# Traffic Sign Classification

By

Foo Wai Jun

Lem Wai Ping

YEO PEI HAN

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

July 2024

# ABSTRACT

Nowadays, the rapid growth of computer vision technology, particularly in the field of machine learning and deep learning, has led to the advancement of traffic sign recognition systems in autonomous vehicles and Advanced Driver Assistance System (ADAS). Traffic sign recognition system can help to identify the traffic rules such as speed limit, road conditions, route direction and promote safe driving. Traffic sign recognition system has two main stages: traffic sign detection and traffic sign classification. Traffic sign detection is to detect and extract the traffic sign from its background while traffic sign classification is to interpret the inner content of traffic sign. Nonetheless, this paper will focus on the second stage, which is traffic sign classification. and explores the existing works on traffic sign classification and provides a general idea to improve and enhance detection and classification methods based on previous works. This project provides a comprehensive overview of the proposed ideas, including data acquisition, data preprocessing (data split, image resize, noise reduction, CLAHE and image normalization), feature classification, model training as well as performance evaluation. Additionally, this paper highlights the achievements and limitations of previous studies. Last but not least, several case studies for traffic sign variations (low brightness, perspective distortion) are conducted to ensure the pre-processing steps are useful.

# TABLE OF CONTENTS

[Traffic Sign Classification i](#_Toc177135275)

[ABSTRACT ii](#_Toc177135276)

[TABLE OF CONTENTS iii](#_Toc177135277)

[LIST OF FIGURES vi](#_Toc177135278)

[LIST OF TABLES viii](#_Toc177135279)

[LIST OF SYMBOLS viii](#_Toc177135280)

[LIST OF ABBREVIATIONS x](#_Toc177135281)

[CHAPTER 1 – Introduction 1](#_Toc177135282)

[1.1 Problem Statement and Motivation 1](#_Toc177135283)

[1.2 Research Objectives 1](#_Toc177135284)

[1.3 Project Scope and Direction 2](#_Toc177135285)

[1.4 Contributions 2](#_Toc177135286)

[1.5 Background information 2](#_Toc177135287)

[CHAPTER 2 4](#_Toc177135288)

[Literature Reviews 4](#_Toc177135289)

[2.1 Literature Review 1 4](#_Toc177135290)

[2.1.1 Towards Real-Time Traffic Sign Classification and Detection 4](#_Toc177135291)

[2.1.2 Strengths and Weakness 6](#_Toc177135292)

[2.2 Literature Review 2 8](#_Toc177135293)

[2.2.1 Traffic Sign Detection using CNN (Convolutional Neural Network) 8](#_Toc177135294)

[2.3 Literature Review 3 13](#_Toc177135296)

[2.3.1 Recent Advances in Traffic Sign Recognition: Approaches and Datasets 13](#_Toc177135297)

[2.3.2 Strengths and Weakness (By: Foo Wai Jun) 15](#_Toc177135298)

[2.4 Literature Review 4 16](#_Toc177135299)

[2.4.1 Automatic Traffic Sign Detection and Recognition: A Review 16](#_Toc177135300)

[2.5 Literature Review 5 19](#_Toc177135302)

[2.5.1 Traffic Sign Classification using K-d tress and Random Forests 19](#_Toc177135303)

[2.5.2 Strength and Weaknesses 20](#_Toc177135304)

[2.6 Comparison of 3 techniques 21](#_Toc177135305)

[Chapter 3 22](#_Toc177135306)

[System Methodology/ Approach and Design 22](#_Toc177135307)

[3.1 System Design Diagram/Equation 22](#_Toc177135308)

[3.1.1 System Architecture Diagram 22](#_Toc177135309)

[3.1.2 Use Case Diagram and Description 23](#_Toc177135310)

[3.1.4 System Performance Definition 23](#_Toc177135311)

[3.2 System Block Diagram 24](#_Toc177135312)

[3.2.1 Data Ingestion 24](#_Toc177135313)

[3.2.2 Images Crop 24](#_Toc177135314)

[3.2.3 Data Split 24](#_Toc177135315)

[3.2.4 Images Resize 25](#_Toc177135316)

[3.2.5 Noise Reduction 25](#_Toc177135317)

[3.2.6 Grayscale Conversion 25](#_Toc177135318)

[3.2.7 CLAHE 25](#_Toc177135319)

[3.2.8 Normalization 25](#_Toc177135320)

[3.2.9 One-hot Encoding 25](#_Toc177135321)

[3.2.10 CNN Model Architecture 25](#_Toc177135322)

[3.2.11 Training 27](#_Toc177135323)

[3.2.12 Model Evaluation 27](#_Toc177135324)

[3.2.13 Hyperparameter Tuning 28](#_Toc177135325)

[3.2.14 Performance evaluation 29](#_Toc177135326)

[3.2.15 Model/Output 30](#_Toc177135327)

[CHAPTER 5 SYSTEM IMPLEMENTATION 31](#_Toc177135328)

[5.1 Hardware Setup 31](#_Toc177135329)

[5.2 Software Setup 31](#_Toc177135330)

[5.3 Setting and Configuration 31](#_Toc177135331)

[5.4 System Operation (with Screenshot) 32](#_Toc177135332)

[5.4.1 Data Exploration 32](#_Toc177135333)

[5.4.2 Data Preprocessing 33](#_Toc177135334)

[5.4.3 Training and evaluation 34](#_Toc177135335)

[5.4.4 Test Evaluation 36](#_Toc177135336)

[5.4.5 Hyperparameter Tuning 38](#_Toc177135337)

[5.4.6 Model Output (DEMO) 39](#_Toc177135338)

[5.5 Implementation Issues and Challenges 40](#_Toc177135339)

[5.6 Concluding Remark 40](#_Toc177135340)

[CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION 41](#_Toc177135341)

[6.1 System Testing and Performance Metrics 41](#_Toc177135342)

[6.2 Testing Setup and Result 41](#_Toc177135343)

[6.3 Project Challenges 41](#_Toc177135344)

[6.4 Objectives Evaluation 41](#_Toc177135345)

[6.5 Comparison with Previous Work 42](#_Toc177135346)

[6.6 Concluding Remark 43](#_Toc177135347)

[CHAPTER 7 CONCLUSION AND RECOMMENDATION 43](#_Toc177135348)

[7.1 Conclusion 43](#_Toc177135349)

[7.2 Recommendation 43](#_Toc177135350)

[REFERENCES 45](#_Toc177135351)

# LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **Figure Number** | **Title** | **Page** |
|  |  |  |
| Figure 2.1.1.1 | The pipeline of the proposed traffic sign detection and classification method [1] | 4 |
| Figure 2.1.1.2 | Structure of CNN [1] | 5 |
| Figure 2.1.2.1 | The Comparison of HOG Features with State-of-art Methods [1] | 6 |
| Figure 2.1.2.2 | The Comparison of CNN and State-of art Methods on GTSDB [1] | 6 |
| Figure 2.1.2.3 | The Comparison of CNN and State-of art Methods on CTSD [1] | 6 |
| Figure 2.2.1 | CNN Architecture [1] | 8 |
| Figure 2.2.2 | Ideal sign of traffic signs [2] | 9 |
| Figure 2.2.3 | Neural Network Dimensions Selection[2] | 9 |
| Figure 2.2.4 | (A) Result of classification for triangular sign [2] | 9 |
| Figure 2.2.4 | (B) Result of classification for circular sign [2] | 10 |
| Figure 2.3.1 | Overview of hyperparameter configuration table of different models [12] | 14 |
| Figure 2.3.2 | Overview of evaluation score of four proposed models[12] | 15 |
| Figure 2.4.1 | Result of Automatic White Balancing [6] | 16 |
| Figure 2.4.2 | Detection Rate of RBG, HSV and HSI Color Spaces [6] | 16 |
| Figure 2.5.1.1 | The Structure of HOG [10] | 19 |
| Figure 2.5.2.1 | Confusion Matrix of Speed Limits Using HOG and K-d Tree[10] | 20 |
| Figure 2.5.2.2 | Classification Results of Distance Transform (DT) features Using K-d Tree [10] | 20 |
| Figure 2.6 | Comparison of 3 techniques | 21 |
| Figure 3.1.1 | Flow of the Proposed Method | 22 |
| Figure 3.2.1 | Block Diagram of the Proposed Method | 24 |
| Figure 3.2.10 | CNN Model Architecture | 26 |
| Figure 3.2.11 | Multiclass Cross Entropy Loss | 27 |
| Figure 3.2.13.2 | A convergence plot of Bayesian optimization | 28 |
| Figure 3.2.13.3 | A table view of loss and accuracy by using Bayesian optimization | 29 |
| Figure 3.2.14.1 | Overview of evaluation between different models based on different dataset | 30 |
| Figure 3.2.14.2 | Comparison table of processing time between different models | 30 |
| Figure 5.4.1 | Train Class Count Distribution | 32 |
| Figure 5.4.2 | Image preprocessing | 33 |
| Figure 5.4.3 | Model performance insight | 35 |
| Figure 5.4.4 | Model Evaluation | 37 |
| Figure 5.4.5 | Hyperparameter Tuning | 38 |
| Figure 5.4.6 | model output (demo) | 39 |
| Figure 5.5.1 | Train Class Count Distribution | 40 |

