INVESTIGATION OF TRAFFIC SIGN IMAGE CLASSIFICATION FOR SELF DRIVING CAR

FARRA HERLIENA BINTI MD ZIN

B.ENG (HONS.) ELECTRICAL ENGINEERING (ELECTRONICS)

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name	: Farra Herliena Binti Md Zin
Date of Birth	: 11/04/1999
Title	: Investigation of Traffic Sign Image Classification For Self-Driving Car
Academic Session	: Semester 2 2021/2022

I declare that this thesis is classified as:

☐ CONFIDENTIAL	(Contains confidential information under the Official Secret Act 1997)*
☐ RESTRICTED	(Contains restricted information as specified by the organization where research was done)*
☑ OPEN ACCESS	I agree that my thesis to be published as online open access (Full Text)

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies of the thesis for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:

farraherliena
(Student's Signature)

[illegible]
(Supervisor's Signature)

990411025268
~~New IC~~/Passport Num~~ber~~
Date: 17/06/2022

Assoc. Prof. IR. DR. Fahmi Bin Samsuri
Name of Supervisor
Date: 17/06/2022

NOTE : * If the thesis is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.

THESIS DECLARATION LETTER (OPTIONAL)

Librarian,
Perpustakaan Universiti Malaysia Pahang,
Universiti Malaysia Pahang,
Lebuhraya Tun Razak,
26300, Gambang, Kuantan.

Dear Sir,

CLASSIFICATION OF THESIS AS RESTRICTED

Please be informed that the following thesis is classified as RESTRICTED for a period of three (3) years from the date of this letter. The reasons for this classification are as listed below.

Author's Name
Thesis Title

Reasons	(i)
	(ii)
	(iii)

Thank you.

Yours faithfully,

____
(Supervisor's Signature)

Date: 17/06/2022

Stamp:

Note: This letter should be written by the supervisor, addressed to the Librarian, *Perpustakaan Universiti Malaysia Pahang* with its copy attached to the thesis.

SUPERVISOR'S DECLARATION

I hereby declare that I have checked this thesis and in my opinion, this thesis is adequate in terms of scope and quality for the award of the degree of Engineering in Electrical Engineering (Power System).

[illegible]

____

(Supervisor's Signature)

Full Name	: Assoc. Prof. Ir. Dr. Ts. Fahmi Bin Samsuri
Position	: Lecturer / Supervisor
Date	: 17/06/2022

ASSOC. PROF. IR. TS. DR. FAHMI BIN SAMSURI
ASSOCIATE PROFESSOR
DEPARTMENT OF ELECTRICAL ENGINEERING
COLLEGE OF ENGINEERING
UNIVERSITI MALAYSIA PAHANG
LEBUHRAYA TUN RAZAK
26300 GAMBANG, KUANTAN, PAHANG
TEL : +609-549 2338 FAX : +09-424 5055

____

(Co-supervisor's Signature) Full Name :

Position	:
Date	:17/06/2022

STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institution.

farra herliena

(Student's Signature)

Full Name	: FARRA HERLIENA BINTI MD ZIN
ID Number	: EA18085
Date	: 17/06/2022

INVESTIGATION OF TRAFFIC SIGN IMAGE CLASSIFICATION FOR SELF DRIVING CAR

FARRA HERLIENA BINTI MD ZIN

Thesis submitted in fulfillment of the requirements
for the award of the B.Eng (Hons.)
Electrical Engineering (Electronics)

College of Engineering
UNIVERSITI MALAYSIA PAHANG

JUNE 2022

ACKNOWLEDGEMENTS

First and foremost, I thank Allah for providing me with the time, good health, and well-being that were crucial for me to write this thesis report. I would like to convey my heartfelt gratitude to my parents for their unwavering support and encouragement during my research project; without them, I would not have been able to continue. I would like to thank Assoc. Prof. Ir. Dr. Fahmi Bin Samsuri for his tremendous counsel, constant support, and patience in leading me through my PSM 1. I am also grateful to my friends, who have always supported me and encouraged me to finish my PSM 1 research project. My research has been a fantastic adventure thanks to their generous assistance and support.

ABSTRAK

Cara hidup kita telah berubah akibat kemajuan teknologi. Kecerdasan Buatan telah memberi kesan yang baik pada semua bidang dan menjadikan kehidupan kita lebih mudah. Pembangunan sistem trafik berbantukan teknologi merupakan satu langkah penting ke hadapan dalam industri automotif. Dengan pertumbuhan kenderaan autonomi, industri automotif bertambah baik dengan pesat. Kenderaan autonomi adalah kesimpulan tertentu dalam masa terdekat, dan ia bertujuan untuk selamat dan mudah. Salah satu isu paling kritikal untuk kenderaan autonomi ialah klasifikasi tanda lalu lintas. Warna pudar, oklusi separa oleh halangan sekeliling, variasi dalam pencahayaan dan keadaan cuaca, bayang-bayang, pantulan pada papan tanda pada waktu siang dan gerakan kabur adalah beberapa isu paling tipikal yang mungkin berlaku semasa mengenal pasti dan mengesan tanda lalu lintas. Prestasi Rangkaian Neural Convolutional (CNN) telah mengatasi manusia yang sama dalam pengenalpastian dan klasifikasi tanda lalu lintas. Fokus kajian adalah untuk menjadikan klasifikasi ini setepat mungkin bagi mengurangkan kemalangan dan meningkatkan kredibiliti kereta pandu sendiri. Jika tidak, ekologi trafik mungkin terancam. Menggunakan pemprosesan imej dan teknologi pemprosesan penglihatan mesin, serta penggunaan pembelajaran mendalam dalam klasifikasi sasaran, kaedah pengecaman tanda trafik berdasarkan CNN dikaji. Kaedah pengesanan dan pengelasan tanda lalu lintas dengan kecekapan tinggi dan kecekapan tinggi dicadangkan. Penanda Aras Pengiktirafan Tanda Lalu Lintas Jerman (GTSRB) digunakan untuk ujian, dan keputusan menunjukkan bahawa kaedah yang dicadangkan boleh memperoleh hasil yang terbaik berbanding dengan pendekatan terkini. Dalam penyelidikan ini, model CNN untuk alam semula jadi dicadangkan. Model CNN Tanda Trafik digunakan dengan menambahkan empat lapisan padat atau lapisan bersambung sepenuhnya. Lapisan bersambung sepenuhnya belajar secara mendalam kerana ia mempelajari ciri daripada semua gabungan ciri lapisan sebelumnya.

ABSTRACT

The way we live has changed as a result of technological advancements. Artificial Intelligence has had a good impact on all fields and is making our lives easier. The development of a technology-assisted traffic system is a significant step forward in the automotive industry. With the growth of autonomous vehicles, the automotive industry is improving rapidly. Autonomous vehicles are a certain conclusion in the future, and they are intended to be both safe and convenient. One of the most critical issues for autonomous vehicles is traffic sign classification. Half occlusion, color fade by surrounding barriers, variations in shadows, reflections on signboards during the day, and movement blurring different lighting and weather situations are some of the most typical issues that might occur when identifying and detecting traffic signs. In the classification and identification of road signs, the performance of a Convolutional Neural Network (CNN) has outperformed the same of humans. The purpose of this study is to boost the accuracy of this classification in order to minimize the accident and enhance the credibility of self-driving vehicles. Otherwise, the ecology of traffic may be jeopardised. Using image processing and machine vision processing technologies, as well as the use of in-depth learning in target classification, the traffic sign recognition method based on CNN is studied. A traffic sign detection and classification method with high efficiency and high efficiency are proposed. The German Traffic Sign Recognition Benchmark (GTSRB) is employed to test the approach method, and the results reveal that it outperforms state-of-the-art approaches. A CNN model for deep nature is suggested in this paper. The CNN Model for Traffic Signs is consists of four dense layers or layers that are entirely linked. Because it learns characteristics from all of the attributes of the preceding layers, a fully connected layer learns deeply.

TABLE OF CONTENT

Contents

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

CNN	Convolution Neural Network
TS	Traffic Sign
GTSRB	German Traffic Sign Recognition Benchma
BTSD	Belgium Traffic Sign Dataset

LIST OF EQUATION

CHAPTER 1

INTRODUCTION

1.1 Introduction

The traffic road is a major element of the road network. It is critical to obey traffic rules and comply traffic signs in order to guarantee safety. Road safety is an issue for both pedestrians and drivers. Weather conditions, traffic signs, congested highways, road conditions, and other factors all have an impact on road safety.[1] We are concentrating on introducing autonomous vehicles as the boom of the automotive industry. It is believed to be both safer and more effective. The classification of traffic signs is a crucial part of self-driving vehicles. It should be done systematically in order to reduce road accidents and boost self-driving vehicle credibility.

Computer systems even now experience a crucial pattern recognition challenges when it comes to classifying traffic signs. Traditional techniques, such as Machine Learning Algorithms, have been proposed to classify traffic signs. Deep learning is the research of artificial neural networks and associated machine learning algorithms that have more than one hidden layer. [2] Computational models with numerous processing layers can develop several levels of abstraction for data representations through deep learning. [3] The capability of deep learning to give precise recognition and forecasting has continually enhanced. [2] As a result, deep learning outperforms machine learning techniques. The project’s scope would involve the study and conduct parametric analysis, as well as image classifications using deep learning techniques.

1.2 Project Background

Deep learning is a type of machine learning in artificial intelligence that utilizes images, text, or sound in the model to perform classification tasks. The neural network architecture explains how they work. The neural network operates on the same principles as the human brain. Deep networks include hundreds of layers, whereas neural networks have only two or three. Autonomous or self-driving car is capable to make decision based on orders where it could assist human and improve traffic safety. A crucial component of the road network is the traffic sign. To maintain safety, it is critical to obey traffic signs and execute road laws. Traffic signs exist in different shapes, colors, and symbols where quiet challenging in the detection and performing image classification using deep learning technique. Image classification is such a task based on assigning a correct label to an image that machines can look on it. The proposed in this research is the accomplishment of a Traffic Sign (TS) CNN model for traffic sign classification by adding four dense layers or fully connected layers. The traffic sign detection and image classification will be performed using Convolution Neural Network in this research, with Python as the interpreter and PyCharm as integrator.

1.3 Problem Statement

There are variety of cases involving drivers mistakenly read or do not notice traffic signs due to the traffic sign damaged, weather, or tiredness. Some of the drivers also refuse to obey the traffic signs which will cause many issues on the road. Road accident usually happens due to the human error such as over-speeding, violation of rules, fatigue, and failure to understand signs. Driver will be able to make the same mistake and probably it will open quite often. Some of them do not aware of the general rules and safety measures while using roads. There are some incidents involving drivers who go the wrong way on purpose and those who travel the wrong way by accident. Other than that, there is an autonomous driving system that has no road sign detection and identification which only can execute a parking system like autonomous driving that is not fully self-driving. Thus, to reduce the fatigue issue that has dominated the major contributing factors to road accidents, and enhance the credibility of an autonomous car, an autonomous vehicle with the ability to detect traffic signs and perform image classification will be designed and the simulation will be run in Python as an interpreter that uses PyCharm as the integrator. The method used in this project is Convolution Neural Network by deep learning involves TensorFlow, OpenCV, and KERAS.

1.4 Objectives

Based on the problem statement stated, the purposes of this project are:

- To detect traffic signs and perform image classification using Traffic Sign (TS) CNN model for traffic sign classification.
- To cultivate the credibility of autonomous/ self-driving car in preventing harm to the traffic ecology.
- To reduce the number of road accident by execute appropriate action on the vehicle’s movement based on traffic sign detection and classification.
- To obtain the highest accuracy and probability of the traffic sign classification.

1.5 Scopes

Python version 3.7 is used as a platform to integrate coding in PyCharm and allows an iteration of training module of traffic sign CNN model. Convolutional neural networks are used to training and recognize traffic signs. This will be performed continuously with OpenCV and a webcam. With the aid of TensorFlow and Keras, traffic signs will be trained on more than 35000 pictures from 43 various classes in this project.Traffic sign Convolutional neural Network Model will be employed. The model will experience data preprocessing divided into three steps which are Gray Scale Conversion, Histogram Equalization, and Normalization. The model will be operated on German Traffic Signs Recognition Benchmark Dataset (GTSRB). The obtained results will show the accuracy, loss, and probability of the traffic sign classification. The target of the accuracy must be greater than 98.44% on the test data. The model should work efficiently and reach better accuracy compared to other techniques.

1.6 Thesis Overview

In the chapter 1, the content will be more focused on the introduction and background of the project, the problem occurs that leads to this project, the objective of this project which is to provide a solution of the problem that arise. Other than that, the scope on this project regarding what kind of method that are going to be implemented, which is Traffic Sign Convolution Neural Network Model. Besides, Python version 3.7 as platform to interpret the codes. Lastly, PyCharm as a platform to perform the simulation on our model using Convolution Neural Network.

Then, in chapter 2 is the literature review covered in this project. This literature study should include a broad range of topics, and it should culminate in a focus on what we want to achieve, or our aims. This chapter begins with the an autonomous self-driving car, Convolution Neural Network, and traffic sign detection and classification. Each subtopic will go through each component in great depth. Finally, the technique for classify the traffic sign, with Convolution neural Network being one of the approaches to assess.

Next, in chapter 3, the whole procedure for this project will be discussed in this chapter. The methods used are Convolution Neural Network and will be explained in detailed according the flowchart of the coding. Also, this chapter also will discuss the software that are required to run the simulation.

While in chapter 4, this chapter will go through the results of the simulation that was conducted in Chapter 3. After that, in the last section of this chapter, the traffic sign classification will be thoroughly discussed based on the output from the result obtained. The accuracy of the model for classifying the traffic sign will be shown.

Finally, in Chapter 5, the general conclusion of this project will be concluded, as well as a proposal or recommendation for improvement in the next project for further consideration.

CHAPTER 2

LITERATURE REVIEW

2.1 Autonomous/ Self-Driving Car

Various technological companies and institutions have demonstrated a high demand in self-driving automobile in recent years, committing enormous financial and technological resources to research and development. Alphabet's subsidiary Waymo, NVIDIA's PilotNet, Delphi's collaboration with MIT-based start-up nuTonomy, GM's Cruise, Tesla's 'S' model, BMW's work with Ford's Agro AI, BAIDU, and others are among those conducting considerable study. As a consequence, autonomous driving is on the edge of becoming conventional, supposing it surpasses technological and practical challenges as well as social, economic, and legal approval. Aside from that, as part of the implementation of these autonomous vehicles, particularly driver assistance systems, various manufacturers and laboratories have concentrated their efforts on the use of visual data for the recognition of road, vehicles, pedestrians, and traffic lights. The basic principle of road sign recognition systems is to detect signs, define their meaning, and then transfer the information to the driver (through a projection on the windshield, a screen, or a smartphone) or to the vehicle itself, which then performs the action without requiring a human decision.

A visual language is formed by the combination of traffic lights and road signs, which forms a set of laws whose interpretation aids in disciplined driving. The perception of traffic discipline has a lot of industrial potential in the field of autonomous driving. By giving important information, the vision system assists in analyzing current traffic conditions on the road, dangers, and challenges surrounding the vehicle, and warning and assisting them in safe, convenient, and healthy navigation. Despite the fact that it is easier said than done, vision-based object detection and recognition in traffic scenes continues to be a problem for the self-driving sector.

2.2 Level of Autonomous Car

The Society of Automotive Engineers (SAE) has defined six stages of vehicle automation, ranging from zero (completely manual) to five (completely automated) . These specifications have been authorised by the US Department of Transportation. To begin with, level 0 (No Driving Automation). The vast majority of cars on the road today are Level 0 (manual) vehicles. Despite the presence of technology to help the driver, the "dynamic driving task" is carried out by humans. A vehicle classed as Level 0 (zero) has no driving automation tools. The driver is solely accountable for the vehicle's operation in this situation, which involves steering, accelerating, braking, parking, and any other movement needed to move the automobile in any position. Nevertheless, at Level 0, a driver assistance system that can assist temporarily while driving can be activated. Stability control, forward-collision alert, automatic emergency braking, blind-spot notification, and lane-keeping assistance are among the available features. These technologies are designated as Level 0 since they do not operate the automobile, but they do offer alerts or take prompt action in specific situations.

Next is Level 1(Driver Assistance). This is the simplest type of automation. A single automated system for driver assistance, such as steering and acceleration (cruise control), is installed in the vehicle. Adaptive cruise control, which keeps the vehicle at a safe distance behind the following car, qualifies as Level 1 since the human driver monitors other aspects of driving such as steering and braking.

After that, we will continue on to level 2 (Partial Driving Automation). ADAS stands for "advanced driver assistance systems." The vehicle manages both steering and acceleration and deceleration. The automation falls short of self-driving because a human stays in the driver's seat and can take charge of the vehicle at any time.

Then there's Level 3 (Conditional Driving Automation). The change from Level 2 to Level 3 is considerable from a technological position, but it is small, if not non-existent, from a human one. Level 3 vehicles are capable of monitoring their environment and making intelligent decisions on their own, such as accelerating past a slow-moving automobile. They do, however, need human assistance. The driver must be vigilant and ready to take command if the system fails to accomplish the task.

The next level that we concentrated on was Level 4 (High Driving Automation). The primary distinction between Level 3 and Level 4 automation is that Level 4 vehicles have the authorization if the system breaks down or something unexpected happens. As a result, most scenarios do not require human interaction with these automobiles. Yet, the system can still be handled manually by a human. Self-driving vehicles at Level 4 are possible. Nevertheless, unless legislation and infrastructure change, they can only do so in a limited region (usually an urban environment where speeds achieve an average of 30mph). This is known as geofencing.

Finally, Level 5 (Full Driving Automation). Because Level 5 cars do not involve human attention, the "dynamic driving task" is removed. Steering wheels and pedals for acceleration and braking will be absent from Level 5 vehicles. Geofencing will be removed, enabling them to go wherever they desire and do whatever a good human driver can. Fully autonomous vehicles are being studied in a variety of areas throughout the world, but none are yet commercially available.

2.3 Convolution Neural Network

The convolutional neural network (CNN) is a deep learning technique. CNN gains abilities to analyze patterns by learning from samples. The trained network can then be used to forecast new layouts. When it comes to feature abstraction, CNN with several layers offers a lot of strength. Upper layers extract corners or more complex composite patterns, whereas lower layers abstract simple structures like lines and oblique lines. The network then analyzes how the extracted features affect the final result.CNN saves computing time by reusing parameters by applying the same calculation to multiple regions using the convolution kernel. This allows for the creation of more sophisticated and accurate architectures.

