

Final Report

|  |
| --- |
| School of Computing  Faculty of Engineering AND PHYSICAL SCIENCES |

AI-based Driver Status Monitoring Application

Leshan Wang

Submitted in accordance with the requirements for the degree of  
BSc Computer Science

**2022/23**

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

*<As an example>*

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

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

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

(Signature of student)

© 2023 The University of Leeds and Leshan Wang

# Summary

*<Concise statement of the problem you intended to solve and main achievements (no more than one A4 page)>*

This project utilizes image recognition technology, aims to achieve accurate fatigue detection with mobile deployment for maximum portability and compatibility. The lightweight, mobile-friendly target detection framework YOLOv5, and NCNN framework have been implemented for this project.

By using cameras on mobile devices to collect real-time footage, then analysed by convolutional neural network technologies, users will be warned if fatigue behaviours were detected.

Technical wise, training of the detection model followed an open source project, which provides a well-optimized dataset of human facial expressions covering several types of drowsiness behaviours. After training and testing, the model went through ONNX simplifying and fp16 optimization process. In order to run on NCNN framework, the model conversion played an important part during the whole process, followed by the development of android application with NCNN implementation. At the final stage, the model was loaded into the application with multiple features developed alongside to fulfil the purpose of the project.

For the mobile application, it is designed not only to operate detection model properly, but also capable of various user friendly functionalities such as camera switching, CPU/GPU switching, custom dataset uploading, and an embedded fatigue judgement threshold.

The achievements of the project have reached the goals with satisfying results. Fatigue detection is very accurate and responsive, with average 20 frames per second running performance. The Mobile application operates without errors, with clear UI design proves to be user-friendly.

The project might be basic, yet it yields great potential value. Compared to wearable devices detection. which monitor drivers’ physical status, image recognition does not cause any uncomfortable feelings thus no interference for driving, therefore a more suitable and portable solution to ensure a safe driving experience.

Keywords: Fatigue detection, Image Recognition, YOLOv5, NCNN, Mobile,

# Acknowledgements

*<This page should contain any acknowledgements to those who have assisted with your work. Where you have worked as part of a team, you should, where appropriate, reference to any contribution made by others to the project.>*

*Note that it is not acceptable to solicit assistance on ‘proof reading’ which is defined as “the systematic checking and identification of errors in spelling, punctuation, grammar and sentence construction, formatting and layout in the text”; see*

https:://www.leeds.ac.uk/secretariat/documents/proof\_reading\_policy.pdf

Yu Xiuying, my tutor, provides great guidance and advice throughout the project with great patience and responsibility.

Suhedaras, an inspirational developer, her project provides a well-constructed dataset crucial to my project.

Nihui, a well-known master in NCNN and YOLO technology field, provides detailed and wise guidance relate to the topic.

My project won’t be successful without their support. I am very grateful.

Table of Contents

[Summary iii](#_Toc133434851)

[Acknowledgements iv](#_Toc133434852)

[Chapter 1 Introduction and Background Research 1](#_Toc133434853)

[1.1 Introduction 1](#_Toc133434854)

[1.2 Background Research 1](#_Toc133434855)

[1.2.1 OEM Solutions 2](#_Toc133434856)

[1.2.1.1 BMW Active Driving Assistant 2](#_Toc133434857)

[1.2.1.2 Ford Driver Alert 2](#_Toc133434858)

[1.2.1.3 Cadillac Super Cruise System 3](#_Toc133434859)

[1.2.2 Third-party Solutions 4](#_Toc133434860)

[1.2.2.1 DENSO DSM 4](#_Toc133434861)

[1.2.2.2 VIGO Smart Headset 4](#_Toc133434862)

[1.2.3 Individual Solutions 5](#_Toc133434863)

[1.3 Project Details 6](#_Toc133434864)

[1.3.1 Main Focus 6](#_Toc133434865)

[1.3.2 Key points 6](#_Toc133434866)

[1.3.2.1 Requirement analysis 6](#_Toc133434867)

[1.3.2.2 Choosing dataset 7](#_Toc133434868)

[1.3.2.3 Choosing framework 7](#_Toc133434869)

[1.3.2.4 Mobile application development 7](#_Toc133434870)

[1.3.2.5 Evaluations 7](#_Toc133434871)

[1.3.2.6 Additional Features 7](#_Toc133434872)

[1.4 Content Overview 7](#_Toc133434873)

[1.5 Conclusion 8](#_Toc133434874)

[Chapter 2 Methods 9](#_Toc133434875)

[2.1 Requirement Analysis 9](#_Toc133434876)

[2.1.1 Functional Requirements 9](#_Toc133434877)

[2.1.2 Non-functional Requirements 10](#_Toc133434878)

[2.2 System Design 10](#_Toc133434879)

[2.2.1 Overall Architecture 10](#_Toc133434880)

[2.2.2 Module Design Breakdown 12](#_Toc133434881)

[2.2.2.1 User Interface 12](#_Toc133434882)

[2.2.2.2 Detection 13](#_Toc133434883)

[2.3 Implementation 13](#_Toc133434884)

[2.3.1 Development Environment 13](#_Toc133434885)

[2.3.2 Module Explanation 14](#_Toc133434886)

[2.3.2.1 YOLOv5 14](#_Toc133434887)

[2.3.2.2 NCNN Framework 19](#_Toc133434888)

[2.3.2.3 Mobile Application 20](#_Toc133434889)

[2.3.3 Testing 26](#_Toc133434890)

[2.3.4 Version Control 27](#_Toc133434891)

[2.4 Conclusion 27](#_Toc133434892)

[Chapter 3 Results 28](#_Toc133434893)

[3.1 Technical Specification 28](#_Toc133434894)

[3.1.1 Development 28](#_Toc133434895)

[3.1.2 Client 28](#_Toc133434896)

[3.2 Demonstration 28](#_Toc133434897)

[3.2.1 User Interface 29](#_Toc133434898)

[3.2.2 CPU/GPU Switch 29](#_Toc133434899)

[3.2.3 Camera Switch 30](#_Toc133434900)

[3.2.4 Take Photo 30](#_Toc133434901)

[3.2.5 Detection In CPU Mode 30](#_Toc133434902)

[3.2.6 Detection In GPU Mode 31](#_Toc133434903)

[3.2.7 Detection In GPU Mode With Glasses 32](#_Toc133434904)

[Chapter 4 Discussion 33](#_Toc133434905)

[4.1 Evaluation 33](#_Toc133434906)

[4.1.1 Development 33](#_Toc133434907)

[4.1.2 Quality 33](#_Toc133434908)

[4.1.2.1 Functional 33](#_Toc133434909)

[4.1.2.2 Structural 33](#_Toc133434910)

[4.1.3 Accuracy 33](#_Toc133434911)

[4.1.4 Performance 34](#_Toc133434912)

[4.1.4.1 GPU Mode 34](#_Toc133434913)

[4.1.4.2 CPU Mode 34](#_Toc133434914)

[4.1.4.3 Consistence 34](#_Toc133434915)

[4.2 Conclusions 34](#_Toc133434916)

[4.3 Ideas for future work 35](#_Toc133434917)

[List of References 36](#_Toc133434918)

[Appendix A Self-appraisal 38](#_Toc133434919)

[A.1 Critical self-evaluation 38](#_Toc133434920)

[A.1.1 Strengths 38](#_Toc133434921)

[A.1.2 Area of Improvements 38](#_Toc133434922)

[A.2 Personal reﬂection and lessons learned 38](#_Toc133434923)