# LIST OF TABLES

|  |  |  |
| --- | --- | --- |
| **Table Number** | **Title** | **Page** |
|  |  |  |
| Table 5.1.1 | Hardware Setup | 31 |
| Table 5.2.1 | Software Setup | 31 |

# LIST OF SYMBOLS

# LIST OF ABBREVIATIONS

|  |  |
| --- | --- |
| *ADAS* | Advanced Driver Assistance System |
| *CNN* | Convolutional Neural Networks |
| (CHE) | Contrast-limited Adaptive Histogram Equalization |
| TSC | Traffic Sign Classification |
| *CV* | Computer Vision |
| *HSV* | Hue, Saturation, Value |
| MSER | Maximally Stable Extremal Region |
| HOG | Histogram of Oriented Gradient |
| SVM | Support Vector Machine |
| RBF | Radial Basis Function |
| CLAHE | Contrast Limited Adaptive Histogram Equalization |
| GTSDB | German Traffic Sign Detection Bechmark |
| CTSD | Chinese Traffic Sign Dataset |
| GPU | Graphics Processing Unit |
| ITS | Intelligent Transportation Systems |
| RGB | Red, Green, Blue |
| HSI | Hue, Saturation, Intensity |
| BRISK | Binary Robust Invariant Scalable Keypoints |
| TSR | Traffic Sign Recognition |
| DT | Distance Transforms |
| ROC | Receiver Operating Characteristic Curve |
| AUC | Area Under ROC Curve |
| ReLu | Rectifier linear unit |
| ADAM | Adaptive Moment Estimation |
|  |  |
|  |  |
|  |  |
|  |  |

# CHAPTER 1 – Introduction

## 1.1 Problem Statement and Motivation

Traffic sign recognition is a critical component in the advancement of autonomous vehicles and Advanced Driver Assistance System (ADAS). Various studies and research have been conducted to improve and enhance traffic sign recognition systems. However, the uncertainty and variability of traffic signs across the road restraints the capability of these technologies to interpret the traffic sign accurately. Having diverse weather conditions, deformed and physically damaged traffic signs as well as the traffic signs with low brightness, cluttered background and similar background colours, causing the accuracy and performance of TSR to be downgraded.

For some methods, they do achieve high accuracy, but often compromise with the computational efficiency. They require extensive complex algorithms and deep learning models, such as Convolutional Neural Networks (CNNs), to predict the inner content of traffic signs. These extensive calculations require large processing power and memory to achieve high accuracy under various challenging conditions, such as diverse weather conditions, bad illumination and so on. Therefore, the processing is slower, making it impractical for real life applications before a balance point between accuracy and speed is struck.

Therefore, in this paper, we propose a traffic sign classification model using deep learning algorithm, which is CNN. The motivation of this project is to explore and improve the accuracy and performance of traffic sign classification from the previous studies.

## 1.2 Research Objectives

The main goal of the project is to develop an effective model in classifying the traffic sign using deep learning algorithms. The model should detect and recognize the traffic sign in different environments with different lighting, weather and angles. Then, classify it under sub classes, for instance, whether it is under regulatory sign, warning sign or informational signs. After that, the model should display the result of accuracy in percentage on the classification process. In this case, the model’s performance on the classification job is indicated by the higher the accuracy displayed.

## 1.3 Project Scope and Direction

The scope of this project is to develop a CNN based system in classifying different traffic signs on the road. The model is expected to display the sub classes of the traffic sign, and the accuracy of the outcome. At first, data cleaning and preprocessing will be performed, for instance, resizing the image into a standardize form since CNN implementation required training images to be in equal size, then adjusting the contrast of the image using Contrast-limited Adaptive Histogram Equalization (CLAHE). Throughout the process of training this model, the adjustment will be made to minimize the computational time and maximize the accuracy rate. This approach will ensure development of efficient and accurate traffic sign classification systems.

## 1.4 Contributions

ADAS is a series of technologies designed to provide drivers with safe driving assistance. By integrating sensors, control units, actuators, and other hardware and software components in vehicles, ADAS can analyze real-time driving conditions and make corresponding adjustments to enhance driver safety.

One of the most crucial elements of ADAS is traffic sign classification (TSC). Implementing a high-performance TSC model can significantly improve the reliability of ADAS and its prospects for the future. As technology rapidly advances, autonomous driving is set to become a mainstream trend in the automotive industry. Autonomous driving offers convenience, time-saving benefits, and most importantly, a reduction in accidents caused by human error. In this project, we will employ CNN models to achieve a highly efficient and reliable TSC task.

## 1.5 Background information

The traffic sign recognition is an important aspect of the autonomous vehicles and for the Advanced Driver Assistance System (ADAS) which heavily relies on the accurate image classification to enable road users to have a safe and efficient driving experience. However, challenges like diverse weather conditions, varied lighting conditions, damaged signboards and some visual impediments such as low brightness and cluttered backgrounds which often cause in reduce of accuracy and performance of these systems.

Explicitly, the current methods for the traffic sign classification needed to be accurate, often face the sacrificial of the computational efficiency and rely too much on the complex algorithms and deep learning models, particularly the Convolutional Neural Networks (CNN). They require substantial processing power and memory which results in slower processing times causing those methods to be rather impractical for real-world use.

This project enables us to propose a CNN-based model to efficiently enhance the traffic sign image classification. With the goals of detecting and recognizing the traffic signs in various conditions and to also classify them into respective categories like regulatory, warning and informational along with the display classification accuracy. All of these involve the data cleaning and preprocessing such as the resizing of images, adjusting contrast and the optimization of the model to balance the computational time and accuracy.

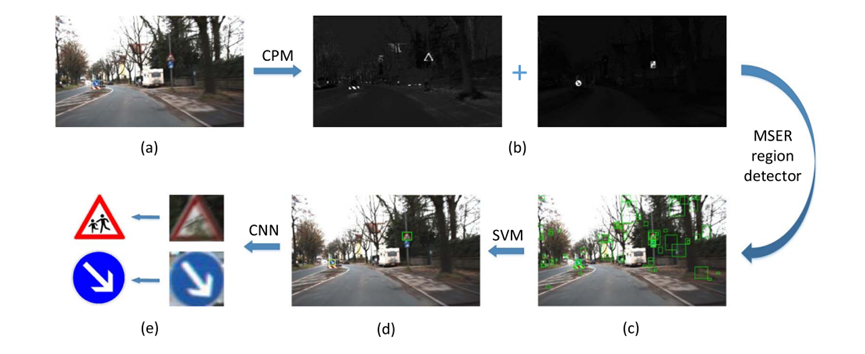
The high-performance traffic sign classification (TSC) model can also significantly enhance the ADAS through the improvising of its reliability and supporting the advancements in the autonomous driving which can greatly reducing the accidents and more time saving. Additionally, this project will provide several valuable pieces for education and research via leveraging CNN with the aim to develop an efficient and reliable image classification system for the traffic signs.

# CHAPTER 2

# Literature Reviews

## 2.1 Literature Review 1

### 2.1.1 Towards Real-Time Traffic Sign Classification and Detection

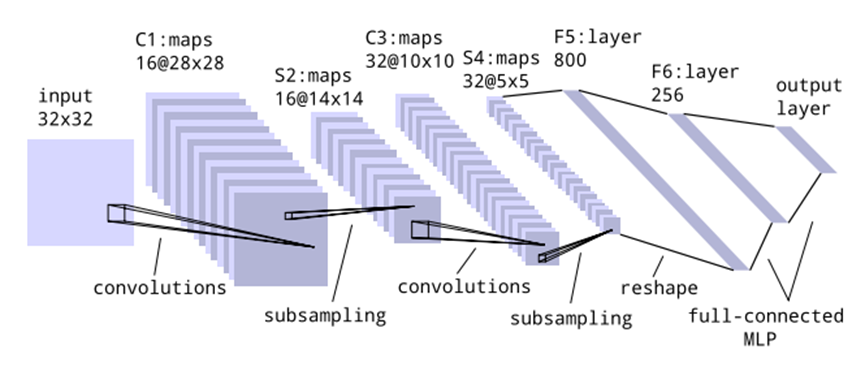
*Figure 2.1.1.1 The pipeline of the proposed traffic sign detection and classification method*

1. *Original image (b) Color probability mode (red + blue) (c) MSER Detection (d) Detection (e) Classification [1]*

The Figure 2.1.1.1 shows the pipeline of the proposed traffic sign detection and classification in this paper. For traffic sign detection, the authors have implemented color threshold segmentation based on HSV color based [1]. To fully utilize the color information of traffic signs, the authors propose a color probability model in enhancing the traffic sign color and suppressing the background color [1]. The output of this model is a gray image with high intensity of traffic sign color and low intensity of other pixels [1]. Then this model is combined with a pre-calculated look up table to speed up the processing and reduce the search space for detection [1]. This is followed by the employment of Maximally Stable Extremal Region (MSERs) to find the maximally stable extremal regions from the probability model, in order to increase the contrast between traffic sign and background, thus achieving high recall rate [1].

