



BE (IoT) Degree Program

Stage 4 Project Group Thesis Receipt

Embedded equipment system for monitoring and broadcasting non-standard driving
Project Title: movements caused by distraction and fatigue detection

Supervisor: Dr. Alzubair Hassan

Student Name(s)	UCD Student Number	BJUT Student Number
Benteng Ma	18206096	18371202
Zhejia Wang	18206070	18371106
Xiaoxiao Yu	18206362	18371205
Yushi Wang	18206384	18371215

Plagiarism: the unacknowledged inclusion of another person's writings or ideas or formally presented work (including essays, examinations, projects, laboratory presentations). The penalties associated with plagiarism designed to impose sanctions seriousness of University's commitment to academic integrity. Ensure that you have read the University's **Briefing for Students on Academic Integrity and Plagiarism** and the UCD **Statement, Plagiarism Policy and Procedures**, (<http://www.ucd.ie/registrar/>)

Declaration of Authorship

I/we declare that all of the following are true:

- 1) I/we fully understand the definition of plagiarism.
- 2) I/we have not plagiarised any part of this project and it is my original work.
- 3) All material in this report is my/our own work except where there is clear acknowledgement and appropriate reference to the work of others.

Signed..... Benteng Ma Date 2022.5.20

Signed..... Zhejia Wang Date 2022.5.20

Signed..... Xiaoxiao Yu Date 2022.5.20

Signed..... Yushi Wang Date 2022.5.20

Signed..... Date

Signed..... Date

Signed..... Date

Office Use Only

Date and Time Received:

N/A

Received by: Brighspace Submission

Report Tracking

Abstract

Traffic accidents are increasingly occurring, with a high percentage caused by drivers' mistakes. Fatigue and distraction of drivers are the leading causes of road accidents and can greatly harm drivers and passengers. Therefore, early warning for these two dangerous driving behaviours can effectively reduce the probability of accidents. This report proposes a system to monitor and detect non-standard driving behaviours. We mainly focused on detecting fatigue driving and manual distraction (e.g., eating, drinking, smoking and talking on the phone) with computer vision technology, and used active learning to increase the accuracy. After non-standard behaviours are detected, an warning signals will be send to vehicle drivers and nearby vehicles. We implement our system using two Raspberry Pi devices to detect the Non-standard driving behaviour. The external module of Raspberry Pi uses 4G, GPS, and MQTT technologies to upload the current location of dangerous vehicles and send warning signals to other vehicles.

Contents

1	Project Code	4
2	Introduction	4
3	Literature Review	5
4	Design & Structure of the System	6
5	Description of Work Division	6
6	Dataset, Communication Hardware, Cloud Implementation - Zhejia Wang	7
6.1	Declration of Authorship	7
6.2	Description of Work Completed	7
6.3	Dataset	7
6.4	Communication Hardware	9
6.5	Implementation and Experimentation on Cloud	13
7	Preprocessing, Model Training, Software and Hardware - Benteng Ma	18
7.1	Declaration of Authorship	18
7.2	General Description of Work completed	18
7.3	Image Preprocessing	18
7.4	Detection and Classification Models	19
7.5	Implementing the Model, Transfer Learning and Training the Models	21
7.6	Model Packaging and Software Implementation	26
7.7	Initialize Raspberry Pis	28
7.8	Code Migration	29
8	Data Augmentation, Active Learning, Model Selection and Evaluation - Xiaoxiao Yu	29
8.1	Declaration of Authorship	29
8.2	Description of all the work completed	30
8.3	Image Data Augmentation	30
8.4	Active Learning	31
8.5	Classification Model Evaluation	36
9	Facial Monitoring, User Interface and Cloud Design - Yushi Wang	40
9.1	Declaration of Authorship	40
9.2	Description of all the work completed	40
9.3	Fatigue model	40
9.4	Extraction of Facial Feature Points	41
9.5	Real-Time Eye Blink and Mouth Yawn Detection	42
9.6	Discussion - Parameter selection	43
9.7	Head Posture	45
9.8	Emotion Monitoring	46
9.9	User Interface	46
9.10	Cloud Computing System Design	47
10	Overall Results, Discussions	48

1 Project Code

Our code is uploaded to GitHub: https://github.com/MBTMBTMBT/degree_project_code;
We will also upload our code to BrightSpace.

2 Introduction

In recent years, traffic accidents caused by dangerous driving behaviours are increasing. Traffic accidents can cause physical and mental injuries to passengers and drivers. Distraction while driving is one of the leading causes of traffic accidents, such as talking on the phone, eating, drinking, etc. In the United States, cell phone use while driving causes thousands of additional fatalities every year [34]. Distraction of driver will seriously affect the driver's concentration, reaction speed and judgment, and is associated with poorer driving performance. In addition to those activities, fatigue driving is also one of the main causes of traffic accidents. According to US survey, 20% of fatal crashes involved a drowsy driver, in the EU, 20% of commercial transport crashes are attributed to fatigue [47]. One of the most dangerous aspects of fatigue driving is that the driver is not aware of his or her drowsiness. Driving for long periods of time requires the driver to concentrate continuously. There will inevitably be a loss of alertness, leading to reduced performance and furthermore to higher risk of accidents [2].

To reduce the probability of accidents caused by drivers' mistakes, in this report, a non-standard driving behaviour monitoring system is designed to monitor the actions and status of the driver and provide timely warnings. There are two categories driving behaviours considered as non-standard driving behaviours: behaviours that cause driver distraction and fatigue driving behaviours. Especially, the focus of the distraction detection of the system is manual distraction, and it is based on computer vision method. Specifically distraction monitoring in the system includes the actions of eating, drinking, smoking, talking on the phone. In order to detect these actions, models are trained for body outlier detection and behaviour classification, which are able to classify driver's behaviour based on detected outliers include driver's head and hands. Active learning is added to help improve the accuracy of model training, and reduce the time required for manual labeling. Fatigue detection is part of driver facial detection which also includes emotion detection and head existing detection, based on the extraction of facial feature points. When the non-standard behaviour of a driver is detected, the warning system is activated. The hardware device used for early warning alerts the driver and uploads the current vehicle location to the cloud, which can do the vehicle area division and send the early warning information to vehicles within a certain range according to the current vehicle location. This system is especially suitable for those people who drive with high intensity or for long periods of time, such as truck and bus drivers, reducing the probability of accidents caused by their non-standard driving.

This report is organized as follows: Section 3 describes the related work. Section 4 describes the design and structure of the system. Section 5 describes the work division. Section 6 describes the design and preparation of dataset, hardware selection and configurations. Then, how the transmitting signals between vehicles and implementation on cloud are introduced. Section 7 presents about how the image data is preprocessed before training; describes how the driver's body outlier detection models and behaviour classifier models are implemented and trained; and which models are selected to be used for software implementation; then it discussed how our

software is implemented; how the hardware is implemented and how the software is migrated to the hardware device. Section 8 describes the process of image data enhancement. Then the design and operation of two active learning algorithms are described. Finally, mathematical method is applied to select the classification model and evaluate the result. Section 9 contains fatigue detection based on facial feature point extraction, emotion detection and head exiting detection. Also, it includes the layout of the user interface and the design of the cloud computing system.

3 Literature Review

At present, there are mainly three approaches to do driving behaviours recognition: physiological information, vehicle status, and computer vision [47]. The first method is biological indicators such as brain waves, heart rate and pulse rate, respiration, etc [51]. It requires variable sensors, and placing on driver's body. Although this method has the highest detection accuracy, but the installation of the sensor will be uncomfortable for drivers and even affect driving [43, 48, 53]. The second method is using information about vehicle status, such as the turning angle, steering wheel movements, car speed, etc. It is easy to collect the data, but limited by the road conditions, driver experience, and so on [51]. In addition, this method will take a huge amount of time to analyze the driving behaviors that cause driver fatigue and distraction [1, 17]. The third method is based on facial features analysis and analysis of body parts, such as status of mouth, blinking frequency, hands, etc. The three approaches can be divided into visual features based and non-visual features based, techniques using visual features utilize computer vision methods to detect drowsiness, and distraction detection can be based on computer vision method as well. There are three main distraction detection approaches: manual distraction, visual distraction, and cognitive distraction [21]. Drivers can easily shift their attention from their driving tasks to non-driving tasks through their eyes (visual distraction), their hands (manual distraction), and/or their minds (cognitive distraction) [37]. A driver is visually distracted when he/she is engaged in an activity that requires he/she to look away from the forward roadway [20].

According to the direction of computer vision, we looked for deep learning models that is able to detect objects. FasterRCNN [41], YOLO [39], and RetinaNet [33] are three very commonly used object detection models used currently. FasterRCNN and RetinaNet are two-stage networks, there is first a convolutional feature extractor extracts the image features, after the extraction there is a region purpose network to purpose possible bounding boxes, then there is a predictor to predict the bounding boxes of objects and their class to output. YOLO is a one-stage network, that all these mentioned task is done simultaneously.

We also reviewed five famous and commonly used image classification neural networks, AlexNet [25], VGG-16 [49], GoogLeNet [50], ResNet-34 [13] and MobileNet-V3 [16]. During our implementation, we implemented, trained and tested these networks.

Image data need first be normalized before feeding into a model, we used the 0-1 normalization introduced in [4], chapter 2. Most computer vision tasks are taken over by neural networks in recent years. When training a model, mini-batch training methods [31] is a common method to training deep neural network with GPU acceleration. When training the model, we also used a method named transfer learning [54] to speed up our training and to have more sophisticated feature extractors.

The dataset format of object detection is [10], consists of annotations and pictures.

Few people have combined active learning so closely with manual distraction in distraction detection, so this is a first attempt in our project. We applied active learning to the object detection and classification models in manual distraction to reduce manual annotation time and increase detection accuracy. The implementation of active learning and the results of its use are in section 8.4.

4 Design & Structure of the System

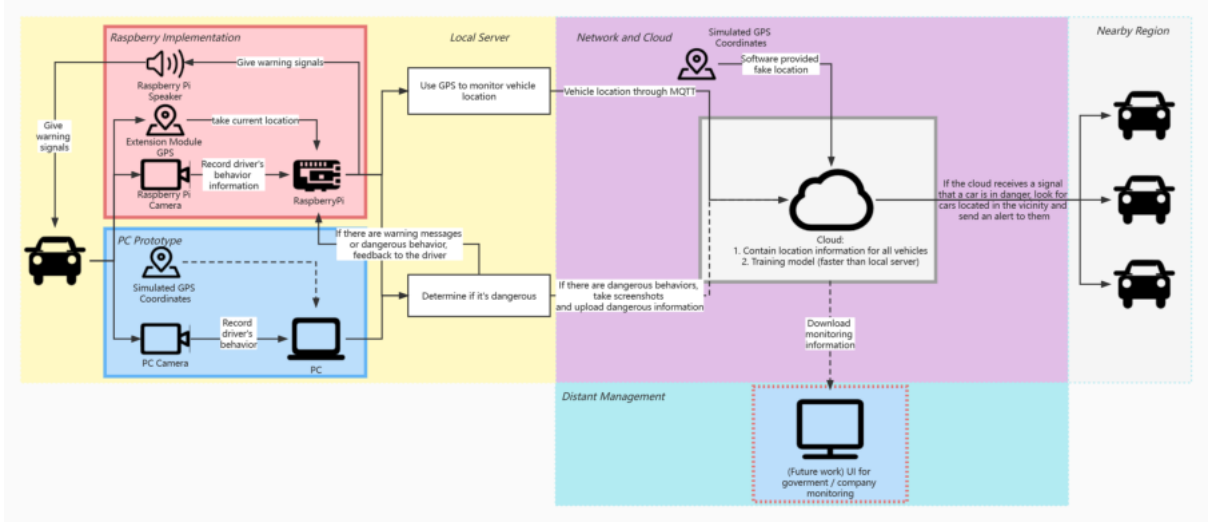


Figure 1: The System Design

5 Description of Work Division

- **Zhejia Wang** worked on designing, preparing and managing the dataset. Select the communication method, and managed to install 4G communication, GPS and MQTT module onto the Raspberry Pi, let the board connected to the Internet. **Zhejia** also choose Ali cloud among multiple clouds to implement the data upload, storage, processing and flow required for communication and warning between vehicles, the experimentation of vehicle area division is finished on cloud as well.
- **Benteng Ma** worked on preprocessing the image data from our purposed datasets; implementing, training and selecting the driver's body outlier model and dangerous behaviour classification; and packaging the selected models, as well as Yushi's code, to implement a whole software. **Benteng** also worked on initializing the Raspberry Pis and build the devices based on them; and migrated the software onto these devices.
- **Xiaoxiao Yu** worked on image data enhancement of our purposed datasets. Design two active learning algorithms for object detection model and classification model to reduce manual labeling time. **Xiaoxiao** also responsible for selecting classification models with mathematical methods and evaluating the final effect of the model.

- **Yushi Wang** worked on facial monitoring, including fatigue detection, emotion detection and face existing detection and selects threshold parameters to adjust the detection system, for which she does experiments and collects data from real people. **Yushi** also work provided the GUI of the whole software and the cloud computing system design.

6 Dataset, Communication Hardware, Cloud Implementation - Zhejia Wang

6.1 Declration of Authorship

I hereby declare that the following statements are true and valid:

- 1) I fully understand the definition of plagiarism and its consequences.
- 2) I promise that no part of the project is suspected of plagiarism and that it is all my original work.
- 3) All material for this project, including code and articles, does not contain anything that has been published or written by anyone unless I explicitly state that we are referring to someone else's project or work.
- 4) I own the full copyright of this project including the code and article and do not use without permission.
- 5) The thesis presented is the result of my independent research under the guidance of our supervisor.
- 6) The paper is written by me and I am responsible for what I wrote.

6.2 Description of Work Completed

I mainly prepared and managed the dataset. Then, I selected different communication protocols, and by analyzing the advantages and disadvantages and discussing the feasibility in the context of our project, I finally selected the right external device to implement the communication between vehicles. The external device is compatible with Raspberry Pi (see Benteng's part [7.7]), and I use Raspberry Pi to configure the device. Next, I chose Ali cloud among different clouds to realize the uploading, storage, processing, and flow of data about vehicles through the collaboration of multiple cloud products. I also used MapReduce to reduce the time complexity of the vehicle area division based on the model (see Yushi's part [9.10]). Then, I analyze different clusters and chose one of them based on their performance.

6.3 Dataset

Since most of the datasets available on the web are side-on shots of people driving, what we needed was a frontal overhead angle shot. Also, due to the occurrence of Covid-19, many drivers start to wear masks while driving, and companies of cabs and buses might request their drivers to keep their masks on all the time. However, there are not many datasets on the web for driving vehicles with masks. Therefore we created our related dataset with **over 10,000** photos from ourselves, friends, family members, and open-source datasets on the web. For preparing our dataset, as Figure 2 shows, we use Pascal VOC to label the images. The detection regions we focus on are labeled by the bounding boxes in the figure, which can facilitate the model to obtain the corresponding data and train later.

To simulate the real driving environment as much as possible, many of the photos were taken inside the vehicle. Also, given the need to detect non-standard driving behavior, the people tested will have the dangerous actions described above. Then, we try to increase the diversity

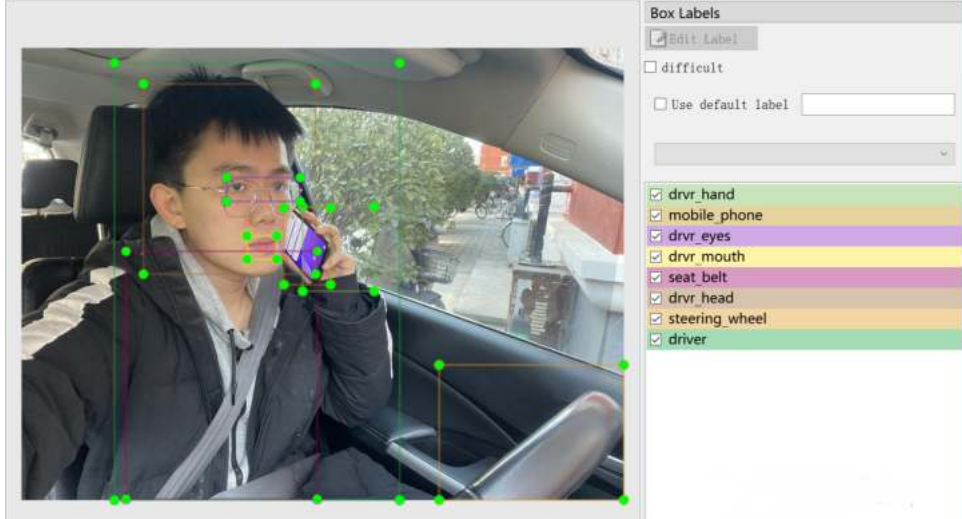


Figure 2: Pascal VOC to Label Images

of the dataset, such as age, gender, etc. Considering the impact that different nationalities of testing people would have on our model training, our dataset includes testing people from different countries, which are from the open-source dataset. However, since we also asked many family members and friends to take pictures, the percentage of other people from other countries is not high.

