

UNDERGRADUATE PROJECT REPORT

|  |  |
| --- | --- |
| **Project Title:** | **A Convolutional Neural Network Approach for the Recognition of Traffic Signs** |
| **Surname:** | **Wu** |
| **First Name:** | **Yile** |
| **Student Number:** | **201918020102** |
| **Supervisor Name:** | **Happy Nkanta Monday** |
| **Module Code:** | **CHC 6096** |
| **Module Name:** | **Project** |
| **Date Submitted:** | **May 5, 2023** |

# **Declaration**

Here, students would sign a statement indicating that they adhered to appropriate academic conduct in carrying out their final project.

# **Acknowledgment**

Firstly, I want to thank my supervisor, Dr. Happy Nkanta Monday. Without the help and great advice, this project cannot be done. Secondly, I really appreciate Oxford Brooks University, department of software engineering. It is the education resources that the school provided helped the development of this project. Moreover, I want to thank Chengdu University of Technology. The lovely environment that this school provided is precious. Lastly, I am very grateful for my family. They always support my study even though they do not understand. It is their support makes my further study possible.

# **Table of Contents**

[**Declaration** i](#_Toc131529425)

[**Acknowledgment** ii](#_Toc131529426)

[**Table of Contents** iii](#_Toc131529427)

[**Abstract** v](#_Toc131529428)

[**Abbreviations** vi](#_Toc131529429)

[**Glossary** vii](#_Toc131529430)

[**Chapter 1 Introduction** 1](#_Toc131529431)

[**1.1** **Background** 1](#_Toc131529432)

[**1.2** **Aim** 4](#_Toc131529433)

[**1.3** **Objectives** 4](#_Toc131529434)

[**1.4** **Project Overview** 4](#_Toc131529435)

[1.4.1 Scope 4](#_Toc131529436)

[1.4.2 Audience 4](#_Toc131529437)

[**Chapter 2 Background Review** 5](#_Toc131529438)

[**Chapter 3 Methodology** 7](#_Toc131529439)

[**3.1** **Approach** 7](#_Toc131529440)

[**3.2** **Technology** 7](#_Toc131529441)

[**3.3** **Project Version Management** 7](#_Toc131529442)

[**Chapter 4 Results** 7](#_Toc131529443)

[**4.1** **Dataset** 7](#_Toc131529444)

[**4.2** **Data Preprocessing** 8](#_Toc131529445)

[**4.3** **Model Design** 8](#_Toc131529446)

[**4.4** **Model Training** 10](#_Toc131529447)

[**4.5** **GUI Design** 10](#_Toc131529448)

[**4.6** **Testing and Evaluation** 12](#_Toc131529449)

[**Chapter 5 Professional Issues** 13](#_Toc131529450)

[**5.1** **Project Management** 13](#_Toc131529451)

[5.1.1 Activities 13](#_Toc131529452)

[5.1.2 Schedule 15](#_Toc131529453)

[5.1.3 Project Data Management 15](#_Toc131529454)

[5.1.4 Project Deliverables 15](#_Toc131529455)

[**5.2** **Risk Analysis** 16](#_Toc131529456)

[**5.3** **Professional Issues** 19](#_Toc131529457)

[5.3.1 Legal Issue 19](#_Toc131529458)

[5.3.2 Social Issue 19](#_Toc131529459)

[5.3.3 Ethnical Issue 19](#_Toc131529460)

[5.3.4 Environmental Issue 19](#_Toc131529461)

[**Chapter 6 Conclusion** 20](#_Toc131529462)

[**References** 21](#_Toc131529463)

[**Appendices** 23](#_Toc131529464)

# **Abstract**

Driverless car has been an increasing popular topic nowadays. And the traffic signs recognition system enables the automatically identification of traffic signs for driverless car system, also generates major discussion. As a matter of fact, the system may able to save drivers’ lives when their make mistakes in identifying of traffic signs. Therefore, the traffic signs recognition system can be used not only in driverless cars, but also as an aid to help drivers identify traffic signs on the road. To achieve this goal, the deep learning technology, convolutional neural networks is used to identify traffic signs for drivers, and it is required to be equipped with the quality of high accuracy and high efficiency. Therefore, the lightweight convolutional neural networks model, MobileNetV2 is used to be developed in this project since it has the advantage of less computation and reducing latency. Also, to better fit the dataset, the attached classifiers is added. The object of this article is to explore the process to generate one lightweight CNN model MobileNetsV2 to identify traffic signs, which is from Chinese Traffic Signs dataset, and design the Graphical User Interface for the model to better understand and operate. As a result, the model achieves the training accuracy of 88.10% and the validation accuracy of 93.75%. And the training loss of 40.78%, validation loss of 17.05%.

***Keywords:*** *traffic signs recognition system, convolutional neural networks, MobileNetsV2, Chinese Traffic Signs datasets*

# **Abbreviations**

CNN: Convolutional Neural Networks

TSR: Traffic Sign Recognition

GUI: Graphical User Interface

ReLU: Rectified Linear Unit

GPU: Graphics Processing Units

IEEE: Institute of Electrical and Electronics Engineers

URL: Uniform Resource Locator

DAS: Driver Assistance System

AI: Artificial Intelligence

ANN: Artificial Neural Network

RGB: Red Green Blue

# **Glossary**

Traffic Signs Recognition system: Traffic Sign Recognition (TSR) is an advanced driver assistance system (ADAS) that recognizes and conveys traffic sign information to drivers

Convolutional Neural Networks: A deep learning algorithm that can perform image recognition.

MobileNetsV2: A convolutional neural network model that aims to achieve a good performance on mobile devices.

Chinese Traffic Signs datasets: A collection of traffic signs images from Chinese Traffic Sign Recognition Database.

Overfitting: It is a problem that often occurs in networks training process when a model trained with too much data and inaccurate data is learned.

Stride: A parameter of the neural network's filter, modifies the number of movements over the image.

Padding: Refer to the number of pixels added to an image when it is processed by the kernel of a CNN.

Epoch: One complete forward and backward pass throughout the whole training dataset.

Batch: Used to divide the dataset to better control.

Imbalanced Dataset: The data in some categories is much higher than others.