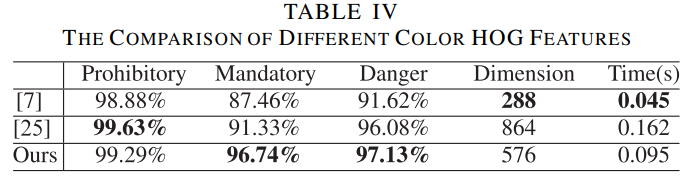
During the traffic sign classification stage, a novel color Histogram of Oriented Gradient (HOG) feature is proposed by conducting computation on both probability map and histogram equalized gray image [1]. Color probability is to encode both color and shape of traffic sign effectively while histogram equalized gray image is to identify the inner content [1]. Then, this feature is trained with a multi-class Support Vector Machine (SVM) classifier via Radial Basis Function (RBF) to reduce redundant samples, which significantly reduces the number of support vectors and increases the testing time [11]. Data augmentation such as rotation, translation and brightness adjustment are conducted to increase the diversity [1].

The second method of traffic sign classification used by authors to compare the processing time is CNNs. The authors only use grayscale images to reduce processing time [1]. All the images are resized to standard 32 x 32 pixels and applied with Contrast Limited Adaptive Histogram Equalization (CLAHE) to increase contrast and decrease the influence of lighting [1]. Three simple CNNs with 2 convolutional layers, 2 subsampling layers and a fully connected MLP are trained to speed up the computation [1]. 5x5 filter kernels and L2-pooling are used for the convolutional layers and subsampling layers respectively [1].

 *Figure 2.1.1.2 Structure of CNN [1]*

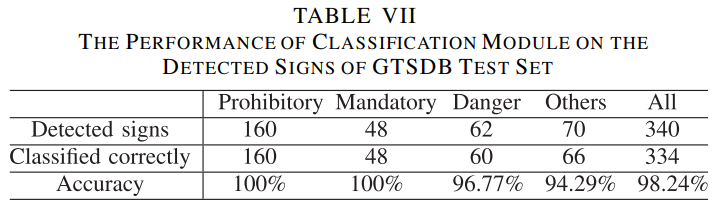
### 2.1.2 Strengths and Weakness

There are two methods used for classification in this paper which is the use of the HOG feature with SVM classifier on color probability model and CNN. The performance of the proposed HOG feature with SVM classifier outperforms the other state-of-art methods in Mandatory and Danger signs for both GTSDB and CTSD (refer to Figure 2.1.2.1). It is proven to be 20 times faster with slightly lower accuracy than other methods [1]. The most important achievement is it only requires 0.162 second to detect an image, but others need a few seconds [1]. The proposed HOG + SVM is proven to have high potential for real life applications.

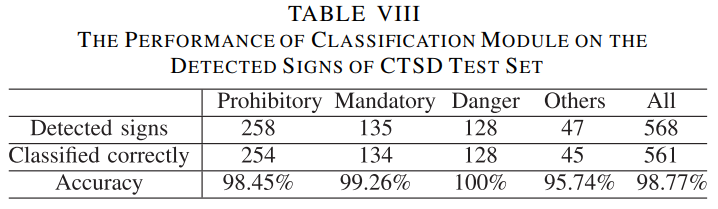


*Figure 2.1.2.1 The Comparison of HOG Features with State-of-art Methods[1]*

The efficiency of CNN is proven when tested with GTSDB and CTSD as it achieves 98.24% and 98.77% of overall accuracies (refer to Figure 2.1.2.2 and Figure 2.1.2.3) [1]. The classification time is ~3 ms per sign and the overall performance is ~6 fps (165 ms per image) on 1360 x 800 images. As compared to the traditional method of using HOG + SVM, this CNN has reached the standard for ADAS and autonomous vehicle equipment [1].



*Figure 2.1.2.2 The Comparison of CNN and State-of art Methods on GTSDB*



*Figure 2.1.2.2 The Comparison of CNN and State-of art Methods on CTSD [1]*

Nonetheless, the uncertainty and variability of weather conditions and traffic sign still pose a significant challenge to both methods. Both methods may misinterpret the traffic sign with discoloration, blur image, partial occlusion and cluttered background. The achievement of HOG + SVM in this paper is due to the use of color probability model. This model requires additional steps like color conversion, LUT and MSER detection to ensure high accuracy, which is not computation efficient. This is proven as the processing speed is slightly slower than other methods. However, the authors believe that by accelerating the GPU, the processing speed can be improved. Therefore, further improvements are required to enhance the recognition system in handle different real-life scenarios before it is putting in use.

## 2.2 Literature Review 2

### 2.2.1 Traffic Sign Detection using CNN (Convolutional Neural Network)

This literature review is about the application of Convolutional Neural Network (CNN) for classifying traffic signs. There are a total of 5 layers included in CNN composed of 3 convolutional layers and 2 hidden layers, which are input layer, convolutional layer, pooling layer, fully connected layer and output layer. By applying the 5 layers of CNN, it is more efficient in extracting more complex and abstract info from the input picture [2].

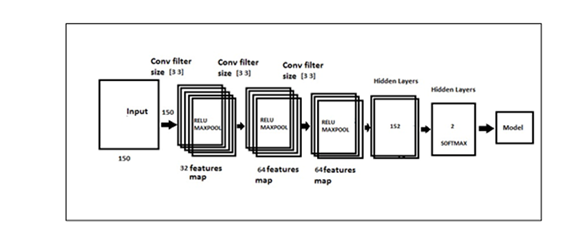


Figure 2.2.1 CNN Architecture [1]

During data collection and preparation, the researchers have applied OpenCV to perform scaling, normalization and augmentation. Then, they have applied Convolutional layers in extracting patterns from the input images and full connected ones in making fine-grained predictions [2].

Then, the CNN model is trained by separating the datasets (GTSRB) into training and validation. To achieve a best result, hyperparameters such as learning rate, batch size and optimizer are fine-tuned periodically [2]. Additionally, during the model training, researchers have applied backpropagation technique to minimize the model’s lose and monitored the validation set’s accuracy to prevent the model from overfitting. In model evaluation stage, the researchers have applied the performance metrices to check the capability or efficiency of the model in classifying the traffic signs. [2]

In another article [3], the researchers have extracted the color and the corners as features to detect the traffic signs, then the detected images are used as input images in classification stages. In the classification stage, two algorithms has been applied to detect the image into circular signs and triangular signs. In the normalization stage, nearest neighbour method has been applied instead of bilinear interpolation method due to the higher cost and insignificant improvements. In the training stage, nine ideal signs are selected from circular and triangular signs. (refer to Figure 2.2.2). The researchers have chosen third neural networks (30X30/15/5/10) to train the network for the circular signs, as it shows the best result among the rest of it (refer to Figure 2.2.3). The new training set patterns have been created, and the outcome of the result as shown in figure 2.2.4(A) and 2.2.4(B) [3].

Based on the article [2], it shows that CNN model has a great performance in classifying the traffic signs in the different environment with an accuracy of 97.3%. Hence, it can be concluded that this approach can be applied to real world scenarios, specifically on the autonomous car system.

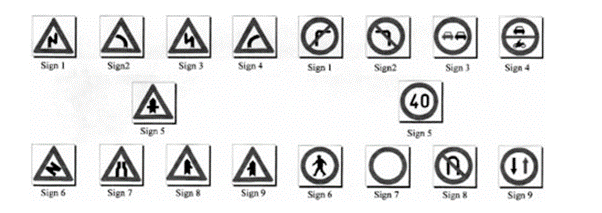


Figure 2.2.2 Ideal sign of traffic signs [2]

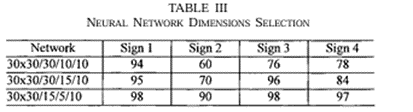


Figure 2.2.3 Neural Network Dimensions Selection [2]

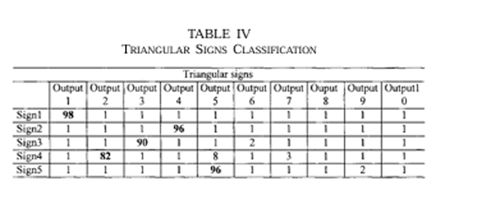


Figure 2.2.4 (A) Result of classification for triangular sign

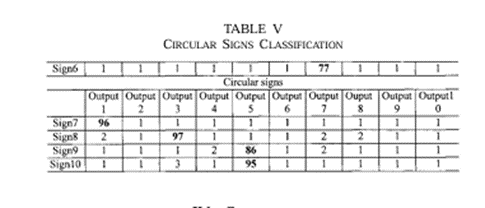


Figure 2.2.4 (B) Result of classification for circular sign

2.2.2 Strengths and Weakness **(By: Phoebe Wong Hui Lei)**

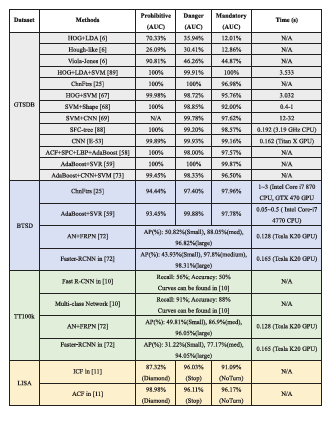


Figure 2.2.5 Overview of performance of different machine learning based methods on different datasets [12]

Based on the figure above, compares four distinct datasets: GTSDB, BTSD, TT100K, and LISA. In this section, a primary focus on comparision methods used in the GTSDB dataset. In the GTSDB dataset, CNN methods achieve high accuracy performance in detecting prohibitive, danger, and mandatory signs, with AUC scores of 99.89%, 99.93%, and 99.16% [12], respectively. Notably, methods such as HOG + LDA + SVM, ChnFtrs, and AdaBoost + SVR demonstrate similarly accuracy levels.