There are different datasets preparing for driver motion classification. Including four 'head and hand crop' datasets: 'sunglasses', 'masks', 'head motion', and 'hand motion' classification. Both head and hand cropped images are first cropped by our detection model, and then manually classified. Labeling the dataset truly plays a key role in supervised learning. The datasets are introduced below:

- **Sunglasses and Masks:** The reason for the detection of sunglasses and masks is to prevent model misclassification. For instance, wearing sunglasses is misjudged as sleeping, and wearing a mask is misjudged as eating, so we need to train our model to detect these two things. These two datasets are for binary classification, that is to distinguish whether the driver is wearing sunglasses or a mask, or not. The cropped images are classified into three folders, 'wearing', 'not_wearing', and 'others'. 'others' is for images mistakenly cropped, or for head images that are unable to be classified by people.
- **Head Motion:** Using the small image of the driver's head, the model can classify if the driver is focusing on the front, eating or drinking, talking on the phone, etc. The head motion class includes 'eating or drinking', 'looking away', 'no motion', 'tired', 'talking on the phone', 'touch face or head', 'smoking', 'others'. Although some categories are not classified as non-standard driving, we still grouped them into specific categories. The reason is that if these behaviors are uniformly specified into one category, and we encounter some hard-to-discriminate images in the future, they will be automatically classified into that category, which affects the training effect of the model. Same as sunglasses and masks, 'others' contain images mistakenly taken, or too difficult to classify.
- **Hand Motion:** Using the cropped image of hands, another model sees if the driver is holding something, such as a mobile phone, or a bottle. The hand motion class includes 'food, drink and cigarette', 'mobile phone', 'steering wheel', and 'others'. The reason for

putting the category of cigarettes and eating and drinking together is that there are so few photos of cigarettes. Thus, putting it into a single category is not effective for training the model. 'Others' contains the images mistakenly taken, or too difficult to classify.

6.4 Communication Hardware

The work I did in this part was to find the right device to implement the communication, and to implement the communication method suitable for our project application scenario. The system designed needs to remind the driver when the person has non-standard driving behaviour, and remind other running vehicles in a certain range as well through our installed equipment.

However, the device I use for the project needs to meet several requirements. Firstly, it needs to be compatible with Raspberry Pi and can be controlled by using Raspberry Pi. Then, it should be robust, small to carry, and energy-saving. Next, it should implement the customized function, which means adjusting parameters based on the needs. Finally, a stable and efficient means of communication between hardware and hardware is needed, which can help the devices send and receive information. Therefore, it is quite difficult to find an appropriate device to satisfy all the requirements, and it is a process of exploration. In this part, the exploration of finding the device, and the choice of communication between devices will be introduced.

6.4.1 Through 4G extension module

The first communication method I started with was ZigBee. However, after finding the signal transmission range of ZigBee is not enough for our project, and this communication method is not perfect or proper for high-speed applications, [29]. Also, the lack of a clearly explained instruction book to debug the equipment effectively, the parameters of those devices are not consistent, and the configurations of it are much more complex than the 4G extension module. Therefore, I chose to find another device and the 4G extension module was more suitable and selected as the communication hardware. In this part, I explored the benefits of using the 4G extension module, and the function it can provide, such as the communication protocols supported, connecting to the cloud, etc. Then, I found this module can meet the requirements that mentioned above. I also implemented how to connect the Raspberry Pi to the extension module, and how to use the relevant commands to accomplish the different functions so that I can reach the communication purpose.

The module (EC200U-CN based), has great compatibility with the Raspberry Pi, and it provides a stable means of communication. Also, the EC200U-CN module has a compact package, and it can meet almost all M2M application requirements, such as automation, intelligent metering, tracking system, mobile computing devices, etc. Figure 3 shows providing many functions and interfaces, and they are important for transmitting data or initializing the function, for instance, the GNSS function and (U)SIM are quite important for our project design.

Considering the functions that the extension module can provide, I decided to combine those functions to transmit the signal. Firstly, I need to get the current locations of the vehicles. When detecting non-standard driving behaviours, the location of the vehicle will be uploaded to the IoT platform, the platform can help that device send a warning message to the running vehicles within a certain range, so those drivers can receive the signal and be alerted.

This is a coordination of several functions provided by the module, such as GPS, 4G, and MQTT, each part is responsible for its own functional partition and can be combined together


```

ec200u=EC200U('/dev/ttyS0',115200)# open relevant port
ec200u.ATSingle('ATE0\r\n','OK')# judge whether module exists
ec200u.ATSingle('AT+QGPS=1\r\n','OK')# open GNSS, set gps power on
time.sleep(0.5)#the unit is second
while True:
    ec200u.ATSingle('AT+QGPSGNMEA="rmc"\r\n','OK')# get gps data
    print(ec200u.rxData)
    if ec200u.rxData.find('$GNRMC')!=-1:#
        line = str(ec200u.rxData).split(',') # let line separated by ","
        if line[4]=='N':# location successfully
            latitude = float(line[3][:2]) + float(line[3][2:])/60
            # Read the fifth string information, from 0 to 3 to the longitude,
            # followed by a string of division 60 to convert the minutes to degrees
            longitude = float(line[5][:3]) + float(line[5][3:])/60
            # the same as above
            print("longitude:",'{:.6f}'.format(longitude))
            print("latitude:",'{:.6f}'.format(latitude))
            time.sleep(0.5)#the unit is second

```

Figure 5: Code implementation about GPS

(2) 4G to Provide Internet Access

The module used 4G to connect to the Internet, and I used 4G to provide the foundation for using MQTT to upload data to the IoT platform (Ali cloud) and download the feedback data from the platform. Inserting the SIM card can provide the module with a 4G network. Also, the module can give Raspberry Pi Internet access by connecting to a USB interface, but the USB interface should be 2.0 version. Like the GPIO port, the interface can be used for AT command transmission, data transmission, and software debugging. USB virtual serial port driver supports USB drivers for Linux operating systems. Installing a USB driver allows the Raspberry Pi to have Internet access, and the Linux operating system is suitable for the Raspberry Pi.

I used 4G to satisfy and improve the application scenario for other several reasons:

- Using 4G to transmit the GPS information of the device is a suitable choice for our project, it has very low latencies, a wider channel and carrier aggregation up to 100Mhz [38]. Therefore, the speed of uploading and transmitting data is quite fast and can reduce the latency, which is essential for getting to the current location.
- Considering the wide range of mobility and high flexibility of the vehicles, I need to make the communication as wide as possible and reduce the impact of rapid changes in location. Correspondingly, 4G is currently an active and mature mobile network technology with a high transmission rate, and low expense, and it can almost full coverage of signals in urban areas [55]. These characteristics are suitable for our project design.

(3) MQTT to Connect to Cloud

The module supports many network protocols, such as TCP, UDP, MQTT, SSL, etc, and MQTT can be used to help the extension module access to IoT platform on Ali cloud. I explored the advantages of using MQTT and the implementation of this protocol on the module. Also, MQTT is very important for the device to upload and download data to the platform, and send messages to another device based on the cloud. Using MQTT was a good fit for our project, and helped a lot with the implementation, and the specific reasons are as follows:

- MQTT connects the networks and devices with middleware and applications. The connection method and routing mechanism of this protocol are very suitable for the project application scenario. The communication patterns, such as machine-to-server can help the device connect to the cloud, and machine-to-machine can enable communications

between devices. Also, the routing mechanism has different patterns (one-to-one, one-to-many, many-to-many) [56], these patterns are all useful, for instance, the one-to-many message distribution mechanism, if there is a dangerous situation occurring in one vehicle, we can easily achieve the purpose of sending messages from one device to a certain range of devices.

- The architecture of this protocol is topic-based with publish-subscribe, which is easy to implement and helps the architecture of communication not be very complex. Figure 6 shows this is the basic principle of how I use MQTT to transfer information. The device uploads the data to the platform on the cloud, and the cloud can forward data to other parts. Communication is achieved by using a topic, and the topic is a transport intermediary between message publishers (Pub) and subscribers (Sub). The broker is like a server to store and forward the data.
- The MQTT application scenario is also well suited for our project, it is designed for lightweight M2M communications in the constrained network, and it can provide real-time and reliable messaging services for remote device connections with minimal code and limited network bandwidth. Correspondingly, in our project, the amount of data transferred between devices is not large, and MQTT can continue to work properly in such situations when the signal is poor or compromised, despite the wide coverage area of 4G. Specifically, in mildly unstable networks, MQTT can satisfy the system response time requirements, also it can successfully deliver all packets in the network constrained environments [45]. Therefore, this protocol is quite suitable for the device's usage scenario.

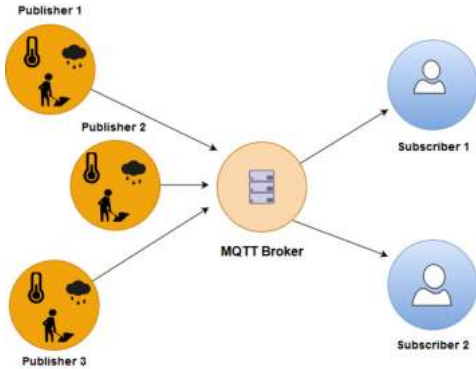


Figure 6: MQTT publisher and subscriber [35]

For the pseudocode implementation about using MQTT to connect to Ali cloud, Figure 8 shows these are the main AT commands used to build the connection of the module to the IoT platform. I firstly need to check whether a SIM card exists, because the card can provide a 4G network and enable the module to upload data. Then, check the network registration status and choose which platform to connect to. The parameters configuration, such as product key, and device secret are used for the cloud and will be introduced in the cloud part.

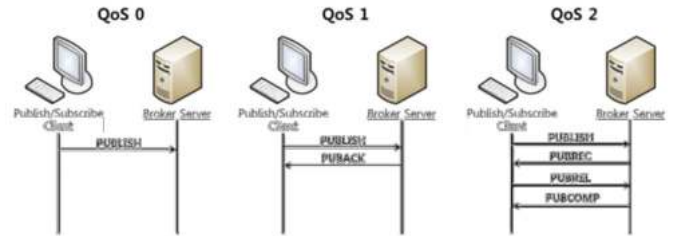


Figure 7: Packet Transmission about QoS Level [28]

Also, the parameter about QoS level is important for our implementation, and I can set it up for different situations because the different levels will influence the performance of communication. There are three levels of MQTT message quality, as Figure 7 shows, the message transmission for different levels. QoS Level 0 sends the message only once based on the message

```

APN, ServerIP, Port, username, ProductKey, DeviceName, DeviceSecret, pubtopic// connection about the platform
'ATE0', 'OK'// judge whether module exists
'AT+CIMI', 'OK'// Check whether the SIM card exists
'AT+CGATT?', '+CGATT: 1'// Check whether the network successfully registered
'AT+QMTDISC=0', 'OK'// For the last disconnection, no need to determine whether the connection successfully closed
'AT+QMTCFG=aliauth', 'OK'// connect to IOT platform
'AT+QMTOPEN=0', '+QMTOPEN: 0,0'// Connect to the MQTT server, confirm the connection status

```

Figure 8: Main AT commands used to connect to Ali cloud

distribution flow, and it does not check whether the message has reached its destination. For uploading the current location, I mainly used QoS 0. If the sending of the message fails this time, the next upload of the message will take place after a few seconds, which will not have a big influence on the result and save more energy than using the other two levels.

Therefore, the combination of 4G, GPS, and MQTT can successfully achieve the purpose of communication, i.e. GPS locates the current position, gets the latitude and longitude, and then connects to the IoT platform by way of MQTT to upload data. The IoT platform can forward the data to other devices based on the topic to achieve communication, or it can flow to other cloud products for massive data processing and send the data to the corresponding devices, which will be fully introduced in the cloud section. Although the GNSS function can work without 4G, 4G plays an important role in all the other processes.

6.5 Implementation and Experimentation on Cloud

The devices are needed to connect through the cloud to transfer, process, and store data. Also, I need to do the vehicle area division based on MapReduce as a way to improve the efficiency of early warning signal transmission. The reasons why I chose Ali cloud are considering that it can provide a variety of services, and it is more convenient to use Ali cloud in China compared to other clouds, which will not be limited by network, region, and other factors. Also, Ali Cloud has a high degree of flexibility in that the services it provides can correspond to our needs and change according to our detailed requirements. Moreover, the ease of data flow between different cloud products, and the design and advantages Yushi Wang 9.10 has introduced about the cloud are also the reasons that I chose Ali cloud and move the implementation to it.

In this section, I successfully connect two devices to the platform. Then, I chose different cloud products and built the environment on the cloud. Next, I moved the implementation of communication among vehicles to Ali cloud, which includes data uploading, data flow, data processing, etc. Also, I use MapReduce to do the comparison of experimentation of vehicle area division locally and on the cloud. The cloud products to do the data upload, storage, and processing are as follows:

(1) ECS server for Data Processing

I used ECS (Elastic Compute Service) server to process the data from the IoT platform and used Mapreduce to achieve the car area division on ECS as well. This server is an IaaS (Infrastructure as a Service) level cloud computing service. It supports scaling and releasing resources at any time and has good stability and reliability, and elastic scalability, which means that I can choose the suitable machine based on the need.

The machine I chose is based on many aspects, for instance, when the project runs locally, the CPU is sufficient, but memory has a heavy load, and we need to solve this problem when

choosing the server. Also, to improve the performance of the project, but at a reasonable cost. Therefore, the machine I used is ECS.c6e.2xlarge, which is 8 vCPU and 16 GiB. It relies on the third-generation Divine Dragon architecture, the ratio of the processor to memory is about 1:2. The configuration of processor is: Intel ® Xeon ® Platinum 8269 (Cascade) with 2.5ghz main frequency and 3.2ghz core frequency, with stable computing performance.

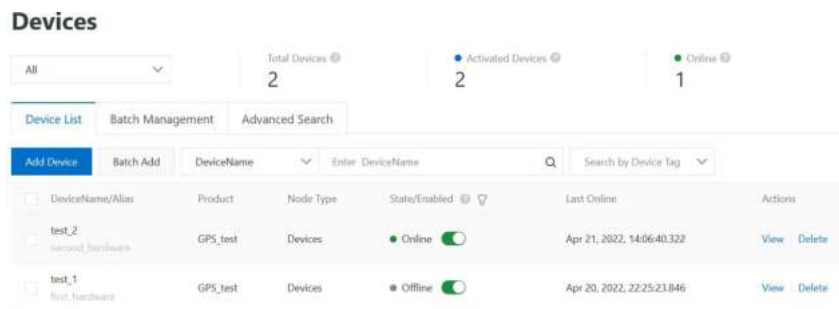
(2) RDS Database to Store Data

The RDS database I used enabled the local machine and ECS server to connect to it by entering the password. Also, the data uploaded to the IoT platform can be flowed to the database to store. RDS (Relational Database Service) supports MySQL, SQL Server, PostgreSQL, and MariaDB TX engines, and provides a complete solution for disaster recovery, backup, recovery, monitoring, migration, etc. Considering we do not need to store too much data, there is not much about performance requirements. Thus, the configurations of RDS are, the database type is MySQL 8.0, there is one CPU, and the memory is 1024M. The maximum number of supported connections is 2000, and the storage space I choose is 20G.

(3) IOT platform to connect devices

I used the IoT platform to enable devices to connect to it, and the platform on Ali cloud is an integrated platform that integrates many capabilities, such as device management, data security communication, message subscription, etc. The device secret and device name are different for each device, and the product name parameter is determined based on whether the two devices are classified under the same product. Then, the topic permission I set can decide whether to publish or subscribe to messages.

Figure 9 shows the status of two devices, after connecting to the platform, it will show online. In testing, I changed the device to automatically disable if no action is taken for a long time, so there is one device showing offline. Then, Figure 10 shows, that the uploaded data are shown in the platform. The data about current humidity and temperature is not what we need, the geographical location contains the longitude and latitude information, which can help us get the current location of the vehicle.



Devices						
All		Total Devices 2		Activated Devices 2		Online 1
Device List Batch Management Advanced Search						
Add Device Batch Add <input type="text" value="DeviceName"/> <input type="text" value="Enter DeviceName"/> <input type="text" value="Search by Device tag"/>						
<input type="checkbox"/> DeviceName/Alias	Product	Node Type	State/Enabled	Last Online	Actions	
<input type="checkbox"/> test_2 <small>second hardware</small>	GPS_test	Devices	Online <input checked="" type="checkbox"/>	Apr 21, 2022, 14:06:40.322	View Delete	
<input type="checkbox"/> test_1 <small>test hardware</small>	GPS_test	Devices	Offline <input type="checkbox"/>	Apr 20, 2022, 22:25:23.846	View Delete	

Figure 9: Connected Devices on Platform

On the downward side, the platform supports connecting massive devices and collecting device data to the cloud. For the upward side, it provides cloud API, and the server-side can send commands to the device side by calling the cloud API to realize remote control. Then, for communication between devices, I realized the purpose of forwarding data between them by setting parser and rule engine. If many vehicles join the communication network and connect to the platform in the future, and I want to do a large amount of data processing, the platform can forward the data to the ECS server for implementation and operation.

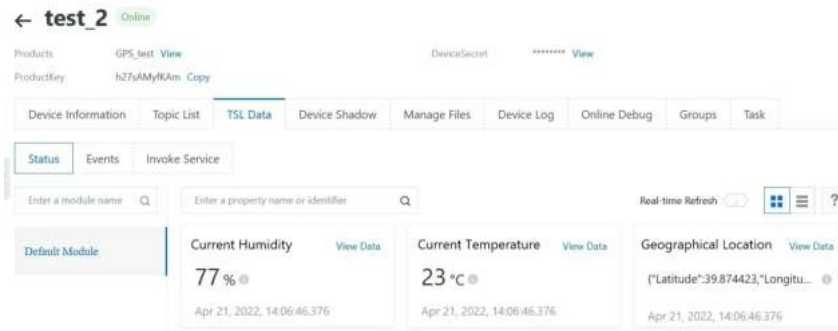


Figure 10: Uploaded Data from Test 2 Device

6.5.1 Data Flow Transformation

In this part, I implemented the way to let our two devices communicate successfully to send and receive data based on the mechanism for forwarding data to another topic. The data forwarding to another topic contains the data collected by the device. However, if I want to add additional information to the data collected by the device, or when tens of thousands of vehicles joined the communication network and generated data that needed to be processed or stored, I explored how I could operate by forwarding the data to other products. i.e. forwarded to ECS server for processing via MNS or RocketMQ, or forwarded to RDS database for storage. Then, I will introduce how these two plans work as follow:

(1) Data Forwarding to Another Topic

Devices pass information through topic, so Figure 11 shows if I want to achieve the purpose of communication between two devices, I can forward the data of this device to the topic of another device through the rule engine. Then, the data on one device can be transmitted to another device, but the data cannot be stored or processed.

