School of Computing
Faculty of Engineering AND
PHYSICAL SCIENCES

UNIVERSITY OF LEEDS

Final Report

<Object Detection Website Based on Deep Learning>

<Yunjia Feng>

Submitted in accordance with the requirements for the degree of

<BSc Computer Science>

<2021/22>

The candidate confirms that the following have been submitted*:*

<As an example>

Items	**Format**	**Recipient(s) and Date**
Final Report	*Hard copy and PDF file*	*Hard copy handed to SSO (DD/MM/YY); PDF uploaded to Minerva (DD/MM/YY)*
Participant consent forms	*PDF file / file archive*	*Uploaded to Minerva (DD/MM/YY)*
Link to online code repository	*URL*	*Sent to supervisor and assessor (DD/MM/YY)*
User manuals	*PDF and/or hard copy*	*Sent to client and supervisor (DD/MM/YY)*

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student) Yunjia Feng

Summary

This project is mainly focused on implementing a website providing users services to execute object detection tasks, which will find objects in the given image with detailed information including classification, location, size, the confidence of each object, and object number.

To offer users a better experience, functionalities such as signing up, logging in, logging out, selecting different weights, marking results into the collection, removing from the collection, are integrated into the system by using the back-end framework Flask in Python, the front-end framework Vue, and MySQL database.

For the deep learning module, this project adopts YOLO-5 5.0 as the basic algorithm, uses the VOC-2007 data set for training, training 4 different sizes of models to be used in different situations, with the highest mAP reaching 74.7%.

To compose this thesis, development history, and important techniques of object detection are reviewed, and the evolvement of YOLO from v0 to v5 is analyzed step by step.

Acknowledgements

In this project, I would like to sincerely thank my instructor Professor Jin Hou, with her help in thesis writing, I mastered more important skills in technical writing. By following her feedback when preparing mid-term defense slides, I obtained satisfactory results in the presentation. With her guidance in project management, I successfully finish my project ahead of the due. Thanks to her careful instructions, I overcome some technical difficulties when building this project. It is quite beneficial and an honor for me to work on this project with Professor Hou.

Table of Contents

Chapter 1

Introduction and Background Research

1.1 Introduction

Computer vision analysis of target motion can be roughly divided into three levels: motion segmentation and object detection, target tracking, action recognition, and behavior description[1]. Object detection has always been of great significance in the field of computer science, with the main objective to enable the computer to accurately classify the objects in a given picture and find the position of each object. Object detection is not only one of the basic tasks to be solved in the field of computer vision but also the basis of video surveillance technology among other tasks. However, object detection is still a challenging task with great potential and has great space for improvement, since the objects in the video have different poses and often appear to overlap, especially when their movements are irregular. Meanwhile, the resolution, weather, illumination, and other conditions of the surveillance video or images as well as the diversity of scenes should also be taken into consideration, which makes this task more challenging.

In recent years, many computer vision researchers both at home and abroad had developed a large number of excellent object detection algorithms in a neural network, including Faster R-CNN, SSD, YOLO.

From the perspective of research, the significance of object detection dues to it is one of the fundamental tasks in the field of computer vision, since it is the basis of many other high-level tasks, including image classification, face recognition, target tracking, pedestrian re-recognition. Meanwhile, there is a large number of well-known national and international research teams had been focused on the field of object detection: MIT Computer Science and Artificial Intelligence Laboratory, Stanford Computer Vision Lab, National Laboratory of Pattern Recognition of Chinese Academy of Sciences, LAMDA Institute of Nanjing University.

While from the perspective of the application, object detection has shown a wide range of practical usages: face detection technology, pedestrian detection technology applied in video surveillance, entrance and exit statistics, traffic sign detection technology, vehicle detection technology applied in aided driving, automatic driving. At the same time, major technology companies, for example, Microsoft, Google, Ali, and Baidu, have also spent a lot of manpower and material resources to explore the object detection field, which also indicates the significance and prospect of the object detection field.

The goal of this project is to select appropriate object detection algorithms and data sets to train a deep learning model, then develop a website for users which allows them to complete object detection tasks easily. Moreover, the website should provide additional functions that exclude basic object detection functionality to offer users a better experience, for instance, allowing them to change weights to suit different tasks, or mark recognition results after logging in.

1.2 Literature Review

1.2.1 Significance of Object Detection

Object detection had long been one of the most important and challenging tasks in computer vision, which detects instances of objects in a given image. It is also the basis of other high-level significant computer vision tasks, instance segmentation[2], object tracking[3], etc.

Over twenty years, object detection had been developed rapidly especially due to the achievement of deep learning recently[4], tremendous improvements pushed many applications into usage, for instance, face detection technology, autonomous driving, pedestrian detection technology applied in video surveillance, entrance and exit statistics.

1.2.2 Methods in Two Periods

Looking back through the history of object detection could be divided into two periods: the traditional method and the deep learning method with the separate line of 2014.

In general, object detection is a task to find all the objects of interest in the image for two sub-tasks, including object positioning and object classification[5]. The traditional object detection method, for example, the sliding window algorithm is generally divided into three stages: firstly, select some candidate regions on a given image, then extract features from these regions, and finally classify them using trained classifiers.

Traditional methods also include Viola Jones Detectors (2001)[6], HOG Detector(2005)[7] and DPM (2008)[8]. DPM, which stands for Deformable Part-based Model, was the winner of the VOC-2007 competition, which also was the best technique in the end era of the traditional method, which focused on detecting smaller parts of an object, for example, a human could be divided into arms, legs, head.

The efforts of traditional methods not only clear the path but also brought much inspirations to the researchers who study deep learning methods. Since 2006, in the lead of Hinton, Bengio, Lecun, and other researchers, an enormous number of deep neural network papers

had been published, especially after Hinton's research group participated in the ImageNet image recognition competition in 2012 and won the championship using AlexNet[9], constructed by CNN (Convolutional Neural Network), then neural networks began to receive extensive attention from then on.

In the deep learning era, the methods were normally divided into two categories: one-stage methods and two-stage methods:

- One-stage object detection algorithms hold the philosophy of detecting the objects in only one step. This kind of detection algorithm does not need the Region Proposal Stage and can directly generate the category probability and position coordinates of objects through only one stage. One-stage typical algorithms include YOLO, SSD, RetinaNet, and CornerNet[10].
- Two-stage object detection algorithms embrace the idea of “from coarse to fine”[11], they divide detection problems into two stages, the first stage is the generation of Region Proposals, which includes the approximate location information of the object, and the second stage is the classification and location refinement of the candidate regions. The representatives of two-stage algorithms are R-CNN, Fast R-CNN, Faster R-CNN, etc.

YOLO stands for “You Only Look Once”, is one of the most famous and first one-stage object detection algorithms, adopts a separate CNN model to achieve end-to-end object detection. YOLO-v0 originates from the idea of transforming the classification network directly into a positioning network by dividing the image into several parts, then predicting with bounding boxes (Figure 1.2.2.1).

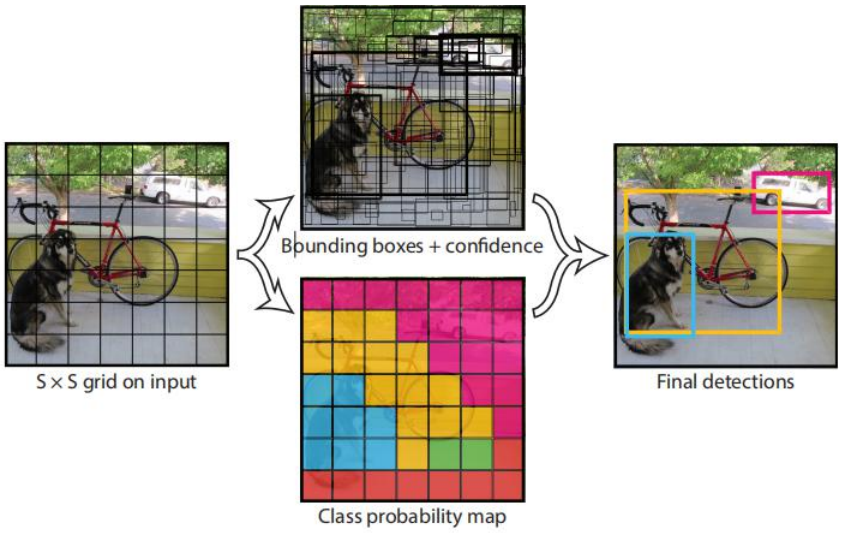


Figure 1.2.2.1 Theory of YOLO Algorithm

In general, the input image is re-sized and sent to the CNN network, then the detected object results are obtained by processing network prediction. Compared with the R-CNN algorithm, YOLO is a unified framework with faster speed while the training process is also end-to-end[12].

Since YOLO both has rapid recognition speed and high accuracy, which is suitable for a lightweight website application, this project adopts YOLO-v5 as the basis to develop an object detection website. More details will be concentrated on YOLO from v0 to v5 in later chapters.

