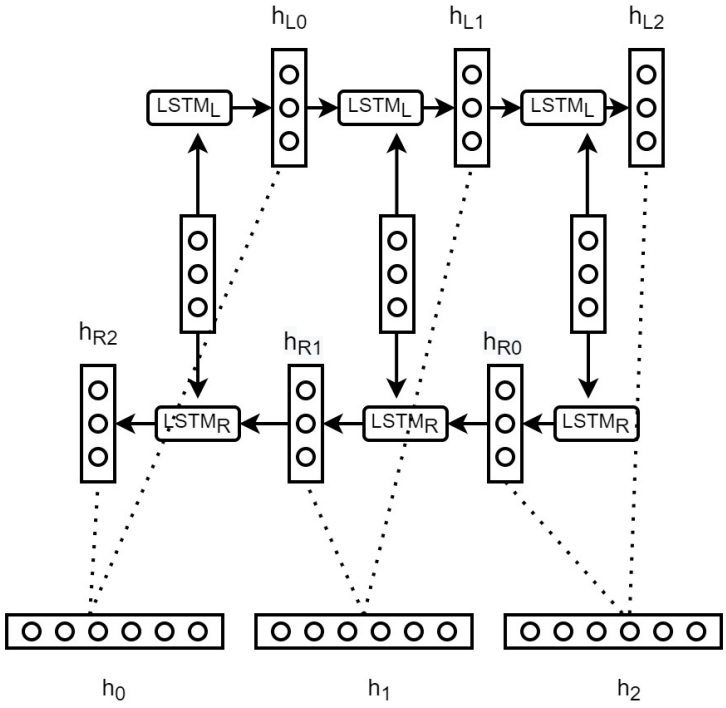

Figure 3.5: The structure of bi-directional LSTM

The bi-directional long short-term memory (Bi-LSTM) is the combination of forward LSTM and backward LSTM, which is applied to the recognition head of ABCNet (Huang et al., 2015). Like the example shown in Figure 3.5, the forward LSTM operation will generate three vectors h_{L0}, h_{L1}, h_{L2} in sequence. The backward LSTM will generate vectors h_{R0}, h_{R1}, h_{R2}. The vectors will be combined to form the result as $[h_{L0}, h_{R2}][h_{L1}, h_{R1}][h_{L2}, h_{R0}]$, which represented by the vectors h_0, h_1, h_2.

The method of calculating losses for ABCNet includes three parts: FCOS loss, Center-ness loss, and Bezier loss:

$$L_{total} = L_{FCOS} + L_{centerness} + L_{Bezier} \tag{3.30}$$

$$L_{FCOS} = L(\{p_{x,y}\},\{t_{x,y}\}) = \frac{1}{N_{pos}}\sum_{x,y} L_{cls}(p_{x,y}, c^*_{x,y}) + \frac{1}{N_{pos}}\sum_{x,y} 1_{\{c^*_X>0\}} L_{reg}(t_{x,y}, t^*_{x,y}) \tag{3.31}$$

$$L_{centerness}(x,y) = mean\{l_1,\cdots,l_n\}, \quad l_n = -[(y_n \cdot log\sigma(x_n) + (1-y_n)\cdot \log(1-\sigma(x_n))] \tag{3.32}$$

$$L_{Bezier}(x,y) = \frac{1}{n}\sum_i z_i, \quad z_i = \begin{cases} 0.5(x_n - y_n)^2, & if|x_i - y_i| < 1 \\ |x_i - y_i| - 0.5, & otherwise \end{cases} \tag{3.33}$$

The FCOS loss is constructed by two loss functions. The Focal loss as classification loss L_{cls} and IoU loss as localization loss L_{reg} (Lin et al., 2017; Yu et al., 2016). The center-ness loss is calculated using binary cross-entropy with logits, and the Bezier loss is computed by using smooth L1 loss (Mao et al., 2016; Pang et al., 2019).

3.2 YOLOv5 For Māori Symbols Recognition

Compared to Māori symbols, English braille characters are used similarly to the English alphabets and punctuations. Māori symbols are more likely to represent things and implied meanings. For example, the Koru represents spiral, meaning new beginnings and growth. The Hei Matau shows fishhook, meaning prosperity, safety and good health. Manaia displays a spirit creature, meaning supernatural and guardian spirit (Lambert, 2009). Rather than using the methods of natural scene text recognition, object detection methods might be more suitable for the Māori symbols recognition task. Therefore, YOLOv5 is applied as the detection method for Māori symbols recognition.

3.2.1 Input

The YOLOv5 has three types of processes to the input images, including Mosaic data augmentation, self-adaptive anchor calculation, and self-adaptive image rescaling.

Through these operations, the training results can be significantly improved.

The mosaic method of data augmentation is inspired by CutMix augmentation, which was introduced in 2019 (Bochkovsky et al., 2020; Yun et al., 2019). It was firstly implemented in YOLOv4, which only includes two images during the process. The mosaic augmentation takes four images and combines them into one image after some conversion.

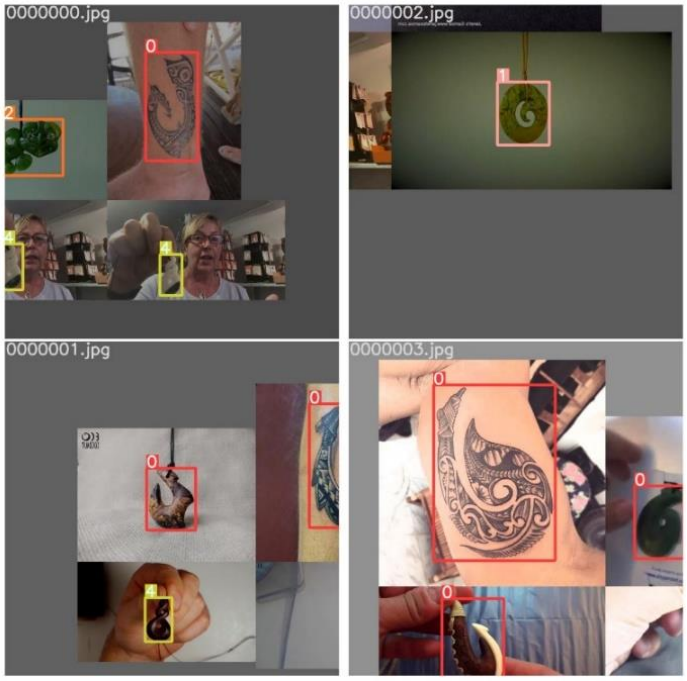

Figure 3.6: The results of mosaic augmentation on Māori symbols dataset

By mixing the images to make the model perform different scenarios and train the model to detect the objects in different scenes, the model will be adaptive to various contexts in those scenes while training. The four original images are multiplied in different ways. The mosaic augmentation brings four images into random cropping, which positively affects translation and occlusion. The classes of the objects might not from the same image in the training data.

The images will then be combined in a random sequence. It is optional to rescale the images into the same size before combining, with the bounding boxes in each image being resized simultaneously (Hao & Zhili, 2020). After the images are stitched together,

the formed image will be cropped again randomly. The augmentation method has various objects in the combined images if original images only contain one object (present one bounding box).

Pertaining to YOLO series methods, the initial shapes of anchors, ratios, and scales are different depending on the dataset. The dataset usually will have visual objects that are not quite under the same distribution of aspect ratios compared to the official dataset, such as COCO dataset (Laroca et al., 2018; Lin et al., 2014). YOLOv5 exploits *k*-means, initial guess, and genetic algorithms that automatically adjust the anchors during training (MacQueen, 1967; Smidstrup et al., 2014; Whitley, 1994). In order to compare the anchor against the data, there is a determination that if they fall below a certain matching threshold, the network will start training new anchors automatically, replacing the original anchors with new anchors in the model and training the model, then saving the model with these new anchors.

The images in the dataset for training the network are usually in various ratios. The general methods to process the images are to scale them into a standard size and then put them through the detection network. The YOLOv5 has introduced a novel method for rescaling the input images for improving the inference speed. The images after rescaling might have varying degrees of black padding, which could be regarded as the redundancy that affects the inference speed in real-time tasks. The pseudocode for this operation is shown below:

Algorithm 1 Image Rescaling

Input: Input image
Output: Rescaled image

function LETTERBOX(*Image*, *new_shape*=(640, 640), *color*=(114, 114, 114), *auto*=True, scaleFill = False, scaleup = True)
shape ← [image − height, image − width]
    if *new shape is integer* **then**
        new_shape ← (new_shape, new_shape)
    end if
    r ← min(new_shape height/shape height, new_shape width/shape width)
    if *not scaleup* **then**
        $r \leftarrow min(r, 1.0)$

```
    end if
    ratio ← [r,r]
    new unpad ← [integer(round(shape width * r)), integer(round(shape height * r))]
    dw, dh ← new_shape width − new_unpad[0], new_shape height − new_unpad[1]
    if auto then
        dw, dh ← MOD(dw, 64), MOD(dh, 64)
    else
        if scaleFill then
            dw, hd ← 0.0, 0.0
            new_unpad ← new_shape
            ratio ← [new_shape height/shape with, new_shape width/shape height]
        end if
    end if
    dw ← dw/2
    dh ← dh/2
    if shape(not includes the last element) is not equal to new_unpad then
        image ← openCV.resize(image, new_unpad, interpolation = cv2.INTER
    LINEAR)
        top, bottom ← integer(round(dh − 0.1)), integer(round(dh + 0.1))
        left, right ← integer(round(dw − 0.1)), integer(round(dw + 0.1))
        image ← openCV.copyMakeBorder(image, top, bottom, left, right,
    cv2.BORDER CONSTANT, value =color)
    end if
    return image, ratio, (dw, dh)
end function
```

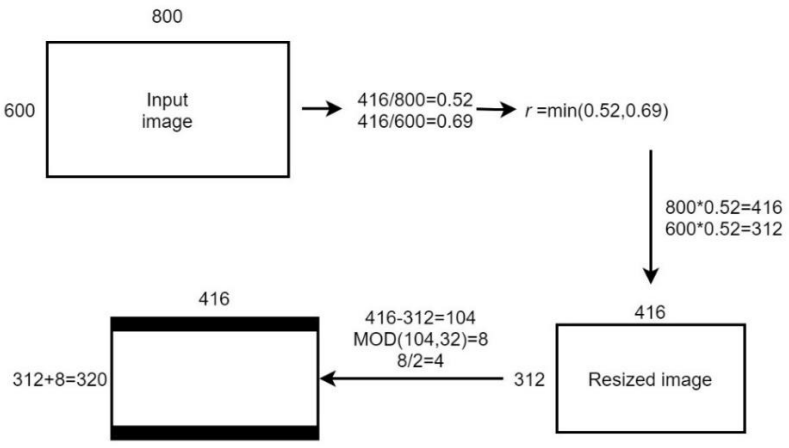

Figure 3.7: The self-adaptive image rescaling

The rescaling process is described in three steps. The first step is to calculate the ratio between the input image and the parameter *new_shape*, which is set as (416,416)

or (640,640) as default. The ratio between the heights and widths will be compared, and the smaller ratio r is chosen for the further processes. The second step is to calculate the size after rescaling, with the height and width of the input image both multiplying r separately.

In last step, the resized height h subtracts the resized width w for getting the total height of the padding h_p. The network YOLOv5 requires five times downsampling, so h_p mod 32(equals to 2^5) and divided by 2 to get the height of top padding and bottom padding (Xu & Jin, 2008).

3.2.2 Network Structure

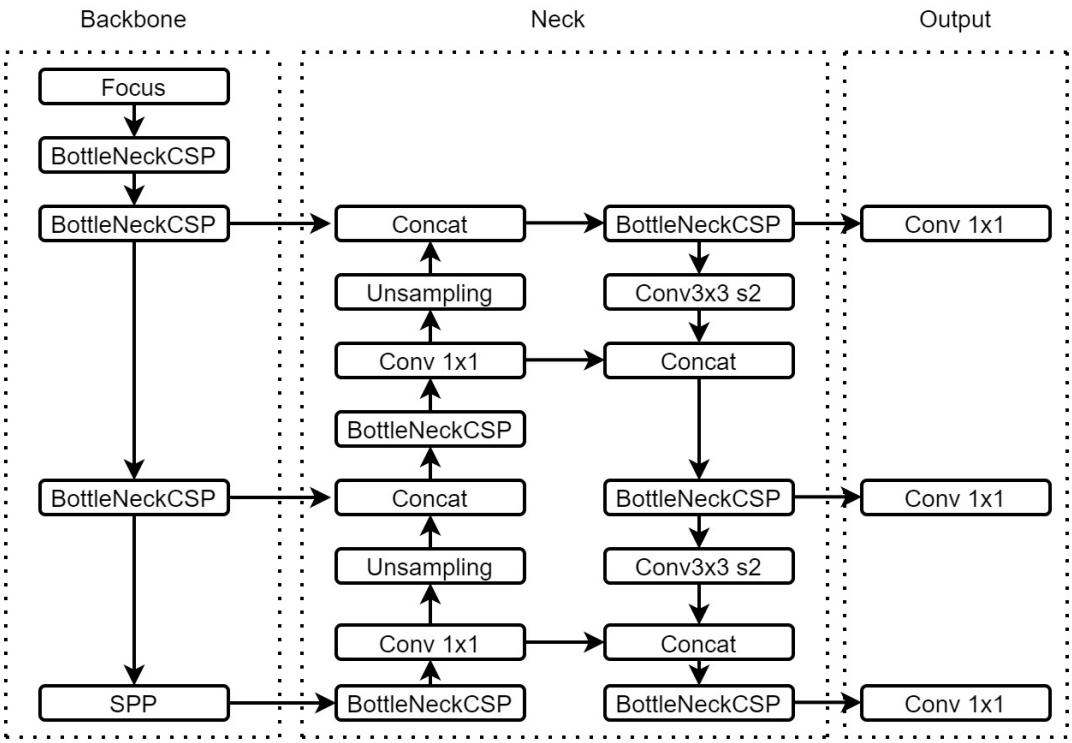

Figure 3.8: The network structure of YOLOv5

The network structure of YOLOv5 contains three major components, including the backbone, the neck structure, and output layers. Inside its backbone, the Focus structure is introduced for implementing slicing operation. The function of the Focus layer is similar to the SpaceToDepth in TResNet (Ridnik et al., 2021).

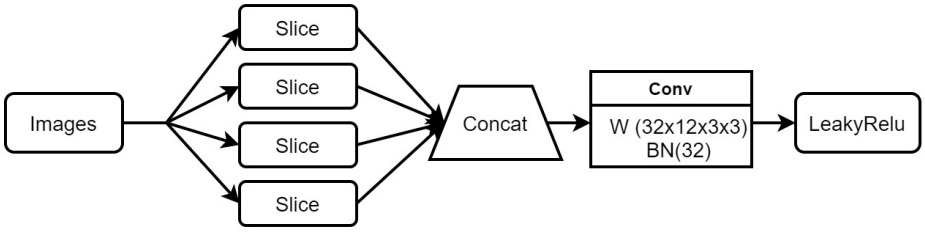

Figure 3.9: The focus structure

The Focus layer provides the input of a transformation from space to depth. The operation is to get one value after every one pixel, just like adjacent downsampling. The result represents four images which complementary to each other without losing information. Therefore, the information of *W* and *H* will be concentrated to the channels, with the number of the channels expanding four times, which also means the original RBG channels have increased into 12 channels (Yao et al., 2021). Those four images will be put through a convolutional layer, to generate doubled feature maps without information spoilage. Implementing Focus structure to the input image is to reduce the cost of computation from 2-dimensional convolutional layer and operate tensor reshaping to reduce the resolution(space) and increase the number of channels(depth).

There is a critical component inside the backbone and neck structure: The BottleNeckCSP structure (Zhou et al., 2021). The CSPNet is regarded as an upgraded version of DenseNet (Wang et al., 2020; Huang et al., 2017). According to the dense block and transition layer, CSPNet optimizes the method of backpropagation and improves the learning ability of the network. As the depth and width of the neural network have increased, the volume of the network becomes more significant and requires more computation. It limits the usage of the network on mobile devices. The primary purpose of CSPNet is to make the deep learning method such as ResNet, DenseNet is deployed on CPU and mobile devices without sacrificing the inference performance.

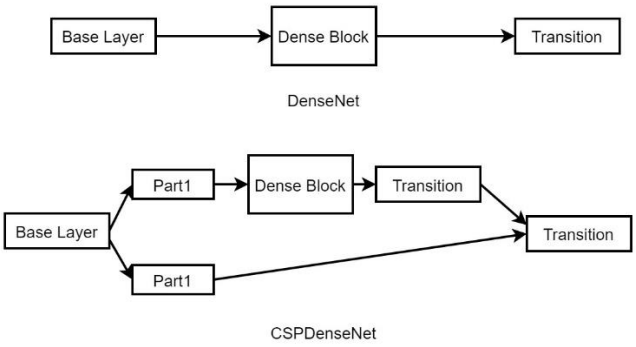

Figure 3.10: Comparing DenseNet and CSPDenseNet

The intention of the Cross Stage Partial (CSP) structure is to reduce the computation and enhance the performance of the gradient. The scheme is to split the input into two parts before entering the dense block. The block calculates one part of the input, and the other part is directly concatenated by using a shortcut connection (Wang et al., 2021). Taken the DenseNet as an example, inside the altered structure CSPDenseNet, the input feature map x_0 has been separated into $[x_0', x_0'']$, x_0' is linked to the last transition layer, and x_0'' is going through the dense block.

Regarding the gradients operated to update the weights, the path contains duplicate gradient information which belongs to the other. Based on retaining the original network structure, the path for passing the gradient has doubled. The cross-stage method can reduce the negative effect of copying the feature map directly for the concatenation process and reduce the computational complexity (Liu et al., 2020). The BottleNeckCSP structure in YOLOv5 integrates the ideas of BottleNeck and CSP structure.

The BottleNeck is a neural network structure that compresses and amplifies information. This structure is generally found in autoencoders, fire-module in squeeze-net, and ResNet (Gehring et al., 2013; Landola et al., 2016). In order to reduce the dimension and increase the dimension of the input, which is similar to NMF, it has the capability of removing high-frequency noise of the images (Lee & Seung, 1999). Generally speaking, the deep learning methods that implement BottleNeck structure into their networks have achieved better accuracy than traditional convolutional networks

and fully connected networks. The basic structure is regarded as a classic residual structure. The first part is 1×1 convolutional block, including 1×1 convolutional layer, batch normalization layer, and leaky ReLU. Then a 3×3 convolutional block, and the result is added with the initial input through the residual structure (Bjorck et al., 2018; Xu et al., 2015).