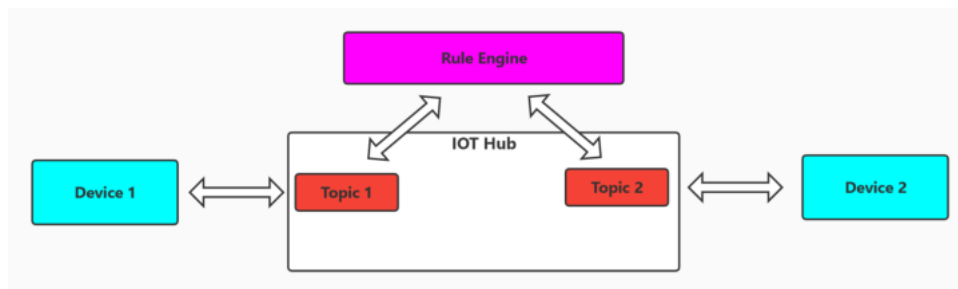


Figure 11: Data forwarded to another topic

(2) Data Forwarding to Other Cloud Products

As shown in the Figure 12, after the devices upload data to the platform, the platform can choose to forward it directly to the RDS database for storage, but this requires both the platform and the RDS database to be in the same area. The platform can also flow the data through the rule engine, and based on MNS or RocketMQ sending to the ECS server for data processing. The processed data then can be passed to the RDS database for storage.

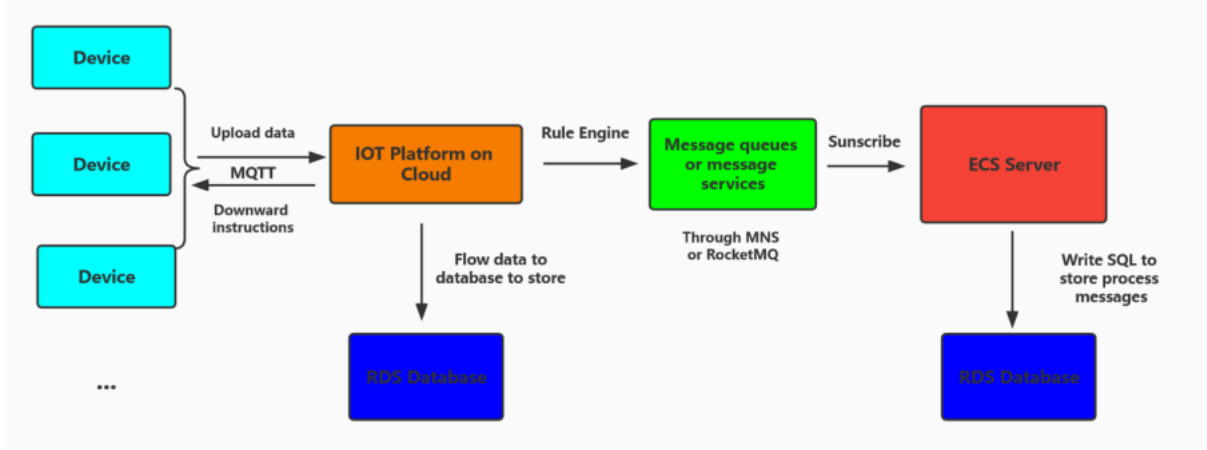


Figure 12: Data forwarded to other cloud products

6.5.2 MapReduce Experimentation

Since we only have two real devices, and more vehicles will be added to the communication network in the future. Therefore, I used the way of dividing the vehicles into different areas and tested the area division of 20 million and 80 million vehicles under different Hadoop clusters respectively. Also, I used the MapReduce method to reduce the time complexity of the operation. As the model shows, the input is the current vehicle ID and the latitude and longitude coordinates, and the output is the ID of each area, and the vehicles present in the area.

At the beginning of the experimentation, I divided the whole area into eight sections, each section overlaps with adjacent sections. When one vehicle has a dangerous situation, the warning signal can be transmitted to other vehicles within the area where that vehicle is located. Thus, the overlap between the sections ensures that if two vehicles are in different sections when one vehicle has a dangerous situation, the signal can be transmitted across the sections. Then, I randomly created 20 million and 80 million vehicles with vehicle ID and the latitude and longitude coordinates.

Next, I used the MapReduce method, which includes splitting, mapping, and reducing three steps. Splitting is dividing the data and sending it to different mappers. Mapping is distinguishing the area of a vehicle based on its coordinates. Reducing is joining the mapped key-value pairs into a set, and combining all vehicles in the same area together. I built a Hadoop environment to run on a local PC, but it is complicated to configure and build the environment, also the overall running time was too long. Thus, I used e-MapReduce on Ali cloud to build three different cluster types, one header with two, three, and four workers. The VM configurations of the header and worker are all the same as the ECS server.

After executing the relevant instructions, the same input data about vehicles can run in the three clusters. Figure 46 shows, that these are the specific results of the map and reduce processes for three different clusters. As the number of workers grows, the reduce process is significantly improved with the same number of the header, which also reduces the overall running time. As Table 1 shows, in the condition of a small amount of data, the time using three workers is less than two workers, but quite similar with four workers. However, in the condition of a larger amount of data, when the number of workers increases, the running time is gradually decreasing. Then, for the performance of the three clusters, as Table 2 (corresponds to each

row of table 1) shows, with the increase in the number of workers, the header CPU usage and worker memory will decrease. Also, in the same amount of data, worker CPUs will decrease with the increase in many workers.



Figure 13: One Header with Two Workers



Figure 14: One Header with Three Workers



Figure 15: One Header with Four Workers

Figure 16: Running Results for Three Clusters

Table 1: Running Time with Different Number of Vehicles and Configurations

Header	Worker	Vehicle number	Time	Vehicles per second
1	2	400M (20 million)	182s	109890
1	2	1.8G (80 million)	334s	239529
1	3	400M (20 million)	165s	121212
1	3	1.8G (80 million)	279s	286738
1	4	400M (20 million)	161s	124223
1	4	1.8G (80 million)	245s	326530

Table 2: Performance with Different Number of Vehicles and Configurations

Header CPU usage	Header memory	Worker CPU	Worker memory
Around 15% Peak 25%	8G	20%-25%	40%*16G
Around 20% Peak 25%	8G	25%-60%	90%*16G
Around 12.5% Peak 25%	≈8G	15%-35%	40%*16G
Around 15% Peak 25%	≈8G	25%-50%	70%*16G
Around 12.5% Peak below 25%	≈8G	12%-15%	25%*16G
Around 12.5% Peak below 25%	≈8G	25%-40%	50%*16G

Because header CPU usage and memory usage do not reach high values for different amounts of data, there is no need to add another header. Based on the results, I chose to use the cluster consists of one header and four workers. Because the number of vehicles will increase significantly in the future, cluster that can handle large amounts of data and have good performance is necessary.

7 Preprocessing, Model Training, Software and Hardware - Benteng Ma

My section includes Data Preprocessing over our datasets, Body Outlier Detection and Behaviour Classification Model Implementation and Training, Software Implementation, Code Migration and Hardware Implementation.

7.1 Declaration of Authorship

I hereby declare that the following statements are true and valid:

- 1) I fully understand the definition of plagiarism and its consequences.
- 2) I promise that no part of the project is suspected of plagiarism and that it is all my original work.
- 3) All material for this project, including code and articles, does not contain anything that has been published or written by anyone unless I explicitly state that we are referring to someone else's project or work.
- 4) I own the full copyright of this project including the code and article and do not use without permission.
- 5) The thesis presented is the result of my independent research under the guidance of our supervisor.
- 6) The paper is written by me and I am responsible for what I wrote.

7.2 General Description of Work completed

I implemented the software of detecting dangerous driving behaviours. To achieve it, we designed, collected and labeled several datasets (see Zhejia's part, section 6.3) including body outliers, motion classification of head and hands, etc (we will introduce in detail). I designed the working principle of my system, implemented the candidate models within PyTorch framework, preprocessed the data and trained them, then Xiaoxiao (see section 8.5) evaluate the models with mathematical methods and we selected the final models to use for fine tuning, with xiaoxiao's (see section 8.4.2) active learning methods.

I then packed the trained models within Python classes and designed prediction interfaces for these models. Yushi (see section 9) implemented fatigue and motion detection models, as well as GUI, and I combined our work together into one software.

We selected Raspberry Pi to be our hardware platform, I managed to assemble two devices based on two Raspberries, and I migrated the software onto them. Zhejia (section 6.4.1) installed the GPS and 4G communication module onto the device, I also helped with some of his work.

7.3 Image Preprocessing

I implemented code for image preprocessing, which shrinks the images from different sizes to 400×300 pixels for detection models, and for classification models, the images are resized to 128×128 . Before resizing these images, they are first filtered by anti-alias filters to avoid aliasing. For resizing I used Resize instance from PyTorch transforms library, the benefit is that anti-alias filter is already inserted.

For both detection and classification models, larger input size result in larger memory occupation and greater computation complexity, smaller images are already enough to provide the information for our tasks. Additionally, resizing all the input images to the same size allows me to use mini-batch training [31].

I also normalized all the pixel values of the images, turning them from 0 to 255 unsigned 8-bit integer to 32-bit floating point values, ranging from 0 to 1 [4]. This first transform discrete unsigned integers of the image data to processable floating-point; and also limited the value of gradient to train more stably.

One difficulty is to keep the correct bounding box coordinates when the image sizes are changed. So I convert the bounding boxes coordinates from pixel coordinates to relative ones, for example, if one coordinate is (20/160, 40/200) then the relative coordinate is (0.125, 0.2), which is not affected by the size change of the image. It is easy to convert relative coordinates back to pixel ones, which is very helpful for cases such as random image cropping in data augmentation.

Part of the code of voc_dataset.py

```

1  # transform function with automatic anti-alias filter
2  self.transform = transforms.Resize(size, antialias=True)
3  # read image, bounding boxes and labels
4  img = VOCDataset.read_image(img_path)
5  bbox, label = VOCDataset.read_annotation(anno_path)
6  # get bounding boxes with relative location
7  bbox_ref = torch.zeros(*bbox.shape)
8  bbox_ref[:, 0] = bbox[:, 0] / img.shape[2]
9  bbox_ref[:, 1] = bbox[:, 1] / img.shape[1]
10  bbox_ref[:, 2] = bbox[:, 2] / img.shape[2]
11  bbox_ref[:, 3] = bbox[:, 3] / img.shape[1]
12  img, bbox_ref = self._resize(img, bbox_ref) # resize the images
13  img = img.type(torch.float32) # convert to floating point
14  img /= 255 # change values into 0 ~ 1

```

7.4 Detection and Classification Models

As mentioned at section 7.2, I first implemented the detection and classification models, make them available to be trained and tested over our dataset. After that we measured the performance of these models and selected some to use in the end. However, for convenience, I will first introduce how these models are selected, before introduce the detailed implementation.

7.4.1 Function of the Models

The detection model gives prediction bounding boxes of the driver's head and hands; classifier models will give classifications based driver's cropped head and hands (cropped with bounding boxes). One classifier is able to classify (with head images) if the driver is eating or drinking, or talking on the phone; another model sees (the hands) if the driver is holding something, such as a mobile phone.

I also trained two models to classify if the driver is wearing a mask (Because of Covid-19, companies of cabs and buses might request their drivers to keep the masks on all the times), or sunglasses (with which, the fatigue model cannot detect the driver's eyes). Detector model also use classifiers to select correct cropped images.

7.4.2 Selecting the Body Outlier Detection Model

Object detection models are more difficult than classification models in implementation, I will give my reasons of my selection from the perspective of implementation, and Xiaoxiao (section 8.5) will give her reasons from the view of accuracy measurements.

Choosing FasterRCNN We selected FasterRCNN [41] as our body outlier detection. It has a lot of examples in literature that supports our usage; the model is also claimed to work better at small object detection than YOLO and SGD (another common one-stage object detection model), reviewed from [58], the article claims that there are still space of improvement for FasterRCNN in detection small objects, but confirms the Region Purposing Network used with FasterRCNN that improve the performance for small sizes. PyTorch provides pre-implemented FasterRCNN and RetinaNet models, that can be called directly from library, greatly saving our time.

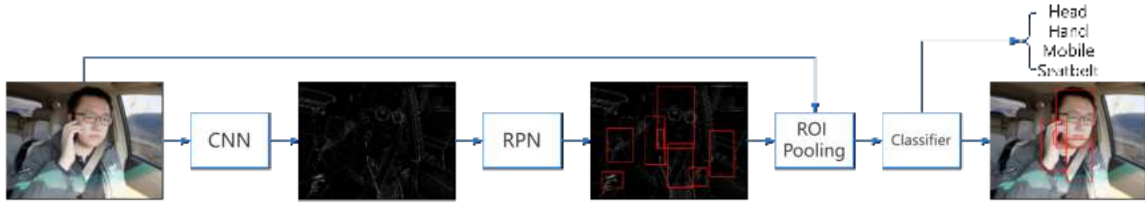


Figure 17: Detection Process of FasterRCNN

Why not choosing YOLO or RetinaNet YOLO [39] is one-stage, faster than most two-stage models, first reason we did not choose it is because we wanted to detect small object including mobile phones, YOLO is claimed not very good at these types of detection [58]; though we changed the original design and decided only to detect the driver's head and hands, we still did not use YOLO, because YOLO is more complicating to be used, with more parameters and difficult non-neural network algorithms. **RetinaNet** [33] has very similar interfaces to FasterRCNN, but it does not have good accuracy as FasterRCNN over some datasets [58], and the PyTorch provided RetinaNet cannot provide reasonable score of bounding boxes after trained by me, since it gives more than one predicted bounding boxes of each outlier (head and hands), and we only peak up the one with the highest score, this made it difficult for me to choose the correct prediction.

7.4.3 Further Discussion about Model Selection

Objective detection models are very difficult to learn and use, because of various parameters and characteristics. We did not manage to have a comparison experiment for object detection models like what we did for classification models in (section 8.5.2), though we really want to. After we changed the target of detection model to only detect head and hands, there are no longer very small objects, yet YOLO has been proved to be the fastest among the three [58], I think it might be quite beneficial to use YOLO. As a result, the future work might include experiment towards YOLO.

In addition, principles of all these three models made them focus on the pattern of the object, ignoring the relation of locations. For example, when we humans look for a person's hands in an image, we might first look for the body, the arm, and finally the hand, it is difficult to

search for hand directly; same thing happens for the models, that it can be quite beneficial if the model is able to consider the relation between body outliers, and I will take the research of tracking body outlier as one of my future works.

7.4.4 Selecting the Behaviour Classification Model

I implemented, trained and tested AlexNet [25], VGG-16 [49], GoogLeNet [50], ResNet-34 [13] and MobileNet-V3 [16]. Xiaoxiao will explain the measurement of classification accuracy and speed (section 8.5.2), which are common concerns of researchers. An additional important factor to measure these networks is size in storage: AlexNet: 625MB, VGG-16: 1.5GB, GoogLeNet: 64MB, ResNet-34: 243MB, MobileNet-V3: 48M. Our two Raspberries have 1GB and 8GB memory respectively, as a result, although VGG has good accuracy in measurement, it is not acceptable for small devices. We selected ResNet-34 and MobileNet-V3 in the end. ResNet-34 has the best accuracy and good speed, MobileNet-V3 has the best speed and good accuracy. Please view Xiaoxiao's detailed comparisons (section 8.5.2).

7.5 Implementing the Model, Transfer Learning and Training the Models

I used PyTorch programming framework for implementing the models, testing them, and also for the final implementation, because it is the most convenient to program. I will discuss later about pros and cons of using other frameworks (section 7.6.1).

7.5.1 Code Implementation of the Models

All the models I used is pre-implemented in PyTorch TorchVision library. I created the model with classes provided by PyTorch, then changed the out layers of the models, to make them output the classification result (object detection is also a classification tasks, with bounding box predictions). Therefore, with correct input data, the model can directly be trained and make predictions.

Within the code below, I use pretrained backbone (feature extractor); the parameter *pretrained* refers to the whole model; and the parameter *trainable.backbone.layers* is to set how many pre-trained layers in the backbone can be trainable in my training session.

Notice: the name "resnet50" and "mobilenet_v3" are not about classifiers, but is backbone feature extractor using same structure of ResNet and MobileNet.

Building the object detection models, in detector_train.py

```

1 # import pre-implemented TorchVision models
2 from torchvision.models.detection import \
3     fasterrcnn_mobilenet_v3_large_320_fpn, fasterrcnn_resnet50_fpn, \
4     fasterrcnn_mobilenet_v3_large_fpn, retinanet_resnet50_fpn
5 from torchvision.models.detection.faster_rcnn import FastRCNNPredictor
6
7 model = None
8 if model.type == 'mobilenet':
9     model = fasterrcnn_mobilenet_v3_large_fpn(
10         pretrained=False, progress=True, pretrained_backbone=True,
11         num_classes=train_dataset.num_classes,
12         trainable_backbone_layers=trainable_backbone_layers,
13     )
14 ...