Deep CNNs have gradually supplanted traditional computer vision algorithms in terms of pattern identification and object classification. However, the largely heuristic architecture design of deep artificial neural networks remains a topic of research. In 2012, Krizhevsky et al. reignited interest in CNNs by demonstrating significant progress in picture classification accuracy at the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). [4] Prior to the usage of CNNs, the most of pattern recognition jobs were completed with the help of a hand-crafted feature extractor and a classifier. [5] The utilization of CNN gives extraordinary, better object detection production than the systems based on simpler HOG-like features by putting objects with a deep network and training a high capacity model with only a small quantity of annotated detection data. [4]

2.3.1 CNN Network Structure Model

The hidden layer (which includes the convolution layer and the sub-sampling layer) is a crucial component in CNN feature extraction. Convolution is an image processing component that is used to enhance certain aspects of an image while suppressing others. A simple convolution neural network schematic diagram is shown in Figure 1. When a computer receives an image, it stores it in the form of a matrix. The gray value of each pixel will be evaluated by the input layer and stored in the form of a matrix if the input is a gray image after processing. If a colour photo is input, the computer will store it as an RGB colour model. The image in this case must be represented by three matrices in a logical order. After then, CNN analyses the data.

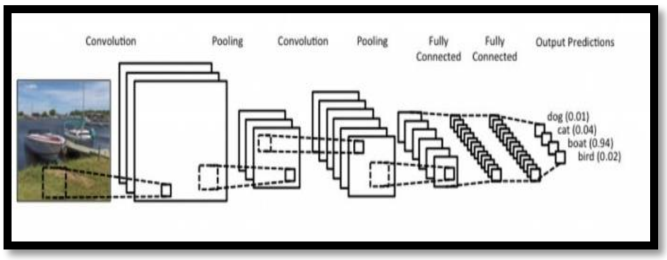


Figure 1Architetecture of CNN

2.3.2 CNN Image Classification

CNN's strength is that its multi-layer structure allows it to automatically learn features and attributes at various layers. The shallow convolution layer's perception region is narrower, and it learns the attributes of specific local areas. These abstract properties are less affected by the size, position, and orientation of the item, resulting in better recognition. These abstract features can also be utilised for categorization and to determine what kinds of items are present in an image.However, because some object details are lost, it is difficult to specify the particular contour of the object and indicate which object each pixel belongs to, making effective segmentation extremely difficult.

CNN's autonomous learning ability can be utilised to extract features rather than manual design, saving time on tedious tasks like manual marking and optimization. As the big data era unfolds, training data is extremely available. As science and technology develop, as does the hardware platform, the speed of computer operation increases. As our knowledge and development of algorithm autonomous learning increases, and the performance of autonomous learning methods grows, recognition systems will rely on more and more learning. CNN has made great advances in picture categorization at the moment.

2.3.3 Design of Deep CNN Architecture

The following five stages are proposed and implemented in the same order to design deep CNN architecture.

1. Choosing the depth of a CNN based on the number of convolutional-pooling pairings.
2. In each convolutional layer, selecting a number of feature maps.
3. Finding a sufficient number of neurons in the completely connected layer that is hidden.
4. Calculating the percentage of dropout needed after each layer.
5. Using sophisticated features by swapping out max pooling layers

Every preceding stage is assumed to be incrementally improved in the proposed approach. This method is unique in that it uses a single pre-processed dataset to select the optimum deep CNN model architecture. After pre-processing, the RGB dataset is divided by 255.0. After determining the optimum deep CNN model architecture for this dataset, it must be applied to the remaining pre-processed datasets. Every model is trained 100 times in order to select the best architecture at each level. The hypothesis is that the model that has been trained 100 times at each stage and has the greatest of the most frequent accuracy among the other models is better at generalizing and hence is the best. The most common accuracy is the model's accuracy, which was discovered not during a single training procedure, but over 100 training procedures.

2.4 Deep Learning in Autonomous/ Self-driving Cars

The Deep Learning (DL) algorithm is a subset of Artificial Intelligence (AI) that can learn and classify things by mimicking the human brain's ability to understand data. It may be used to forecast the future and make decisions based on present factual information. Deep learning is a synonym for Representation Learning (DL). Deep learning is based on machine learning algorithms that use nonlinear modifications to extract top-level concepts from data. [6] The term "deep" comes from the training process, and it refers to the number of layers in a neural network.

The deep learning process is divided into two parts, the first of which is the training stage, followed by the inferring step. The training stage entails discovering matching features and labelling large volumes of data, whereas the inferring stage entails labelling fresh data based on existing knowledge. Each neuron in a deep learning or deep neural network takes many signals as input. It linearly mixes the weights with these signals and feeds them to the nonlinear function to get the desired result. Deep learning algorithms, domain expertise, and manual labor are used to extract features automatically.

On a single image, the majority of deep learning algorithms have been extensively studied for object detection and instance segmentation (still image detection). That is, a succession of frames is treated as a collection of individual images, with information between neighboring frames ignored. Deep learning is the best method in classification and recognition especially for the traffic sign which is the famous object to detect in autonomous vehicles.

2.5 Traffic Sign Detection and Classification

The goal of the traffic sign recognition technology is to inform drivers to the importance of following current traffic laws. This technology detects and categorises a variety of traffic signals, advising drivers of the current maximum speed limit, prohibitory signs, and dangerous road bends. When it comes to identifying traffic signs, computer systems still face a difficult pattern recognition challenge. [4] Traditional methods such as Machine Learning Algorithms have been used to classify traffic indicators. Machine learning, which developed from the field of artificial intelligence, is critical because it allows robots to acquire human-like intelligence without having to train them directly. [7]

Most car manufacturers include traffic signs recognition systems into ADS (Auto Driving System) and ADAS (Advanced Driver-Assistance System). This technology makes use of computer vision and image recognition techniques to recognize and classify traffic signs. To boost classification efficiency, some systems include a tracking phase when processing video sequences. [8] Due to the strong development of machine learning algorithms, especially deep learning, traffic sign recognition methods based on machine learning have to turn into the foremost algorithms in recent years.

2.6 Comparison of Journal

The best five journals are chosen, then will be discussed in this section in terms of purpose, method, and result. The title of the journal, year, and author also will be stated.

2.6.1 Research paper 1

Research paper 1 is Traffic Sign Classification Using Deep Neural Network from Merin Annie Vincent, Vidya K R, Santhosh P Mathew published in 2020. By adding four dense layers or completely linked layers to the (Traffic Sign) TS CNN Model, this paper presents a (Traffic Sign) TS CNN Model. The focus of this research is to use a CNN model to improve the categorization accuracy of traffic signs. Using GTSRB dataset.

RELU and SOFTMAX as activation functions, Adam Optimizer is utilized, Keras and TensorFlow are usedOn test data, the model has an accuracy of 98.44 percent. The model outperforms current state-of-the-art techniques.

2.6.2 Research Paper 2

Research paper 2 is CNN based Traffic Sign Classification using Adam Optimizer from Smit Mehta, Chirag Paunwala, Bhaumik Vaidya published in 2019.This paper proposes an approach for traffic sign detection which uses the Convolutional Neural Network (CNN) for classifying the traffic signs. Belgium traffic sign dataset (BTSD) has 62 classes. The purpose is to classify the traffic signs using Adam Optimizer. SOFTMAX as activation function (it calculates a probability of every possible class), Adam Optimizer is utilized (very fast), for the methodology have 3 stages: data acquisition, preprocessing, and classification. [9] It employs dropout to combat overfitting by dropping part of the units from the neural network at random, and it is also thought to be the most efficient method of model averaging. The experimental findings based on this network architecture reveal that CNN is more fast and accurate.

2.6.3 Research Paper 3

Research paper 3 is design of Deep CNN Model for Effective Traffic Signs Recognition from Valentyn N. Sichkar, Andrey V. Lyamin published on 2021. This research proposes a method for constructing deep Convolutional Neural Network (CNN) architecture for the classification of traffic signs. The purpose is to find the effectiveness of the CNN design architecture. GTSRB dataset is used. Design of deep CNN architecture: Picking the network's depth via the number of convolutional pooling pairs, selecting the required number of feature maps in each convolutional layer, determining the required number of neurons in the hidden fully connected layer, and predicting the required percentage of dropout after each layer, with the goal of determining whether substituting max-pooling with convolution with strides 2 or average pooling enhances the constructed architecture. [10] The following processes are included in the created approach for effective traffic sign classification: creating a deep CNN architecture, applying data pre-processing, and using data augmentation approach. As activation functions, Keras deep learning library, RELU, and SOFTMAX were used for training and testing. The results of the experiments reveal that this method can achieve a testing accuracy of more than 99% for traffic sign classification. The model, however, was unable to classify several of the traffic signs.

2.6.4 Research Paper 4

Research paper 4 is Traffic Sign Detection by Image Preprocessing and Deep Learning from Mert Çetinkaya, Tankut Acarman published in 2021. Using a deep learning-based object detection framework and a deep learning-based classifier, this paper suggested a traffic sign detection and recognition system. The use of a blurring-based image preprocessing strategy to improve the training efficiency of object detectors by reducing noise is elimination. GTSRB dataset is used. The objective is to demonstrate the efficiency of the proposed strategy by comparing it to existing techniques. Faster R-CNN, a very effective object detection system, is combined with the Inception Resnet V2 system's feature extractor, and it is analysed and validated for traffic sign identification and recognition. A new approach for preprocessing images to improve the detection performance of a traffic sign detector is also proposed to improve the effectiveness of the presented architecture. Pre-trained on Microsoft COCO is the publicly accessible Faster R-CNN Inception Resnet V2 model. [9] The dataset comes from the TensorFlow model zoo on GitHub. [11] The results reveal that the proposed system operates properly, and that the proposed image preprocessing method increases the object detector's performance.

2.7 Summary

Four best journals are listed. Research paper 1 is chosen for methodology in Investigation of Traffic Sign Image Classification for Self-Driving Car where the Traffic sign CNN model is employed in detecting and classifying traffic sign. This is because method used in paper 1 is the best compared to the others since TS CNN model can outperform the existing state-of-the-art algorithms. The algorithm was explored and implement using Python version 3.7 and PyCharm as software to integrate the codes. The research papers listed due to the best methodology, result, and the purpose where is same as our aim which we want to detect and classify the traffic sign and perform image classification for self-driving car. Compared to other solutions in papers listed, in research paper 3, the results of the experiments reveal that this method can achieve a testing accuracy of more than 99 percent for traffic sign classification even though the model was unable to classify several of the traffic signs. The paper design the CNN model architecture using five steps. In Final Year Project 2, the second method is examined and implement in this study.

CHAPTER 3

METHODOLOGY

3.1 Introduction

The approach for the Traffic Sign Image Classification for Self-Driving Car using Traffic Sign Convolution Neural Network model will be described fully in this chapter. There are two major software that simulates Python (3.7) and PyCharm for this project. To improve the accuracy of traffic sign classification, this study used a CNN model. On the GTSRB dataset, data preprocessing was done first, which included Gray Scale conversion, Histogram Equalization, and Normalization. Data augmentation comes after data preparation. Data augmentation enhanced the volume of training data to prevent overfitting. Following that, the GTSRB dataset was used to train the Traffic Sign CNN model.

3.2 Data Preprocessing

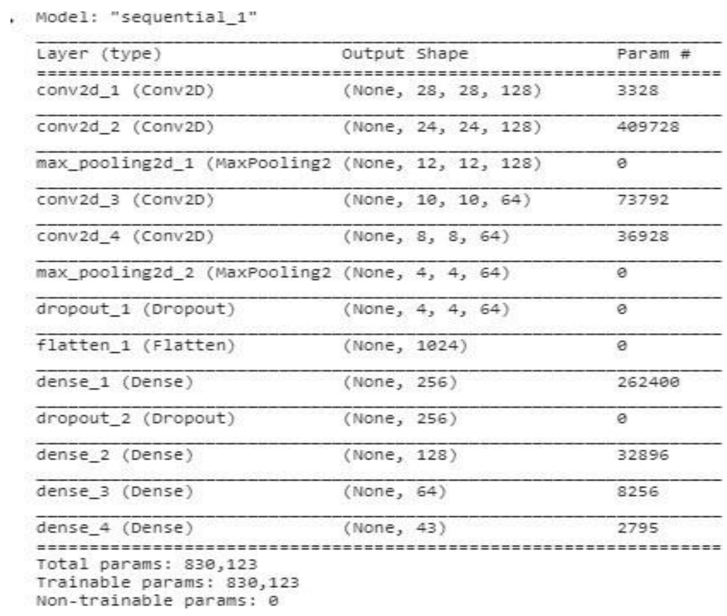
The proposed system utilized the GTSRB dataset which consists of traffic images. There are 34799 labelled images in the training set, 4410 labelled images in the validation set, and 12630 images in the test set. There are 43 classes in the dataset, with imbalanced class frequencies. [9] In GTSRTB dataset, all of the images were selected from a series of videos recorded at different times of the day and in varied lighting conditions, and the video was taken from a moving vehicle to avoid motion blur. [12] As a result, the GTSRB dataset has low resolution and low contrast. Subsequently, data preparation is required to improve accuracy. A variety of traffic signs such as Stop, Pedestrian, Road work, No passing, Yield, Bumpy road, Speed Limit, Priority road, Double curve, Slippery Road, etc. are available in GTSRB Dataset. There are 3 steps of data preprocessing in this model, which are Gray Scale conversion, Histogram Equalization, and Normalization. Preprocessing images is a crucial aspect of image classification. This model's complexity is increased by using a colour image. As a consequence, converting a colour image to a grayscale image is one of the image processing technique.[13] The RGB values (24 bit) are converted into grayscale when converting a colour image to a grayscale image (8 bit). [14] Displaying colour contrasts in grayscale with minimal detail loss provides a more informative picture for visual inspection and interpretation.[10]

After that, due to the poor contrast of the images in the dataset, Histogram Equalization is implemented. The histogram of oriented gradients is now widely acknowledged as a useful tool for capturing shape information. The purpose of histogram equalization is to reduce the number of grey levels in addition to improving the image's contrast. After that, the image will have an even grey distribution. After that, normalization is carried out. Normalization is the process of transforming an input image into a range of pixel values that are normal to the senses.

3.3 Data Augmentation and Model Summary

Employing Data Augmentation to artificially augment the dataset using label preserved transformations is the easiest and most frequent approach to reduce overfitting on image data. [15] By flipping, rotating, and zooming images, data augmentation is used to increase the number of images in a dataset. [14] The Keras ImageDataGenerator function is used to augment data in this model.

This is a sequential Keras model. Keras is an open source library for creating deep neural networks that provides highly powerful and abstract building elements. [16] On top of TensorFlow, it also functions as an API. The Traffic Sign CNN model consists of four convolutional layers, two pooling layers, one flatten layer, and four fully connected layers. Figure 2 shows Traffic Sign CNN Model Summary in this study.



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 128)       3328
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 24, 24, 128)       409728
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 10, 10, 64)        73792
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 8, 64)          36928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 4, 4, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               262400
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               32896
_________________________________________________________________
dense_3 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_4 (Dense)              (None, 43)                2795
=================================================================
Total params: 830,123
Trainable params: 830,123
Non-trainable params: 0
```

Figure 2 TS CNN Model Summary

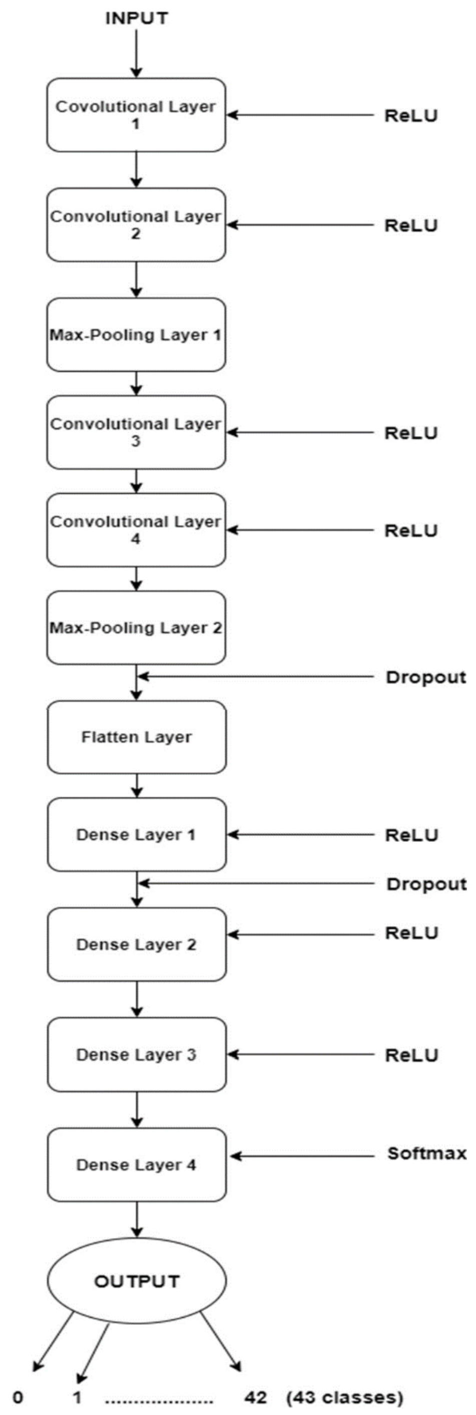


Figure 3 TS CNN Model Architecture

Convolutional layers convolute the input image, and after each convolutional layer, a ReLU activation function is utilized. RELU with Deep Convolutional Neural Networks are more efficient and faster in training.

Table 1 render a model hyperparameter for this project. Epoch utilized for this case is 60 since epoch value will affect the time taken iteration in PyCharm. Epoch also effect the accuracy and loss gained because the higher epoch will increase the accuracy and decrease the loss. Batch size utilized is 128 where it represents how many it processes together. Because the output has more than two classes, the 'categorical crossentropy' is used for Loss function in our model. Adam optimizer is employed since it is the efficient and fastest compared to the other optimizer. Two activation function are utilized which are RELU and SOFTMAX. RELU could improve the fastness in the training while SOFTMAX will calculate the probability of every possible class. 20% and 50% of dropout. The dropout method is used in neural network training to avoid over-fitting by turning off some neurons during the training phase to provide the network a flexible margin to react to inputs do not present in the training examples data.

EPOCH	**60**
BATCH SIZE	**128**
LOSS FUNCTION	**categorical_crossentropy**
OPTIMIZER	**Adam**
ACTIVATION FUNCTIONS	**RELU & SOFTMAX**
DROPOUT	**20% & 50%**

Table 1Model Hyperparameter

3.4 Traffic Sign Dataset

For the classification of traffic signs, the German Traffic Sign Recognition Benchmark (GTSRB) is utilized. [17] As shown in table I, there are 43 classes divided into three categories. Table 2 below shows the dataset distribution.

Category	**Task**	**Number of images**	**Shape**
Training data	Utilized to train the Network	34799	4 dimensions tensor to discover the image index in the dataset, the pixel's row-column and the data it carries (Red Green Blue value)
Validation data	Enables to keep a check on the network's performances while training it (a reduced version of testing data)	4410	
Testing data	Employ to analyze the final network	12630	

Table 2The Dataset Distribution [17]

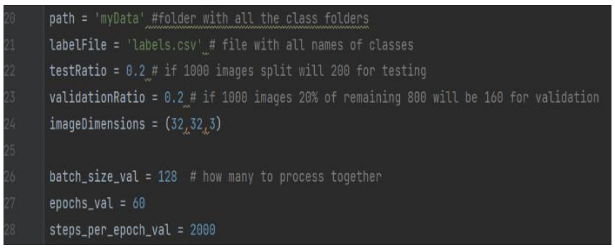
3.5 Python (3.7) and PyCharm

Python (3.7) is used as an interpreter and PyCharm as integrator software to execute the coding for this project. PyCharm is an integrated development environment (IDE) for computer programming that concentrates on the Python programming language. PyCharm is designed by programmers to offer all the tools needed for productive Python development. PyCharm is a hybrid-platform developed by JetBrains as an IDE for Python. It is a common tool for creating Python applications. With OpenCV and a simple webcam, this will be done in real time. In this research, traffic signs will be trained from GTSRB images from 43 different classes using TensorFlow and Keras.

3.6 Coding from PyCharm and Python (3.7)

All the steps of coding will be shown in this part. The coding is built and run in the platform of PyCharm as integrator and Python version 3.7 as interpreter. Two part of coding were built and simulate which are the first part is to code to train the model while the second part is to code to test the model. For the first part which are training code, this part will allow the coding to iterate. In the second part, from the test code, after the code is run, the web camera will be generated. This allows the traffic sign to be detected after it is shown in front of the camera. This will be done in real time with OpenCV and a simple webcam. With the help of TensorFlow and Keras, traffic signs will be trained with around 35000 images from 43 different classes in this project.

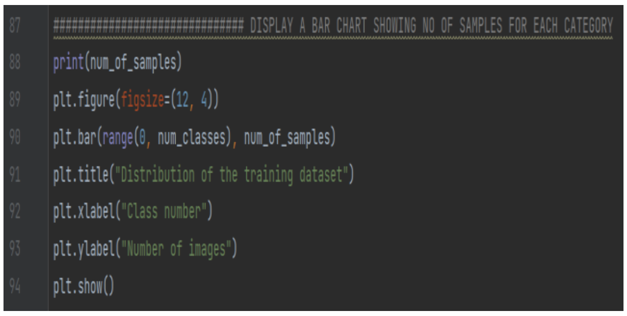
For the first part where the code to train, Epoch value used is 60 where the epoch value will affect accuracy and lost. Highest value of epoch will affect the time taken for the iteration where for 60 of epoch taken more than two hours for training code to iterate. Step per epoch value is 2000 means every 1 of epoch value will iterate until 2000 and it will iterate until 60 which is around 120000 of iteration. The graph of accuracy and loss for training and validation data will be shown in the preliminary result. The batch size used is 128 while the size image utilized is 32*32*3. Figure 4 depicts the initialization Coding.



