Look Ahead: Street View Based Navigation of Roads with Traffic Signs Recognition

Yolo model base traffic sign detection Chrome Extension

Author: Yitong Yu (C22048550)
Degree: MSc Computing
Supervisor: Dr Bailin Deng
School of Computer Science and Informatics, Cardiff University

Sep 2023

Contents

List of Chart

List of Image

List of Table

List of Abbreviations

AI: Artificial Intelligence

CNN: Convolutional Neural Network

CSS: Cascading Style Sheets

DVSA: Driver and Vehicle Standards Agency

FP: False Positive

FPS: Frames Per Second

GANs: Generative Adversarial Networks

GPU: Graphics Processing Unit

GTSRB: German Traffic Sign Recognition Benchmark

HTML: Hypertext Markup Language

HTTP: Hypertext Transfer Protocol

IoU: Intersection over Union

JSON: JavaScript Object Notation

mAP: Mean Average Precision

MPH: Miles Per Hour

ML: Machine Learning

QR Code: Quick Response Code

ResNet: Residual Neural Network

RNN: Recurrent Neural Network

SVM: Support Vector Machine

TP: True Positive

TTK-100: Tsinghua-Tencent 100K

WLAN: Wireless Local Area Network

YOLO: You Only Look Once

YOLOv5: You Only Look Once Version 5

Abstract

Getting familiar with the road and traffic signs is vital for the driver learner. Using a Chrome Extension to help the learner see the street within Google Street View is an economical and convenient option. At the same time, the extension will also provide a traffic sign recognition function and explain the content of traffic signs. The aim is to build a Chrome extension that can find the traffic signs and draw the bounding box, and a window can show and explain the results. This paper is going to explain the project development of a Chrome Extension for Google Chrome Browser. The project is based on the Yolov5 Machine Learning model and uses Flask as the server to deal with the data and cooperate with each component of the Chrome extension. By using the specific selected data for training, the accuracy of the model has reasonably good performance. Besides, the paper fully introduces the structure of the project and the implementation of the extension. It introduces some project management methods used in the project as well.

Keyword: Traffic sign detection, Yolo, Chrome Extension, Flask

Acknowledge

I want to thank my supervisor, Dr Bailin Deng, who guided me with patience and gave me a lot of suggestions to inspire me and help me to solve my confusion during the dissertation writing period.

I also want to say a big thank you to my family and friends. They've always been there for me, understanding and supporting me while I was studying. Their encouragement helped me stay focused and keep going.

1. Introduction

Driving is a necessary skill in our daily lives; getting this skill can increase personal mobility and make our lives more convenient. However, the journey to obtaining a driving license involves a key hurdle — the practical driving test. Statistics from the Driver and Vehicle Standards Agency (DVSA) in 2022 reveal that the passing rate for this test is around nearly 50% (DVSA, 2022). A significant factor contributing to this challenge is learners' struggles with comprehending and responding correctly to the diverse meanings of traffic signs that show on our roadways. It is important for prospective drivers to understand the rules and regulations of driving on the road and obey the rules.

While familiarising oneself with driving may seem straightforward, the practicality of achieving this through traditional means is far from simple. Walking the streets to get familiar with the driving environment is an impractical approach. Moreover, extending on-road learning time is not economical and not environmentally, given the costs associated with driving instructors and fuel consumption. AA Driving School (2023) indicates that certain areas are experiencing waiting times for driving tests that extend up to 24 weeks. Such a significant investment shows the necessity of adequate preparation before facing the actual test.

In this project, leveraging technology emerges as a promising solution. Google Street View, a feature offering panoramic imagery on Google Maps, provides a scenario to virtually travel roads, gaining familiarity with routes and landmarks. Developing a Chrome Extension that seamlessly integrates with Google Street View to recognise and display traffic signs becomes attractive and vital. This extension aims to aid aspiring drivers, especially those preparing for practical driving tests, by acquainting them with the roads they will navigate during the practical examination. By utilising this technology-driven assistant, learners can gain confidence in recognising and reacting appropriately to traffic signs while undertaking practical examination conditions.

The architecture of the Chrome Extension is multi-faceted, using multiple components like a Flask server and the Yolo (You Only Look Once) detection model. The Flask server handles communication between the extension and external data sources, while the Yolo model's object recognition capabilities form the recognition ability of traffic sign detection and recognition. The extension's development is a sophisticated process involving integrating these components, resulting in a user-friendly tool that helps learners prepare for the driving test.

2. Aim and Objectives

With a primary focus on assisting users while navigating Google Street View, this extension offers a unique way to enhance driving learner understanding and recognition of critical road signs and components. The process begins when a user activates the extension within the Chrome Extension interface.

As the user explores the panoramic image of the street view, the extension starts working. The extension will keep capturing the current street image and then sending the image to the local server.

The role of the local server is essential. Upon receiving the captured image, it initiates a call to the detection and prediction Yolo model, which serves as the core part behind recognising the various road components. This model has been trained to detect and classify elements, such as traffic warning signs, traffic lights, speed limit plates, and speed limit markings on the road.

Upon successful detection of these components, the model will let the local server tell the Chrome Extension to draw bounding boxes directly onto the web page. These bounding boxes serve as markers, indicating the presence and location of the recognised road elements. This visual cue ensures that users are promptly alerted to the existence of these components, enabling them to get familiar with the road and make correct decisions while exploring the road environment.

3. Review

3.1 About Machines Learning

Machine Learning is a subset of artificial intelligence (AI) that provides systems with the ability to automatically learn and improve from experience. The learning process is based on recognizing complex patterns in data and making intelligent decisions based on them. Deep Learning involves deep neural networks with multiple layers, learning complex patterns and representations from large volumes of data (Goodfellow et al., 2016). There are many types of Deep Learning. Convolutional Neural Networks (CNN) have become widely used in image analysis and computer vision (Simonyan and Zisserman, 2014). Also, it is one of the most popular models nowadays. Recurrent neural networks (RNNs) are commonly used for processing sequential data, such as speech and text analysis (Vaswani et al., 2017). Generative Adversarial Networks (GANs) has made significant progress in areas such as image generation and style transfer (Radford et al., 2015).

3.2 Historical

Image detection and recognition is one of the core components of the project. And it has been discussed more than fifty years ago (Fradkov, 2020). The methods also have been iterated many times as time went by. The main methodologies of it are Traditional Image Processing, Template Matching, Machine Learning, Deep Learning, and Subsequent Deep Learning.

Traditional Image Processing techniques can be traced back to the 1960s. These methods mainly rely on mathematical and statistical techniques such as edge detection, segmentation, filtering, and morphological operations. Canny's edge detection algorithm (1986) is a representative work of this generation, detecting image

edges through mathematical algorithms. These methods primarily focused on low-level image feature extraction and were used for basic image analysis tasks.

Template Matching involves using predefined templates to recognize specific objects or patterns within images. Brunelli and Poggio (1993) provide a detailed introduction to the underlying theories of template matching. These methods found some success in areas like character recognition and face detection. However, there are some limitations in their adaptability to variations in size, rotation, and lighting conditions, resulting in possible ineffectiveness in complex scenarios.

The beginning of the 21st century breakthroughs in image recognition through Machine Learning. Cortes and Vapnik's (1995) Support Vector Machines (SVM) provided a new solution to image classification problems. Random forests, AdaBoost, and other algorithms became effective symbols for object recognition and image classification. Methods during this generation began to handle more complex image features and patterns but still relied on manually designed feature extraction techniques.

The emergence of Deep Learning marked a significant milestone in image recognition. Convolutional Neural Networks (CNN) (LeCun et al., 1998) greatly propelled this field forward. Krizhevsky et al.'s (2012) AlexNet achieved outstanding success in the competition, directing a new generation of deep Learning in image recognition. Since then, Deep Learning has become the mainstream technology in image recognition, gradually replacing traditional Machine Learning techniques.

With the continuous development of Deep Learning, many new methods and architectures emerged, such as Residual Networks (ResNet) (He et al., 2016), Generative Adversarial Networks (GANs) (Goodfellow et al., 2014), etc. These new methods opened possibilities for image synthesis, style transfer, super-resolution, and

other cutting-edge applications. Simultaneously, new training techniques and optimization methods, such as transfer learning and meta-learning, provided new momentum for the development of the image recognition field.

3.3 Tools

Talking about Machine Learning and Deep Learning, the frameworks and tools are necessary. Nowadays, there are a few kinds of tools that are very popular, and they are PyTorch, TensorFlow, Keras, and MXNet (Ketkar et al., 2021). Machine Learning and Deep Learning is not simple work for most corporations, researcher and developer. If they need to achieve the target by building everything, it would be a sophisticated work with a huge workload. The framework and library can solve this problem. They do the basic tasks for the user and integrate all necessary components. Users can easily use them to do high-level jobs for Machine Learning and Deep Learning directly by calling the function in the coding part. And the framework can support high performance hardware to do the calculation. This allows for a large number of calculations and saves time. For this project, because of using Yolo as the model, and the model is adopted PyTorch as the library. All in all, using the library tools is because their user-friendly design and flexibility feature. So that it can help the users reach the target easier, and provide the possible opportunity for the author to modify the code to fit the project.

3.4 Chrome Extension

Chromes extension refers to the small program that can augment the functionality of the Google Chrome browser, let the user customize their online surfing experience, simplify their operation or get more flexibility function on Chrome. Google published Chrome Extension in 2010, and this extension framework can allow developers to integrate the application into the browser (Chrome Developers, 2023). Chrome

extension is based on using HTML, CSS, and JavaScript technologies. The extension can interact with the web pages and communicate and modify the content.

3.5 Related work

Traffic Sign detection is a hot topic in the industry field and in the academic field. This topic has been discussed for a very long time. Early in 1997, Escalera et al. (1997) realised that traffic signs can provide the driver with valuable information. Traffic sign detection and classification will be helpful for self-driving car research in the future. They use colour thresholding and neural networks to implement the segment and recognition in the project. Saadna and Behloul (2017) conclude a practical way by using three main methods to detect and classify. They are based on colour, shape and the way of learning. The accuracy of their program is probably spread between 90% and 100%, which is a pretty good result. As for higher accuracy, they figured out that using the hand-crafted feature and a suitable CNN model can reach 99% accuracy. The research of traffic sign detection and classification is wider than static images. It could be considered applied in the real-time scenario. Yiqiang et al. (2020) developed a flexible and robust model for real-time recognition. They designed the model as two modules corresponding to two stages in the whole recognition process. First, the model will find out the traffic sign's approximate location, and then the other module will do the classification work within the approximate location image.

Chen et al. (2011) built a robust system for recognition by using a method to identify the colour of the data first, they try to find out the location of the traffic sign by colour. Then using Harr wavelet features to pinpoint the signs. After that, using a tool called Speeded Up Robust Features to comparing the target and the dataset they own. This system has over 90% of accuracy.

In dataset aspect, Wali et al. (2015) introduce some challenges. The abnormal environment in the data may have effect. Such as light condition, the visibility of the data, the traffic sign with chaotic background, broken traffic signs in data and so on. They can affect recognition accuracy to varying degrees. This is necessary features need to be considered in every dataset. Even though the Street View in this project is consisted by some good quality images, but in some condition, the image still exist the noise or light problem.

The type of discussion is broad; traffic sign detection and classification can be a competition held on an online platform, for example, Kaggle. The other kind is using it in the real world for accurate detection and cooperation with some specific action. This project is closer the later one, applied the recognition into the real-world application. So, the project which focus on the applying the model into recognition is worth making reference more.

3.6 YOLO (You Only Look Once) Model

Yolo is a popular computer vision model for object detection. Generally, Yolo divides the input image into multiple small grids and does the detection and prediction at one time. Yolo has developed many versions since 2016 (Redmon et al., 2016). Nowadays, it has been updated to version eight. Each version has a different level of updates and breakthroughs. Besides, Yolo is fully packed and uploaded to the GitHub repository provided for the public to use, and it has an entire discussion community and people who maintain it. Users can use it directly through the cloud computing platform, and it can be deployed on their own computers. All the functions, such as training, detecting, and training visualization, are packed and can be used by a small number of commands, so convenience is one of the advantages.