```

```

15 elif model_type == 'resnet':
16     model = fasterrcnn_resnet50_fpn(pretrained=True)
17     in_features = model.roi_heads.box_predictor.cls_score.in_features
18     model.roi_heads.box_predictor = FastRCNNPredictor(
19         in_features, train_dataset.num_classes)
20 elif model_type == 'retina':
21     model = retinanet_resnet50_fpn(
22         ...
23     )

```

For classification models, I remove the last layer of the pre-implemented model (they do not match the output shape of ours if I do not change them), added a new fully connected layer, with output neurons matching the number of output classes. Then finally I put a Softmax layer [11] which converts all the outputs from all the neurons into " $sum = 1$ " probabilities, that the *argmax* of the output is the classified result.

Building the classification models, in classifier_model.py

```

1 class ClassifierModel(nn.Module):
2     def __init__(self, net_type: str, num_classes: int):
3         super(ClassifierModel, self).__init__()
4         self.oriNet = None
5         self.get_classification_model(net_type, num_classes)
6
7     def get_classification_model(self, net_type, num_classes):
8         model = None
9         if net_type == "alexNet":
10             model = torchvision.models.alexnet(pretrained=True)
11             model.classifier[6] = nn.Linear(4096, num_classes)
12         elif net_type == "vgg": ...
13         elif net_type == "googleNet": ...
14         elif net_type == "resNet":
15             model = torchvision.models.resnet34(pretrained=True)
16             in_features = model.fc.in_features
17             model.fc = nn.Linear(in_features, num_classes)
18         elif net_type == "mobileNet":
19             model = torchvision.models.mobilenet_v3_large(pretrained=True)
20             model.classifier[3] = nn.Linear(1280, num_classes)
21         self.oriNet = model
22
23     def forward(self, x: torch.Tensor):
24         x = self.oriNet(x)
25         outputs = nn.functional.softmax(x, dim=1) scores[idx]}
26         return outputs

```

7.5.2 Using Transfer Learning

Transfer learning is to borrow the layers from some models, already trained for some different tasks, migrate them into my own model, and use for my own purpose [54]. Doing this is because our own collected dataset is not big enough to provide all the possible patterns of images, yet there are many networks already trained over very complex datasets that have strong ability to extract patterns, I can use these layers as my own feature extractor and only to train the rest layers. There are doubts that whether transfer learning over ImageNet dataset (The one PyTorch models are pre-trained) is truly helpful to the accuracy of the model [24], when I trained

the models, with transfer learning, training takes less iterations to reach the peak performance than initialize the model with random values of parameters - without transfer learning, the values of parameters of a model should be initialized randomly with carefully designed distributions, luckily, I do not need to worry on such things.

For object detection models, I used pre-trained backbone (trained over ImageNet dataset [25]) and blank detection layers; for classification models, I first load pre-trained parameters of the whole model, and replace the last layer of my own, with freshly initialized parameters. I set first half of the detection backbone layers and classification model layers to be settled, and half to be trainable, so that the lower levels of pattern extraction can be protected while upper levels are trained for new tasks.

7.5.3 Before Training: Implementing PyTorch Dataset and Dataloader

I split all the datasets of 7:3 rate of train and validation datasets randomly. Therefore, the performance of the model is traced during training.

I followed Pascal VOC dataset Format for Object Detection Dataset Which is a commonly used dataset format. I chose a dataset labeling tool named ImgLabel, which is able to output VOC format data, and I need to load these data according to the format.

Example of VOC data format of objects

```
1 <object>
2   <name>drv_r_head</name> <pose>Unspecified</pose>
3   <truncated>0</truncated> <difficult>0</difficult>
4   <bndbox>
5     <xmin>309</xmin> <ymin>18</ymin> <xmax>457</xmax> <ymax>234</ymax>
6   </bndbox>
7 </object>
8 <object>
9   <name>drv_r_hand</name> <pose>Unspecified</pose>
10  <truncated>0</truncated> <difficult>0</difficult>
11  <bndbox>
12    <xmin>455</xmin> <ymin>401</ymin> <xmax>534</xmax> <ymax>479</ymax>
13  </bndbox>
14 </object>
```

I Implemented the PyTorch abstract Dataset class I implemented `__getitem__(self, idx)` method, and `__len__(self)` methods, which map every piece of data in the dataset with an index, thus the dataloader knows how to fetch the data, as well as shuffle them (it only needs to shuffle the index list).

voc_dataset.py

```
1 class VOCDataset(Dataset):
2     def __init__( self,
3                   dataset_dir: str, size: tuple, resize_mode: str,
4                   use_random_augmentation: bool, return_filename=False, ):
5         ...
6     def __getitem__( self, index):
7         img_path, anno_path = self.img_path_list[index], self.anno_path_list[index]
8         img = VOCDataset._read_image(img_path)
```

```

9         bbox, label = VOCDataset._read_annotation(anno_path)
10        img, bbox = detector_preprocess(img, bbox)
11        if self.augmentation:
12            img, bbox, label = detector_augment(img, bbox, label, other_params...)
13        return img, bbox_ref, label
14    @staticmethod
15    def _read_annotation(anno_path: str) -> tuple:
16        anno = ElementTree.parse(anno_path)
17        bbox, label = list(), list()
18        for obj in anno.findall('object'):
19            bndbbox_anno = obj.find('bndbbox')
20            name = obj.find('name').text.lower().strip()
21            if name in CLASS_NAMES_MAP.keys():
22                bbox.append([
23                    int(bndbbox_anno.find(tag).text) - 1
24                    for tag in ('xmin', 'ymin', 'xmax', 'ymax')])
25                label.append(CLASS_NAMES.index(CLASS_NAMES_MAP[name]))
26        bbox, label = torch.from_numpy(bbox), torch.from_numpy(label)
27        return bbox, label
28    @staticmethod
29    def _read_image(img_path) -> np.ndarray:
30        img = cv2.imread(img_path)
31        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
32        return img.transpose((2, 0, 1))

```

My dataset for detection instance first looks for images and referred labels in xml files (according to VOC data format); then put the images into a list, mapped with the list index which will be used as their own index. For images/xml files stored in different folders, the dataset instance can also be joint together. When `__getitem__` method is called, the image, labels and bounding boxes are first read from the disk; the data augmentation methods implemented by Xiaoxiao (section 8.3) will then randomly change the image (if the image is cropped or flipped, the bounding boxes will be changed too); after that, preprocessing method will run over the image, and the outcomes will be pushed to the dataloader instance.

There are no xml files for classification. For classification, I only need to load the images from the folders, their labels are dependent on the folders they are in.

The dataloader instance of Dataloader class is pre-implemented by PyTorch, it allows multiprocessing for data fetching. I built the dataloader for training and validation by establishing the dataset instances and passing to it. I decided the number of working processes, batch size (of mini-batch [31]), as well as whether to shuffle the index list. In reality, dataloader does not know how to manage multiple bounding boxes for batches, so I set the batch size into one and pack a larger batch manually (with my own code) later.

7.5.4 Training Models with Pytorch

When training these models, I ran over the whole dataset, each time with shuffled sequence, over and over again. The dataloder first preprocess and augment the data by calling dataset functions, then give the data to the trained model. The parameters of the model are updated during each "backward propagation" of the loss value of each prediction over an input example.


```

1 for epoch_idx in range(start_epoch, epochs):
2     images, targets = [], []
3     for img, bbox_ref, labels in tqdm(train_loader, desc='epoch %d' % epoch_idx):
4         if cuda:
5             bbox_ref, img, labels = bbox_ref.cuda(), img.cuda(), labels.cuda()
6         target = {'boxes': bbox_ref, 'labels': labels,}
7         images.append(img)
8         targets.append(target)
9         if count % batch_size == 0 or len(train_dataset) - count <= batch_size:
10            model.train()
11            model.zero_grad()
12            optimizer.zero_grad()
13            loss_dict = model(images, targets)
14            losses = sum(loss for loss in loss_dict.values())
15            losses.backward()
16            optimizer.step()
17            images, targets = [], []

```

Above is part of my code for training detection models, below is for training classification models. Detection models have loss functions of both bounding box accuracy and classification accuracy; classification models only have loss function of classification accuracy.

Training Classification Models

```

1 for epoch_idx in range(start_epoch, epochs):
2     for imgs, labels in tqdm(train_loader, desc='epoch %d' % epoch_idx):
3         one_hot_labels = \
4             torch.nn.functional.one_hot(labels.type(torch.int64), len(class_list))
5         if cuda:
6             imgs, one_hot_labels = imgs.cuda(), one_hot_labels.cuda()
7         model.train()
8         model.zero_grad()
9         optimizer.zero_grad()
10        out = model(imgs)
11        loss_ = loss(out, one_hot_labels)
12        loss_.backward()
13        optimizer.step()

```

With mini-batch [31], the propagation are taken over after the model has finished prediction over a whole batch of samples (and the gradient is computed based on the mean of the losses). One reason for doing this is that when training on graphic processing units (GPU), the data runs in a single instruction multi data-stream state (SIMD), as a consequence, time taken by running 4 images in parallel might be similar to running with only one, so training on batch saves time; additionally, taking each steps by considering more samples makes the direction of gradient descend more stable.

The model was trained on my laptop (Intel 8-core CPU, 64GB memory, NVIDIA 16GB GPU), with 4 dataloader process workers, 4 images each batch for detectors, 8 each for classifiers, learning rate of 1e-4, and was trained with Adam optimizer [23] for gradient optimization. The detection model and classification model is first trained by me over around 3,000 images, then given to Xiaoxiao to do active learning.

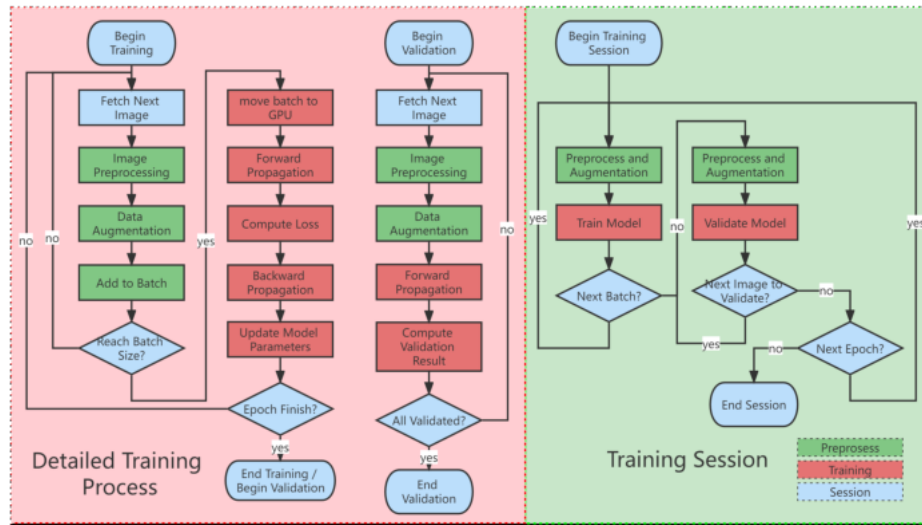


Figure 18: Process of Training the Model

7.6 Model Packaging and Software Implementation

What I did this part is to combine the detection and classification models so that they can work together. The detection model gives the predicted bounding boxes, the program cropped these sub-image boxes and send them to classification models to classify the behaviour, just like what is shown in the image.

I packed my detection models and classification models into python classes, and unified the data interfaces, made them much easier to be called and used by higher level functions. Based on the packed classes, I can easily combine the models to achieve more more tasks, in my case, decide whether the driver is doing dangerous behaviour when driving.

The image includes my design of classes and instances. The results of object detection will be contained by instances of class *DetectorRefBoundingBoxes*, which stores the bounding box information in unified method, and can only be read and modified with enumerate keys. I used an Abstract Detector class to unify the Detector classes, which is similar to interfaces in Java, allowing me to call different model implementations with the same method. Although there is only one detector implementation for now (FasterRCNN), this framework is ready for other implementations in the future. There are classifier instances of Classifier class, and one body outlier detector instance of *FasterRCNNDetector* I've just mentioned. These instances are managed by detector_classifier instance of *DetectorClassifier* class.

7.6.1 Implementation of *DetectorClassifier* Class

I've designed different algorithms in *DetectorClassifier*, I managed to implement the first one, and the rest are designs for future experiments.

My Current Implementation is to take the objects of the highest scores to be the selected objects. Then, I cropped the sub-images of the diver's head and hands, sent them to several classifiers. These classifiers classifies if the driver is wearing a mask or sunglasses, or is likely to be eating/drinking, talking on a phone, or focus on driving (section 20). As drawn in my

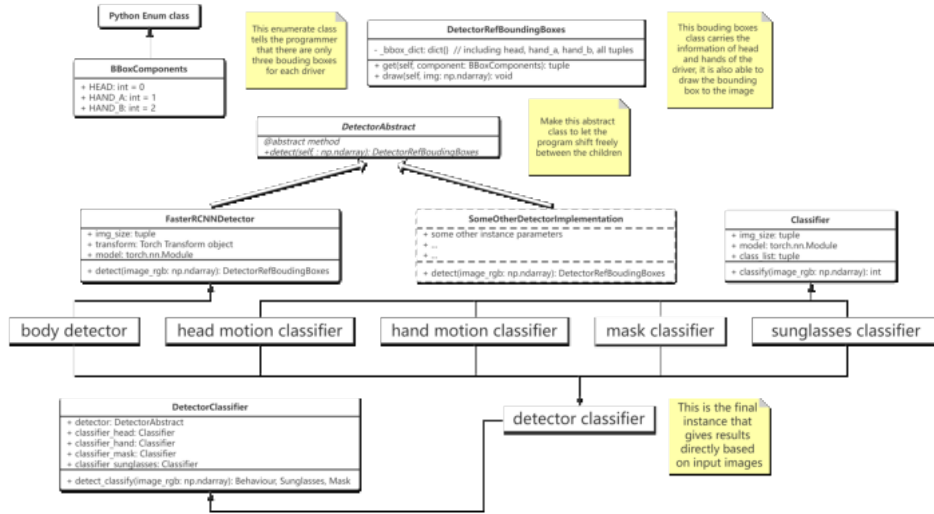


Figure 19: Process of Training the Model

graph, I use two MobileNets for mask and sunglasses (easier tasks) classification, since they are "fast and accurate" as we've measured (section 8.5.2); I used two ResNet-34 for classification of head and hand motion (more difficult), since they are the most accurate, and not very big in size. In current design, classification of driver's behaviour based on driver's head and hands runs in parallel, if any one result is classified to be dangerous, the alarm will be triggered (to other drivers, use Zhejia's method in section 6.4.1). The system has already achieved most our designed plan, it works well under complex situations. However, from a higher standard than original expectation, this design is not satisfactory. I thought of other ways of implementations and I would like to have a discussion.

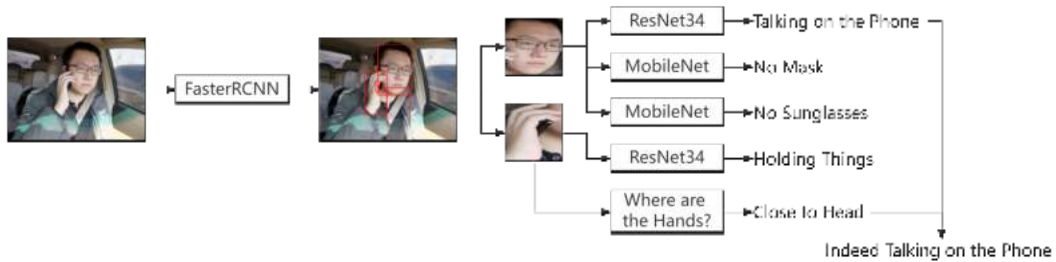


Figure 20: Detection-Classification Process

Discuss about better solutions The first thing might be able to improve is the two classifiers for masks and sunglasses. I hope to design a new classifier which is able to do **multi-label** classification (the current classifier it called multi-class classifier, that selects one result from multiple choices). Different from current models, this model is able to select more than one label for each time, therefore, I can make one label "mask" and another "sunglasses" and make decision for only once. To achieve this, I will remove the SoftMax layer (multi-class classification) at the end of the model, replace it with a parallel Sigmoid layer (originally for binary classification) for selecting two True-False options. The benefit is that I can totally cancel one of the running models to save precious computation power of Raspberry Pi.

Next issue is that sometimes the model might select same hand twice, but totally miss another one, therefore, a solution can be adding another classifier to detect right or left hand. Similar method can also help to solve the case that more than one bounding boxes are purposed around a head or hand, classifiers are good at distinguishing whether it is a good crop or not. However, this adds extra work load for CPU, so the task on first priority is still to increase the accuracy of detection.

Additionally, the smartness of the classification might still be able to improve. Currently, when the driver touches his/her face or mouth, they might be mistakenly recognized eating or talking on the phone, because the information from head and hands are not well combined. I think there is necessity to add more complex logic, such as when the head classifier detect that the driver is talking on the phone, check whether the hand is holding a mobile, and whether the hand location is close to the head. For a longer future, such justification might also be implemented by a machine learning model.

I used PyTorch for model implementation, which is convenient to implement. However, PyTorch is not the fastest framework. Some compiled model frames works such as TensorFlow Lite [32] has smaller weight and can run faster on a small device. If our project is going to be pushed to the market, it can be beneficial if we use those light-weight frameworks.