[A.3 Legal, social, ethical and professional issues 39](#_Toc133434924)

[A.3.1 Legal issues 39](#_Toc133434925)

[A.3.2 Social issues 39](#_Toc133434926)

[A.3.3 Ethical issues 39](#_Toc133434927)

[A.3.4 Professional issues 39](#_Toc133434928)

[Appendix B External Materials 41](#_Toc133434929)

# Chapter 1 Introduction and Background Research

## Introduction

Fatigue is a well-known but always underestimated factor leading to car accidents, a secret killer in almost 30% of cases and a cause of serious injuries in 15% of cases. Long hours of driving, monotonous landscape, sleepless night, shift work - no matter the cause, driver’s fatigue can be dangerous any time, day or night, even on short distance. Fatigue doesn’t always have to lead to falling asleep at the wheel. Even brief moments of deconcentrating can prove fatal. In fact, an estimated 6,400 people died annually in crashes involving drowsy driving, according to the National Sleep Foundation (NSC, 2022). Apart from human lives being endangered, NHTSA estimates fatigue-related crashes resulting in injury or death cost society $109 billion annually, not including property damage (NSC, 2022).

In order to mitigate the risk of fatigue driving, various methods have been established during the past two decades. From encouraging adequate sleeping time to crash avoidance technologies, such as drowsiness alert and lane departure warnings (NSC, 2022). However, for fatigue monitoring, ‘no single measure alone may be sensitive and reliable enough to quantify driver fatigue’(Heitmann et al, 2001). Because alertness is a complex phenomenon, there are several noticeable warning signs, like blinking more than usual, yawning, drifting on or over lane lines, eyes going out of focus or head lowering. Therefore, a multi-parametric approach needs to be developed. Existing technologies with similar approach like DENSO DN-DSM analyse the image of driver’s face to detect certain postures, if the face is deviated from centre, or eyes are closed for a period, warnings will be generated (DENSO, 2022). However, many of these current prevalent methods usually involve installing sophisticated hardware that are complex and not very compatible for all types of vehicles, or come as an OEM ware exclusively to certain types of vehicle.

## 1.2 Background Research

As fatigue-related accidents cannot be overlooked, various methods have been established to mitigate the risks through the recent decade. From factory pre-installed wares to third-party customizable systems, from monitoring vehicle momentum patterns to driver facial behaviours, it is important to research and analyse on the prevailing solutions in order to understand the project better.

### 1.2.1 OEM Solutions

#### 1.2.1.1 BMW Active Driving Assistant

BMW Active Driving Assistant, shown in Figure 1.1, is a collection of driving assistance technologies, which is capable of detecting lane departure, lane change, forward and rear collision, pedestrian detection, blind-spot warning, rear cross-traffic waring, emergency automatic braking and tiredness detection (BWM, 2022). The system is highly integrated and has direct control over the vehicle. Due to the cost and complexity of the system, it only comes with the higher-end models of BWM, such as 5 series and X5. There is also a professional upgrade which includes more advanced features such as adaptive cruise control and active collision avoidance.

A picture containing text, electronics, camera, projector

Description automatically generated

**Figure 1.1** BMW Active Driving Assistant (BWM, 2022)

According to the research conducted by BMW Accident Research Program (ARP), between 2006 and 2017 in United States, across all types of causes of accidents, drowsy-driving related has the lowest distribution of 4 percentage as under the protection of Active Driving Assistant (Bahouth et al, 2018), which undoubtably proves the usefulness of the system.

#### 1.2.1.2 Ford Driver Alert

Straight and simple, initially implemented on Focus series from 2011, the Ford Driver Alert uses only a forward looking camera to monitor vehicle position in the lane when the vehicle speed is greater than approximately 40 mph , thus to calculate a vigilance level of driver (Ford, 2022). If the calculation indicates potential drowsiness, the system will issue a visual and audible warning of two stages: the first stage will issue a temporary warning to advice a rest; if further drowsiness were detected, further warnings will be issued which remains in display until manually cancelled. Based on extensive trials carried out by Ford, the steering patterns which characterise drowsy driving are fairly accurate (Euro NCAP, 2011). However, the system is limited to issue warnings; it is not able to control vehicle behaviours under emergency conditions, even as an embedded system.

#### 1.2.1.3 Cadillac Super Cruise System

The Cadillac Super Cruise System uses FOVIO vison technology, which utilizes an infrared camara to accurately monitor and determine driver’s attention state (Cadillac, 2023). Head orientation and eyelid movements under a full range of time and lighting conditions are taken into measurements. A total three stages of alert might be issued: If the system detects fatigue behaviours for the first time, the light-bar on steering wheel will flash green as a reminder, shown in Figure 1.2. If continuous drowsiness are detected, the light-bar will turn red. For further fatigue being detected, a voice prompt will be heard with the vehicle automatically slows down and eventually brakes to a stop, associated auto driving features will be disengaged as well. The Cadillac Super Cruise System is a highly advanced system which not just works as a drowsiness prevention, but mainly provides the autonomous driving feature, as the result, only 2023 LYRIQ, 2021-2013 Escalade, 2022-2023 XT6 and a few newer models support this system. It has earned the ‘Best Hands-Free System’ award in the Technological category for Popular Science’s Best of What’s New 2022 (Popular Science, 2022).



**Figure 1.2** Cadillac Super Cruise System steering wheel lightbar (Cadillac, 2023)

### 1.2.2 Third-party Solutions

#### 1.2.2.1 DENSO DSM

DENSO DSM is a drowsiness detection system uses high-performance image recognition technology. It is equipped with a near-infrared camera to capture the driver’s face and establish driver condition prediction based on visual analysis (DENSO, 2022). The system uses a facial image database of more than 6000 people, which is declared to detect eye characteristics of distinct personals. The system consists of a camera and a main body, shown in figure 1.3, the camera only serves to capture images, while the body contains processors and built-in speakers, which will issue voice alerts if the analysis of images indicates fatigue behaviours. Although DENSO DSM is highly accurate and performs well, it is not compatible to many types of vehicles and costs much. With increasing numbers of OEM installed driver assistant systems, the DENSO DSM is not likely to become a popular choice.  
A picture containing electronics

Description automatically generated

**Figure 1.3** DENSO DSM equipment, camera and main body (DENSO, 2022)

#### 1.2.2.2 VIGO Smart Headset

The VIGO Smart Headset is a wearable equipment which poses an alternative way to monitor driver status. The device uses micro cameras to track blink rates, durations and drooping eyelids, through analysing bio signal patterns, the system will issue voice and vibration if potential drowsy behaviours occurred. It is also equipped with accelerometer and gyroscope to monitor head movements (VIGO, 2022). The system is highly customizable when connected to VIGO mobile application, where users can modify several aspects of alert behaviours including flashing lights, playing music or automated calls to family members. This is an interesting idea, which inspires certain user-friendly designs in the project.

The VIGO Smart Headset is shown in Figure 1.4

Graphical user interface, timeline

Description automatically generated with medium confidence

**Figure 1.4** VIGO Smart Headset (VIGO, 2022)

### 1.2.3 Individual Solutions

Many induvial projects have also been researched into (GitHub, 2023). The majority of developers favour image recognition method using machine learning, it is suspect that this is relatively easier to achieve with existing devices and frameworks, compared to physical detection like monitoring brain signals, which requires specific hardware and software protocols. Popular computer vison tools are OpenCV, TensorFlow, Deep Face, YOLO, and some other titles. Although there already exists a great number of individual drowsiness detection projects, most of them are very theoretically and basic, which merely works as a test example; the majority are nowhere near an operation-ready solution, which cannot be accessed or used by ordinary users, nor do they applicable to mobile devices. This yields the need to integrate detection ability and usability into a user-friendly and compatible solution.