Regarding computational efficiency, the time required varies among methods. For instance, HOG + LDA + SVM stands out with its high accuracy but consumes 3.533 seconds per process. In contrast, CNN completes the task in 0.162 seconds [12]. Despite CNN's speed advantage, the decision on method selection weighs heavily on balancing accuracy with the computational time required.

In conclusion, while some methods like HOG + LDA + SVM excel in accuracy, the rapid processing time of CNN highlights the crucial trade-off between performance and efficiency in choosing the appropriate method for traffic sign detection and classification.

## 2.3 Literature Review 3

### 2.3.1 Recent Advances in Traffic Sign Recognition: Approaches and Datasets

In this research paper, it presents a comparative study on traffic sign classification using deep learning techniques. It implements and evaluates the performance of four models, which are CNN, ResNet50, VGG19, and EfficientNetB7 based on the German Traffic Sign Recognition Benchmark (GTSRB) dataset. The study highlights the importance of accurate traffic sign classification in Intelligent Transportation Systems (ITS) for enhancing road safety, optimizing traffic flow, and supporting autonomous vehicles [4].

In [4], the methodology section introduces four deep learning models, with detail descriptions are shown as below:

CNN:

The CNN model is designed to classify 43 different traffic signs. It uses convolutional layers with filters ranging from 16 to 128 and a (3, 3) kernel size with ReLU activation to extract features. Max pooling and batch normalization enhance feature extraction and normalization. A flattening layer converts 2D feature maps into a 1D vector, which is then processed by a dense layer with 512 units, ReLU activation, and batch normalization. A dropout layer (rate = 0.5) is applied to prevent overfitting. Finally, the final dense layer uses softmax activation to output 43 class probabilities.

ResNet50:

The ResNet50 model is pre-trained on ImageNet, which is used for traffic sign classification. It starts with the ResNet50 base to extract features. This model includes batch normalization, a flattening layer to convert output into a 1D array, and a dense layer with 512 units and ReLU activation for feature processing. Next, another batch normalization layer follows, and a dropout layer (rate = 0.05) helps prevent overfitting. The final dense layer uses softmax activation to predict the classes. During training, data augmentation techniques like rotation, zooming, shifting of w, h, and shear transformations are applied; however, flips are avoided to keep the consistency of traffic sign orientation.

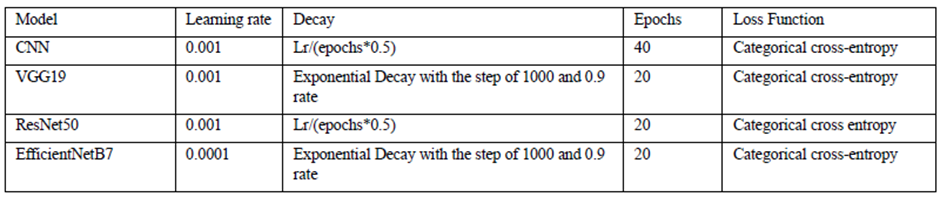
VGG19:

The VGG19 model is also pre-trained model. It starts with convolutional layers that use filters (16 to 128) and a (3, 3) kernel size with ReLU activation. After that, max pooling with a (2, 2) window size is applied, and followed by batch normalization. Same as ResNet50, a flattening layer converts 2D into a 1D array, and a dense layer with 512 units, ReLU activation, and batch normalization. A dropout layer (rate = 0.05) is applied. The final dense layer uses SoftMax activation to generate predictions for 43 classes for the classification task.

EfficientNetB7:

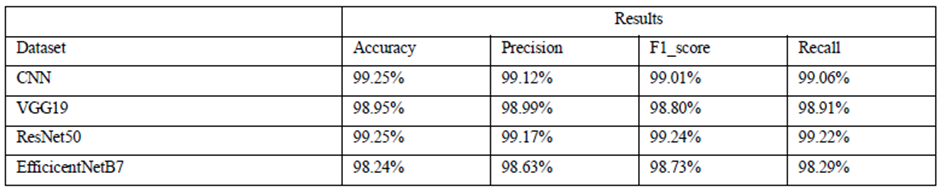
The EfficientNetB7 model was used to classify 43 classes of traffic signs. The learning rate starts at 0.0001 and adjusts using exponential decay to improve optimization. This model is also pre-trained on ImageNet and fine-tuned for this task. It included batch normalization, L2 regularization, dropout for preventing overfitting, and a final dense layer with softmax activation to make predictions. During training, early stopping with 5 epochs is applied to prevent overfitting.

During hyperparameter tuning, [4] implements a decay to avoid trapped in local optima and improve searching for global optima. Moreover, the Adam optimizer is employed to optimize the models. It can adjust the learning rate for each weight in the model dynamically. As an illustration, a hyperparameter configuration table is referenced from [4] is shown as below:

Figure 2.3.1 Overview of hyperparameter configuration table of different models [12]

### 2.3.2 Strengths and Weakness (By: Foo Wai Jun)

During performance evaluation, the four proposed models’ accuracy scores are shown as below [4]:

Figure 2.3.2 Overview of evaluation score of four proposed models [12]

The proposed CNN model achieved a high accuracy of 99.25%, which is better than the comparable studies. ResNet50 also performed well, and it is consistent with the studies with an accuracy of 99.25%. VGG19 had a competitive accuracy of 98.95%, and EfficientNetB0 reached a good accuracy of 98.24%.

Besides accuracy, the processing time of the model is also another crucial factor in real time implementation. Therefore, it is essential to compare the processing time of proposed models with other studies’ models as well [5]. In addition, it is suggested that the proposed model of this project can be evaluated not only based on the GTSRB dataset, also based on other dataset, which is BTSDB dataset, Chinese traffic sign database, etc. Therefore, it can validate the applicability of the proposed model on different dataset.

In conclusion, this study analysed and compared various of deep learning models based on different datasets. Moreover, it also categorized them by detection, classification, and combined task. The observation between other models and the author proposed models provided a valuable insight for the future research [4].

## 2.4 Literature Review 4

### 2.4.1 Automatic Traffic Sign Detection and Recognition: A Review

In this paper, the author states that applications of Advanced Driver Assistance Systems (ADAS) involve numerous functions, such as blind spot detection, speed limit recognition, emergency brake assist, and many more. Traffic indeed provides information, such as warnings about incoming dangerous road conditions ahead or helping guide the driver to a safer direction. This literature review will discuss existing methods for traffic sign detection and recognition [6].

First, there are existing techniques such as color-based detection, shape-based detection, both color and shape-based detection, and other approaches to detect traffic signs. Moreover, a comparative study was performed in RGB, HSV, and HSI color spaces in traffic sign detection; detection rates are referred to in Figure 2.4.1 For traffic sign recognition, feature matching techniques such as Binary Robust Invariant Scalable Keypoints (BRISK) and machine learning approaches like SVM, neural networks, artificial neural networks, and many more are used.

Furthermore, detection and recognition of traffic signs face challenges such as illumination variations, obstruction by trees, motion blur, and more. Morphological operations were used to remove background noise, and white balancing was used to decrease the effect of illumination (refer to Fig. 2.4.1).