Early version: The original Yolo is a light weight and quick response model with a straightforward structure. The neural network and directly find out the target location

and classify. It can also be used into detect both image and video. And Yolo can be generalise into many other fields, however the accuracy needs to be improved. In this stage, it has not enough accuracy in positioning and the recall rate is low. In version two of Yolo, the maintenance team do the improvement base on the two shortcomings (Jiang et al., 2022).

Yolo v3: This version of Yolo, for the first time, introduces multiple kernels with different detection sizes. It provides the function that not only one label for each bounding box but also provides better feature extractor ability (Redmon. and Farhadi, 2018).

Yolo v4: Version four of Yolo was introduced by Alexey Bochkovskiy in 2020. Its goal is to balance the running speed and the accuracy so that it can be applied to more scenarios, especially the scenarios in which machines don't have the high-performance calculation ability. The three parts consist of the Yolo version four; they are the backbone using CSPDDarket53, the neck using PANet and the last version, which is Yolo version three, as the head used to detect the target. This version of Yolo is good at accuracy (Bochovskiy et al., 2020).

Yolo v5: The same as the updated version, the developers and maintainers have contributed a lot of time and effort to improve the speed and accuracy of this release. It adopts the Anchor-free technology to let the model become more flexible and increase the performance in different scenarios. Version five of Yolo provides a large-scale amount of pre-train model. Diverse pre-train models can effectively help with future training and detection (Jocher., 2020).

Yolo v6: The version is created and modified by Meituan, which is a large internet company. Meituan added and improved some of the features of the Yolo model, such as the BiC (Bidirectional Concatenation) Module, AAT (Anchor-Aided Training)

Strategy, and Self-distillation Strategy. At the same time, Meituan also optimizes the backbone and neck of the model. Basically, all the improvements are centred around performance (Li et al., 2023).

Yolo v7 and later version: Yolo version seven is a very new version; they have the highest performance among all Yolo models. The same as previous versions, version seven and later version also see efficiency and accuracy as the target. It has surpassed other models of the same level in detecting the object between 5 Frames per Second (FPS) and 160 FPS. It can adapt the hardware to the previous version. The target is going to optimize the training process but not increase the consumption of computing resources. (Wang et al., 2022). The same as the before version, the version after version seven (hasn't published the paper) keep improving their performance and accuracy. The maintenance team always try to improve those two aspects.

4. Problem

Within the scope of developing this Chrome Extension to enhance traffic signs and components understanding through Google Street View, there are several critical challenges that demand thorough consideration and strategic solutions. These challenges, though tough, are not impossible to solve and can be effectively addressed in the subsequent chapters of this effort.

4.1 Connection

For this project to work, some crucial problems need to be solved. One big challenge is the connection between the Chrome Extension and the local server so that the server can call the Yolo to detect the image. This connection is a vital part of the entire work, enabling the extension to accurately recognize and interpret the diverse road components within the Google Street View interface. One of the primary obstacles to overcome is creating a robust communication infrastructure. It is essential to design and implement a method through which the Chrome Extension can transmit the images it captures to the local server to call the detection model. This makes the project need to create a reliable channel that can handle the stable and rapid transmission of image data, thereby ensuring interaction close to real-time detection.

4.2 Dataset

Another important aspect that demands careful attention is the dataset used for training and the adjustment of the dataset in order to get the best out of the model. While the parts of traffic sign detection have mature technology used by other projects or some competition, the suitability of a dataset for the specific environment of this project cannot be underestimated.

One dataset in the field of traffic sign recognition is the German Traffic Sign Recognition Benchmark (GTSRB) dataset, which enjoys substantial popularity (Stallkamp et al., 2012). However, this dataset primarily comprises traffic signs that occupy a large portion of the images. This is on the opposite side of this project. In this project, traffic signs are a smaller fraction of the overall image. Thus, adopting the GTSRB dataset could lead to suboptimal outcomes in practical application scenarios.

Utilizing a dataset like GTSRB in this project could lead to the detection of model behaviours that are inadequate for the original objectives of the project. The model may excel only when users zoom in to a very close proximity of the traffic signs. This limitation contradicts the essence of the project, where users need the model to identify signs as they navigate the Google Street View. While GTSRB does offer the advantage of containing road signs that are similar to those in the UK, its alignment with the project's scenario could be better.

The Tsinghua-Tencent 100k dataset (Zhu et al., 2016) also can be considered and be made a reference. While it offers a diverse range of traffic sign instances, a significant portion of these signs might not correspond to the UK's traffic sign system. Consequently, the dataset's applicability to enhancing the project's dataset or maintaining recognition capabilities is limited due to the significant variation in traffic sign classes. However, the excellent point is that the TTK dataset is not like GTSRB, the sign will pack most of the areas of the image. In contrast, the signs are only a tiny part of the image, and all the views of the dataset are similar to the Google Street View. The perspective of this dataset is well suited to this project.

Image 1. Example of GTSRB

Image 2. Example of TTK-100

Given these complexities, the project may benefit from a well-designed dataset that aligns more closely with the specific use case. This could involve collecting and annotating images of UK road environments and capturing instances where traffic signs are integrated into a broader scene. By creating a custom dataset, the model can be trained to excel in identifying traffic signs when they constitute a smaller portion of the overall image, precisely mirroring the project's objectives.

4.3 Detect model

The selection of an appropriate detection model constitutes the third vital challenge in the project’s development. While many open-source traffic sign detection models are

available, the task requires a model that aligns seamlessly with the project's accuracy recognition needs. Despite the availability of various models designed for competition purposes, their applicability to real-world detection scenarios, such as those encountered in this project, demands careful evaluation.

A central factor that influences the choice of the detection model is the unique nature of the street view images. These images, captured through wide-angle cameras, introduce complexity only sometimes encountered in standard detection scenarios. The non-planar nature of traffic signs in these images and potential image deformations pose a significant challenge to achieving accurate and reliable detection results.

Another critical consideration lies in the flexibility and adaptability of the chosen model. The chosen model should possess the capability to not only handle the existing classes of traffic signs but also to accommodate potential expansion in the future. As the project evolves, the ability to seamlessly integrate additional classes of traffic signs.

4.4 Result showing

The fourth key challenge involves effectively presenting the detection results to the extension's target users: individuals preparing for driving tests seeking to familiarize themselves with the road environment. The success of this endeavour hinges on providing a user-friendly, accessible, and easily understandable presentation of the detected road components.

Understanding that the users may vary widely in their technical expertise, the design of the result window should prioritize simplicity and clarity. The interface should resonate with individuals who may not be versed in computer operations, ensuring

they can easily comprehend the information presented.

Utilizing visualizations that are straightforward and intuitive is paramount. Coloured bounding boxes to correspond with different types of detected components (such as traffic signs, lights, etc.), can provide a quick and visual means of understanding the presence and location of these elements within the street view image.

5. Approach

5.1 Approach of Connection

Addressing the initial challenge of establishing a connection between the Chrome Extension and the detection model involves the implementation of a local server. This server serves as a vital intermediary, facilitating the smooth exchange of data and commands between the extension and the detection model. To overcome this communication difficulty, the creation and deployment of a backend server emerges as a crucial solution.

The backend server operates as a central hub that bridges the gap between the Chrome Extension and the detection model. It acts as a middleman that receives image data captured by the extension and transmits it to the detection model for analysis. Similarly, when the detection model processes the image and generates results, the backend server transmits these outcomes back to the extension.

One of the primary roles of the backend server is to ensure effective coordination between different components of the system. It seamlessly manages the flow of data, directing the movement of images, commands, and results among the extension and the detection model.

By establishing this backend server as a mediating tool, the project takes a significant step toward addressing the challenge of communication between the extension and the detection model. This solution lays the base work for an efficient flow of data, enabling the extension to capture street view images, trigger the detection process, and receive real-time results, all while positioning the project to achieve its core objectives of enhancing road awareness and preparedness for driving tests.

5.2 Approach of Choosing Dataset

To align the dataset with the project's context, the ideal source is Google Street View itself, the platform that the extension is designed to work on. By manually capturing screenshots from Google Street View, developers can directly incorporate real-world application scenarios into the dataset. This approach ensures that the training images closely simulate the content the extension interacts with during user navigation.

The manual collection process guarantees that the images provided to the detection model precisely match the content of the extension processes. This alignment reduces the chances of model behaviours splitting due to differences between training data and real-world scenarios.

While creating a dataset exclusively from manual captures may limit the volume, integrating a smaller portion of similar datasets can contribute to data diversity. By strategically incorporating images from datasets that align with the project's objectives and content, developers can enhance the model's ability to recognize a broader range of traffic signs and components.

The ideal dataset should strike a balance between capturing real-world application scenarios and introducing the variation necessary for training a robust detection model. This involves selecting diverse street views that incorporate different traffic signs, lighting conditions, and environmental contexts.

5.3 Approach of Choosing detecting model

For the third challenge of selecting an appropriate detection model, adopting Yolo (You Only Look Once) version 5 (Jocher et al., 2022) emerges as a strategic choice. This decision is underpinned by a multitude of reasons that align with the project's

objectives and requirements, including Yolo's ability to deliver strong results with limited datasets and its compatibility with small-scale training data.

Yolo's remarkable effectiveness with a limited amount of data is a standout attribute. This is particularly significant for this project, where the availability of large datasets may be constrained. Yolo's dependence on transfer learning and sample enhancement techniques (Li et al., 2018) empowers it to derive meaningful insights even from a smaller training dataset. The dependence on transfer learning ensures that Yolo can leverage pre-trained models to bootstrap its learning process. This accelerates the model's ability to recognize and detect traffic signs and components within the project's specific context. Furthermore, sample enhancement techniques contribute to the model's adaptability and resilience, even in scenarios with limited training data. Even in its earlier iterations, Yolo has demonstrated an exceptional ability to balance detection speed and accuracy. This trait is critical, especially within a real-time interaction scenario like the one presented by the Chrome Extension. Yolo's facility for optimizing these aspects makes it a natural fit for the project's demands. Built on the PyTorch framework, Yolo benefits from the robustness and versatility of this open-source machine-learning library developed by Facebook. Because Yolo uses PyTorch and based on the features of PyTorch, the dataset training step can easily use GPU to help; this will greatly increase the training efficiency. For Yolo, the extensive community support, rich documentation, and availability of pre-trained models ensure that developers can access the resources required for successful implementation and fine-tuning.

5.4 Approach of showing result

Resolving the fourth challenge of presenting detection results to users with differing technical backgrounds and ensuring user-friendliness involves the creation of a reliable result window. This window will serve as the canvas through which users can

seamlessly access real-time detection outcomes straightforwardly and intuitively.

The project expects the development of a dedicated result window that contains the detection results. Upon detection, the window will have the reaction, presenting the user with the relevant results. The basic principle guiding the design of this result window is simplicity. It aims to help any potential user hurdle by obtaining the information in a way that requires minimal operation effort. The window ensures that users can rapidly comprehend the results by stripping away unnecessary complexities.

The contents displayed within the desktop window are synchronized with the detection results produced by the model. This synchronization ensures granting users a real-time insight into their navigation environment. The detection results, as shown by the model, are stored within the database. Subsequently, the result window accesses the database to retrieve and display the relevant information.