```
path = 'myData' #folder with all the class folders
labelFile = 'labels.csv' # file with all names of classes
testRatio = 0.2 # if 1000 images split will 200 for testing
validationRatio = 0.2 # if 1000 images 20% of remaining 800 will be 160 for validation
imageDimensions = (32,32,3)

batch_size_val = 128  # how many to process together
epochs_val = 60
steps_per_epoch_val = 2000
```

Figure 4 Initialization Coding

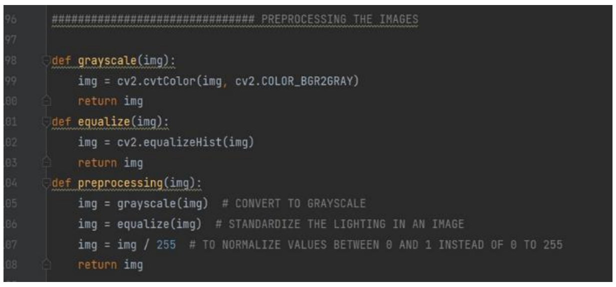
Figure 5 depict to display distribution of dataset bar and images corresponding to class number and name. There are 43 classes of traffic sign images in GTSRB where it owns names and class number respectively. Traffic sign only can be detected if we show images available from GTSRB.



```python
############################ DISPLAY A BAR CHART SHOWING NO OF SAMPLES FOR EACH CATEGORY
print(num_of_samples)
plt.figure(figsize=(12, 4))
plt.bar(range(0, num_classes), num_of_samples)
plt.title("Distribution of the training dataset")
plt.xlabel("Class number")
plt.ylabel("Number of images")
plt.show()
```

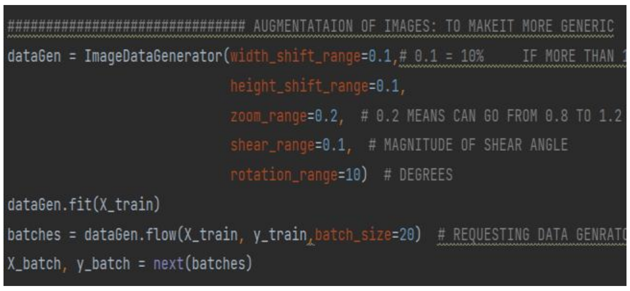
Figure 5 Distribution of Dataset and Images Corresponding to Class number and Name

Figure 6 below shows the data preprocessing coding which involves of Gray Scale conversion, Histogram Equalization, and Normalization on the GTSRB dataset. After that, data augmentation comes after data preprocessing. To avoid overfitting, data augmentation expanded the number of training data. Figure 7 shows data augmentation of image to make the image s become more generic.



```python
############################## PREPROCESSING THE IMAGES

def grayscale(img):
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    return img
def equalize(img):
    img = cv2.equalizeHist(img)
    return img
def preprocessing(img):
    img = grayscale(img)  # CONVERT TO GRAYSCALE
    img = equalize(img)  # STANDARDIZE THE LIGHTING IN AN IMAGE
    img = img / 255  # TO NORMALIZE VALUES BETWEEN 0 AND 1 INSTEAD OF 0 TO 255
    return img
```

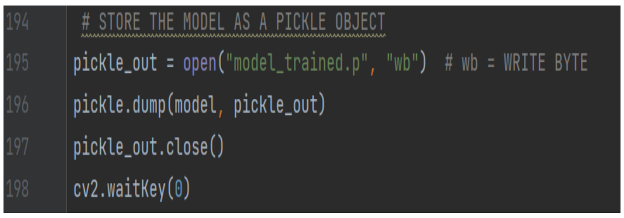
Figure 6 Data Preprocessing Coding



```python
############################### AUGMENTATAION OF IMAGES: TO MAKEIT MORE GENERIC
dataGen = ImageDataGenerator(width_shift_range=0.1,# 0.1 = 10%     IF MORE THAN
                             height_shift_range=0.1,
                             zoom_range=0.2,  # 0.2 MEANS CAN GO FROM 0.8 TO 1.2
                             shear_range=0.1,  # MAGNITUDE OF SHEAR ANGLE
                             rotation_range=10)  # DEGREES
dataGen.fit(X_train)
batches = dataGen.flow(X_train, y_train,batch_size=20)  # REQUESTING DATA GENRAT
X_batch, y_batch = next(batches)
```

Figure 7 Data Augmentation of image

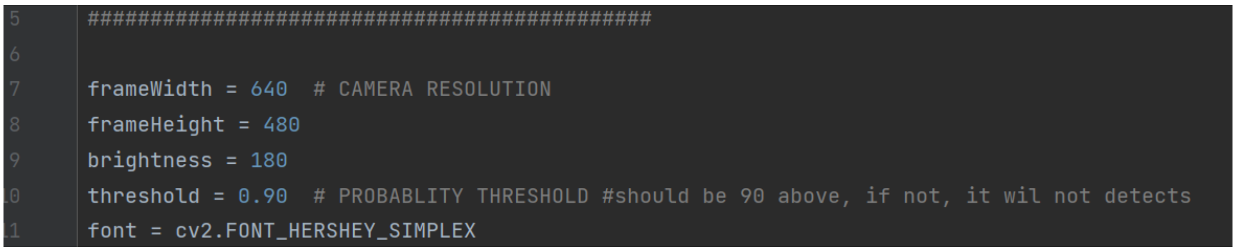
Lastly, the code stores the model as a Pickle object, which is a Python module that allows you to serialize a Python object into a binary format and then deserialize it again to a Python object. The coding depicts in figure 8.



```
# STORE THE MODEL AS A PICKLE OBJECT
pickle_out = open("model_trained.p", "wb")  # wb = WRITE BYTE
pickle.dump(model, pickle_out)
pickle_out.close()
cv2.waitKey(0)
```

Figure 8 To Store the Model as a Pickle Object

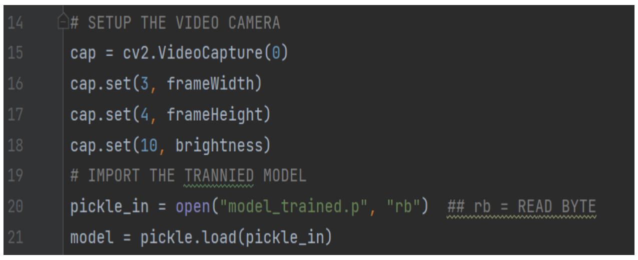
From the figure 9 below, the threshold value is setup as 90. This is because the higher value of threshold will produce good probability of detection. Images that detect below than 90% of probability will be considered as low probability of detection and will not be detected.



```
#############################################

frameWidth = 640  # CAMERA RESOLUTION
frameHeight = 480
brightness = 180
threshold = 0.90  # PROBABLITY THRESHOLD #should be 90 above, if not, it wil not detects
font = cv2.FONT_HERSHEY_SIMPLEX
```

Figure 9 Threshold value setting

In the figure 9 shows the second part of the coding which is test code. In this part, we will declare all the traffic signs class number and class name. Then, the video will be generated.