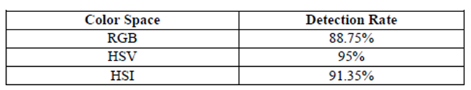


Figure 2.4.1 Detection Rate of RBG, HSV and HSI Color Spaces [6]

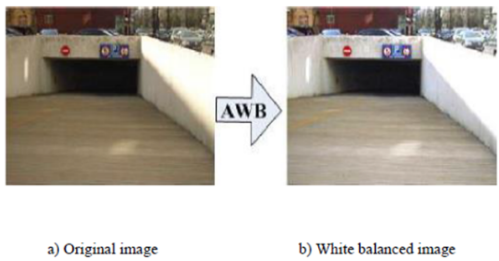


Fig.2.4.2 Result of Automatic White Balancing [6]

2.4.2 Strengths and Weakness **(By: Lem Wai Ping)**

For traffic sign detection [6], under color-based detection, although RGB Space has low computation cost, can process color information, and can perform morphological operations when dealing with noise, it still faces potential challenges such as color fading and weather conditions. While HSI and HSV color spaces can handle illumination conditions as they can separate achromatic and chromatic components with high accuracy when detecting traffic signs, they still have weaknesses such as high computational cost and limitations with noise after image segmentation is done. For YCbCr color space, it is efficient when dealing with traffic sign segmentation in dynamic thresholding; it helps to improve detection.

For shape-based detection [6], Hough Transform only has high accuracy for detecting circles and triangles but comes with a high computation cost. Radial Symmetry Image is effective for detecting circular shapes but struggles with distorted shapes.

Other than color and shape-based detection [6], combined detection of color and shape-based methods also shows effective edge approximation, even though it has some distortions when using the DP Algorithm. Moreover, combined detection also provides multiple methods for detection using color segmentation and feature analysis, but it also has some limitations with changes in image sizes and the shape of traffic signs.

For other approaches like MSERs, it can handle low weather factors and can retain shapes in good condition even after multiple thresholding levels, although it can be computationally intensive. While the color probability map works well with SVM for classifications, it has high computational cost and performance may drop when handling a more complex color variation model [6].

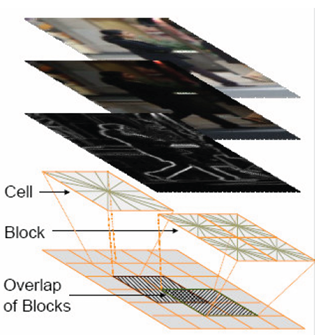
The Template-Based Method is effective when needing to find a specific shape but is expensive due to distance transformation and template matching. BRISK is faster than SURF and provides rotation and scale invariance, but it has low accuracy when dealing with complex scenes because it is a binary descriptor. SURF offers higher accuracy and better feature description compared to BRISK but is slower, which might impact real-time processing applications. For Template Matching with NCC, it is simple to implement but has less adaptability to the robustness of traffic sign types [6].

For machine learning techniques, CNN and ANN have high accuracy and high learning capabilities but still have high complexity in computation. The next technique is GA, which does not require training phases and can handle weather condition factors effectively, but it still needs careful image comparison. SVM with Zernike Moments and HOG Features, on the other hand, combines well with traditional features but the performance will be affected when increasing the computation time. Lastly, Hu Moments and LBP features offer a trade-off between accuracy and computational simplicity but have weaknesses like limitations when image size changes or grayscale variation [6].

## 2.5 Literature Review 5

### 2.5.1 Traffic Sign Classification using K-d tress and Random Forests

In this paper, the author defined the traffic sign recognition (TSR) as a critical component of the Driver Assistance Systems (DAS) which provides the drivers with essential safety and all the precautionary information. In the previous work, we developed a Histogram of Oriented Gradients (HOG) basely on the Support Vector Machine (SVM) detector mainly to identify the traffic sign candidates in variety and diverse environments. This paper focus on the evaluation of the classification phase by comparing all the performance of the K-d trees and Random Forests using the HOG descriptors along with the Distance Transforms on the images from the German Traffic Sign Benchmark datasets.[10]

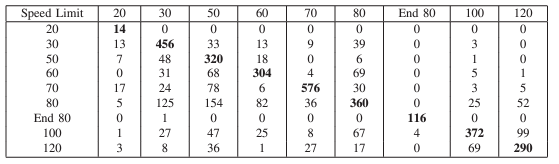


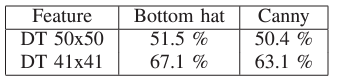
*Figure 2.5.1.1 The Structure of HOG [10]*

The traffic signs are illustrated and designed to match with its high visibility and comprehension while utilizing the high contrast, vibrant colours, understandable pictograms and characters. With this standardization, it able to facilitates a nearest-neighbour approach and hence, efficiently implement the usage of the tree structures which are advantageous due to their speed, ease of the construction and the robustness with the imbalanced datasets as compared to the SVMs and Neutral Networks that require extensive parameter tuning.

### 2.5.2 Strength and Weaknesses

The traffic sign recognition system requires to process an input of the traffic sign image through several stages. It will undergo the pre-processing including the resizing and normalization with then extracted using the Histogram of Oriented Gradients (HOG) and Distance Transforms (DT). The HOG method involves dividing the image into blocks and cells, computing the gradient orientations and all the magnitudes and forming histograms while the DT method which calculates the distance to the nearest nonzero pixel in the binary image. Thus, for the classification, K-d Trees perform a nearest neighbour search and the Random Forests use an ensemble of random trees to determine the traffic sign class.

 *Figure 2.5.2.1 Confusion Matrix of Speed Limits Using HOG and K-d Tree[10]*



*Figure 2.5.2.2 Classification Results of Distance Transform (DT) features Using*

*K-d Tree*

## 2.6 Comparison of 3 techniques

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Input Size** | **Parameters** | **Accuracy** | **Precision** | **F1 Score** | **Recall** | **Dataset** |
| CNN [4] | 32x32 | ~1.4M | 99.25% | 99.12% | 99.01% | 99.06% | GTSRB datasets |
| VGG19 [4] | 224x224 | ~143.6M | 98.95% | 98.99% | 98.80% | 98.91% | GTSRB datasets |
| ResNet50 [4] | 224x224 | ~25.6M | 99.25% | 99.17% | 99.24% | 99.22% | GTSRB datasets |
| EfficientNetB7 [4] | 600x600 | ~66M | 98.24% | 98.63% | 98.73% | 98.29% | GTSRB datasets |
| HOG 324 [1] | 18x18 | N/A | 73.70% | - | - | - | GTSDB and CTSD |
| HOG 576 [1] | 24x24 | N/A | 66.40% | - | - | - | GTSDB and CTSD |
| HOG 900 [1] | 30x30 | N/A | 78.50% | - | - | - | GTSDB and CTSD |
| Auto-Associative NN [6] | - | - | 94.7% (Shadow) / 100% (Daylight) | - | - | - | - |
| Gene Coding Method [6] | - | - | 94.7% | - | - | - | - |

Figure 2.6 Comparison of 3 techniques

Combined Dataset = combination of existing datasets like GTSRB (Germany), KLU (Belgium), MASTIF (Croatia), Stereo polis (France), RUG (Netherland), STS (Sweden), and additional images was captured around the UTBM laboratory at the site Belfort, France.

# Chapter 3

# System Methodology/ Approach and Design

## 3.1 System Design Diagram/Equation

3.1.1 System Architecture DiagramA diagram of a model

Description automatically generated

*Figure 3.1.1.1 Flow of the Proposed Method*

The proposed method of this project is categorized into different phases in the development, which are data acquisition, data pre-processing, model architecture construction, hyperparameter tuning and performance evaluation.

During data acquisition stage, thousands of traffic sign images are collected from Chinese Traffic Sign Database (CTSD). Then these images are cropped and pre-processed, such as data split, image resizing, noice reduction, grayscale conversion, CLAHE and image normalization and one-hot encoding.

After the data pre-processing stage, the images will go through the CNNs model. Our CNNs model will consist of four Convolutional layers followed by two MaxPooling layers, a Flatten layer and a fully connected Dense layer. The output will then be trained with appropriate epochs and batch size.

The model will be evaluated based on accuracy, loss and confusion matrix on the test set, and tuned by setting new batch size and epochs as well as performing Bayesian optimization. Last but not least, the performance is evaluated via precision, recall, F1-score, ROC curve and AUC. Eventually, a traffic sign classification model is delivered. When the users upload or show traffic sign image, the model will interpret and display the meaning of the traffic sign.

### **3.1.2** Use Case Diagram and Description

* User shall select and input the traffic signs to the traffic sign classification model
* User shall able to classify the traffic signs
* User shall able to identify the result of accuracy on the classification process

### **3.1.4** **System Performance Definition**

System performance is one of the crucial aspects to be considered when performing training in the traffic sign classification model, as one mistake might lead to undesirable consequences like car accident and etc.

Firstly, cross validation method should be applied to during model training by separating the datasets into k, each iteration will use (k-1) for the training folds, the rest will be the validation folds. And under each iteration, the validation folds will be switched alternately. By applying this approach, it prevents the model from overfitting as different subsets of data are used for validation and testing.

Besides that, accuracy of the classification model that we should take into consideration on, to maximize the effectiveness of classifying the traffic signs. Metrics such as F1 score, Recall, Receiver Operating Characteristic (ROC) Curve and Area Under the Curve (AUC) in our training model to understand the robustness of the trained model. For example,

Other than that, computational time should be considered in the process of model training. For example, the faster the time taken for model to classify the traffic sign the better it is, specifically when applying in the real time environment.