SSD[13] stands for Single Shot Multi-Box Detector, which is a single-stage, multiple proposal object detection algorithm. The uniqueness of SSD is its multi-scale detection techniques which allow predicting different sizes of objects using different layers in the network, which significantly enhance the accuracy of the single-stage method. SSD also uses CNN with a multi-scale feature map to detect objects, the structure is shown as the Figure1.2.2.2:

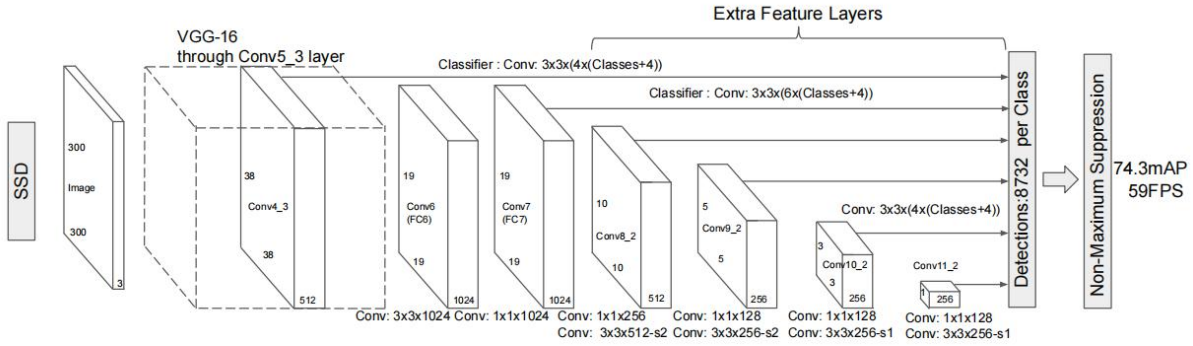


Figure 1.2.2.2 SSD Network Structure

Additionally, SSD adopts VGG16 as the basic model, and then adds a convolution layer based on VGG16 to obtain more feature maps for detection[14]. There are also some improved algorithms based on SSD, for instance, DSSD[15] and FSSD[16], which have a different structure for their CNN module.

RetinaNet was proposed in 2017, which deeply analyses the differences between one-stage methods and two-stage methods, and designed a new loss function called “Focal Loss”[17] in order to solve the problem of imbalanced data.

R-CNN, Fast R-CNN, and Faster R-CNN are a series of two-stage algorithms. R-CNN introduces selective search to generate 2000 proposals, then re-sizes them to feed into a CNN network for feature extraction, a SVM classifier will handle the features and give predictions[18].

Since the 2000 proposals of R-CNN needs large computational resources, leading to a really slow speed, Fast R-CNN extracts feature from the whole image instead of proposal regions to update weights and adopt selective search to the outputs of convolution rather than the raw image which significantly reduces computation cost and enhance detecting speed[19].

Faster R-CNN is a model after the evolvement of R-CNN and Fast R-CNN when Ross B. Girshick proposed it in 2016[18]. It adopts RPN (Region Proposal Network) to replace the original selective search, reducing a large amount of computation redundancy and improving accuracy at the same time. In terms of structure, the Faster R-CNN integrated feature extraction, bounding box regression (rectangular refine), and classification into one network, which greatly improves the overall performance[20], especially in the detection speed. The structure of the network is shown in Figure 1.2.2.3 below:

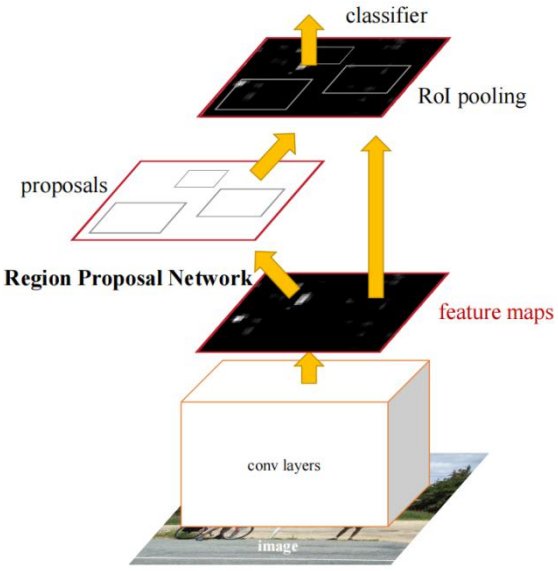


Figure 1.2.2.3 Faster R-CNN basic structure

Furthermore, Cascade R-CNN was proposed as a multiple-stage method on the basis of previous versions[21] but has better accuracy in detection.

1.2.3 Data Set and Evaluation

Excellent algorithms still need proper data set to train and evolve, competitions normally will be held to test performance on datasets for famous algorithms, also serving as the benchmark for later research. Among these datasets, Pascal VOC, MS-COCO, and Open Images are frequently used for object detection tasks.

Pascal VOC stands for The PASCAL Visual Object Classes (VOC) Challenges[22], supports many computer vision tasks, for example, object detection, instance segmentation, object tracking as Figure 1.2.3.1 shows.

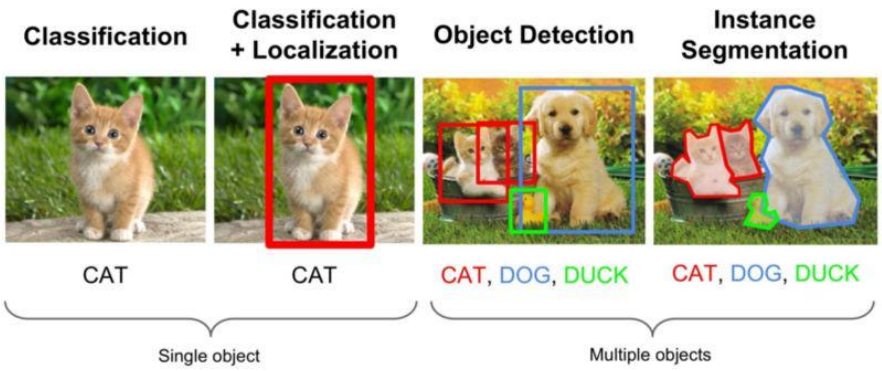


Figure 1.2.3.1 Computer Vision Tasks

MS-COCO is also a famous data set that is used in the object detection task, which has more than 160 thousand images with about 900 thousand objects for 80 classes. Open Image data set is normally used in two tasks: object detection and predicting relationships between objects.

Within the appropriate algorithm and datasets for training, then an evaluation matrix is needed to test the performance of a method. In the object detection task, mAP (mean Average Precision) is one of the most important indicators, which was first mentioned in VOC-2007. To define AP, PR (Precision-Recall) curve is requested with a specified IoU (Intersection over Union) threshold (usually 0.5), the area under the PR curve means the AP value. After that, sum up all the AP for every class then divide the class number can obtain the final mAP.

1.2.4 Important Concepts and Future Work

Among the history of object detection, there are some important concepts that brought huge influence to current object detection research that need to be mentioned.

CNN, which stands for convolutional neural networks, had been used since 1990 in a variety of computer vision fields. CNN had begun the new era of deep learning by allowing parameter sharing and sparsity of connections, which serve as the basis of all the one-stage and two-stage methods mentioned above.

NMS, which stands for Non-Maximum Suppression, is a technique that is used to eliminate redundant bounding boxes for the same object generated by the network. There are also many other NMS versions, for instance, Soft-NMS, DIoU NMS, Conv NMS, Learning NMS, etc. The invention of NMS had greatly enhanced the accuracy of the algorithms, which makes it remains a necessary component of the modern network.

Moreover, some new strategies that are used to be applied in other filed are now put into an experiment in object detection and yield remarkable results.

Adversarial training or GAN (Generative Adversarial Networks) [23] is quite popular these days, especially in research that allows AI to generate their own production (composing songs, drawing paintings, etc.). Typically, GAN includes two networks, one “teacher” network to criticize the AI production, while another “student” network generates AI production, two networks will learn together and yields better results after training. For object detection tasks, GAN had been put into usage to enhance the performance when detecting small or overlapping objects.

In spite of the great progress that object detection ever had since its start, there are still some serious challenges that had troubled many researchers till now, for instance, when detecting bad weather situations (strongly snowy, foggy, etc.), or many small targets overlapping (a really busy street with hundreds of pedestrians) performance of algorithms normally are barely satisfactory.

To solve these problems above, further research is needed to enable object detection techniques to be applied in more situations to provide people with better lives. In the future, object detection research may be focused more on real-time techniques, which could provide video surveillance on the road or autonomous cars. Therefore, video instead of images would become the mainstream media to apply object detection, which means more fast but accurate algorithms will be developed to serve the needs. Object detection application in smartphones is also a promising direction, which may also trigger the development of more lightweight models that could be used in mobile devices. It is reasonable to believe that in the future, object detection techniques will be more important in people’s lives than ever.

1.3 Description of Software Prototypes

This project is focused on developing a website that allows users to experience object detection conveniently. The software prototype of the project can be divided into three parts:

(1) Application that could provide service for object detection tasks users can upload images, and view results on the website, including object classification, location, confidence, size, object number, etc.

(2) Users are allowed to select different trained weights flexibly for object detection tasks.

(3) Application that could connect to the database, where stored user registration information. User can also bookmark their results into the collection, and remove them through an interface.

Chapter 2

Methods

2.1 Justification of YOLO

The traditional method -- sliding window algorithm uses different sizes of boxes to go through the image step by step. However, this method has a serious drawback: to obtain higher accuracy, the stride of the boxes needs to be smaller, which needs an incredibly huge amount of computational resources and costs a long time. Moreover, two potential problems exist in this method:

- Different sizes of boxes mean different sizes of inputs. Thus, the normalization process needs to be added int the network.
- Since the method will go through the whole image in a brute-force way, the background areas must be larger than the object areas, which will cause imbalanced data (imbalance between positive and negative samples).