Deep learning: Deep learning is a branch of machine learning that is entirely based on artificial neural networks, just as neural networks are designed to mimic the human brain, so deep learning is also a kind of mimic of the human brain.

# **Introduction**

## **Background**

Traffic Sign Recognition (TSR) is becoming more and more promising since the recent interested development of unman driving cars. Firstly, according to market research, the U.S. unman driving car market will swell from $4 billion in 2021 to $186 billion in 2030 industry [1]. Moreover, human error made in traffic sign identification has always been a risk when driving. There are approximately 23,000 avoidable traffic deaths due to traffic signs in the United States each year [2]. However, the advent of artificial intelligence in recent years has updated the vehicle-aided driving system's pre-existing driving mode. The system prevents automotive accidents caused by driver weariness by prompting drivers to execute accurate maneuvers by collecting real-time information about the state of the road. The creation of autonomous vehicles involves the quick and precise identification of traffic signs from digital images in addition to the auxiliary driving systems. Therefore, the development of TSR is indispensable for both the driverless market and for ensuring safe driving.

It is known that TSR is able to detect the location of traffic signs from digital images or video, and then give a specific classification. However, conventional TSR algorithms are facing three challenges in real-time tests. Firstly, being easily restricted by driving situations such as lighting, camera angle, obstruction, driving speed and there is also difficulties in achieving multitarget detection [3]. Secondly, low image quality caused by low resolution, bad weather conditions, and over-illumination or under-illumination [3]. Lastly, it is easy to miss visual objects because of slow recognition speed [4]. However, the limitation of above conventional TSR can be effectively solved with the development of Deep Neural Network.

## **Convolutional Neural Networks**

The term Deep Neural Network or Deep Learning refers to multi layers Artificial Neural Network (ANN) . A Deep Learning model simulates the neuronal architecture of the human brain when processing information. And one of the most popular deep neural networks is Convolutional Neural Networks (CNN) since it has great performance in dealing with image data classification. Because CNN is to obtain abstract features when input propagates toward the deeper layers [5]. For example, in image classification, the edge is most likely to be detected in the first layer, and then the simpler shapes in the second layers, and then the higher level features in the next layers. Therefore, CNN have multiple layers, including input layer, convolutional layer, pooling layer and fully-connected layer [5]. Here is a simple CNN representation to help understand.

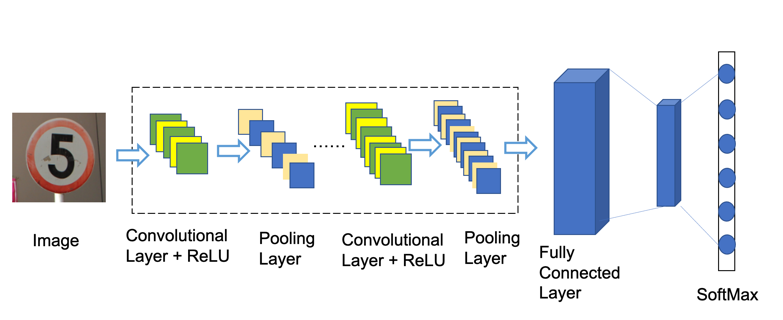


Figure 1. A representation of CNN architecture

### Input Layer

The input Layer could be a colored image or an video. For example, an image with a width and height of 32\*32 pixels and a depth of 3, which is Red Green Blue (RGB) Channel [5]. Generally speaking, an input is with width, height and color channel.

### Convolutional Layer

Firstly, the convolutional layer plays an important role in CNN. The layer parameters focus on the use of convolutional kernels, which are typically small in spatial dimensionality, but spread across the entire depth of the input [6]. The convolutional layer could be visualized as many convolutional kernels, which slide over the image and look for pattens. The following figure is the visual representation of a convolutional layer.

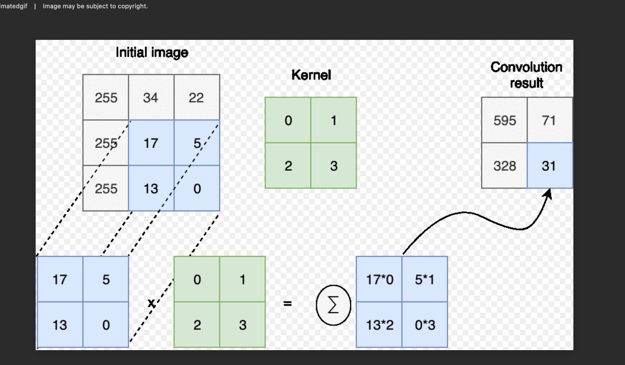


Figure 2. Visual representation of convolutional layer.

### Pooling Layer

After the convolutional layer, it is the pooling layer and it aims to reduce the dimensionality of the representation, and further decrease the number of parameters and the computational complexity [6]. Max-pooling is one of the most common pooling methods with the dimension of 2\*2. It divides the image into sub-regions and returns only the maximum value of the interior of that sub-region.

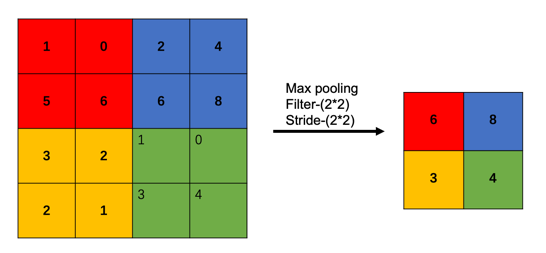


Figure 3. Max-pooling layer example

Another pooling method is average pooling, which computes the average of the elements in the region of feature map covered by the filter. Therefore, while max pooling gives the most prominent feature, average pooling gives the average of features present in a patch.

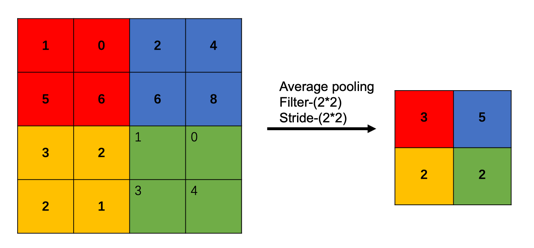


Figure 4. Average pooling layer example

### Fully-Connected Layer

Lastly, fully-connected layer contains neurons of which are directly connected to the neurons in the two adjacent layers, without being connected to any layers within them.