## 3.2 System Block Diagram

A diagram of a machine

Description automatically generated

*Figure 3.2.1 Block Diagram of the Proposed Method*

3.2.1 Data Ingestion

Data ingestion involves reading the TSRD-Train Annotation and test annotation then transforming the text format file into a CSV format file. The format of the TSRD-Train annotation is (filename, width, height, Bounding rectangle top-left X, Bounding rectangle top-left Y, Bounding rectangle bottom-right X, Bounding rectangle bottom-right Y) for each line. The image will be undergo image cropping in the next phase .

3.2.2 Images Crop

By following the filtered CSV file’s bounding box positions, training and testing images will then be cropped and saved into respective folder.

3.2.3 Data Split

The training dataset will then be split into category of training, validation which is 85% and 15% respectively. Training set will be used to train the model , validation set will be used in tuning the model’s hyperparameter and monitor for overfitting during the training. Lastly , the test set will be held out entirely during the training and it will be used at the end to evaluate the model performance .

3.2.4 Images Resize

In image preprocessing , image will be resized into a standard size to 128x128 pixels to preserve the necessary spatial information .

3.2.5 Noise Reduction

Bilateral Filter is used to reduced noise and smoothen the image while maintaining edges [8]. The parameter of diameter of the pixel neighbourhood (d) , color space filtering (sigmacolor) , spatial distance filtering (sigmaSpace) are set to 9 , 75 and 75 respectively . d mean it look at a 9x9 neighbourhood around each pixel which sigmaColor will smooth the pixel the surround with different color which help to reduce noise in the image. Lastly, sigmaSpace will smooth the larger regions of pixel while still preserving the edges .

3.2.6 Grayscale Conversion

The images will then undergo grayscale conversion, and it helps to reduce the complexity and highlight features as it will help in undergo texture analysis. Furthermore, it helps in training efficiency.

3.2.7 CLAHE

Contrast Limited Adaptive Histogram Equalization (CLAHE) will be applied to every image to enhance the contrast, improving the visibility and quality of features.

3.2.8 Normalization

Scaling the pixel values between 0 and 1 in images helps stabilize the training and increase the speed of the model.

3.2.9 One-hot Encoding

The category labels are converted into a binary matrix representation. This is a necessary step for the model to interpret the labels correctly during the training process.

3.2.10 CNN Model Architecture

A diagram of several papers

Description automatically generated with medium confidence*Figure 3.2.10 CNN Model Architecture*

The Figure above shows our CNN architecture for traffic sign classification. The input is a sequence of images with width 32, height 32 and depth 1. The images will go through four convolutional layers, four activation layers with ReLu function, two Max Pooling Layer and a fully connected layer.

Convolutional layer is the layer for feature extraction [9]. It applies filters, known as kernels to the input images [9]. The kernel used in this model is 3 x 3 since smaller kernels can capture fine details effectively and it is computational efficiency [9]. The number of filters applied will be increasing successively (32, 64, 128, 256) to capture increasingly complex features [9]. This layer will calculate the dot product between input patch and kernel weight to produce a feature map [9].

Activation layer is the layer to add non-linearity to the output of convolutional layer [9]. It allows the CNN to learn the patterns and relationships of the data [9]. Therefore, the selected activation function should be able to identify and differentiate the significant feature. In this model, Rectifier linear unit (ReLu) function is selected among Tanh, Sigmoid, etc [9]. The most important reasons of choosing this function are due to its less costly computation and simpler mathematical operations to change all negative values to positive numbers [9].

*f(x)ReLU = max(0, x)*

Pooling layer is to create feature maps with smaller dimensions while maintaining the important features information [9]. This can reduce the number of parameters, thus decreasing the computation cost [9]. Max pooling is chosen to extract the maximum value on the feature map covered by a fixed size of window (2 x 2).

Fully connected layer is to aggregate all the features from previous layers for later classification task. Softmax function is picked between Sigmoid function for multi-class classification. The model is compiled with Adaptive Moment Estimation (ADAM) optimizer.

During model training stage, the hyperparameters, including batch size and number of epochs are tuned to achieve high accuracy and high performance. Batch size is the number of samples to pass the layers at the same time while number of epochs are the number of iterations to pass the layers [9]. In this model, a common 32 batch size is initially set since it does not consume more memory space and followed by a standard 50 epochs to avoid overfitting [9]. These batch size and epochs will be tuned accordingly until high accuracy and performance is achieved.

3.2.11 Training

Training is where the process of images and labels being feed to CNN model , the time it takes depends on the batch size , epochs and there will be validation data to help decrease the training loss

3.2.12 Model Evaluation

After training finished, model evaluation is carrying out by printing the history of validation, which is the accuracy of the model on the validation set after the last epoch .

Moreover, the model is evaluated by returning the final loss of training and validation and accuracy of the model then being plotted to visualize the learning process which help in generalize well to unseen data during training

Lastly , The categorical cross-entropy which is a loss function will be measuring how well the model’s prediction and match tat with the true class labels . The lower loss mean the better prediction/. The loss function can refer to the figure 3.2.11



Figure 3.2.11 Multiclass Cross Entropy Loss

printing the history of the validattesting the model with test data . By using cross-entropy loss formulae , it can Accuracy, precision , recall , and support will be evaluated .

3.2.13 Hyperparameter Tuning

The comparison between training and testing accuracy can identify that the model is underfitting or overfitting. In [10], it specifies that some of the widely used methods to overcome this issue, which are dropout, batch normalization, size of dataset, max pooling and relu layer. According to the method proposed in [10], the comparison between SGD and Adam optimizer with dropout rate and epochs (=10) is used to figure out the best hyperparameter in its case. As an illustration, a figure is taken from the article as shown as below [10]: On the other hand, in [11], the hyperparameter tuning of the proposed model is using Bayesian Optimization.

A black text on a white background

Description automatically generatedwhere x is the hyperparameter.

According to [11], by using scikit-optimize (learning rate, number of fully connected Dense layers, number of nodes for each Dense layer and activation function) with Bayesian optimization can find minimum value of f(x). As an illustration, a figure is referenced from [11] as shown as below:

A graph with a line graph

Description automatically generated

Figure 3.2.13.2 A convergence plot of Bayesian optimization

From the figure above, 13th call has the minimum value, which is the best parameter grid that has highest accuracy and lowest lose value. As an illustration, a table is referenced from [11] as shown as below:

A table with numbers and a number of functions

Description automatically generated

Figure 3.2.13.3 A table view of loss and accuracy by using Bayesian optimization

3.2.14 Performance evaluation

In this process, the model can be evaluated based on various performance metrics, which are accuracy, precision, recall, F1 score and confusion matrix. As an illustration, the equation of the metrics is referenced from [11] as shown as below:

A group of math equations

Description automatically generated

According to [11] as a reference, the proposed model of this project can be evaluated not only from the GTSRB dataset; moreover, it can compare with other recent studies’ proposed model based on other dataset, which is BTSDB dataset, Chinese traffic sign database, etc. Therefore, this can validate the applicability of our proposed model on different datasets compares to other researchers’ model. As an illustration, some referenced tables are taken from [11] as shown as below:

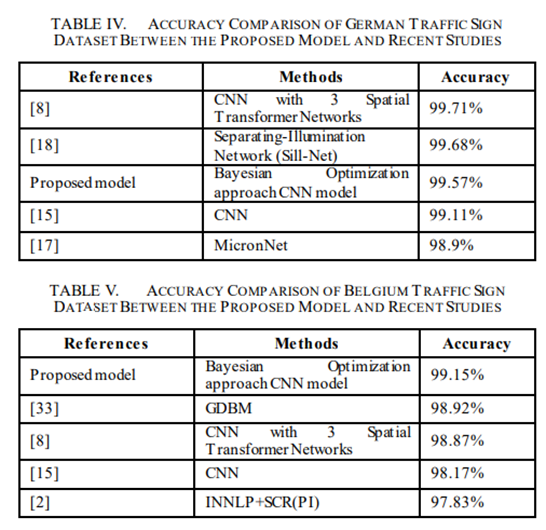


Figure 3.2.14.1 Overview of evaluation between different models based on different dataset

In real time application, the processing time of the model is also another crucial factor. Therefore, it is essential to compare our proposed model with other studies’ model as [5]. As an example, a table is referenced from [5] as shown as below:

A table with numbers and text

Description automatically generated

Figure 3.2.14.2 Comparison table of processing time evaluation between different models

### **3.2.15 Model/Output**

In this stage, the proposed model can save and export out by using TensorFlow; therefore, it can be easily loaded and implemented in real time application.