To solve the problem above, the origin of YOLO is designed by simply transforming a classification detector into an object detector that could predict objects' locations. The traditional classification network normally ends with a fully-connected layer outputs N-dimensional one-hot vector, so the author of YOLO just changes the output layer into another vector (x,y,w,h,c) (x and y denote the coordinates location of the top-left location of the bounding box, w and h mean the width and height the bounding box, c is the confidence of the object), forming an object detector.

From the description above, YOLO-v0 can only output one object for one image, in order to detect multiple objects in one image, YOLO-v1 was introduced. First of all, instead of outputting only one set of one-hot vectors (x,y,w,h,c), the network divided the image into 7*7 into a total of 49 regions, one region is corresponding to a set of one-hot vector output. Moreover, if one object spanned more than one region, NMS is used to select the most confident region and give out a prediction. If there are multiple classes, YOLO-v1 just simply increases the output and repeats the 7*7 region detect for each class.

To solve the problem of detecting small objects, YOLO-v1 specifically add extra neuron network layers to handle them, which means there are two sets of one-hot vector (in total 98 bounding boxes), one for big targets while the other handling small objects[12]. For the other components, YOLO-v1 takes GoogleNet as the backbone without neck, which belongs to the

dense prediction detector. For training, YOLO-v1 trains the classification network with a 224*224 resolution rate, then trains the detection network with a 448*448 resolution rate.

Nevertheless, YOLO-v1 still had some problems left unsolved. Despite the fast speed, the accuracy of the network is not satisfying, and the recall is relatively low which means many targets are missing.

To tackle the first problem, YOLO-v2 used an anchor box for width and height, which was first introduced in R-CNN, aiming to predict relative offsets instead absolute offsets. Because the offset after regularization is smaller than the original height and weight values, the network can learn better and provide better accuracy.

For the second problem, YOLO-v2 evolve the network structure from 7*7 to 13*13 to promote recall, with a maximum of 169 objects that could be detected. Even though YOLO-v2 still does not have a neck component, the backbone is switched to darknet-19, and the fully-connect layer is substituted by the GAP (Global Average Pooling) layer to enhance the accuracy for small objects[24]. The training process for YOLO-v2 is also much more complicated: first of all, Darknet-19 is pre-trained in the ImageNet classification data set for 160 epochs with a 224*224 resolution rate, then they fine-tune the classification model for 10 more epochs with a 448*448 resolution rate since the usage of GAP allows training could accept the different size of inputs.

Even though YOLO-v1 added an extra output layer to detect small objects, the result was still not promising. The author of YOLO noticed this and changed CNN down-sampled rate into three branches: 32x-downsampling, 16x-downsampling, and 8x-downsampling, respectively detects for big, medium, and small objects, in total, generate 10467 bounding boxes, way larger than YOLO-v2 845 bounding boxes.

YOLO-v3 also uses FPN as the neck, which could pass the feature information from the bottom layer to the upper layer. For the other components, YOLO-v3 updates the backbone to Darknet-53 with the introduction of Resnet, which makes the network deeper, contributes to higher accuracy to a large degree[25].

YOLO-v4 improved the head of the network by using multiple anchors to predict single ground truth, which could increase the number of positive samples, aiming to mitigate the problem of imbalance between the positive and negative samples. Moreover, from v0 to v3, YOLO adopts the traditional MSE (Mean Squared Error) loss when calculating loss function, but MSE loss can not tell the difference between overlapping areas or IOU are the same (Figure2.1.1).

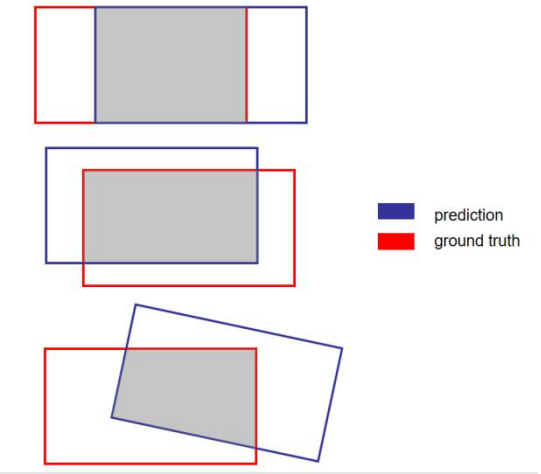


Figure 2.1.1 Same IOU but Different MSE Loss

Therefore, CIoU-loss is used to calculate the central point distance between ground truth and prediction. Based on YOLO-v3, YOLO-v4 further enhance the network structure in the neck, by adding an SPP module, allowing the multi-scale integration for pooling, and PAN structure to concatenate adjacent feature layers for prediction[26]. Besides, YOLO-v4 also put more effort into refining the inputs by using Mosaic, which could largely enhance data richness, leading to better network robustness.

Finally, YOLO-v5 was released in 2020. Compare to the earlier version, YOLO-v5 uses an adaptive anchor, which allows the anchor box could learn with the network, so previously predefined fixed (x,y,w,h) values will change by the learning process to compare with ground truth boxes, then update network parameters to obtain better training results. Therefore, the loss function also changed into GIoU loss.

YOLO-v5 also adopts the Focus module in its backbone, which slices the data into 4 groups then concatenates them with the channel, executing the down-sampling procedure without largely losing information. The most interesting part of YOLO-v5 is that there are four options of pre-trained weights for choosing: s (small),m (medium),l (large),x (extra-large). The larger the weight is, the higher precision goes, while the processing time also increases. These four weights are the results of different parameters in network depth, width, and a different number of res units when training, which results in different numbers of the convolutional kernel in each layer. At last, YOLO-v5 also uses an adaptive image re-scale module to improve detection speed.

2.2 Data Set and Training

Since Pascal VOC is one of the most important data sets for object detection, so there are already benchmarks created by other famous algorithms. Moreover, with a relatively smaller size than others, Pascal VOC is suitable for this lightweight project. This project finally selects VOC 2007 for training, because VOC 2007 focused on object detection tasks, but the latter version (VOC 2012) was also used for other tasks (instance segmentation and object tracking). Despite its small size, it also has enough data to train a decent model, with 9963 images (train, validation, and test), in a total of 24640 objects for 20 different classes.

Before training the model with the chosen data set, format transform code is needed to convert the original VOC format (XML format) into a compatible YOLO version (txt format). Data set subdivision is also necessary to divide the whole data set into three parts: train, validation, and test, which is important to obtain better training results.

Moreover, this project needs some dependency to install through pip: Flask, Pytorch, torchvision, torchaudio, cudatoolkit, flask_sqlalchemy, numpy. Other packages used in the front-end could be installed through npm: ant-design-vue, axios, echarts, element-ui, jQuery, socket.io, vue.

In the training phase, the whole process was carried on by the server in the school laboratory through SSH, equipped with Anaconda to prepare a virtual environment with Python 3.8, the hardware configuration with the driver version is listed in Table 2.2.1:

	Specification
GPU	4 NVIDIA GTX TITAN X
GPU Memory	12212MB (12GB) per GPU
CUDA Version	11.2
Driver Version	460.80
CPU	Intel Core i7-5930K 3.50GHz
Memory	32834268KB(32GB)

Table 2.2.1 Training Hardware Configuration

In the developing phase, the application is running on a personal laptop, the hardware configuration with the driver version is listed in Table 2.2.2:

	Specification
GPU	NVIDIA GTX 1070
GPU Memory	8192MB (8GB)
CUDA Version	11.6
Driver Version	511.79
CPU	Intel Core i7-7700HQ 2.80GHz
Memory	16659528KB(16GB)

Table 2.2.2 Developing Hardware Configuration

The following picture shows temporary results when training weight size ***m*** during the last 10 epochs, using the visualization tool ***wandb***:

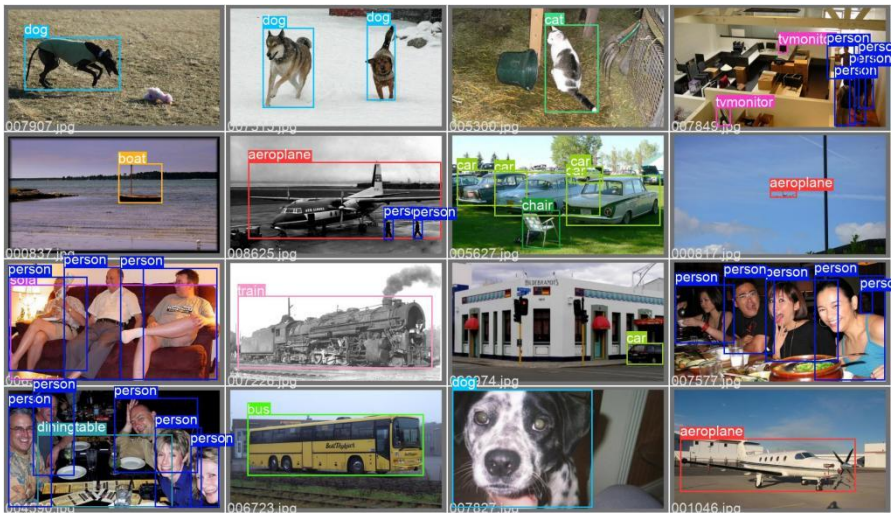


Figure 2.2.1 Temporary Training Results in Wandb

All four weights (s,m,l,x) were trained until reaching the convergence loss, so there exist differences in training epochs for different weights, more details will be discussed in Chapter 3.