### Activation Function

After passing the image to a convolutional layer, the output is normally passed through an activation function. The non-linearity can be used to adjust or limit the generated output. This layer is used to saturate the output or to limit the generated output.

One of the most popular activation function is sigmoid function, which can convert the output of model into a probability and makes it easier to interpret. Here is the formula.

However, Rectified Linear Unit (ReLU) is the most popular one recently. The functions are as followed. The main reason behind it is the saving of computation. Compared to traditional activation functions, the derivative of ReLU is 1 for a positive input. Therefore, deep neural networks do not need to take additional time for computing error terms during training phase.

### Loss Function

After introducing the architectural of CNN, the methods to evaluate models should be mentioned. And loss functions or cost function is a measure of how well the model predicts the expected outcome. One of the common loss functions used in CNN is cross entropy. It measures the performance of a classification model whose predicted output is a probability value between 0 and 1 [7]. The formula is as followed when the number of classed is 2.

When the number of classes is more than 2, which means it is categorical cross entropy, the formula become as followed.

### Evaluation metrics

Evaluation metric is important in optimal classifier during the classification task and used to optimize the model during the training stage [8]. A confusion matrix is a method to assessing the performance of a model, which compare between the actual values and predicted values made by the model [9]. The confusion matrix has a positive and a negative class. The positive class represents the not-normal class or behavior and the negative class represents normality or a normal behavior. Therefore, TP and TN mean the number of positive and negative instances that are correctly classified in confusion matrix while FP and FN respectively mean the number of misclassified negative and positive instances. And here are some most used evaluation metrics formula.

|  |  |  |
| --- | --- | --- |
| Evaluation Metrics | Formula | Explanation |
| Accuracy |  | The accuracy metric measures the ratio of correct predictions over the total number of evaluated instances. |
| Precision |  | Precision measures the positive patterns that are correctly predicted from the total predicted patterns in a positive class. |
| F1-Score |  | F1-Score measures the harmonic average between recall and precision rates. |

Table 1. Common evaluation metrics

With this deep learning model, TSR can take the road image and extract the important features [4]. And the study findings in TSR assist prevent traffic accidents, safeguard drivers, and effectively and accurately inspect traffic signs on roadways, which saves time, money, and resources. Additionally, it offers technical assistance for auxiliary and unmanned driving. As a result, deep learning-based research is extremely important and beneficial to our daily lives.

The rest of the paper is organized as follows. Section 1.2 covers the aim of this project and section 1.3 discusses the objectives of this study. Section 1.4 explains the significance and targeted audience of this project. Important and relevant literature related to this project is discussed in section 2. Section 3 details the approach, technology and project vision management. Section 4 presents the results while the project management activities, risk analysis and professional issues are presented in section 5.

## **Aim**

The aim of this project is to develop a TSR system based on a MobileNetv2 convolutional neural network model.

## **Objectives**

The objectives are as follows.

1: Finish a background review of the existing CNN technology and current literature.

2: Design the CNN architecture.

3: Preprocess the dataset, using data augmentation.

4: Train and test the CNN architecture with a public dataset named Chinese traffic sign using appropriate technology.

5: Evaluate the model with testing and validation accuracy, testing loss and validation loss.

6: Design and implement a Graphical User Interface (GUI) of the model.

7: Explain the work to related audience.

## **Project Overview**

### Scope

The project seeks to develop a CNN based on MobileNetV2 that is capable of identifying and classifying traffic signs. Developing a computer vision system for TSR with high precision is paramount, since the outputs in TSR system enable to avoid traffic accidents and protect drivers by identifying traffic signs on roads efficiently and accurately. And this system would greatly benefit road traffic management.

### Audience

Firstly, the project definitely would help the drivers to avoid traffic accidents from misidentification of traffic signs. Furthermore, the project would bring a large benefit to government agency, automobile manufacturers, driving school owners and trainees within the domain who can utilize this computer vision technology to recognize traffic with high accuracy.

# **Background Review**

TSR is the detection of the location of traffic signs from digital images or video frames, given a specific classification [10]. Also an important part of a Driver Assistance System (DAS) , which provides drivers with safety and precaution information. However, TSR is challenging since shape distortion due to rapid changes in environment and illumination, background clutter, low resolution, and the changes of camera angle [11]. Therefore, current algorithms are dedicated to improving the speed and accuracy of TSR.

Current TSR methods are divided into two groups: one is based on conventional methods and the other is based on deep learning methods [12]. Conventional machine learning methods usually selected specified visual features and use those features to classify the traffic signs. The features are usually based on hand-crafted features such as color, texture, edge, and other low-level features to detect traffic sign in an image [12]. However, due to the diversity of the traffic sign appearance in driving environment, the problems of illumination changes, color fading of traffic signs, the deformation as well as the occlusion of traffic signs affected traditional methods for traffic sign detection and showed poor performance [12]. Hence, there is still limitation for traditional machine learning to classify traffic signs under real-time condition.

Nevertheless, for better image processing results, deep learning uses multilayer neural networks to automatically extract and learn features of visual objects [13]. Meanwhile, one of the most popular deep learning methods for TSR is CNN models. CNN is a widely used deep neural network that shows great advantages in image processing, and vision recognition, including traffic sign classification [14].

One of common problems for CNN model is to find a balance between accuracy and the network depth since large network with small data would lead to model overfitting. It is an obvious fact that one single layer is not enough for achieving satisfactory accuracy. There is an example of a CNN ensemble to identify traffic signs [15]. The authors use a feed-forward network with six convolutional layers, 3 max pooling layers and 2 fully connected layers. The suggested ensemble model of 6 convolutional layers aggregate the output which is a great technique of improving the accuracy. The authors implemented fully connected hidden layers with dropout between the layers to avoid overfitting problem. The authors claimed that their proposed model achieved 98.11% accuracy for triangular traffic signs and 99.18% accuracy for the circular traffic signs.