It can be concluded that most present OEM solutions still remain exclusive to minor and high-end models of car product tress, meanwhile third party solutions more or less requires dedicated hardware to be purchased. For individual solutions, as it was mentioned, a proper integration is required, and it is the purpose of this project.

## Project Details

### 1.3.1 Main Focus

The project focuses on implementation of drowsiness detection on Android platform, by collecting real-time video footage through smart phone cameras then analysed and predicted by YOLOv5 model, users will be alerted by visual, sound and potentially other means if fatigue behaviours were detected. Users will also be able to choose processing unit to be GPU or CPU of their device for a better performance, and switch cameras for different scenarios. Besides, A simulated customizable dataset upload function is implemented.

The project has gone through the work of choosing dataset, choosing CNN framework, model training, optimization, mobile application development, and final integration.

The main process of this project is as follow:

1. Research and discuss with tutor about the main tasks and requirements of the project.
2. Conduct requirement analysis, including research on prevailing drowsiness detection solutions.
3. Confirm the system design of overall architecture, functional and non-functional designs, technological choices, and time table.
4. Study online the knowledge of image recognition, machine learning, android application development, required programming languages (Python, Java, C++).
5. Install and build environment, software and IDEs for the project.
6. Choose a suitable dataset and frameworks.
7. Model training and testing.
8. Model optimization.
9. Android application development.
10. Model implementation of mobile platform.
11. Additional features.
12. Final evaluation and optimization.

### 1.3.2 Key points

#### 1.3.2.1 Requirement analysis

By conducting research on fatigue driving and existing prevention methods to construct a knowledge base of behaviours relating to fatigue driving, and the pros of cons of current in-use methods. Confirmation of functional and non-functional requirements for system architecture design.

#### 1.3.2.2 Choosing dataset

Machine learning needs a suitable and accurate classified data set. Such datasets are hard to establish from scratch, luckily, there are many well-established human facial feature datasets that can be utilized. After several comparison, the dataset constructed by ***suhedaras*** was selected, which is well organized and takes a lot of common scenarios into account, coving 63 videos taken from 21 people (suhedaras, 2023).

#### 1.3.2.3 Choosing framework

Convolutional Neural Networks (CNN)is a class of artificial neural networks; it’s commonly applied to analyse visual imagery. CNN can be implemented to extract features of images and convert them into lower dimensions without losing characteristics. In this project, YOLOv5 framework is chosen for its high compatibility, accuracy and efficiency. This will be furtherly discussed in following chapters. Spyder and Jupyter Notebook are the IDEs used, the programming language is Python.

#### 1.3.2.4 Mobile application development

Develop an Android application which uses the trained model and NCNN framework to process visuals captured by mobile device’s camera and launch warnings when facial fatigue features have been detected. Android Studio is the choice of IDE; Java, C++ are the programming languages used in the development.

#### 1.3.2.5 Evaluations

After the initial integration of the system, the system is evaluated on its detection abilities, running performance, user experience, and room for improvements.

#### 1.3.2.6 Additional Features

Several additional features are be added in this stage, including camera switching, customizable dataset, and GPU / CPU switch. The UI and detection alerts have also been optimized.

## Content Overview

The thesis is divide into three chapters.

The first chapter is introduction and background research: this chapter introduces the topic and meaning of this project, followed by the background research of several existing drowsiness detection solutions, and finishes with summarized project detail explanations.

The second chapter is methods: this chapter first analyzes the project with both functional and non-functional requirements. The functional requirements are about the specific features that need to be implemented within the application, including each functionality and expected behaviors. The non-functional requirements focus on the usability, user experience, qualities, and performance of the system. Then the overall system architecture is explained, followed by breakdowns of each module which covers the development process with code snippets, screenshots, technical specifications, software and library requirements.

The third chapter is results: this chapter explains the operation and technical information indicatively, with an brief installation guide. Afterwards, each functionality is showcased with screenshots and description about operative procedures and outcomes. Readers will get a clear understanding of the key features of this applciaiton.

The final chapter is discussions: in this chapter, both development process and product quality are evaluated, with pros, cons, potential improvements, and errors related to the project discussed.

## Conclusion

The first chapter analyses the project thoroughly with examples ranging from OEM to individual solutions; points out the meaning, aim, main focus, and key points of the project, with an overview of the thesis and its three chapters.

# Chapter 2 Methods

## 2.1 Requirement Analysis

### 2.1.1 Functional Requirements

Using UML diagram in Figure 2.1 for functional requirements analysis. The application needs to function as follow:

1. user is able to choose the processing unit between CPU or GPU; this is crucial for maximizing performance since different device configuration has distinct strongness and weakness.
2. user is able to choose image collection portal between front camera or back camera;
3. real-time detection will issue no visual warning with alert flag set to false if no fatigue was detected;
4. real-time detection will issue visual warning with alert flag set to false if a brief moment of fatigue behaviour was detected;
5. real-time detection will issue visual warning with alert flag set to true if successive fatigue behaviours were detected; the alert flag could be used for any means of alert such as voice prompt or SMS notification etc.;
6. user is able to take selfie and send to the developers as custom data to improve dataset (simulation);

Diagram

Description automatically generated

**Figure 2.1** Functionality diagram (UML)

### 2.1.2 Non-functional Requirements

Non-functional requirements are crucial parts to ensure good quality of the software, these specifications describe and qualify the system’s capabilities and aim to promote its functionality. This part will focus on usability, user experience, and performance of the system.

1. Usability: The application should operate normally 24/7 on modern smart phones running latest Android OS, no crashes or fatal errors should occur.
2. User experience: The application should be easy to understand and operate without any complex instructions. The UI design should be clear and simple, there should be no interference in interfacing with each component. The visual alert should be strongly emphasized thus cannot be overlooked.
3. Performance: The application should be running smoothly (prefer 20 frames per second) on mid-end smart phones without stutters. Performance should be consistent; good power efficiency and minimal storage space are valued. The application should not harm the hardware by any means.

## 2.2 System Design

The project management follows waterfall model. Each procedure is finished after entering the next.

### 2.2.1 Overall Architecture

By conducting requirement analysis, the system will be running on Android platform, utilizing either front or back camera to collect visual information; the processing unit can be switched between CPU and GPU by simple interface. After processing, the YOLOv5 prediction results are furtherly processed to decide alert behaviours. The user interface needs to be clear to understand and simple to operate; key information are displayed and alert should be conspicuous. Users are also able to take photo as potential data to upload.

The system architecture and module functions are shown in Figure 2.2 (a). The system workflow is shown in Figure 2.2 (b).

Diagram

Description automatically generated

**Figure 2.2 (a)** System architecture graph

Diagram

Description automatically generated

**Figure 2.2 (b)** System workflow

### 2.2.2 Module Design Breakdown

This part makes a series of detailed breakdown of design of each module presented on the system architecture graph.

#### 2.2.2.1 User Interface

##### 2.2.2.1.1 Display

The main window contains interaction to all functions, with a camera view displaying performance, detection results and warning.

##### 2.2.2.1.2 Functions

1. **CPU/GPU Switch**

The feature enables user to switch between CPU or GPU to process detection. The choice is user’s based on the processing unit with better performance. The default processing unit is CPU. Toggle this button will appear a dropdown menu; the selected processing unit will remain on top.