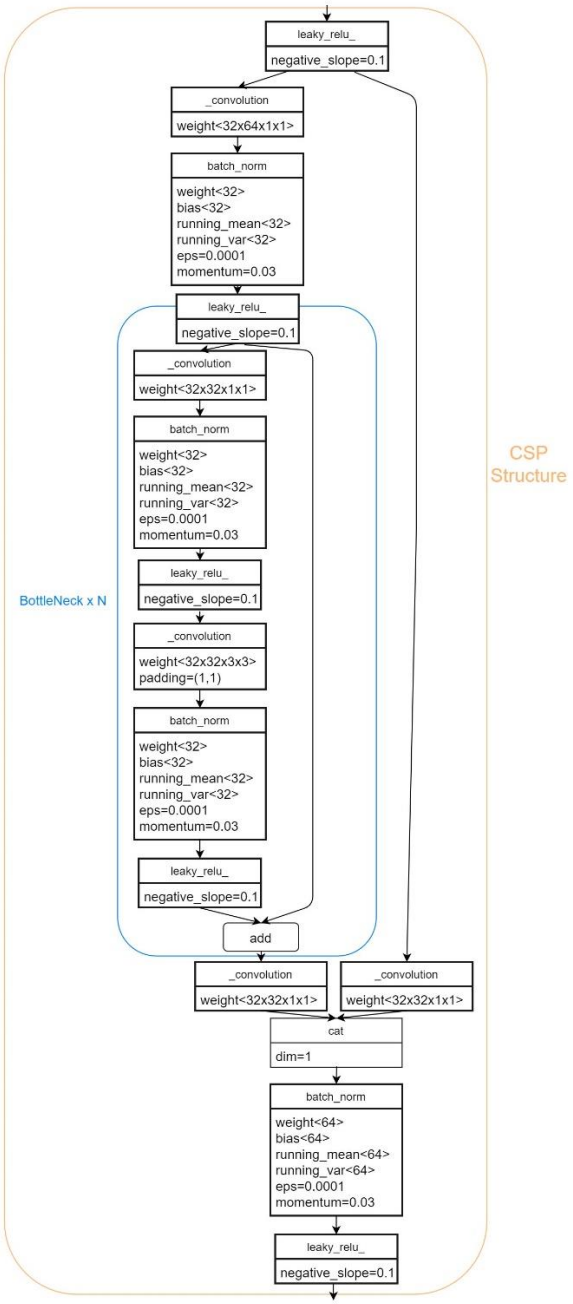

Figure 3.11:BottleNeckCSP structure

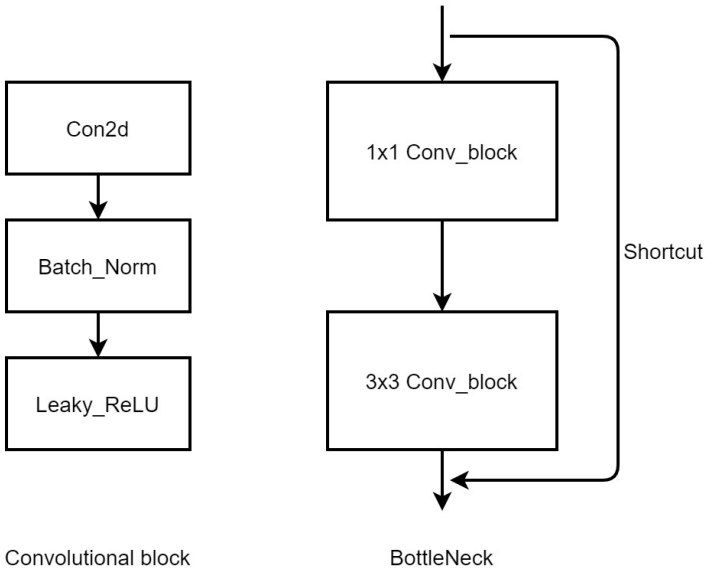

Figure 3.12: Structure of convolutional block and BottleNeck

Due to various reasons, video and audio data contain extremely abundant and diverse information. For the specific tasks (human face recognition, object tracking, voice recognition, etc.), only a small amount of data is useful for the tasks, and most of the data is redundant and irrelevant. According to the theory, the extracted feature from the initial data, the classification results are just a “form of expression” of the information. The algorithms need to eliminate the useless information from the collected data and retain the valuable information for the specific tasks. We hope to obtain a short expression of the image. Thereby, the parameters inside the networks and the complexity of the model are reduced. The BottleNeck structure is one of the solutions.

The BottleNeck reduces the number of channels through a 1×1 convolutional block. The number of the channels of the convolution in the middle of the network has diminished to $\frac{1}{4}$. The convolution in the middle contains the same number of channels as the input channels. The 3×3 convolutional block is exploited to increase or restore the number of the channels. The number of channels of the BottleNeck output equals the input channels. The BottleNeck in the deeper networks can reduce the use of the

parameters and computation, improving the performance of the networks (Rezende et al., 2017).

The primary function of the Spatial Pyramid Pooling (SPP) module is to solve the problem that the sizes of the input images are not uniform. For most object detection methods, the output layer would be fully connected, which requires that input images be managed into the same size. The present image preprocessing methods, such as resizing, and cropping will cause a certain degree of image distortion and affect the final accuracy. The SPP module implements multiple pooling layers to generate same size outputs from the input data. The SPP module (He et al. (2015) is mainly designed for two problems:

- To Avoid the image distortion caused by the image cropping, rescaling, etc.
- To solve the problem of the CNN extracting feature maps repeatedly, which dramatically improves the speed of generating candidate bounding boxes and saves computational cost.

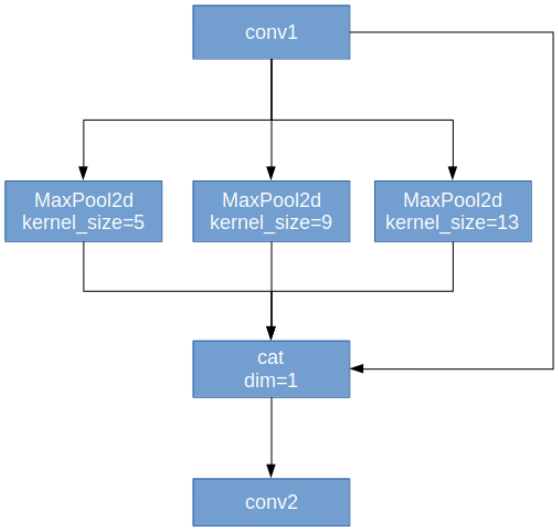

Figure 3.13: SPP module in YOLOv5

The SPP module in YOLOv5 contains four parallel paths for forwarding the information. Three paths have been connected with max-pooling layers, with kernel

sizes as 5×5, 9×9, 13×13, respectively. And the fourth path transmits the output of the first convolutional layer to the second convolutional layer directly using a shortcut connection. The module is referenced by the idea of the spatial pyramid to achieve the fusion of local features and global features. After the fusion, the expression ability of the feature maps is enriched. This is conducive to the situations when the sizes of the targets have significant differences during the detection. Especially for the complex multi-target detection methods such as the YOLO series, it greatly improves the detection accuracy.

Pertaining to object detection methods, in order to achieve better performance on extracting features, extra layers will be inserted into the backbone and output layers, which is called the Neck structure. The PANet has inspired the Neck structure in YOLOv5. It is introduced for instance segmentation tasks.

The PANet adds a bottom-up path augmentation structure to the FPN network, which does not exceed 10 layers (Liu et al., 2018). The shallow layers in FPN will be connected to the last layer P_2 of the top-down structure through a horizontal shortcut, and the information is transferred from P_2 to the top layers along with the augmentation structure. The number of layers within this process will not exceed 10, so the information of shallower features can be preserved in a better way. Inside the augmentation structure, except the first layer N_2, the rest of the layers are fusion results of the feature maps in FPN.

The bottom-up path augmentation structure is a conventional feature fusion operation (Tan et al., 2019). The general combination can be represented as feature map N_i passes through a convolutional layer in size 3×3, with $stride = 2$, the size of the feature map will be reduced to half. Then it will be added to feature map P_{i+1}. The result needs to get through another convolutional layer in size 3×3, with $stride = 1$ to form the final feature map N_{i+1}. The bottom-up path augmentation structure improves the speed of information fusion and shortens the length of the path between the low-level features and high-level features.

In FPN, visual objects in different sizes are allocated to different layers, such as the most miniature objects will be assigned to feature map P_2, the biggest will be allocated to P_5. This method is straightforward and effective, but the result might not be the best. For example, two objects with a difference of only 10 pixels might be assigned to different feature maps. To receive a better result, PANet proposed adaptive feature pooling, which is described:

- Applying feature extraction using RoIAlign and generating four sets of feature maps that are in the same shape.
- To fuse the feature maps by calculation methods such as sum, product, etc.
- Operating the fused feature maps for classification tasks, bounding box prediction, and mask prediction

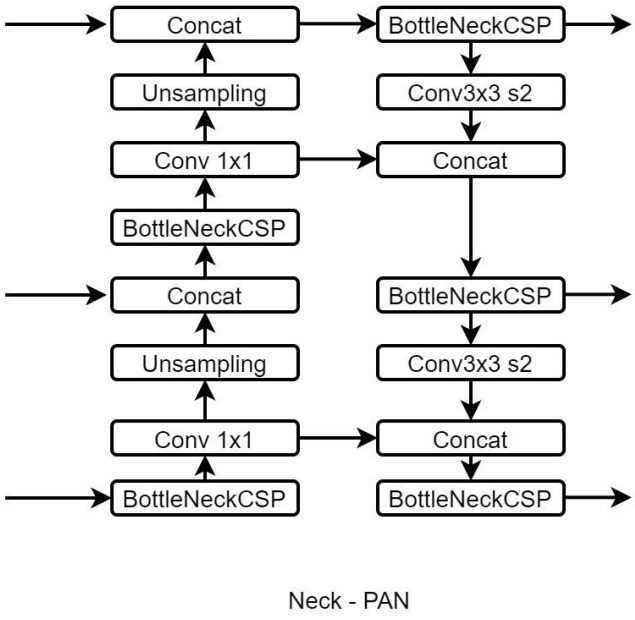

Figure 3.14: The neck structure in YOLOv5

However, to simplify the network and facilitate the deployment in actual situations, PAN structure in YOLOv5 does not implement adaptive feature pooling. The Neck structure has replaced the addition operation with concatenation to improve the

performance of the predictions. The BottleNeckCSP in Neck structure is slightly different. The residual units are replaced by the CBL block, which combines convolutional layer, batch normalization layer, and activation function layer (Leaky ReLU).

3.2.3 Loss Function

The method of loss calculation in YOLOv5 contains three parts, including calculating the loss value of localization, classification, and confidence. Calculating localization loss, which also means bounding box prediction, is an important task in object detection methods. In order to give a bounding box to the target, which needs to predict the location of that box, a typical method utilized for the prediction is calculated by using the squared loss function as eq. (3.34),

$$L_{local} = (x - x^*)^2 + (y - y^*)^2 + (w - w^*)^2 + (h - h^*)^2 \qquad (3.34)$$

where (x, y) represents the top-left coordinates of the predicting box, (w, h) shows the width and height of the box (Domingos, 2000). For predicting the bounding boxes, the network needs the information of the overlap area between the prediction and the ground truth bounding box, it is better if the network gets a bigger ratio of the overlapping area to the union area. Although the value cannot be well measured by only applying the squared loss function. Thus, various loss calculation methods are introduced to solve the problem, including Mean Squared Error Loss (MSE), IoU loss, GIoU loss, DIoU loss, and CIoU loss (Wang & Bovik, 2009; Zhou et al., 2019). Regarding YOLOv5, the methods for calculating the localization loss include GIoU, DIoU, CIoU. And the network sets CIoU as the default method.

Intersection over Union (IoU) presents the ratio of the intersection area and the union area of the prediction and ground truth:

$$\mathrm{IoU}(B_1, B_2) = \frac{|B_1 \cap B_2|}{|B_1 \cup B_2|} \qquad (3.35)$$

If the prediction and the ground truth overlap entirely, the IoU should be 1.00, and it

will be 0 if the boxes do not interact with each other. The IoU loss is defined as eq. (3.36)

$$\mathrm{L_{IoU}} = 1 - \mathrm{IoU}\left(\mathrm{B}, B_{gt}\right) \tag{3.36}$$

However, two problems need to be considered. If the prediction and the ground truth box have no intersection area, the loss value will be 1.00, which cannot describe the distance between the boxes. Another problem is when the ground truth box includes the prediction box, the ratio between the boxes and the value of the IoU are fixed, which leads to the problem that the loss value will have no change wherever the prediction stays in the ground truth.

For solving the problems of IoU, GIoU is introduced to the object detection methods. GIoU adds one extra box based on IoU that can contain the ground truth and the smallest prediction box. Because IoU is a concept of ratio, it is insensitive to the scale of the objects. The purpose of the GIoU is equivalent to add a closure penalty relevant to the prediction and the ground truth (Rezatofighi et al., 2019). The optimization of the regression loss of the bounding box, such as MSE loss and L1-smooth loss, is not entirely equivalent to IoU optimization. Also, L_n norm is sensitive to the scale of the object. IoU cannot optimize the area that is not included in the intersection directly.

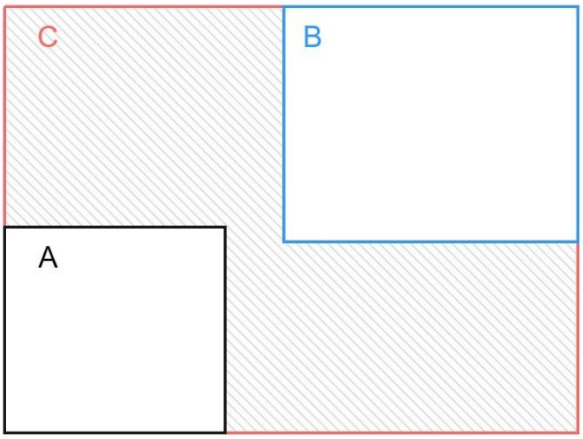

Figure 3.15: The penalty is the minimal area of the shaded area

Suppose area A represents ground truth, B shows the prediction, and C is the closure area of A and B, the GIoU is shown as eq. (3.37).

$$\text{GIoU} = \text{IoU} - \frac{|C \backslash (A \cup B)|}{|C|} \tag{3.37}$$

where the area of $|C \backslash (A \cup B)|$ represents by area of C minus the area of $(A \cup B)$. For applying the GIoU as the method to calculate the loss of the bounding box regression, we assume the coordinates of the prediction and ground truth are $B^p = (x_1^p, y_1^p, x_2^p, y_2^p)$ and $B^g = (x_1^g, y_1^g, x_2^g, y_2^g)$, the area of the boxes and be represented as :

$$A^g = \left(x_2^g - x_1^g\right) * (y_2^g - y_1^g) \tag{3.38}$$

$$A^p = \left(x_2^p - x_1^p\right) * (y_2^p - y_1^p) \tag{3.39}$$

The intersection of B^p and B^g is shown as eq. (3.40).

$$I = \begin{cases} (x_2^I - x_1^I) * (y_2^I - y_1^I) & x_2^I > x_1^I, y_2^I > y_1^I \\ 0 & otherwise \end{cases} \tag{3.40}$$

where (x_1^I, y_1^I) and (x_2^I, y_2^I) are the top-left and bottom-right coordinates of the intersection area, can be calculated as:

$$x_1^I = \max\left(\hat{x}_1^p, x_1^g\right), x_2^I = \min\left(\hat{x}_2^p, x_2^g\right) \tag{3.41}$$

$$y_1^I = \max\left(\hat{y}_1^p, y_1^g\right), y_2^I = \min\left(\hat{y}_2^p, y_2^g\right) \tag{3.42}$$

where $\hat{x}^p$ and $\hat{y}^p$ are calculated by getting the minimum and maximum values between x_1^p, x_2^p and y_1^p, y_2^p:

$$\hat{x}_1^p = \min\left(x_1^p, x_2^p\right), \hat{x}_2^p = \max\left(x_1^p, x_2^p\right) \tag{3.43}$$

$$\hat{y}_1^p = \min\left(y_1^p, y_2^p\right), \hat{y}_2^p = \max\left(y_1^p, y_2^p\right) \tag{3.44}$$

Based on the steps which are shown above, the coordinates of the minimum area B^c that can contain both B^p and B^g can be calculated by:

$$x_1^c = \min\left(\hat{x}_1^p, x_1^g\right), x_2^c = \max\left(\hat{x}_2^p, x_2^g\right) \tag{3.45}$$

$$y_1^c = \min\left(\hat{y}_1^p, y_1^g\right), y_2^c = \max\left(\hat{y}_2^p, y_2^g\right) \tag{3.46}$$

and B^c is

$$A^c = (x_2^c - x_1^c) * (y_2^c - y_1^c) \tag{3.47}$$

Thus, eq. (3.47) for calculating the IoU is changed into the following form,

$$\text{IoU} = \frac{I}{U} = \frac{I}{A^p + A^g - I} \tag{3.48}$$

and the value of GIoU and the GioI loss are calculated by the following eq. (3.49-3.50)

$$GIoU = IoU - \frac{A^c - U}{A^c} \tag{3.49}$$

$$L_{GIoU} = 1 - GIoU \tag{3.50}$$

The GIoU loss optimizes the situation if the prediction and ground truth have no intersection area. If the distance between the boxes is very close, the GIoU loss draws near the IoU loss. Therefore, the results of the two loos functions are similar, although the GIoU loss permits the network to have a faster convergence.

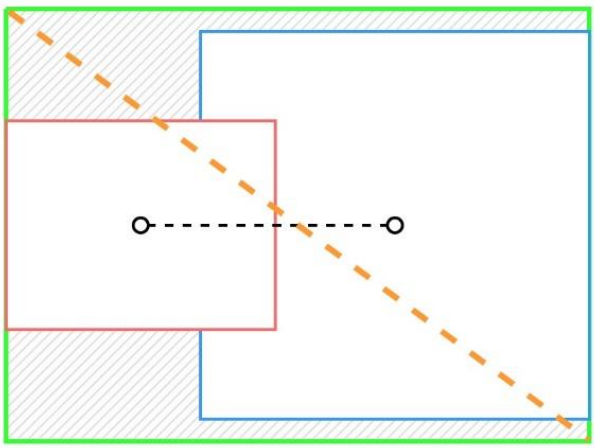

Figure 3.16: DIoU

Compared to GIoU, Distance-IoU (DIoU) is more consistent with the mechanism of regression, considering the distance between the boxes, intersection rate, and the sizes of the boxes at the same time, which allows the regression result to become more stable, reduces the rate of the divergence during the training processes (Zheng et al., 2020). The blue grid represents the ground truth B^{gt}, and the red grid shows the prediction B. The black dash line connects the central points of those two boxes, the green box represents the minimum closure box C, and the red dash line c is the diagonal of C.