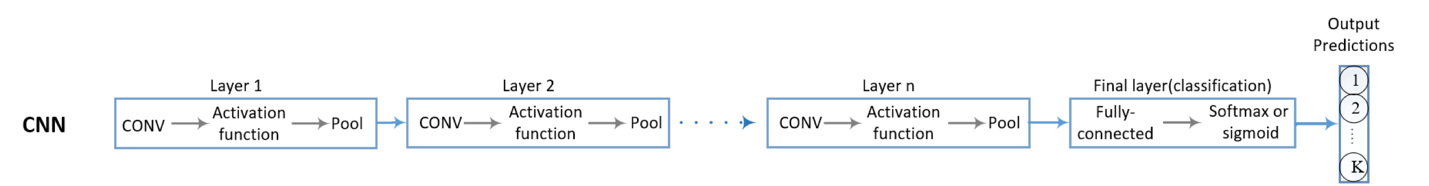


Figure 5. CNN ensemble architecture

However, some model are aimed at being lightweight. And there is one light-weight CNN model for traffic sign recognition [16]. The architecture consists of four convolutional layers, two overlapping max-pooling layers and then followed by one fully-connected layer, which is presented in figure 6. The authors implemented overlapping max pooling and sparsely used stride convolution to made training faster in order to reduce overfitting issue. The benefit of a light-weight structure is to lower computational cost with a considerable depth of four convolutional layers. The proposal model achieved 98.97% accuracy as the authors claimed.

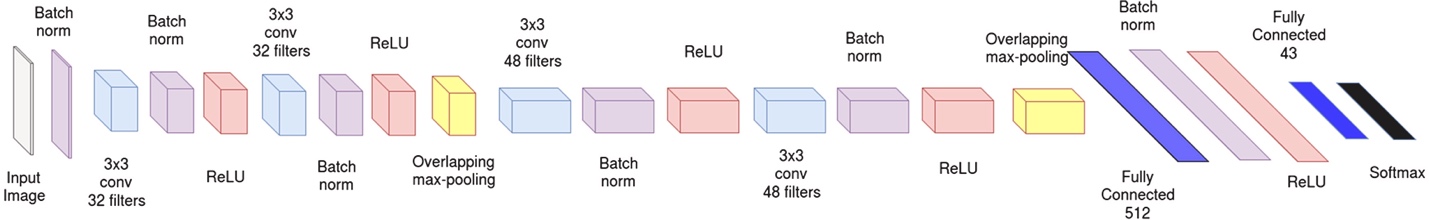


Figure 6. A light-weight CNN architecture

In addition, there is another example of a light-weight CNN model [17]. The network is based on ESPNetv2 network which first takes image to generate convolution feature maps. Then, a deconvolution module is used to aggregate an output feature map at layer 2 with an output feature map at layer 3 to create an enhanced feature map. Next, adopting two improved region proposal networks which include a convolution layer to reduce the number of parameters in the subsequent convolutional layers and a dilated convolution to improve the detection accuracy. In the detection network, adopting pooling layer to adjust the size of proposals to fixed size feature maps. Lastly, fully connected layers are used for classifying proposals. The model achieves the accuracy of 97.02%.

However, although deep learning methods for traffic sign detection performed well in complex driving environments, the performance of large models is not ideal in challenging conditions due to the size of traffic signs and recognition speed [18]. There is a performance comparison of different models in figure 6. It is clear to see that MobileNetV2 has the shortest computation time, meanwhile the accuracy is not greatly reduced compared to other models. Therefore, this project decided to adopt a light weight CNN model, MobileNetV2 to solve these problems.

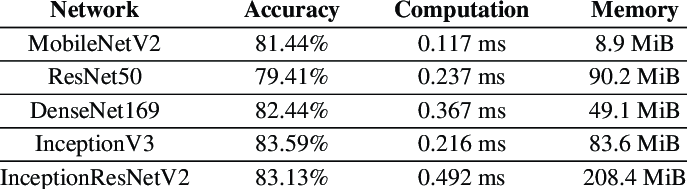


Figure 6. Performance comparison between different network [19]

MobileNets for mobile and embedded version application that are based on depth-wise separable convolutions, aiming to build light weight neural networks [20]. Depth-wise separable convolutions is a depth-wise convolution which is the channel-wise DK\*DK spatial convolution. Then followed by a pointwise convolution, which is 1\*1 convolution to change the dimension. Compared to standard convolution cost, less computation can be achieved but only slightly reduction in accuracy. Based on the above MobileNets, a new mobile architectural, MobileNetV2 is more promising [21]. MobileNetV2 contains the initial fully convolution layer with 32 filters, followed by 19 residual bottleneck layers overall. Moreover, MobileNetV2 introduces a new CNN layer, the inverted residual and linear bottleneck layer. It enables high accuracy in mobile applications. The new layer builds on the depth-wise separable convolutions which is introduced above in MobileNets. More importantly, the MobileNetV2 network is built around this new layer. In MobileNetV2, there are two types of blocks. One is residual block with stride of 1 and the other one is block with stride of 2 for downsizing. There are 3 layers for both types of blocks. The first layer is 1×1 convolution with ReLU6. The second layer is the depth-wise convolution. The third layer is another 1×1 convolution.

# **Methodology**

## **Approach**

In order to reduce the work of computation, the core deep learning model that this project focuses on is MobileNetV2. It is a convolutional neural network model that aims to achieve a good performance on mobile devices. It is based on an inverted residual structure where the residual connections are between the bottleneck layers. Non linearity comes from the intermediate expansion layer which uses lightweight depth-wise convolutions to filter features. Moreover, Keras API was adopted for the project, playing the role of helping to build layers and model. The optimizer utilized in the project is Adam, which could be used to update network weights iterative based on training data instead of the classical stochastic gradient descent procedure.

## **Technology**

The software adopted for coding the model is jupyter notebook and the program language is Python and PyCharm is used to code GUI of the model. The hardware adopted to implement the code is Intel Xeon Cascade Lake (2.5 GHz).

## **Project Version Management**

The source codes would be upload to this URL <https://github.com/Haleywuyile/Traffic-Sign-Recognition> on GitHub.