```
# SETUP THE VIDEO CAMERA
cap = cv2.VideoCapture(0)
cap.set(3, frameWidth)
cap.set(4, frameHeight)
cap.set(10, brightness)
# IMPORT THE TRANNIED MODEL
pickle_in = open("model_trained.p", "rb")  ## rb = READ BYTE
model = pickle.load(pickle_in)
```

Figure 10 Set up the Video Camera

3.7 Flowchart of the Traffic Sign Convolution Neural Network Model

In the figure 11 contained flowchart of programming is shown, the process in the coding is discussed and will be explained. The images from GTSRB dataset contained 43 classes will be import and the data will be split. Then, the number of images will be examined whether it is matches to the number of labels for each dataset. The number of images in the dataset must be corresponding to the number of id and name of traffic signs. After that, the coding to read CSV file is executed. Then, the coding will lead to exhibit some samples images of all the classes. The distribution of the training dataset output will be displayed after the coding is built and run where it is showed in a bar chart displaying number of samples for each category. It also will show the images proportional to its id number and name of traffic signs.

After that, data preprocessing is executed to the images. We are preprocessing our images. Firstly, convert it into grayscale. Then, equalizing our images so that we have standard standardized lighting for each image. Lastly, we are normalizing the values. Then, depth of 1 is added. Then, it followed by data augmentation to the images where it produces the image to become more generic. The augmentation images samples are displayed. Now, the data is trained for the Convolution Neural network model, the data is plotted. We have convolutional layers, pooling, and drop-out layers. In the end, we have our dense layer which is our output layer. Our plot is loss and validation loss, and accuracy and validation accuracy. Lastly, In the coding, the model is stored as pickle object. Code for training model is run, the video camera is generated. Images of any traffic signs are shown in front o the camera. Then, the probability in percentage, class id, and name of traffic sign will appear after the image is shown.

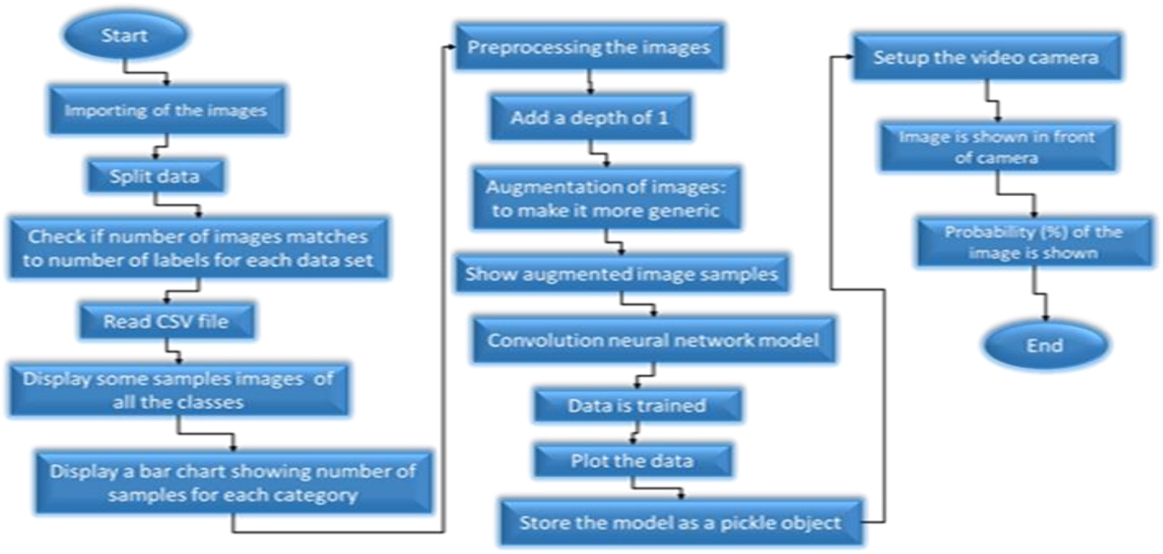

Start
Importing of the images
Split data
Check if number of images matches to number of labels for each data set
Read CSV file
Display some samples images of all the classes
Display a bar chart showing number of samples for each category
Preprocessing the images
Add a depth of 1
Augmentation of images: to make it more generic
Show augmented image samples
Convolution neural network model
Data is trained
Plot the data
Store the model as a pickle object
Setup the video camera
Image is shown in front of camera
Probability (%) of the image is shown
End

Figure 11 TS CNN Flowchart System

CHAPTER 4

PRILIMINARY RESULTS AND DISCUSSION

4.1 Introduction

This chapter consists analysis for the outputs from the simulation of Traffic Sign CNN model in PyCharm and Python (3.7). The technique is operated on GTSRB dataset which consists of 43 classes of traffic signs. Keras Library was used to classify traffic signs using the TS CNN model. Keras is an opensource library that acts as an API on top of TensorFlow and provides very powerful and abstract building blocks for building deep learning networks. The sequential API of Keras permit to generate models layer-by-layer for most problems.

4.2 Outputs from the coding

Figure 12 below shows the distribution of the training dataset where exhibit the number of images corresponding to the class number. The number of images for each class number is not same to the other class.

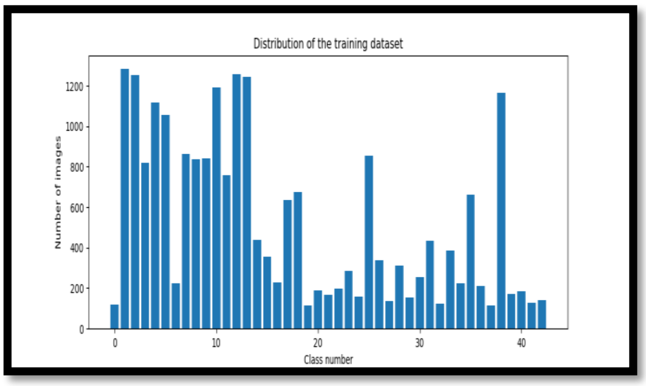


Figure 12 The distribution of the training dataset

The type of images on GTSRB contained 43 classes of traffic sign images are shown. Subsequently, the other images other than images shown in the figure is exhibited in front of camera, the detection and classification operation cannot be performed. This is because this method is operated on GTSRB dataset, the traffic signs images need to be same as in the images shown. It is can be view in figure 13 below.

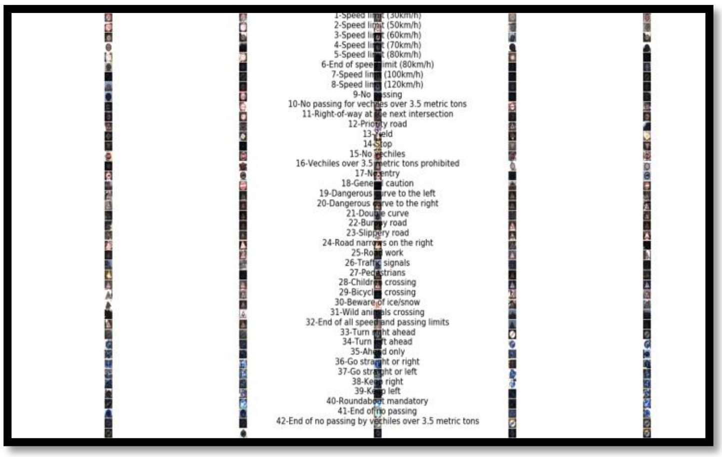


Figure 13 Images corresponding to the id number and name of traffic signs

The model used 60 epochs. Figure below is the plot of Model Accuracy/Loss and Epoch. The model accuracy is increasing due to the highest epoch. Similarly, the graphic shows that model loss diminishes as the epoch increases. The result of the figure can be concluded that the highest epoch will produce the highest accuracy. Figure 14 shows TS

CNN Model Accuracy vs Epoch while Figure 15 shows TS CNN Model Loss vs Epoch.

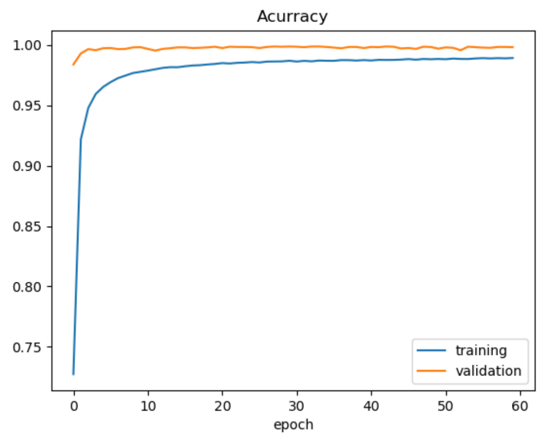


Figure 14 TS CNN Model Accuracy vs Epoch

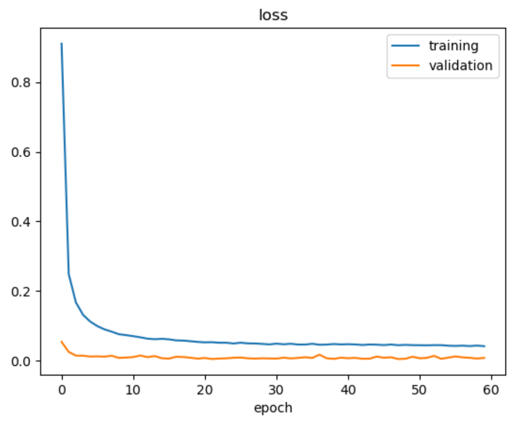


Figure 15 TS CNN Model Loss vs Epoch

4.3 Traffic Sign Detection and Classification Result

The video camera is generated. It will show the class id, names and probability of the traffic sign detected. Image of the traffic sign is showed in front of the camera, the probability in term of percentage will appear and the image is successful classified. There are 43 classes of traffic signs. The probability of detection and classification is greater than 90% as shown in the figures below.

Class No	Traffic Sign	Percentage of probability (%)
0	Speed Limit 20 km/h	96.04
1	Speed Limit 30 km/h	99.43
2	Speed Limit 50 km/h	98.55
3	Speed Limit 60 km/h	97.02
4	Speed Limit 70 km/h	96.46
5	Speed Limit 80 km/h	99.27
6	End of Speed Limit 80 km/h	98.14
7	Speed Limit 100 km/h	99.16
8	Speed Limit 120 km/h	96.35
9	No passing	99.71
10	No passing for vehicles over 3.5 metric tons	98.95
11	Right-of-way at the next intersection	99.27
12	Priority road	98.57
13	Yield	97.19
14	Stop	98.52
16	Vehicles over 3.5 metric tons prohibited	99.81
17	No entry	99.60
18	General caution	98.31

19	Dangerous curve to the left	98.96
20	Dangerous curve to the right	100.00
21	Double Curve	100.00
22	Bumpy road	97.50
23	Slippery road	96.23
24	Road narrows on the right	97.69
25	Road Work	99.38
26	Traffic Signals	99.48
27	Pedestrians	97.27
28	Children crossing	90.56
30	Beware of ice/snow	91.11
31	Wild animals crossing	98.14
32	End of all speed and passing limits	95.93
33	Turn right ahead	97.10
34	Turn left ahead	99.09
35	Ahead only	99.95
36	Go straight or right	97.38
37	Go straight or left	97.79
38	Keep right	99.85
39	Keep left	99.79
40	Roundabout mandatory	99.29
41	End of no passing	96.10
42	End of no passing by vehicles over 3.5 metric tons	98.21

Table 3 Probability of traffic sign images with class is

There are two traffic sign that can be considered as confused class in the table below.

Class id	Traffic Sign	Confused Class detected
15	No vehicles	Yield (94.99%)
29	Bicycle Crossing	Slippery Road (100.00%)

Table 4 Confuse classes detected in traffic sign identification

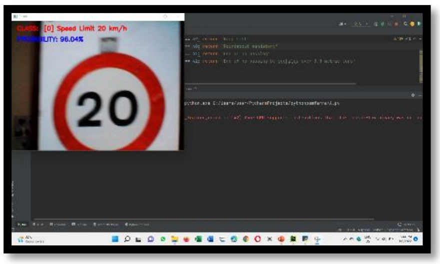


Figure 16 Class 0 with 99.43% of Probability

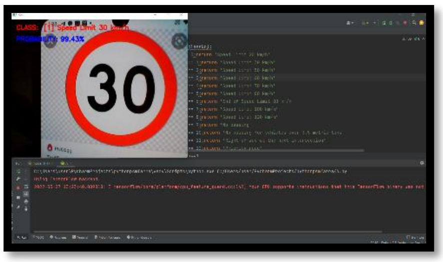


Figure 17 Class 1 with 99.43% of Probability

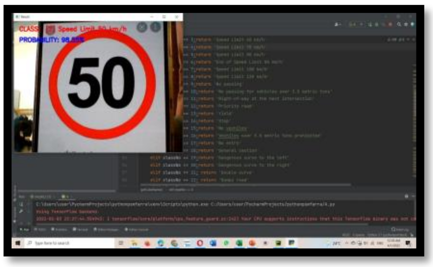


Figure 18 Class 2 with 98.55% of Probability

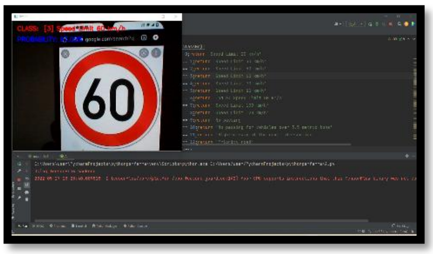


Figure 19 Class 3 with 97.20% of Probability

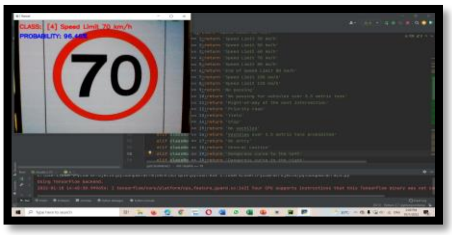


Figure 20 Class 4 with 96.46% of Probability

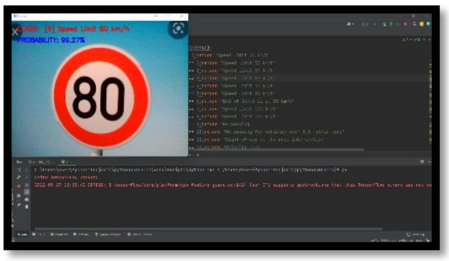


Figure 21 Class 5 with 99.27% of Probability

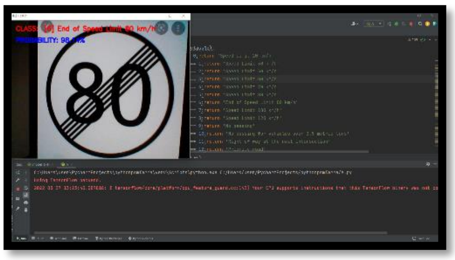


Figure 22 Class 6 with 98.14% of Probability

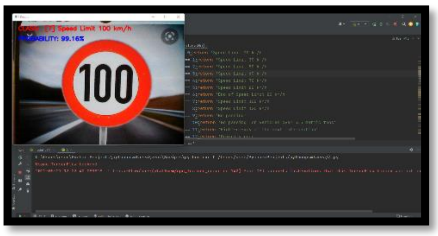


Figure 23 Class 7 with 99.16% of Probability

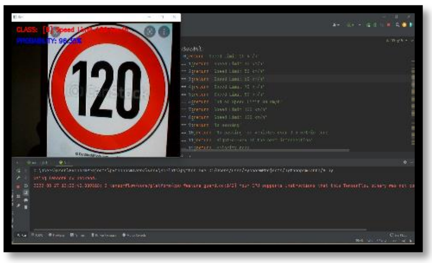


Figure 24 Class 8 with 96.35% of Probability

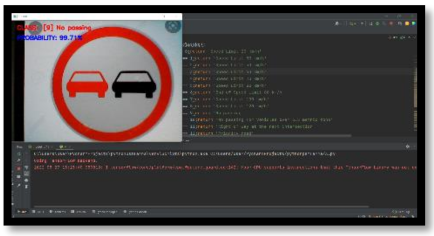


Figure 25 Class 9 with 99.17% of Probability

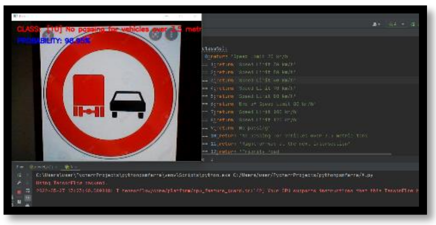


Figure 26 Class 10 with 98.95% of Probability

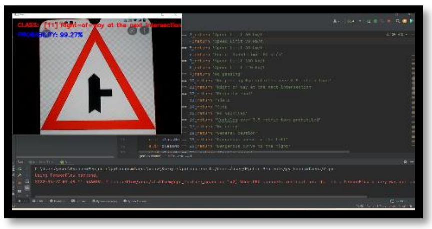


Figure 27 Class 11 with 99.27% of Probability

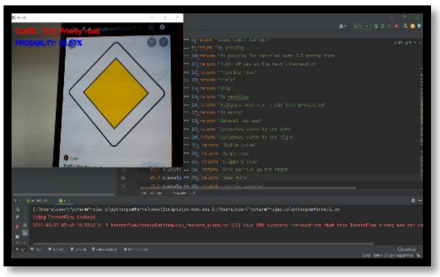


Figure 28 Class 12 with 98.57% of Probability

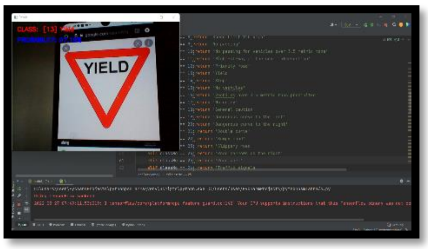


Figure 29 Class 13 with 97.17% of Probability

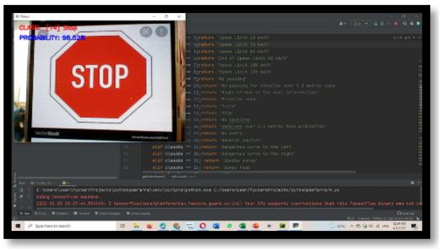


Figure 30 Class 14 with 98.52% of Probability

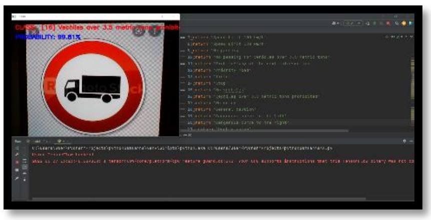


Figure 31 Class 16 with 99.81% of Probability

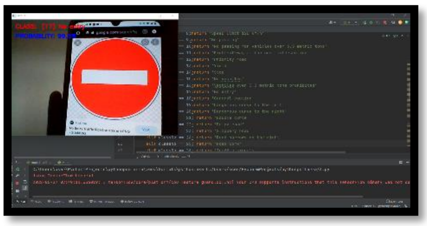


Figure 32 Class 17 with 99.60% of Probability

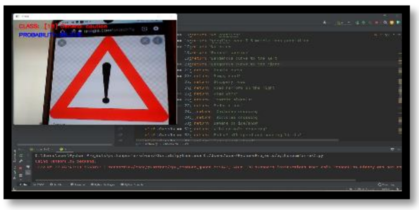


Figure 33 Class 18 with 98.31% of Probability

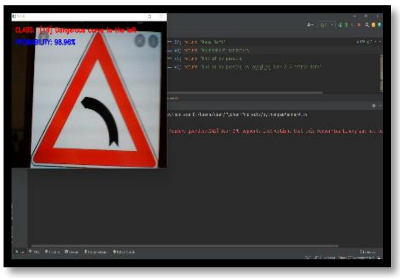


Figure 34 Class 19 with 97.50% of Probability

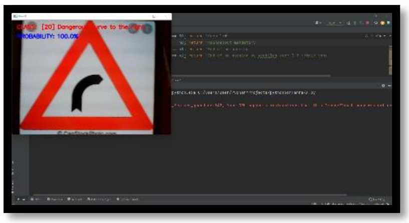


Figure 35 Class 20 with 100% of Probability

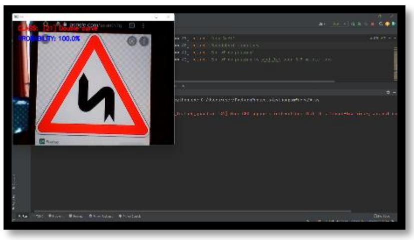


Figure 36 Class 21 with 100% of Probability

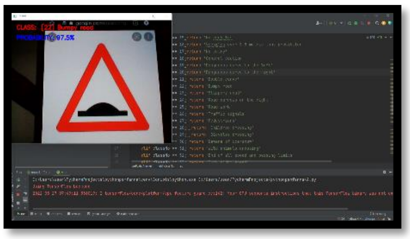


Figure 37 Class 22 with 97.50% of Probability

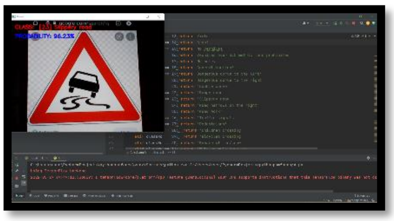


Figure 38 Class 23 with 96.23% of Probability

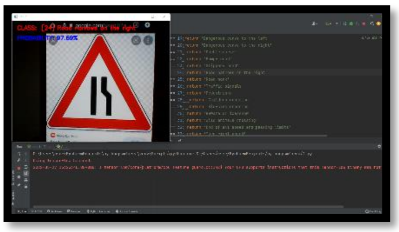


Figure 39 Class 24 with 97.69% of Probability

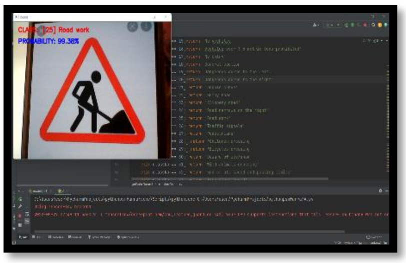


Figure 40 Class 25 with 99.38% of Probability

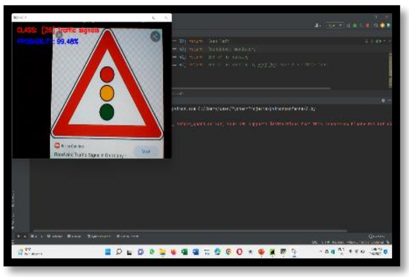


Figure 41Class 26 with 99.48% of Probability

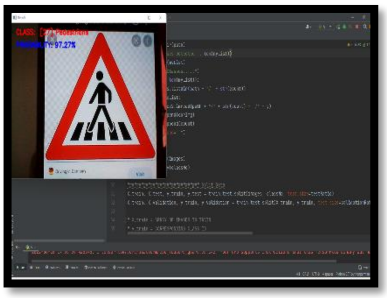


Figure 42 Class 27 with 97.27 of Probability

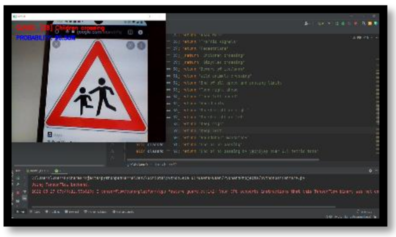


Figure 43 Class 28 with 90.56% of Probability

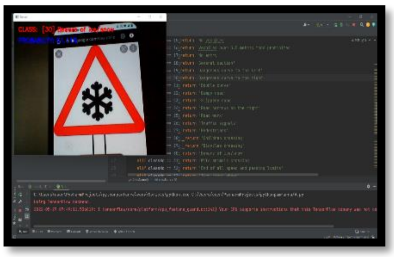


Figure 44 Class 30 with 91.11% of Probability

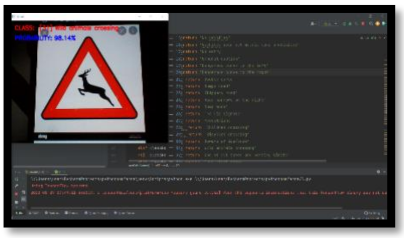


Figure 45 Class 31 with 98.14% of Probability

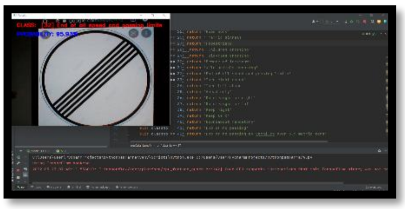


Figure 46 Class 32 with 95.93% of Probability

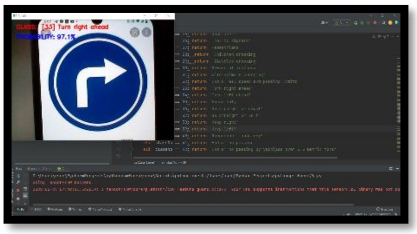


Figure 47 Class 33 with 97.01% of Probability

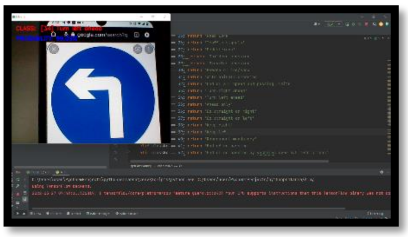


Figure 48 Class 34 with 99.09% of Probability

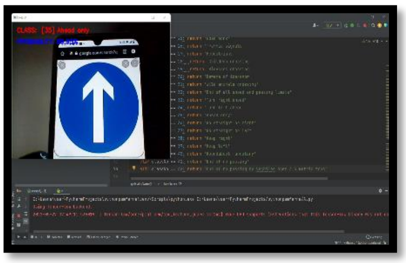


Figure 49 Class 35 with 99.95% of Probability

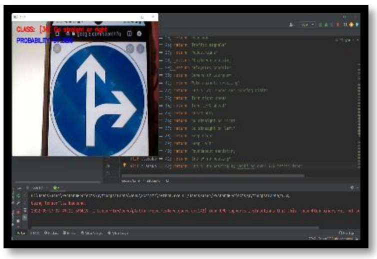