The IoU-based loss is defined as $L = 1 - IoU + R(B, B^{gt})$, where the element $R(B, B^{gt})$ is the penalty of the loss function. The penalty in DIoU is defined as eq. (3.51),

$$R_{DIoU} = \frac{\rho^2(b, b^{gt})}{c^2} \tag{3.51}$$

where $\rho(\cdot)$ represents the Euclidean Distance between two central points, which is the length of the black dash line, b and b^{gt} show the central point of B and B^{gt} (Danielsson, 1980). The Euclidean Distance is calculated by the eq. (3.52),

$$\rho = \sqrt{\rho^2(B, B_{gt})} = \sqrt{(x_1^p - x_2^p)^2 + (y_1^p - x_2^p)} \tag{3.52}$$

where x^p and y^p are defined based on the coordinates of prediction and ground truth:

$$x_1^p = x_2^B - x_1^B, y_1^p = y_2^B - y_1^B \tag{3.53}$$

$$x_2^p = x_2^{B_{gt}} - x_1^{B_{gt}}, y_2^p = y_2^{B_{gt}} - y_1^{B_{gt}} \tag{3.54}$$

based on the elements above, DIoU is described as eq. (3.55)

$$\mathrm{DIoU}\left(\mathrm{B}, B_{gt}\right) = \mathrm{IoU}\left(\mathrm{B}, B_{gt}\right) - R_{DIoU} \tag{3.55}$$

and DIoU loss is calculated as eq. (3.56)

$$\mathrm{L_{DIoU}} = 1 - \mathrm{DIoU} = 1 - \mathrm{IoU} + R_{DIoU} = 1 - \mathrm{IoU} + \frac{\rho^2(b, b^{gt})}{c^2} \tag{3.56}$$

Similar to GIoU loss, DIoU provides the moving direction for the bounding box while it does not overlap with the ground truth (Yuan et al., 2020). It minimizes the distance between two boxes more directly, which gives a faster convergence while training compared to GIoU. While dealing with the circumstances, if the central points of the prediction and ground truth are on the same line vertically or horizontally, DIoU loss increases the speed of calculating regression, while GIoU loss is degrading into IoU loss.

Complete-IoU (CIoU) is regarded as the upgraded version of DIoU. Considering the ratio issues of width and height between the bounding boxes, CIoU has added one extra impact factor av into the calculation process. The αv is combined by two elements, α is a parameter utilized for trade-off, which is defined as eq. (3.57)

$$\alpha = \frac{v}{(1-IoU)+v} \tag{3.57}$$

where v is the parameter which is employed for measuring the consistency of the aspect ratio:

$$v = \frac{4}{\pi^2}(arctan\frac{w^{gt}}{h^{gt}} - arctan\frac{w}{h})^2 \tag{3.58}$$

and the equation of CIoU and CIoU loss are shown as (3.59),

$$CIoU = IoU - (\frac{\rho^2(b,b^{gt})}{c^2} + av) \tag{3.59}$$

and

$$L_{CIoU} = 1 - CIoU = 1 - IoU + \frac{\rho^2(b,b^{gt})}{c^2} + av. \tag{3.60}$$

For calculating the classification loss and confidence loss, YOLOv5 implements the method of BCEWithLogitsLoss and Focal Loss. The BCEWithLogitsLoss combines the sigmoid function layer with Binary Cross-Entropy (BCE) loss together as one unit, which is more stable than simply applying the BCE loss after the sigmoid function,

$$L(x,y) = \{l_1, \dots, l_N\}^T, l_n = -w_n[y_n \cdot \log(\sigma(x_n)) + (1-y_n) \cdot \log\big(1-\sigma(x_n)\big)] \quad (3.61)$$

where x_n represents the score of predicting the *n*-th positive sample, y_n shows the label of the *n*-th sample, σ is the sigmoid function, and N denotes the batch size.

Pertaining to Focal loss, it is introduced as a strategy for solving the problem of serious disequilibrium between positive and negative samples. The class imbalance is a serious problem while training the object detection network. Because the network will apply intensive sampling in every position on the input image, if the image only contains a small number of objects, then the number of negative samples will be much more than the positive samples (Tran et al., 2019). For current detection methods, the RPN module can filter out negative samples. The detection head also adopts a fixed proportion for sampling positive and negative samples (1:3 for example) or applies Online Hard Example Mining (OHEM) for dealing with the class imbalance issue (Tang et al., 2018; Shrivastava et al., 2016).

Different from sampling methods, focal loss adjusts the cross-entropy dynamically based on the confidence. Suppose the confidence of correct prediction increases, the coefficient of loss will gradually decay to 0. In that case, the training loss pays more attention to the hard samples, while the loss values for most of the easy samples give less contribution. The cross-entropy for binary classification is defined as eq. (3.62),

$$CE(p,y) = \begin{cases} -\log(p) & if\ y = 1 \\ -\log(1-p) & otherwise \end{cases} \quad (3.62)$$

where $y \in \{-1,1\}$ represents the true label, as 1 stands for positive sample, and -1 stands for negative sample. The element $p \in [0,1]$ is the probability that the model predicts a positive sample:

$$p_t = \begin{cases} p & if\ y = 1 \\ 1-p & otherwise \end{cases} \quad (3.63)$$

and the equation of cross-entropy is simplified as eq. (3.64)

$$CE(p,y) = CE(p_t) = -logp_t \quad (3.64)$$

While applying cross-entropy for calculating the loss, the easy samples might contribute a large proportion of loss values, which may cause the imbalance issue. For eliminating the problem, focal loss adds an extra regulatory factor$(1-p_t)^\gamma$ into the function as shown in eq. (3.65),

$$FL(p_t) = -(1-p_t)^\gamma log p_t \tag{3.65}$$

where if p_t is small, the factor approach to 1, which will not affect the loss value. If p_t approaches to 1.00, the factor will be close to 0, the number of correct easy sample losses will be reduced effectively.

3.2.4 Dataset， Modification and Resources

Table 3.2: Image distribution inside datasets

Network	Train	Validation	Total
ABCNet	1600	400	2000
YOLOv5	2400	600	3000

Regarding our experiments, there are two individual datasets for training and validating their corresponding networks, ABCNet and YOLOv5. The dataset for ABCNet named “ABC” contains 2,000 images in total. The dataset “v5” for YOLOv5 includes 3,000 images.

The dataset for ABCNet comprises four types of characters, including capital letters, lowercase letters, punctuations, and numbers. There are 7,677 instances inside the dataset, where the English letters are the main components to form the instances. According to the annotation files, there are 37,380 English letters, where capital letters are 5,312, and lowercase letters are 32,068.

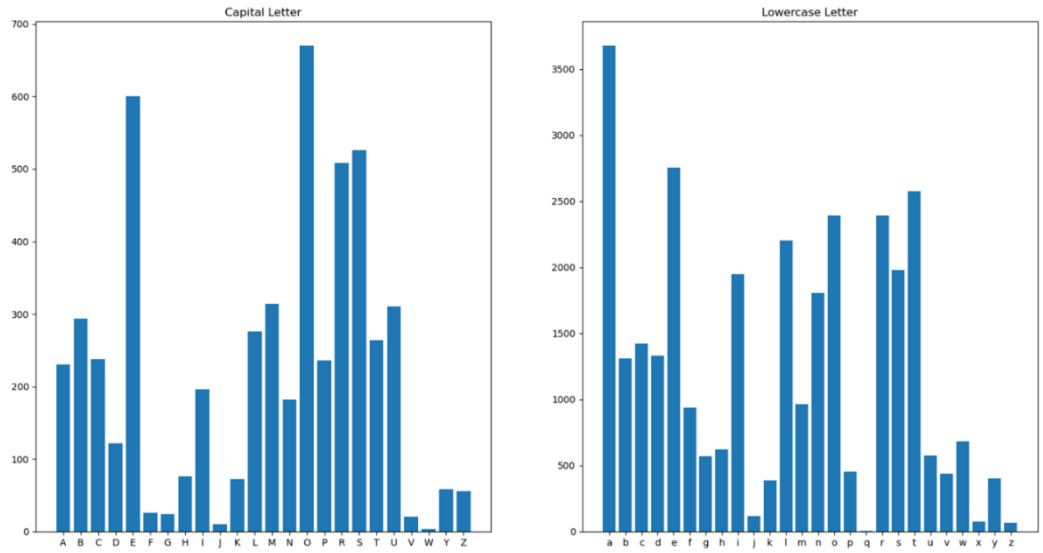

Figure 3.17: The distribution of Alphabets in dataset ABC

The lowercase letters include the letters from ‘a’ to ‘z’, but the capital letter ‘X’ is missing. Although the number of each letter is not equal because some letters appear more often in instances, and some are employed less for expression. For capital letters, letter ‘O’ is applied the most as 670 times, letters ‘J’ and ‘W’ only appear four times and one time respectively, which are the least used capital letters in the dataset. Letter ‘a’ is applied for lowercase letters, which appears 3,676 times in the instances. Letter ‘q’ is the least used lowercase letter that only appears two times.

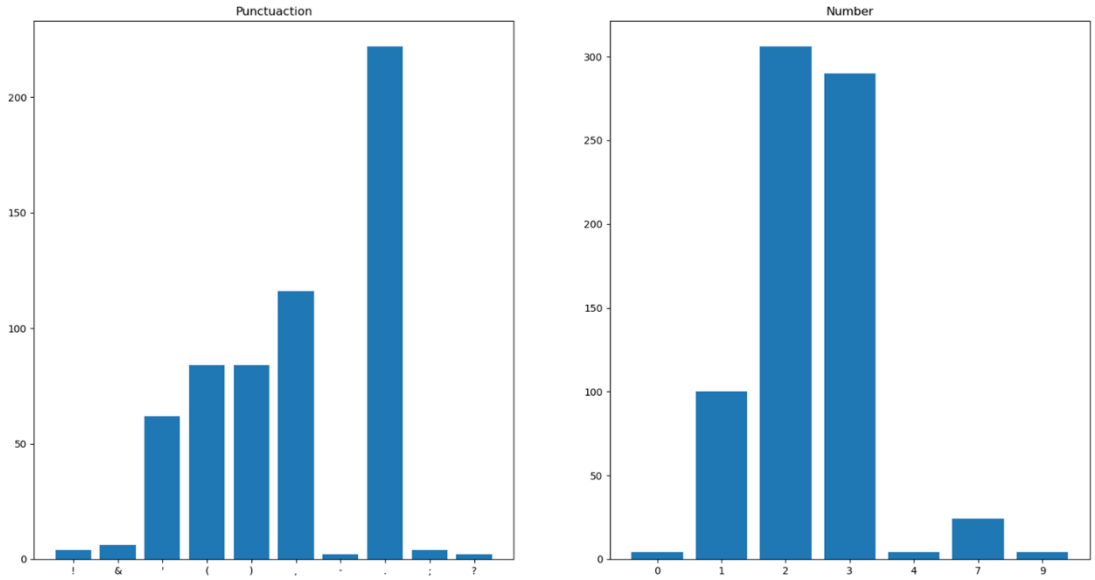

Figure 3.18: The distribution of punctuations and numbers in dataset ABC

The components to form the instances also include punctuations and numbers, not all the punctuations are included inside the dataset, only ten general symbols. The full comma is the most employed punctuation, appearing 222 times in the dataset. For the numbers, the dataset does not include all the numbers, missing 5, 6, and 8. The number 2 appears the most in the dataset as 306 times.

The format of the data is to mimic the structure of dataset CTW1500 with minor changes. The first step is to generate the CTW1500 like annotations stored as txt files. The labelling tool for the labelling process is a tool which is designed for ABCNet, to draw two types of bounding boxes, including rectangular-shape and arbitrarily-shape boxes. The labelling tool requires 14 points for each bounding box. The format of the label for each box is represented as $[x_1, y_1, x_2, y_2 \dots \dots x_{14}, y_{14}, t]$, where t stands for the text that needs to be detected and recognized inside the box.

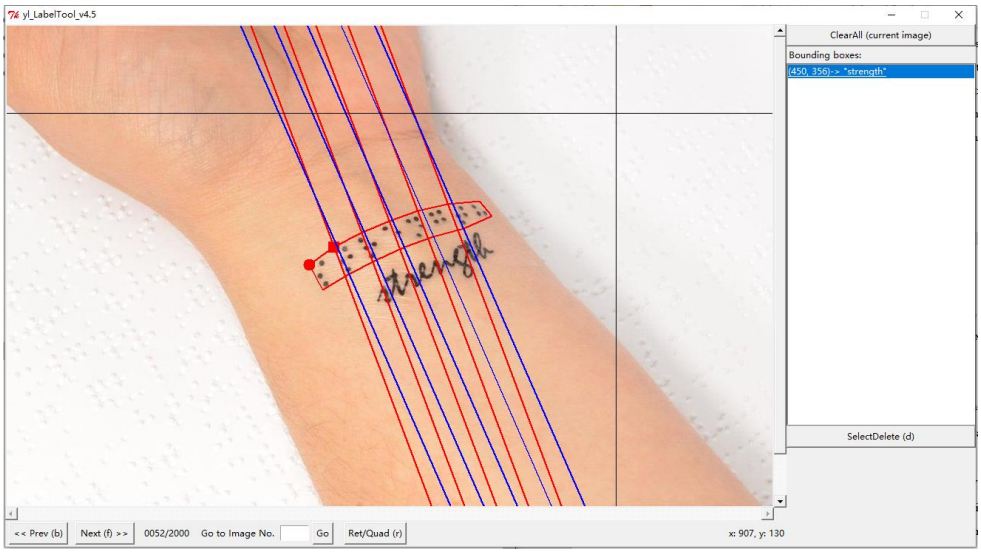

Figure 3.19: Labeling tool

After the CTW1500 style annotations are created, it is needed to transfer the label format into Bezier ground truth, with eight control points in total. After the transformation, the label will be changed into $[x_1, y_1, x_2, y_2 \dots \dots x_8, y_8|||t]$, the ground truth within eight control points will be drawn on the image to ensure the Bezier curves are nicely placed.

Figure 3.20: The Bezier ground truth

The dataset “ABC” has 2,000 images in total. There are 1,000 images collected through the Internet. Another 1,000 images are generated based on the first half images.

Figure 3.21: Comparison of original images and augmented images

Figure 3.21 shows the difference between the original images and the augmented images. The random noises are injected into the images using a script file, including Gaussian-distributed additive noise, Poisson-distributed noise, salt noise, pepper noise, salt & pepper noise (Russo, 2003; Dytso & Poor, 2020; Azzeh et al., 2018).

After adding the noises, the color of the images also changed according to the script. The augmentation only applies extra noises and color changes to the images. The pseudocode of the script is shown in Algorithm (2).

Algorithm 2 Image Augmentation

Input: Original image

Output: Augmented image

1: *path* ← *Image folder path*

2: *image names* ← *os.listdir*(*path*)

3: save direction ← path to save the modified images

4: **for** every image name in image names **do**

5: **if** image name ends with ('.jpg') **then**

6: img ← Image.open_image_file from pathjoin(path + image name)

7: *img* ← *Numpy.array*(*img*)

8: *noise img* ← skimage.util.randomnoise(img, mode='speckle')

9: *noise img* ← *noise img* ∗ 255

10: *noise img* ← *noise img.astype*(*numpy.int*64)

11: OpenCV.imwrite[(save direction + image name), noise img]

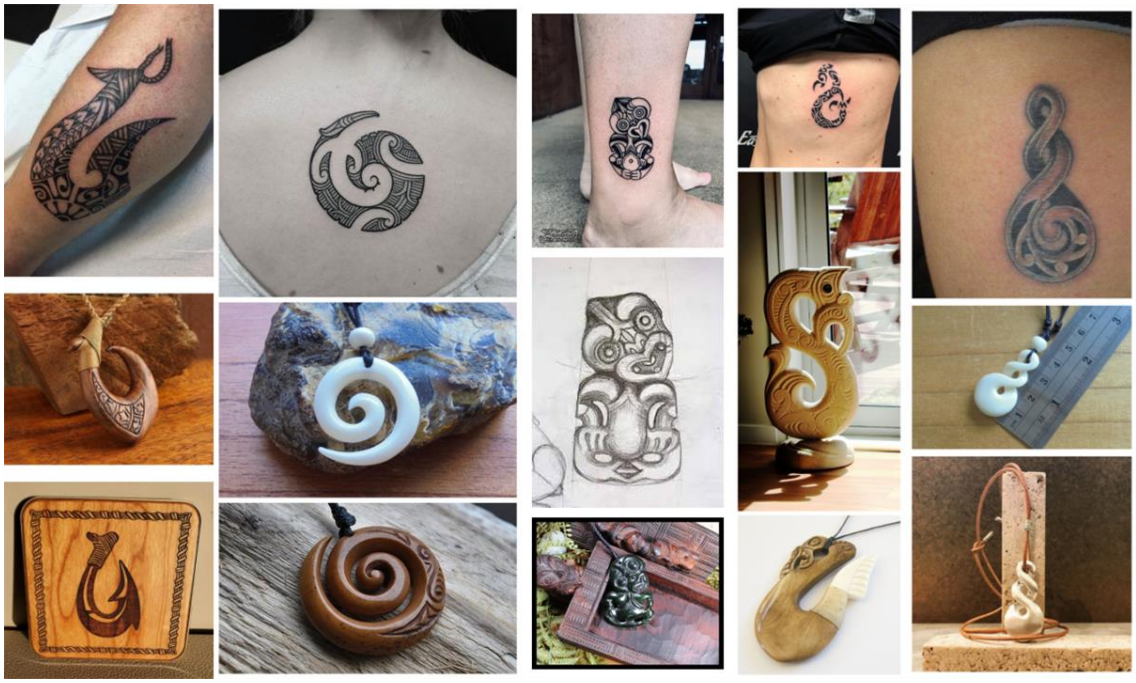

Figure 3.22: There are five classes of Maori symbols in dataset "v5". The symbols from left to right are Hei Matau, Koru, Hei Tiki, Manaia, and Pikorua.

Dataset "v5" for training YOLOv5 contains 3,000 images in total. It is split into two parts, with 80% of the images for training and the rest 20% exploited for validating the trained network. There are five types of Māori symbols in the dataset, including Hei Matau, Koru, Hei Tiki, Manaia, and Pikorua. The proportion of the classes in "v5" is very balanced compared to dataset "ABC". Each type has 600 images. All of them are

legal public images collected through the Internet, some are individual images, and some are video frames taken from multiple video files. The images include symbols represented mainly as Tattoos, engravings, and pieces of jewelry. We have not applied data augmentation to “v5” because YOLOv5 has already arranged the data augmentation method to be implemented during training, and a duplicate pre-made augmentation is not needed.