# **Results**

## **Dataset**

The dataset that apply to the project is Chinese Traffic Signs. This dataset contains 5998 traffic sign images of 58 categories. Each image is a zoomed view of single traffic sign. As for the data processing, the approach is Keras image data generator. It generates batches which contains the data of tensor images and help loop over the data in batches.

## **Data Preprocessing**

To train a CNN model, the first step is to prepare training and testing dataset. The dataset adopted is Chinese Traffic Sign dataset, whose annotation contains image information. Therefore, pandas library that used to process and analysis data is imported. Then change the category type column to string type. In addition, it is necessary to unzip the images in jupyter notebook since the images is zipped into a zip file. Now, the images are ready to be divided into training and testing dataset.

In this step, Image data generator from Keras is used to generate batches of image.

It enables the transformation of images in random and give outputs. Transformation like rotations, scale changes, horizontal flips and translations, are randomly operated by an image data generator. Firstly, process the training dataset. Preprocess input method from mobilenet\_v2 is applied on each input. Then set validation split to 0.1, since part of the training data need to be used as validation data. The model would set apart this part of the training data and not train on it. Then evaluate the loss and model metrics on this data at the end of each epoch. Set 15 for degree range in random rotations, and horizontal position and up-down position panning between 0 and 0.2. Secondly, use flow from data frame to take the path to image directory and generate batches of augmented data. Set the height and width to 224 \*224, and class mode to categorical which makes labels 2D one-hot encoded. Set the batch size to 16 and seed 42. Set the x column to file names and y column to category.

Lastly, process the validation dataset. Create the image data generator for validation, then adopt preprocess input method from mobilenet\_v2 as training dataset and set the validation split to 0.1. Apply flow from data frame to make batches of data normalized as above.

## **Model Design**

After preparing the training and testing data, designing the model is the next step. There are 3 methods to define a model, sequential methos is applied in this case since the flexibility is low when it comes to in-depth debugging. It enables to build the model layer by layer by making array of multiple hidden layers. The CNN model in this project adopted Keras pretrained MobileNetV2 model. For the arguments in MobileNetV2 function, set false to include top which means not include the fully-connected layer at the top of the network. Weight is imagenet since it is pre-training on ImageNet. And the input shape is 224\*224\*3. Moreover, the model of this project is added custom layers and this can be achieved by attaching classifier. The added layers are to adjust the existing model to better carry out current tasks. To make sure TensorFlow would not retrain the MobileNetV2, should set the trainable to false.

After the MobileNetV2, the model includes one flatten layer before adding a dense layer. The reason behind this layer is the former layer output is 4D. Since the dense layer accepts 2D input, so it is necessary to convert the 4D output from the convolution layer to 2D. The next step is to add one batch normalization layer to reduce the learning process and the number of epochs. The last dense layer has 58 units since the dataset has 58 categories. The overall model is presented in the following figure 1. The total params is all the parameters in the network, trainable params is the number of neural that the network would train and non-trainable params is the neural that MobileNetV2 already trained.

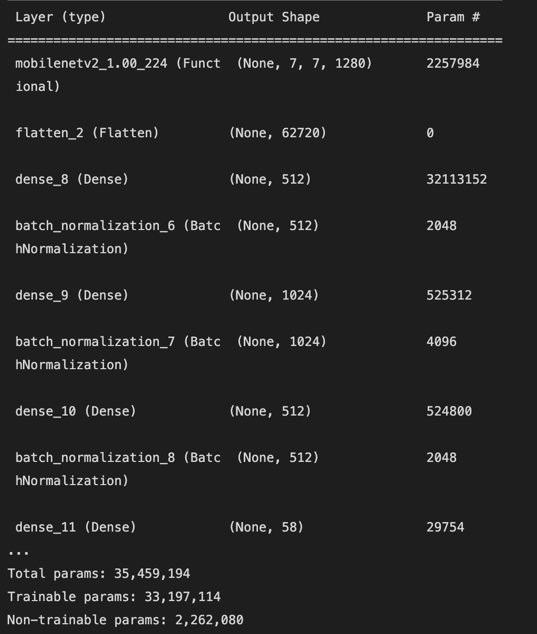


Figure 1. The model summary

After deciding the layers, it is necessary to add activation functions in the hidden layers as well as the output layer. The role of activation function is making non-linearity transformation to the input so that the model can learn. For this model, RELU and Softmax are adopted to introduce the non-linearity. RELU is added in the dense layers and the Softmax is added in the output layer.

The callbacks used are early stopping, ReduceLROnPlateau and model check point, which enable the control of training process. Early stopping could be needed to log performance or to stop training if performance metric reaches a certain threshold. Using ReduceLROnPlateau is to reduce the learning rate when the metric stops improving for long time. And model check point is combined with the model fit, it save the model at certain point so that the model can be loaded and continue training in the status of saved.

## **Model Training**

This model is compiled by the method called compile. There are 3 parameters, the first is model optimizer, which decrease errors generated in training so that the model could perform well. Set the optimizer to Adam, which has the learning rate of 0.001, beta 1 of 0.9 and beta 2 of 0.99. The second parameter is model loss, it focus on the error in the model training process, set it to categorical crossentropy. The last parameter is model metric which calculate the accuracy. Then fit the network in 100 epochs and batch size of 16.

## **GUI Design**

The adopted Python framework to complete GUI is Tkinter, which is the standard Python interface to the GUI toolkit. In order to improve the designability of GUI, style from ttkbootstrap is adopted, which is an extension for Tkinter and equipped with a collection of themes. First, use Tkinter to build the GUI window, which is the foundational element of a Tkinter GUI. Windows are the containers in which all other GUI elements live. Set the window title to “Traffic Sign Recognition” and size to 100\*100. Then the second step is to create widgets, such as text boxes, labels, and buttons, which are all contained inside of windows. Therefore, the upload image button can be created to enable users choose one traffic sign image to identify. The GUI of this step is presented in figure 4. Then use Tkinter command to realize the function of uploading image. Button has a command parameter which enable function assignment. Therefore, upload image function is defined so that user click the button there is response. The GUI of this step is presented in figure 5. Then make the image display in the window as a label. After choosing the image, show the classify button. The GUI of this step is in figure 6. Lastly, define the classify button as the upload button, and add classify function of it. In this step, the image should be processed into the format that the model can take as an input, otherwise it cannot be predicted. First, resize the image to 224\*224, transfer it into array and expend the dimension. Secondly, use image data generator to make the image more fit. Lastly, use predict method of the model to identify the image and print it out. The GUI in this step is in figure 7.