Figure 50 Class 36 with 97.38% of Probability

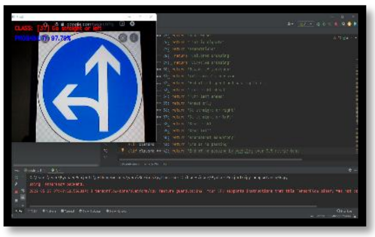


Figure 51 Class 37 with 97.79% of Probability

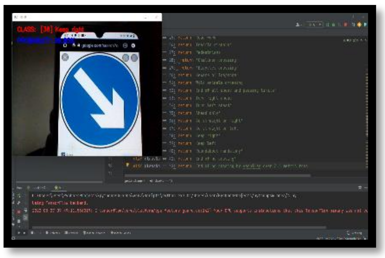


Figure 52 Class 38 with 99.85% of Probability

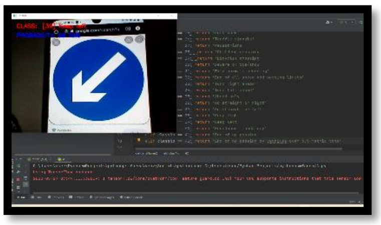


Figure 53 Class 39 with 99.79% of Probability

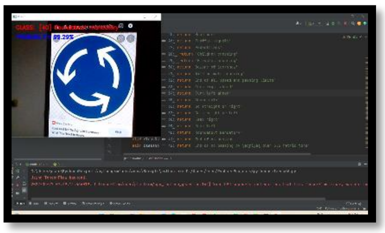


Figure 54 Class 40 with 99.29% of Probability

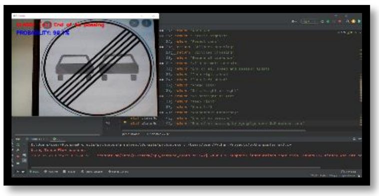


Figure 55 Class 41 with 96.01% of Probability

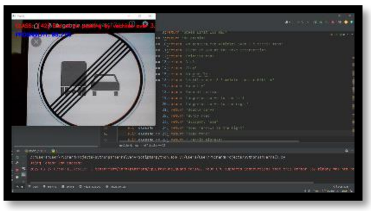


Figure 56 Class 42 with 98.21% of Probability

There are two traffic signs that detected as confused class as shown below in Figure 51 and Figure 52. No vehicle sign is detected as a yield sign while bicycle crossing sign is detected as slippery road sign.

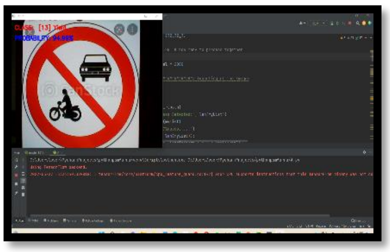


Figure 57 Class 15 No Vehicle Sign

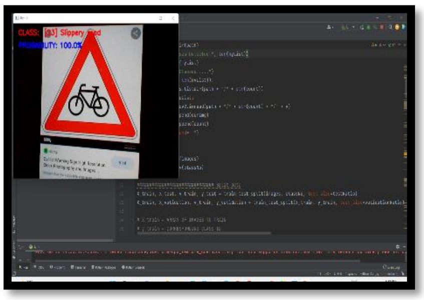


Figure 58 Class 29 Bicycle Crossing Sign

4.5 Comparison Method

From the GTSRB dataset, the accuracy of TS CNN Model was tested utilizing 12630 images. It was found to be 98.44 percent in the previous solution. There are many methods were compared to this method. VGG16 model [18] distributing an accuracy of 74.5%, whereas CVOG + other features [19] model achieving an accuracy of 94.73%. Capsule Network [20] attained an accuracy of 97.6%. GoogLeNet based CNN [21] accomplished an accuracy of 98%. Hough transform with CNN [11] delivered an accuracy of 98.2%. Table 3 exhibits the model and accuracy obtained and it was compared to the method that used in this study. It is obvious that the suggested model beats all previous comparisons.

MODEL	ACCURACY
VGG16[2]	74.5%
CVOG + other features [3]	94.73%
Capsule Network [4]	97.6%
GoogLeNet based CNN [5]	98%
Hough transform with CNN [6]	98.2%
Proposed Model	**99.80%**

Table 5 A comparative study with existing method [9]

4.5.1 Critical Analysis

This part of the chapter is the analysis part where the hypothesis is gained. Table 4 below shows the comparison between graph, accuracy, value of epoch, and description about the probability will be discussed. The accuracy obtained from the coding for the project solution is 99.80% while for previous solution is 98.44%. After running the coding, accuracy is 0.9892 and loss is 0.0418. the validation accuracy is extremely good which is 0.9982 while validation loss is 0.0081. the test score is 0.0083 while test accuracy is 0.9980. Overall, our data is well.

In the coding, there are about 22000 images for training, around 5000 images for validation, and about 7000 images for testing. The data shape should be 32 by 32 by 3 which means 3 channels. Therefore, if we have 22000 images, we should have 22000 ids that correspond to where it belongs. During the traffic sign images scanning to the camera, several challenges were facing where two traffic signs have mistakenly detect the other traffic sign. Therefore, if the distribution data is examined, class 15 and 29 are referred to the no vehicle sign and bicycle crossing respectively is below than 500. Any dataset below 500, it will give us not good probability. Highest data of images, highest probability of detection, and easy to classify the object according to class id. Based on the dataset, class number 14 which is stop image has about 1000 images which give us a good result. Class 1, 2 and 3 should have high detection rates with high probability due to the lot of data.

Graph for different epoch has different result. Graph for 60 of epoch and 50 of epoch is compared in term of loss and accuracy for validation and training data. The most important is threshold value. This is the percentage which it will accept it as a classification. In this way, if we have more than 90% of probability, it will accept it as a class being detected.

Variable	Project solution	Previous solution
Accuracy	99.80% of accuracy (Accuracy is obtained after running the coding)	98.44% of accuracy (Accuracy is obtained after running the coding)
Probability	Challenges: lightning conditions, complex traffic sign images, and shaking during the traffic sign images are captured (even though already used equipment to hold it), distance, complex shape, or too many colors in the traffic sign images. Occurred to 2 traffic sign images (No vehicle, bicycle crossing) Problems have been experienced by both solutions for a different traffic sign. Highest data of images, highest probability of detection, and easy to classify the object according to class id.	
Epoch Value	60	50
Graph	Different graphs due to different values of the epoch. The higher value of epoch will increase the accuracy and decrease the loss of test data.	
Threshold Value	90%	

Table 6 The Analysis and Comparison

4.5.2 Analysis on Image Traffic Sign Classification based on value of epoch

The epoch value that used in this study is 60 with 2000 of iteration while the value of batch size is 128 in the training code. Firstly, the training code was run and the output from coding obtained which are the distribution of the training dataset, images corresponding to the id number and name of traffic signs, and graph of accuracy and loss as can see in the figure 11, figure 12, figure 13, and figure 14. After that, the final value of accuracy is successfully produced from the simulation. The value of accuracy that acquired is 99.80% compared to previous method which are 98.44% of accuracy. This is because paper that was referred was used 50 of epoch value. The hypothesis that can be achieved is the higher value of epoch will increase the accuracy and decrease the loss. Nevertheless, boosting epochs makes sense only if the dataset contains a large amount of data. Eventually, the model will reach a point when adding more epochs will not enhance accuracy.

The graph is produced for accuracy and loss for 2000 iteration in 60 epoch. The graph shown the accuracy and loss for training and validation test. The graph obtained as can see in figure 55 and figure 56. Next, graph for 50 of epoch is shown in figure 56 and figure 57 to exhibit the difference. Lastly, the graph for 10 of epoch is shown in figure 54 and figure 55 where these graphs were obtained at the first trial of my simulation.

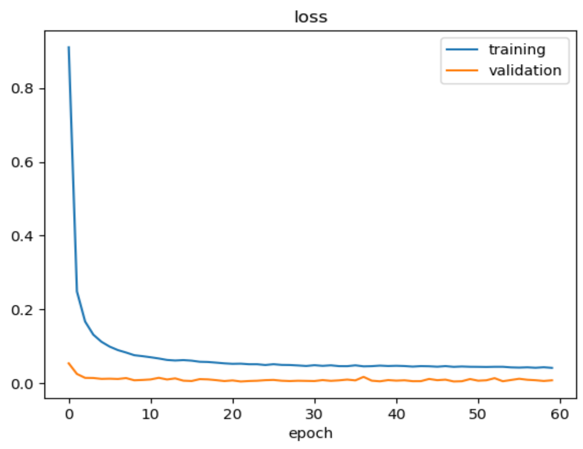


Figure 59 Loss Graph for 60 epoch

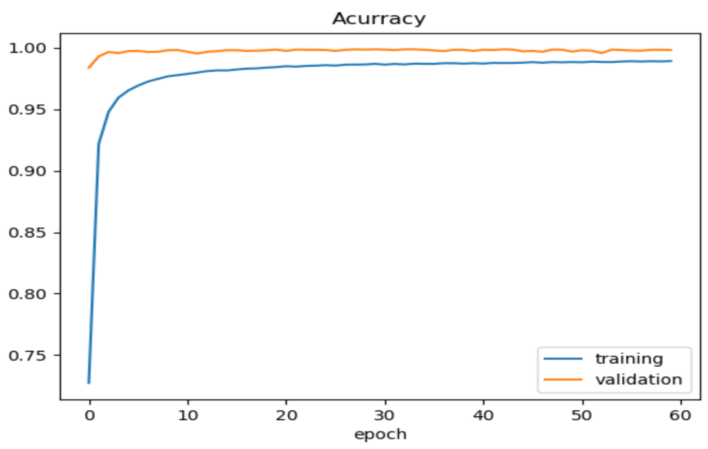


Figure 60 Accuracy graph for 60 epoch

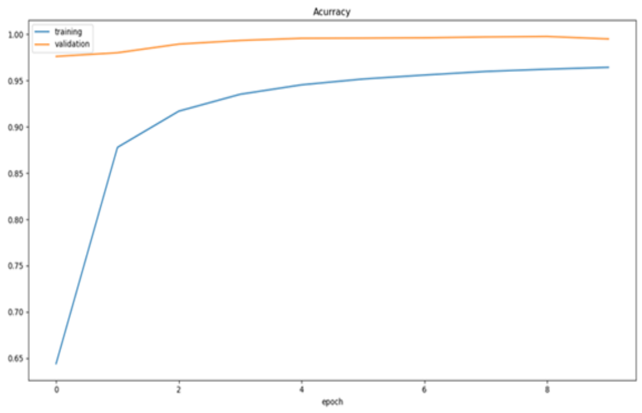


Figure 61 Accuracy graph for 10 epoch

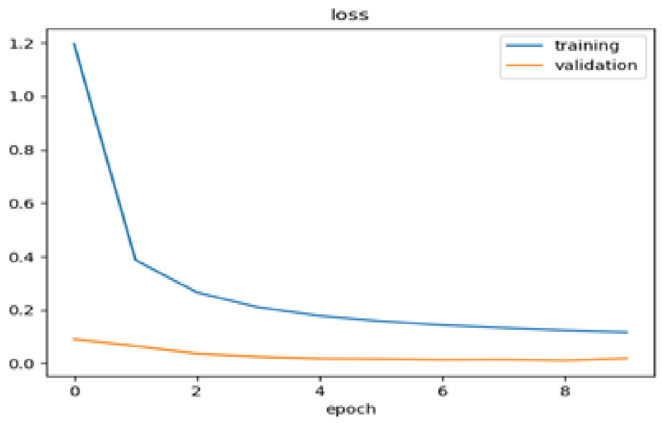


Figure 62 Loss graph for 10 epoc

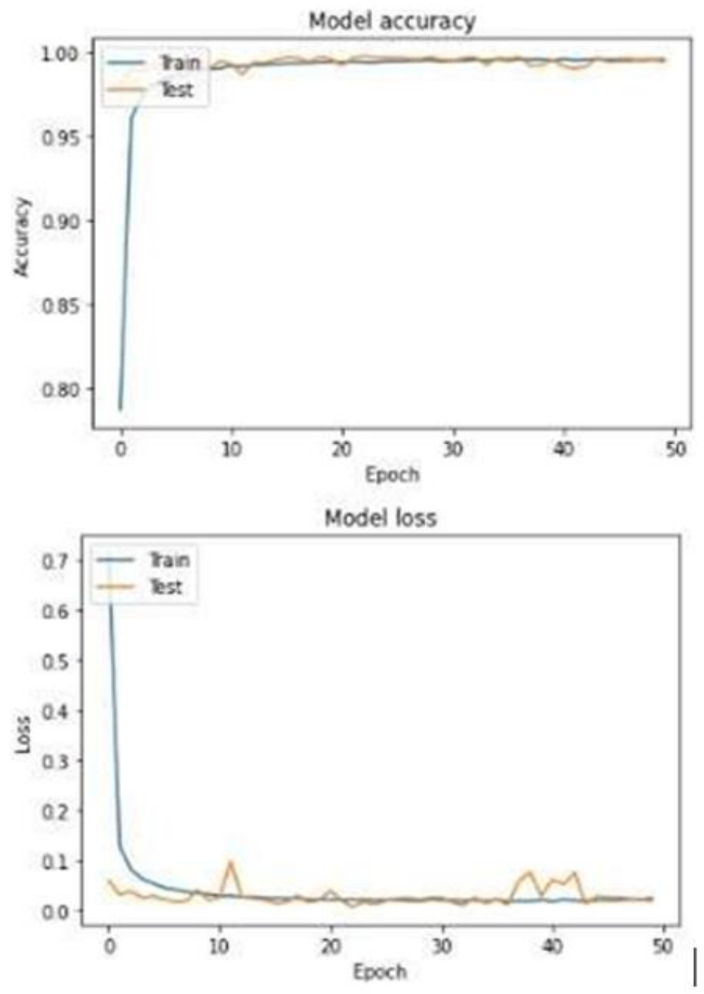
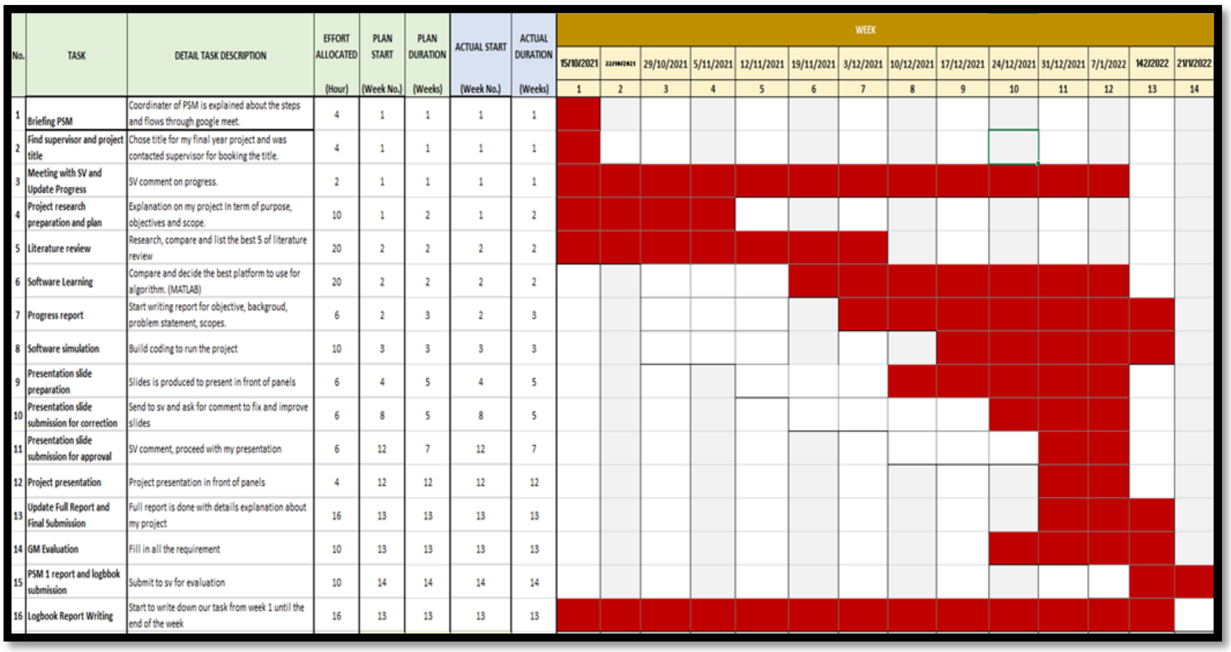


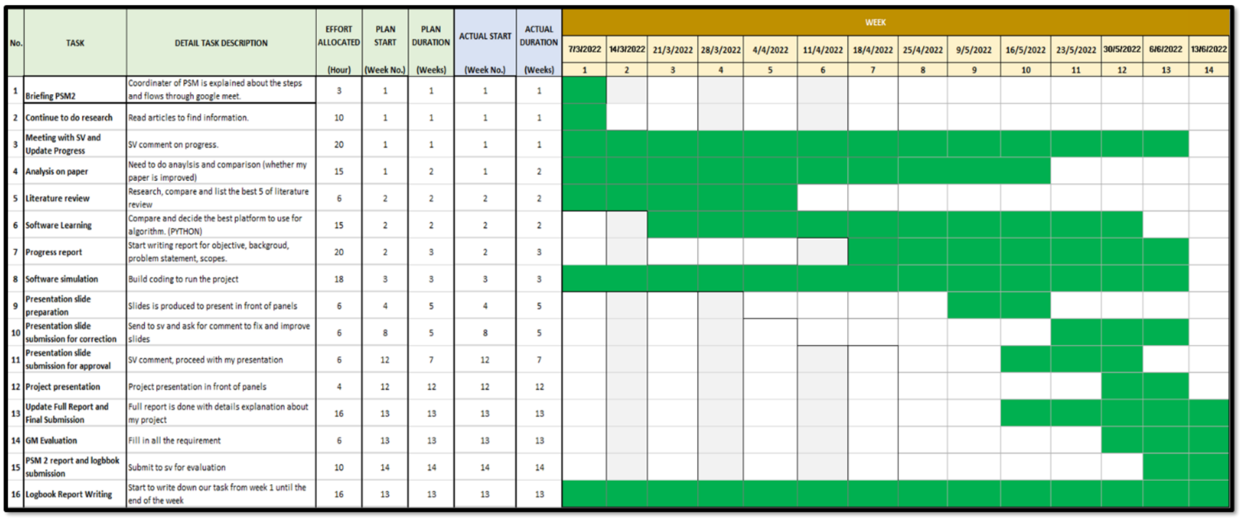
Figure 63 Graph for 50 of epoch in the previous solution [7]

4.6 Gant Chart



No.	TASK	DETAIL TASK DESCRIPTION	EFFORT ALLOCATED (Hour)	PLAN START (Week No.)	PLAN DURATION (Weeks)	ACTUAL START (Week No.)	ACTUAL DURATION (Weeks)	WEEK													
								15/10/2021	22/10/2021	29/10/2021	5/11/2021	12/11/2021	19/11/2021	3/12/2021	10/12/2021	17/12/2021	24/12/2021	31/12/2021	7/1/2022	14/1/2022	21/1/2022
								1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Briefing PSM	Coordinater of PSM is explained about the steps and flows through google meet.	4	1	1	1	1	🟥													
2	Find supervisor and project title	Chose title for my final year project and was contacted supervisor for booking the title.	4	1	1	1	1	🟥													
3	Meeting with SV and Update Progress	SV comment on progress.	2	1	1	1	1	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥		
4	Project research preparation and plan	Explanation on my project In term of purpose, objectives and scope.	10	1	2	1	2	🟥	🟥	🟥	🟥										
5	Literature review	Research, compare and list the best 5 of literature review	20	2	2	2	2	🟥	🟥	🟥	🟥	🟥	🟥	🟥							
6	Software Learning	Compare and decide the best platform to use for algorithm. (MATLAB)	20	2	2	2	2						🟥	🟥	🟥	🟥	🟥	🟥	🟥		
7	Progress report	Start writing report for objective, backgroud, problem statement, scopes.	6	2	3	2	3							🟥	🟥	🟥	🟥	🟥	🟥	🟥	
8	Software simulation	Build coding to run the project	10	3	3	3	3									🟥	🟥	🟥	🟥	🟥	
9	Presentation slide preparation	Slides is produced to present in front of panels	6	4	5	4	5								🟥	🟥	🟥	🟥	🟥		
10	Presentation slide submission for correction	Send to sv and ask for comment to fix and improve slides	6	8	5	8	5										🟥	🟥	🟥		
11	Presentation slide submission for approval	SV comment, proceed with my presentation	6	12	7	12	7											🟥	🟥		
12	Project presentation	Project presentation in front of panels	4	12	12	12	12											🟥	🟥		
13	Update Full Report and Final Submission	Full report is done with details explanation about my project	16	13	13	13	13											🟥	🟥	🟥	
14	GM Evaluation	Fill in all the requirement	10	13	13	13	13										🟥	🟥	🟥	🟥	
15	PSM 1 report and logbbok submission	Submit to sv for evaluation	10	14	14	14	14													🟥	🟥
16	Logbook Report Writing	Start to write down our task from week 1 until the end of the week	16	13	13	13	13	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	

Figure 64 Gant Chart of FYP 1



No.	TASK	DETAIL TASK DESCRIPTION	EFFORT ALLOCATED (Hour)	PLAN START (Week No.)	PLAN DURATION (Weeks)	ACTUAL START (Week No.)	ACTUAL DURATION (Weeks)	WEEK													
								7/3/2022	14/3/2022	21/3/2022	28/3/2022	4/4/2022	11/4/2022	18/4/2022	25/4/2022	9/5/2022	16/5/2022	23/5/2022	30/5/2022	6/6/2022	13/6/2022
								1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Briefing PSM2	Coordinater of PSM is explained about the steps and flows through google meet.	3	1	1	1	1	🟩													
2	Continue to do research	Read articles to find information.	10	1	1	1	1	🟩													
3	Meeting with SV and Update Progress	SV comment on progress.	20	1	1	1	1	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	
4	Analysis on paper	Need to do anaylsis and comparison (whether my paper is improved)	15	1	2	1	2	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩				
5	Literature review	Research, compare and list the best 5 of literature review	6	2	2	2	2	🟩	🟩	🟩	🟩	🟩									