While preparing the dataset “v5”, a few labeling tools have been employed for generating the annotations for the images, including Labelme, LabelImg, and voTT (Russel et al., 2008; Yu et al., 2019; Ezhilarasi & Varalakshimi, 2018). After a few rounds of tests, LabelImg is chosen to label the images.

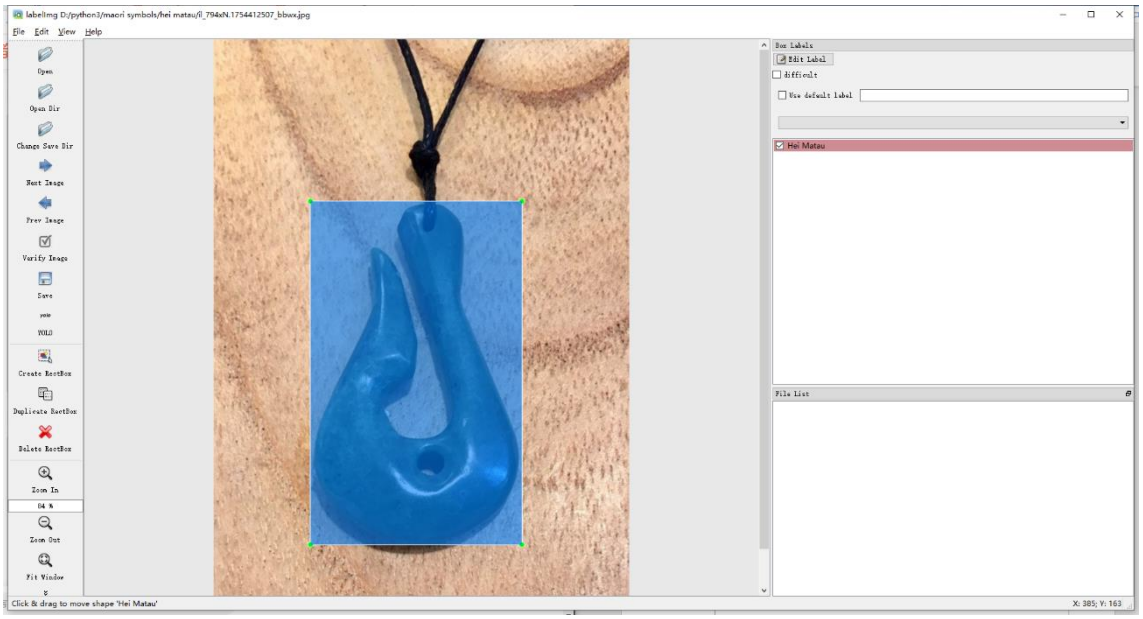

Figure 3.23: The user interface of LabelImg

Labelme is more suitable for dealing with the labels for image segmentation, voTT is very convenient to install and execute. But it does not support the YOLOv5 annotation format itself, which needs an extra process to transform the labels into the correct format. Compared to the proposed tools, LabelImg is also easy to install and be implemented for labeling. It has the option for generating the labels in YOLO required format. The generated annotation files are represented as txt files.

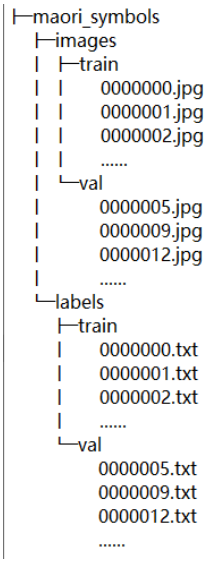

Figure 3.24: The structure of the dataset folder

The annotation format for the ground truth box is recorded as $[class, x_c, y_c, w, h]$, where the first value represents the class of the subject. Instead of describing the class using a word(words), the classes will be transformed into serial numbers (e.g., 0,1,2), and the numbers will be recorded instead while labeling the images. x_c and y_c are the coordinates of the central point of the bounding box after normalization, w represents the normalized width of the box, and h shows the normalized height of the box. The final structure of the dataset folder is presented in Figure 3.24. The reason why we choose it to generate the annotation files as txt files instead of XML files is that there are 3,000 images required to be labeled. The mistakes might have occurred during the labeling process. The .txt files are very easy to be modified. While an error is found, it is easy to adjust the file in a concise period.

ABCNet is integrated within a toolbox called AdelaiDet. The base functionalities of the toolbox are depending on the opensource framework Detectron2, which is introduced and managed by FaceBook AI (FAIR) (Joulin & Paris, 2015). The Adelaidet includes multiple deep learning methods such as FCOS, BlendMask, MEInst, ABCNet, ConInst (Chen et al., 2020; Zhang et al., 2020; Tian et al., 2020). Like other general detection frameworks, the predictions will be presented as rectangle boxes, wherever the detecting media is an image or a video clip. The toolbox has their own arrangement

or adjustments for each detection method to modify the detection box into its required styles. For ABCNet, the detection box is generated based on Bezier curves and Bernstein Polynomials. In order to perform the curved ground truth, there is a script file visualizer.py that is responsible for the task. The class has referred to the Visualizer algorithm designed by Detecron2. There is one class called TextVisualizer created in the script, and the class includes five functions. They are designed to get multiple coordinates on the parameterized Bezier curves to draw the prediction box, apply the functionality for decoding the classification result into general text, and draw the prediction boxes.

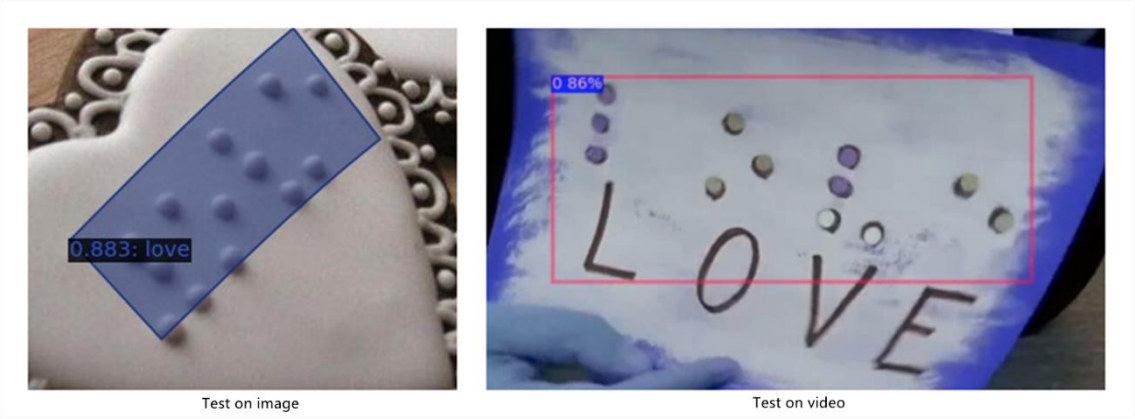

Figure 3.25: Comparisons of the prediction results between image and video.

We run a prior training before starting the actual project with a small Braille dataset for checking the prediction results. The network generates the boxes built with curved boundaries. The label shows the value of the score with the predicted text. But suppose we switch the detecting source from image to video. In that case, the prediction result is shown in the rectangular box. And the label only shows the percentage of the score without the predicted text, which is unable for the user to estimate if the prediction has fully matched all the characters or there still exist some false predicted elements for the word. Based on this result, by checking through the documents and script files corresponding to ABCNet inside AdelaiDet toolbox, a possible algorithm inside the script file predictor.py in the ‘demo’ folder might be the main reason for the differences.

Inside the script file, there are two functions. The first function is responsible for

producing prediction outputs for images. The second function has the duty for videos. Read through the details of two functions. The first function utilizes the TextVisualizer as the visualizer to produce the prediction. The second function still applies the visualizer algorithm designed by Detectron2. The original visualizer can only produce a hollow rectangular box for prediction, which cannot perform curved or polygon boundaries for the area. Therefore, it is necessary to modify the function to make it possible when utilizing the right visualizer for detecting the target video. After a few rounds of adjustment, the function can generate the right style of boundaries for the prediction. The pseudocode of the function is shown as Algorithm (3).

Algorithm 3: Produce Prediction on Video

Input: Video for inference
Output: Predictions

```
function RUN_ON_VEDIO(video)
    function PROCESS_PREDICTIONS(frame, predictions)
        frame ← OpenCV.cvtColor(frame, OpenCV.COLOR_RttB2BttR)
        if panoptic segment is in predictions then
            panoptic segment, segment info ← predictions["panoptic_segment"]
            vis_frame ← video visualizer.draw_panoptic_seg predictions[frame,
                                panoptic_segment.to(transfer to cpu), segments info]
        else
            if instances are in predictions then
                frame ← frame[:, :, :: −1]
                predictions ← predictions["instances"].to(transfer to cpu)
                vis_frame ← visualizer.draw_instance_predictions(predictions)
            else
                if semantic segment in predictions then
                    vis_frame ← video visualizer.draw_semantic_segment(frame,
predictions["semantic_segment"].argmax(dimension=0).to(transfer to cpu))
                end if
            end if
        end if
        vis_frame ← OpenCV.cvtColor(vis_frame.get_image(), OpenCV.COLOR
                                                RttB2BttR)
        vis_frame ← OpenCV.cvtColor(vis_frame, OpenCV.COLOR_BttR2RttB)
        return vis_frame
    end function
    created_frame ←frame_from_video(video)
    if the process is parallel then
        terminal print('− − − − − − > parallel')
```

```
buffer_size ← predictor.default_buffer_size
frame_data ← deque()
    for sequence of the frame and frame in created_frame do
        frame data.append(frame)
        predictor.put(frame)
        if sequence of the frame >= buffer size then
            frame ← frame_data.popleft()
            predictions ← predictor.get()
            yield process_predictions(frame, predictions)
        end if
    end for
    while length of frame_data is not null do
        frame ← frame_data.popleft()
        predictions ← predictor.get()
        yield process_predictions(frame, predictions)
    end while
else
    terminal print('— — — — — — > notparallel')
    for each frame in frame_gen do
        visualizer ← TextVisualizer(frame, metadata, instance mode = instance mode)
        yield process_predictions[frame, predictor(frame)]
    end for
end if
end function
```

After correcting the presentation of predicting videos, less important issues of the predicted labels need to be adjusted. The colors for the prediction area, boundaries, and the text in the label use darker colors to present the subject. The observed target is covered in heavy colors, which is hard for users to check the braille characters while inferencing. The text in the label is small, which also brings difficulty for comparing the prediction and the real label and checking whether the prediction fully matches all the characters or symbols in the real label.

Therefore, little changes are made for the function in the script visualizer.py. Changing the original color from default RGB values (0.1,0.2,0.5) into ‘y’, which represents the color yellow. By changing the predicted label from top-left position [0] to position [39], we modified the label to be shown at the bottom-left position, which provides a better view of the target without covering some area inside the box shown in

Figure 3.25. One more change for the function is the font size of the label, by letting the variable multiply by 1.50, which allows the text to be shown more evident while inferencing the objects.

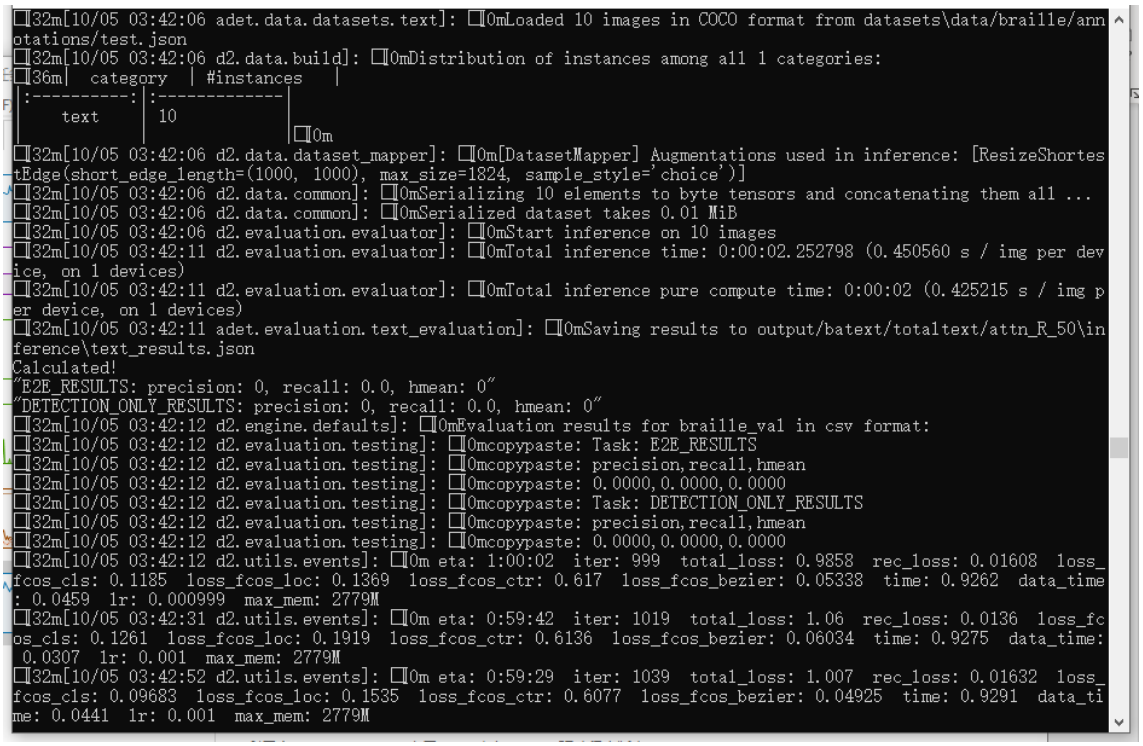

Figure 3.26: The ABCNet training progress represented on the terminal in Windows 10

This project includes two networks, ABCNet and YOLOv5. It is important to set up the proper systems and environment for running the networks. Through official installation structure for both networks, it is possible to implement the networks on Microsoft Windows 10 operating system. By following the installation guidelines, the networks are installed on Windows 10. There will be a virtual environment for each network, which permits the network to run smoothly in their own environment and no conflicts between the two networks. A few tests are required to ensure the functionalities of the networks are working properly. The tests are based on running the training process with a small dataset. After testing the networks, we figure out that the YOLOv5 runs correctly on Windows 10, though ABCNet has an issue while training.

In the beginning, the terminal shows the progress of the training without any issue. It runs smoothly and the values for the losses are shown for every 20 iterations. We set

every 1,000 iterations as the checkpoint. After the first 1,000 iterations, the network processes the evaluation for a temporary state.

In Figure 3.26, the precision, recall, and H-mean rates in evaluation metrics are all represented as 0, including the E2E and detection-only results. In this case, this is impossible for us to understand whether the training process can proceed with no technical issues relevant to the system and running environment. Therefore, we must build the environment for ABCNet using Ubuntu. By following the same instruction to install the network to the Ubuntu system, the checkpoints' evaluation metrics are shown with correct values. Thus, we arrange to run the networks separately on two different systems with suitable environments, as YOLOv5 will be trained using Windows 10, and ABCNet will be trained on Ubuntu.

We choose Python as the programming language for implanting both deep learning methods and training the networks. The framework for running the scripts is PyTorch (Paszke et al., 2019). Regarding the hardware, we mainly operate a laptop with GPU GeForce GTX1060 for the entire project. The size of GPU memory is limited to only 6G bytes, the batch sizes for training the networks are very small. Regarding YOLOv5, the batch size is set as two, while ABCNet is one.

Chapter 4

Results

In this chapter, the results of the experiments will be demonstrated. The results include the training result of ABCNet, the training results of YOLOv5 also encapsulate comparisons.

4.1 Prediction Display of ABCNet

Based on the methods, algorithms, modifications, and resources, we train the networks with their corresponding dataset which all are prepared by ourselves. In this chapter, we show the training results of the networks. The images and videos that are not included in the datasets will be applied for testing the inference performance of the networks. To simulate the situations when the networks are implemented for solving the real-life problem.

Figure 4.1: Comparing the video prediction after using parameterized Bezier curves

Figure 4.1 shows the adjusted inference result compared to the prediction displayed using the original rectangular bounding box. From the image, the results are detailed. The upper and bottom boundary are represented in slightly curved lines. Other two short boundaries are not shown vertically but drawn with certain angles according to the final trained ABCNet. Instead of using a hollow box for the predicted area, the prediction fills the box with color after adjustment. It highlights the area that has been detected, which gives us a benefit while evaluating the prediction, whether it has covered the braille characters nicely without too much unwanted background, more accessible for us to view the actual result.

The original display of the prediction label only shows the score in percentage, same as the left-hand result in Figure 4.1, where 86.0% has shown in the label, which is not sufficient for us to evaluate the result. It is impossible to compare the predicted

characters and the actual characters in the video. Only the score of the detection cannot be utilized to estimate whether some characters in the prediction have been misjudged. Therefore, the update adds the predicted characters into the prediction label. As we see the adjustment results, the predicted characters are shown as “love” with a score of 0.862. Thus, we ensure that the network perfectly predicts the braille word “love” shown in the test video.