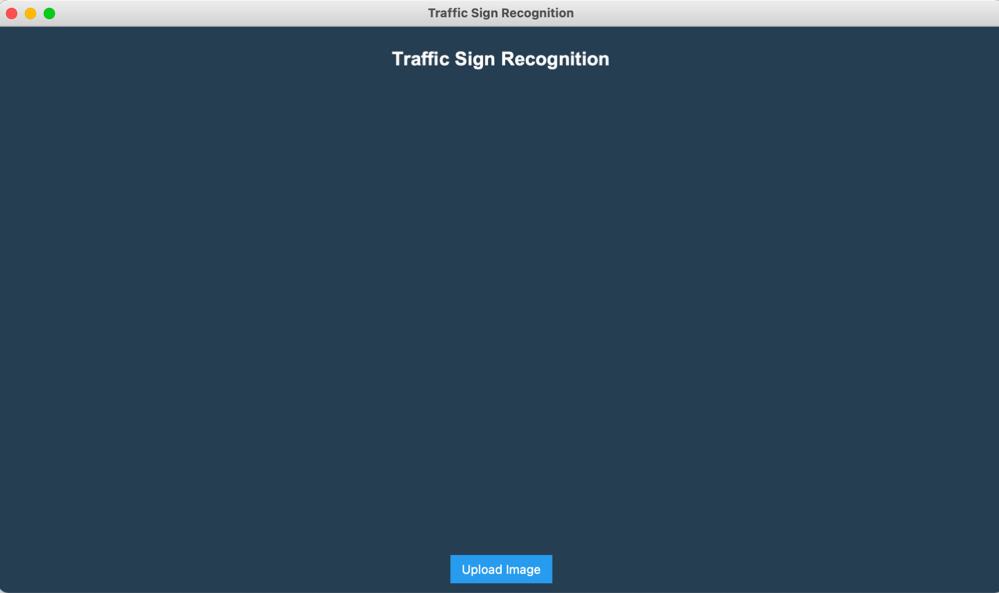


Figure 4. The Start Page of Traffic Sign Recognition GUI

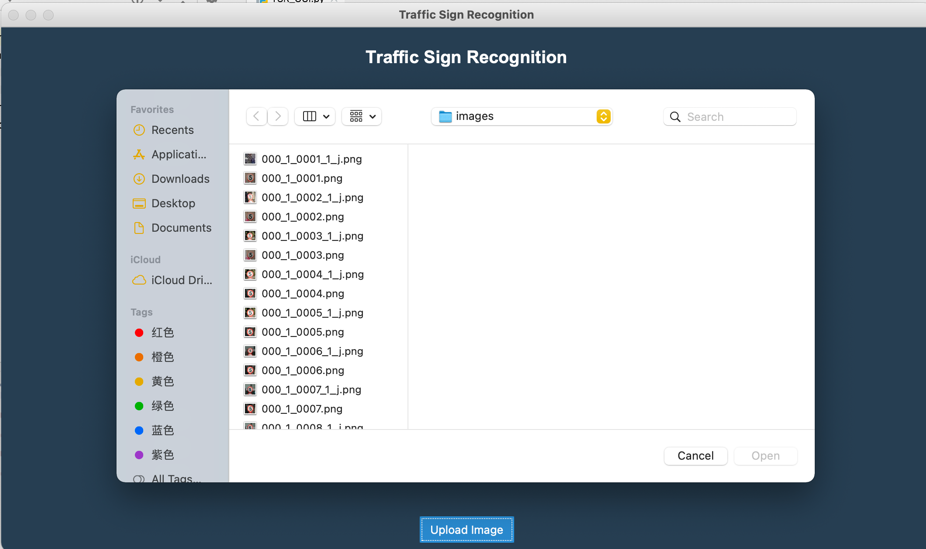


Figure 5. The File Dialog of Traffic Sign Recognition GUI

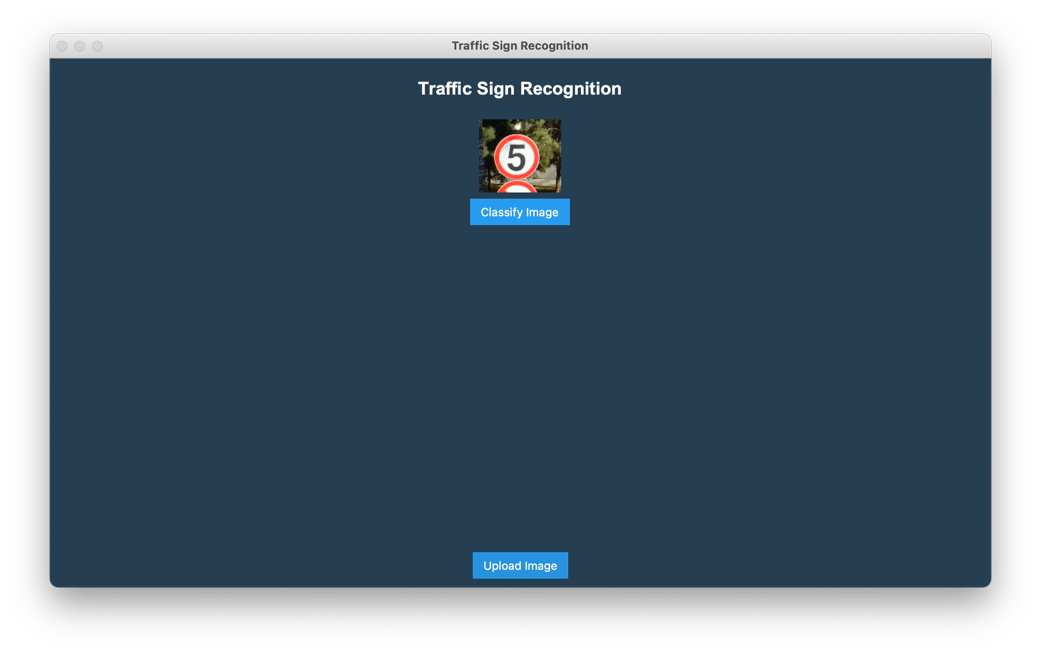


Figure 6. The Classify Button of Traffic Sign Recognition GUI

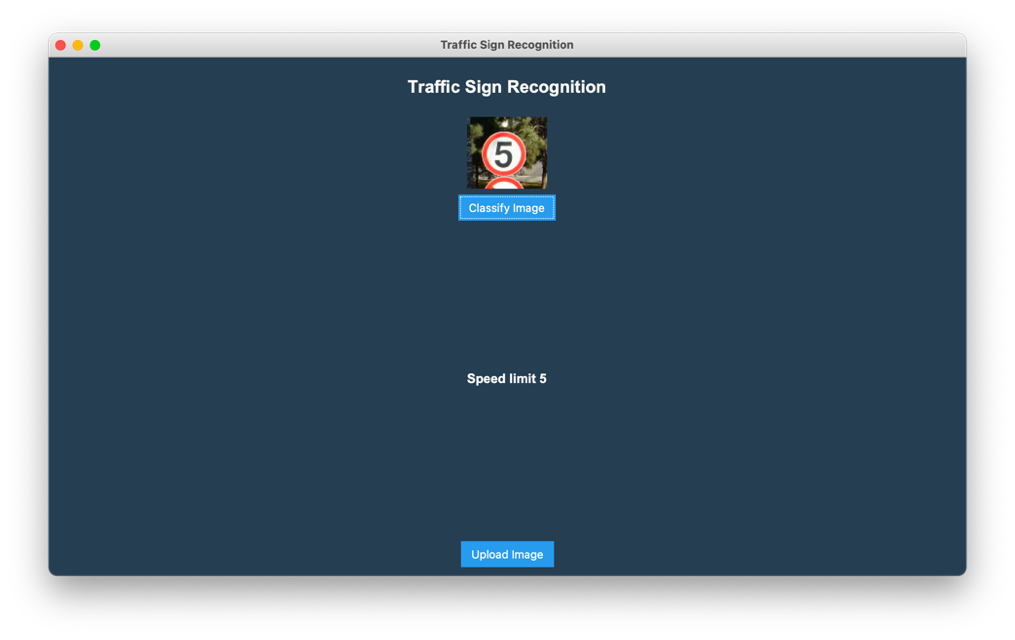


Figure 7. The Classify Result of Traffic Sign Recognition GUI

## **Testing and Evaluation**

Firstly is to test the data to make sure the dataset satisfies the requirement. The Annotation of the dataset provide image properties like file name, width, height and traffic sign coordinates within image and category. And all images originally collected by camera under nature scenes or from BAIDU Street View. Therefore, this dataset is satisfied for traffic sign recognition.

Secondly is to evaluate and test the model. The accuracy metric that uses calculates how often predictions equal labels. This metric creates two local variables, total and count that are used to compute the frequency with which prediction matches actual. This frequency is ultimately returned as binary accuracy. And the model achieves accuracy of 88%. Moreover, to calculate train loss and test loss, cross-entropy loss is used for loss function. And the model achieves the loss of 38%.