Figure 21: Left: current decision process; Right: possible solution

7.7 Initialize Raspberry Pis

I used two Raspberries as base platform of hardware. I first installed 64-bit Raspbian Linux OS, a mobile light version of Debian Linux, on the board. Since the device uses BroadCom ARM CPU, the assembly language should be Aarch-64 for all the software and libraries installed. The reason for using Raspberry Pi is because it is a well-known type of microcomputer and IoT

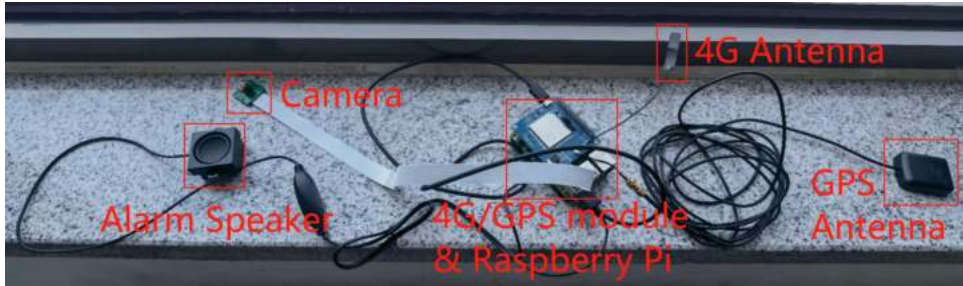


Figure 22: Components of One Device

device, with size of a credit card, but owns same computational power as a smart phone. It provides various of serial port interfaces and large unified memory.

In this project, I used two types of Raspberry Pi devices. Raspberry 3b+ and Raspberry 4b. 3b+ version (I bought this years ago) uses a ARM Cortex-A53 CPU, 1GB LPDDR2 memory; 4b version (bought in this semester) uses ARM Cortex-A72 CPU, with 8GB LPDDR4 memory, both requires 15-Watt power supply. Most of the port interfaces are same for the two versions. Both the Pi boards are connected with a EC200U-CN communication module, through serial ports, providing both GPS and 4G connection.

There are Python related libraries needed to install before the program is runnable on Raspberry Pi. Dlib, "pip install dlib"; Pyside2, "pip install pyside2"; OpenCV-Python, "pip install opencv-python==4.3.0": the ARM version of Pyside2 is only able to accept OpenCV-Python versions include and lower than 4.3.0; PyTorch with the pip command recommended on the PyTorch official website; Tensorflow2.x first run "sudo apt install libhdf5-dev", find "tensorflow-2.6.0-cp39-none-linux_aarch64.whl" and install it with pip command.

7.8 Code Migration

Based on the libraries just installed, I combined my part of software and Yushi's (section 9) together to be the software prototype, and migrate it to the Raspberry Pi. The performance cannot reach what we got over PC platform. One reason is that neither single core frequency nor instruction per cycle can be comparable to a PC; we do not have more time to implement better schedule method for our program, and all the models are running in serial manner, not parallel - not all the cores is running in 100%. I can combine my models to one in the future, as mentioned; additionally, I think fatigue detection and emotion detection (section 9) should be scheduled to run in parallel, and the speed can further improve if the Dlib (section 9.4.1) framework and TensorFlow framework can be unified; these can also be my work in the future.

Another method instead of parallel is that only to do one type of detection during each in-taken frame. For instance, the input FPS is 30, and fatigue model need all the frames for eyes blink detection; then I may schedule the emotion model, and the detection-classification model to only in-take one frame out of ten, one take odd frames and another take even, because detection of behaviour and emotion do not need such high frame rate, and this can save a lot of computation power.

8 Data Augmentation, Active Learning, Model Selection and Evaluation - Xiaoxiao Yu

8.1 Declaration of Authorship

I hereby declare that the following statements are true and valid:

- 1) I fully understand the definition of plagiarism and its consequences.
- 2) I promise that no part of the project is suspected of plagiarism and that it is all my original work.
- 3) All material for this project, including code and articles, does not contain anything that has been published or written by anyone unless I explicitly state that we are referring to someone else's project or work.
- 4) I own the full copyright of this project including the code and article and do not use without permission.
- 5) The thesis presented is the result of my independent research under the guidance of our supervisor.
- 6) The paper is written by me and I am responsible for what I wrote.

8.2 Description of all the work completed

In the project, I am mainly responsible for three sections which are data augmentation, module selection and evaluation and active learning.

I started by learning about different image augmentation methods and eventually selected four methods to program and implement for our project. After that, I started to learn active learning to reduce the manual annotation time of the project and achieve better results with fewer data. I programmed two different active learning methods for image object detection and image classification respectively. In particular, I designed the Valid information rate and Consistency based active learning for object detection algorithm (V+C) based on the requirements of this project. I then applied them to our project. Finally, I learned about various model evaluation methods, selected the right model for the project using appropriate model evaluation methods, and finally evaluated the performance of all models.

8.3 Image Data Augmentation

In this project, Image data augmentation is used to prevent over-fitting and enhance the generalization ability of the model. I used four broad categories of image data augmentations which are flipping, cropping, color space transformations and Gaussian filter. These enhancements are randomly appended to the images. I'll show the reasons, code, and results for each of these data augmentations below. The full code is in "augmentation.py"

Flipping Flipping is used to simulate image flipping caused by different mounting angles of the vehicle camera, and different driving positions in different countries. This method has been proved to have a good effect on CIFAR-10 and ImageNet data sets [46]. The label boxes will change position during flipping. The code and effect pictures are shown in figure 23.



Figure 23: Flipping



Figure 24: Cropping

Cropping Cropping is used to simulate the camera being blocked. Training when the head, hands, or other objects are blocked, the object can still be classified and recognized. In cropping, the label boxes are reduced in size according to cropping ways. The code and effect pictures are shown in figure 24.

Color space transformations Color space transformations can change the image brightness, contrast, saturation, and hue to simulate the image of the different external environments, such

as strong light, late-night, tunnels, and so on. Chatifleet et al. [6] found that changing color space reduced classification accuracy by approximately 3% on ImageNet [7] and PASCAL [9] VOC datasets, demonstrating the effectiveness of this augmentation. The code and effect pictures are shown in figure 25.



Figure 25: Color space transformations



Figure 26: Kernel filters

Gaussian filter In this project, Gaussian filters can make noise or blur images. Gaussian blur is used to improve the ability to resist motion blur, and Gaussian noise can improve the details of more interesting objects. Kang et al. [18] proved in ResNet [13] CNN architecture that the prediction error rate could be reduced by 0.67%. The code and effect pictures are shown in figure 26.

8.4 Active Learning

8.4.1 Application Background in Project

In the project, we encountered many difficulties in image object detection and image classification. Firstly, due to the insufficient sample size of annotation, although we collected a large number of data sets, due to the problem of time and energy, we could not label all the data sets. At the same time, we found that there were a large number of similar photos in the data collected, and repeated labelling of similar photos not only cost energy but also failed to improve the training effect. In addition, our object detection and prediction effect are not good, and a large number of effective annotation data are urgently needed for training. So based on these questions, I decided to use active learning.

Active learning [42] can help to select the most needed annotation data from the unlabeled pool and hand it to human labelling. It solves the problem of insufficient and difficult data annotation in machine learning. Through active learning, we can achieve better predictions with less annotated data. The application process of active learning in our project is shown in Figure 27. Our project involves active learning in image classification and image object detection and all based on 3000 training base. The Machine Learning Model in the figure is done by Benteng Ma shown in section 7.5. The Manual Annotation in the figure is done by all the group-mates in section 6.3. I will introduce the active learning algorithm of those two models respectively in the following part.

8.4.2 Active Learning for Image Classification

(1) Algorithm - Entropy Method

In active learning for image classification, I choose pool-based active learning [30] and use the entropy method [44] as the query strategy frameworks.

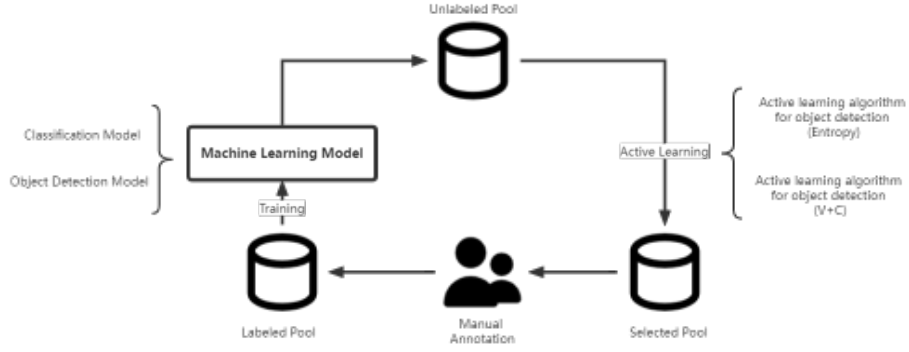


Figure 27: Active learning flow chart

Pool-based active learning can extract a certain number of unlabeled samples from the unlabeled sample pool, rather than select by threshold. It is more suitable for our project.

The formula of entropy method is shown in equation 1. This algorithm can determine whether the unlabeled sample contains more information. Compare to other active learning algorithm in classification, the entropy method takes into account the impact of all category predictions on the final result. Other methods only consider maximum probability or secondary probability, which is not comprehensive for the multi-classification model. The core code of this algorithm is shown below. More relevant code can be seen in GitHub "al-classification.py".

$$x_{ENT}^* = \arg \max_x - \sum_i P(y_i|x; \theta) \log P(y_i|x; \theta) \quad (1)$$

The core part of active learning algorithm for image classification

```

1  ...
2  for imgs in tqdm(unlabeled_loader):
3      task_model.eval()
4      # Do the classification and get the score
5      score_list = task_model(imgs).tolist()
6      # Entropy calculation
7      log_probs = np.log2(score_list[0])
8      entropy = -1 * np.sum(score_list[0] * log_probs, axis=None)
9      entropy_list.append(entropy)
10     # Choose the picture that is most informative
11     if len(entropy_list) > labeled_number:
12         uncertainty_list = pd.Series(entropy_list).sort_values().index[:labeled_number]
13     else:
14         uncertainty_list = pd.Series(entropy_list).sort_values().index[:len(entropy_list)]
15     ...

```

(2)Result

After using this active learning algorithm for image classification training, our final classification effect reaches our expectation. At the same time, Manual labelling time and training time are greatly saved. Due to time limitation, we were not able to conduct a comparative experiment on this part (Random selection vs Entropy selection), however, I lot of people have already

shows the effectiveness of Entropy selection. Devis et al.?? reduced 10% required number of training samples by using Entropy selection. The previous experience is sufficient to prove that this algorithm is effective.

8.4.3 Active Learning for Image Object Detection

(1)Algorithm - Valid Information Rate and Consistency Based Active Learning for Object Detection(V+C)

I encountered a lot of difficulties when designing the algorithm. Compared with active learning in image classification, object detection has not been explored much. The difference between active learning based on object detection and traditional active learning lies in that it not only needs to correctly identify objects(classification), but also need to judge the box location of objects(localization). Therefore, its evaluation index will not only include confidence, but also need to consider the accuracy of positioning. At the same time, object detection often needs to recognize multiple objects in a picture, and different objects have different recognition requirements. For example, some objects need to recognize the contour very accurately, while some objects only need to recognize their existence, so how to balance the recognition of each object is also a problem.

There are few active learning algorithms for object detection. Most relevant studies [8, 12] in this field basically focus on classification and ignore the results brought by localization to detection results. Kao et al. [19] put forward two concepts which are Localization Tightness with the classification information (LT/C) and Localization Stability with the classification information (LS+C), taking both classification and localization into consideration, and the experimental results are higher than the method that only considers classification. Yu et al. [57] added steps to balance the advantages and disadvantages of different object detection based on the consideration of positioning and classification. But the above methods all do not take into account different object recognition requirements.

Referring to previous algorithms and combining with the requirements of this project for object detection, I proposed Valid Information Rate and Consistency Based Active Learning for Object Detection(V+C). The flow of the algorithm is shown in Figure 28. Compared with existing algorithms, This algorithm not only considers the classification and localization, but also introduces the concept of valid information rate to solve the problem of different object recognition requirements problem. I'll describe my algorithm design in detail below. The code is in "al-detection.py".

(1.1) Algorithm Component

The active learning algorithm for image object detection is shown in Figure 28. The unlabeled sample pool is called I , and each image in the sample pool is called i_x . The algorithm includes image detector D and parameter d , image data augmentation A , filter F and parameter f , and matcher M .

Image detector D It is an object detection model trained by machine learning which is shown in section 7.4.2.

Image data augmentation A Five augmentation methods are selected in the section of Image data augmentation 8.3. These are brightness, contrast, saturation, hue and Gaussian blur.

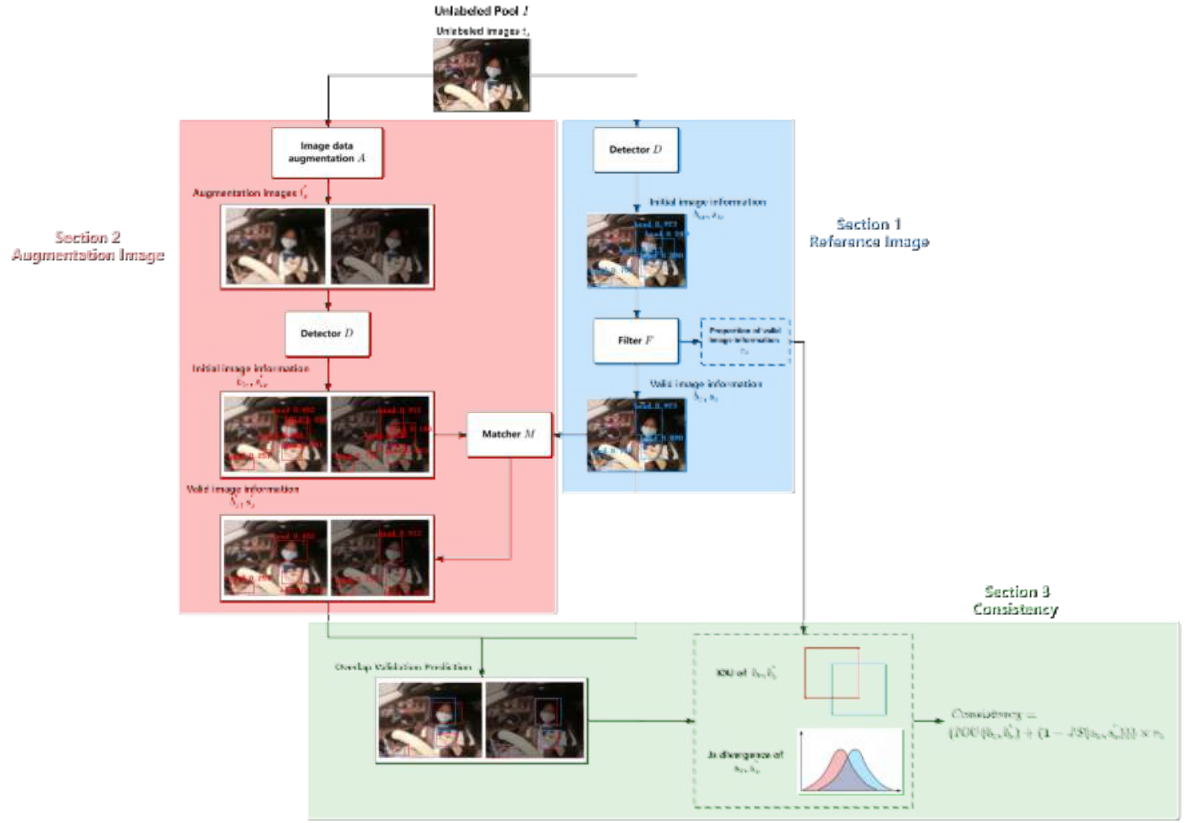


Figure 28: Valid Information Rate and Consistency Based Active Learning for Object Detection(V+C)

Filter F The Filter can filter valid image information and get valid information rate from the initial image information according to the different recognition requirements of different classes. Parameter f contains thresholds of valid information confidence set by different classes. Threshold Settings will be selected empirically and adjusted as the model is optimized. the confidence of Images above the threshold will be selected as valid images. The ratio of images higher than the confidence threshold to all images is valid information rate.

Matcher M Matcher will make one-to-one match between the augmentation image information and the reference image information. When the box label of the augmentation image is the same as that of the reference image, the box with the highest IOU is taken as the matching box. If no box match occurs, each parameter is assigned a value of 0.

(1.2) Algorithm Process

The algorithm is divided into three sections. The first section deals with reference image information. The second section deals with augmentation image information. The third section get the consistency of the image information.

First section In the first section, the unlabeled image sample i_x is detect by detector $D(i_x; d)$ and get the initial image information b_{ix} and s_{ix} . b_{ix} is box location, s_{ix} is confidence. The initial image information is put into filter F to get the valid image information and valid information rate. The formula is shown below.

$$b_x, s_x, r_x = F(D(i_x; d); f) \quad (2)$$

Second section The second section applies multiple self-selected image augmentation to the same unlabeled image sample i_x , and obtains multiple groups of augmentation images. The augmentation images were detect by the same detector and get the initial image information. Put the obtained initial image information into matcher M for matching and get valid image information. The formula is shown below.

$$b'_x, s'_x = M(D(A(i'_x); d)) \quad (3)$$

Third section The third section overlaps the obtained b_x, s_x with b'_x, s'_x . IOU is obtained from b_x and b'_x . The JS divergence is obtained from s_x and s'_x . The trend of JS divergence is opposite to that of IOU, so I reverse JS divergence to 1-JS. Finally, the sum of IOU and reversed JS divergence multiplied by valid information ratio to get image information consistency. The formula is shown below.

$$Consistency = (IOU(b_x, b'_x) + (1 - JS(s_x, s'_x))) * r_x \quad (4)$$

(2) Result

To verify the effectiveness of the V+C algorithm, I conducted two sets of experiments. In experiment 1, I selected a data set of 1000 unlabelled images of 8 volunteers. The V+C algorithm was used to select the 500 photos that were most in need of labelling. The selection rate of each volunteer's photos is shown in Table 3. As can be seen from the table, different volunteers have different selection rates. The selection rates for SUN.Y and ZHU.Y were as high as 90.3% and 83.7% respectively. Whereas the selection rates for YU.X and LIU.Z were only 14.3% and 21.5%. Based on the results, we found that the body types of the volunteers with high selection

rates were not the same as those in the training set, and at the same time, the photographic luminosity was different too. Therefore, in the subsequent collection for the dataset, we will focus more on this type of images.