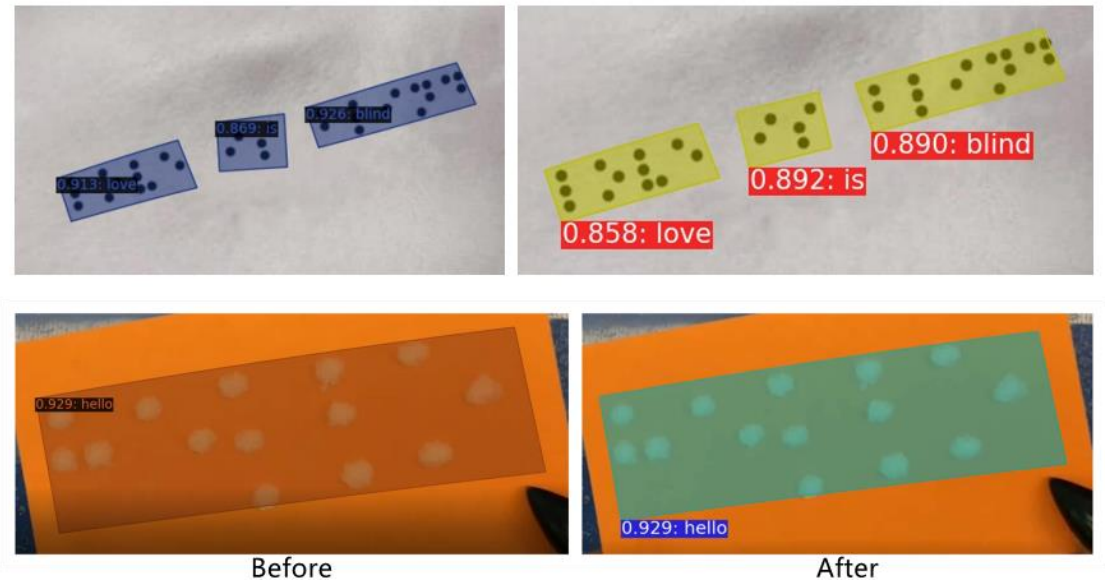

Figure 4.2: Prediction labels before and after the adjustment

Figure 4.2 shows the adjustment of the display and position to the prediction label. For the original display methods shown on the left side, the top-left is prediction when inferencing image and the bottom-left is video prediction. The labels are placed at the top-left corners of the predictions. The labels overlap with the detected area of braille, a part of the braille characters cannot be viewed by the user. We change the position of the prediction label from top-left to bottom-left, which will not intersect with the prediction area in most cases. Furthermore, we enlarge the size of the text of the label. We replace the colors of the prediction, including the label box, text inside the label, and the detected area. More vivid colors make it easier for us to observe the prediction results.

4.2 Result of ABCNet

Table 4.1: Iterations and time costs for training ABCNet

Iteration	Checkpoint period	Time cost/ each iteration	Total training time
100,000	1,000	0.55 sec	15h 21min 23sec

In Section 4.1, we demonstrate the adjustments for the display of the detection generated from ABCNet. In this section, we will describe the training and evaluating results of ABCNet.

Table 4.2: Time consumption for evaluation

Data loading	Inference	Evaluation	Total time cost per image	Total time cost
0.0011 s/per image	0.3 s/per image	0.0006 s/per image	0.3017 s/per image	95 sec

We manage to train the network by 100K iterations with every 1K iterations as a checkpoint. The GPU memory usage for training the network is 2,783MB as the maximum memory during the process. The entire training process takes nearly 16 hours to finish.

For each checkpoint, the network will be evaluated periodically by using the validation set. The 400 images in the validation set are managed into various batches. According to the batch size, which is 1.0, there are 400 batches for validation. The average time costs for the inference at each checkpoint are shown in Table 4.2. For each image, the total evaluation time is 0.3017 seconds. The network takes 95 seconds to finish the evaluation process. Therefore, the average speed for the network to infer the objects we operate is 3.3FPS. Compared to the official inference speed from GitHub, which is the maximum 11.3 FPS, ours is much slower.

Table 4.3: Evaluation metrics of ABCNet

	Precision	Recall	H-mean(F1-score)
E2E	0.89	0.90	0.90
Detection Only	0.98	0.99	0.98

Table 4.3 shows the evaluation metrics of the final trained ABCNet. Regarding E2E detection, the precision of ABCNet is 0.89, recall is 0.90, and the H-mean reaches 0.90. For detection-only, the precision is 0.98, recall is 0.99, and the H-mean is 0.98. The values of the evaluation metrics for detection only are slightly higher than E2E detection, with each metric has an approximate 0.10 difference.

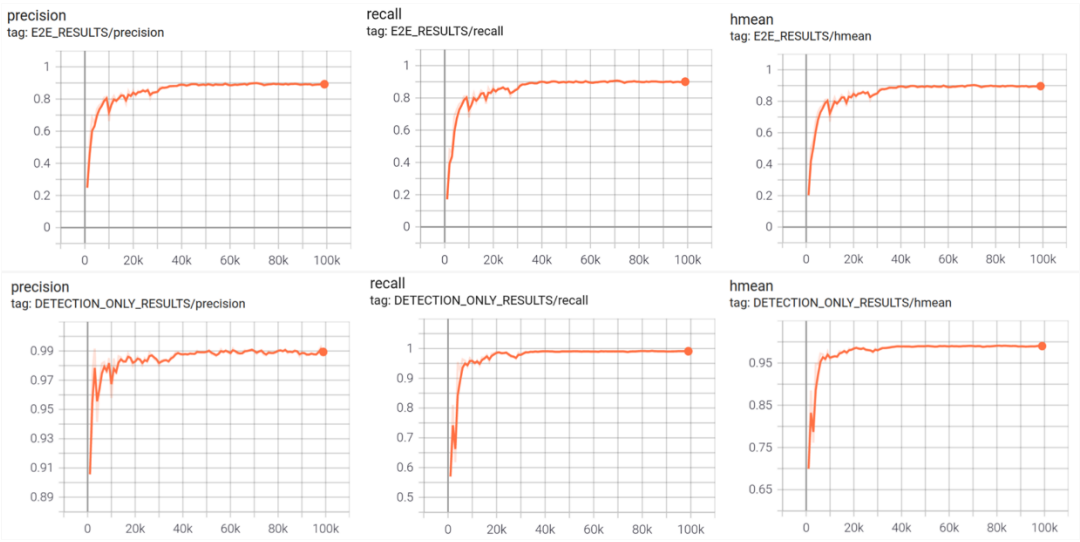

Figure 4.3: The evaluate metrics

Figure 4.3 shows the trends of the evaluation metrics of detection only and E2E detection during the training. All the metrics increase rapidly during the first 10K iterations. The values increase slowly between 10K and 40K iterations. While the training process reaches and exceeds 40K iterations, the metrics keep at relatively stable values with only small fluctuations. Regarding E2E detection, the values of the metrics stopped growing around 0.90, cannot be as close to 1.00 as the metrics.

Figure 4.4: Inferencing images

Figure 4.5: Inferencing videos

Figure 4.4 and Figure 4.5 show the detection results with inferencing images and videos. We choose a yellow color for images to demonstrate the detected braille, the red color is applied to the label box with white text inside. The text is viewed clearer with the red color as the background. Regarding videos, the detected braille is covered in wathet blue. The label box is represented in blue with white text. The prediction results all gain very high confidence scores. None of them is less than 0.70. The highest score in the figures is 0.925 while detecting the braille word “mommy”.

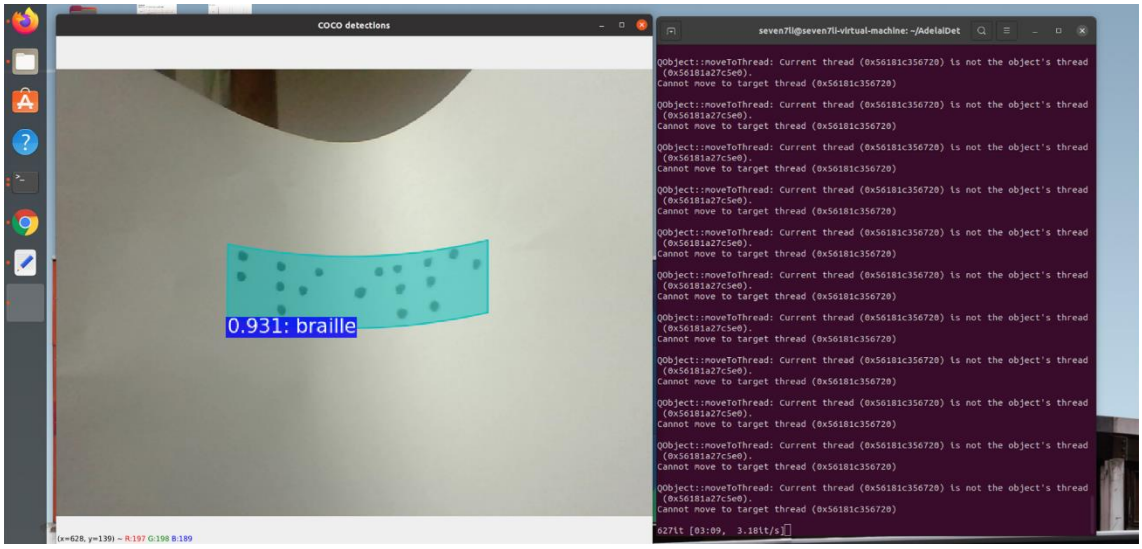

Figure 4.6: Real-time inferencing through webcam

The ABCNet is defined as an E2E scene text detection and recognition method, where it can apply the functionality of real-time text spotting. Therefore, after testing its performance with inferring images and videos, we must test how the network accomplishes the real-time tasks using the webcam. Figure 4.6 is a moment of the network is being inferenced with the braille characters by using webcams. From the image, we test whether the network can detect the arbitrary-shaped braille and draw the boundaries of the area with Bezier curves, we deliberately bend the paper with a hand-written braille word. The result shown in the image is relatively perfect. The braille shown by the webcam is correctly detected and enclosed by the curved box. Inside the box, we can find very little unwanted background.

The confidence score of the prediction is 0.931, which is very high, and the predicted characters all match to the braille characters. The terminal shows on the right side, which is next to the webcam window. It displays the status of the network while inferencing. The inference speed shown in the terminal is 3.18 FPS. One frame is regarded as one iteration for the network. The test proves the ABCNet can detect and classify braille words in plain English in real time, though the inference speed is slow, which cannot satisfy the user if they require a smooth-running state.

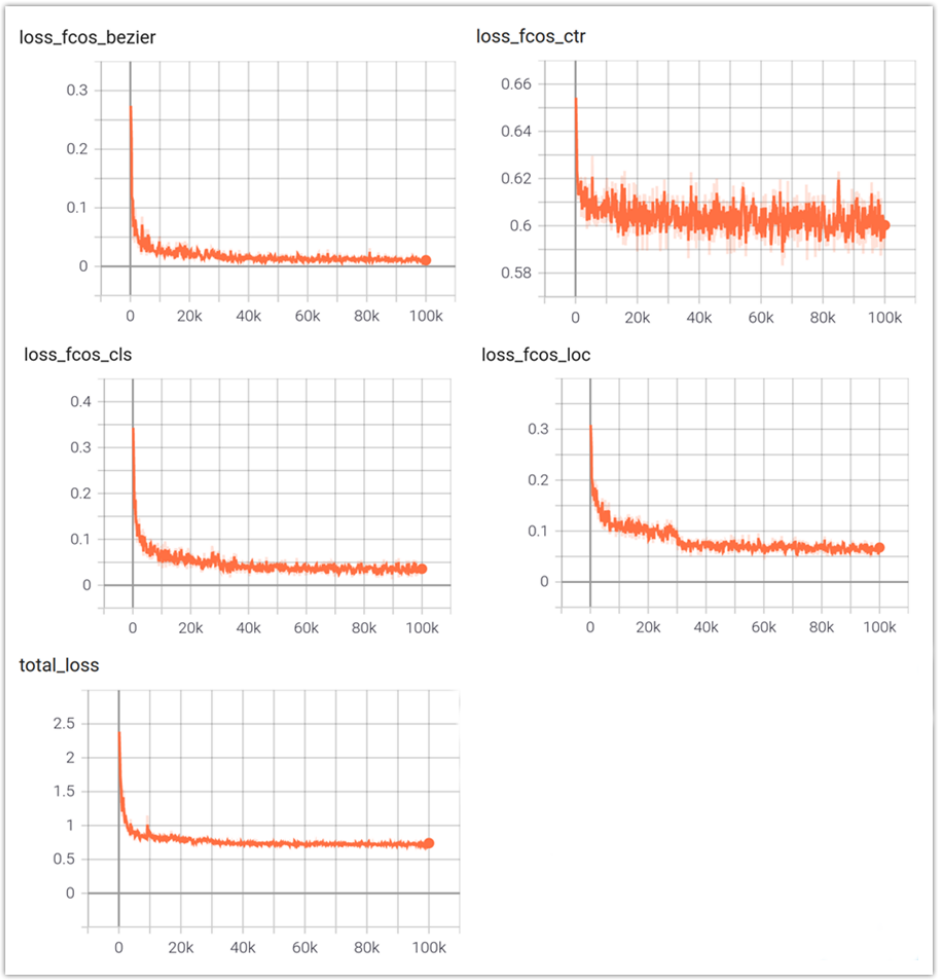

Figure 4.7: The loss curves of the training process

The loss function of ABCNet includes three main parts, FCOS loss, center-ness loss, and Bezier loss. FCOS loss is combined with two loss functions responsible for classification and regression (localization).

Table 4.4: Values of losses at the final training iteration

Bezier loss	Center-ness loss	FCOS classification loss	FCOS Localization loss	Total loss
0.0099	0.6018	0.0368	0.0689	0.7174

Table 4.4 represents the values of each loss after the training. The total loss is 0.7174. All the losses are reduced at a rapid speed at the first 10K iterations. The values

of Bezier loss, classification loss, and localization loss become stable at 40k iterations shown in Figure 4.7. Compared to other losses, though Center-ness loss is reduced very fast at the beginning, it has relatively unstable fluctuation during the entire training process. It also contributes most of the loss value as 0.60.

4.3 Result of YOLOv5x

Table 4.5: Training volume and time consumption

Dataset size	Epoch	Batch per epoch	Time cost per epoch	Total training time
1500	300	600	8.99min	44hr 56min
3000	200	1200	16.37min	54h 33min

We explore the results of YOLOv5x in this chapter. We have trained the network with two datasets of different sizes. The dataset “v5” with 3,000 images is the main dataset. Another dataset contains 1,500 images, collected as the control group to get a comparison result.

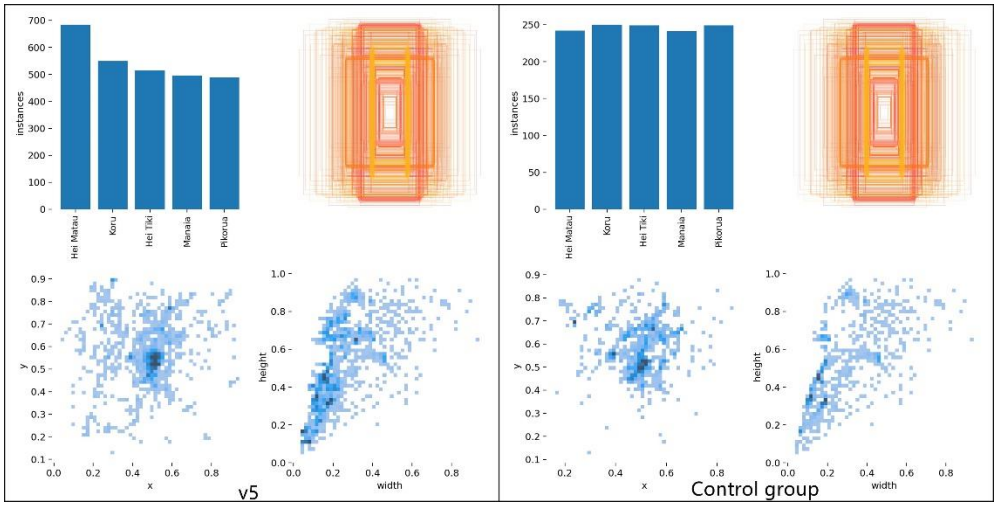

Figure 4.8: Distribution of labels in the datasets

Figure 4.8 represents the labels distribution status in two datasets. In dataset “v5”, the instances in all the classes are around 500. Hei Matau has the largest number of

instances for nearly 700. Most of the instances are captured in the middle of the images. From the scattergram, most of the instances have minor ratios of width between 0 and 0.40, the ratios of heights are between 0.10 and 0.90. Inside the control group, the instances for each class are around 250, which is half of the amounts in “v5”. As same as “v5”, most of the instances in the control group are placed in the middle of the images. Most of the width ratios in the control group are gathered between 0.0 to 0.3. The height ratios are between 0.00 to 0.80.

Table 4.6: Evaluation metrics of YOLOv5x

Dataset	Class	Precision	Recall	mAP@.5	mAP@[.5:.95]
v5	All	0.989	0.98	0.991	0.973
Control Group	All	0.981	0.948	0.976	0.95
v5	Hei Matau	0.994	0.946	0.977	0.954
Control Group	Hei Matau	1	0.877	0.931	0.909
v5	Koru	0.976	0.985	0.995	0.975
Control Group	Koru	0.983	0.972	0.984	0.956
v5	Hei Tiki	1	0.993	0.996	0.975
Control Group	Hei Tiki	1	0.981	0.986	0.964
v5	Manaia	0.974	0.992	0.994	0.987
Control Group	Manaia	0.922	0.983	0.984	0.97
v5	Pikorua	1	0.983	0.995	0.976
Control Group	Pikorua	1	0.929	0.995	0.951

Table 4.5 shows the training volumes and time costs of the training. Regarding

dataset “v5”, we set up 200 epochs training steps, and each epoch has 1,200 batches. The overall time consumption is 54 hours and 33 minutes. For the comparisons, we apply 1,500 images to train the network with 300 epochs, but each batch only has 600 batches, the overall time costs are 44 hours and 56 minutes, respectively, which are less than the training time with dataset “v5”. Although the number of epochs is more than the training with dataset “v5”, the time cost is 10 hours less by using 3,000 images for training.

Table 4.6 shows the results of the training using two datasets. The results include the overall value of precision, recall, mAP@0.5, and mAP@[0.50:0.95]. mAP@0.50 stands for the mean Average Precision (mAP) if the IOU threshold is 0.50, and mAP@[0.50:0.95] represents the average mAP at different thresholds, from 0.50 to 0.95. From Table 4.6, by exploiting the “v5” dataset for the training, the overall precision gets 0.99, recall as 0.98, mAP@0.50 as 0.99, and mAP@[0.50:0.95] as 0.97. Compared to the control group results, all the metrics are slightly higher. While checking through the evaluation metrics for every class of Māori symbol, only the precision of the symbol Hei Matau in the control group is 1.00, which is a little higher than the precision of the main results. The precisions of symbols Hei Tiki and Pikorua all reach 1.0. The recall of Pikorua of the control group gets the lowest score as 0.93.