To better observe the result of accuracy and loss, it is necessary to plot it using matplotlib. The training and validation accuracy is presented in figure 8. Training and validation is presented in figure 9.

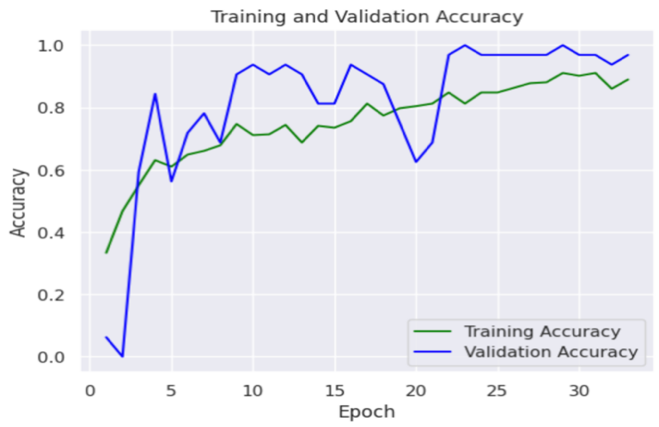


Figure 8. Training and Validation Accuracy

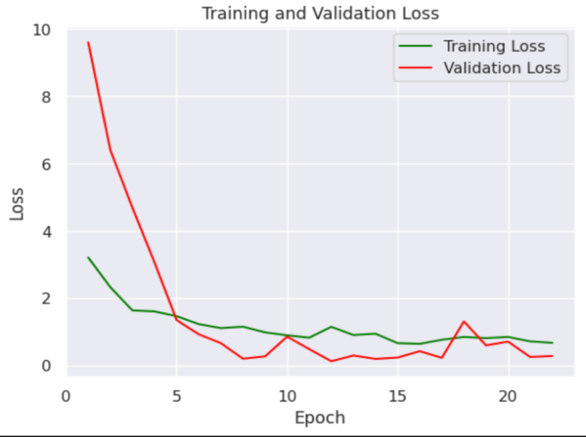


Figure 9. Training and Validation Loss

Traffic sign recognition system would largely be applied to driver-less car. And the accuracy and speed of the recognition result is very important, since the driver needs fast and accurate judgement when the car is driving at high speed, and any error will cause unpredictable consequences to the driver. Therefore, this project adopted a lightweight model MobileNetV2 as the core deep learning neural networks. This model has the advantage of reduced computation but slight decrease in accuracy. Moreover, some traffic signs are not common such as speed limit 5, cycle crossing. Therefore the dataset is highly imbalance, which becomes a problem for classification. However, the project adopted image data generator to increase the dataset. Lastly, the model achieves the training accuracy of 88.10% and the validation accuracy of 93.75%. In the future, the dataset should be more balanced and the accuracy of the model should achieve higher.