6	Software Learning	Compare and decide the best platform to use for algorithm. (PYTHON)	15	2	2	2	2			🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		
7	Progress report	Start writing report for objective, backgroud, problem statement, scopes.	20	2	3	2	3							🟩	🟩	🟩	🟩	🟩	🟩	🟩	
8	Software simulation	Build coding to run the project	18	3	3	3	3	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	
9	Presentation slide preparation	Slides is produced to present in front of panels	6	4	5	4	5									🟩	🟩				
10	Presentation slide submission for correction	Send to sv and ask for comment to fix and improve slides	6	8	5	8	5											🟩	🟩	🟩	
11	Presentation slide submission for approval	SV comment, proceed with my presentation	6	12	7	12	7										🟩	🟩	🟩		
12	Project presentation	Project presentation in front of panels	4	12	12	12	12												🟩	🟩	
13	Update Full Report and Final Submission	Full report is done with details explanation about my project	16	13	13	13	13										🟩	🟩	🟩	🟩	🟩
14	GM Evaluation	Fill in all the requirement	6	13	13	13	13												🟩	🟩	🟩
15	PSM 2 report and logbbok submission	Submit to sv for evaluation	10	14	14	14	14													🟩	🟩
16	Logbook Report Writing	Start to write down our task from week 1 until the end of the week	16	13	13	13	13	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩

Figure 65 Gant Chart of FYP2

CHAPTER 5

CONCLUSION

5.1 Introduction

In this chapter, the whole procedure for the techniques used will be reviewed, and based on the output produced, the method's use may be concluded, as well as some advised actions for further consideration, because this project is lacking in its own manner.

5.2 Conclusion

This research employed Traffic Sign Convolution Neural Network model to classify the traffic signs on GTSRB dataset contains 43 classes of images. The simulation was conducted using Python version 3.7 as the platform to interpret the coding, and PyCharm to run the algorithm. Data preprocessing techniques were applied to the dataset to improve contrast and resolution. To avoid overfitting, Data Augmentation is comes after Data Preprocessing, which expand the amount of images in the training set. On the test data, the model had a significantly higher accuracy of 99.80%. For training, the Adam optimizer is utilized, and it has been discovered to adapt faster than other optimizers. Because Softmax calculates the probability of each potential class, it is used as an activation at an output layer in the suggested network architecture. The obtained result of accuracy is compared to the previous algorithm and our algorithm outperforms current state-of-the-art algorithms.

This algorithm is implemented to notice accurate classification of various traffic signs since most of the problem traffic sign classification is greatly influenced by the uncontrollable factors. We conclude that the experimental results obtained show that the proposed method can attain high classification accuracy and detection rate in traffic sign classification.

5.3 Problem encountered and Recommendation

In the future, the aim is to improve the accuracy of the traffic sign classification by using CNN variation to operate in real time in Malaysia by using our own dataset. Image of traffic signs in Malaysia will be considered and gathered to ensure the self-driving car able to operate successful. There are some problems encountered during the detection and classification using Traffic Sign Convolution Neural Network. Firstly, sometimes it mistakenly detects traffic sign where it wrongly displays the name of the traffic sign. It is happening to complex traffic sign such as no vehicle sign and bicycle crossing sign. The second problem is it does not detect the traffic sign due to the condition of the traffic sign whether it is dark, weather, degree of illumination and environmental condition. Subsequently, new algorithm will be explored and implemented to compare the results in terms of probability, accuracy, and loss. The higher dataset will produce better probability of detection and accuracy .

REFERENCES

[1] K. S. Wickramasinghe and G. U. Ganegoda, “Pedestrian detection, tracking, counting, waiting time calculation and trajectory detection for pedestrian crossings traffic light systems,” *20th Int. Conf. Adv. ICT Emerg. Reg. ICTer 2020 - Proc.*, no. ICTer, pp. 172–177, 2020, doi: 10.1109/ICTer51097.2020.9325479.

[2] S. Liu, “A traffic sign image recognition and classification approach based on convolutional neural network,” *Proc. - 2019 11th Int. Conf. Meas. Technol. Mechatronics Autom. ICMTMA 2019*, pp. 408–411, 2019, doi: 10.1109/ICMTMA.2019.00096.

[3] D. R. Bruno and F. S. Osorio, “Image classification system based on deep learning applied to the recognition of traffic signs for intelligent robotic vehicle navigation purposes,” *Proc. - 2017 LARS 14th Lat. Am. Robot. Symp. 2017 5th SBR Brazilian Symp. Robot. LARS-SBR 2017 - Part Robot. Conf. 2017*, vol. 2017-Decem, pp. 1–6, 2017, doi: 10.1109/SBR-LARS-R.2017.8215287.

[4] R. Kulkarni, S. Dhavalikar, and S. Bangar, “Traffic Light Detection and Recognition for Self Driving Cars Using Deep Learning,” *Proc. - 2018 4th Int. Conf. Comput. Commun. Control Autom. ICCUBEA 2018*, pp. 1–4, 2018, doi: 10.1109/ICCUBEA.2018.8697819.

[5] S. Mehta, C. Paunwala, and B. Vaidya, “CNN based traffic sign classification using adam optimizer,” *2019 Int. Conf. Intell. Comput. Control Syst. ICCS 2019*, no. Iciccs, pp. 1293–1298, 2019, doi: 10.1109/ICCS45141.2019.9065537.

[6] C. F. Paulo and P. L. Correia, “Automatic detection and classification of traffic signs,” *8th Int. Work. Image Anal. Multimed. Interact. Serv. WIAMIS 2007*, pp. 1–4, 2007, doi: 10.1109/WIAMIS.2007.24.

[7] M. A. Vincent, K. R. Vidya, and S. P. Mathew, “Traffic Sign Classification Using Deep Neural Network,” *2020 IEEE Recent Adv. Intell. Comput. Syst. RAICS 2020*, pp. 13–17, 2020, doi: 10.1109/RAICS51191.2020.9332474.

[8] V. N. Sichkar and A. V. Lyamin, “Design of Deep CNN Model for Effective Traffic Signs Recognition,” *Proc. - 2021 Int. Russ. Autom. Conf. RusAutoCon 2021*, pp. 367–373, 2021, doi: 10.1109/RusAutoCon52004.2021.9537445.

[9] Y. Yang, H. Luo, H. Xu, and F. Wu, “Towards Real-Time Traffic Sign Detection and Classification,” *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 7, pp. 2022–2031, 2016, doi: 10.1109/TITS.2015.2482461.

[10] Y. Zhang, Z. Wang, Y. Qi, J. Liu, and J. Yang, “CTSD: A Dataset for Traffic Sign Recognition in Complex Real-World Images,” *VCIP 2018 - IEEE Int. Conf. Vis. Commun. Image Process.*, pp. 18–21, 2018, doi: 10.1109/VCIP.2018.8698666.

[11] A. Golchubian, O. Marques, and M. Nojoumian, “Photo quality classification using deep learning,” *Multimed. Tools Appl.*, vol. 80, no. 14, pp. 22193–22208, 2021, doi: 10.1007/s11042-021-10766-7.

[12] M. A. A. Sheikh, A. Kole, and T. Maity, “Traffic sign detection and classification using colour feature and neural network,” *2016 Int. Conf. Intell. Control. Power Instrumentation, ICICPI 2016*, pp. 307–311, 2017, doi: 10.1109/ICICPI.2016.7859723.

[13] J. Akbar, M. Shahzad, M. I. Malik, A. Ul-Hasan, and F. Shafait, “Runway Detection and Localization in Aerial Images using Deep Learning,” *2019 Digit. Image Comput. Tech. Appl. DICTA 2019*, pp. 1–8, 2019, doi: 10.1109/DICTA47822.2019.8945889.

[14] M. Cetinkaya and T. Acarman, “Traffic sign detection by image preprocessing and deep learning,” *Proc. - 5th Int. Conf. Intell. Comput. Control Syst. ICICCS 2021*, no. Iciccs, pp. 1165–1170, 2021, doi: 10.1109/ICICCS51141.2021.9432088.

[15] Y. Swapna, M. S. Reddy, J. V. Sai, N. S. S. Krishna, and M. V. Teja, “Deep Learning Based Road Traffic Sign Detection and Recognition,” *Proc. 3rd Int. Conf. Inven. Res. Comput. Appl. ICIRCA 2021*, pp. 1–4, 2021, doi: 10.1109/ICIRCA51532.2021.9545080.

[16] C. Wang, “Research and application of traffic sign detection and recognition based on deep learning,” *Proc. - 2018 Int. Conf. Robot. Intell. Syst. ICRIS 2018*, pp. 150–152, 2018, doi: 10.1109/ICRIS.2018.00047.

[17] R. Kulkarni, S. Dhavalikar, and S. Bangar, “Traffic Light Detection and Recognition for Self

Driving Cars Using Deep Learning," *Proc. - 2018 4th Int. Conf. Comput. Commun. Control Autom. ICCUBEA 2018*, pp. 2–5, 2018, doi: 10.1109/ICCUBEA.2018.8697819.

[18] H. S. Dikbayir and H. Ibrahim Bulbul, "Deep Learning Based Vehicle Detection from Aerial Images," *Proc. - 19th IEEE Int. Conf. Mach. Learn. Appl. ICMLA 2020*, pp. 956–960, 2020, doi: 10.1109/ICMLA51294.2020.00155.

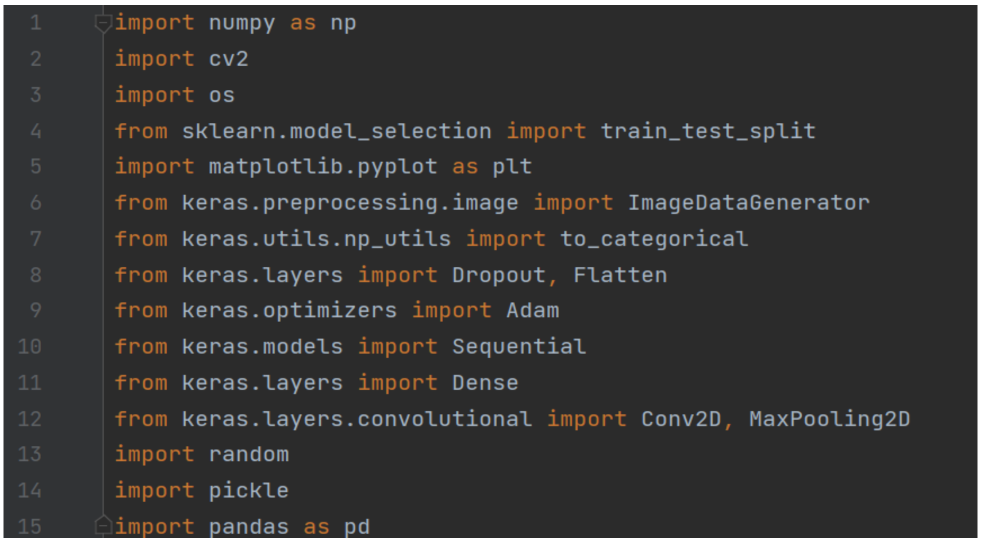
[19] K. B. Lee and H. S. Shin, "An Application of a Deep Learning Algorithm for Automatic Detection of Unexpected Accidents under Bad CCTV Monitoring Conditions in Tunnels," *Proc. - 2019 Int. Conf. Deep Learn. Mach. Learn. Emerg. Appl. Deep. 2019*, pp. 7–11, 2019, doi: 10.1109/Deep-ML.2019.00010.

[20] Y. He and W. Li, "Image-based encrypted traffic classification with convolution neural networks," *Proc. - 2020 IEEE 5th Int. Conf. Data Sci. Cyberspace, DSC 2020*, pp. 271–278, 2020, doi: 10.1109/DSC50466.2020.00048.

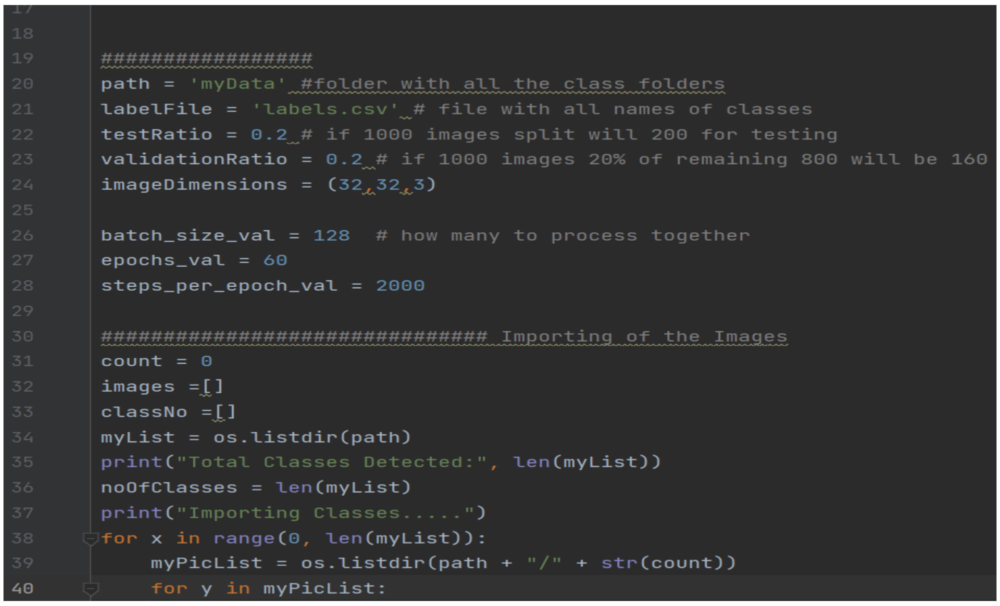
[21] D. Yasmina, R. Karima, and A. Ouahiba, "Traffic signs recognition with deep learning," *Proc. 2018 Int. Conf. Appl. Smart Syst. ICASS 2018*, no. November, pp. 1–5, 2019, doi: 10.1109/ICASS.2018.8652024.

APPENDIX A
SAMPLE APPENDIX 1

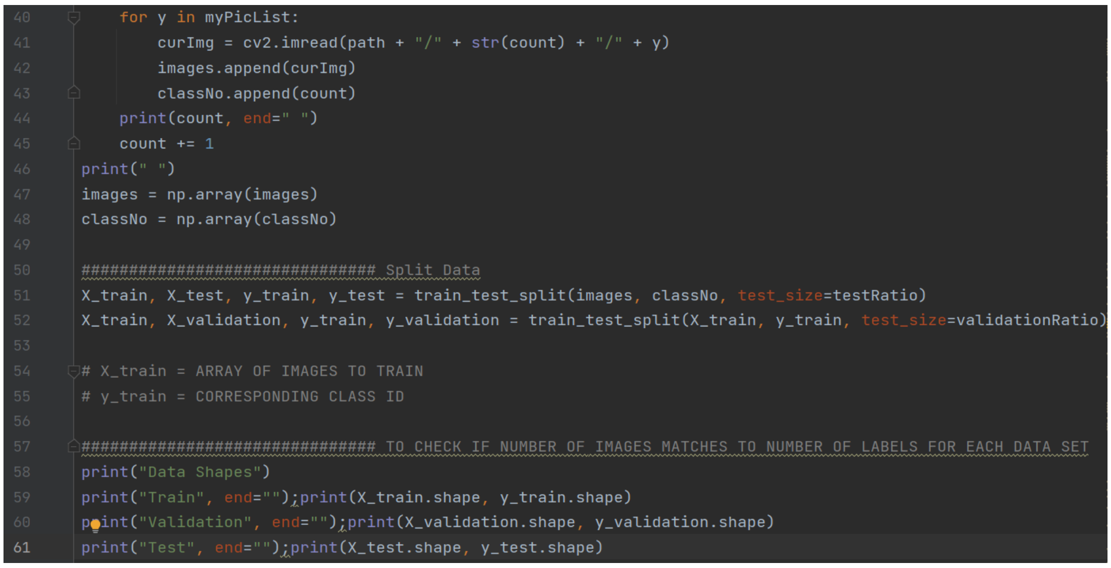
1. CODES FOR IMAGE DETECTION



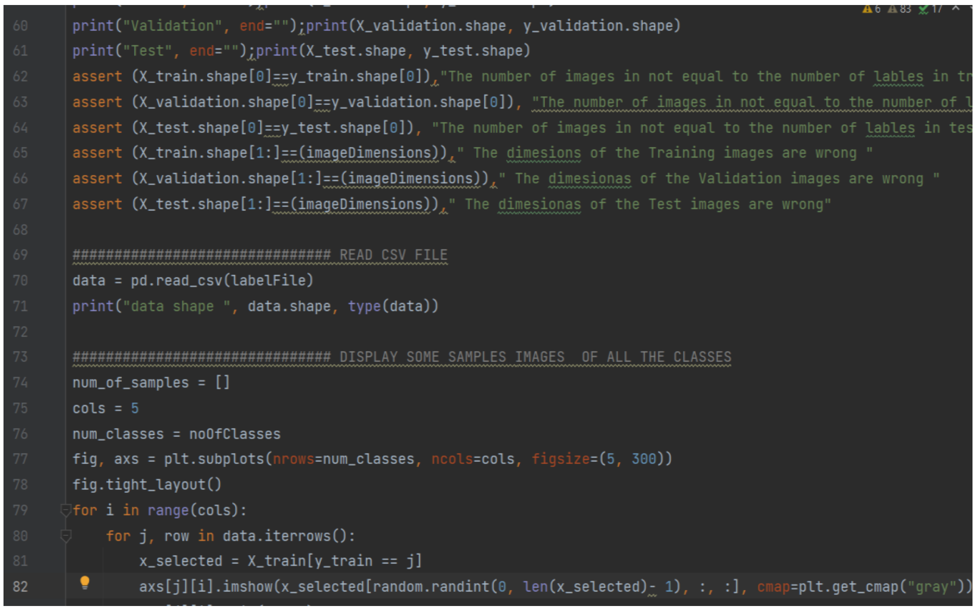
```python
import numpy as np
import cv2
import os
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from keras.preprocessing.image import ImageDataGenerator
from keras.utils.np_utils import to_categorical
from keras.layers import Dropout, Flatten
from keras.optimizers import Adam
from keras.models import Sequential
from keras.layers import Dense
from keras.layers.convolutional import Conv2D, MaxPooling2D
import random
import pickle
import pandas as pd
```



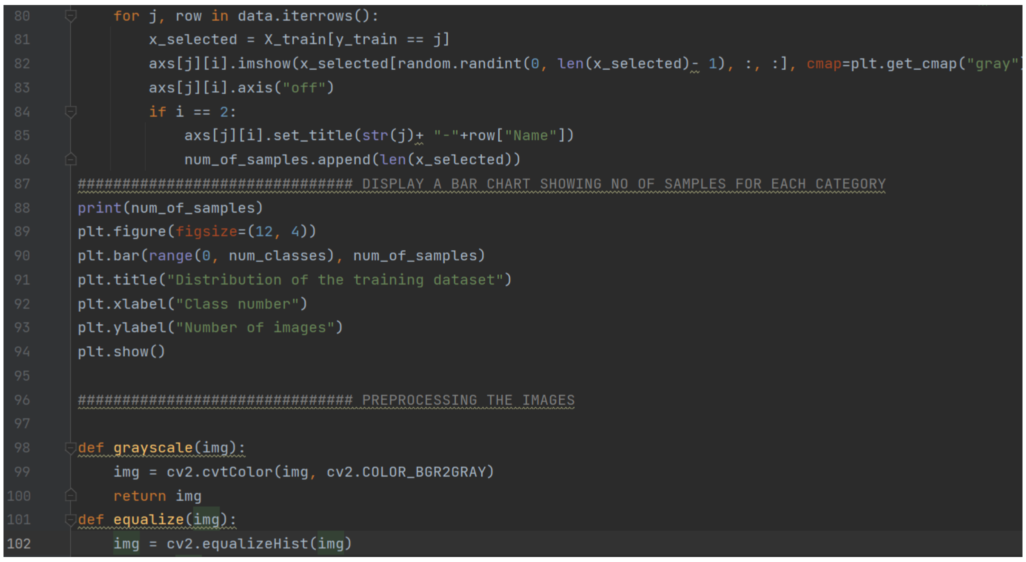
```python