In experiment 2, I labelled and trained the 500 images selected in Experiment 1. In the same time, I random selected another 500 images from the same 1000 unlabelled images for labelling and training. The model is a FasterRCNN model that has been trained with 3000 basic data. The results were listed in figure 29. From table, applying active learning, the mAP value after training was increased by 5%. In contrast, the mAP boost from random selection was only 3.3%. The above two experiments validate the effectiveness of V+C algorithm. Due to time constraints I was not able to compare V+C with other active learning algorithms on more authoritative datasets (VOC2012, VOC2007). This will be my future work.

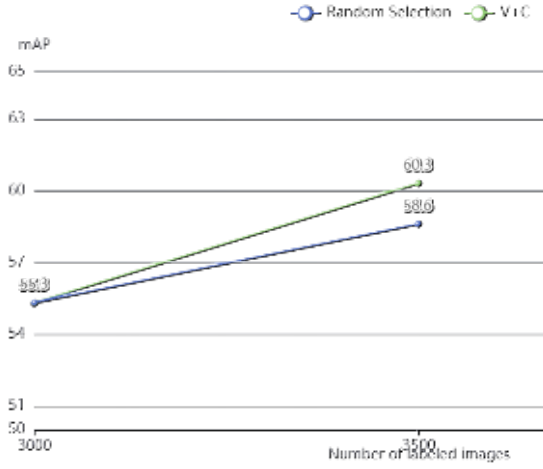


Table 3: selected rate of volunteers

Volunteer Name	Screening rate
SUN.Y	0.903
JU.Y	0.837
JL.Y	0.653
ZHOU.S	0.517
ZHANG.J	0.417
LI.J	0.375
LIU.Z	0.215
YU.X	0.143

Figure 29: Comparison Experiment of Random selection and V+C

8.5 Classification Model Evaluation

The model evaluation section is used to evaluate the strengths and weaknesses of five selected classification models which are AlexNet [25], VGG-16 [49], GoogLeNet [50], ResNet-34 [13] and MobileNet-V3 [16]. Use mathematical methods to select the optimal model and provided to Benten Ma to do classification training in section 7.4.4. Finally evaluate the final performance of binary classification and multiple classification after using the selected optimal model.

8.5.1 Evaluation Criteria

Most criteria can only one-sidedly reflect part of the characteristics of the model, so in the process of model evaluation, I use a variety of different criteria to evaluate the quality of a model. Image classification model used P-R graph, Average Precision (AP), Weighted F1 and Frame Per Second(FPS) as the evaluation criteria for model comparison, and the confusion matrix and accuracy were used to analyze the final performance. The reasons for the selection of some evaluation criteria will be described below.

P-R Curve

There are two commonly used performance evaluations of models, the precision-recall (P-R) curve, and the receiver operating characteristic (ROC) curve. I use P-R curve because when

there is a large difference between positive and negative sample sizes, the P-R curve can show obvious changes to show the defects of the model. However, the ROC curve only shows a small range change, giving an over-optimistic estimate for the model.

F1-score and Weighted F1-score

In our project, the binary classification model uses F1-Score as the evaluation standard without using precision and recall. The reason is that precision and recall are equally important to project objectives. It is necessary to increase the probability of being right in the positive sample, and it is also necessary to pay attention to the probability of being right in the positive sample. F1-score is the harmonic mean of precision and recall. So, I finally choose F1-score.

For the multi-classification model, Weighted F1-Score was selected as the evaluation standard. Compared with Micro F1-Score and Macro F1-Score, Weighted F1-score is more affected by a large number of categories and reduces the influence of rare categories. Because it is weighted by the proportion of the number of samples in each category to the total number of samples in all categories. In our multi-classification model, there are many rare categories. Their classification effect is not good, the meaning of existence is only to improve the accuracy of effective classification. So, for the validity of evaluation, I finally choose Weighted F1-Score.

8.5.2 Classification model selection

Table 4 shows the performance comparison between five commonly used models in image classification, which are AlexNet, VGG16, GoogleNet, ResNet-34 and MobileNet. The information is collected by Yushi Wang. I will evaluate the selected models from two aspects: classification accuracy and classification speed in our data set and choose the best model for our project.

	Layer num	ImageNet Top-1	ImageNet Top-5	Conv Layer	Kernel size	Feature
AlexNet	8	37%	16.40%	5	11,5,3	maximum parameters
VGG16	19	28.07%	9.30%	16	3	Pre-training
GoogleNet	22	30.20%	6.70%	21	7,1,3,5	Inception network
ResNet-34	34	24.19%	7.04%	33	7,1,3,5	Forward skip connection
ResNet-152	152	21.43%	5.71%	151	7,1,3,5	Deepest network
MobileNet	28	29.4%(v1) 28%(v2) 24.8%(v3)		14	3,1	Tailorable

Table 4: Performance comparison between AlexNet [25, 26], VGG16 [49], GoogLeNet [50], ResNet [13] and MobileNet [14–16]

Classification accuracy

The evaluation criteria used for classification accuracy are P-R diagram, AP and Weighted F1 score. Firstly, I used the five trained models to predict the data in the validation set and plot the P-R curve shown in figure 30. From the figure, the curves of the five models intersect at different nodes, and there is no obvious winner among the models. It is possible to see which model is more dominant in different situations. For example, for high-threshold applications that value precision more, GoogLeNet outperformed the other four models, while ResNet-34 was

the worst performer. ResNet-34 and VGG16 are better suited to applications where a balance of accuracy and recall is desired. For our project, both precision and recall are important evaluation criteria. To further obtain a more comprehensive assessment of the overall performance of the model, I used the weighted F1 scores and the area of the P-R curve (AP) as shown in Table 5 for further analysis.

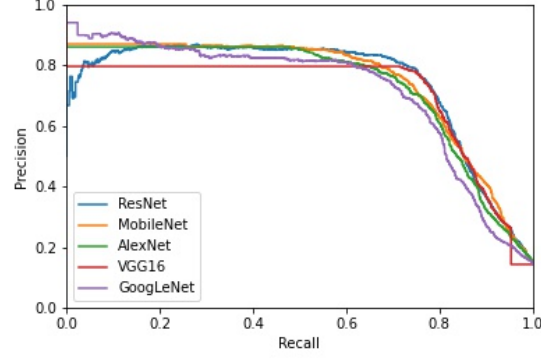


Figure 30: Five Models PR Curve

According to the AP value in Table 5, ResNet-34 has the largest AP value of 0.51, larger than the other four models, with MobileNet in second (AP = 0.49), followed by AlexNet (AP = 0.48). Based on the weighted f1 scores in Table 5, ResNet-34 is still in first place (weighted f1 score = 0.75). VGG 16 is slightly below ResNet-34 (weighted f1 score = 0.743). The other three do not perform so well.

Therefore, at this stage, I can conclude that ResNet-34 is the best performing of all the models. MobileNet and VGG 16 are slightly below ResNet-34.

Classification speed

The table 5 illustrates the FPS of each model, which I obtained in the same hardware environment. As can be seen from the table, AlexNet is the fastest of the five models, achieving 106.016 frames per second. This is followed by MobileNet (FPS = 78.246fps). GoogleNet and ResNet-34 have an FPS of 69.693fps and 61.358fps respectively. the last is VGG16 with only 19.658 frames per second.

	Average Precision (AP)	Weighted F1 score	Frames Per Second
AlexNet	0.48	0.718	106.016
VGG16	0.46	0.743	19.658
GoogLeNet	0.45	0.689	69.693
ResNet-34	0.51	0.750	61.358
MobileNet	0.49	0.722	78.246

Table 5: 5 Models Average Precision, Weighted F1 score and Frames Per Second

Model selection conclusion

The combined analysis of classification accuracy and classification speed concluded that ResNet-34 excelled in classification accuracy, but not in classification speed. However, AlexNet excelled in classification speed but could not achieve the best classification accuracy. In our project, the model runs on a Raspberry PI, so there are hardware limitations in the size of the model

and the speed of classification. Classification on a Raspberry PI is much slower than on a computer. Although there are gaps in the running speed of the resnet-34, MobileNet, AlexNet and GoogLeNet models, they all meet the needs of our application. The VGG16, which has the lowest classification speed, is not. Also according to section ?? VGG16 and AlexNet are 1.5G and 652M in size, which also exceeds the size limit of the Raspberry PI model. Even if the hardware could be upgraded, they do not have the advantage of being smaller than the other models. So I ruled out the VGG16 and AlexNet applications. Of the remaining resnet-34, MobileNet and GoogLeNet models, resnet-34 has the best classification accuracy, followed by MobileNet. googLeNet has much lower value than the first two. Finally, we chose MobileNet as the binary classification model and GoogLeNet as the multi-classification model.

8.5.3 Mask Binary Classification Model Performance Evaluation

Figure 31 shows the confusion matrix of mask binary classification model. The dark areas are concentrated on the diagonal and the accuracy of this model is 0.964. Our project can well complete the identification of masks.

8.5.4 Sunglasses Binary Classification Model Performance Evaluation

Figure 32 shows the confusion matrix of sunglasses binary classification model. The dark areas are concentrated on the diagonal, only a few are in other areas and the accuracy of this model is 0.83. Sunglasses may not be as accurate as masks because they are not as well-defined as masks, and they can also be affected by the way glasses are worn. However, sunglasses are only an auxiliary function of our project, which can assist in the fatigue detection of Yushi. Therefore, the accuracy has already achieved our expected goal.

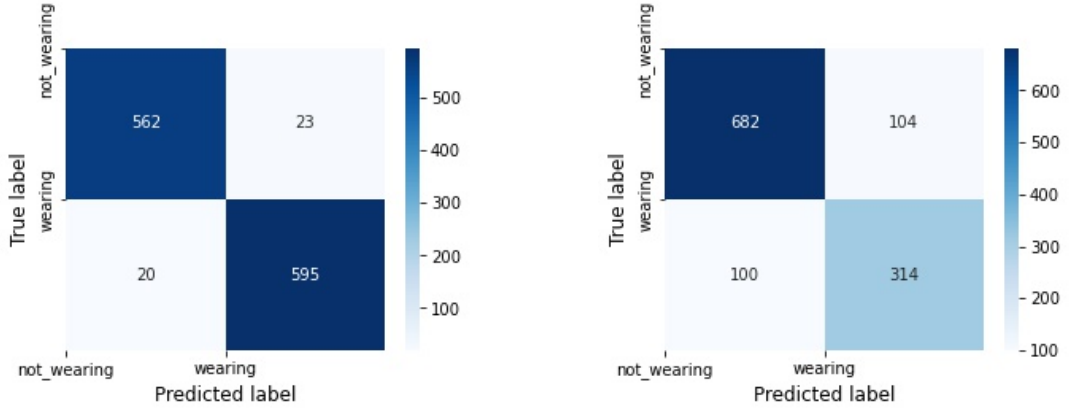


Figure 31: Confusion Matrix of Mask Binary Classification Model

Figure 32: Confusion Matrix of Sunglasses Binary Classification Model

8.5.5 Multiple classification model performance evaluation

Table 6 shows the precision, recall and F1-Score of the ResNet-34 model for each category, as well as the overall accuracy and weighted mean. According to the table, except for the irrelevant categories (looking away, smoking, touch face or head, tired), which were poorly classified due to the small sample size, the three categories that were really needed (no motion, eating or drinking, Talking on the phone) were classified well. The F1 scores for the three were 0.7888, 0.8799 and 0.7876 respectively. In the performance comparison table 4, the Top-1 for

ResNet-34 was 24.19%, that is, accuracy equals to 0.758. In our project, ResNet-34 achieved a classification accuracy of 0.7621, which exceeded the value in the literature review and achieved the classification we expected.

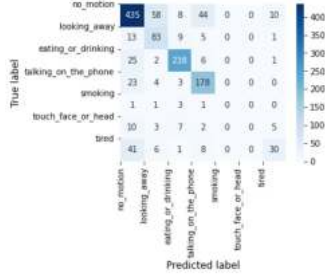


Figure 33: Confusion Matrix of Multiple Classification Model

Table 6: ResNet-34 classification accuracy

	precision	recall	f1-score	support
no_motion	0.7938	0.7838	0.7888	555
eating_or_drinking	0.8848	0.8750	0.8799	272
talking_on_the_phone	0.7295	0.8558	0.7876	208
looking_away	0.5287	0.7477	0.6194	111
smoking	0.0000	0.0000	0.0000	6
touch_face_or_head	0.0000	0.0000	0.0000	27
tired	0.6383	0.3488	0.4511	86
accuracy			0.7621	1265
weighted avg	0.7482	0.7621	0.7498	1265

9 Facial Monitoring, User Interface and Cloud Design - Yushi Wang

9.1 Declaration of Authorship

I hereby declare that the following statements are true and valid:

- 1) I fully understand the definition of plagiarism and its consequences.
- 2) I promise that no part of the project is suspected of plagiarism and that it is all my original work.
- 3) All material for this project, including code and articles, does not contain anything that has been published or written by anyone unless I explicitly state that we are referring to someone else's project or work.
- 4) I own the full copyright of this project including the code and article and do not use without permission.
- 5) The thesis presented is the result of my independent research under the guidance of our supervisor.
- 6) The paper is written by me and I am responsible for what I wrote.

9.2 Description of all the work completed

9.3 Fatigue model

Fatigued driving is a phenomenon where a driver's physiological and psychological functions become dysfunctional after a long period of continuous driving, resulting in dangerous driving behaviour which is one of the main causes of traffic accidents. Poor or insufficient sleep and prolonged driving of vehicles make it happen, which affects the driver's attention, thinking, judgement, and decision-making in driving. Therefore, fatigue detection is an important part of dangerous behaviour monitoring.

9.3.1 PERCLOS Model

In our final model, we used face detection techniques based on human behavioural characteristics image processing determine the driving status mainly by detecting and analysing facial features. As stated in the literature review, we discarded the models based on physiological

signals and on vehicle motion parameters.

Based on this, we design a system for driver fatigue detection algorithms based on the widely used PERCLOS algorithm. PERCLOS stands for the percentage of eyelid closure over the pupil over time, the average of which is the percentage of time the eye is closed per unit of time [52]. It defines the percentage of time that the eye is closed between 80% and 100% of the time over a period of time. PERCLOS was first proposed by Walt Wierwille [52] and has been shown experimentally to be small, low cost and real-time which has been validated as a viable non-contact fatigue assessment method [27].

In my model, different from the standard PERCLOS model only contain the eye closing model, I add the weights of yawning, the selection of weights is given in 9.7. I will continuously time track the running average of PERCLOS measurements which contains the tracker of the driver's eye closure and mouth opening size in real-time, giving feedback and calculating the fatigue level over a certain period of time.

9.4 Extraction of Facial Feature Points

Face landmark detection, or face alignment, which is based on face detection, means that features on a person's face, such as the corners of the mouth and eyes, are localised. The ultimate goal of face alignment is to locate the exact shape of a face on a known face box and shape is made up of feature points, i.e. locations on the face where there are features.

9.4.1 The Open Source Library - Dlib

I used Dlib [22](<http://dlib.net/>) in our model for face feature point extraction, which is a Python or C++ based open-source toolkit algorithm. It simplifies the development of software for a lot of machine learning applications. For example, Dlib-based applications and the open-source face recognition library *face_recognition* effectively help to locate and recognise faces.

One of the reasons we chose Dlib is high quality and widely compatible. It does not rely on third-party libraries and requires no installation or configuration and can be used directly on Windows, Mac OS, and Linux. In particular, it is very easy to install on the Raspberry Pi which is the ideal hardware for implementing our local server. Dlib library makes it very easy for us to port the code to Raspberry Pi.

The other reason is that it runs very quickly and accurately. Ren et al [40] proposed that due to the 68 landmarks method using extraction and regression of local binary features, the computational cost is very low. In our model, facial feature points tracking is used to monitor the opening and closing of the eyes and mouth in fatigue detection. It is more important for our model to make fast judgements to continuously tracking the level of fatigue over a period of time.

9.4.2 Extraction Method in Dlib

For facial feature prediction, the Dlib library provides a key point predictor called *shape_predictor_68_face_landmarks.dat* which can be called directly by method *dlib.shape_predictor()* method to mark face feature. Then use method *predictor()* with the input of a numpy ndarray containing an 8-bit greyscale or RGB image and a bounding box of internal shape predictions to locate the face feature and return the locations of 68 face feature landmarks.

9.5 Real-Time Eye Blink and Mouth Yawn Detection

I wish to propose an algorithm for the real-time detection of blinks and yawns in video sequences from a standard camera. As mentioned in the introduction to the fatigue model above, our fatigue model is based on the percentage of time spent blinking and yawning frame (i.e. fatigue frame) over a period of time. Therefore, I will estimate the level of eye opening and mouth opening based on the obtained facial feature points.

9.5.1 EAR Model for Blinking Detection

This part used the algorithm proposed by Tereza [5] et al. estimates the coordinate positions and calculates the eye aspect ratio (EAR) based on the positions, i.e. characterising the level of eye openness in each frame. Tereza [5] et al. proved that EAR keypoint detectors trained on field datasets showed good robustness to camera head orientation, different illumination and facial performance.