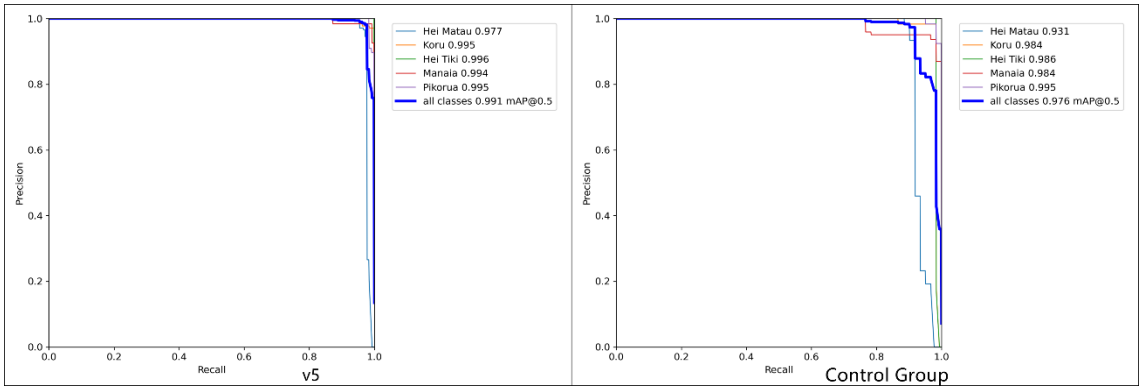

Figure 4.9: The PR (precision *vs* recall) curves

Figure 4.9 represents the PR curves of the two training results. The overall PR curve of the “v5” dataset is able to wrap the PR curve defined by the control group. The

overall precision of “v5” is 0.99, the overall precision of the control group is 0.98, which is slightly lower. The PR curves for each class of “v5” are more compact to each other, and the shapes are closer to all-classes PR curve. While observing the PR curves of the control group, the curves of the classes have more distances between each other. The PR curve of Hei Matau in the control group shows more fluctuation. Once its recall rate reaches around 0.9, the rate of the Hei Matau precision in the control group starts reducing.

Table 4.7: The losses of the training

Dataset	Gpu_mem	Box loss	Object-ness loss	Classification loss	Total loss
v5	3.25G	0.0054	0.0034	0.0004	0.0091
Control group	3.25G	0.0055	0.0034	0.0005	0.0094

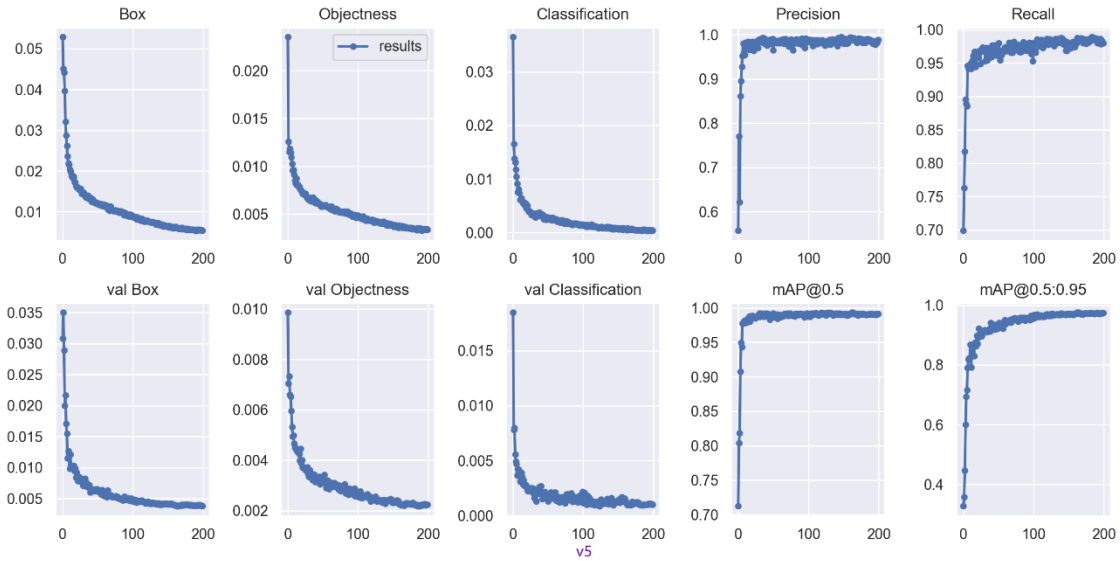

Figure 4.10: The trends of the metrics during training by using dataset “v5”

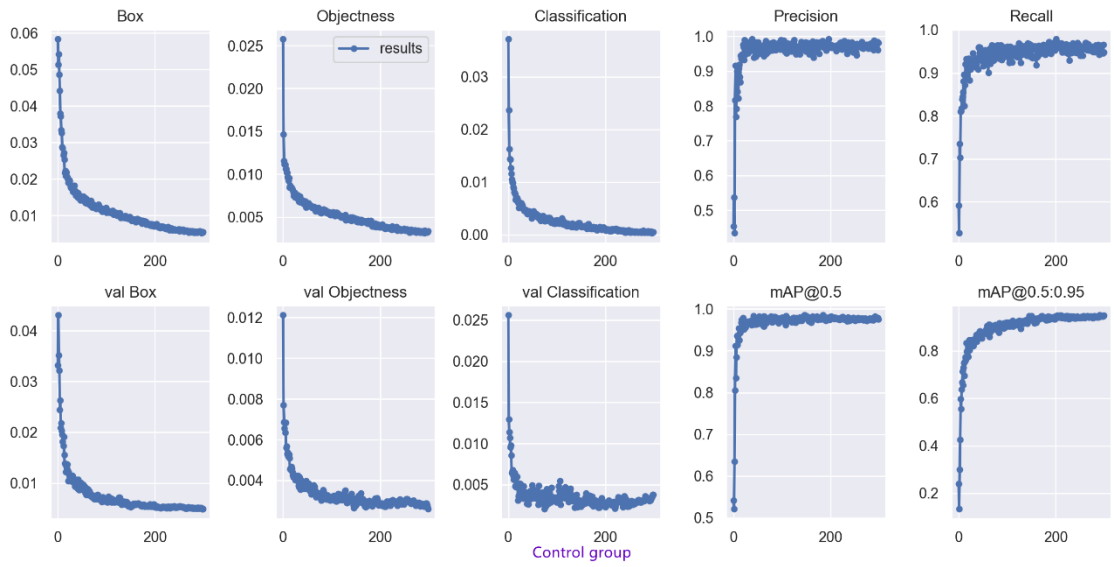

Figure 4.11: The metrics during model training by using control group dataset

In Table 4.7, we show the final loss values. The two training processes all apply 3.25G of GPU memory for training. The total loss trained using "v5" is 9.10×10^{-3}, and the total loss of the control group is 9.40×10^{-3}, with the box loss and the classification loss slightly gaining more loss values than the experiment group. And the object-ness losses are all equal to 3.40×10^{-3}.

Figure 4.10 and Figure 4.11 gather the trends for each metric, including evaluations, training, and validation losses of two training processes. The classification loss of validation of dataset "v5" is reduced more smoothly. In the control group, the classification loss of validation is decreased rapidly at the beginning, which is down to 5.00x10-3. But the loss cannot be minimized in a smooth rhythm. The loss value slightly increases at the end of the training.

Figure 4.12 Inference results on images of YOLOv5x

Figure 4.12 demonstrates the detection results with inferencing images using YOLOv5x. The networks detect all types of Māori symbols with high confidence scores. Most of the scores reach over 0.90. There is one Hei Tiki shown as the image background is detected with a score of 0.48. Although the score is not high, the network still detects the symbols correctly and localizes the position with a bounding box in feasible size.

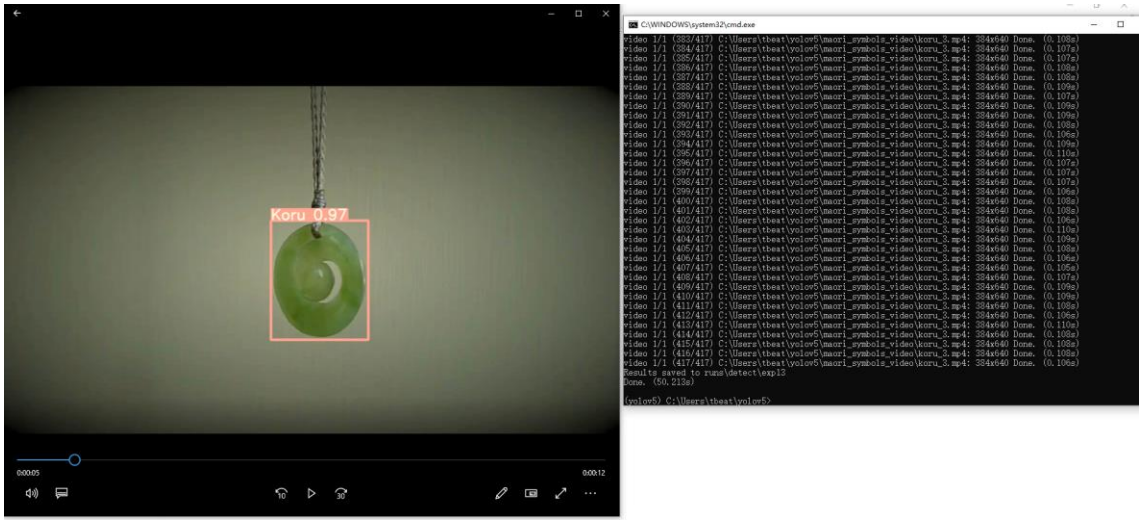

Figure 4.13: Inferencing video using YOLOv5x

The YOLOv5x generates nice results while inferencing the objects in the images. We bring the test further by applying detection using videos. On the left side of Figure 4.13, the generated predicted video clip is shown, the right side is the terminal that execute the inference process.

The detection result in the video is shown in Figure 4.13. The network predicts the symbol Koru correctly with a confidence score of 0.97, which is very high. The boundaries of the predicted bounding box are very close to the Koru symbol. The terminal shows the detection process. The video is detected as frames. As the example in the image above, the network takes around 0.11 seconds to detect one frame of the video. It takes 50.21 seconds to get all 417 frames. Therefore, the approximate inference speed of YOLOv5x is 8-9FPS by our computer. After inferring the videos, we also test the real-time detection using the same webcam like ABCNet.

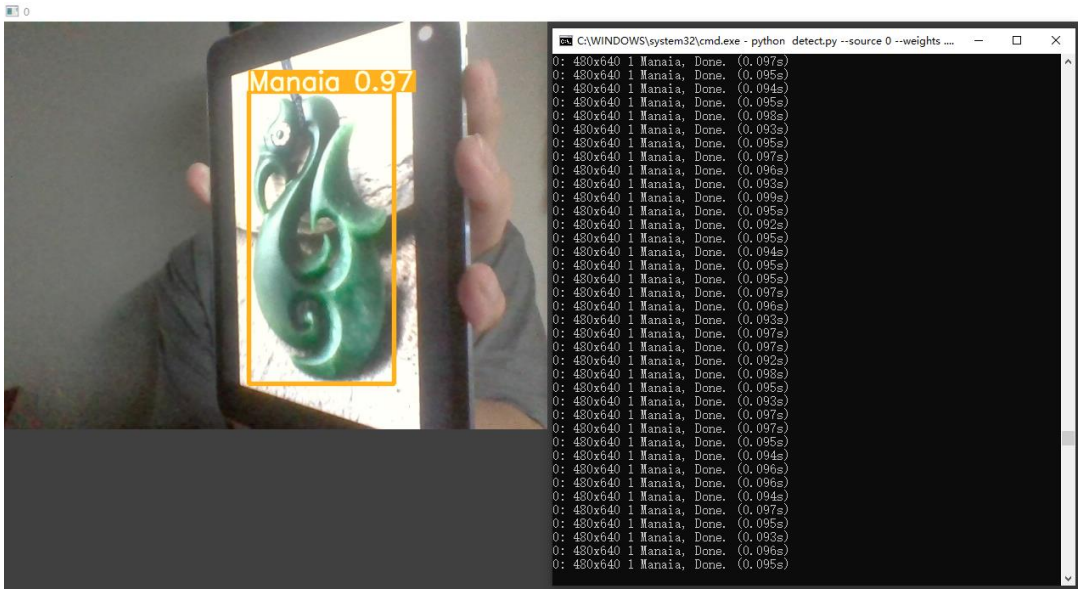

Figure 4.14: Inferencing Māori symbols by real-time webcam

Unfortunately, we do not have the chance to gather Māori symbols as real-life entities. We display the symbol by using a small screen device. Figure 4.14 shows the real-time process. The webcam is shown on the left side, which represents a Manaia symbol. The network detects the symbol into the correct class, with a confidence score of 0.97, which is very high. The symbol is warped tightly with a bounding box. Inside the terminal, the information describes the real-time inferencing status, where number 0 represents the webcam. The status shows the size of the webcam, detection result, and time cost for each frame. The webcam size is 480×640, and the inference speed for each frame is around 9.20×10^{-2} – 9.70×10^{-3} second, which is 10.3FPS - 10.87FPS. It is faster than other videos.

Chapter 5
Analysis and Discussions

In this chapter, the results of the experiment are analyzed and compared. The results are also detailed. The possible reasons for the results will be discussed.

5.1 Analysis

In this chapter, we analyze the results in the previous chapter, and deficient inference results will also be included. The relevant results and analysis will be discussed in the following sections.

5.1.1 ABCNet

The results show how the ABCNet predicts braille characters using images, videos, and a real-time webcam. Within the training schedule, we set 100K iterations for the training. In most situations, the ABCNet detects and classifies the braille characters with high accuracy by using parameterized Bezier curves to draw the prediction box with a less unwanted background. Moreover, the confidence scores are usually high with correct predictions. However, the network cannot fully recognize all the braille characters in a word.

Figure 5.1: Misjudgment braille words

In Figure 5.1, there are two braille words, “happy” and “anniversary” that need to be detected and recognized. Two words are detected and warped nicely with the bounding boxes. The first word “happy” has been recognized correctly without wrong characters, but the score is 0.56, which is low. The second word “anniversary” has been detected,

but the network predicts the word as “annikersny” with two misspellings, though the score is 0.87, which is higher than the first prediction.

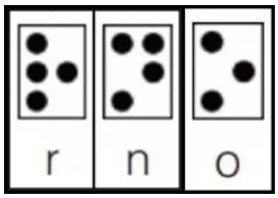

Figure 5.2: Braille character ‘r’, ‘n’, and ‘o’

We observe the braille word “anniversary” in the image. The overall angles of the characters are deflected about 30 degrees counterclockwise related to the vertical direction of the image. However, the positions of the mispredicted characters are parallel to the vertical direction of the image. This might be one of the reasons why the network produces the wrong prediction. Regarding the mispredicted character ‘r’, it has two dots difference to ‘n’. Inside the image, the second dot in the first column is inconspicuous, leading to a misprediction by classifying ‘r’ into ‘n’. As the shape of character ‘o’ is similar to ‘r’ and ‘n’, we assume there is a probability that the network might mispredict ‘r’ into ‘o’.

Figure 5.3: Mis-predicted and undetected braille

In Figure 5.3, the network deals with multitarget detection where there are 6 braille words represented in the image. Half of the words are correctly detected and recognized.

The top-left braille represents “fabulous”, the network only detects five characters, and two of them are correctly classified, which are “ou”. This braille word is rotated approximately 60 degrees clockwise relative to the horizontal direction of the image. Because each braille character is constituted by dots. Unlike English alphabets that are written by connected lines. Therefore, if the position is extremely rotated, an inevitable mistake may occur.

Other two braille characters at the bottom-right represent the same word “the”. These are shorthand braille characters, which show an English word with only one braille character. Our dataset covers only the basic characters and punctuations, and no shorthand characters are included. Therefore, the network cannot detect and recognize the braille in simplified way.

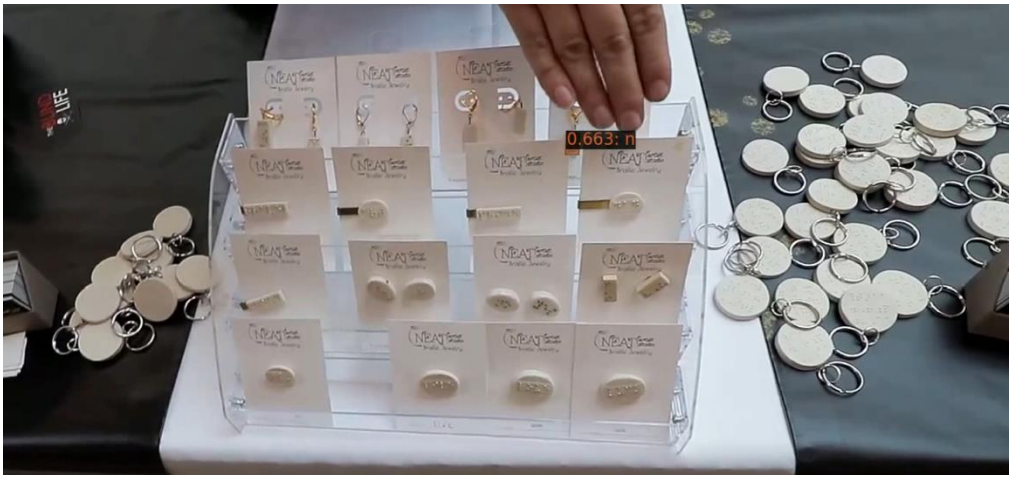

Figure 5.4: Detecting small targets using ABCNet

Figure 5.4 shows the result while ABCNet is utilized to detect small targets. As we see in the image, some pieces of jewelry are decorated using braille. Only one has been detected correctly, with a confidence score of 0.66. The result is unexpected that we expect all the braille to be detected and recognized by the network. The ABCNet applies FPN with ResNet as its backbone, which should be able to handle the detection when the targets are in various sizes.

Figure 5.5: Comparison if the braille is big enough

We assume the braille characters in the image are too small. Most of the dots become inconspicuous. The network might classify the dots as background noises, so the braille cannot be detected. The sizes of braille words in Figure 5.5 are bigger than the words in Figure 5.4. The dots have better clarity. Hence, the network detects all the braille correctly with high scores.

We show the final loss values of each loss function of ABCNet in the result. Among them, the center-ness loss contributes a large portion of loss values to the total loss. The center-ness is designed to suppress the number of low-quality prediction bounding boxes which is generated by calculating their centrality, in order to improve the performance of the network. ABCNet applies BCE loss as the center-ness loss for each prediction. It sums up all the center-ness loss and takes the mean value as the overall center-ness loss. The center-ness loss is also volatile during the training compared to other losses. We believe that the network will generate multiple candidate prediction boxes during the detection, most of which are low-quality. The ABCNet implements the BCE loss for comparing the center-ness values of each prediction with the target. The low-quality boxes will gain more loss value, increasing the final mean value of center-ness loss. Therefore, the center-ness loss fluctuated during the training and received a large loss value at the end.

Compared with other real-time spotting methods, the real-time inference speed of ABCNet is very slow which only reach 3.18 FPS which mentioned in Chapter 4. Although the parameterized Bezier curves only generate a small amount of computation overhead when calculating the control points of the Bezier curves, which should not have too much impact on the inference speed. However, if the BezeirAlign takes effect, the calculation is at the pixel level, multiple positions need to be calculated to obtain the required values. It might need a large amount of computation. Therefore, the number of frames detected by the network in a second will be relatively small, and the detection speed will be slower than other methods.