6. Applications of the chosen approach

6.1 Application of Connection

For implementing the backend server that bridges communication between the Chrome Extension and the detection model, Flask, a Python-based web framework, emerges as the chosen solution. Flask's selection is bacause of its simplicity, flexibility, and adeptness at maintaining continuous background operation (Ghimire, 2020). Flask's ability to run continuously in the background is a basic part of its role as the intermediary between the extension, the detection model and the result window. This ensures that the server remains available to handle incoming data from the extension, while send back the transmission of detection outcomes to the extension. The advantage of Flask lies in its user-friendly design and smooth learning curve. Its simplicity facilitates easy understanding and rapid implementation, a crucial factor in maintaining a streamlined development process. While Flask is often regarded as a lightweight web framework, it nevertheless incorporates robust HTTP request handling capabilities (Singh et al., 2019). This functionality empowers the server to efficiently process the data like captured Street View images from the Chrome Extension, effectively interpreting and responding to each request. Flask's integration capabilities extend to database management. It provides the necessary interfaces to read data from and insert data into the database. This forms the foundation for the TK window (Tkinter result window), which accesses the database to retrieve and display the detection results (For spare way of displaying the result, they use HTML web page base methods to access the database). Flask's ability to set up different routes to execute various functionalities aligns perfectly with the server's role in this project. By defining distinct routes, the server can effectively manage incoming requests from the extension and route them to the appropriate processes, ensuring a coherent and organized flow of data.

6.2 Application of Choosing Dataset

As mentioned before, most of the training images are screenshots manually by the author. The images cover various angles and distances to the target traffic sign. Covering more scenarios can increase the robustness of the model and can cover more real-life environments. In Google Street View, users can not only use the mouse to navigate on the road, but they also can use the mouse wheel to zoom in or out to see more details on the road. When the user zooms in or out, the proportion of the traffic signs or other components in the image changes. For the detection model, the scenarios are different for each change done by the user. Users can zoom in very close to the traffic sign and cover the whole web page, and signs could be minimal and vice versa as well.

Also, though the dataset is collected from the street view, it must be considered for collection from different cities and countryside. In the real scenario, for example, navigating in a big city centre like London. Google Street View will provide much more information, like restaurants or shopping stores, and they will appear on the map as circles. This is the potential problem causing wrong recognition.

There are a total of 1218 original images collected from many cities in the UK, including Cardiff, London, York, Aberdeen and so on, before sending the images to training. Data enhancement is indispensable. One data enhancement significant result is to improve the robustness of classification models (Tang et al., 2017). Also, this can help improve the accuracy when the traffic sign is not fully in the correct place; for example, some signs may be hit by lorries, and there is an angle of view compared to the normal position. For traffic signs and other traffic components, the amounts of different classes are different. Here is an example: some of the classes, like speed limit 20 and 30, are more common than give way signs. Part of the classes of traffic signs is hard to collect enough data to train. This is the other reason to do the data augmentation to increase the dataset (Ge, 2023). Data augmentation can also increase

the data variety so that the model can handle the traffic signs under sunshine, backlight, nightfall and fog environments. Except for the night, these scenarios will show in street view. There are eight augmentations for the dataset. They are, grayscale, hue, saturation, brightness, exposure, blur, affine and noise. And the total image amount comes to 5119 after all the enhanced operations; all the datasets will be labelled by LabelImg (A desktop labelling software) and then processed by Roboflow (2023). The reason for using this dataset augmentation platform is the users can manage the dataset easily. Users can adjust the augmentation in the visualization interface; this is different from the traditional use of code for enhancement. Before sending the dataset into the training model, all images in the dataset are resized to six hundred and forty by six hundred and forty pixels. The resize action not only can normalize all images but also can reduce the burden on machines and increase efficiency. The action can let the dataset fit the training model as well. Besides, for the training model, the whole dataset is split into three parts; they are Training Set, Validation Set and Testing Set. The portion of them is around eighty-seven point four per cent (87.4%), eight point four per cent (8.4%) and around four point two per cent (4.2%). The amount of them is four thousand seven hundred and thirty-one (4476), four hundred and fifty-four (431), two hundred and twenty-three (212).

Image 3. Origin Image

Image 4. Labelled Image

Besides, there are a total of eight classes for this project. These are warning signs, speed on the road, speed limit twenty, speed limit third, traffic light, No enter sign, keep left sign, and give way sign. The warning sign refers to the triangle plate with a red edge, and the content may be varied. The speed on the road refers to the speed limit painted on the ground. An important point needs to be claimed: in the dataset, each image does not only have one traffic sign or traffic light. This means that each

image could have multiple different traffic signs and traffic lights. There are six hundred and fourteen “warning signs”, one thousand and eighty “Speed On Road” Signs, one thousand and sixty-five “Speed limit 20 mph” signs, two thousand two hundred and eighty-four “Traffic light” items, four hundred and thirty-four “No Enter” signs, one thousand and thirty-six “Keep Left” signs and three hundred and fifty-five “Give way” signs and three hundred and seventy-six “Speed limit 30 mph” signs.

Class	Amount
Warning Signs	614
Speed On Road	1080
Speed Limit 20 MPH	1065
Traffic Light	2284
No Enter	434
Keep Left	1036
Give Way	355
Speed Limit 30 MPH	376

Table 1. Item Amount of Each Class

Image 5. Speed Limit 30 MPH

Image 6. Speed Limit 20 MPH

Image 7. No Enter

Image 8. Keep Left

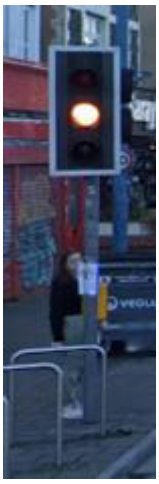


Image 9. Traffic Light

Image 10. Speed On Road

Image 11. Give Way

Image 12. Warning Sign

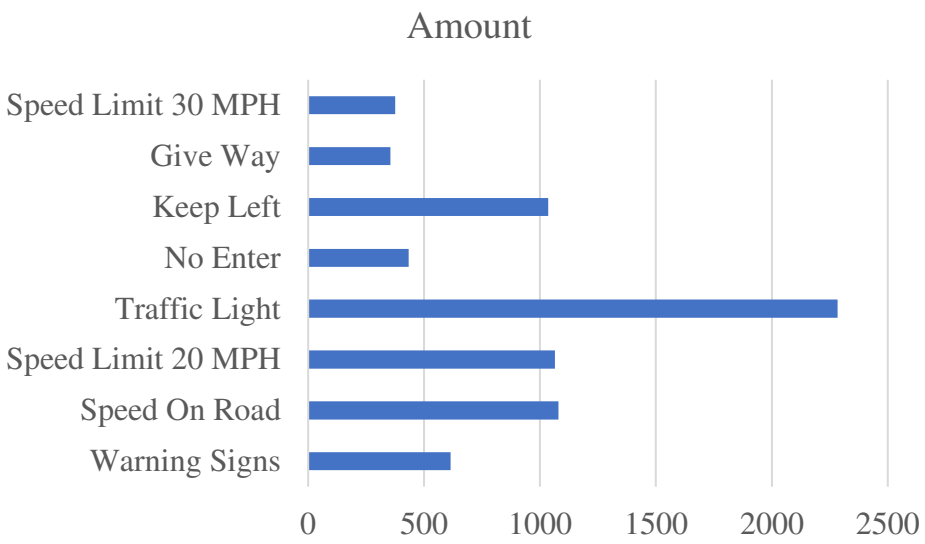


Chart 1. Item Amount of Each Class

However, there is one thing that needs to be pointed out: in traffic sign augmentation, it would be best not to do the flip and rotation because part of the traffic signs is directional. For example, keep left and keep right sign, turn left and turn right sign. If the augmentation is mentioned above, the direction of the image will be the opposite of the label and then confuse the model, finally causing the mistake in training to reduce the accuracy of the model.

Image 13. Left Turn Only and Right Turn Only (GOV.UK, 2023)

6.3 Application of Choosing detecting model

In order to solve the detection model choosing problem mentioned above. A popular model called Yolo version five was selected as the core part of this project. Yolo v5 is available on GitHub (Yolo v5, 2023). One of Yolo v5's standout attributes is that its ready availability. The packed code, well organized and well-documented, simplifies the process for users to access, understand, and deploy the model for their specific tasks. Also, one advantage of Yolo v5 is that it already has good performance, and this can be applied to the project directly with no need to modify. The only one thing that needs to be done is to train the data and gain the weight for use. Especially when tasks of the project need to be balanced. On the other hand, Yolo v5 also supports modifying if the users need to implement specific functions. All in all, the model provides a huge creative space for users. For this project, author did not make the specific change to the Yolo v5 model, by choosing a balanced way, paying attention on how to let the whole Chrome Extension run.

Yolo v5's flexible design allows users to deploy the model not only on local machines but also on cloud computing platforms. This versatility is particularly significant in the context of training.

The model has twenty-five layers in total. It begins with a series of convolutional layers in layer zero and layer one. More convolutional layers are in layers three, five, seven, ten, fourteen, eighteen and twenty-one. And the main mission of them is extracting the feature of the input image. Layer two, four, six, eight, thirteen seventeen, twenty and twenty-three are using the Yolo specific C3 module. Basically, each module contains more convolutional layer to extract higher-level features and let the model accept more complex objects. Layer nine is a spatial pyramid pooling layer, and it can let the model recognize the target in multiple scales. Layers eleven, fifteen, nineteen and twenty-two are UpSample and Concat layers. They can let the model get more detailed information on the input data. Finally, the last layer is the detect layer,

which provides the final results by using anchors and class count.

Layer	Type	Explanation
0	Conv	Convolution layers for feature extraction
1	Conv	Convolution layers for feature extraction
2	C3	YOLO specific C3 modules with multiple convolutional layers
3	Conv	Convolution layers for feature extraction
4	C3	YOLO specific C3 modules with multiple convolutional layers
5	Conv	Convolution layers for feature extraction
6	C3	YOLO specific C3 modules with multiple convolutional layers
7	Conv	Convolution layers for feature extraction
8	C3	YOLO specific C3 modules with multiple convolutional layers
9	SPPF	Spatial Pyramid Pooling for recognizing objects of different sizes
10	Conv	Convolution layers for feature extraction
11	UpSample	Upsampling layers to enlarge feature maps
12	Concat	Concatenate layers to merge multiple feature maps
13	C3	YOLO specific C3 modules with multiple convolutional layers
14	Conv	Convolution layers for feature extraction
15	UpSample	Upsampling layers to enlarge feature maps
16	Concat	Concatenate layers to merge multiple feature maps
17	C3	YOLO specific C3 modules with multiple convolutional layers
18	Conv	Convolution layers for feature extraction
19	Concat	Concatenate layers to merge multiple feature maps
20	C3	YOLO specific C3 modules with multiple convolutional layers
21	Conv	Convolution layers for feature extraction
22	Concat	Concatenate layers to merge multiple feature maps
23	C3	YOLO specific C3 modules with multiple convolutional layers
24	Detect	Detection layer with anchors and class count for object detection

Table 2. YOLO v5 Structure (Jocher et al., 2022)

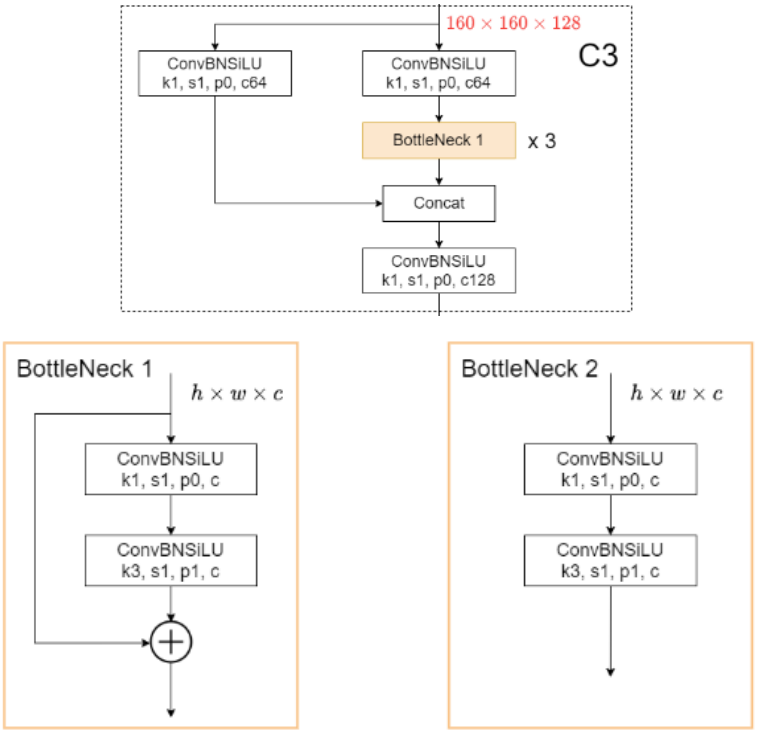


Image 14. Yolo v5 C3 Module (Jocher et al., 2022)

For this project, the training model is set to deploy Yolo v5 on a cloud computing platform called Featurize. The reason is that the cloud computing platform can rent high-performance hardware (Nvidia GTX 4090, AMD EPYC 7453, 64GB Memory) for training in order to improve the efficiency so that it can earn more opportunities to adjust the dataset and suitable model. In conclusion, choosing to rent a cloud computing platform can obtain more opportunities for trial.