For each frame, the key points of the human eye are detected and processed. The formula for calculating the human EAR is defined as follows:

$$EAR = \frac{||p_2 - p_6|| + ||p_3 - p_5||}{2||p_1 - p_4||} \quad (5)$$

where p_1, \dots, p_6 are the key point locations given by Dlib before, as shown in Figure 34.

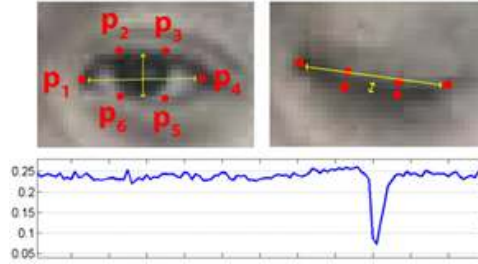


Figure 34: EAR auto-detection in the aspect ratio equation 34 for eyes with open and closed eyes, image from paper [5]

In the theory, the value of EAR is very small or even zero when the eyes are closed; the value of EAR with open eyes varies depending on the individual eye threshold, but is essentially constant. The advantage of the EAR method is that it is insensitive to the body and head posture of the person: the aspect ratio when the eyes are open varies very little between individuals and has little effect on the uniform scaling of the image and the in-plane rotation of the face. As both eyes are blinking simultaneously, I use the average of the EAR of both eyes as one of the conditions for determining fatigue.

9.5.2 Model for Yawning Detection

For the threshold determination of mouth opening, I used the same aspect ratio judgement as Tereza [5] for the eye. The formula is the same, but the selection of feature points for the mouth is different from that for the eye, as shown in Figure 35.

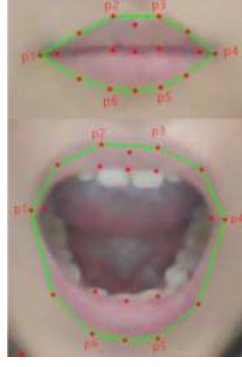


Figure 35: Calculation of the open and closed mouth schematic. The red dots are the points marked by Dlib and we only use six of them to calculate by formula 5

9.6 Discussion - Parameter selection

This section is used to present the method and results of my choice of parameters in the fatigue model.

9.6.1 Selection of eye parameters in the reference paper

The final aspect ratio threshold given by Tereza [5] with the test set ZJU, consisting of 80 short clips from 20 subjects, given the result = 0.2. However, for the mouth test set I did not find a publicly available suitable result and dataset, so I decided to find 20 testers to help me with the selection of the mouth threshold (as the test set in the article was 20 people, I considered 20 to be an appropriate number of people to test the threshold).

9.6.2 Test of mouth data

This test is used to determine the threshold of area ration of moth opening. In the calculation of the test mouth data, perform 3 tests for each tester: five yawns with the mouth wide open(blue line), five yawns with the mouth small open(orange line) and normal speaking(grey line). During these processes, the mouth aspect ratio values were tracked. Two of the tester mouth tracking data is shown as an example in Fig. 36 and Fig.37 below.

From most of the mouth data tracking results, it appears that the majority of subjects opened their mouths much wider when yawning (blue line) than they normally would have spoken (grey line) (as shown on in Fig. 36). In some cases, there was little difference between the magnitude of mouth opening during yawning and speech (as shown in the Fig. 37), however, testers maintained their mouths open significantly longer when yawning than when speaking.

Extract data from all subjects with open mouths and calculate averages, the results are shown in the table 7. The threshold I determined in our project for mouth opening is 0.6 because of all the testers yawned threshold larger than 0.6, and only one person spoke threshold over 0.6.

9.6.3 Frame threshold

Frame threshold means the duration of the frames of the fatigue feature, which is to avoid misjudgement of the tracker and differentiation of the normal driving state. As mentioned before, the determination of yawning can be controlled by the threshold of frames: if the user opens their mouth (threshold greater than 0.6) for only 5 frames or less, then the user is talking, and

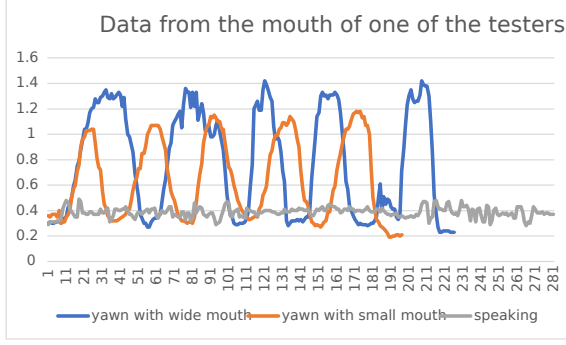


Figure 36: One of the clear Visualisation mouth data.

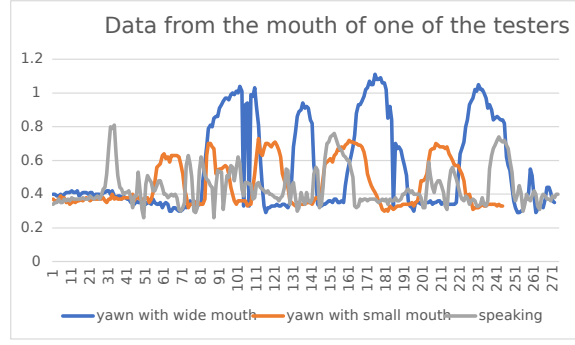


Figure 37: One of the unclear Visualisation mouth data.

vice versa for yawning. The ‘5 frames’ threshold here should be determined by the amount of computation and performance of different hardware devices (Based on my testing of frame per second (fps), the Raspberry Pi usually performs much worse than the PC, so this threshold should be smaller on the Raspberry Pi than on the PC).

For the other frame threshold, the number of frames with eyes closed, my test on the PC resulted in 2. That is, I considered it a valid eye closure only if the EAR value was less than 0.2 for more than 2 frames consistently. Otherwise, I think it’s a tracker error issue.

Table 7: Values and averages for twenty testers in seven tests

Tester	1	2	...	19	20	average
Yawning with wide mouth	1.33	0.86	...	1.1	1.2	1.094737
Yawning with small mouth	1.08	0.75	...	0.83	0.85	0.840526
Speaking	0.4	0.55	...	0.44	0.44	0.525789
Fatigue(eye=0.9;mouth=0.1)	0.642	0.693	...	0.828	0.884	0.802337
Awake(eye=0.9;mouth=0.1)	0.2297	0.419	...	0.214	0.121	0.334226
Fatigue(eye=0.8;mouth=0.2)	0.574	0.632	...	0.776	0.725	0.621526
Awake(eye=0.8;mouth=0.2)	0.079	0.0856	...	0.174	0.1	0.202353

9.6.4 Fatigue parameters

For the degree of fatigue and the weighting of closed eyes and open mouth in the fatigue model, I tested eye weight = 0.9, mouth weight = 0.1 and eye weight = 0.8, mouth weight = 0.2 with 20 testers. The results table 7 is shown above: The difference between mean fatigue and wakefulness was larger at higher eye weights than the other one ($0.46811 < 0.41918$). It can also be seen from the results table 7 that the values for both fatigue and wakefulness are high for high eye weights, which is in line with what we would expect from the results: the eyes are more important than the mouth in the detection of fatigue. Therefore, we chose the model with higher eye weights (eye=0.9; mouth=0.1).

For the threshold of fatigue, I choose 0.5 because no one is awake at an average of 0.5. After determining the parameters I re-invited several testers to carry out fatigue tests and found that 0.5 largely met the requirements for fatigue determination. However, I also found that the fatigue detection was biased when the driver turned his head frequently or when there was a lot of eye movement. For this case, I think it is a factor of Dlib not being able to detect faces,

especially eyes, for a long time, and that the ratio of frames with closed eyes is set too high in the above formula. I will explain this in more detail in the section on head-turning below.

9.7 Head Posture

In our project proposal, we planned to use head-down as an alternative criterion for fatigue modelling. Therefore I looked for two currently widely used ways of tracking head position: deep learning and calculation of angles in 3D mode. Then I discuss the necessity and accuracy of head posture in fatigue driving models and giving possible alternative ways.

9.7.1 Two Possible Method

For the first deep learning approach, I tried to find and produce a standard dataset, but I found that this was not achievable during the application of the real system. This is because the real application scenario is a multi-angle head dataset and it is difficult to determine the correct angle.

The second way is practically possible, by calculating the angle at which the current head deviates from the initialised photo. Mallick et al. [36] proposed to rotate the 3D standard model at an angle until the "2D projection" of the model on the "3D feature points". Fortunately, I managed to reproduce the model in the article and calculate the 3D deflection angle of the head with respect to the camera.

9.7.2 Discussion

We have implemented the detection of head movements at different angles before. However, in the course of our tests, we discovered something very important - normal driving also involves turning head behaviour (Look backwards when parking; Look to the left before turning to the left) so that turning head can not be a feature of dangerous behaviour!

Back to the proposal: there are two reasons why we need head monitoring in the system

1. Frequency of head bowing and fatigue monitoring
However, according to the PERCLOS model above, fatigue can be determined by blinking frequency and yawning frequency. There is little experimental evidence that head lowering can be associated with fatigue and therefore head posture is not required for fatigue monitoring.
2. Turning head may be talking to a passenger and it is a dangerous distraction
However, as normal driving cannot force the driver to look forward only. It is difficult to say whether someone is turning their head because they are talking to someone else or whether it is just normal driving behaviour.

9.7.3 Alternative Approach of Head Posture

I propose a new method, just sending a warning message feedback to the driver to look forward, instead of sending a dangerous head posture signal to the cloud. Then becomes much simpler: while the warning message can be many, it does not need to be precise. We don't need the exact angle of the head, but just the driver looking ahead so that the condition for the judgement can be changed to determine whether the detector can see the face, and in particular the eyes, through the camera.

The previous way of posing the head was very accurate and we were able to get a specific head angle. The disadvantage was that it was too complex and unnecessary. The new method tests whether more than half of the total number of frames possessing the eye over a period of time, and if not, sends a warning message back to the driver. The new method solves two problems: one is alerting to head-turning movements in normal driving; the other is whether the camera is correctly positioned. It is simpler and makes more sense.

9.7.4 Solution to Speaking Distraction

Reviewing the issue of drivers being distracted from talking with their passengers, we still haven't fully addressed them. I therefore proposed a new way to monitor drivers turning their heads to talk to passengers, which is called emotion monitoring. If the driver is very happy or very sad (maybe talking to a passenger, maybe just having a mood swing), it is very dangerous.

9.8 Emotion Monitoring

Emotion detection means recognising faces and their corresponding emotions from video. As this is the part I tried to use later on as a replacement for head movements, I did not have enough time and ability to train a complete network for emotion detection myself. So I found a trained emotion detection network based on OpenCV and deep learning and applied it.

The Convolutional Neural Network (CNN) deep learning architecture (<https://github.com/petercunha/Emotion>) used is from Arriaga's paper [3] and scaffolding powered by Keras with Tensorflow. The Pretrained Keras model and much of the OpenCV code provided by GitHub user oarriaga.

This section detects the five basic emotions angry, sad, happy, surprised and neural and uses cv2 to frame them on the screen and use different shades of colour to indicate the degree to which the person is expressing the emotion. The model uses trained hdf5 files for emotion detection. Take frame from the video and the image is exported by the load xml cascade classifier. Then grey out and resize exported image. Multi-scale detection implement by calling *detectMultiScale()* method.

9.9 User Interface

The core part of opening camera through overflow slot function

```
1 ...
2 class CamConfig:
3     def __init__( self ):
4         # set up a timer
5         self.v_timer = QTimer()
6         # open camera by cv2
7         self.cap = cv2.VideoCapture(0)
8         # Set the timer period in milliseconds
9         self.v_timer.start(20)
10        # Connect timer cycle overflow slot function used to display a frame of video
11        self.v_timer.timeout.connect(self.show_pic)
12 ...
```

As the full detection model is too large and the time required to run a frame is too long to do the risky behaviour detection on every frame, we need to control the frame rate. As fatigue

detection is a continuous judgement, a single closing of a person’s eyes or opening of their mouth does not indicate their level of fatigue. Therefore fatigue detection must be continuous, with fatigue detection performed on every monitored frame. But distraction detection and emotion detection can be performed by jumping frames, so I only do these detection every 10 frames. By doing this, I found that the frame rate was approximately 30fps (frames per second) for the part where only fatigue detection was performed and 5fps for the part where all detection was performed (both figures were performed on a PC).

9.10 Cloud Computing System Design

In this section, I discuss the algorithmic implementation of the cloud platform receiving a danger signal from one vehicle and warning surrounding vehicles of the danger (e.g. within 200m of the dangerous vehicle), provided that the cloud platform knows the location of all vehicles in the system. I will give a discussion of the advantages and disadvantages of several possible implementations of the algorithm.

9.10.1 Range Search

One possible way is range search, which is to find a good data structure that can efficiently perform Range queries on collections of objects (points, rectangles, polygons).

The simplest traversal algorithm in range search, which is to go through the distance of all other vehicles in the cloud centre server to the danger vehicle and then compare. This approach is possible, but when the system is too large (if it can be spread over a city), the complexity of traversing all the vehicles in the system is clearly too high and unconscionable ($O(n^2)$).

We want to find some method of range search to improve the complexity, however, unlike finding relatively static objects, such as finding a nearby building, the coordinates of a moving vehicle are updated in real-time. Therefore using the KD-tree or Rtree, commonly used in general range finding, to update once round to build a new tree does not solve the time complexity problem.

9.10.2 Distributed thinking

The reason for the high complexity of the range search is mainly due to the high volume of requests to the cloud resulting in the ‘central node’ taking on too much computational work. The central idea of a distributed system is decentralisation, i.e. the distribution of the computational load from the ‘central node’ to the ‘vehicle nodes’.

In our project, we change our strategy to when the cloud platform receives the coordinates of a dangerous vehicle, it distributes the positions of dangerous cars instead of warning messages to vehicles in the same ‘area’, and the vehicles themselves calculate whether they are within 200 metres of each other. The calculation of distances is distributed to the local side and the cloud platform is only responsible for calculating which ‘area’ the vehicle is in.

The definition of ‘area’ emerged to reduce excessive and non-essential communication processes. Before broadcasting the coordinates in the cloud, we need to remove some vehicles that are quite far away from the volume of dangerous vehicles - that is, to zone the vehicles in the system in real-time. We divide an area into areas that can overlap but do not contain areas, calculate which area or areas each vehicle falls into, and broadcast the location of dangerous vehicles within the area. The method is an iterative algorithm with time complexity of $O(n)$.

9.10.3 Why cloud servers

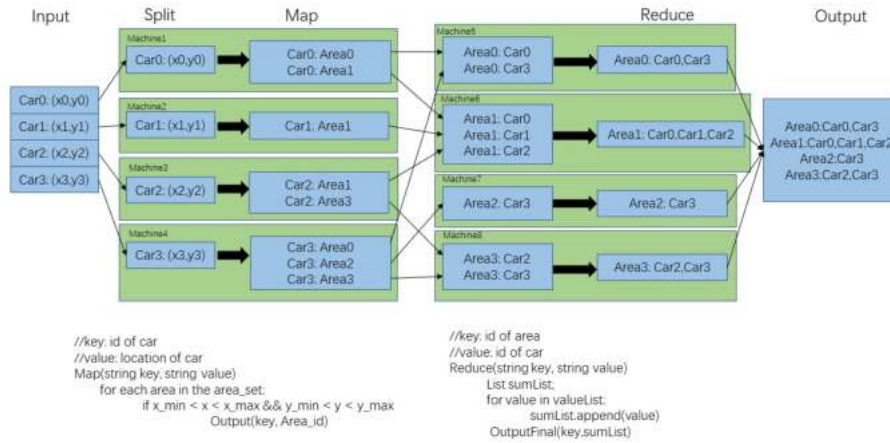
The project needed to be implemented on a cloud server instead of a local server for three reasons.

1. The geographical coverage of our system was expected to be scalable, for example, from covering only the Chaoyang district of Beijing to covering all cities in China. It is not desirable to keep setting up local servers in each region as it expands.
2. The system is readily expandable as users are added. Then as users grow, a system that can only serve a small number of users is undesirable and we will have to constantly update the size of our servers. However, with AliCloud, the scalability of cloud servers can be easily achieved.
3. Cars have a much higher traffic volume during the day than at night within the time frame of a day. Keeping the computing volume high during the day is wasteful in terms of cost, so we want the servers to be scalable throughout the day.

9.10.4 Map-Reduce

In order to achieve parallel operations on large datasets and solve the complexity of $O(n)$, we choose the map-reduce method to optimise the traversal model. The model means specifying a map function to map a set of key-value pairs into a new set of key-value pairs and specifying a concurrent reduce function to ensure that each of all mapped key-value pairs share the same set of keys. The final model example is shown below.

Mapping key-value pair in our model is key: id of car; value: location of car, with highly parallel calculating each vehicle in which area and creating a new list to keep. Reducing key-value pair is key: id of area; value: id of car, output the merging of vehicles included in each area.



10 Overall Results, Discussions

Figure 38 shows the whole system operation, including two Raspberry Pis, two 4G extension modules, two speakers, and one screen. Because of the epidemic, screens suitable for Raspberry

Pi size could not arrive by delivery.