1. **Camera Switch**

The feature enables user to switch between front and back camera. The default is front camera, as it makes more sense. Press this button will change the camera to the idle one, the name of selected camera will be shown above the button.

1. **Take Photo**

The feature enables user to take their selfie and potentially upload it to servers, which is merely a simulation for now. By engaging the button, the system camera application will be called, where user can take photos and adjust basic settings related to camera.

#### 2.2.2.2 Detection

1. **Camera**

Utilize camera functions by calling *ndkcamera* methods included in NCNN framework. The camera captures real-time footage and feed to YOLOv5 model by pixels.

1. **YOLOv5 Model**

After incoming images preprocessed and possibly resized, the detection model will generate proposal and probability rate, which are used as results shown on screen and evaluated by the alert module.

1. **Result Processing and Alert Generation**

With the results generated by YOLOv5, the alert module makes evaluation based on the proposal, probability rate and occurrence with reference to a given standard. In this implementation, if probability of fatigue behavior is greater than 70 percent, then user is evaluated to be drowsy; drowsiness happens over 10 times is tagged as extreme fatigue and the alert flag will be set top true, otherwise it is set to false and only visual warning will be issue.

## 2.3 Implementation

### 2.3.1 Development Environment

IDE: Spyder, Jupyter Notebook, Android Studio

Language: Python, Java, C++

Component: YOLOv5, ONNX, NCNN

Additional Software: Netron

Operating System: Windows11, Android 13

### 2.3.2 Module Explanation

#### 2.3.2.1 YOLOv5

##### **2.3.2.1.1 Overview**

YOLO, which stands for ‘You Only Look Once’, is a popular detection and image segmentation model created by Joseph Redmon and Ali Farhadi in 2015 (Ultralytics, 2023). The model used in this project – YOLOv5, is a product from Ultralytics.

Figure 2.3 shows the architecture of YOLOv5.



**Figure 2.3** YOLOv5 architecture map (OpenGenus, 2022)

##### **2.3.2.1.2 Architecture**

YOLOv5 shares a similar architecture from YOLO family, which consists of three main architectural blocks: Backbone, Neck, and Head.

The Backbone of YOLOv5 employs CSP-Darknet53 for feature extraction from images. As a deep network, YOLOv5 implements residual and dense blocks in order to enable the flow of information to the deepest layers and to overcome the vanishing gradient problem (roboflow, 2020). The CSPNet strategy gives YOLOv5 great advantages as it helps reducing the number of parameters and amount of computation, therefore leads to an increase of inference speed, which is crucial in real-time object detection (OpenGenus, 2022).

The Neck of YOLOv5 implements Path Aggregation Network and Spatial Pyramid Pooling. The PANet is modified from YOLOv4 to improve information flow and localization of pixels; The SPP operates as aggregation of information which returns a fixed length output with proper input. It is good at increasing the receptive filed and segregating most relevant context without damaging network performance (OpenGenus, 2022).

The Head of YOLOv5 is composed from three convolution layers that predicts the location of bounding boxes and scores the object classes.

##### **2.3.2.1.3 Activation Function**

YOLOv5 uses SiLU function (a) for convolution operations in the hidden layers, while Sigmoid function (b) is implemented with the same operation in the output layer.

Both function are illustrated in Figure 2.4.

Chart, line chart

Description automatically generated

**Figure 2.4** SiLU function **(a)** and Sigmoid function **(b)** (Xu et al, 2018)

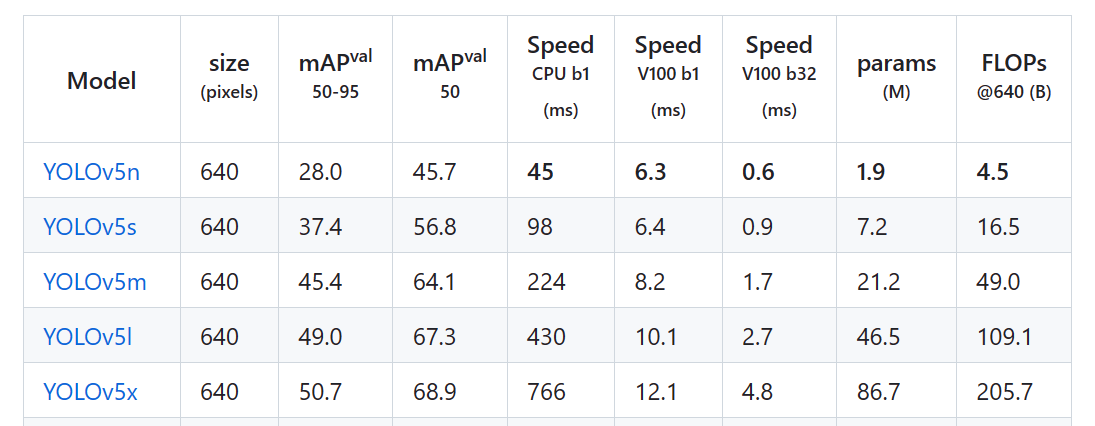
##### **2.3.2.1.4 Loss Function**

As YOLOv5 returns three outputs – classes, bounding boxes and objectness scores, it uses Binary Cross Entropy to compute classes loss and objectness loss, with Complete Intersection over Union loss to compute the location loss (OpenGenus, 2022).

##### **2.3.2.1.5 Sizes**

YOLOv5 consists of five different sizes, which are n, s, m, l, and x. The five models are the same in terms of operation, however, each has different number of layers and parameters (OpenGenus, 2022).

Figure 2.5 and 2.6 shows detail information of YOLOv5 models.



**Figure 2.5** YOLOv5 models list (Ultralytics, 2023)

Chart, line chart

Description automatically generated

**Figure 2.6** YOLOv5 models graph (Ultralytics, 2023)

This project chooses YOLOv5s, as it is lightweight and relatively accurate, which is recommended for mobile implementation because of the limitation of computational power on mobile devices. Although YOLOv5n performs better, its accuracy is not favored.

##### **2.3.2.1.6 Preparation**

###### **a. Dataset**

This project uses an open source dataset, established by *suhedaras*, which consists of 63 videos taken from 21 different people. According to author , three categories were considered in the videos: normal, yarning, and head position. Various lighting conditions and use of glasses were also taken into consideration (suhedaras, 2023).

All images were labeled in a total four classes: ‘normal’, ‘drowsy’, ‘drowsy2#’, ‘yawning’. ‘drowsy’ indicates eyes closed but head is upright, ‘drowsy2#’ indicates head dropping forward. There are total 1975 labeled images where 80 percent are used as training set, the rest 20 percent used for testing (suhedaras, 2023).

###### **b. Model Training**

This project installed the basic dependencies required in YOLOv5 official requirements.txt file. Before actual training, the YAML under data folder is edited, whereas the number and names of labels, the file path of train and test data are edited, shown in Figure 2.7.

Text

Description automatically generated

**Figure 2.7** Code snippet of YAML file

Then use *!python train.py --resume --imgsz 640 --batch 16 --epochs 600 --data data/data.yaml --cfg models/YOLOv5s\_drowsy.yaml --weights weights/YOLOv5s.pt --name drowsy\_result --cache --device 0* to start training process. The batch size is kept as default 16, image size of 640, and train for 600 epochs.

Part of the process is shown in Figure 2.8.