6.4 Application of showing result

To deal with the result showing issues, the project adopts SQLite 3 as the chosen database solution and uses the TK window to show the result. SQLite, a versatile and lightweight database system, is an optimal fit for storing and processing the project's relatively lightweight data requirements (Bhosale et al., 2015). The nature of the project, which involves recording and storing detection results, is well-suited to

SQLite's capabilities. The database's design accommodates the storage of individual rows, each representing a detected result. This design choice minimizes complexity while ensuring efficient data organization. The TK window accesses the SQLite database to retrieve and display the detection results. This integration ensures that the displayed information is synchronized with the database's contents, providing users with an accurate and real-time report of the detected traffic signs and components. The TK window is designed to automatically refresh at fixed intervals, providing users with the latest detection results. This independent behaviour further enhances user convenience and minimizes the need for active user engagement. Also, this project provides a spare way for users to see the result by using the window written by Electron.js. Basically, Electron.js provides a window where the content inside the window is based on a web page. The reason to choose Electron.js is because it is easy to be controlled by the author to implement the specific operation. For example, control the window maintained on the top of the screen among all opened windows. So that when the user navigates in the Street View, the result window will not jump to the back, and users cannot read the results. In other words, the aim of the spare way is to implement the same effect of the main way (using the TK window) of showing the results.

7. Products

7.1 Requirement

The project's requirements are going to enhance the driver learner's knowledge of traffic signs and road components and driving test preparation through the development of a Chrome extension that integrates with Google Street View. The extension's functionalities contain real-time traffic sign detection, visual representation, and live result presentation. The core requirement entails the extension's capability to detect traffic signs within Google Street View as users navigate through it. This real-time detection process is a crucial element in empowering users with accurate and live information about the road environment. To provide users with a tangible representation of the detected traffic signs, the extension must draw bounding boxes directly onto the Google Street View web page. These bounding boxes visually indicate the location of traffic signs within the Street View page. Integral to the project is the development of a live result window that presents users with real-time updates of the detection outcomes. This window is an illuminating dashboard, providing users with clear explanations of the detected traffic signs and their implications. All in all, this part is very similar to the Aim and Objective of the project.

7.2 Design and Implementation

The whole project mainly consists of two parts. The first part is the Chrome extension part, which is run on the Chrome browser. Users need to install the Chrome extension before they use it. The other part is the local server, which deals with the detection function and the result window. The local server needs to be opened before using it every time. As mentioned before as well, the result window is decided to use Tkinter. However, in order to increase flexibility and prepare a backup plan, a web-based page

also can access the result. It can be opened by the normal webpage; it also can be opened by the desktop window created by Electron.js. All in all, there are three main ways to get the results. Electron.js is chosen to replace the TK window as the spare plan, it is a desktop window and it bases on the JavaScript (Rao et al., 2022). And the advantage is mentioned before, Electron.js can control the place of the window and show the web content in it. For the webpage access method, a web technology called WebSocket is adopted. Because the convenience and the efficiency it provided, it reduces the communication between the local server and the client (Pimentel and Nickerson, 2012). It can save the resource of the local server and reduce the access load. A QR code is provided as well; the user can use it to access the result by mobile phone browser once the phone is in the same Wireless Local Area Network (WLAN) with the local server. Basically, the page mobile phone access is also a web page version, which is the same as the content of Electron.js. Users can choose any one way to read the detection result according to their habits.

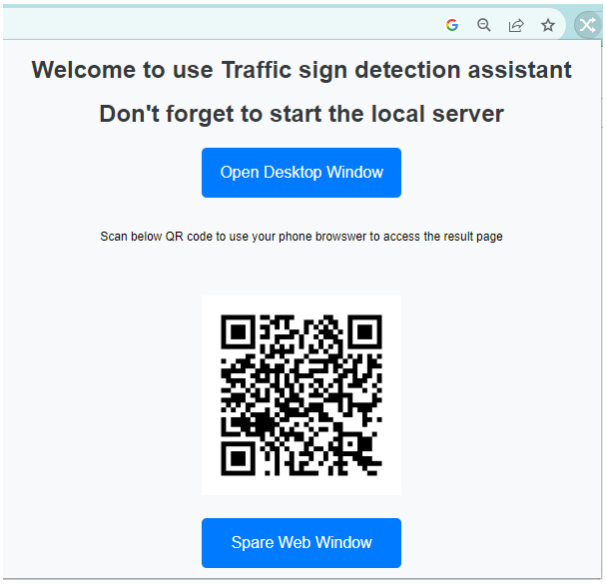


Image 15. Extension Page

Here Is the general flow of how this extension works:

Assume a user has already correctly installed the Chrome extension and started the local server. Users will go to Google Maps, select the route they would like to go and

go into the street view interface. The extension will check whether the user is on the Google Maps website by checking the page address. If a user is on the street view page, the extension starts to work. At the same time, the user can open the desktop result window through the icon of the extension in the web browser. On the same page, the QR code is also shown there. It is because when the local server starts, it will gain the IP address and generate the code image to send there.

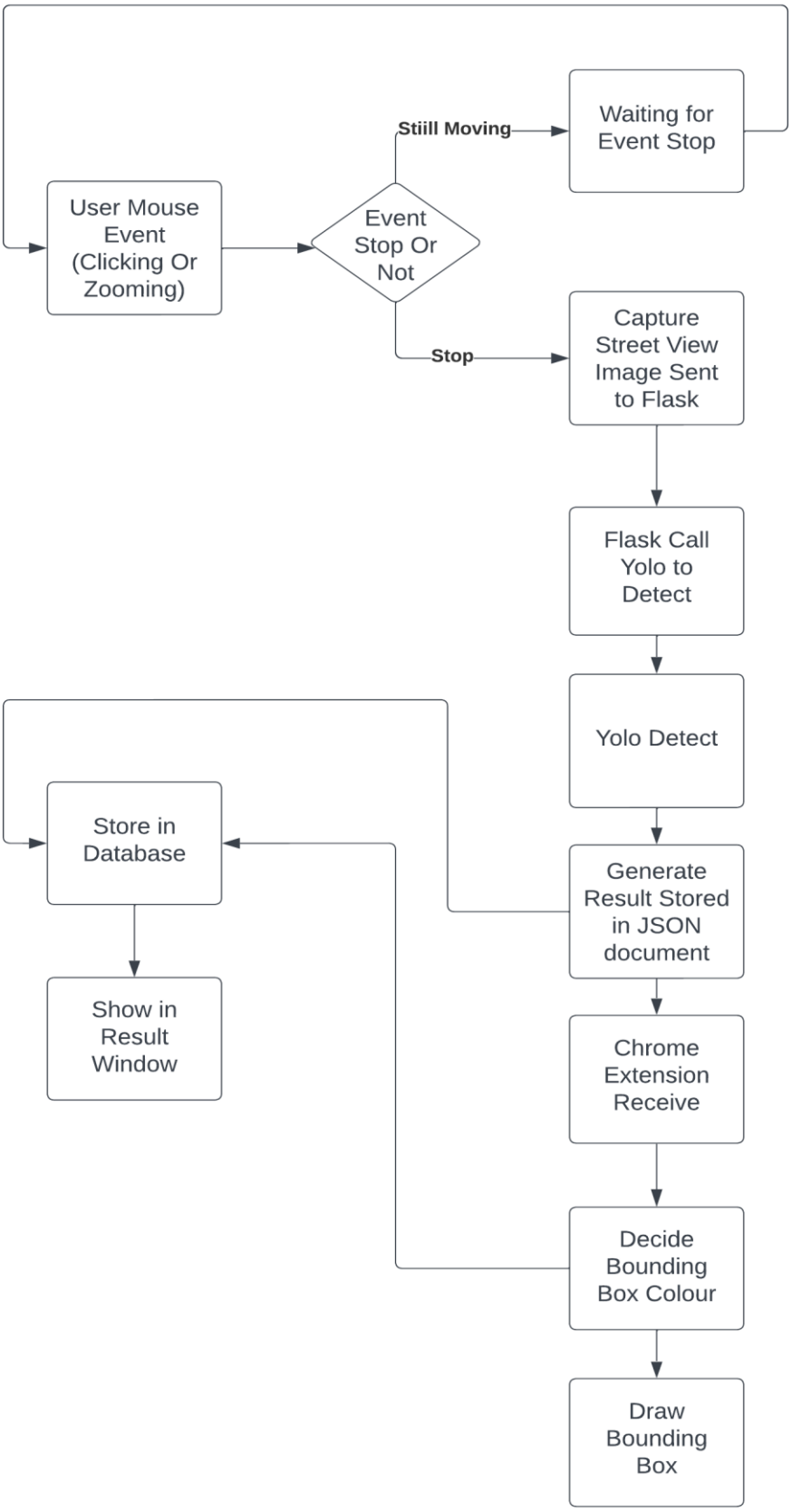


Chart 2. Each Clicking Running Flow Chart

The extension will keep listening; when the user has a mouse event, including a click and wheel event, the Chrome extension will capture a web page screenshot and then send it to the local server. Then, the Chrome extension will keep listening to the message from the local server.

Once the local server receives the image from the extension, it will call the Yolo and start detecting. After detecting, the Yolo will return the result to the local server in JSON (JavaScript Object Notation) format. Using JSON format is suitable for solving the problem that the local server and the Chrome extension are written by different programming languages. The local server is written by Python, and the Chrome Extension is based on JavaScript. If there exists the detection result (Traffic sign or Traffic Light in the image). The local server will send the result data to the Chrome extension. The extension will decide the colour of the bounding box. The decided colour will be sent back to the local server and then go into the database in order to let the colour in the result window correspond to the colour of bounding box. At the same time, the Chrome extension draws the bounding box on the web page. All the data will be stored in the SQLite database.

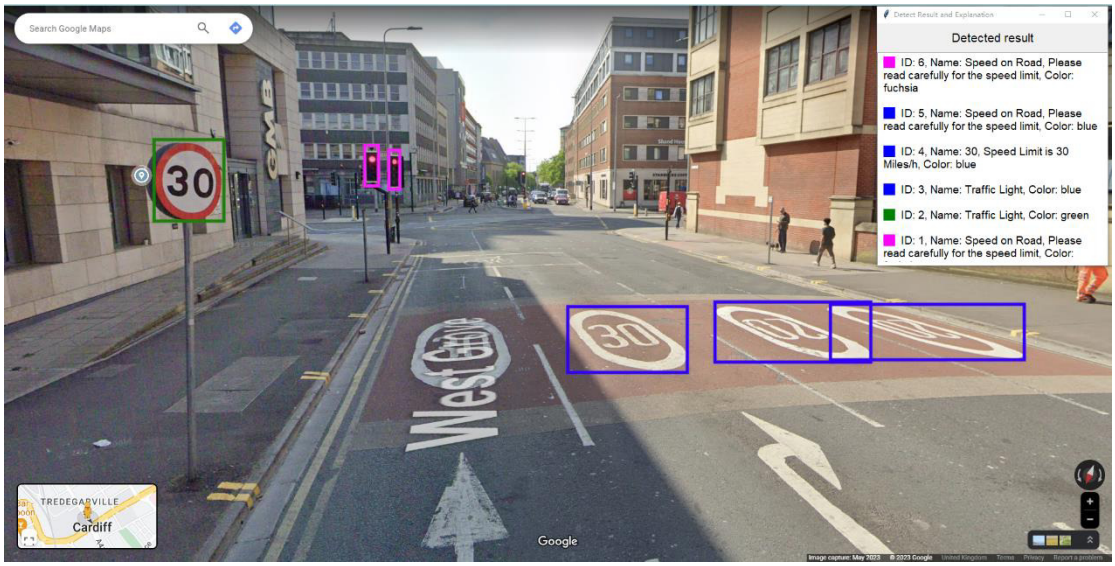


Image 16. Extension Effect Image (TK result window version)

Assume that the result window is already opened; it will keep accessing the database and refresh the window to show the latest result. There is a point needs to be cautioned; the result window is not going to show the history result, which means it will only show current detected results. Every time the user clicks and uses the wheel to zoom in or zoom out, the existing detected result will be cleared, and once the user stops the mouse event, the Chrome extension will do the next detection.