#################
path = 'myData' #folder with all the class folders
labelFile = 'labels.csv' # file with all names of classes
testRatio = 0.2 # if 1000 images split will 200 for testing
validationRatio = 0.2 # if 1000 images 20% of remaining 800 will be 160
imageDimensions = (32,32,3)

batch_size_val = 128  # how many to process together
epochs_val = 60
steps_per_epoch_val = 2000

############################### Importing of the Images
count = 0
images =[]
classNo =[]
myList = os.listdir(path)
print("Total Classes Detected:", len(myList))
noOfClasses = len(myList)
print("Importing Classes.....")
for x in range(0, len(myList)):
    myPicList = os.listdir(path + "/" + str(count))
    for y in myPicList:
```



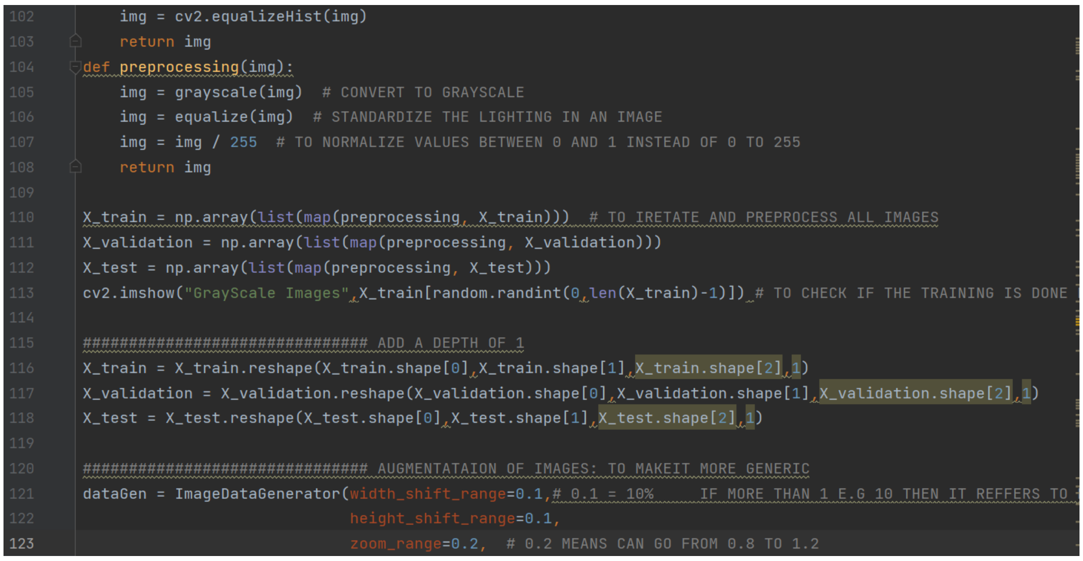
```python
    for y in myPicList:
        curImg = cv2.imread(path + "/" + str(count) + "/" + y)
        images.append(curImg)
        classNo.append(count)
    print(count, end=" ")
    count += 1
print(" ")
images = np.array(images)
classNo = np.array(classNo)

################################ Split Data
X_train, X_test, y_train, y_test = train_test_split(images, classNo, test_size=testRatio)
X_train, X_validation, y_train, y_validation = train_test_split(X_train, y_train, test_size=validationRatio)

# X_train = ARRAY OF IMAGES TO TRAIN
# y_train = CORRESPONDING CLASS ID

################################ TO CHECK IF NUMBER OF IMAGES MATCHES TO NUMBER OF LABELS FOR EACH DATA SET
print("Data Shapes")
print("Train", end="");print(X_train.shape, y_train.shape)
print("Validation", end="");print(X_validation.shape, y_validation.shape)
print("Test", end="");print(X_test.shape, y_test.shape)
```



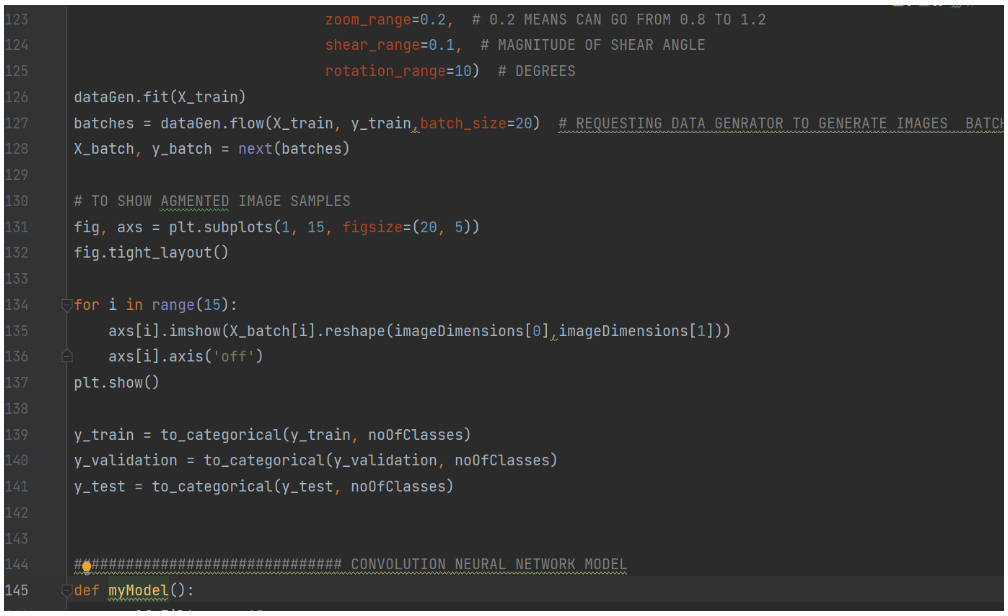
```python
print("Validation", end="");print(X_validation.shape, y_validation.shape)
print("Test", end="");print(X_test.shape, y_test.shape)
assert (X_train.shape[0]==y_train.shape[0]),"The number of images in not equal to the number of lables in tr
assert (X_validation.shape[0]==y_validation.shape[0]), "The number of images in not equal to the number of l
assert (X_test.shape[0]==y_test.shape[0]), "The number of images in not equal to the number of lables in tes
assert (X_train.shape[1:]==(imageDimensions))," The dimesions of the Training images are wrong "
assert (X_validation.shape[1:]==(imageDimensions))," The dimesionas of the Validation images are wrong "
assert (X_test.shape[1:]==(imageDimensions))," The dimesionas of the Test images are wrong"

################################ READ CSV FILE
data = pd.read_csv(labelFile)
print("data shape ", data.shape, type(data))

################################ DISPLAY SOME SAMPLES IMAGES  OF ALL THE CLASSES
num_of_samples = []
cols = 5
num_classes = noOfClasses
fig, axs = plt.subplots(nrows=num_classes, ncols=cols, figsize=(5, 300))
fig.tight_layout()
for i in range(cols):
    for j, row in data.iterrows():
        x_selected = X_train[y_train == j]
        axs[j][i].imshow(x_selected[random.randint(0, len(x_selected)- 1), :, :], cmap=plt.get_cmap("gray"))
```



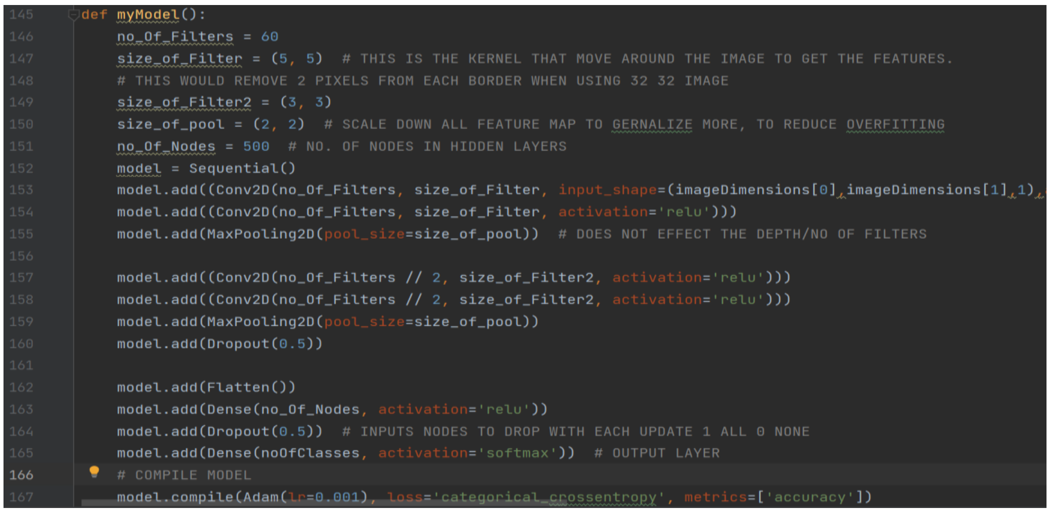
```python
    for j, row in data.iterrows():
        x_selected = X_train[y_train == j]
        axs[j][i].imshow(x_selected[random.randint(0, len(x_selected)- 1), :, :], cmap=plt.get_cmap("gray"
        axs[j][i].axis("off")
        if i == 2:
            axs[j][i].set_title(str(j)+ "-"+row["Name"])
            num_of_samples.append(len(x_selected))
################################ DISPLAY A BAR CHART SHOWING NO OF SAMPLES FOR EACH CATEGORY
print(num_of_samples)
plt.figure(figsize=(12, 4))
plt.bar(range(0, num_classes), num_of_samples)
plt.title("Distribution of the training dataset")
plt.xlabel("Class number")
plt.ylabel("Number of images")
plt.show()

################################ PREPROCESSING THE IMAGES

def grayscale(img):
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    return img
def equalize(img):
    img = cv2.equalizeHist(img)
```



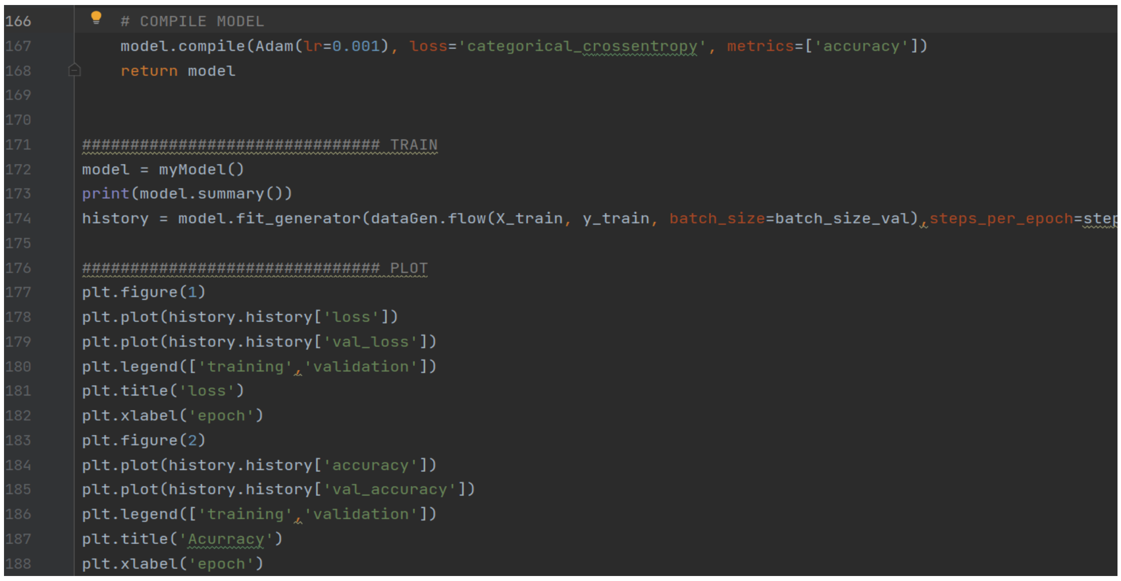
```
    img = cv2.equalizeHist(img)
    return img
def preprocessing(img):
    img = grayscale(img)  # CONVERT TO GRAYSCALE
    img = equalize(img)  # STANDARDIZE THE LIGHTING IN AN IMAGE
    img = img / 255  # TO NORMALIZE VALUES BETWEEN 0 AND 1 INSTEAD OF 0 TO 255
    return img

X_train = np.array(list(map(preprocessing, X_train)))  # TO IRETATE AND PREPROCESS ALL IMAGES
X_validation = np.array(list(map(preprocessing, X_validation)))
X_test = np.array(list(map(preprocessing, X_test)))
cv2.imshow("GrayScale Images",X_train[random.randint(0,len(X_train)-1)]) # TO CHECK IF THE TRAINING IS DONE

################################ ADD A DEPTH OF 1
X_train = X_train.reshape(X_train.shape[0],X_train.shape[1],X_train.shape[2],1)
X_validation = X_validation.reshape(X_validation.shape[0],X_validation.shape[1],X_validation.shape[2],1)
X_test = X_test.reshape(X_test.shape[0],X_test.shape[1],X_test.shape[2],1)

################################ AUGMENTATAION OF IMAGES: TO MAKEIT MORE GENERIC
dataGen = ImageDataGenerator(width_shift_range=0.1,# 0.1 = 10%     IF MORE THAN 1 E.G 10 THEN IT REFFERS TO
                             height_shift_range=0.1,
                             zoom_range=0.2,  # 0.2 MEANS CAN GO FROM 0.8 TO 1.2
```

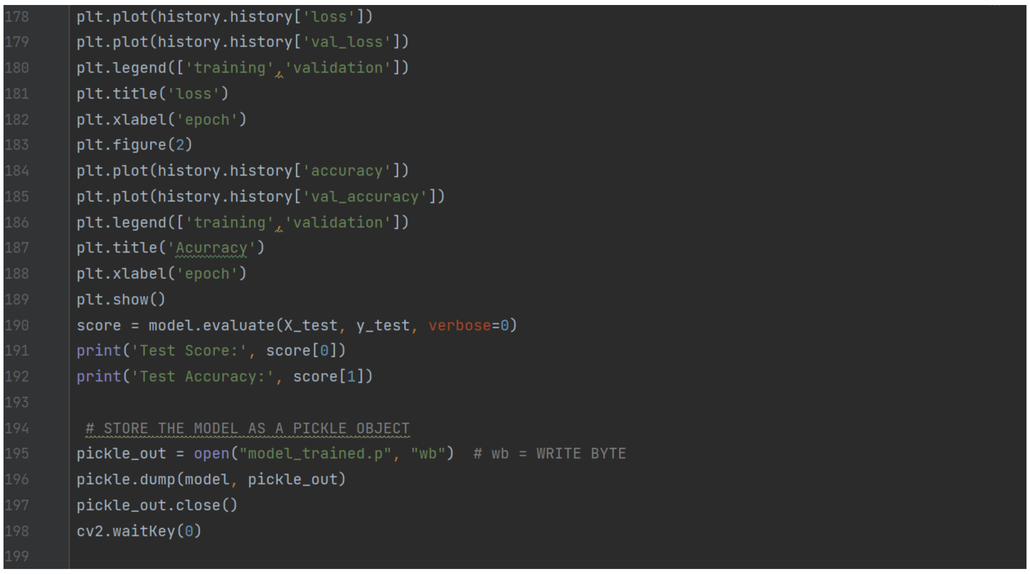


```
                             zoom_range=0.2,  # 0.2 MEANS CAN GO FROM 0.8 TO 1.2
                             shear_range=0.1,  # MAGNITUDE OF SHEAR ANGLE
                             rotation_range=10)  # DEGREES
dataGen.fit(X_train)
batches = dataGen.flow(X_train, y_train,batch_size=20)  # REQUESTING DATA GENRATOR TO GENERATE IMAGES  BATCH
X_batch, y_batch = next(batches)