2.3 Front-end, Back-end, and Database

The project file structure can be divided into two parts, one directory storing documents (reports, thesis, and other materials), and the other directory is the main code folder, which is composed of two sub-folder, the back-end module, and the front-end module respectively.

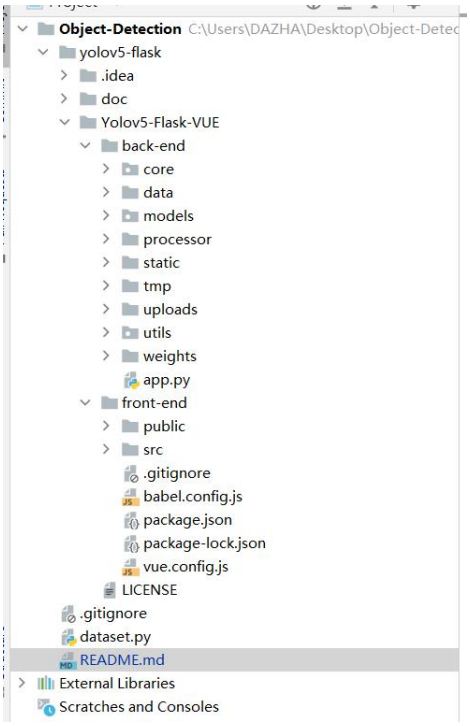


Figure 2.3.1 Project File Structure

This project adopts MySQL as a database and uses SQL-Alchemy to connect Python Flask with MySQL. Since MySQL is a medium-weight relational database, it is suitable for this project, and it is also quite convenient to connect MySQL API through Python. There are two tables in total, users and images table designs are shown below as Figure 2.3.2:

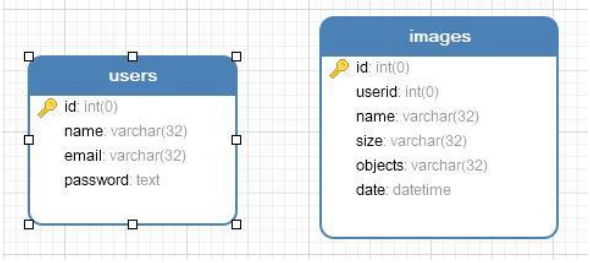


Figure 2.3.2 Database ER Diagram

In the ***users*** table, there are 6 fields with constraints:

Names	Type	Description	Constraints
id	Integer	User Identification Number	Primary Key, Auto Increment
name	Varchar	User Name	Unique, 32 bits
email	Varchar	User Email	Unique, 32 bits
password	Text	User Password (SHA-256)	None

Table 2.3.1 User Table Constraints

In the ***images*** table, there are 4 fields with constraints:

Names	Type	Description	Constraints
id	Integer	Image Identification Number	Primary Key, Auto Increment
userid	Integer	User Identification Number	Nullable
name	Varchar	Image Name	32 bits
size	Varchar	Image Size(MB)	32 bits
objects	Varchar	Object Number	32 bits
date	Datetime	Time Added To Collection	Nullable

Table 2.3.2 Images Table Constraints

2.4 Functionality

The user has three options when first starting the application: sign up, log in, and select weight for object detection tasks. The user could skip logging and use object detection functionality directly by uploading the image. However, if the user logged in, object detection

results are allowed to add to the user's collection, deletion is also permitted after login. The following use case diagram (Figure 2.4.1) shows the process:

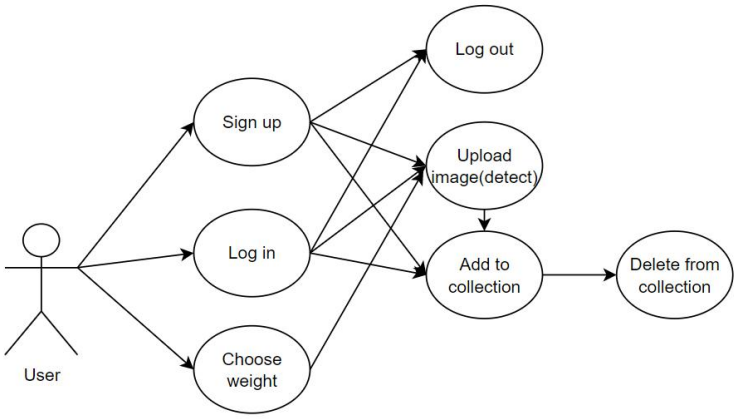


Figure 2.4.1 Use Case Diagram

The architecture of the system could be divided into 5 layers separately (Figure 2.4.2): Presentation Layer, Control Layer, Logic Layer, Application Layer, and Basis Layer. From the Presentation Layer, web pages are rendered by Vue.js, where users could interact with the website and make requests through routers. Then, Python Flask in the Control Layer will handle the requests with the help of Logic Layer components (view functions and YOLO-v5 API) integrated into the code. After that, the Data model and trained YOLO-v5 model in the Application Layer will be loaded for data storage and object detection tasks respectively. Finally, the data will be transmitted from the back-end to the MySQL database in Basis Layer.

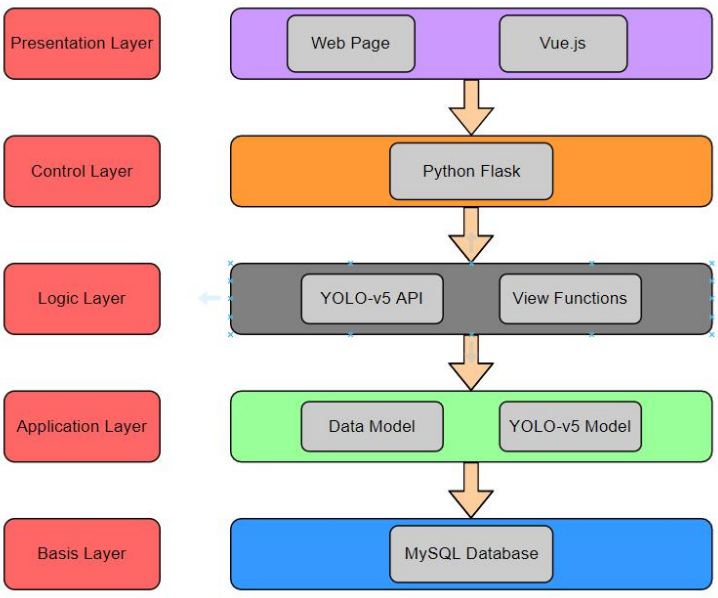


Figure 2.4.2 Project Architecture Diagram

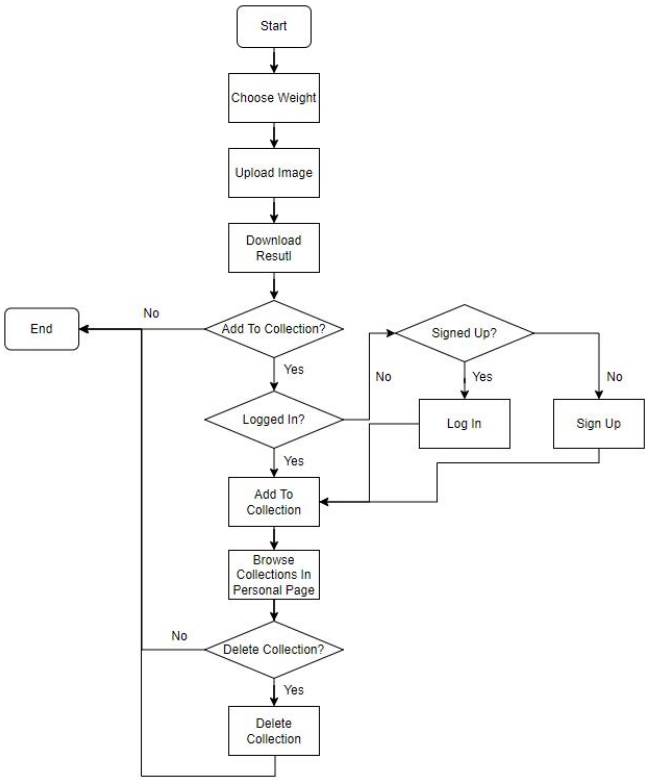


Figure 2.4.3 Flow Diagram

The user could configure the weight (s,m,l,x) at first, then upload the image for object detection tasks and download results. If the user wants to bookmark the result for collection, then the user needs to log in or sign up. After bookmarking the image, the user could continue browsing the remaining collection on their personal page where they could also delete them. The flow diagram (Figure 2.4.3) shows the details when interacting with the website.