Scatter chart

Description automatically generated

**Figure 2.8** Screenshot of model training

The model performed well at 173 epochs as the training completed.

###### **c. Test and Result**

After training completion, file *best.pt* is created and stand for the trained model. Then use *python drowsy\_detect.py --weights runs/train/drowsy\_result/weights/best.pt --source data/drowsy\_training/test/images --hide-conf* to test on the remaining 20 percent dataset. The results are accurate and suitable for mobile implementation. The training process and result examples are shown in Figure 2.9 and 2.10.



**Figure 2.9** Screenshot of model resting

A person in a car

Description automatically generated with medium confidenceA person in a car with his mouth open

Description automatically generated with medium confidenceA picture containing text, person, indoor, helmet

Description automatically generated

**Figure 2.10** Result examples

###### **d. Conversion and Optimization**

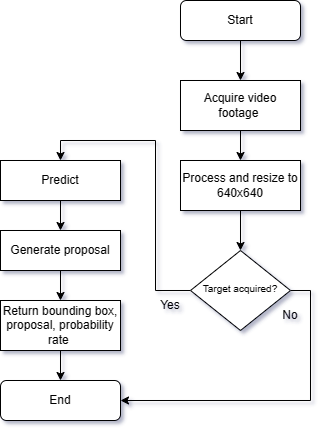
In order to fit in NCNN framework, the original .pt model needs to be converted to NCNN compatible format. ONNX and ONNX Simplifier are required in this step. First convert best.pt to best.onnx by using command *python models/export.py –weights best.pt --img 640 --batch 1*; then use ONNX Simplifier to simplify the model: *python -m onnxsim best.onnx best-sim.onnx*. The ONNX Simplifier works to simplify the ONNX model by replacing redundant operators with their constant outputs, thus to improve performance (daquexian, 2023).

After converting to ONNX format, use *onnx2ncnn best-sim.onnx best.param best.bin* to generate NCNN compatible format. However, this step occur error ‘Unsupported slice step’. This implies the Focus module needs manually adjustments. To fix this, Netron is used to visualize the model. By replacing content from Split to Concat with a customized YOLOv5Focus module and updating layer\_count to the actual value, the problem is solved. Additionally, the Reshape layer output grid needs to be set as -1, for NCNN is able to dynamically padding input size of images (Tencent, 2023). Following this, ncnnoptimize function is used to automatically adjust blob\_count and generate fp16 model, which furtherly reduces model size and improves performance. At this point, the best-fp16.param and best-fp16.bin is ready to work with NCNN framework.

##### **2.3.2.1.7 Implementation**

After video footages captured by camera, the video data is processed per frame as an image, then resized into 640 x 640 pixels and feed to the model. The model processes and predicts each image, if target is acquired then return bounding box, proposal, and probability rate.

The flow chart of detection process is shown in Figure 2.11.



**Figure 2.11** Flow chart of detection process

#### 2.3.2.2 NCNN Framework

##### **2.3.2.2.1 Overview**

NCNN is a high-performance neural network inference framework optimized for mobile platforms. Developed by Tencent, NCNN highly concentrates on deployment and usability on mobile phones as the core design. Tencent claimed that NCNN surpasses all existing open source mobile detection framework (Tencent, 2023). NCNN can be registered with custom layer implementation, which is very suitable for this project, as YOLOv5Focus is a custom layer.

##### **2.3.2.2.2 Implementation**

The mobile application is developed based on a highly rated NCNN-Android project, developed by *nihui* (nihui, 2020). The project is simple an depends only on NCNN library, it is lightweight and highly customizable. NCNN framework needs several configuration steps: firstly, ncnn-android-vulkan package needs to be installed under app\src\main\jni; then put the trained model – best-fp16.bin and best-fp16.param into app\src\main\assests, the framework will look for models inside this folder.

After configuring NDK in Android Studio, the yolov5.cpp needs to be edited:

1. In Yolov5::detect() function, the blob\_name inside stride 8, stride 16, and stride 32 are changed to the values of Permute layer of the model.
2. In Yolov5::draw() function, replace the class names with the four labels for this project, which are ‘normal’, ‘drowsy’, ‘drowsy#2’, ‘yawning’. Shown in Figure 2.12.

Text

Description automatically generated

**Figure 2.12** Code snippet of updating class names

#### 2.3.2.3 Mobile Application

##### **2.3.2.3.1 User Interface**

Following the UI requirement, the layout is constructed in Android Studio. Shown in Figure 2.13. The UI is divided into two main parts. The upper part is the dashboard containing all operable features. The button to switch camera is on the top, with information regarding to currently in-use detection model and CPU/GPU switch below. The button to take photo is on the right side, which will open the camera applciaiton if toggled. A reserved space is set at the top right corner for potential additional features.

The lower part is a camera view window, which also displays the fps counter, detection result, and visual alert if fatigue was detected.

Graphical user interface, diagram, application

Description automatically generated with medium confidence

**Figure 2.13** Application UI design

##### **2.3.2.3.2 Features**

###### **a. Camera Switch**

The feature is realized in MainActivity class, where a button object is bound with the switch camera button widget in layout. An on click listener is set to monitor if the button is pressed: this will trigger a variable new\_facing set to the opposite of current facing, in other words the other camera. Then closeCamera() is called to close current view, and openCamera(new\_facing) opens the other one. A tone will also be generated to indicate operation success.

Camera switch flow chart is illustrated in Figure 2.14, code snippet is shown in Figure 2.15.

Diagram

Description automatically generated

**Figure 2.14** Flow chart of camera switch function

Text

Description automatically generated

**Figure 2.15** Code snippet of camera switch function

###### **b. CPU/GPU Switch**

The implementation of this feature is similar to camera switch. The interface is a dropdown menu, where the default value is CPU. An on click listener is also bound with it. A variable current\_cpugpu is set to 0 initially, which indicates using CPU; it is changed to 1 if GPU is selected. Then the reload() function is called, with current\_cpugpu passed as the argument, it will call loadModel() function to launch detection procedure.

CPU/GPU switch flow chart is illustrated in Figure 2.16, code snippet is shown in Figure 2.17.

Diagram

Description automatically generated

**Figure 2.16** Flow chart of CPU/GPU switch function

Text

Description automatically generated

**Figure 2.17** Code snippet of CPU/GPU switch function

###### **c. Take Photo**

This feature simulates a portal to upload custom data. As designed, developers will be receiving raw data from users if they had error detection results due to facial features not commonly registered in the original dataset. This provides specific data and potential improvements to the dataset, which is valuable. A button widget is used to enable interaction with an on click listener. The function will open the device front camera for user to operate.

Upload photo flow chart is illustrated in Figure 2.18, code snippet is shown in Figure 2.19.

Diagram

Description automatically generated

**Figure 2.18** Flow chart of take photo function

Text

Description automatically generated

**Figure 2.19** Code snippet of take photo function

###### **d. Main window**

This is the main window to display detection results, alerts and fps counter. It follows the template structure.

Main window flow chart is illustrated in Figure 2.20, code snippet is shown in Figure 2.21.