5.1.2 YOLOv5

We have two sets of results of YOLOv5 that are trained by using two different datasets, the dataset “v5” for our experiments. The evaluation metrics of the results are both in very high scores. In the control group, the precision of class Hei Matau, Hei Tiki, and Pikorua all reach 1.00. The precision of class Hei Tiki and Pikorua in the experiment group is also up to 1.00.

The network of the control group has been trained for 300 epochs. The network has been trained 200 epochs with the experimental group. But the overall values of the evaluation metrics of the experiment group are higher than the control group. It achieves better results with a larger dataset and fewer training iterations. From the PR curves shown in Figure 4.9, we see that all PR curve of the experiment group fully wraps the curve of the control group, which means that the network performance of the experiment group is better than the control group network. Therefore, if the network has the same structures and hyper-parameters in various experiments, the one employing the largest dataset for the training can get the best result with less training.

Figure 5.6: Multitarget detection with one object has not been detected

After being trained, YOLOv5 has shown robust results while proceeding with object detection tasks, which can detect and recognize the five Māori symbols with high accuracy in most cases. For dataset “v5”, we have collected 600 images for each symbol, which is relatively equal. Compared to the “v5” dataset with dataset “ABC”, there is no case that a class with a very small proportion. Because there are only 5 types of symbols in “v5”, which is easier for us to manage the data balance. A more balanced dataset is applied to train the model so as to perform better on inferencing targets. Although the number of images of each class is equal, most of the symbols in the images are represented as pieces of jewelry, jade carvings, and wood carvings. However, we only collect a few Hei Matau shown as tattoos and paintings with more abundant patterns.

Figure 5.6 shows the result if YOLOv5 is applied to multitarget detection on Māori symbols. There are four symbols in the image that are drawn on stones. Three of them were detected, while the Hei Matau at the top has not been detected. We checked the

dataset and found that only 58 images on Hei Matau are represented as tattoos and paintings, which is not enough to enhance the data variety. Therefore, we assume this is why the proposed network cannot detect the Hei Matau painting in the image, as only a few relevant images are collected. The network does not have enough data to learn the relevant features.

Figure 5.7: The detection result shows the symbols are misjudged.

Regarding the jade and wood carving items of Māori symbols, a symbol has multiple forms of expression. It incorporates visual features from other symbols. For example, the Hei Mautau jades in Figure 5.7, the shape of the tips is similar to the tail part of Mania. Also, the overall structure of the two symbols is similar. Thus, the network will have a probability to detect and recognize the symbols as Manaia. Similarly, in the bottom-right of the image, there are two Hei Matau jade pieces combined with the spiral feature of the symbol Koru. The pieces of jewelry are occluded by the upper boxes, which are not fully displayed. Thus, the network classifies the symbols as Koru.

5.2 Discussion

In our experiment, we applied ABCNet for braille spotting and YOLOv5 for Māori symbol detection. Both methods take advantage of relatively complex deep network structures with many layers. The residual structure is implemented in both networks to

eliminate the problem of network degradation and gradient disappearance. The FPN structures are cited in the networks to enhance the ability to detect multiscale network targets. The training results show that the networks can efficiently detect and classify the targets with high accuracies in different scenarios and have no problem dealing with multitarget detection by using images and videos from webcams. The evaluation metrics of the networks all reach very high scores.

In the previous section, we analyzed erroneous results in the experiment. To apply ABCNet for braille detection and recognition, we figure out that because of the particularity of braille itself, the braille character is represented by single or multiple dots organized with a certain pattern to match with the corresponding English alphabet or numerous letters.

Compared to braille, English alphabets or unique characters utilized by other countries are represented using continuous lines. The characters have unique characteristics, which can be distinguished easily from other objects or backgrounds. Although braille characters are composed of dots with orders, there is a probability that the proposed network is confused with other circular objects while inferencing. For example, the network might have a possibility to recognize the vertical traffic lights as the braille character ‘L’ because it is formed by three dots vertically. Also, if the braille is too small, it might become vague in the image. And the dots of the braille may be regarded as noises of the background.

YOLOv5 is a very powerful object detection method, which has a good performance in our experiment. In order to compare the YOLOv5 with previous YOLO series methods, it applies the residual structure to the Backbone and Neck. While the network is deeper, the ability of feature extraction and feature fusing of the network will not be reduced. The Focus structure in the Backbone affects the width of the feature map, the learning ability is enhanced. The PANet is applied to the Neck structure to enhance the feature extraction ability further. The evaluation metrics of the trained network all reach more than 0.90, and the metrics even is up to 1.00. YOLOv5 fully

utilizes the features of the Māori symbols in the training set. Throughout the experiment al results, we understand that the dataset provides a big impact on the training process. A dataset with more images effectively reduces the time consumption of the training while maintaining accuracy and even getting better performance. If a type of targets have multiple expressions, we need to collect enough data for each type of targets, that the network can learn the feature more adequately.

5.3 Limitations

In the previous chapter, we show the results of our experiment. The overall results satisfy our expectations. There are still limitations during the preparation and experiment. Due to the special circumstances and restrictions by the travel policies, we cannot collect the data of Māori symbols from its region country and the territories where the symbols are popular. Similarly, related to investigate and collect the braille data, due to the different versions of braille used in different regions, we cannot find and collect images and videos of English braille in public areas where we conduct the experiment, such as English braille documents in libraries, braille interpretations besides the notice, etc.

Hence, the method of collecting data is limited by searching relevant publicly available images and videos through the Internet. In methodology, we show the distribution of the characters in the dataset ABC. Numerous characters appear very rarely. The reason is that these characters are seldom shown in the images that can be found on the Internet. For example, the uppercase ‘X’ is not included in the dataset. The lowercase ‘x’ appears less in the dataset. The data distribution in dataset ABC is not like the Māori symbols dataset “v5” where each class in “v5” has 600 images, the amounts of instances for each type only have minor differences. Thus, applying dataset ABC for the training might negatively affect the performance of the network. Also, the shorthand braille characters are not included in the dataset. The recognition ability of ABCNet is limited as it only can detect the general braille characters and punctuations.

Regarding the dataset “v5”, though the number of images of each class is relatively equal, with 600 images per class, most of Māori symbols are represented as jewelry and sculptures. In contrast, only a small number of the symbols are represented as tattoos and paintings. This may be the reason why the trained YOLOv5 gives better performance while detecting the symbol shown as jewelry or sculpture. This might make a misjudgment or cannot detect the symbol if it is a tattoo or art of painting. Due to limited time and finite ways of data collection, we cannot obtain more images for the datasets, which affects the final performance of the proposed networks.

Regarding hardware, because the computer to run the experiment only has a single 6G GPU, the training processes for the networks are very time-consuming. We explain the inference speeds of ABCNet and YOLOv5 on our computer are 3 PFS and 10FPS, respectively, while the official inference speeds are 11 PFS and 30-140 FPS. We assume the main reason is a big gap between the hardware and the software, so our experiment cannot get a similar speed.

Chapter 6
Conclusion and Future Work

In this chapter, we summarize deep learning methods that we selected for this thesis. We explore how the methods have experimented with the results and the deficiency of the experiments. We will discuss the plan of what we need to improve in the future.

6.1 Conclusion

The main purpose of this thesis is to detect and recognize special symbols/characters (braille characters and Māori symbols) by applying appropriate deep learning methods. We expound on the history of CNN, a deep learning method in the literature review. We understand that the residual and bottleneck structure from ResNet gives the networks an opportunity to increase their depth without reducing the performance. R-CNN series methods as two-stage methods which adopt the deep learning methods for object detection tasks. YOLO significantly improves inference speed as a one-stage method and allows the network to proceed with detection in real-time. We choose ABCNet and YOLOv5 as the methods among various deep learning models. Both methods are relatively novel and have good detection performance.

ABCNet, an E2E scene text spotting method, applies Bezier to detect and recognize text ingeniously. Most text detection methods involve square or rectangular bounding box to represent the detected target. ABCNet implements parameterized Bezier curve to generate the prediction box for arbitrary-shape text in natural scenes and does not generate too much computation cost, which will not negatively affect the network. The prediction result of ABCNet is represented in a curved text box, without too much unwanted background. The angle of rotation, degree of distortion of the target can be shown more clearly.

YOLOv5 integrates a variety of excellent structures, such as combing the residual structure and SCP structure into BottleNeckCSP and applying PANet as its Neck structure. It makes use of various tricks for the input, which includes self-adaptive anchors. We received a better training result by using the genetic algorithm of natural inspired computing to adjust the preset anchors. There is also a trick of self-adaptive image rescaling to reduce the padding and improve the detection speed of the network. Regarding outputs, YOLOv5 provides GIoU, DIoU, and CIoU. We apply CIoU to our experiment for a better training result. The experimental results of YOLOv5 prove its excellent ability of object detection.

Throughout the experiments, we prove that ABCNet and YOLOv5 perform well in handling detection tasks. However, due to the resource constraints, our training datasets have limited data, we cannot get more high-quality data through other ways instead of the Internet. The dataset ABC only contains general braille characters corresponding to single English letters and punctuations, not related to a single braille character representing multiple letters or an English word. Therefore, the trained ABCNet cannot detect the braille in shorthand form. From the experiment results of YOLOv5, we get a conclusion about the dataset. The quality of the dataset, including a number of images and a diversity of the representation of instances, affects the time cost of the training and training result. The bigger dataset effectively reduces the time consumption of the training process while improving the inference ability of the network.

6.2 Future Work

Based on our experiment results and analysis, we believe that the dataset is not sufficient. The networks cannot detect the target or even make a misjudgment in some situations. We will collect more data in the future. For braille images, we will collect the braille data in abbreviated form to enhance the ability of detection and recognition of ABCNet. More data of high resolution with small braille characters will be added to the dataset to improve the network ability when detecting small targets. We will add more images to the Māori symbols dataset so as to enrich the different expressions of the symbols. More new symbols also will be added in the future. We plan to eliminate the hardware obstacle, which limits the speed, by upgrading the GPU and CPU of the computer. We hope the networks can produce a better performance on real-time detection and recognition with the changes.

References

An, N., Yan, W. (2021) Multitarget tracking using Siamese neural networks. *ACM Transactions on Multimedia Computing, Communications and Applications*.

Azzeh, J., Zahran, B., & Alqadi, Z. (2018). Salt and pepper noise: Effects and removal. *JOIV: International Journal on Informatics Visualization*, *2*(4), 252-256.

Baek, Y., Lee, B., Han, D., Yun, S., & Lee, H. (2019). Character region awareness for text detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 9365-9374).

Bjorck, J., Gomes, C., Selman, B., & Weinberger, K. Q. (2018). Understanding batch normalization. *arXiv preprint arXiv:1806.02375*.

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). YOLOv4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*.

Bodla, N., Singh, B., Chellappa, R., & Davis, L. S. (2017). Soft-NMS: Improving object detection with one line of code. *In IEEE International Conference on Computer Vision (ICCV)*, 5562-5570.

Chen, H., Sun, K., Tian, Z., Shen, C., Huang, Y., & Yan, Y. (2020). BlendMask: Top-down meets bottom-up for instance segmentation. In *IEEE/CVF CVPR* (pp. 8573-8581).

Ch'ng, C. K., & Chan, C. S. (2017). Total-text: A comprehensive dataset for scene text detection and recognition. In *IAPR International Conference on Document Analysis and Recognition (ICDAR)* (Vol. 1, pp. 935-942). IEEE.

Choi, J.-w., Curry, R., & Elkaim, G. (2008). Path planning based on Bézier curve for autonomous ground vehicles. In *Advances in Electrical and Electronics Engineering-IAENG Special Edition of the World Congress on Engineering and Computer Science 2008* (pp. 158-166).

Chowdhury, G. G. (2003). Natural language processing. *Annual Review of Information Science and Technology*, *37*(1), 51-89.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3),

273-297.

Cui, W., Yan, W. (2016) A scheme for face recognition in complex environments. *International Journal of Digital Crime and Forensics* (IJDCF) 8 (1), 26-36

Danielsson, P.-E. (1980). Euclidean distance mapping. *Computer Graphics and Image Processing*, *14*(3), 227-248.

De Boer, P.-T., Kroese, D. P., Mannor, S., & Rubinstein, R. Y. (2005). A tutorial on the cross-entropy method. *Annals of Operations Research*, *134*(1), 19-67.

Domingos, P. (2000). A unified bias-variance decomposition for zero-one and squared loss. *AAAI/IAAI*, *2000*, 564-569.

Dytso, A., & Poor, H. V. (2020). Estimation in Poisson noise: Properties of the conditional mean estimator. *IEEE Transactions on Information Theory*, *66*(7), 4304-4323.

Er, M. J., Wu, S., Lu, J., & Toh, H. L. (2002). Face recognition with radial basis function (RBF) neural networks. *IEEE Transactions on Neural Networks*, *13*(3), 697-710.

Ezhilarasi, R., & Varalakshmi, P. (2018). Tumor detection in the brain using Faster R-CNN. In *International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC) I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC), 2018 2nd International Conference on* (pp. 388-392).

Fu, Y., Yan, W. (2021) Fruit freshness grading using deep learning. *Springer Nature Computer Science.*

Fukushima, K., & Miyake, S. (1982). Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and Cooperation in Neural Nets* (f ed., pp. 267-285). Springer.

Gal, Y., & Ghahramani, Z. (2016). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. *ArXiv*, *abs/1506.02142*.

Gao, X., Nguyen, M., Yan, W. (2021) Face image inpainting based on generative adversarial network. *International Conference on Image and Vision Computing New Zealand*

Gehring, J., Miao, Y., Metze, F., & Waibel, A. (2013). Extracting deep bottleneck

features using stacked auto-encoders. In *IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 3377-3381).

Gilbert, C. D., & Wiesel, T. N. (1992). Receptive field dynamics in adult primary visual cortex. *Nature*, *356*(6365), 150-152.

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

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

Gonzalez, J., & Yu, W. (2018). Non-linear system modeling using LSTM neural networks. *IFAC-PapersOnLine*, *51*(13), 485-489.

Gowdra, N., Sinha, R., MacDonell, S., Yan, W. (2021) Maximum categorical cross entropy (MCCE): A noise-robust alternative loss function to mitigate racial bias in convolutional neural networks (CNNs) by reducing overfitting. *Pattern Recognition*.

Grave, E., Joulin, A., Cissé, M., Grangier, D., & Jégou, H. (2017). Efficient softmax approximation for GPUs. *ArXiv, abs/1609.04309*.

Graves, A., Fernández, S., Gomez, F., & Schmidhuber, J. (2006). Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *International Conference on Machine Learning* (pp. 369-376).

Gu, Q., Yang, J., Yan, W., Li, Y., Klette, R. (2017) Local Fast R-CNN flow for object-centric event recognition in complex traffic scenes. In *Pacific-Rim Symposium on Image and Video Technology*, 439-452.

Hao, W., & Zhili, S. (2020). Improved mosaic: Algorithms for more complex images. In *Journal of Physics: Conference Series* (Vol. 1684, pp. 012094).

Hazewinkel, M. (1997). *Encyclopaedia of Mathematics: Supplement* (Vol. 1). Springer Science & Business Media.

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. In *IEEE ICCV* (pp. 2961-2969).

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(9), 1904-1916.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770-778.

He, Y., Zhu, C., Wang, J., Savvides, M., & Zhang, X. (2019). Bounding box regression with uncertainty for accurate object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 2888-2897).

Hecht-Nielsen, R. (1992). Theory of the backpropagation neural network. In *Neural Networks for Perception* (pp. 65-93). Elsevier.

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

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In *IEEE CVPR* (pp. 4700-4708).

Huang, Z., Xu, W., & Yu, K. (2015). Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

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

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. *arXiv preprint arXiv:1602.07360*.

Ji, H., Liu, Z., Yan, W., Klette, R. (2019) Early diagnosis of Alzheimer's disease based on selective kernel network with spatial attention. In *Asian Conference on Pattern Recognition* 2 (1), 503-515

Jocher, G., Nishimura, K., Mineeva, T., & Vilariño, R. (2020). YOLOv5. *Code repository https://github. com/ultralytics/yolov5*.

Kannadaguli, P. (2020). FCOS based human detection system using thermal imaging for UAV-based surveillance applications. In *IEEE Bombay Section Signature*

Conference (IBSSC) (pp. 79-83)

Khalil, Y. H., & Mouftah, H. T. (2021). Integration of motion prediction with end-to-end latent RL for self-driving vehicles. In *International Wireless Communications and Mobile Computing* (IWCMC) (pp. 1111-1116).

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, *25*, 1097-1105.

Lambert, J. (2009). Maori symbolism: The enacted marae curricuslum. *WINHEC: International Journal of Indigenous Education Scholarship* (1), 29-38.

Laroca, R., Severo, E., Zanlorensi, L. A., Oliveira, L. S., Gonçalves, G. R., Schwartz, W. R., & Menotti, D. (2018). A robust real-time automatic license plate recognition based on the YOLO detector. In *International Joint Conference on Neural Networks (IJCNN)* (pp. 1-10).

Le, R., Nguyen, M., Yan, W. (2020) Machine learning with synthetic data – A new way to learn and classify the pictorial augmented reality markers in real-time. In *International Conference on Image and Vision Computing New Zealand*

Le, R., Nguyen, M., Yan, W. (2021) Training a convolutional neural network for transportation sign detection using synthetic dataset. In *International Conference on Image and Vision Computing New Zealand*

Le, R., Nguyen, M., Yan, W., (2021) Augmented reality and machine learning incorporation using YOLOv3 and ARKit. *Applied Sciences*

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

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324.

Lee, D. D., & Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, *401*(6755), 788-791.

Lee, Y., & Park, J. (2020). Centermask: Real-time anchor-free instance segmentation. In *IEEE/CVF CVPR* (pp. 13906-13915).

Li, B., Yan, J., Wu, W., Zhu, Z., & Hu, X. (2018). High performance visual tracking with Siamese region proposal network. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8971-8980.

Li, C., Yan, W. (2020) Gait recognition using deep learning. *Handbook of Research on Multimedia Cyber Security, 214-226.*

Li, C., Yan, W. (2021) Braille recognition using deep learning. In *International Conference on Control and Computer Vision*.

Li, F., Zhang, Y., Yan, W., Klette, R. (2017) Adaptive and compressive target tracking based on feature point matching. In *International Conference on Pattern Recognition (ICPR)*

Li, H., Wang, P., & Shen, C. (2017). Towards end-to-end text spotting with convolutional recurrent neural networks. In *IEEE ICCV* (pp. 5238-5246).