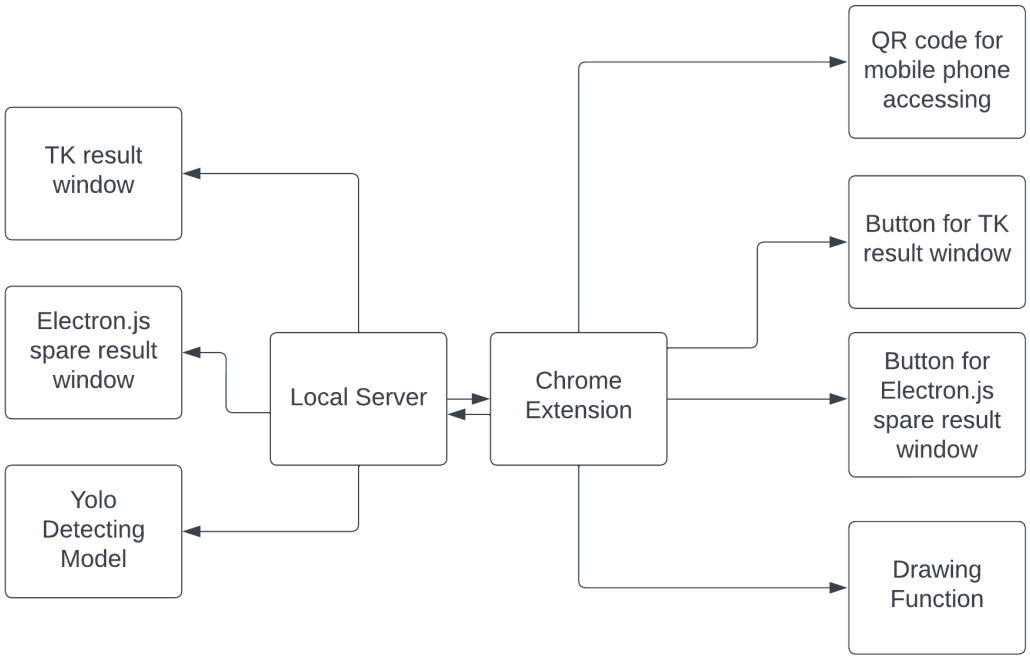


Chart 3. Chrome Extension Components

7.2.1 Chrome Extension

JavaScript is the primary programming language responsible for the Chrome extension part; it will cover most of the actions that happen in the extension and the web page. Chrome Extension consisted of popup.html, background.js, content.js and manifest.json. manifest.json is the fixed Chrome extension format document. It claims the extension and the whole extension can be read and accepted by the Chrome browser, depending on this document. Content.js is responsible for the whole page

working. Specifically, the content.js document is going to capture the screenshot, listen to the mouse event, decide on the bounding box colour and draw the result bounding box. Background.js is used to listen to the whole extension event. It controls sending and receiving messages, such as captured images and detected results, with the local server. In other words, background.js is responsible for transferring the information between the local server and Chrome Extension. In addition, the content.js, popup.html, is used to handle the page when the user clicks the Chrome extension icon on the right-top corner of the web page. The popup.html is the place to let the user open the result window; it shows the QR code for the user to use smart phone to scan as well.

7.2.2 Local Server

The local server is written in Python, and the development framework is Flask, as mentioned in the approach application chapter. This is one of the core codes of the whole project, and it handles the connection with each component of the program. Basically, there are many routes set in the Flask; each route has a specific function. For example, it uses the route to get the image from the Chrome extension and can use the specific route to call the Yolo to predict.

Also, the Flask server handles the database connection by another specific route. As we mentioned before, each user does the mouse event, the program will make the new detection progress, and the database will be clean and prepared for storing the next time result. This is also done by the Flask local server, each action has its own route to execute.

At the same time, the Flask server also handles the result window open function. It is listening from the button event in the Chrome extension popup page. Once the users click on it, the Flask server will call the showing result window function and make it

the subprocess of the program in order to let all the components be under control. When the server shuts down, the result window will close automatically.

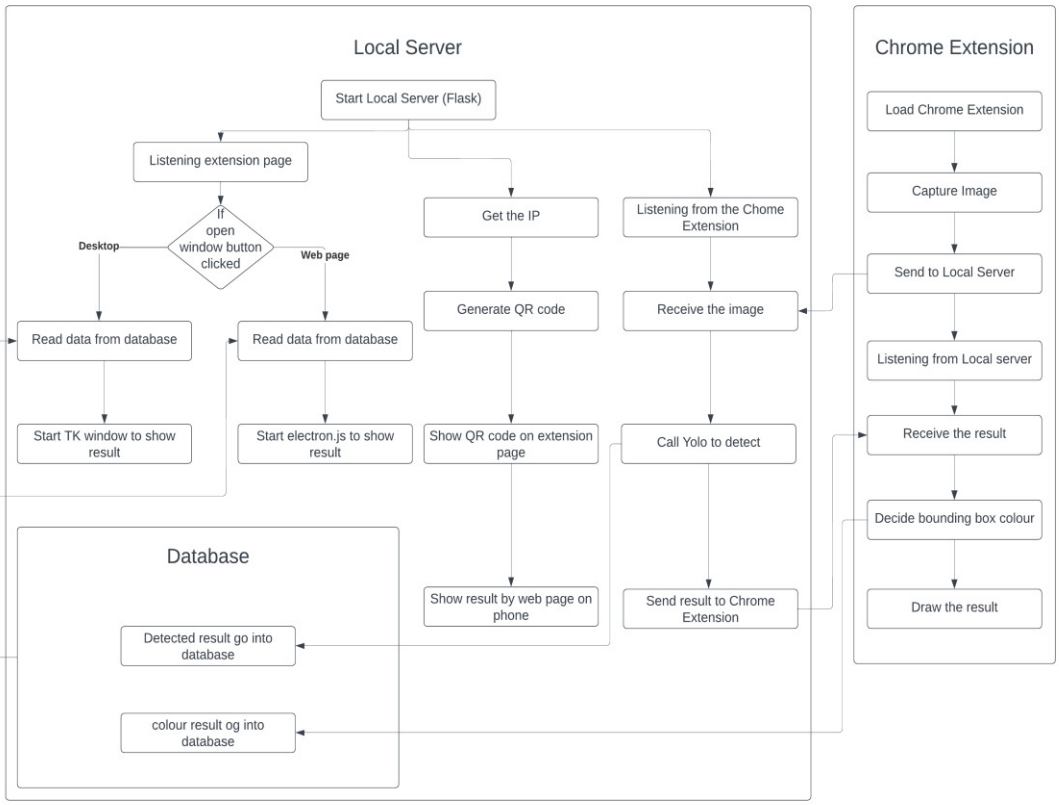


Chart 4. Chrome Extension Working Flow

7.3 Project Management

The project's development adheres to the agile methodology, a widely adopted approach in software engineering that facilitates iterative and evolutionary progress throughout the project lifecycle (Boehm, 2007). This methodology has evolved, gaining popularity since the mid-1990s for its effectiveness in managing complex development tasks (Williams, 2010). The adoption of the agile methodology involves breaking down the project into smaller, manageable tasks that can be accomplished within a two-week iteration cycle. This approach enables the project to advance in gradual steps, with each iteration building upon the accomplishments of the previous one. This iterative nature not only enhances progress visibility but also enables

adaptability to developing requirements (Lan et al., 2009). One of the key strengths of the agile methodology is its capacity to adapt to changes and growing priorities. As the project develops, the iterative nature allows for adjustments based on new understandings, feedback, and shifting requirements. This dynamic responsiveness ensures that the project remains aligned with the intended goals and outcomes. The project integrates the Kanban methodology, a visual system for managing workflow and stimulating continuous improvement (Zayat and Senvar, 2020). Adopting Trello as a tool, the project visualizes tasks within each iteration. This tool is particularly effective in solo development scenarios, offering a clear overview of task progression and providing author-driven project management. The Kanban model provides a structured framework to monitor project progress even in a solo development setting. This visual representation enables an intuitive understanding of completed tasks, ongoing work, and upcoming activities. As a result, the author gains enhanced control over project dynamics and can prioritize efforts accordingly. The Kanban methodology enhances the author's efficiency by streamlining task management and creating a tangible sense of achievement with each completed task. Using this methodology during the whole development period contributes to sustained motivation and focused productivity (Kaur, 2018).

7.3.1 User Story

As a driving learner, I would like to navigate in the Google Street View to get familiar with the street. The Chrome Extension can draw the bounding box when I meet the traffic sign. There is also a result window on the corner of the page that can explain the traffic sign for me.

7.3.2 Acceptance Criteria

Given a traffic sign appears in the Street View, when the extension is active, then a bounding box should be drawn around the detected traffic sign.

Given a traffic sign is detected, when the user views the bounding box, then a result window must appear with a clear explanation of the traffic sign.

Given the user is navigating through Street View with the extension active, when they move from one location to another, then the extension must detect and explain traffic signs in real time or with minimal delay.

7.3.3 Test Case

In software engineering, providing a test case for a product is also a necessary part. Tester or other users can quickly test the product comprehensively and figure out the potential faults and errors. The chart below will provide a version of the test case. It specifies all the details of the project including the system environment, software environment, precondition and details of each step. It tells the tester where to do the operation and what is the result of the operation. During the development of this extension, when the testing, all the steps in the table below are passed. In general, the test case is for other tester using to have a review for the software. Due to this project is basically satisfied the aim, so the pass or fail result is added into the table as well.

Test Case Id: 1	**Test Purpose:** Check if the whole chrome extension function work		
Environment: Window 10 Home (22H2), Anaconda 23.7.2, Python 3.11.3, JavaScript 18.6.0, Google Chrome 115.0.5790.171 (Official Build) (64-bit), Flask 2.3.2, PyTorch 2.0.1 + cpu			
Preconditions: The local server is started, the chrome extension is added into the chrome browser, all the requirement are installed. User is already in Google Street view.			
Step No.	Procedure	Expected Response	Pass/Fail
1.	Click the extension icon on the right top corner of the Chrome browser.	A page will show buttons and QR code. Buttons are for use to open result window. The names are 'Open Desktop Window' and 'Spare Web Window' QR code for mobile phone scanning.	(Pass)
2.	Click the 'Open Desktop Window' button	A tkinter window will jump out on the right top corner of the screen.	(Pass)
3.	Use mouse to navigate within the Google street view.	If there are traffic sign in the Street view. Bounding box will be draw on the page. Result will show in the result window. The colour of bounding box is correspond to the colour block in the result window.	(Pass)

Table 3. Test Case

7.4 Result

On the software development aspect, the main objectives of the project have been accomplished. The Chrome extension demonstrates the capability to detect a wide range of traffic signs within the dynamic context of Google Street View. This real-time detection process operates as envisioned, furnishing users with timely and precise information about their surroundings while navigating through the street view environment.