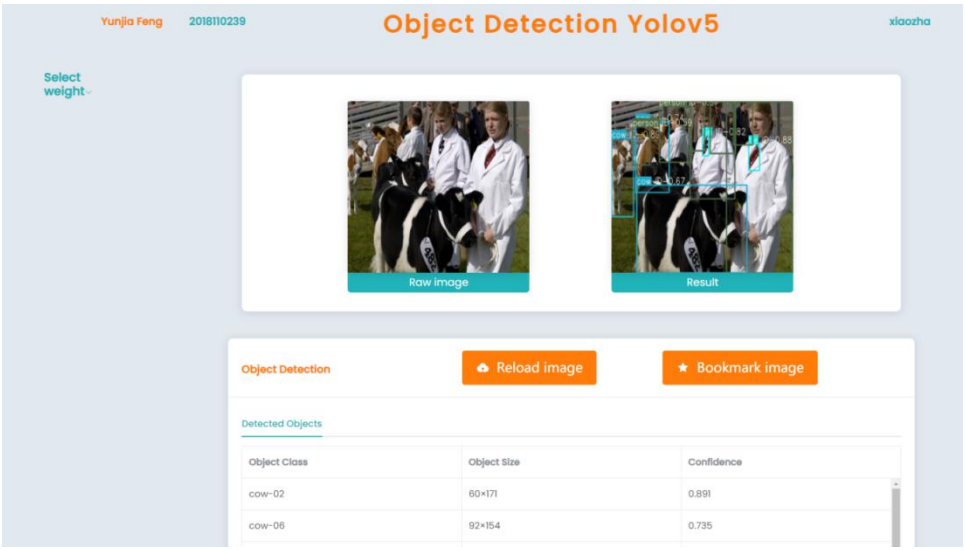


Figure 2.4.4 Application Page

Figure 2.4.4 shows the layout of the application page, where users could select 4 different weights in the top left corner, while the top right side indicates the current logged-in username. The user could upload or reload images and add results to the collection through two buttons in the middle, the results will show both by image and details(object class, size, and confidence) below the two buttons.

2.5 Version Control and Project Management

This project used Github as a version control tool and marked some important releases as major versions by adding tags to some commits.

As the Figure 2.5.1 shows, there were 79 commits in the Github repository from September 19th in 2021 till April 6th in 2022.

Figure 2.5.1 Github Commits

The Figure 2.5.2 Gantt chart shows the whole project management and plans from 2021 September to 2022 April (there were changes when executing these plans).

Figure 2.5.2 Gantt chart of Project Management

There were four sprints in total to finish according to four important releases on Github:

1. This sprint implemented basic functions to allow the server to handle object detection tasks.

2. This sprint added functionalities of sign up, log in, log out, and allow users to choose weights. User information was stored in the database with encryption.

3. This sprint introduced collections, which enable users to mark their results to collections, which they could browse and delete on their personal page.

4. This sprint was focused on checking for potential bugs, and adding comments for others to understand the code, Readme text was also created to start using the project, and unit tests scripts were implemented to test this final version.

Chapter 3

Results

3.1 Experiments

Table 3.1.1 shows the training evaluation matrix, indicators including mAP, Class loss, Box loss, training epochs, and speed (milliseconds per image) of four weights respectively.

Table 3.1.1 Training Evaluation Metrix

YOLO-V5	Small	Medium	Large	Extra Large
mAP(0.5)	0.2831	0.7147	0.7470	0.4277
Cls_loss	0.0274	0.0068	0.0052	0.0195
Box_loss	0.0500	0.0255	0.0225	0.0402
Train(epoch)	150	100	120	120
Speed(ms)	0.33	0.46	0.59	0.69

The following Figure shows the trend of mAP (threshold > 0.5), class loss, and box loss of 4 different weights (Figure 3.1.1, 3.1.2, 3.1.3 respectively).