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

Liang, S., Yan, W. (2022) Multilingual speech recognition based on the end-to-end framework. *Multimedia Tools and Applications.*

Lin, K., Yang, H.-F., Hsiao, J.-H., & Chen, C.-S. (2015). Deep learning of binary hash codes for fast image retrieval. *In IEEE International Symposium on Antennas & Propagation & USNC/URSI National Radio Science Meeting*, 27-35.

Lin, T.-Y., Dollár, P., Girshick, R. B., He, K., Hariharan, B., & Belongie, S. J. (2017). Feature pyramid networks for object detection. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 936-944.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. In *IEEE International Conference on Computer Vision* (pp. 2980-2988).

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. In *European Conference on Computer Vision* (pp. 740-755).

Liu, S., Qi, L., Qin, H., Shi, J., & Jia, J. (2018). Path aggregation network for instance segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*

(pp. 8759-8768).

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

Liu, W., Chen, C., & Wong, K.-Y. K. (2018). Char-net: A character-aware neural network for distorted scene text recognition. In *AAAI Conference on Artificial Intelligence*.

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

Liu, X, Yan, W., Kasabov, N. (2020) Vehicle-related scene segmentation using CapsNets. In *International Conference on Image and Vision Computing New Zealand*

Liu, X., Yan, W. (2021) Traffic-light sign recognition using Capsule network. *Springer Multimedia Tools and Applications*

Liu, X., Liang, D., Yan, S., Chen, D., Qiao, Y., & Yan, J. (2018). FOTS: Fast oriented text spotting with a unified network. In *IEEE CVPR* (pp. 5676-5685).

Liu, Y., Chen, H., Shen, C., He, T., Jin, L.-W., & Wang, L. (2020). ABCNet: Real-time scene text spotting with adaptive Bezier-curve network. *In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 9806-9815.

Liu, Y., & Jin, L. (2017). Deep matching prior network: Toward tighter multi-oriented text detection. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1962-1969).

Liu, Y., Jin, L., Zhang, S., & Zhang, S. (2017). Detecting curve text in the wild: New dataset and new solution. *ArXiv, abs/1712.02170*.

Liu, Y., Wen, Q., Chen, H., Liu, W., Qin, J., Han, G., & He, S. (2020). Crowd counting via cross-stage refinement networks. *IEEE Transactions on Image Processing*, *29*, 6800-6812.

Liu, Z., Yan, W., Yang, B. (2018) Image denoising based on a CNN model. In *International Conference on Control, Automation and Robotics* (ICCAR)

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for

semantic segmentation. In *IEEE CVPR* (pp. 3431-3440).

Lorentz, G. G. (2013). *Bernstein Polynomials*. American Mathematical Soc.

Lu, J., Shen, J., Yan, W., Boris, B. (2017) An empirical study for human behavior analysis. *International Journal of Digital Crime and Forensics* 9 (3), 11-17

Lu, J., Nguyen, M., Yan, W. (2018) Human behavior recognition using deep learning. In *IEEE International Conference on Advanced Video and Signal Based Surveillance* (AVSS).

Lu, J., Nguyen, M., Yan, W. (2020) Comparative evaluations of human behavior recognition using deep learning. *Handbook of Research on Multimedia Cyber Security, 176-189.*

Lu, J., Nguyen, M., Yan, W. (2020) Human behavior recognition using deep learning. In *International Conference on Image and Vision Computing New Zealand.*

Lu, J., Nguyen, M., Yan, W. (2021) Sign language recognition from digital videos using deep learning methods. *International Symposium on Geometry and Vision.*

Lu, L., Shin, Y., Su, Y., & Karniadakis, G. E. (2019). Dying ReLU and initialization: Theory and numerical examples. *arXiv preprint arXiv:1903.06733.*

Luo, W., Li, Y., Urtasun, R., & Zemel, R. (2016). Understanding the effective receptive field in deep convolutional neural networks. In *International Conference on Neural Information Processing Systems* (pp. 4905-4913).

Luo, Z., Nguyen, M., Yan, W. (2021) Sailboat detection based on automated search attention mechanism and deep learning models. In *International Conference on Image and Vision Computing New Zealand.*

Ma, X., W Yan, W. (2021) Banknote serial number recognition using deep learning. *Springer Multimedia Tools and Applications.*

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Berkeley Symposium on Mathematical Statistics and Probability* (Vol. 1, pp. 281-297).

Malhotra, P., Vig, L., Shroff, G., & Agarwal, P. (2015). Long short-term memory networks for anomaly detection in time series. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning,*

(ESANN) (Vol. 89, pp. 89-94).

Mao, X., Li, Q., Xie, H., Lau, R. Y., & Wang, Z. (2016). Multiclass generative adversarial networks with the L2 loss function. *arXiv preprint arXiv:1611.04076*, *5*, 1057-7149.

Mead, H. M. (2016). *Tikanga Maori (revised edition): Living by Maori values*. Huia publishers.

Mehtab, S., Yan, W. (2021) FlexiNet: Fast and accurate vehicle detection for autonomous vehicles-2D vehicle detection using deep neural network. In *International Conference on Control and Computer Vision*

Mehtab, S., Yan, W., Narayanan, A. (2021) 3D Vehicle detection using cheap LiDAR and camera sensors. In *International Conference on Image and Vision Computing New Zealand.*

Mehtab, S., Yan, W. (2022) Flexible neural network for fast and accurate road scene perception. *Multimedia Tools and Applications.*

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., & Khudanpur, S. (2010). Recurrent neural network-based language model. In *Interspeech* (Vol. 2, pp. 1045-1048).

Nagi, J., Ducatelle, F., Di Caro, G. A., Cireşan, D., Meier, U., Giusti, A., Nagi, F., Schmidhuber, J., & Gambardella, L. M. (2011). Max-pooling convolutional neural networks for vision-based hand gesture recognition. In *IEEE International Conference on Signal and Image Processing Applications (ICSIPA)* (pp. 342-347).

Naranjo-Alcazar, J., Perez-Castanos, S., Martín-Morató, I., Zuccarello, P., & Cobos, M. (2019). On the performance of residual block design alternatives in convolutional neural networks for end-to-end audio classification. *ArXiv*, *abs/1906.10891*.

Qin, Z., W Yan, W. (2021) Traffic-sign recognition using deep learning. In *International Symposium on Geometry and Vision*.

Oruç, H., & Phillips, G. M. (2003). q-Bernstein polynomials and Bézier curves. *Journal of Computational and Applied Mathematics*, *151*(1), 1-12.

Pan, C., Yan, W. (2018) A learning-based positive feedback in salient object detection.

In *International Conference on Image and Vision Computing New Zealand*.

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

Pan, C., Liu, J., Yan, W., Zhou, Y. (2021) Salient object detection based on visual perceptual saturation and two-stream hybrid networks. *IEEE Transactions on Image Processing*.

Pang, J., Chen, K., Shi, J., Feng, H., Ouyang, W., & Lin, D. (2019). Libra r-cnn: Towards balanced learning for object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 821-830).

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., & Antiga, L. (2019). PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing systems*, *32*, 8026-8037.

Popel, M., Tomkova, M., Tomek, J., Kaiser, Ł., Uszkoreit, J., Bojar, O., & Žabokrtský, Z. (2020). Transforming machine translation: A deep learning system reaches news translation quality comparable to human professionals. *Nature Communications*, 11(1), 1-15.

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

Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, faster, stronger. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6517-6525.

Ren, S., He, K., Girshick, R. B., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*, 1137-1149.

Ren, Y., Nguyen, M., Yan, W. (2018) Real-time recognition of series seven New Zealand banknotes. *International Journal of Digital Crime and Forensics* (IJDCF) 10 (3), 50-66

Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., & Savarese, S. (2019). Generalized intersection over union: A metric and a loss for bounding box

regression. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 658-666).

Rezende, E., Ruppert, G., Carvalho, T., Ramos, F., & De Geus, P. (2017). Malicious software classification using transfer learning of ResNet-50 deep neural network. In *IEEE International Conference on Machine Learning and Applications (ICMLA)* (pp. 1011-1014)

Ridnik, T., Lawen, H., Noy, A., Ben Baruch, E., Sharir, G., & Friedman, I. (2021). TResNet: High performance GPU-dedicated architecture. In *IEEE/CVF Winter Conference on Applications of Computer Vision* (pp. 1400-1409).

Russell, B. C., Torralba, A., Murphy, K. P., & Freeman, W. T. (2008). LabelMe: A database and web-based tool for image annotation. *International Journal of Computer Vision*, *77*(1-3), 157-173.

Russo, F. (2003). A method for estimation and filtering of Gaussian noise in images. *IEEE Transactions on Instrumentation and Measurement*, *52*(4), 1148-1154.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, *61*, 85-117.

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

Shen, Y., Yan, W. (2018) Blind spot monitoring using deep learning. In *International Conference on Image and Vision Computing New Zealand* (IVCNZ)

Shi, B., Bai, X., & Yao, C. (2016). An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(11), 2298-2304.

Shrivastava, A., Gupta, A., & Girshick, R. (2016). Training region-based object detectors with online hard example mining. In *IEEE CVPR* (pp. 761-769).

Smidstrup, S., Pedersen, A., Stokbro, K., & Jónsson, H. (2014). Improved initial guess for minimum energy path calculations. *The Journal of Chemical Physics*, *140*(21), 214106.

Song, C., He, L., Yan, W., Nand, P. (2019) An improved selective facial extraction model for age estimation. In *International Conference on Image and Vision*

Computing New Zealand (IVCNZ)

Sun, Y., Zhang, C., Huang, Z., Liu, J., Han, J., & Ding, E. (2018). TextNet: Irregular text reading from images with an end-to-end trainable network. In *Asian Conference on Computer Vision* (pp. 83-99). Springer.

Tan, G., Guo, Z., & Xiao, Y. (2019). PA-RetinaNet: Path augmented RetinaNet for dense object detection. In *International Conference on Artificial Neural Networks* (pp. 138-149). Springer.

Tang, P., Wang, X., Wang, A., Yan, Y., Liu, W., Huang, J., & Yuille, A. (2018). Weakly supervised region proposal network and object detection. In *European Conference on Computer Vision (ECCV)* (pp. 352-368).

Tian, Z., Shen, C., & Chen, H. (2020). Conditional convolutions for instance segmentation. In *ECCV, Part I 16* (pp. 282-298). Springer.

Tian, Z., Shen, C., Chen, H., & He, T. (2019). FCOS: Fully convolutional one-stage object detection. *In IEEE/CVF International Conference on Computer Vision (ICCV)*, 9626-9635.

Tran, G. S., Nghiem, T. P., Nguyen, V. T., Luong, C. M., & Burie, J.-C. (2019). Improving accuracy of lung nodule classification using deep learning with focal loss. *Journal of Healthcare Engineering*.

Uijlings, J. R., Van De Sande, K. E., Gevers, T., & Smeulders, A. W. (2013). Selective search for object recognition. *International Journal of Computer Vision, 104*(2), 154-171.

Vaswani, A., Bengio, S., Brevdo, E., Chollet, F., Gomez, A. N., Gouws, S., Jones, L., Kaiser, Ł., Kalchbrenner, N., & Parmar, N. (2018). Tensor2Tensor for neural machine translation. *arXiv preprint arXiv:1803.07416*.

Viola, P. A., & Jones, M. J. (2001). Rapid object detection using a boosted cascade of simple features. *In IEEE CVPR*.

Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2021). Scaled-YOLOv4: Scaling cross stage partial network. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 13029-13038).

Wang, C.-Y., Liao, H.-Y. M., Wu, Y.-H., Chen, P.-Y., Hsieh, J.-W., & Yeh, I.-H.

(2020). CSPNet: A new backbone that can enhance learning capability of CNN. In *IEEE/CVF CVPR Workshops* (pp. 390-391).

Wang, G., Wu, X., Yan, W. (2017) The state-of-the-art technology of currency identification: A comparative study. *International Journal of Digital Crime and Forensics* 9 (3), 58-72

Wang, G., Ren, G., Wu, Z., Zhao, Y., & Jiang, L. (2013). A robust, coarse-to-fine traffic sign detection method. In *International Joint Conference on Neural Networks (IJCNN)*.

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, J., Bacic, B., Yan, W. (2018) An effective method for plate number recognition. *Multimedia Tools and Applications* 77 (2), 1679-1692.

Wang, J., Yan, W. (2020) BP-neural network for plate number recognition. *Deep Learning and Neural Networks: Concepts, Methodologies, Tools, and Applications.*

Wang, K., Babenko, B., & Belongie, S. J. (2011). End-to-end scene text recognition. In *International Conference on Computer Vision*, 1457-1464.

Wang, L., Yan, W. (2021) Tree leaves detection based on deep learning. In *International Symposium on Geometry and Vision.*

Wang, X., Yan, W. (2019) Human gait recognition based on frame-by-frame gait energy images and convolutional long short-term memory. *International Journal of Neural Systems* 29 (12)

Wang, X., Yan, W. (2019) Multi-view gait recognition based on ensemble learning. *Springer Neural Computing and Applications.*

Wang, X., Yan, W. (2019) Gait recognition using multichannel convolutional neural networks. *Neural Computing and Applications.*

Wang, X., Yan, W. (2019) Human gait recognition based on self-adaptive hidden Markov model. *IEEE/ACM Transactions on Biology and Bioinformatics*

Wang, X., Yan, W. (2020) Non-local gait feature extraction and human identification. *Multimedia Tools and Applications.*

Wang, Z., & Bovik, A. C. (2009). Mean squared error: Love it or leave it? A new look at signal fidelity measures. *IEEE Signal Processing, 26*(1), 98-117.

Weng, J., Ahuja, N., & Huang, T. S. (1992). Cresceptron: a self-organizing neural network which grows adaptively. In *IJCNN International Joint Conference on Neural Networks, 1,* 576-581 vol.571.

Weygand, Z. (2020). *The Blind in French Society from the Middle Ages to the Century of Louis Braille.* Stanford University Press.

Whitley, D. (1994). A genetic algorithm tutorial. *Statistics and Computing, 4*(2), 65-85.

Xiang, Y., Yan, W. (2021) Fast-moving coin recognition using deep learning. *Multimedia Tools and Applications.*

Xiao, B., Nguyen, M., Yan, W. (2021) Apple ripeness identification using deep learning. *International Symposium on Geometry and Vision.*

Xin, C., Nguyen, M., Yan, W. (2020) Multiple flames recognition using deep learning. *Handbook of Research on Multimedia Cyber Security*, 296-307.

Xing, J., Yan, W. (2021) Traffic sign recognition using guided image filtering. *International Symposium on Geometry and Vision*.

Xing, J., Yan, W. (2021) The improved framework of traffic sign recognition by using guided image filtering. *Springer Nature Computer Science.*

Xu, B., Wang, N., Chen, T., & Li, M. (2015). Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853.*

Xu, Y., & Jin, Z. (2008). Down-sampling face images and low-resolution face recognition. In *International Conference on Innovative Computing Information and Control* (pp. 392-392).

Yan, W., Ding, W., Qi, D. (2001) Rational many-knot spline interpolating curves and surfaces. *Journal of Image and Graphics* 6 (6), 568-572.

Yan, W., Ding, W., Qi, D. (1999) Many-knot spline interpolating curves and their applications in font design. *Computer Aided Drafting, Design and Manufacturing.*

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

Yan, W. (2021) *Computational Methods for Deep Learning: Theoretic, Practice and Applications*. Springer.

Yao, J., Qi, J., Zhang, J., Shao, H., Yang, J., & Li, X. (2021). A real-time detection algorithm for Kiwifruit defects based on YOLOv5. *Electronics*, *10*(14), 1711.

Yu, C.-W., Chen, Y.-L., Lee, K.-F., Chen, C.-H., & Hsiao, C.-Y. (2019). Efficient intelligent automatic image annotation method based on machine learning techniques. In *International Conference on Consumer Electronics-Taiwan (ICCE-TW)* (pp. 1-2)

Yu, J., Jiang, Y., Wang, Z., Cao, Z., & Huang, T. (2016). UnitBox: An advanced object detection network. In *ACM International Conference on Multimedia* (pp. 516-520).

Yu, Z., Yan, W. (2020) Human action recognition using deep learning methods. In *International Conference on Image and Vision Computing New Zealand*. *s*

Yuan, D., Fan, N., Chang, X., Liu, Q., & He, Z. (2020). Accurate bounding-box regression with distance-IOU loss for visual tracking. *arXiv preprint arXiv:2007.01864*.

Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., & Yoo, Y. (2019). CutMix: Regularization strategy to train strong classifiers with localizable features. In *IEEE/CVF International Conference on Computer Vision* (pp. 6023-6032).

Zhang, J., Wang, W., Huang, D., Liu, Q., & Wang, Y. (2019). A feasible framework for arbitrary-shaped scene text recognition. *arXiv preprint arXiv:1912.04561*.

Zhang, L., Yan, W. (2020) Deep learning methods for virus identification from digital images. In *International Conference on Image and Vision Computing New Zealand.*

Zhang, Q., Yan, W. (2018) Currency detection and recognition based on deep learning. In *IEEE International Conference on Advanced Video and Signal Based Surveillance* (AVSS)

Zhang, Q., Yan, W., Kankanhalli, M., Overview of currency recognition using deep learning. *Journal of Banking and Financial Technology* 3 (1), 59–69.

Zhang, R., Tian, Z., Shen, C., You, M., & Yan, Y. (2020). Mask encoding for single

shot instance segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 10226-10235).

Zhao, K., Yan, W. (2021) Fruit detection from digital images using CenterNet. In *International Symposium on Geometry and Vision*.

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

Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R., & Ren, D. (2020). Distance-IoU loss: Faster and better learning for bounding box regression. In *AAAI Conference on Artificial Intelligence* (Vol. 34, pp. 12993-13000).

Zhou, D., Fang, J., Song, X., Guan, C., Yin, J., Dai, Y., & Yang, R. (2019). IoU loss for 2D/3D object detection. In *International Conference on 3D Vision (3DV)* (pp. 85-94)

Zhou, F., Zhao, H., & Nie, Z. (2021). Safety helmet detection based on YOLOv5. In *IEEE International Conference on Power Electronics, Computer Applications (ICPECA)* (pp. 6-11)

Zhu, C., He, Y., & Savvides, M. (2019). Feature selective anchor-free module for single-shot object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 840-849).

Zhu, Y., Yan, W. (2022) Traffic sign recognition based on deep learning. *Multimedia Tools and Applications.*

Zou, Z., Shi, Z., Guo, Y., & Ye, J. (2019). Object detection in 20 years: A survey. *arXiv preprint arXiv:1905.05055*.