**Diagram

Description automatically generated**

**Figure 2.20** Flow chart of main window

Text

Description automatically generated

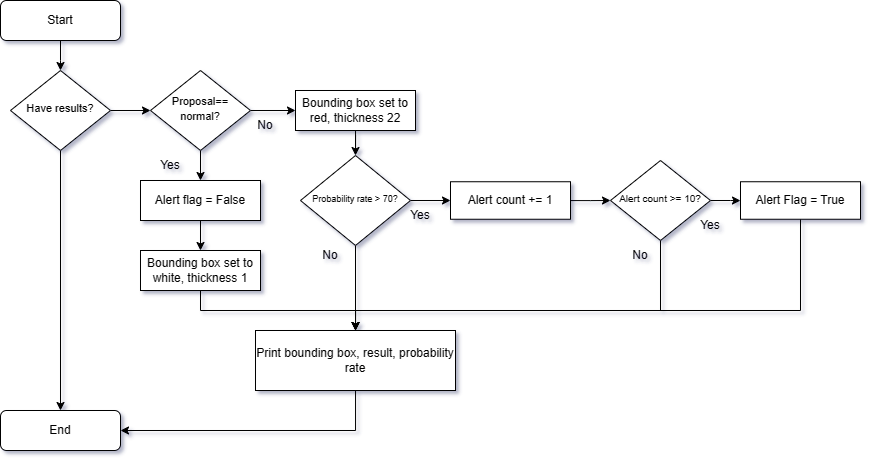
**Figure 2.21** Code snippet of main window

###### **e. Result Processing and Warning**

After getting proposals and probability rate from YOLOv5, they are furtherly analyzed and qualified for more reasonable outcomes. The proposals is stored in *class\_names[obj.label]*, which will be one of four labels. The probability rate is stored in *obj.prob*. This module follows the logic:

1. If proposal equals to ‘normal’, then no alert is issued, alert flag set to false. Bounding box is set to white with proposal and probability rate displayed.
2. Else, visual alert will be issued immediately, which is a red bounding box with thickness set to 22, with proposal and probability rate displayed. If the probability rate is greater than 70 percent, then a drowsy count will add one; when the counter passes ten, user will be judged as extreme drowsy, than the alert flag is set to True, which will trigger other alert modules, such as voice prompt, text messages etc.\

Flow chart of this module is shown in Figure 2.22.



**Figure 2.22** Flow chart of result processing and warning

###### **f. Fps Counter**

As the performance indicator, a FPS counter is set at the top right corner of the main camera view window. This is realized with the draw\_fps function, by dividing the total frames by time elapsed, and calculates the arithmetic mean of last ten results, the avg\_fps is returned and displayed onscreen with sprint() function.

Flow chart of this fps counter is shown in Figure 2.23.

Diagram

Description automatically generated

**Figure 2.23** Flow chart of fps counter

### 2.3.3 Testing

The project is tested on both windows emulated device and across several mobile devices using Android 13. There appears to be no problems in terms of functioning. 10 test cases were set for this project:

1. CPU detection using front camera, no glasses (normal, yawning, drowsy, drowsy2)
2. CPU detection using back camera, no glasses (normal, yawning, drowsy, drowsy2)
3. GPU detection using front camera, no glasses (normal, yawning, drowsy, drowsy2)
4. GPU detection using back camera, no glasses (normal, yawning, drowsy, drowsy2)
5. CPU detection using front camera, glasses (normal, yawning, drowsy, drowsy2)
6. CPU detection using back camera, glasses (normal, yawning, drowsy, drowsy2)
7. GPU detection using front camera, glasses (normal, yawning, drowsy, drowsy2)
8. GPU detection using back camera, glasses (normal, yawning, drowsy, drowsy2)
9. Take photo test
10. Alert flag test (normal, yawning, drowsy, drowsy2)

Test results are discussed in Chapter 3 and 4.

### 2.3.4 Version Control

The project follows strict version control and backup strategy. Each significant modification has been backup, recorded, and classified both locally and online.

GitHub Repository: Shadows2049/Project (github.com)

## 2.4 Conclusion

In this chapter, the system architecture and design are discussed specifically, with UML diagram and flow charts for demonstration. Features are explained with background information, design, environment, usage, and realization. Testing method and version control are also introduced. The project trained YOLOv5s model based on a well-established dataset, then simplified and converted to ONNX then NCNN format. The NCNN Android application is built on an open source model, with camera switch, CPU/GPU switch, detection, warning processing and alert generator, FPS counter, and customizable data upload feature implemented. The application runs smoothly with all function working on various Android devices.

# Chapter 3 Results

## 3.1 Technical Specification

### 3.1.1 Development

Equipment: MSI Raider GE78HX13V

GPU: RTX4080

CPU: Intel I9 13980HX

Memory: 64GB DDR5 5200mhz

Operating system: Windows 11 Pro

Libraries:

matplotlib>=3.2.2 numpy>=1.18.5 opencv-python>=4.1.2 Pillow>=7.1.2 PyYAML>=5.3.1 requests>=2.23.0 scipy>=1.4.1 torch>=1.7.0 torchvision>=0.8.1 tqdm>=4.41.0 tensorboard>=2.4.1 pandas>=1.1.4 seaborn>=0.11.0 coremltools>=4.1 onnx>=1.9.0 onnx-simplifier>=0.3.6 scikit-learn==0.19.2 tensorflow==2.9.1 tensorflowjs>=3.9.0 albumentations>=1.0.3 Cython pycocotools>=2.0 roboflow thop

### 3.1.2 Client

Equipment: Google Pixel 6

GPU: Mali-G78 MP20

CPU: Octa-core (2x2.80 GHz Cortex-X1 & 2x2.25 GHz Cortex-A76 & 4x1.80 GHz Cortex-A55)

Memory: 8GB

Operating system: Android 13

## 3.2 Demonstration

The project is demonstrated on Google Pixel6.

### 3.2.1 User Interface

This is the main menu (in CPU mode by default). The design is straight and simple, where user is able to access all features on the top dashboard; the window below shows the camera view with detection box, visual warning, fps counter, and detection results.

The user interface is shown in Figure 3.1.

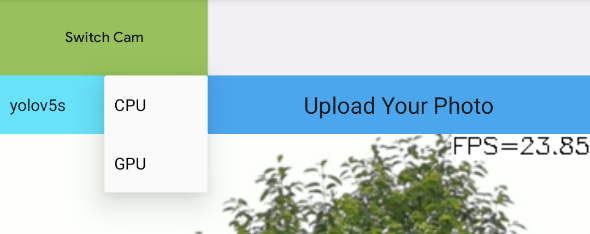
****

**Figure 3.1** User Interface

### 3.2.2 CPU/GPU Switch

A drop down menu which is capable of switching processing unit. The chosen method is shown after the menu collapsed.

Cpu/GPU switch is shown in Figure 3.2.

A picture containing text, screenshot

Description automatically generated

**Figure 3.2** CPU/GPU switch

### 3.2.3 Camera Switch

A button widget to press which will switch between front and back camera. A brief ‘beep’ sound indicates camera has been switched. Shown in Figure 3.3.

**A picture containing graphical user interface

Description automatically generated**

**Figure 3.3** Camera switch button

### 3.2.4 Take Photo

A button widget, shown in Figure 3.4, which takes user to a camera application to take photos, shown in Figure 3.5.

****

**Figure 3.4** Take photo button

A person taking a selfie

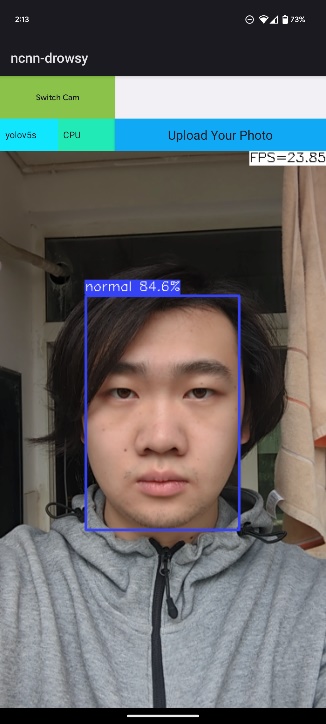
Description automatically generated

**Figure 3.5** Camera application

### 3.2.5 Detection In CPU Mode

Figure 3.6 shows real-time detection in CPU mode using front camera.