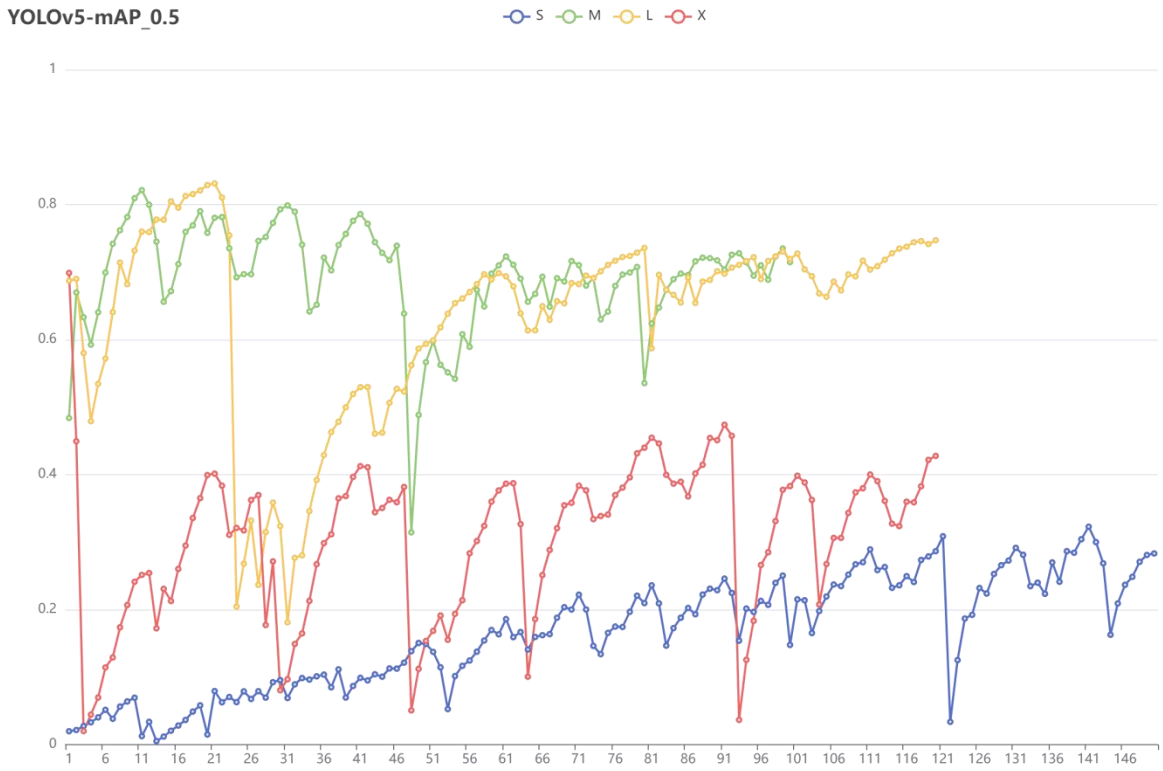


Figure 3.1.1 YOLO-v5 mAP(threshold > 0.5) for 4 Weights

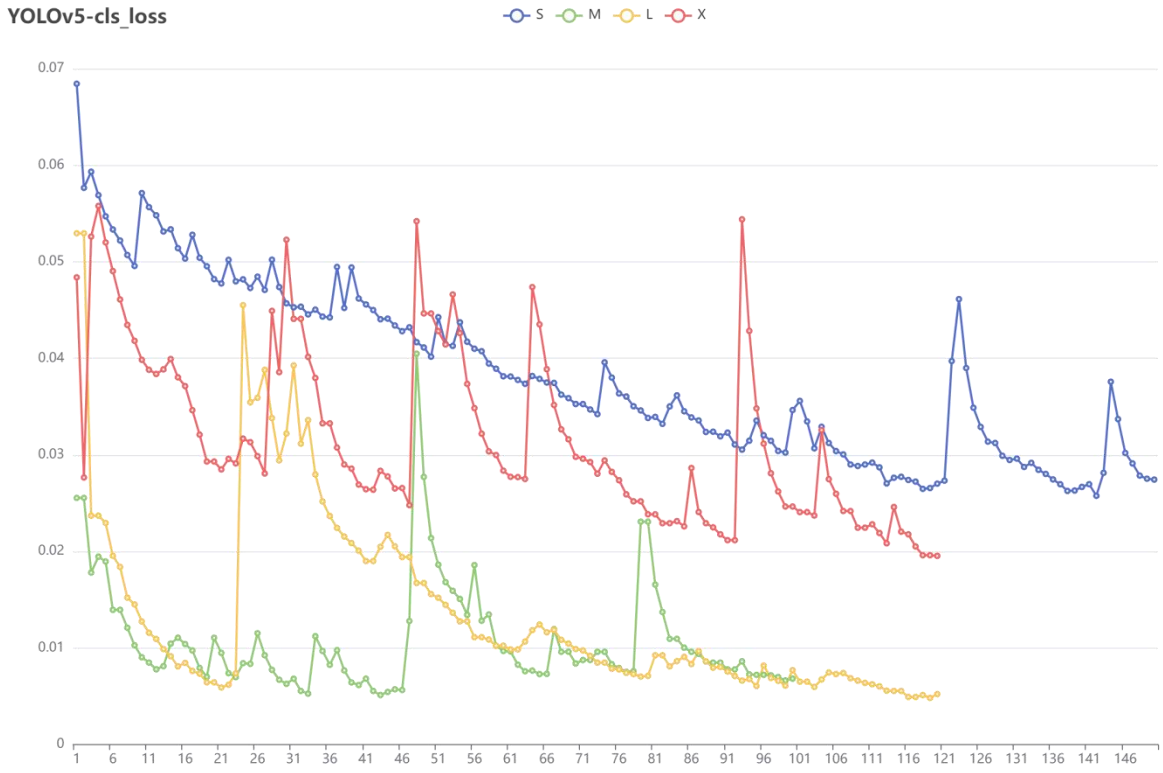


Figure 3.1.2 YOLO-v5 Class Loss for 4 Weights

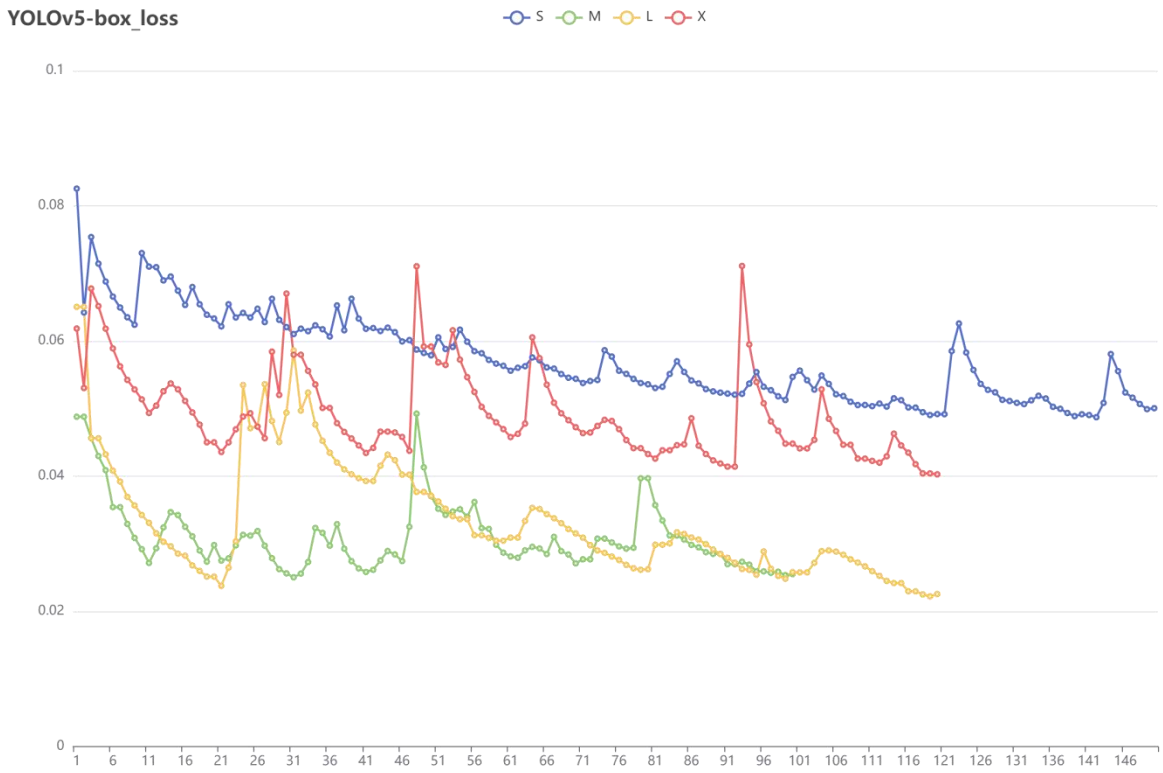


Figure 3.1.3 YOLO-v5 Box Loss for 4 Weights

From the figures above, it is clear to denote that weights M and L have higher mAP and lower losses than S and X.

3.2 Test Results

Table 3.2.1 shows unit test results of 6 different APIs using 3 Test tools: Postman (API test), Selenium (graphical user interface test), and Pytest (unit test with Python code).

API\Tool	Postman	Selenium	Pytest
Log In	Pass√	Pass√	Pass√
Log Out	Pass√	Pass√	Pass√
Sign Up	Pass√	Pass√	Pass√

Browse Collection	Pass√	Pass√	Pass√
Add Bookmark	Pass√	Pass√	Pass√
Remove Bookmark	Pass√	Pass√	Pass√

Table 3.2.1 Unit Test Results

Figure 3.2.1 shows an example when testing API browsing collections in Postman by submitting a username in the request, the server will return a response with an image and detailed information (image name, size, object number, creating date-time), other APIs were also tested through Postman.

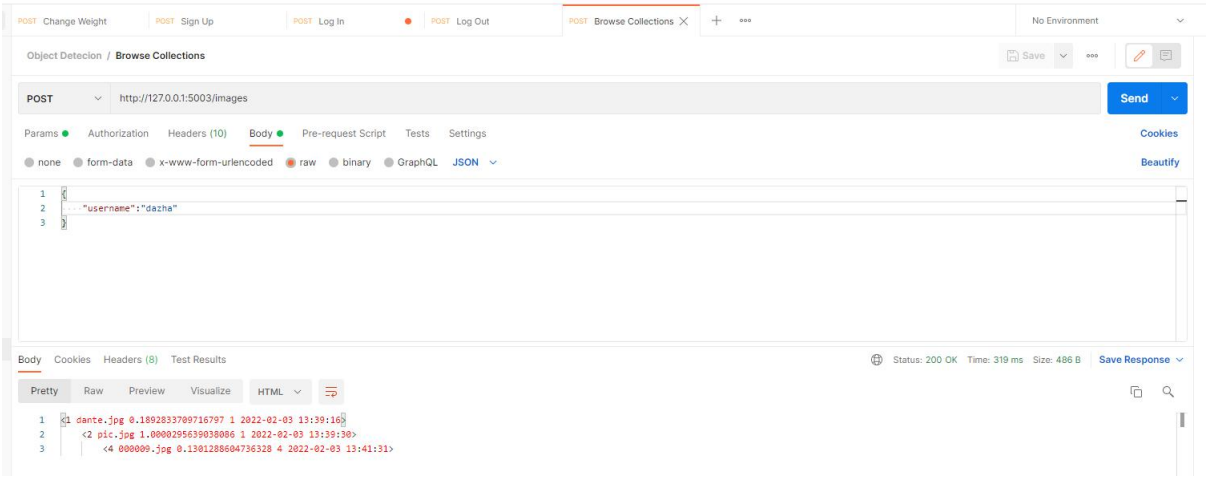


Figure 3.2.1 Postman Test for API Browsing Collections

Figure 3.2.2 shows an example when testing API sign-up in Selenium, which is a test tool simulating user behavior through browsers. Other APIs were also tested through Selenium.

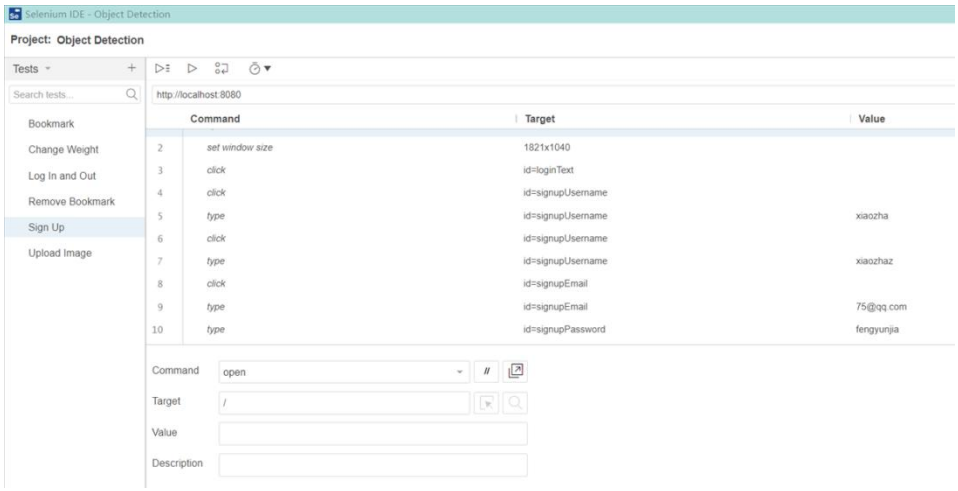
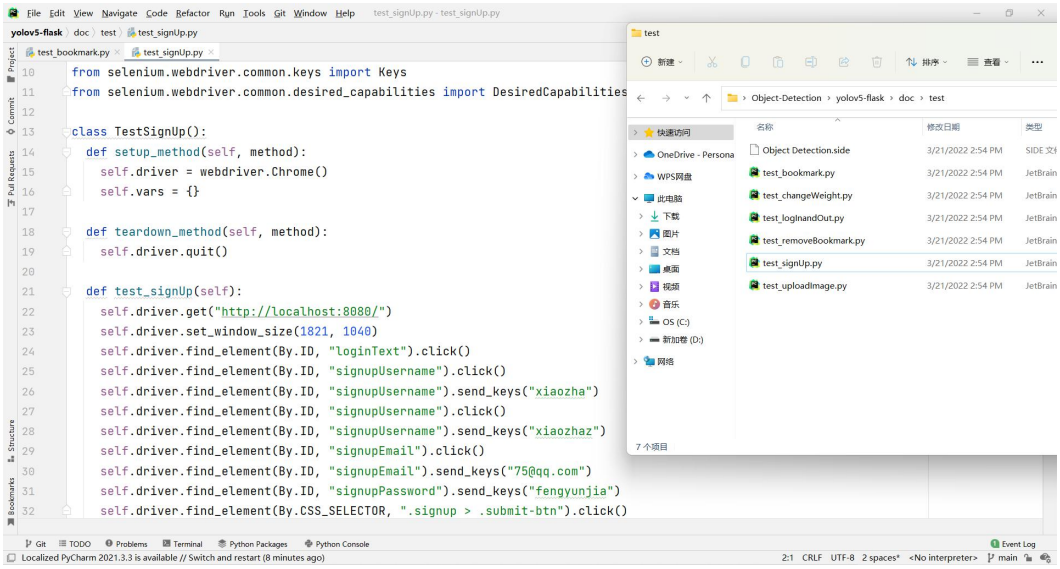