However, the bounding box is not going to show all the time. When the user is doing the mouse event, in other words, when the user clicks the mouse or uses the wheel to zoom in or out, the bounding box will disappear. Only when there is no mouse event, the Street View, in a static situation, the bounding box will be shown. This is because the transmission between the Extension running on the web browser and the local server needs time. The local server calling the Yolo to recognize the image needs time as well. If the Extension keeps showing the bounding box, the result will not stay in the correct place. In case to avoid this situation, an event listener is added in the Extension to listen to the mouse event within the web page. When there is a mouse event, the bounding box will be hidden temporarily. This will make the result more straightforward and not going to misleading the user.

On the other side, detection and recognition are the core parts of the Extension and play an essential role by using the Yolo v5 model and the dataset discussed in section 'Application of Choosing Dataset'. For this project, the mAP (mean Average Precision) value is going to be discussed. Two kinds of mAP are provided by the model: the mAP50 and mAP50-95. The meaning of the number after mAP is the threshold of Intersection over Union (IoU).

IoU formular:

$$IoU(Intersection\ over\ Union) = \frac{Overlap\ Area}{Union\ Area}$$

Mean Average Precision is an evaluation metric to make an assessment of the machine learning model (Deval, 2023). mAP50 means the Mean Average Precision value when the Intersection over Union thresholds is 0.5.

Except for the Mean Average Precision, there are other values that can be used for reference. They are Precision, Recall and F1-Score. Precision means that the model's ability can actually correctly classify the specified class. Recall is the standard that shows the model's ability to grab all the items for a class. F1-Score is going to demonstrate the balance ability of the model (Sportelli, 2023). Because the model has to make a balanced choice between Precision and Recall. It is easy to bias towards one indicator if focuses on one standard too much; using F1-Score can observe the model in a neutral and objective way. In other words, that means the ability of the model detecting and recognising at the same time.

Average Precision formular:

$$Average\ Precision = \int_0^1 P(R)dR$$

Mean Average Precision formular:

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i$$

Precision Formular:

$$Precision = \frac{True\ Positive\ (TP)}{True\ Positive\ (TP) + False\ Positive(FP)}$$

F1-Score formular:

$$F1 = \frac{2\ \times (precision\ \times recall)}{precision + recall}$$

Recall formular:

$$Recall = \frac{True\ Positive(TP)}{True\ Positive(TP) + False\ Nagetive(FN)}$$

The model reference result is provided by the table below:

Class	mAP50	mAP50-95
All	0.928	0.775
Speed Limit 20 MPH	0.893	0.746
Speed Limit 30 MPH	0.882	0.806
Give Way	0.936	0.805
Keep Left	0.961	0.726
No Enter	0.927	0.75
Speed on Road	0.931	0.854
Traffic Light	0.944	0.74
Warning Sign	0.954	0.776

Table 4. mAP Value for Each Class

Generally speaking, all classes got 0.928 in the mAP50 level. That means when using this model to detect traffic signs, it should return a reasonable, accurate result. When the standard goes up to the mAP50-95 level, the IoU threshold increases, the performance of the model decreases to a certain degree, and the value is 0.775. Among all classes, the “Keep Left” class has the best result, in the mAP50 level, the value is 0.961. But in the mAP50-95 level, “Speed on Road” has the best result, which is 0.854. For the worst result, “Speed Limit 20 MPH” and “Speed Limit 30 MPH” are the two classes lower than 0.9 in the mAP50 level. In mAP50-95, all classes are more average. As the training went by, the meaningful indices like Precision, Recall and F1-Score were close to 0.9 at the end. For all classes, the model has the ability to classify the traffic signs at around 0.9 level. This means the model has a sufficient ability to classify the class, and the performance of accuracy is enough as well.

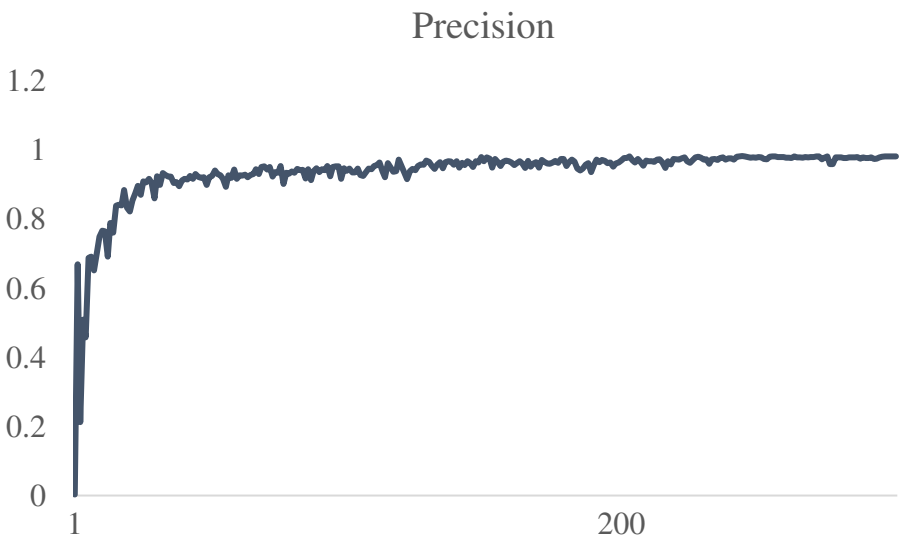


Chart 5. Precision

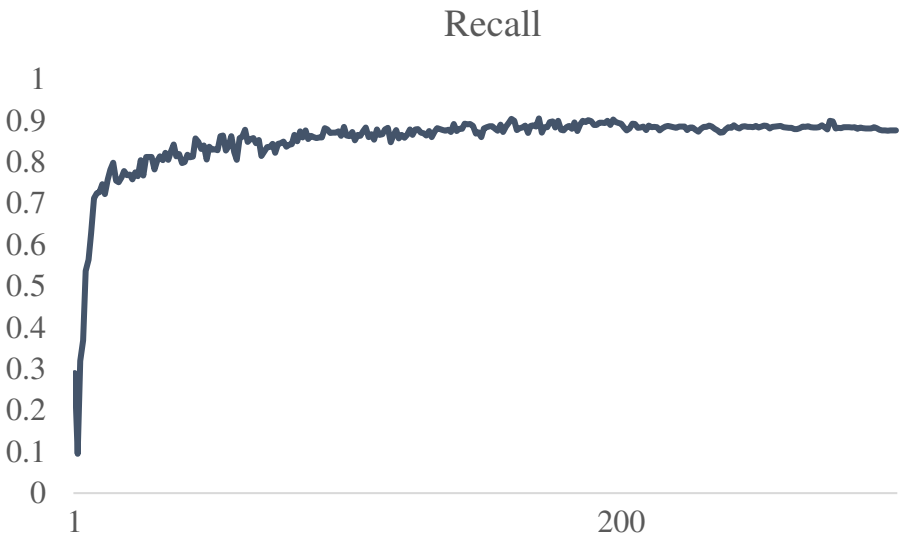


Chart 6. Recall

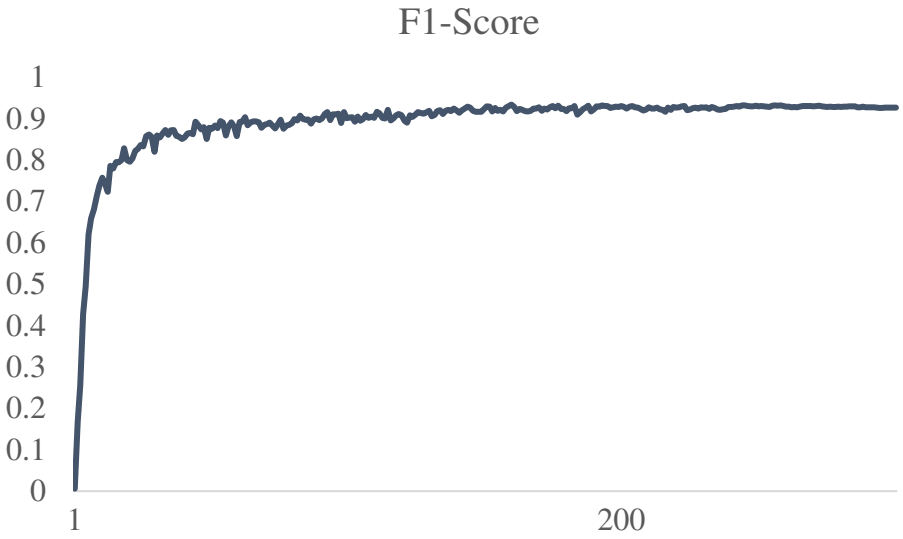


Chart 7. F1-Score

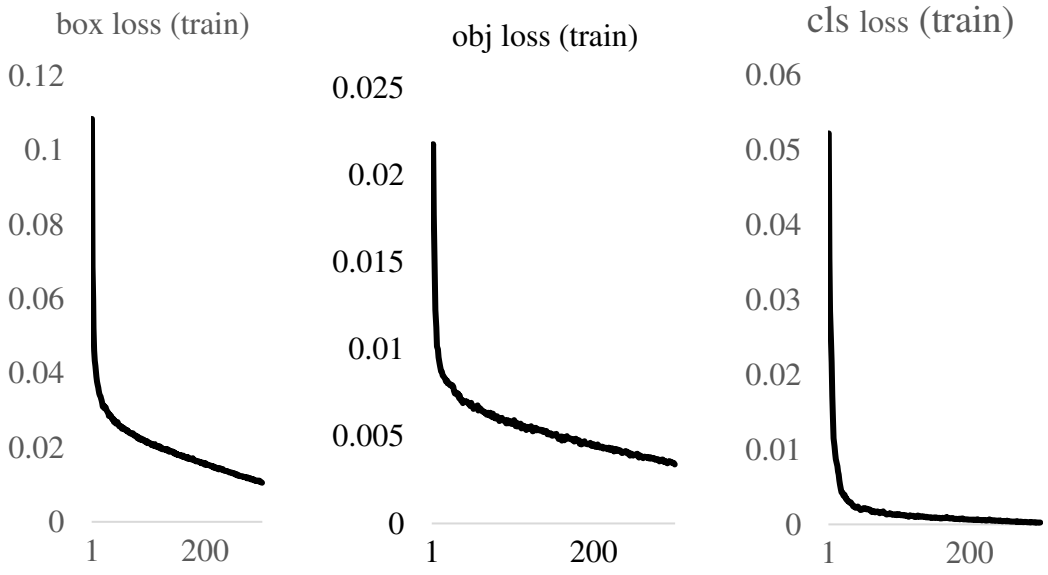


Chart 8 Training, Objectness and Classification Loss

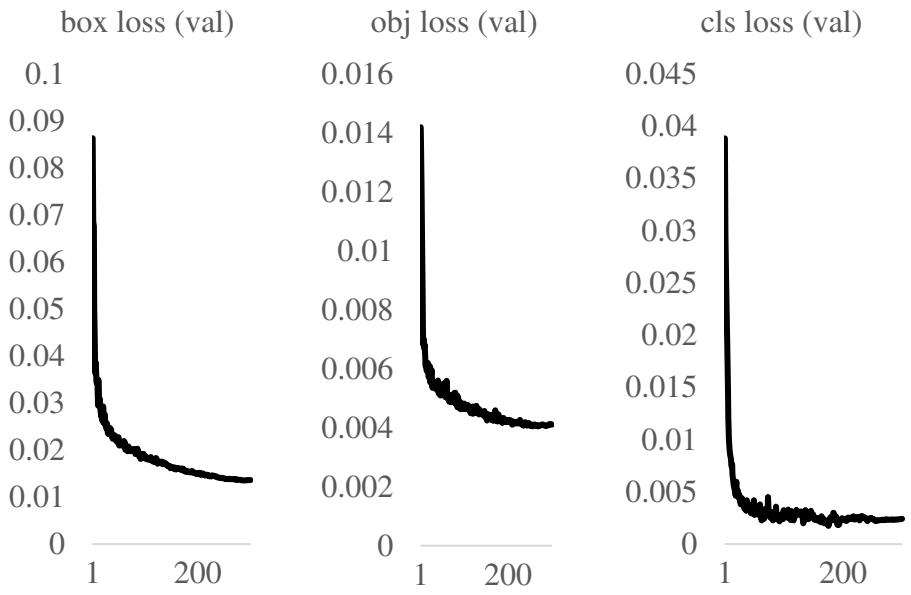


Chart 9. Validation Box, Objectness and Classification Loss

The Choosing model Yolo v5 also provide the data of Loss while training so that it is easier to monitor the training progress and figure the not ideal situation in the right time. There are two main kinds of Loss data provided, they are about training and validation. For each kind, Yolo gives three subset loss data. They are Box, Objectness and Classification. For Objectness, this is measuring how accurately the network predicts the presence of a target, in other words, the presence or absence of a target in the grid. For classification loss, it is going to measure the accuracy of the network in predicting the category of the target. For Box loss loss, this is the indicator to find out the difference between the network's predicted bounding box and the actual bounding box, including centre location and box size. Yolo assign them the weight and combine these three figures to calculate the loss.