1. Normal: no drowsy behaviour, no visual alert.
2. Yawning: user is yawning, visual alert.
3. Drowsy: user is closing his eyes, visual alert.
4. Drowsy2: user’s head is dropping forward, visual alert.

Graphical user interface

Description automatically generatedA picture containing text, person, indoor

Description automatically generated

**(a) (b) (c) (d)**

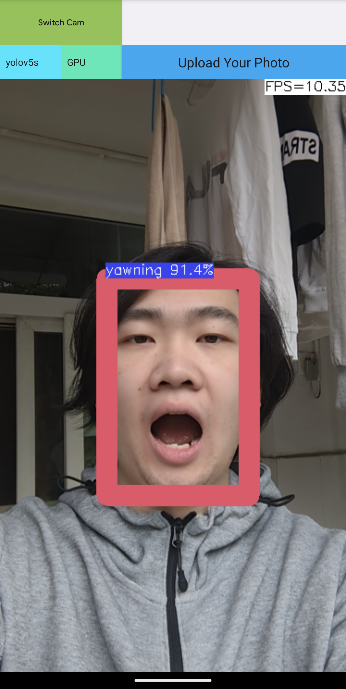
**Figure 3.6** Detection in CPU mode using front camera

### 3.2.6 Detection In GPU Mode

Figure 3.7 shows real-time detection in GPU mode using front camera.

1. Normal: no drowsy behaviour, no visual alert.
2. Yawning: user is yawning, visual alert.
3. Drowsy: user is closing his eyes, visual alert.
4. Drowsy2: user’s head is dropping forward, visual alert.

A person taking a selfie

Description automatically generatedA picture containing text, indoor, person

Description automatically generated

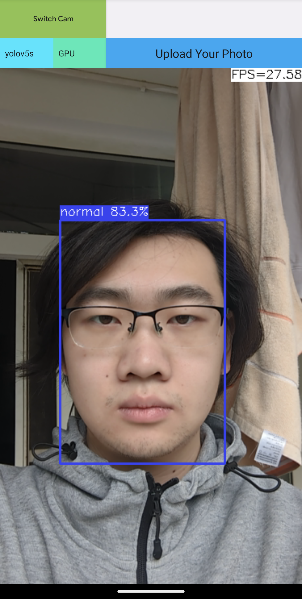
**(a) (b) (c) (d)**

**Figure 3.7** Detection in GPU mode using front camera

### 3.2.7 Detection In GPU Mode With Glasses

The following shows real-time detection in GPU mode using front camera, with user wearing glasses.

1. Normal: no drowsy behaviour, no visual alert.
2. Yawning: user is yawning, visual alert.
3. Drowsy: user is closing his eyes, visual alert.
4. Drowsy2: user’s head is dropping forward, visual alert.

A person wearing glasses

Description automatically generated with low confidenceA screenshot of a child

Description automatically generated with low confidence

**(a) (b) (c) (d)**

**Figure 3.8** Detection in GPU mode using front camera with user wearing glasses

# Chapter 4 Discussion

In the final chapter, the outcome of the project is evaluated .

## 4.1 Evaluation

### 4.1.1 Development

The development of this project follows the waterfall model. After requirement analysis follows system design, implementation, testing and deployment. All procedures are cascaded to each other, which works very efficiently because the requirements are clear and fixed for this project, the product definition is stable, the technology choices are confirmed, and all development resources are available.

### 4.1.2 Quality

#### 4.1.2.1 Functional

The solution strictly follows and completes all functional requirements. The application performs well on Android 13 with accurate results. Among all test cases: user wearing glasses, bright and dim lighting conditions, the solution detects drowsiness with no errors; only around 5 percentage performance loss in dim environment. Camera switch, CPU/GPU switch, take photo, and fps counter works flawlessly. In short, the system has good functional quality.

#### 4.1.2.2 Structural

The solution meets all non-functional requirements. The performance is good and consistent, with average 20 frames per second. The usability is good, with peer review and testing, multiple users have shown instant recognition and operated features without any instruction. The installation of application and usage has not occurred any errors yet. No fatal errors or system interruptions have been spotted since deployment.

### 4.1.3 Accuracy

The detection accuracy is promising. After testing, the success detection of normal is around 99 percent, the success detection of yawning is 100 percent, the success detection of drowsy is around 99 percent, and the success detection of drowsy#2 is 100 percent. Meanwhile, in dim environment, some normal expressions with eyes not fully opened tend to be recognized as drowsy; this poses no severe problem since the probability rate is usually below 70 percent thus will not count for the alert flag.

### 4.1.4 Performance

The running performance meets the expectation, with averaging 20 frames per second. Specifically, GPU mode performs 20 percent better than CPU mode on Google Pixel 6.

#### 4.1.4.1 GPU Mode

The average performance in GPU mode is around 22 fps, with up to 28 fps when idle and lowest 11 fps when result process is engaged.

#### 4.1.4.2 CPU Mode

The average performance in CPU mode is around 18 fps, which is slower than GPU mode, with up to 23 fps when idle and lowest 10 fps when result process is engaged.

#### 4.1.4.3 Consistence

The application runs consistently, with low battery consumption and no overheating behaviours. After an hour testing, it uses only 10 percent of battery, with no significant fps losses. Considering modern vehicles have mobile charging ports, this application is ready to run for a considerable long time.

## 4.2 Conclusions

Following the waterfall model, the project is managed and tracked by both cloud and local version control method.

This project has achieved the goal of developing a light weight mobile deployed driver status monitoring system, which will issue alert if fatigue behaviours were detected; and user can customize their experience in several aspects: change detection camera to be front or back, switch to CPU or GPU to run detection model, and take selfie as custom data to potentially upload to developers.

The detection model uses YOLOv5s, trained by the dataset established by ***suhedaras***, then transformed into NCNN format with a custom focus layer, ONNX simplified, and fp16 optimization.

The mobile detection framework uses NCNN and developed based on ***nihui***’s template. It uses the YOLOv5s model with several modifications.

The mobile application is developed using Android Studio, designed and implemented to detect, alert, choose camera, choose CPU or GPU, and a simulation of customizable data upload.

After development, the project has been deployed and tested across several devices using Android 13 and each operated normally with accurate results.

The performance is acceptable and consistent. However, the design of result processing module needs to be improved, as it might takes too much resource which has caused noticeable performance degradation.

Overall, the project is solid and meets the requirements given at the beginning.

## 4.3 Ideas for future work

For future work with similar structure, the waterfall model can be implemented as it is very efficient and simple. The code structure in this project is mostly integrated, which is not preferable for debugging or changes; therefore, future work should use modularized design to separate each function, which should improve reusability and maintenance convenience.

Due to time limitation, the project only implements YOLOv5s model. Meanwhile, there are other suitable candidates that are NCNN compatible. The mobile applciaiton framework has interface for changing detection model, which would work easily with other model also loaded. For future work, more models shall be included for variety and comparison.