# CHAPTER 5 SYSTEM IMPLEMENTATION

## 5.1 Hardware Setup

|  |  |
| --- | --- |
| **Description** | **Specifications** |
| Model | VivoBook\_ASUSLaptop X521FL\_S533FL |
| Processor | Intel Core i5-10210U |
| Operating System | Windows 10 |
| Graphic | NVIDIA GeForce MX250 |
| Memory | 4GB DDR4 RAM |
| Storage | 1TB SATA HDD |

Table 5.1.1 Hardware Setup

## 5.2 Software Setup

|  |  |
| --- | --- |
| **Description** | **Specifications** |
| Operating System | Kaggle Environment |
| Programming Language | Python 3.9 |
| Development Environment | Kaggle Notebooks |
| Deep Learning Framework | TensorFlow 2.16.1, Keras 3.3.3 |
| Image Processing Libraries | OpenCV 4.10.0 |
| Data Manipulation Libraries | NumPy 1.26.4 , Pandas 2.2.2 |
| Data Visualization | Matplotlib 3.7.5 |
| Machine Learning Library | Scikit-learn 1.2.2 |
| Dataset | Traffic Sign Recognition Dataset (TSRD) |
| GPU Acceleration | GPU P100 |

Table 5.2.1 Software Setup

## 5.3 Setting and Configuration

Traffic Sign Recognition Dataset (TSRD) consists of 6,164 images of 58 distinct traffic sign categories, divided into a training set of 4,170 images and a testing set of 1,994 images. Each image in the dataset is annotated with the coordinates of the traffic sign and its category.

1. Environment Setup

We begin by setting up the necessary Python environment for loading, preprocessing, and training the model using the dataset. Below is a list of required libraries, along with the code to install them if they are not already available.

* NumPy: For numerical operations and matrix manipulations.
* Pandas: For data manipulation and analysis.
* OpenCV: For image processing.
* Matplotlib: For data visualization.
* Scikit-learn: For splitting the data and encoding the labels.
* TensorFlow: For building and training the deep learning model.
* Keras-Tuner: For hyperparameter tuning.

## 5.4 System Operation (with Screenshot)

### 5.4.1 Data Exploration

A screenshot of a graph

Description automatically generated

Figure 5.4.1 Train Class Count Distribution

This code reads CSV files containing training and test data, then prints the first few rows of the training data to verify its structure. It also visualizes the distribution of classes in the training dataset. In this case, the class distribution shows an imbalanced class count.

### 5.4.2 Data Preprocessing

A screenshot of a computer program

Description automatically generated

A blue sign with white images

Description automatically generated A close-up of a sign

Description automatically generated

Figure 5.4.2 Image preprocessing

This code defines functions for various image preprocessing steps, including resizing, gray scaling, noise reduction, contrast enhancement, and normalization. These steps prepare images for model training by improving quality and consistency. Preprocessing improves image quality and consistency, which is essential for accurate model training and evaluation. Visualizing images before and after preprocessing helps assess the effectiveness of these steps.

### 5.4.3 Training and evaluation

A screenshot of a computer program

Description automatically generated

A screenshot of a graph

Description automatically generated

Figure 5.4.3 Model performance insight

The model is trained using the training dataset (X\_train and y\_train\_one\_hot) over 10 epochs with a batch size of 32. During training, the model's performance is monitored on a validation set (X\_val and y\_val\_one\_hot) to evaluate how well it generalizes to new data. The training process is complete once the specified number of epochs is reached. The final validation accuracy is reported to provide an indication of how well the model has learned.

After training, the model's performance is evaluated on the validation set to determine the final loss and accuracy. These metrics are crucial as they provide insights into the model’s performance on data it has not seen during training. The code then plots the training and validation loss and accuracy over the epochs, which helps visualize the model's learning curve and its ability to generalize.

The plots are divided into two parts: one showing the loss and the other showing the accuracy. The loss plot displays both training and validation loss, indicating how the model’s error changes over time. The accuracy plot shows how the model’s performance improves as training progresses. These visualizations are essential for understanding whether the model is overfitting or underfitting.

### 5.4.4 Test Evaluation

A screenshot of a computer code

Description automatically generated A table of numbers with numbers in it

Description automatically generated with medium confidence A screenshot of a computer

Description automatically generated

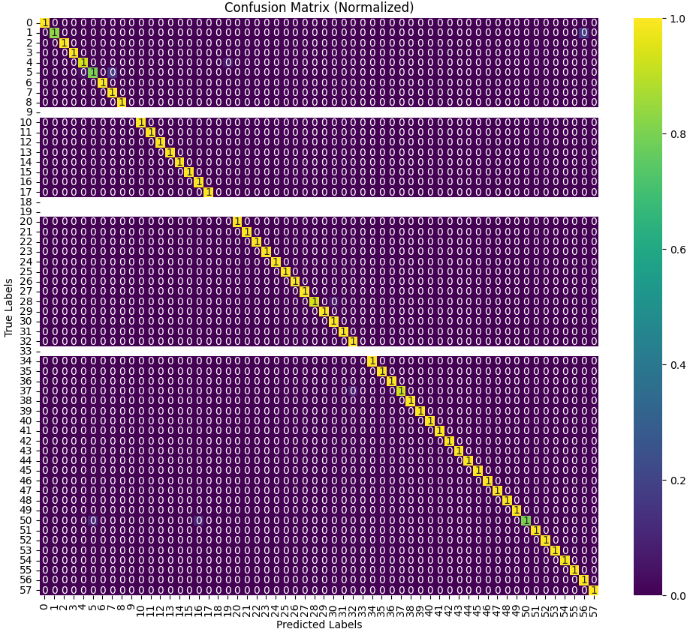


Figure 5.4.4 Model Evaluation

From figure 5.4.4 , The model's performance is assessed using the test set to determine its loss and accuracy which is 0.0718 and 0.986 respective. Predictions are made on the test set, and the predicted probabilities are converted to class labels. A classification report is generated to evaluate precision, recall, F1-score, and support for each class then the overall accuracy of the model is computed which is 0.986. Lastly , The confusion matrix is calculated and normalized to better understand the model's performance across different classes.

### 5.4.5 Hyperparameter Tuning

A screenshot of a computer program

Description automatically generated

A close-up of a computer screen

Description automatically generated

Figure 5.4.5 Hyperparameter Tuning

From Figure 5.4.5 , The Random Search tuner is instantiated to explore different hyperparameter configurations. The tuning process involves running several trials, each with a different set of hyperparameters, to identify the configuration that achieves the highest validation accuracy. The search process is conducted over 10 epochs to ensure sufficient training of each configuration. After completing the tuning process, the best model is selected based on its validation accuracy. The model's performance is evaluated, and the validation accuracy is reported which is 1.0.

### 5.4.6 Model Output (DEMO)

A screenshot of a computer program

Description automatically generated

A black and white sign with a arrow

Description automatically generatedA triangle sign with a letter on it

Description automatically generatedA blurry image of a arrow in a circle

Description automatically generatedA number on a sign

Description automatically generated

Figure 5.4.6 model output (demo)

From figure 5.4.6 , Five images are randomly selected from the test dataset for demonstration purposes. These images are chosen to show how the model performs on different examples it has not seen during training. For each selected image, the model predicts the class and compares it to the true class label. The predictions are visualized by displaying the images along with their predicted and true labels. Additionally, the top three predicted classes and their associated probabilities are displayed, giving an overview of the model’s confidence in its predictions.

## 5.5 Implementation Issues and Challenges

In the implementation phase of this proposed project, we have one challenges . One of the few challenges faced is during the data ingestion by reading the TSRD-Train Annotations and transforming the data into a CSV format that need to be particularly due to the need for filtering out the unnecessary files in the need of ensuring the correct format and matching the CSV entries with available training images adding the complexity.

A graph of number of numbers and a number of people

Description automatically generated with medium confidence

Figure 5.5.1 Train Class Count Distribution

## 5.6 Concluding Remark

In conclusion, this project successfully implemented a traffic sign recognition model with an accuracy of 0.986. The challenges faced during data ingestion and class imbalance were addressed through careful preprocessing and hyperparameter tuning. The model demonstrates high accuracy and efficiency, making it a suitable candidate for real-world traffic sign recognition systems.

# CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION

## 6.1 System Testing and Performance Metrics

In this section, we evaluate the performance of the model using various metrics to understand its effectiveness and accuracy. We assess how well the model performs on the test set by analysing key performance indicators such as loss, accuracy, precision, recall, F1-score, and overall accuracy.

## 6.2 Testing Setup and Result

The model's performance is evaluated using a set of testing metrics. The test loss is 0.0718 and the test accuracy is approximately 98.6%, indicating that the model performs exceptionally well on the test data.

To further evaluate the model’s performance, a classification report is generated which includes precision, recall, and F1-score for each class. The report shows high precision and recall for most classes, although some classes (e.g., classes 9, 18, 19, and 33) have very low values. This indicates that the model struggles with certain classes, which is reflected in the confusion matrix. The confusion matrix is normalized to visualize the proportion of correct versus incorrect predictions across all classes, using a heatmap for better clarity.