According to the line chart shows the training loss for this project, all losses are in the low value. Classification is the lowest one, whatever in training or validation. It is lower than 0.005 in validation and lower than 0.01 in training. After that is the Objectness loss, it close to 0.004 in validation and lower than 0.005 in training. The value of box loss is higher relatively compare to the other two. It converges to

between 0.01 and 0.02 in validation. In training, it is lower than 0.02 as well. All the line chart are finally maintain in the reasonable value.

Yolo Loss Fomular:

$$Loss = \lambda_1 L_{obj} + \lambda_2 L_{cls} + \lambda_3 L_{box}$$ (Jocher et al., 2022)

Generally speaking, the loss is the difference between the target detected by the model and the real correct target which labelled in the pre-process stage. The lower loss value means the model can get the better predict. For a reasonable result of the loss, it will converge to the very low value smoothly with the iteration of training, the lines will flatten out without too many strong fluctuations. In addition, training loss and validation loss should receive the same attention. Because there are two kinds of situation need be to be cautioned, they are underfitting and overfitting. Through monitoring the loss, if training loss and validation loss are both in high value, that means the underfitting, the model cannot catch the data feature. The other one, overfitting, during the training, the training loss and classification loss are both going down, but in some time, the validation loss starts going up or shows some fluctuation (Google for Developers, 2023). That means the model may absorb too much feature and noise.

8. Analysis

Although the Chrome extension can reach the target set as before, there are some parts that can be improved. First of all, the classification dataset can be increased. Traffic sign is varied, and more class of traffic signs can be included. But this is challenging because most of the traffic signs are in circles (regulatory signs) and triangles (warning signs). In other words, the edges of a lot of traffic signs are the same, but the content is different. This will lead to difficulty in recognition. The dataset must be large enough to be sent to training, and the model needs to do more figures to adapt to the scenario. At the same time, according to the mAP value in this project, some traffic signs like "Speed limit 20 MPH" may have similarities to others. One way is to increase the training dataset; the other way is to adjust the training model to fit the class of traffic signs that have a high degree of similarity.

Second, the whole speed of the Chrome extension can be faster. In this version of the extension, when the users do the mouse event on the page, they need to wait a while, about one to two seconds, to wait for the bounding box to show. This issue is also discussed in the Result chapter. This is because a few steps are behind these few seconds. Capture the image, send it to a local server, detect, send back the result and draw the bounding box. These steps rely on the speed of the internet, server responding speed, and the model detecting speed. All of these can be increased. Once this part is increased, the problem mentioned before, the bounding box showing the issue, would probably have a way to be solved.

Third, the server of the extension can be upgraded. The server can not only deploy on the user's computer because this will have a certain degree of requirements, like the performance and the software environment for the user's computer. There are two main ways to improve. The server code can be packed as a software application, and the user only needs to install it with one click; this will make the use of the whole program easier. The other way more convenient is to deploy the server on the cloud

computing platform. Then, users will not need to do anything about the server; what they only need to do is install the Chrome extension on the web browser and it will automatically call the rest of the functions from the cloud platform to do the same task mentioned above.

More potential problems and designs can be considered. For example, the extension can set up the user configuration. For example, some users may not be familiar with one kind of traffic sign or component. The extension can be set as a preference to give more instruction on that kind of sign or component to specific users. In other words, the extension can make different optimization bias settings or configurations for different users. At the same time, the security problem also needs to be addressed. Especially once the extension has the preference of the user, the user privacy and the user data must be handled in a way that is compatible and reliable with the local requirements of use.

Besides, developing the Chrome Extension, the extension can provide a place for users to put on their suggestions. So that during the development life cycle, the author can receive the opinions from users with the most realistic feelings. This option can let the author have more ideas to improve the extension and can make the extension more suitable for broad users. Lack of feedback is not enough for this project and can be changed and improved in the future.

9. Conclusions

The paper reviews the past of Machine Learning and Deep Learning. Also, take a look at the Chrome Extension and Yolo model in order to prepare for the project. Also, knowledge of HTML, CSS, JavaScript, Flask and Python is critical. Adopting Agile with Kanban project management method is valuable and helpful for monitoring the development progression, and it is possible to make the best adjustment to fit the requirements during the development.

The aim of the project is to build a Traffic sign recognition Chrome extension that uses Yolo v5 as the detection model and can draw the result bounding box directly on the Google Street View web page. Over a period of time developing, the extension can achieve the basic function assumed at the beginning. The dataset selected and the model can let the mAP value reach a reasonable level. The paper checks the loss value as well, and all the line chart and the figure show the acceptable situation for this project. And other indicators like F1-score, precision and recall show the model has a not bad performance when using it to detect the reality situation. Besides, the extension provides multiple ways for users to access the result. Users can access the results through the TK window, Electron.js window and access through the smart phone web page. The paper lists the main problems met during the project and explains how to solve them. The paper also discusses the potential improvements that can be made in the future. For example, the speed of reaction, more functionality can be added, the personal configuration option and so on. All in all, the project reach the target in this stage, but more can be improved.

10. Reflection and Learning

During this project, a lot of new knowledge has been learned, and the ability has been improved. Programming ability is one of the major points. In this project, the main programming language used is JavaScript and Python. Both writing skills and construction coding skills improved. The ability to communicate, coordinate and work with different programming languages is also enhanced. Especially connecting the result between the Yolo v5 and the Chrome Extension to draw the bounding box. This needs skills to find a specific format of document which can store the result data so that can do more operation between the different components among the project. The project management skills also get a deeper understanding. Time allocation is very important for developing a product that lasts more than a month. Proper assignment of tasks allows for more flexibility and facilitates later modifications and enhancements to the project to meet requirements. If the distribution of tasks between the early stage and the late stage is uneven, and there are too many tasks in the late stage, the whole project will be out of balance, and it will be easy to mess up in the late stage of development, which will lead to some unpredictable and bad results, such as potentially undiscovered bugs and so on.

At the same time, the programming ability includes neat lines of code, readability, maintainability, and the ability to upgrade existing code in the future needs a long way to go to be improved. Continuous learning is also an essential skill for a developer. In the process of development, there are always different kinds of problems. Memorization alone is not enough to solve the problems encountered; there are always problems that have never been encountered before. Therefore, rather than memorizing all the solutions to problems, it is more important for developers to have the ability to solve problems, such as the ability to reasonably use the resources on the Internet to find the problems encountered, read the developer's documentation, and visit the technical idea exchange forums. Or ask for help from experienced people around you. Finally, the accumulation of time to form their own experience. There is

no such thing as an overnight success but more of a failure to move forward, just as the development process will always encounter different errors. To calm down, do not give up on the problems encountered in the study should always keep in mind.

11. References

AA. 2023. AA Driving School reveals bookings delayed at 88% of driving test centres. Available at: https://www.theaa.com/about-us/newsroom/driving-test-waits-2023. [Accessed: 5 Sep 2023]

Bhosale, S.T., Patil, T. and Patil, P. 2015. Sqlite: Light database system. Int. J. Comput. Sci. Mob. Comput, 44(4), pp.882-885.

Bochkovskiy, A., Wang, C-Y. and Liao, H-Y. M. 2020. YOLOv4: Optimal Speed and Accuracy of Object Detection. Available at: https://arxiv.org/pdf/2004.10934.pdf [Accessed: 5 Sep 2023].

Boehm, B. 2007. A survey of agile development methodologies. Laurie Williams, 45, p.119.

Brunelli, R. and Poggiot, T. 1997. Template matching: Matched spatial filters and beyond. Pattern recognition, 30(5), pp.751-768.

Canny, J. 1986. A computational approach to edge detection. IEEE Transactions on pattern analysis and machine intelligence, (6), pp.679-698.

Chen, L., Li, Q., Li, M. and Mao, Q. 2011, June. Traffic sign detection and recognition for intelligent vehicle. In 2011 IEEE Intelligent Vehicles Symposium (IV) (pp. 908-913). IEEE.

Chrome developers. 2023. Chrome Extensions. Available at: https://developer.chrome.com/docs/extensions/ [Accessed: 5 Sep 2023]

Corona, E. and Pani, F.E. 2013. A review of lean-kanban approaches in the software development. WSEAS transactions on information science and applications, 10(1), pp.1-13.

Cortes, C. and Vapnik, V. 1995. Support-vector networks. Machine learning, 20, pp.273-297.

De La Escalera, A., Moreno, L.E., Salichs, M.A. and Armingol, J.M. 1997. Road traffic sign detection and classification. IEEE transactions on industrial electronics, 44(6), pp.848-859.

Deval Shah. 2023. Mean Average Precision(mAP) Explained: Everything You Need to Know. Available at: https://www.v7labs.com/blog/mean-average-precision#h1. [Accessed: 5 Sep 2023]

Du, J. 2018. Understanding of object detection based on CNN family and YOLO. In Journal of Physics: Conference Series (Vol. 1004, p. 012029). IOP Publishing.

DVSA. 2022. Top 10 reasons for failing the driving test in Great Britain. Available at: https://www.gov.uk/government/publications/top-10-reasons-for-failing-the-driving-test/top-10-reasons-for-failing-the-driving-test-in-great-britain#contents [Accessed: 5 Sep 2023]

Dwyer, B., Nelson, J. 2022. Solawetz, J., et. al. Roboflow (Version 1.0) [Software]. Available at: https://roboflow.com. computer vision. [Accessed: 5 Sep 2023]

Fradkov, A.L. 2020. Early history of machine learning. IFAC-PapersOnLine, 53(2), pp.1385-1390.

Ge, J. 2023. Traffic Sign Recognition Dataset and Data Augmentation. arXiv preprint arXiv:2303.18037.

Ghimire, D. 2020. Comparative study on Python web frameworks: Flask and Django.

Goodfellow, I., Bengio, Y. and Courville, A. 2016. Deep learning. MIT press.

Google for Developers. 2023. Interpreting Loss Curves. Available at: https://developers.google.com/machine-learning/testing-debugging/metrics/interpretic. [Accessed: 5 Sep 2023]

GOV.UK. 2023. The Highway Code. Available at: https://www.gov.uk/guidance/the-highway-code/traffic-signs. [Accessed: 5 Sep 2023]

He, K., Zhang, X., Ren, S. and Sun, J. 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

Jiang, P., Ergu, D., Liu, F., Cai, Y. and Ma, B. 2022. A Review of Yolo algorithm developments. Procedia Computer Science, 199, pp.1066-1073.

Jocher, G. 2020. Ultralytics YOLOv5, version 7.0. DOI: 10.5281/zenodo.3908559. Available at: https://github.com/ultralytics/yolov5 [Accessed: 5 Sep 2023].