# TO SHOW AGMENTED IMAGE SAMPLES
fig, axs = plt.subplots(1, 15, figsize=(20, 5))
fig.tight_layout()

for i in range(15):
    axs[i].imshow(X_batch[i].reshape(imageDimensions[0],imageDimensions[1]))
    axs[i].axis('off')
plt.show()

y_train = to_categorical(y_train, noOfClasses)
y_validation = to_categorical(y_validation, noOfClasses)
y_test = to_categorical(y_test, noOfClasses)


############################### CONVOLUTION NEURAL NETWORK MODEL
def myModel():
```

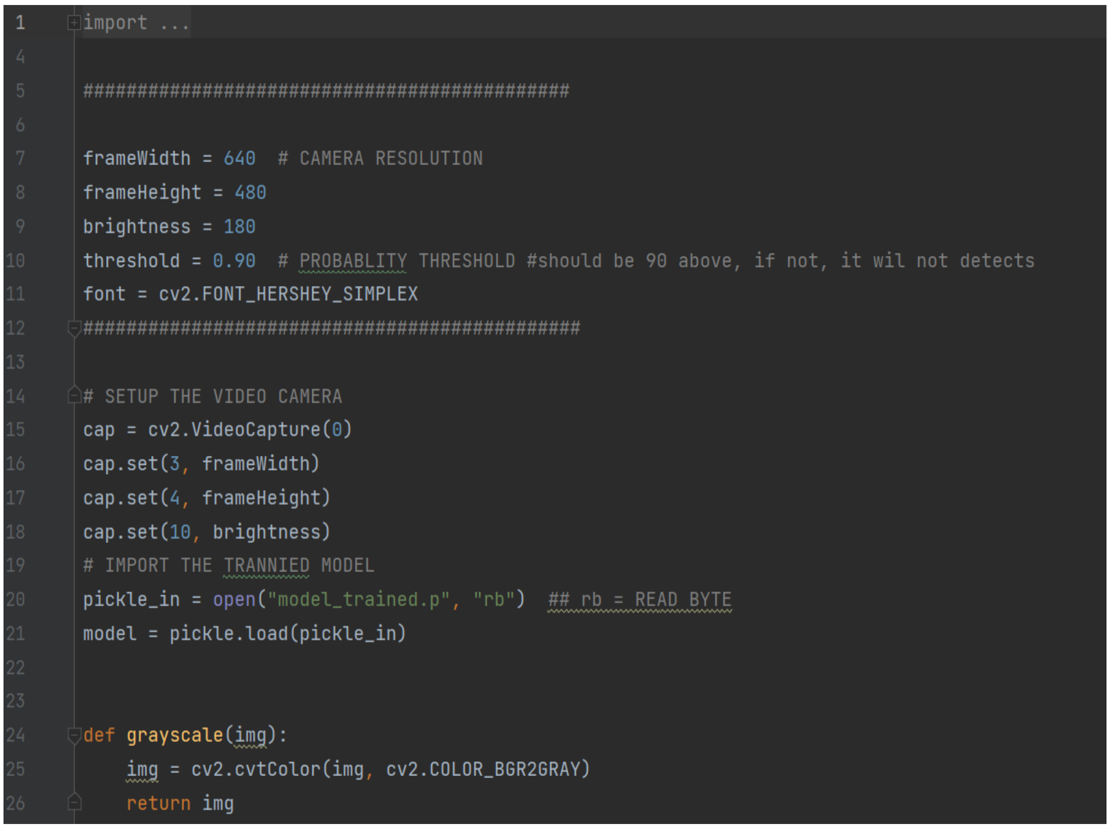


```
def myModel():
    no_Of_Filters = 60
    size_of_Filter = (5, 5)  # THIS IS THE KERNEL THAT MOVE AROUND THE IMAGE TO GET THE FEATURES.
    # THIS WOULD REMOVE 2 PIXELS FROM EACH BORDER WHEN USING 32 32 IMAGE
    size_of_Filter2 = (3, 3)
    size_of_pool = (2, 2)  # SCALE DOWN ALL FEATURE MAP TO GERNALIZE MORE, TO REDUCE OVERFITTING
    no_Of_Nodes = 500  # NO. OF NODES IN HIDDEN LAYERS
    model = Sequential()
    model.add((Conv2D(no_Of_Filters, size_of_Filter, input_shape=(imageDimensions[0],imageDimensions[1],1),
    model.add((Conv2D(no_Of_Filters, size_of_Filter, activation='relu')))
    model.add(MaxPooling2D(pool_size=size_of_pool))  # DOES NOT EFFECT THE DEPTH/NO OF FILTERS

    model.add((Conv2D(no_Of_Filters // 2, size_of_Filter2, activation='relu')))
    model.add((Conv2D(no_Of_Filters // 2, size_of_Filter2, activation='relu')))
    model.add(MaxPooling2D(pool_size=size_of_pool))
    model.add(Dropout(0.5))

    model.add(Flatten())
    model.add(Dense(no_Of_Nodes, activation='relu'))
    model.add(Dropout(0.5))  # INPUTS NODES TO DROP WITH EACH UPDATE 1 ALL 0 NONE
    model.add(Dense(noOfClasses, activation='softmax'))  # OUTPUT LAYER
    # COMPILE MODEL
    model.compile(Adam(lr=0.001), loss='categorical_crossentropy', metrics=['accuracy'])
```

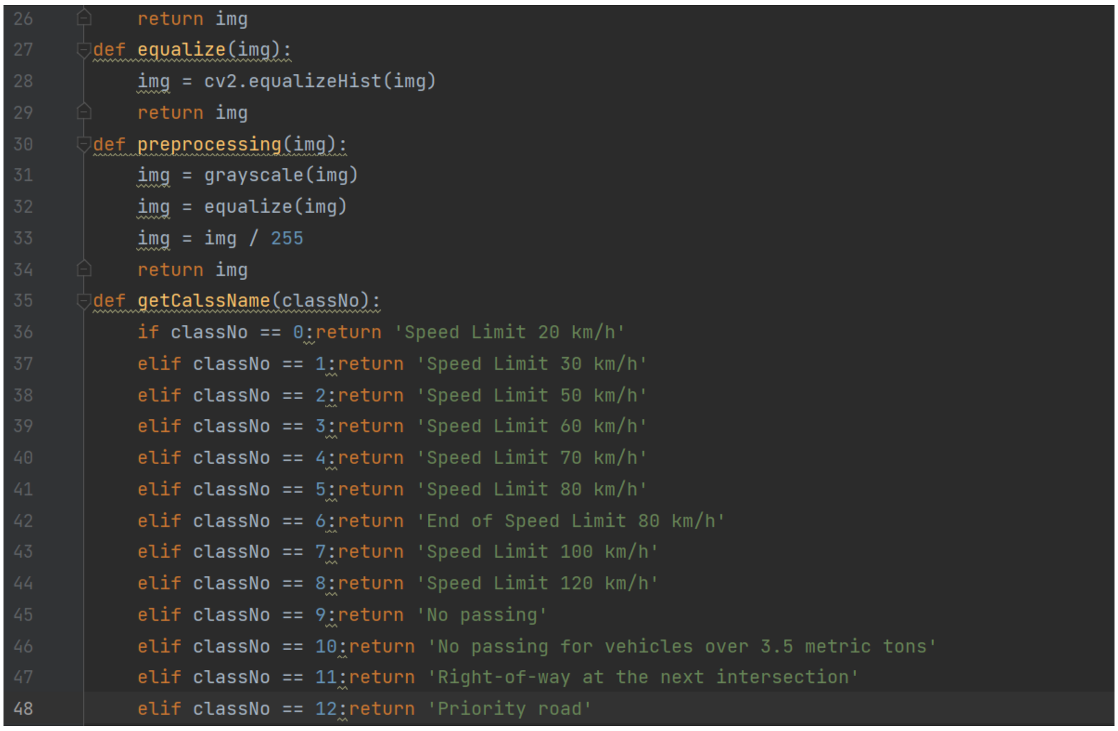

```
    # COMPILE MODEL
    model.compile(Adam(lr=0.001), loss='categorical_crossentropy', metrics=['accuracy'])
    return model


################################ TRAIN
model = myModel()
print(model.summary())
history = model.fit_generator(dataGen.flow(X_train, y_train, batch_size=batch_size_val),steps_per_epoch=step

################################ PLOT
plt.figure(1)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['training','validation'])
plt.title('loss')
plt.xlabel('epoch')
plt.figure(2)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['training','validation'])
plt.title('Acurracy')
plt.xlabel('epoch')
```


```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['training','validation'])
plt.title('loss')
plt.xlabel('epoch')
plt.figure(2)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['training','validation'])
plt.title('Acurracy')
plt.xlabel('epoch')
plt.show()
score = model.evaluate(X_test, y_test, verbose=0)
print('Test Score:', score[0])
print('Test Accuracy:', score[1])

 # STORE THE MODEL AS A PICKLE OBJECT
pickle_out = open("model_trained.p", "wb")  # wb = WRITE BYTE
pickle.dump(model, pickle_out)
pickle_out.close()
cv2.waitKey(0)
```

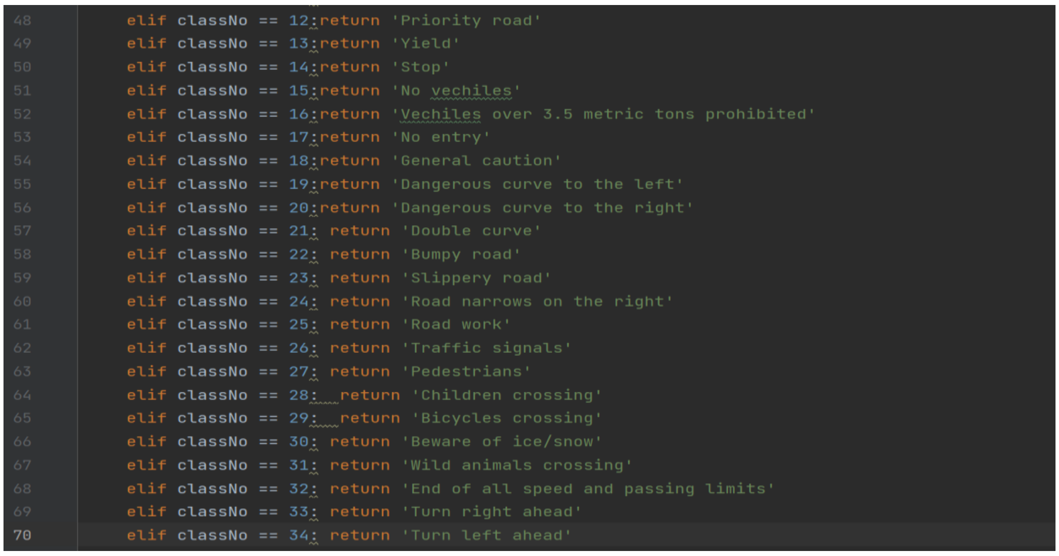
2. CODES FOR TRAINING DATA



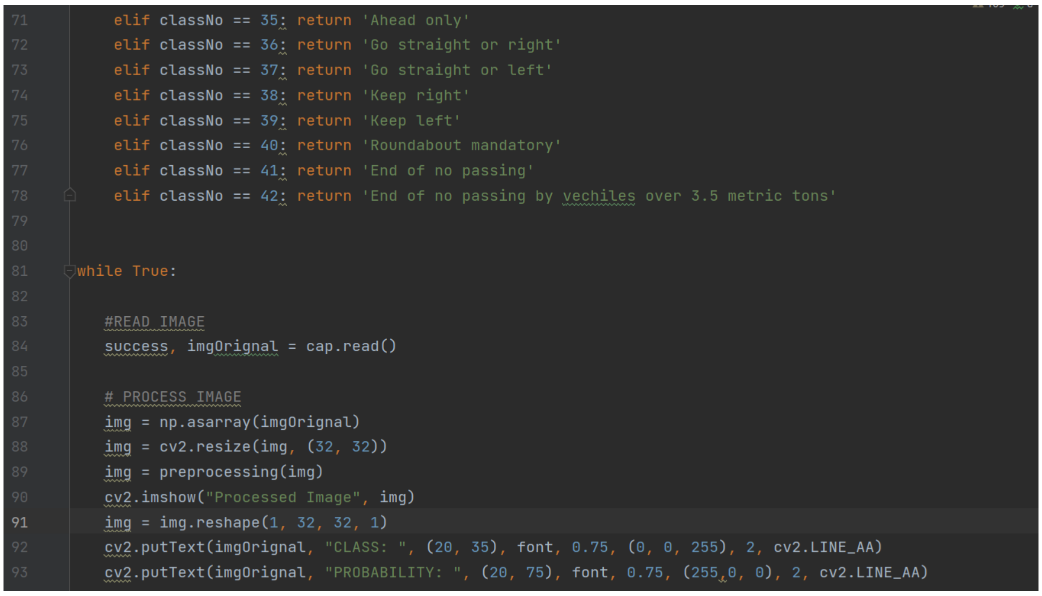
```python
import ...

#############################################

frameWidth = 640  # CAMERA RESOLUTION
frameHeight = 480
brightness = 180
threshold = 0.90  # PROBABLITY THRESHOLD #should be 90 above, if not, it wil not detects
font = cv2.FONT_HERSHEY_SIMPLEX
##############################################

# SETUP THE VIDEO CAMERA
cap = cv2.VideoCapture(0)
cap.set(3, frameWidth)
cap.set(4, frameHeight)
cap.set(10, brightness)
# IMPORT THE TRANNIED MODEL
pickle_in = open("model_trained.p", "rb")  ## rb = READ BYTE
model = pickle.load(pickle_in)


def grayscale(img):
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    return img
```



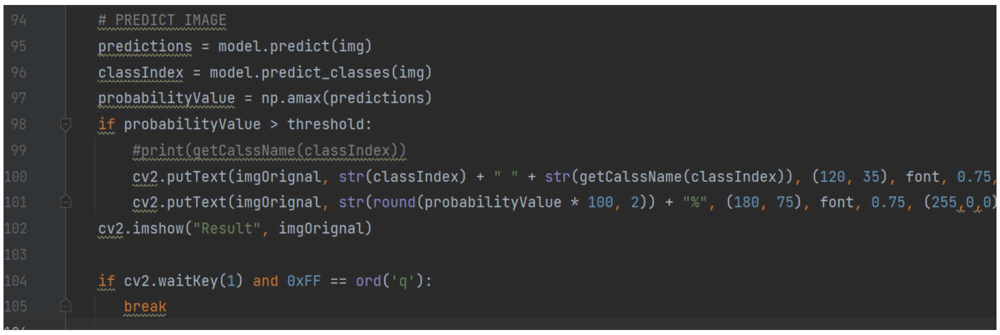
```python
    return img
def equalize(img):
    img = cv2.equalizeHist(img)
    return img
def preprocessing(img):
    img = grayscale(img)
    img = equalize(img)
    img = img / 255
    return img
def getCalssName(classNo):
    if classNo == 0:return 'Speed Limit 20 km/h'
    elif classNo == 1:return 'Speed Limit 30 km/h'
    elif classNo == 2:return 'Speed Limit 50 km/h'
    elif classNo == 3:return 'Speed Limit 60 km/h'
    elif classNo == 4:return 'Speed Limit 70 km/h'
    elif classNo == 5:return 'Speed Limit 80 km/h'
    elif classNo == 6:return 'End of Speed Limit 80 km/h'
    elif classNo == 7:return 'Speed Limit 100 km/h'
    elif classNo == 8:return 'Speed Limit 120 km/h'
    elif classNo == 9:return 'No passing'
    elif classNo == 10:return 'No passing for vehicles over 3.5 metric tons'
    elif classNo == 11:return 'Right-of-way at the next intersection'
    elif classNo == 12:return 'Priority road'
```



```python
    elif classNo == 12:return 'Priority road'
    elif classNo == 13:return 'Yield'
    elif classNo == 14:return 'Stop'
    elif classNo == 15:return 'No vechiles'
    elif classNo == 16:return 'Vechiles over 3.5 metric tons prohibited'
    elif classNo == 17:return 'No entry'
    elif classNo == 18:return 'General caution'
    elif classNo == 19:return 'Dangerous curve to the left'
    elif classNo == 20:return 'Dangerous curve to the right'
    elif classNo == 21: return 'Double curve'
    elif classNo == 22: return 'Bumpy road'
    elif classNo == 23: return 'Slippery road'
    elif classNo == 24: return 'Road narrows on the right'
    elif classNo == 25: return 'Road work'
    elif classNo == 26: return 'Traffic signals'
    elif classNo == 27: return 'Pedestrians'
    elif classNo == 28:  return 'Children crossing'
    elif classNo == 29:  return 'Bicycles crossing'
    elif classNo == 30: return 'Beware of ice/snow'
    elif classNo == 31: return 'Wild animals crossing'
    elif classNo == 32: return 'End of all speed and passing limits'
    elif classNo == 33: return 'Turn right ahead'
    elif classNo == 34: return 'Turn left ahead'
```



```python
    elif classNo == 35: return 'Ahead only'
    elif classNo == 36: return 'Go straight or right'
    elif classNo == 37: return 'Go straight or left'
    elif classNo == 38: return 'Keep right'
    elif classNo == 39: return 'Keep left'
    elif classNo == 40: return 'Roundabout mandatory'
    elif classNo == 41: return 'End of no passing'
    elif classNo == 42: return 'End of no passing by vechiles over 3.5 metric tons'


while True:

    #READ IMAGE
    success, imgOrignal = cap.read()

    # PROCESS IMAGE
    img = np.asarray(imgOrignal)
    img = cv2.resize(img, (32, 32))
    img = preprocessing(img)
    cv2.imshow("Processed Image", img)
    img = img.reshape(1, 32, 32, 1)
    cv2.putText(imgOrignal, "CLASS: ", (20, 35), font, 0.75, (0, 0, 255), 2, cv2.LINE_AA)
    cv2.putText(imgOrignal, "PROBABILITY: ", (20, 75), font, 0.75, (255,0, 0), 2, cv2.LINE_AA)
```



```python
    # PREDICT IMAGE
    predictions = model.predict(img)
    classIndex = model.predict_classes(img)
    probabilityValue = np.amax(predictions)
    if probabilityValue > threshold:
        #print(getCalssName(classIndex))
        cv2.putText(imgOrignal, str(classIndex) + " " + str(getCalssName(classIndex)), (120, 35), font, 0.75,
        cv2.putText(imgOrignal, str(round(probabilityValue * 100, 2)) + "%", (180, 75), font, 0.75, (255,0,0)
    cv2.imshow("Result", imgOrignal)

    if cv2.waitKey(1) and 0xFF == ord('q'):
        break
```

APPENDIX B
SAMPLE APPENDIX 2