Figure 3.2.2 Selenium Test for API Sign Up

Figure 3.2.3 shows an example when testing API sign up in Pytest, which is a test tool sending requests to the server in Python. Other APIs were also tested through Pytest.



```
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.desired_capabilities import DesiredCapabilities

class TestSignUp():
  def setup_method(self, method):
    self.driver = webdriver.Chrome()
    self.vars = {}

  def teardown_method(self, method):
    self.driver.quit()

  def test_signUp(self):
    self.driver.get("http://localhost:8080/")
    self.driver.set_window_size(1821, 1040)
    self.driver.find_element(By.ID, "loginText").click()
    self.driver.find_element(By.ID, "signupUsername").click()
    self.driver.find_element(By.ID, "signupUsername").send_keys("xiaozha")
    self.driver.find_element(By.ID, "signupUsername").click()
    self.driver.find_element(By.ID, "signupUsername").send_keys("xiaozhaz")
    self.driver.find_element(By.ID, "signupEmail").click()
    self.driver.find_element(By.ID, "signupEmail").send_keys("75@qq.com")
    self.driver.find_element(By.ID, "signupPassword").send_keys("fengyunjia")
    self.driver.find_element(By.CSS_SELECTOR, ".signup > .submit-btn").click()
```

Figure 3.2.3 Pytest for API Sign Up

3.3 Comparing with Other Methods

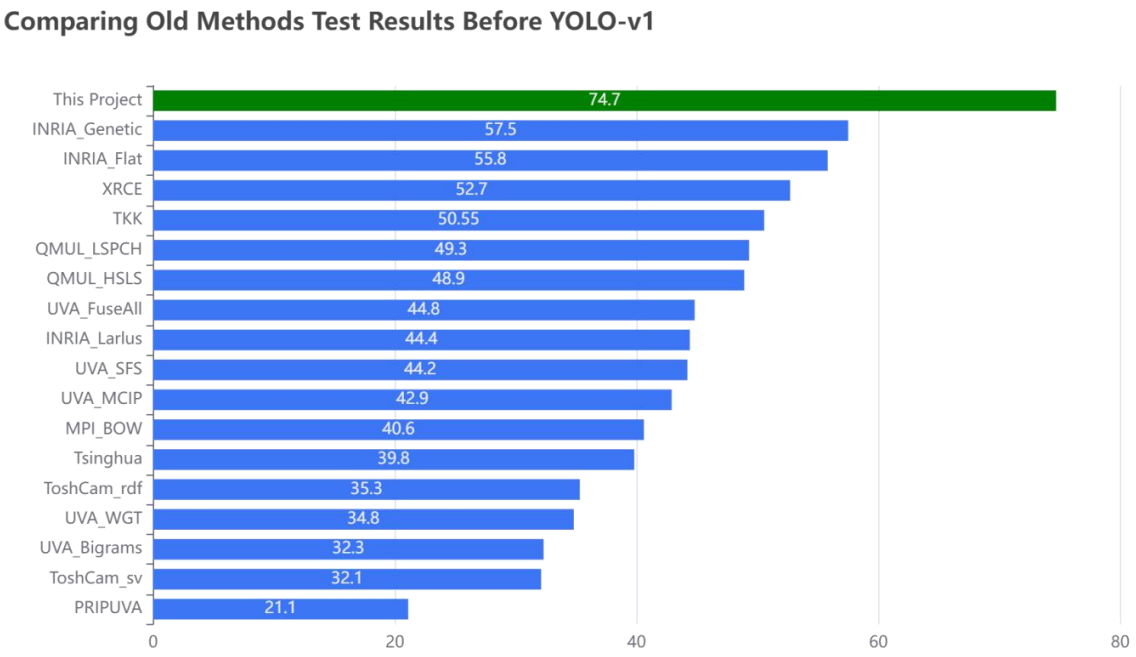


Figure 3.3.1 Comparing Old Methods Test Results Before YOLO-v1

The Figure 3.3.1 shows state of art algorithms in 2007 with their mAP performance rank in the VOC-2007 competition[12], the reason why comparing with some algorithms that were not

used in recent years is that some new methods only show their test results in COCO instead of VOC-2007. Compared with these old methods, it is clear to see that the models in this project are way better than the others.

The following Table 3.2.1 shows different methods(Selective Search, RPN+VGG) in SSD test in VOC-2007 and VOC-2012 (07 denotes for VOC-2007, 12 denotes for VOC-2012 in column data). Only compared with gray row, which is trained by VOC-2007, the model in this project still has some advantages over other methods[14].

Table 3.3.1 Comparing Results with SSD

Method	Proposals	data	mAP (%)
Selective Search	2000	07	66.9
Selective Search	2000	07+12	70.0
RPN + VGG, unshared	300	07	68.5
RPN + VGG, shared	300	07	69.9
RPN + VGG, shared	300	07+12	73.2
RPN + VGG, shared	300	COCO+07+12	78.8
This Project			**74.7**

The following Table 3.3.2 shows YOLO-v1 and Fast R-CNN tests in VOC-2007 and VOC-2012. Compared with Fast R-CNN trained by VOC-2007, there is about 7% mAP improvement in this project model[12].

Table 3.3.2 Comparing with Faster R-CNN in VOC-2007

Method	mAP (%)	Combined	Gain
Fast R-CNN	71.8	-	-

Fast R-CNN (VOC-2007)	66.9	72.4	.6
Fast R-CNN (VGG-M)	59.2	72.4	.6
Fast R-CNN (CaffeNet)	57.1	72.1	.3
YOLO (old version)	63.4	75.0	3.2
This Project	**74.7**	-	-

The following Table 3.3.3 shows Fast R-CNN, Faster R-CNN test, and SSD300, SSD512 in VOC-2007 and VOC-2012(07 denotes for VOC-2007, 12 denotes for VOC-2012 in column data). Compared with the method trained by VOC-2007, the model in this project outweigh Fast R-CNN, Faster R-CNN, and SSD500[20].

Table 3.3.3 Comparing with Fast R-CNN, Faster R-CNN, SSD-300, SSD-512 in VOC-2007

Method	data	mAP (%)
Fast R-CNN	07	66.9
Fast R-CNN	07+12	70.0
Faster R-CNN	07	69.9
Faster R-CNN	07+12	73.2
Faster R-CNN	07+12+COCO	78.8
SSD300	07	68.0
SSD300	07+12	74.3
SSD300	07+12+COCO	79.6

SSD512	07	71.6
SSD512	07+12	76.8
SSD512	07+12+COCO	81.6
This Project	07	**74.7**

Overall, the model in this project obtains satisfactory training results in VOC-2007 even when put into the current context, whose mAP is necessary enough for normal object detection tasks.

Chapter 4

Conclusion

4.1 Results

This project implemented a website that can enable users to execute object detection tasks by uploading images, configuring 4 different weights in different using situations, marking results, browsing, and delete in collections after logging in. With the model mAP of 74.7% in the VOC-2007 data set, the service will provide the user with the location, classification, size, and confidence of each object, and object number for a single image.

4.2 Ideas for Future Work

There still exists some deficiencies in this project, which need future work for improvement.

1. All the training in this project is taken by 10-30 consecutive epochs at one time instead of training for 100 epochs or until reaching the convergence. Although there is no solid evidence showing huge differences between these two methods, it is believed to train the model for once. However, due to the hardware limitation, the school laboratory can not support doing that. In the future, a high-performance machine equipped with larger GPU memory or using an online laboratory could eliminate this problem for all.

2. This project is running in a local host instead of providing web service because it needs the server to equip a high-performance Nvidia display card which most current server renting services could not provide. The future work for this project may need to rent a server for machine learning purposes specifically to provide users using the service online.

3. This project now only supports object detection tasks for the image. However, YOLO API can also support video and cameras. The reason why this project abandoned video object detection is that it takes a long time for video to upload and the server to process even in a local host situation. To overcome this problem, lower network latency and larger network bandwidth are needed in the future.

4. This project now is only focusing on developing YOLO for object detection tasks. However, other state-of-art algorithms also have their own strength and benefits in different situations. To provide users better experience and choices, multiple options to use different algorithms may be the direction for future development. Nevertheless, this needs extra workload to implement multiple API to use different algorithms, and also costs time and performance of

server since switching between different algorithm need bootstrap and load models from scratch.

List of References

[1] Vishwakarma S, Agrawal A. A survey on activity recognition and behavior understanding in video surveillance [J]. The Visual Computer, 2012: 1-27.

[2] K. He, G. Gkioxari, P. Doll´ar, and R. Girshick, “Mask rcnn,” in Computer Vision (ICCV), 2017 IEEE International Conference on. IEEE, 2017, pp. 2980–2988.

[3] K. Kang, H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, X. Wang et al., “T-cnn: Tubelets with convolutional neural networks for object detection from videos,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 28, no. 10, pp. 2896–2907, 2018.

[4] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” nature, vol. 521, no. 7553, p. 436, 2015.

[5] Jiang, B., Luo, R., Mao, J., Xiao, T., & Jiang, Y. (2018). Acquisition of localization confidence for accurate object detection. In ECCV (pp. 784–799).

[6] P. Viola and M. J. Jones, “Robust real-time face detection,” International journal of computer vision, vol. 57, no. 2, pp. 137–154, 2004.

[7] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, vol. 1. IEEE, 2005, pp. 886–893.

[8] Felzenszwalb P F, Girshick R B, McAllester D, et al. Object detection with discriminatively trained part-based models[J]. IEEE transactions on pattern analysis and machine intelligence, 2009, 32(9): 1627-1645.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Advances in neural information processing systems, 2012, pp. 1097–1105.

[10] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In Proc. Eur. Conf. Comp. Vis., pages 734–750, 2018.

[11] Zou, Zhengxia, et al. "Object detection in 20 years: A survey." arXiv preprint arXiv:1905.05055 (2019).