Jocher, G., Chaurasia, A., Stoken, A., Borovec, J., NanoCode012, Kwon, Y., Michael, K., TaoXie, Fang, J., imyhxy, Lorna, Zeng Yifu, Wong, C., V, A., Montes, D., Wang, Z., Fati, C., Nadar, J., Laughing, UnglvKitDe, Sonck, V., tkianai, yxNONG, Skalski, P., Hogan, A., Nair, D., Strobel, M. & Jain, M. 2022. ultralytics/Yolo v5: v7.0 - Yolo v5 SOTA Realtime Instance Segmentation. Zenodo. v7.0. https://doi.org/10.5281/zenodo.7347926

Kaur, A. 2018. App Review: Trello. Journal of Hospital Librarianship, 18(1), pp.95-101.

Ketkar, N., Moolayil, J., Ketkar, N. and Moolayil, J. 2021. Introduction to pytorch. Deep Learning with Python: Learn Best Practices of Deep Learning Models with PyTorch, pp.27-91.

Krizhevsky, A., Sutskever, I. and Hinton, G.E. 2012. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25.

Lan Cao, Kannan Mohan, Peng Xu & Balasubramaniam Ramesh. 2009. A framework for adapting agile development methodologies, European Journal of Information Systems, 18:4, 332-343, DOI: 10.1057/ejis.2009.26

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

Li, C., Li, L., Geng, Y., Jiang, H., Cheng, M., Zhang, B., Ke, Z., Xu, X. and Chu, X. 2023. YOLOv6 v3.0: A Full-Scale Reloading. Available at: https://arxiv.org/abs/2301.05586 [Accessed: 5 Sep 2023].

Li, G., Song, Z. & Fu, Q. 2018. A New Method of Image Detection for Small Datasets under the Framework of YOLO Network. In: 2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chongqing, China, pp. 1031-1035. https://doi.org/10.1109/IAEAC.2018.8577214.

Pimentel, V. and Nickerson, B.G. 2012. Communicating and displaying real-time data with websocket. IEEE Internet Computing, 16(4), pp.45-53.

Python document. 2023. tkinter — Python interface to Tcl/Tk. Available at: https://docs.python.org/3/library/tkinter.html. [Accessed: 5 Sep 2023]

Radford, A., Metz, L. and Chintala, S. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434.

Rao, M.S., Sandhya, P., Sambana, B. and Mishra, P. 2023, June. Application for Mood Detection of Students Using TensorFlow and Electron JS. In Machine Learning and Big Data Analytics: 2nd International Conference on Machine Learning and Big Data Analytics-ICMLBDA, IIT Patna, India, March 2022 (Vol. 401, p. 235). Springer Nature.

Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. 2016. You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

Redmon, J. and Farhadi, A. 2018. YOLOv3: An Incremental Improvement. arXiv preprint arXiv:1804.02767.

Saadna, Y. and Behloul, A. 2017. An overview of traffic sign detection and classification methods. International journal of multimedia information retrieval, 6, pp.193-210.

Simonyan, K. and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Singh, M., Verma, A., Parasher, A., Chauhan, N. and Budhiraja, G. 2019. Implementation of Database Using Python Flask Framework. International Journal of Engineering and Computer Science, 8(12), pp.24890-24893.

Sportelli, M., Apolo-Apolo, O.E., Fontanelli, M., Frasconi, C., Raffaelli, M., Peruzzi, A. and Perez-Ruiz, M. 2023. Evaluation of YOLO Object Detectors for Weed Detection in Different Turfgrass Scenarios. Applied Sciences, 13(14), p.8502.

Stallkamp, J., Schlipsing, M., Salmen, J., & Igel, C. 2012. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. Neural Networks, 32, 323-332. ISSN 0893-6080. https://doi.org/10.1016/j.neunet.2012.02.016.

Tang, Q., Kurnianggoro, L. and Jo, K.H. 2018. Traffic sign classification with dataset augmentation and convolutional neural network. In Ninth International Conference on Graphic and Image Processing (ICGIP 2017) (Vol. 10615, pp. 828-832). SPIE.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I. 2017. Attention is all you need. Advances in neural information processing systems, 30.

Wali, S.B., Hannan, M.A., Hussain, A. and Samad, S.A. 2015. Comparative survey on traffic sign detection and recognition: a review. Przeglad Elektrotechniczny, 1(12), pp.40-44.

Wang, C-Y., Bochkovskiy, A. and Liao, H-Y. M. 2022. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. Available at: https://arxiv.org/pdf/2207.02696.pdf [Accessed: 5 Sep 2023].

Williams, L. 2010. Agile software development methodologies and practices. In Advances in computers (Vol. 80, pp. 1-44). Elsevier.

Wu, Y., Li, Z., Chen, Y., Nai, K. and Yuan, J. 2020. Real-time traffic sign detection and classification towards real traffic scene. Multimedia Tools and Applications, 79, pp.18201-18219.

Yolo v5 Github repository. 2023. Ultralytics/Yolo v5 repository. Available at: https://github.com/ultralytics/Yolo v5. [Accessed: 5 Sep 2023]

Zayat, W. and Senvar, O. 2020. Framework study for agile software development via scrum and Kanban. International journal of innovation and technology management, 17(04), p.2030002.

Zhu, Z., Liang, D., Zhang, S., Huang, X., Li, B., & Hu, S. 2016. Traffic-Sign Detection and Classification in the Wild. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

12. Appendices

12.1 Author Running environment

Window 10 Home, 22H2

Anaconda 23.7.2

Python 3.11.3

JavaScript 18.6.0

Google Chrome 115.0.5790.171 (Official Build) (64-bit)

Chrome extension manifest version 3

Flask 2.3.2

PyTorch 2.0.1 + cpu

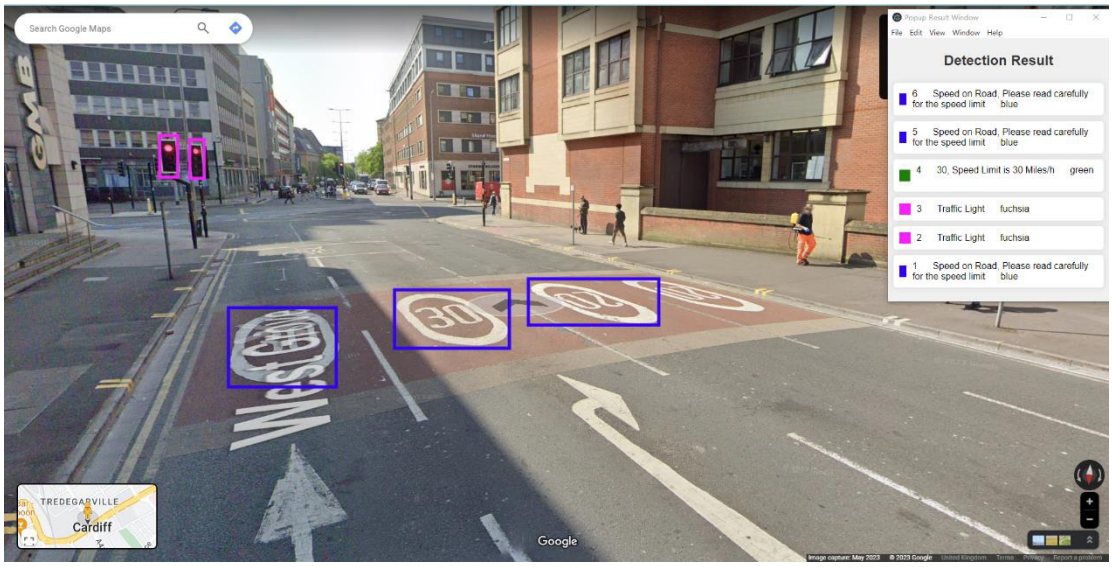
12.2 Test Case of Phone version and Electron JS version

Test Case Id: 2	**Test Purpose: Check if the phone browser can access the result**		
Environment: Window 10 Home (22H2), Anaconda 23.7.2, Python 3.11.3, JavaScript 18.6.0, Google Chrome 115.0.5790.171 (Official Build) (64-bit), Flask 2.3.2, PyTorch 2.0.1 + cpu, IOS 15.2, Safari 15			
Preconditions: The local server is started, the chrome extension is added into the chrome browser, all the requirement are installed. User is already in Google Street view. The mobile phone and the local server are within the same WLAN.			
Step No	Procedure	Expected Response	Pass/Fail
1.	Click the extension icon on the right top corner of the Chrome browser.	A page will show buttons and QR code. Buttons are for use to open result window. The names are 'Open Desktop Window' and 'Spare Web Window' QR code for mobile phone scanning.	(Pass)
2.	Using phone scan the QR code.	Once scanned, the phone will open the default web browser and go to the result page.	(Pass)
3.	Use mouse to navigate	If there are traffic sign in the Street	(Pass)

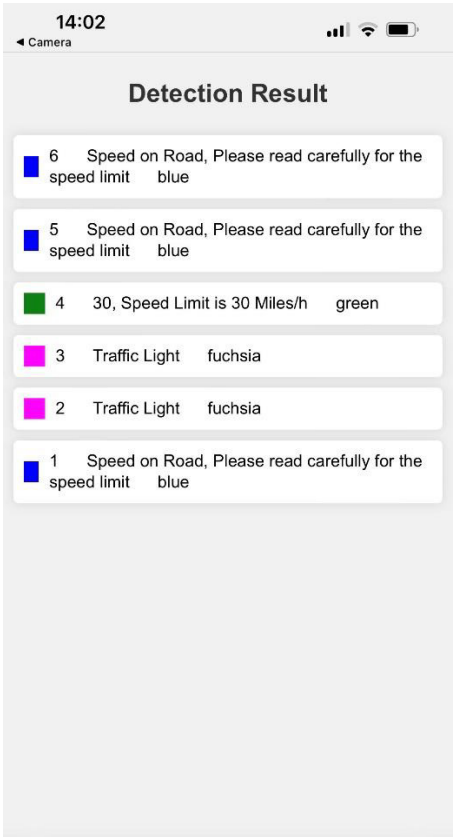
	within the Google street view.	view. Bounding box will be draw on the page. Result page will show in the result window. The colour of bounding box is correspond to the colour block in the result window. The information is synchronized	

Test Case Id: 3	**Test Purpose: Check if the Electron JS window can access the result**		
Environment: Window 10 Home (22H2), Anaconda 23.7.2, Python 3.11.3, JavaScript 18.6.0, Google Chrome 115.0.5790.171 (Official Build) (64-bit), Flask 2.3.2, PyTorch 2.0.1 + cpu, IOS 15.2, Safari 15, Electron.js 25.3.2			
Preconditions: The local server is started, the chrome extension is added into the chrome browser, all the requirement are installed. User is already in Google Street view. The mobile phone and the local server are within the same WLAN.			
Step No	Procedure	Expected Response	Pass/Fail
1.	Click the extension icon on the right top corner of the Chrome browser.	A page will show buttons and QR code. Buttons are for use to open result window. The names are 'Open Desktop Window' and 'Spare Web Window' QR code for mobile phone scanning.	(Pass)
2.	Click 'Spare Web Window'	Once clicked, the Electron JS result window will be shown after few seconds waiting.	(Pass)
3.	Use mouse to navigate within the Google street view.	If there are traffic sign in the Street view. Bounding box will be draw on the page. Result will show in the result window. The colour of bounding box corresponds to the colour block in the result window. The information is synchronized	(Pass)

12.3 Effect Image (Spare Electron.js version)



12.4 Effect Image (Phone access version)



12.5 How can the user install and run the extension

1. Install Anaconda first, all of the code will run within conda environment (The code also can run in non-conda environment, install all requirement will be fine), then install requirements.txt.

2. Go to Chrome browser, click the setting list icon on the right top corner => Extensions => Manage Extensions => turn on Developer mode => Load unpacked, then choose the chrome extension folder

3. Go to ‘Yolo_part’ folder, run app.py (make sure you are in the conda environment and install all library).

4. Go to electron folder, open the Anaconda Prompt as Administrator. Input 'npm install module_name' in command line to install library of electron.js (Please make sure the JavaScript and npm installed).

5. Go to Google Street View, start navigating.

*If the extension do not draw the bounding box, please try to refresh the page.

*If using smart phone to access, please make sure it is within the same WLAN with local server.