## 6.3 Project Challenges

One of the significant challenges encountered in this project is the imbalance in class distribution within the training set. With 58 classes, the dataset exhibits a significant imbalance, particularly with class 28 having a much higher count compared to others like classes 9, 25, and 53. This imbalance can skew the performance metrics and lead to lower accuracy for underrepresented classes. The evaluation phase may be impacted by this imbalance, leading to a model that performs well overall but poorly on specific, less-represented classes.

## 6.4 Objectives Evaluation

The objectives of this project include evaluating the model's performance using various metrics, understanding the impact of class imbalance, and improving model performance through hyperparameter tuning. The model has shown high accuracy and good performance metrics in general, but there is room for improvement in handling less-represented classes.

## 6.5 Comparison with Previous Work

In the field of traffic sign recognition, the effectiveness of a model is assessed through various metrics such as accuracy, false alarm rates, computational efficiency, and the ability to handle specific challenges. This comparison evaluates the performance of our model against a K-D tree classifier[7] , shedding light on their respective strengths and weaknesses.

Accuracy is a primary metric for evaluating model performance, and in this regard, our model demonstrates a significant advantage. Achieving an impressive accuracy of 98.6%, our model outperforms the K-D tree classifier, which recorded an accuracy of 92.7%. This superior accuracy suggests that our model has a more robust capability to generalize across various traffic sign categories. The K-D tree classifier, on the other hand, faced difficulties with certain classes, particularly dynamic signs, which were adversely affected by external factors such as illumination changes. This limitation highlights a critical area where the K-D tree classifier falls short compared to our model.

When considering false alarms, the K-D tree classifier was explicitly engineered to minimize false positives. It succeeded in this aspect by achieving a reduction rate between 94-100%. This impressive figure underscores its effectiveness in reducing incorrect classifications, which is crucial for minimizing erroneous alerts in real-world applications. Our model, while not specifically optimized for false alarm reduction, also exhibits strong performance in terms of precision and recall metrics. This suggests that our model is adept at handling false positives, although a direct comparison of false alarm rates would require additional data to fully substantiate this.

Computational efficiency is another important aspect of model evaluation. The K-D tree classifier processes each image in 17.9ms, while our model achieves a slightly faster prediction time of 17ms per image. This marginal difference indicates that both models operate with comparable efficiency, with our model showing a slight edge in terms of speed.

Finally, both models faced unique challenges in their classification tasks. The K-D tree classifier struggled with dynamic signs due to illumination changes, an issue that impacts its performance in varying real-world conditions. Conversely, our model's primary challenge was dealing with imbalanced classes, a common problem in multi-class classification scenarios. While imbalanced classes can affect the model's ability to accurately classify underrepresented categories, this challenge is somewhat inherent to complex classification tasks and is an area where ongoing refinement could yield significant improvements.

In conclusion, our model exhibits superior accuracy and competitive computational efficiency compared to the K-D tree classifier. While both approaches have their strengths and face distinct challenges, our model’s overall performance suggests it is well-suited for traffic sign recognition tasks, particularly when considering accuracy and false alarm handling. However, addressing class imbalance remains a crucial area for future enhancement to further improve the model's effectiveness across all classes.

## 6.6 Concluding Remark

In conclusion, while the model demonstrates strong overall performance with high accuracy and good precision and recall for most classes, the challenge of class imbalance remains. Addressing this imbalance through techniques like data augmentation or resampling could improve the performance for underrepresented classes and lead to a more robust model.

# CHAPTER 7 CONCLUSION AND RECOMMENDATION

## 7.1 Conclusion

In this chapter, we summarize the findings from the previous analyses and testing phases of the project. The system's performance metrics, including accuracy, loss, and classification report, indicate that the model performs well overall, with a high accuracy rate and robust predictions across various classes. However, challenges such as imbalanced class distribution were identified, which have impacted the accuracy of predictions for certain classes.

The performance evaluation revealed that while most classes were well-recognized, some classes, particularly those with fewer samples, exhibited lower precision and recall. This is a common issue in machine learning, particularly when dealing with datasets where class distribution is skewed. The confusion matrix and classification report highlighted specific areas where the model struggled, providing valuable insights into its weaknesses and areas for improvement.

## 7.2 Recommendation

To address the identified challenges and improve the model's performance, several recommendations are proposed to enhance the robustness and accuracy of the model. These recommendations focus primarily on dataset enhancement and handling class imbalance, which are critical factors influencing model performance.

Firstly, acquiring a better dataset is essential. A well-balanced dataset is crucial for training models that perform consistently across all classes. By seeking additional data sources or exploring alternative datasets that offer a more balanced distribution of classes, we can ensure that the model is exposed to a representative sample of each class. This not only helps in achieving better generalization but also reduces the likelihood of the model developing biases towards more frequently represented classes. A more balanced dataset enhances the model's ability to learn from all classes equally, leading to improved overall performance.

In addition to acquiring a better dataset, augmenting the existing dataset is a valuable approach. Data augmentation involves applying various transformations to the training data, such as rotation, scaling, cropping, and flipping. These techniques artificially increase the diversity of the training set, which can be particularly beneficial for underrepresented classes. Augmentation helps the model to become more robust and better equipped to handle variations in the data that it may encounter in real-world scenarios. By artificially expanding the dataset, we can improve the model's ability to generalize and enhance its performance on classes with fewer samples.

# REFERENCES

[1] Y. Yang, H. Luo, H. Xu, and F. Wu, “Towards Real-Time Traffic Sign Detection and Classification,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 7, pp. 2022–2031, Jul. 2016, doi: <https://doi.org/10.1109/tits.2015.2482461>.

[2] H. K. P. P., S. Ravindran, and V. Vijean, "Traffic Sign Classification for Road Safety using CNN," 2024 International Conference on Emerging Systems and Intelligent Computing (ESIC), 2024, pp. 462-465. doi: 10.1109/ESIC60604.2024.10481587.

[3] A. de la Escalera, L. E. Moreno, M. A. Salichs, and J. M. Armingol, “Road traffic sign detection and classification,” IEEE Transactions on Industrial Electronics, vol. 44, no. 6, pp. 848–859, Dec. 1997, doi: <https://doi.org/10.1109/41.649946>.

[4] Hamza, A., & Nawal, S. (2024). Traffic Sign Classification using Deep Learning Comparative Study. *Procedia Computer Science*, *233*, 939–949. <https://doi.org/https://doi.org/10.1016/j.procs.2024.03.283>

[5] Hannan, M. A. & Wali, Safat & Pin, Tan & Hussain, Aini & Samad, Salina. (2014). Traffic Sign Classification Based On Neural Network for Advance Driver Assistance System. Przeglad Elektrotechniczny. 1. 169-172. 10.12915/pe.2014.11.44.

[6] M. Swathi and K. V. Suresh, “Automatic traffic sign detection and recognition: A review,” in 2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET), Feb. 2017, pp. 1–6. doi: 10.1109/ICAMMAET.2017.8186650.

[7] F. Zaklouta and B. Stanciulescu, “Warning traffic sign recognition using a hog-based K-d tree,” in *Intelligent Vehicles*, 2011.

‌

[8] “Python | Bilateral Filtering,” *GeeksforGeeks*, Sep. 02, 2019. https://www.geeksforgeeks.org/python-bilateral-filtering/

[9] L. Alzubaidi *et al.*, “Review of deep learning: concepts, CNN architectures, challenges, applications, future directions,” *Journal of Big Data*, vol. 8, no. 1, Mar. 2021, doi: <https://doi.org/10.1186/s40537-021-00444-8>.

‌[10] S. Mehta, C. Paunwala and B. Vaidya, "CNN based Traffic Sign Classification using Adam Optimizer," 2019 International Conference on Intelligent Computing and Control Systems (ICCS), Madurai, India, 2019, pp. 1293-1298, doi: 10.1109/ICCS45141.2019.9065537. keywords: {Training;Testing;Color;Convolutional neural networks;Shape;Convolution;Adam optimizer;Belgium traffic sign dataset (BTSD);Convolutional Neural Network (CNN);Dropout;Traffic Sign Classification},

[11] Si Thu Aung, Jartuwat Rajruangrabin and Ekkarut Viyanit, “Sequential Model-based Optimization Approach Deep Learning Model for Classification of Multi-class Traffic Sign Images” International Journal of Advanced Computer Science and Applications(IJACSA), 14(7), 2023. [http://dx.doi.org/10.14569/IJACSA.2023.0140764](https://dx.doi.org/10.14569/IJACSA.2023.0140764)[3] Hannan, M. A. & Wali, Safat & Pin, Tan & Hussain, Aini & Samad, Salina. (2014). Traffic Sign Classification Based On Neural Network for Advance Driver Assistance System. Przeglad Elektrotechniczny. 1. 169-172. 10.12915/pe.2014.11.44.

[12] C. Liu, S. Li, F. Chang, and Y. Wang, “Machine Vision Based Traffic Sign Detection Methods: Review, Analyses and Perspectives,” IEEE Access, vol. 7, pp. 86578–86596, 2019, doi: <https://doi.org/10.1109/access.2019.2924947>