[12] Redmon, Joseph, et al. "You Only Look Once: Unified, Real-Time Object Detection." (2015):779-788.

[13] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y.Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in European conference on computer vision. Springer, 2016, pp. 21–37.

[14] Liu, Wei, et al. "SSD: Single Shot MultiBox Detector." European Conference on Computer Vision Springer International Publishing, 2016:21-37.S

[15] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg. DSSD: Deconvolutional single shot detector. arXiv:1701.06659, 2016. 1, 2, 8

[16] Z. Li and F. Zhou. Fssd: Feature fusion single shot multibox detector. arXiv preprint arXiv:1712.00960, 2017.

[17] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Doll´ar, “Focal loss for dense object detection,” IEEE transactions on pattern analysis and machine intelligence, 2018.

[18] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR, 2014.

[19] Girshick R. Fast r-cnn[C]//Proceedings of the IEEE international conference on computer vision. 2015: 1440-1448.

[20] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In NIPS, 2015.

[21] Cai, Z., Vasconcelos, N.: Cascade r-cnn: Delving into high quality object detection. arXiv preprint arXiv:1712.00726 (2017)

[22] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes challenge: A retrospective,” International journal of computer vision, vol. 111, no. 1, pp. 98–136, 2015.

[23] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in Advances in neural information processing systems, 2014, pp. 2672–2680.

[24] Redmon J, Farhadi A. YOLO9000: better, faster, stronger[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 7263-7271.

[25] Joseph Redmon and Ali Farhadi. YOLOv3: An incremental improvement. arXiv preprint arXiv:1804.02767, 2018.

[26] Alexey Bochkovskiy, Chien-Yao Wang, and HongYuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934, 2020.

[27] Berle, Ian. "The Future of Face Recognition Technology and Ethico: Legal Issues." Face Recognition Technology. Springer, Cham, 2020. 163-184.

Appendix A

Self-appraisal

A.1 Critical Self-evaluation

During developing this project, I am satisfied with my overall performance, but there are still some places that need to be improved.

I had acquired many software development skills as well as a deep learning knowledge. Implementing the front-end by Vue.js allows me to have a deeper understanding of using component reuse techniques to reduce code redundancy. The process to combine the back-end framework Flask with deep learning YOLO-v5 API requires me to read large project source code and learn from it. Moreover, when training the deep learning model, I also learned skills to fine-tune the weights to obtain better training results. When writing the thesis, reading many papers in a relatively short time largely enhanced my ability to grasp the gist of a certain paper, and improve my ability for technical writing in English.

However, there also were some places I can improve. Because of the budget limitation, this project only provides local services instead of deploying them online, and training results were not optimal because of the machine hardware limitation. Although there were improvements in my technical writing as mentioned above, there is still an obvious gap compared with high-quality papers, which encourage me to read more papers and write more. I also noticed that I was not proficient in searching papers in the field that I would like to read, the ability to find more related, high-quality papers may demand me to spend more time in the academic research area.

A.2 Personal Reflection and Lessons Learned

1.There was a huge mistake that happened when developing this project, which taught me a lesson and I will definitely avoid it in my next project. During the process of implementing the YOLO-v5 object detection API, I upgraded my IDE Pycharm from the Community version to the Professional version. However, the whole project was crushed and no clear error information existed, and I did not know it was the reason from IDE. After deleting the whole virtual environment in Anaconda and reinstalling them again, the project still can not work. Finally, I reinstall my IDE to the Community version, then everything went well. This experience alarms me that, do not change or update the developing environment including third library, language edition, package version, and even IDE when developing an important

project, because you will never know whether it is the bug you code or the incompatibility cause the error, which will waste much time to find it.

2.Since I used to develop a medium-sized website that contains 10-12 pages using the front-end framework Vue successfully, so this project I still choose Vue for the development. However, this project only has 3 pages in total, while the advantage of Vue is that it has strong support for components, which is designed for code reuse to remove redundancy. Nevertheless, in this project, there do not exist many places for code reuse, but tedious router settings and other configurations make the experience of using Vue in a project with few pages insufferable. This mistake let me reconsider the choice when designing a technique stack, I should be more careful, and analyze the project and the advantages and shortages of each technique.

3.When this project started, I went to the Github page of YOLO, and download the latest release for YOLO-v5 as soon as they put it online. However, it turned out some crashes will happen and there was no similar error information when searching online. After one month, I finally realize the code I downloaded was not stabilized version by checking the updated information on the Github pages. Therefore, I switched to the latest stabilized version and repaired the errors at once. I understood that trying out new things may also mean unknown error, choosing stabled and classical release sometimes is the best choice for careful decision.

4.In this project, I also learned many skills and knowledge which I found to be quite useful. During the model training phase, I learned some skills when fine-tuning the parameters to obtain better results, studied different data set formats, and how to convert them. In the developing phase, I mastered more skills in Vue development. Integrating YOLO-v5 deep learning API was also quite challenging for me and I learned to read the source code of a large project. At last, writing the thesis requires me to do more paper reading which enhanced my reading and writing technique content ability in English.

A.3 Legal, Social, Ethical and Professional Issues

A.3.1 Legal Issues

In object detection tasks, face recognition is one of the most important applications. However, this technique is also controversial, which may raise some legal issues. The most common problems for face recognition technology are privacy and confidentiality since biological features like faces had been largely used in modern society to replace simple passwords for identification. Moreover, excluding commercial applications mentioned above, video

surveillance is also widely adopted for security concerns, aiming at stopping potential terrorism[27] or accident monitoring.

Despite the purpose of monitoring is decent, the right to protect privacy is also important to individuals, and they still lack impeccable legal regulations or law support in this burgeoning field. Solving these legal issues needs constant efforts from governments and legislation departments.

A.3.2 Social Issues

Still, for the face recognition task, the strong desire to protect the privacy of citizens will cause social concerns and sensitivities. As A.3.1 had mentioned, the main applications for this task can be divided into two parts: commercial usage and video surveillance by the government.

For the commercial side, using biometric features could bring convenience to users and reduce identity theft to some degree. However, the data storage for sensitive face features remains a serious problem. It is true that a biological password can not be theft by personal negligence, but a database attacked by professional hackers will lead to thousands of information leakage, which is the main source that causes social anxiety and concerns. Furthermore, people tend to hold distrusting attitudes towards technology companies, some even believe they would sell their personal information for profits. Hence, tackling these social issues from the corporation side still needs every technology company's effort to earn back the trust of people and show them the benefits this technology could bring.

Back to the video surveillance by the government, citizens are quite sensitive to their private information captured by the government, especially after Snowden, but this is also a quite solid solution to prevent potential terrorism. In the famous 911 incident, the images of terrorists are recorded under the camera, they would be stopped if face recognition techniques were developed as the present. This ambivalence between private information protection and social stability will continue under fierce debate, and finally need both sides to reach a consensus.

Additionally, each individual will share multiple identities in modern society (at work, on social media, at home, etc.) instead of only one character. Therefore, to formulate a complex model that corporate all the human beings with their biometric information and manage them together is not feasible. Such a system will cause a huge amount of money and manpower from different organizations, while the system is also not able to guarantee high efficiency and accuracy.

Considering all these above, face recognition, or object detection still has social issues remaining for us to solve in the future.

A.3.3 Ethical Issues

Ethical issues are similar to legal issues and social issues in the object detection field. Using a camera equipped with face recognition techniques in public places to prevent terrorism also violates an individual’s privacy. Using face features of potential terrorists stored in technology companies’ databases to track their activities is equivalent to decrypting terrorists’ cellphones with the phone company, it is right to do so, but may not be ethical. Ethical issues are also cannot be solved immediately, but requires long-term efforts along with society's progress.

A.3.4 Professional Issues

During the training phase, most of the training was carried on by 10 consecutive epochs (for example, 1-10 epochs, 11-20 epochs, etc.), few were adopted by 20 or 30 continuous epochs. The reason for not adopting training 100 epochs or until reaching convergence is due to the performance limitation of laboratory machines. Although there is no solid evidence showing a huge difference between training 10 sets of 10 epochs and 100 consecutive epochs, the results may be altered by this situation. It is difficult to solve this since online laboratories are expensive and also suffered from network issues.

Appendix B

External Materials

Code available from a research group: Yolov5-5.0, whose API is integrated into this project:

https://github.com/ultralytics/yolov5

Data set available from a research group: VOC2007, which is used in training the model:

http://host.robots.ox.ac.uk/pascal/VOC/voc2007/

Ready-made components for template (CSS and JavaScript), used as sign up and log in page:

http://www.bootstrapmb.com/item/9991

Appendix C

Software Manual

Dependencies：

Install through pip (recommended in a virtual): ***flask, pytorch, torchvision, torchaudio, cudatoolkit, flask_sqlalchemy, numpy***.

Nvidia-driver Version: 511.79

Cuda Version: 11.6

Python Version: 3.8

Install the front-end dependencies through npm:

go to directory "***Object-Detection/yolov5-flask/Yolov5-Flask-VUE/front-end***", use command:

npm install

For starting the project:

Firstly, run the front-end framework by (in the directory "***Object-Detection/yolov5-flask/Yolov5-Flask-VUE/front-end***"), use command:

npm run serve

Then, run the "***app.py***" in the directory "***Object-Detection/yolov5-flask/Yolov5-Flask-VUE/back-end***".

Appendix D

Code Repository

The source code of this project can be accessed by visiting the URL on Github:

https://github.com/DAZHAdazha/Object-Detection/releases