Below we show the running result of our system (Figure 38). In our user interface (see Figure ??), the title of the interface is "Dangerous driving behavior". When the interface is opened the camera is turned off and the feedback screen prompts you to turn on the camera as shown above. Clicking on the 'start' button in the menu bar to turn on the camera and the monitoring video will show via the feedback screen. The feedback screen is in 3:2 ratio, which matches the ratio of the external camera we purchased for our Raspberry Pi, so that the faces displayed are not exaggerated and distorted. Unlike the normal way for opening the camera and displaying it via the pyside2, I set up one action trigger called "open" and in class CamConfig, the video frame will display by connecting the overflow slot function of the timer cycle. The video frame is from the camera opened via cv2 and is displayed on the feedback screen after the frame has been processed (during the monitoring, the face feature and expressions will be marked on the frame via cv2).

In the upper right hand corner is the section of the window for feedback on dangerous movements. The fatigue label is the result of the test fatigue detection test: 'awake'/'tired'/'unknow', with the last two tabs in red to warning. The blinking and yawning labels show the number of times the tester blinked and yawned. The distraction label shows the results of the tester passing the dangerous action test: 'no motion', 'looking away', 'eating or drinking', 'talking on the phone', 'smoking', 'touch face or head', The seatbelt and mask labels are the condition of the testers' wearing of seat belts and masks: 'yes'/'no'.

The bottom right section is the text browser section, which will indicate the real-time status of the detection process. For example, shown 'The computer is turning on its camera...' when the camera is turned on, 'Failed to open camera' when it fails, and 'Load video successfully and start dangerous behaviour' when it succeeds. At the start of each round of dangerous behaviour detection, it shows 'Start a new round of monitoring...'. Also, after each round of detection, indicates the current driving status of the tester and when the camera cannot find the tester's face, the message, it shows 'Can't detect your face, please adjust camera position or look forward '.

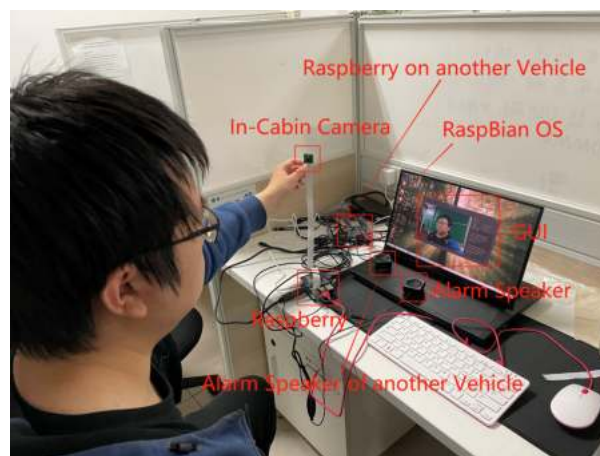


Figure 38: Overall system operation

Different from the original plan, the communication between devices on different vehicles should be achieved by using point-to-point communication based on techniques such as ZigBee, we used 4G plus cloud server to achieve the same task, which is a trade-off between latency and mobility.



Figure 39: Drink Detection Figure 40: Mask Detection Figure 41: Phone Detection

Figure 42: Three Kinds of Manual Distraction Detection

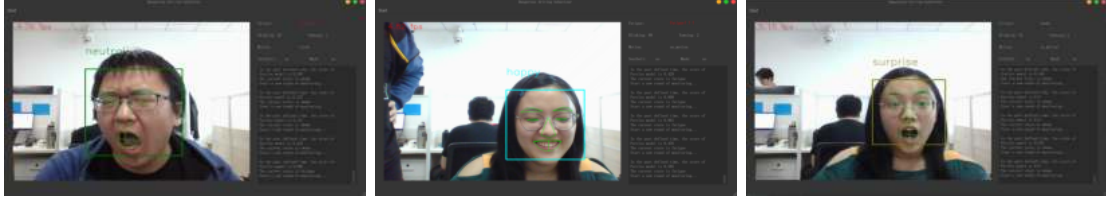


Figure 43: Fatigue Detection Figure 44: Happy Emotion Figure 45: Surprise Emotion

Figure 46: Fatigue Detection and Two Emotions

Due to the bandwidth limitation of the PC, performance is significantly degraded by the upload of GPS coordinates of a large number of vehicles, and is not related to the architectural design of the model. We move the generation of simulated GPS coordinates part on cloud. Since the communication between real devices is quite different from the simulated one, we implemented the common part of them i.e., local and on-cloud vehicle area division, and finally chose the on-cloud implementation based on the performance performance.

Object detection models are more difficult than our expectation. We wanted to use a detection model to achieve the location of the driver, including head and hands, the steering wheel, the seat belt, mobile phone, as well as food and drinks. However, such tasks overpowered the model, and we have to change our plan to let it focus on less objects.

Active learning is a newly introduced method, we did not planed for it at the beginning, but it turns out to be successful, It helps us greatly reduce the cost of manual labeling and improve the accuracy of detection. In the same time it lead to a new possibility that combine active learning with enhance learning, so that the system can training of these models continuously on the cloud, when the data is collected daily, and updates the models from time to time.

The fatigue model was successfully implemented by the extraction of facial points. However, as there is no complete and mature fatigue data-set available online, most of the parameters test were done through the tests of our classmates. Therefore, there may be problems with compatibility for people from different countries.

The head posture component of the fatigue model is replaced by monitoring head emergence and emotion detection. Emotion detection only appears in the user interface as a reference and is not used as a criterion for dangerous movements because of the lack of reference data for this standard.

We think it is very creative to introduce the map-reduce cloud computing framework to our task. The framework is so powerful that it is able to tracing the grouping thousands of vehicles with limited latency. Therefore, we might even consider adding more functions to the cloud machines as they are so much more powerful (than local devices).

One importance issue we have not considered is the privacy of the users. For a mature design, all the user’s information, including the image of the cabin, their face ID, current location, etc., should all be carefully protected. We should also consider the way to protect people’s privacy when we work on collecting data for active learning.

11 Conclusion and Future work

In conclusion, we achieved most of the targets in our original design. We expected our model to be able to detect the situation that the driver is fatigue, or is doing dangerous behaviours; we trained our own models for behaviour detection, and introduced model for fatigue detection, and they all work. We successfully build small devices, capable to be carried within a vehicle, and is able to run the whole detection software with GUI, giving warning alarms to the drivers within the car, or within a nearby region. Additionally, our system is simple and low-cost, but is still able to manage relatively complex tasks.

However, there are still a lot of things we have not achieve because of the limit of time, we conclude them to be the future work, and we would really love to have a discussion about them:

The cellular network can cause more latency than point-to-point connections. Additionally, when the vehicle leaves the signal region of cellular towers, the communication no-longer works. Based on these issues, looking for a hybrid structure, combining the cellular system and a mobile, self-organized point-to-point communication method will be our future work. We hope the system is able to keep both latency, mobility, and robustness.

For our detection and classification models, we have several future plans. First is to transfer current dynamic PyTorch models into compiled and settled versions, with frameworks such as TensorFlow Lite [32], which is likely to speed up the process for the code is well compiled with optimized algorithm.

Another improvement is to use fast one-stage algorithms like YOLO, to replace the current two-stage models; or to find a body-outlier specialized model to do the task. This would significantly speed up the prediction, and the latter method might also help to increase accuracy.

It is also worth to spend time on designing a new model which is able to combine the work of both detection and classification. Currently, we came out the idea of using multi-label classifier, but this combined model should work in a even higher efficiency, so that the speed of the process can improve further.

We also plan to combine active learning with enhanced learning. In this way the daily updated data on the cloud can be continuously analysed and learned, gradually enhancing the detection ability of our models. We believe that continuous improvement and updating is essential for a detection system.

For the fatigue model, the main future work is to add the effect of head posture on the in

fatigue model and to find and test different national human datasets to obtain a higher degree of usefulness. Another future work is to increase the impact of emotion detection in dangerous driving, rather than just displaying it on the user interface.

References

- [1] Maysoon F Abulkhair, Hesham A Salman, and Lamiaa F Ibrahim. Using mobile platform to detect and alerts driver fatigue. *International Journal of Computer Applications*, 123(8), 2015. 3
- [2] Eugene Aidman, Carolyn Chadunow, Kayla Johnson, and John Reece. Real-time driver drowsiness feedback improves driver alertness and self-reported driving performance. *Accident Analysis & Prevention*, 81:8–13, 2015. 2
- [3] Octavio Arriaga, Matias Valdenegro-Toro, and Paul Plöger. Real-time convolutional neural networks for emotion and gender classification. *arXiv preprint arXiv:1710.07557*, 2017. 9.8
- [4] Oswald Campesato. *Artificial Intelligence, Machine Learning, and Deep Learning*. Stylus Publishing, LLC, 2020. 3, 7.3
- [5] Jan Cech and Tereza Soukupova. Real-time eye blink detection using facial landmarks. *Cent. Mach. Perception, Dep. Cybern. Fac. Electr. Eng. Czech Tech. Univ. Prague*, pages 1–8, 2016. 9.5.1, 34, 9.5.2, 9.6.1
- [6] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014. 8.3
- [7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 8.3
- [8] Sai Vikas Desai, Akshay L Chandra, Wei Guo, Seishi Ninomiya, and Vineeth N Balasubramanian. An adaptive supervision framework for active learning in object detection. *arXiv preprint arXiv:1908.02454*, 2019. 8.4.3
- [9] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 8.3
- [10] Mark Everingham, Andrew Zisserman, Christopher KI Williams, Luc Van Gool, Moray Allan, Christopher M Bishop, Olivier Chapelle, Navneet Dalal, Thomas Deselaers, Gyuri Dorkó, et al. The pascal visual object classes challenge 2007 (voc2007) results. 2008. 3
- [11] Steven Gold, Anand Rangarajan, et al. Softmax to softassign: Neural network algorithms for combinatorial optimization. *Journal of Artificial Neural Networks*, 2(4):381–399, 1996. 7.5.1
- [12] Elmar Haussmann, Michele Fenzi, Kashyap Chitta, Jan Ivanecy, Hanson Xu, Donna Roy, Akshita Mittel, Nicolas Koumchatzky, Clement Farabet, and Jose M Alvarez. Scalable active learning for object detection. In *2020 IEEE intelligent vehicles symposium (iv)*, pages 1430–1435. IEEE, 2020. 8.4.3

- [13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3, 7.4.4, 8.3, 8.5, 4
- [14] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324, 2019. 4
- [15] Andrew Howard, Andrey Zhmoginov, Liang-Chieh Chen, Mark Sandler, and Menglong Zhu. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. 2018. 4
- [16] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 3, 7.4.4, 8.5, 4
- [17] Sang-Joong Jung, Heung-Sub Shin, and Wan-Young Chung. Driver fatigue and drowsiness monitoring system with embedded electrocardiogram sensor on steering wheel. *IET Intelligent Transport Systems*, 8(1):43–50, 2014. 3
- [18] Guoliang Kang, Xuanyi Dong, Liang Zheng, and Yi Yang. Patchshuffle regularization. *arXiv preprint arXiv:1707.07103*, 2017. 8.3
- [19] Chieh-Chi Kao, Teng-Yok Lee, Pradeep Sen, and Ming-Yu Liu. Localization-aware active learning for object detection. In *Asian Conference on Computer Vision*, pages 506–522. Springer, 2018. 8.4.3
- [20] Sinan Kaplan, Mehmet Amac Guvensan, Ali Gokhan Yavuz, and Yasin Karalurt. Driver behavior analysis for safe driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 16(6):3017–3032, 2015. 3
- [21] Alexey Kashevnik, Roman Shchedrin, Christian Kaiser, and Alexander Stocker. Driver distraction detection methods: A literature review and framework. *IEEE Access*, 9:60063–60076, 2021. 3
- [22] Davis E King. Dlib-ml: A machine learning toolkit. *The Journal of Machine Learning Research*, 10:1755–1758, 2009. 9.4.1
- [23] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 7.5.4
- [24] Simon Kornblith, Jonathon Shlens, and Quoc V. Le. Do better imagenet models transfer better? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 7.5.2
- [25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012. 3, 7.4.4, 7.5.2, 8.5, 4
- [26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional. *Neural Netw*, 60:84–90, 2017. 4

- [27] Liying Lang and Haoxiang Qi. The study of driver fatigue monitor algorithm combined perclos and aecs. In *2008 International Conference on Computer Science and Software Engineering*, volume 1, pages 349–352. IEEE, 2008. 9.3.1
- [28] Shinho Lee, Hyeonwoo Kim, Dong-kweon Hong, and Hongtaek Ju. Correlation analysis of mqtt loss and delay according to qos level. In *The International Conference on Information Networking 2013 (ICOIN)*, pages 714–717. IEEE, 2013. 7
- [29] Yu Lei and Jian Wu. Study of applying zigbee technology into foward collision warning system (fcws) under low-speed circumstance. In *2016 25th Wireless and Optical Communication Conference (WOCC)*, pages 1–4. IEEE, 2016. 6.4.1
- [30] David D Lewis and William A Gale. A sequential algorithm for training text classifiers. In *SIGIR’94*, pages 3–12. Springer, 1994. 8.4.2
- [31] Mu Li, Tong Zhang, Yuqiang Chen, and Alexander J Smola. Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 661–670, 2014. 3, 7.3, 7.5.3, 7.5.4
- [32] Shuangfeng Li. Tensorflow lite: On-device machine learning framework. *Journal of Computer Research and Development*, 57(9):1839, 2020. 7.6.1, 11
- [33] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 3, 7.4.2
- [34] Tianchi Liu, Yan Yang, Guang-Bin Huang, Yong Kiang Yeo, and Zhiping Lin. Driver distraction detection using semi-supervised machine learning. *IEEE transactions on intelligent transportation systems*, 17(4):1108–1120, 2015. 2
- [35] Lukas Malina, Gautam Srivastava, Petr Dzurenda, Jan Hajny, and Radek Fujdiak. A secure publish/subscribe protocol for internet of things. In *Proceedings of the 14th international conference on availability, reliability and security*, pages 1–10, 2019. 6
- [36] Satya Mallick. Head pose estimation using opencv and dlib, 2016. 9.7.1
- [37] Mélanie Née, Benjamin Contrand, Ludivine Orriols, Cédric Gil-Jardiné, Cédric Galéra, and Emmanuel Lagarde. Road safety and distraction, results from a responsibility case-control study among a sample of road users interviewed at the emergency room. *Accident Analysis & Prevention*, 122:19–24, 2019. 3
- [38] Sagarkumar Patel, Vatsal Shah, and Maharshi Kansara. Comparative study of 2g, 3g and 4g. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 3(3):1962–1964, 2018. 6.4.1
- [39] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 3, 7.4.2
- [40] Shaoqing Ren, Xudong Cao, Yichen Wei, and Jian Sun. Face alignment at 3000 fps via regressing local binary features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1685–1692, 2014. 9.4.1

- [41] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015. 3, 7.4.2
- [42] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017. 8.4.1
- [43] Anwesha Sengupta, Sibsammbhu Kar, and Aurobinda Routray. Study of loss of alertness and driver fatigue using visibility graph synchronization. In *Innovative Research in Attention Modeling and Computer Vision Applications*, pages 171–193. IGI Global, 2016. 3
- [44] Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948. 8.4.2
- [45] Adnan Shaout and Brennan Crispin. Using the mqtt protocol in real time for synchronizing iot device state. *Int. Arab J. Inf. Technol.*, 15(3A):515–521, 2018. 6.4.1
- [46] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019. 8.3
- [47] Gulbadan Sikander and Shahzad Anwar. Driver fatigue detection systems: A review. *IEEE Transactions on Intelligent Transportation Systems*, 20(6):2339–2352, 2018. 2, 3
- [48] Michael Simon, Eike A Schmidt, Wilhelm E Kincses, Martin Fritzsche, Andreas Bruns, Claus Aufmuth, Martin Bogdan, Wolfgang Rosenstiel, and Michael Schrauf. Eeg alpha spindle measures as indicators of driver fatigue under real traffic conditions. *Clinical Neurophysiology*, 122(6):1168–1178, 2011. 3
- [49] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 3, 7.4.4, 8.5, 4
- [50] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 3, 7.4.4, 8.5, 4
- [51] Xinxing Tang, Pengfei Zhou, and Ping Wang. Real-time image-based driver fatigue detection and monitoring system for monitoring driver vigilance. In *2016 35th Chinese Control Conference (CCC)*, pages 4188–4193. IEEE, 2016. 3
- [52] Udo Trutschel, Bill Sirois, David Sommer, Martin Golz, and Dave Edwards. Perclos: An alertness measure of the past. In *Proceedings of the Sixth International Driving Symposium on Human Factors in Driver Assessment, Training and Vehicle Design*, pages 172–179, 2011. 9.3.1
- [53] Deniz Tuncel, A Dizibuyuk, and M Kemal Kiymik. Time frequency based coherence analysis between eeg and emg activities in fatigue duration. *Journal of medical systems*, 34(2):131–138, 2010. 3
- [54] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):1–40, 2016. 3, 7.5.2
- [55] Kun Xia, Jiawen Ni, Yanhong Ye, Po Xu, and Yiming Wang. A real-time monitoring system based on zigbee and 4g communications for photovoltaic generation. *CSEE Journal of Power and Energy Systems*, 6(1):52–63, 2020. 6.4.1

- [56] Muneer Bani Yassein, Mohammed Q Shatnawi, Shadi Aljwarneh, and Razan Al-Hatmi. Internet of things: Survey and open issues of mqtt protocol. In *2017 international conference on engineering & MIS (ICEMIS)*, pages 1–6. Ieee, 2017. 6.4.1
- [57] Weiping Yu, Sijie Zhu, Taojiannan Yang, Chen Chen, and Mengyuan Liu. Consistency-based active learning for object detection. *arXiv preprint arXiv:2103.10374*, 2021. 8.4.3
- [58] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11):3212–3232, 2019. 7.4.2, 7.4.2, 7.4.3