# List of References

1. NSC, 2022, ‘Drivers are Falling Asleep Behind the Wheel’, [online]. Available at: Fatigued Driving - National Safety Council (nsc.org)

2. Heitmann, A. & Guttkuhn, R. & Aguirre, A. & Trutschel, U. & Moore-Ede, M., 2001, “Technologies for the Monitoring and Prevention of Driver Fatigue”, Driving Assessment Conference 1(2001), 81-86. doi: <https://doi.org/10.17077/drivingassessment.1013>

3. DENSO, 2022, ‘Driver Status Monitor’, [online]. Available at: <https://www.denso.com/global/en/business/products-and-services/mobility/pick-up/dsm/>

4. BMW, 2022, ‘BWM Driver Assistance’, [online]. Available at: <https://www.bmw.ca/en/topics/experience/connected-drive/bmw-connecteddrive-driver-assistance.html>

5. Bahouth, J. Withum, K. Bahouth, G. Digges, K. Stratton, J. Spicer, R. Steinhauser, F. Baur, P. Schulman, C., 2018, ‘Ten-year review of BMW Accident Research Program-investigated cases’, [online]. Available at: <https://pubmed.ncbi.nlm.nih.gov/30841809/>

6.Ford, 2022, ‘Driving Aids – Driver Alert’, [online]. Available at: <https://www.fordservicecontent.com/Ford_Content/vdirsnet/OwnerManual/Home/Content?variantid=>

7.Euro NCAP, 2011, ‘Ford Driver Alert’, [online]. Available at:

<https://preview.thenewsmarket.com/Previews/NCAP/DocumentAssets/220990.pdf>  
8. Cadillac, 2023, ‘Super Cruise’, [online]. Available at: <https://www.cadillac.com/ownership/vehicle-technology/super-cruise>

9. Popular Science, 2022, ‘The 100 Greatest Innovations of 2022’, [online]. Available at: <https://www.popsci.com/technology/best-of-whats-new-2022/>

10. VIGO, 2022, ‘Science’, [online]. Available at: <https://www.wearvigo.com/science>

11. GitHub, 2023, [online]. Available at: <https://github.com/>

12. suhedaras, 2023, [online]. Available at: <https://github.com/suhedaras>

13. [Ultralytics](https://ultralytics.com/yolov5), 2023, [online]. Available at: <https://ultralytics.com/yolov5>

14. Xu, R. Lin, H. Lu, K. Cao, L., 2021, ‘A Forest Fire Detection Based on Ensemble Learning’, [online]. Available at: <https://www.researchgate.net/publication/349299852_A_Forest_Fire_Detection_System_Based_on_Ensemble_Learning>

15. OpenGenus, 2022, ‘YOLOv5 Model Architecture Explained’, [online]. Available at: <https://iq.opengenus.org/yolov5/>

16. Ultralytics, 2023, [online]. Available at: <https://github.com/ultralytics/yolov5/issues/6998>

17. roboflow, 2020, ‘What is YOLOv5? A Guide for Beginners’, [online], Available at: <https://blog.roboflow.com/yolov5-improvements-and-evaluation/>

18. daquexian, 2023, [online]. Available at: https://github.com/daquexian/onnx-simplifier

19. Tencent, 2023, [online]. Available at: <https://github.com/Tencent/ncnn/wiki/how-to-implement-custom-layer-step-by-step>

20. nihui, 2020, [online]. Available at: <https://github.com/nihui/ncnn-android-yolov5>

21. Termly, 2022, ‘9 Legal Requirements for Apps and Tips to Meet Them’, [online]. Available at: <https://termly.io/resources/articles/legal-requirements-for-apps/>

22. Prezi, 2014, ‘Social and Ethical Issues in Software Development’, [online]. Available at: <https://prezi.com/0rujph02s_fv/social-and-ethical-issues-in-software-deveopment/>

# Appendix A Self-appraisal

This section covers self-evaluation, reflections, legal, social, ethical and professional considerations.

## A.1 Critical self-evaluation

### A.1.1 Strengths

I am confident with my study skills and dedication towards the work. The great amount of time spent on the project did not deflect any of my passion. I have accomplished all the requirements set for the project, with a variety of problems dealt with. I am strong at problem solving for my ability to discern symptoms, good practice of online key-word searching, and great patience. Besides, my time arrangement of work is reasonable; even when incidents happened, I was able to quickly make adjustments and emendations.

### A.1.2 Area of Improvements

Sometimes I struggled to a determine a single point without viewing as the whole, this led to several design errors which eventually caused more time to redesign.

## A.2 Personal reﬂection and lessons learned

The project was carried out in November 2022, with development started in December 2022 – finished in early April 2023. Through the process, I worked hard and have been an active learner. Before the project, I only have basic and theoretical knowledge of machine learning. After countless hours of studying on GitHub, YouTube, Stack Overflow, reading articles, following documents of each library and frameworks, I have now managed to use YOLOv5, NCNN, and develop Android Application. Even though, my mastery of Java and C++ is weak, especially talking about JNI method, which till now is a big headache for me. In this project, the mobile application relies greatly on cooperation and cooperation between Java and C++, e.g. the sound alert feature has not been successfully implemented due to the way C++ calls Java, I haven’t figure out a way to achieve this properly. Well, at least I made a sound module and the idea of alert flag, which only needs a suitable way to interact with each other. As I don’t plan to be a C++ developer, I would look into C++ during my master programme, still. To summarize, this project is successful, though with some minor problems, I am pretty satisfied with it and myself.

## A.3 Legal, social, ethical and professional issues

As this is a mobile application which would be deployed and used by various individuals, especially as it is a driver status monitoring system that directly related to driving safety, the legal, social, ethical and professional issues need to be addressed.

### A.3.1 Legal issues

As the driver monitoring system acquires enormous user facial images, the privacy and data security cannot be overlooked. To ensure the security of processing user’s personal data is the fundamental legal obligation, as the developers must store user data in secured databases which meet the legal requirements set by governments and regions, and should also be strictly supervised by authorities (Termly, 2022). The application should also meet accessibility requirements, copyright and plagiarism requirments, and the end-user licence agreement (EULA).

### A.3.2 Social issues

The software developers have the responsibility to maintain a healthy online environment and make sure any problems met by users responded and resolved in a timely, accurate and efficient manner (Prezi, 2014). A way to help realize this is to classify problems efficiently, which could be: critical, major, normal, cosmetic.

The software should also operate without damaging hardware or poses danger to user’s physical health.

### A.3.3 Ethical issues

Being a driver status monitoring system, it must consider the possibilities of operational failure and faulty detection result. Therefore, user must be warned about any potential errors made by the software, and they must be aware that the software only works as a reminder, the judges and choices are theirs to made.

Users must also be acknowledged and agreed the collection and usage of their personal data, the security methods taken and associated supervisions should be clear to public.

### A.3.4 Professional issues

As a prototype, the application can only be used as educational purpose, it should not be used in real on-road situation by any means for users safety. It is recommended to use Google Pixel 6 with Android 13 to run the software to achieve similar results with the development environment.

All resources used in the applciaiton are open source and licenced. All resources used in the thesis are referenced.

# Appendix B External Materials

Open source resource used in this project:

1. YOLOv5 framework: <https://github.com/ultralytics/yolov5>
2. NCNN framework: <https://github.com/Tencent/ncnn>
3. Drowsy facial dataset: <https://github.com/suhedaras>
4. NCNN-Android framework: <https://github.com/nihui/ncnn-android-yolov5>
5. ONNX-Simplifier: <https://github.com/daquexian/onnx-simplifier>