# **Professional Issues**

## **Project Management**

### Activities

|  |  |  |
| --- | --- | --- |
| Objective | Complete Task | Uncomplete Task |
| 1: Finish a background review of the existing CNN technology and current literature. | Conduct a systematic research of current CNN models.  Complete a feature competitive analysis.  Complete a literature review.  Conduct the requirement gathering. |  |
| 2: Design the CNN architecture. | Draw a scratch of the traffic sign recognition model including details the CNN architecture.  Study the dataset and summary features.  Complete GUI of the proposed product. |  |
| 3: Preprocess the dataset, using data augmentation. | Pre-process the images from dataset using image data generator. |  |
| 4: Train and test the CNN architecture with a public dataset named Chinese traffic sign using appropriate technology. | Realize the CNN architecture with code according to the scratch.  Train the CNN model with dataset.  Test the CNN model with dataset. | Reflect the model. |
| 5: Evaluate the model with testing and validation accuracy, testing loss and validation loss. | Obtain the accuracy of the proposed model. |  |
| 6: Design and implement a Graphical User Interface (GUI) of the model. |  |  |
| 7: Explain the work to related audience. |  | Complete the report.  Conduct the presentation. |

### Schedule

The following figure 10 shows the Gantt chart of this project.

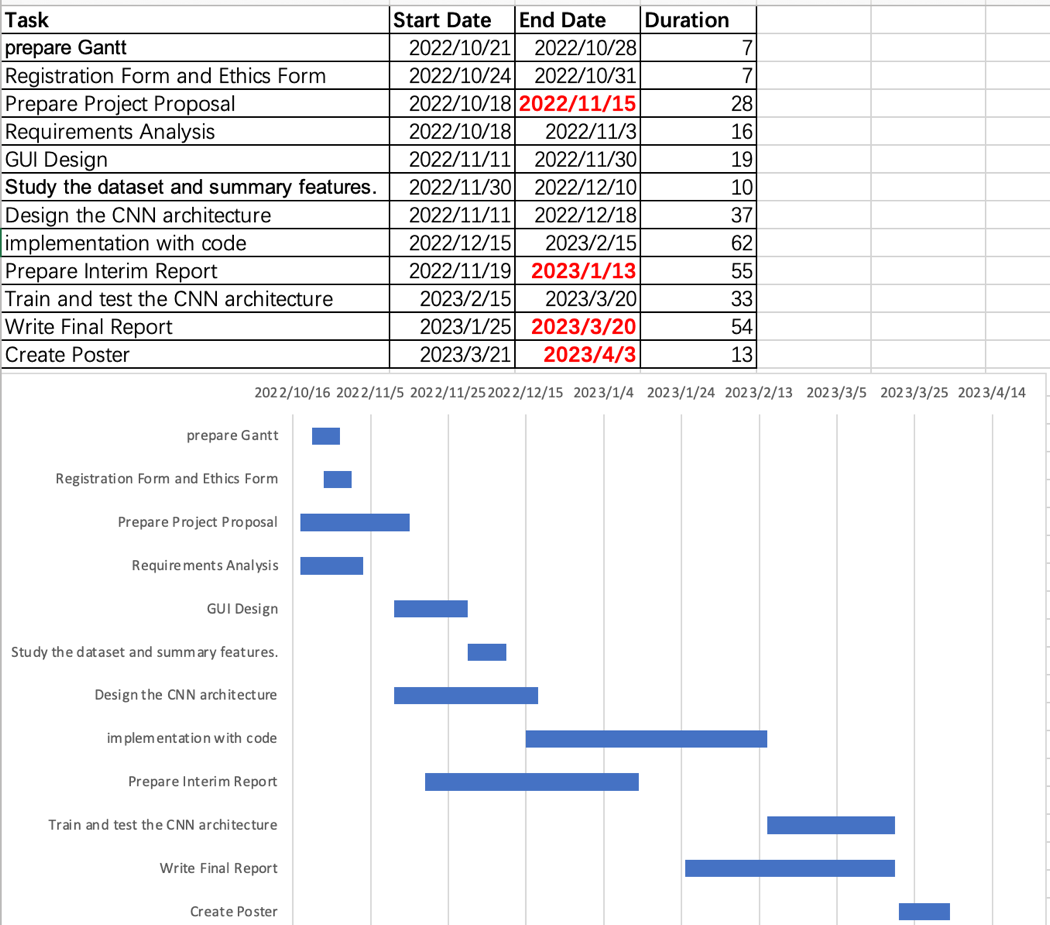


Figure 10. Gantt Chart

### Project Data Management

All the related documentation would be upload to this URL <https://github.com/Haleywuyile/Traffic-Sign-Recognition> on GitHub.

### Project Deliverables

The following are the deliverables:

1. Weekly meeting logs includes;
2. Progress report
3. Next steps and supervisor comments
4. Requirements gathering
5. Deep learning workflow model
6. Testing documentation
7. Reports includes;
8. Proposal
9. Interim report
10. Final report
11. Ethics forms
12. Useful links of literature.
13. Source code

## **Risk Analysis**

Firstly, the risk that this project faces now by current progress is technical, which is one sudden drop of validation accuracy in around the epoch of 20. Secondly, the technical risk has been solved which is about the hardware. The model used to train on a MAC book computer. However, the training process is extremely slow since the GUP memory on this computer is small. So the change the hardware is necessary. The strategy towards the risk is renting GPU server, which helped to speed up the process indeed. In addition, since this hardware risk, there is change of project plan. The process of training model had to extend since learning how to use a GPU server takes time. Lastly, the future risk of the project might be improving the accuracy. From the above figure 2, the training and validation accuracy of the model is not stable at last. So there is need to refine the model. The following tables would help to quantify the above risks.

A risk is equipped with the possibility of occurring and the impact of error. The possibility and impact are ranked from 1 to 5, which is unlikely to likely. Risk exposure is equal to possibility multiply impact. Therefore, risk is going to scale in this following matrix. The following table shows the risk severity matrix.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | Possibility | | | | |
| 1 | 2 | 3 | 4 | 5 |
| Impact | 1 | 1 | 2 | 3 | 4 | 5 |
| 2 | 2 | 4 | 6 | 8 | 10 |
| 3 | 3 | 6 | 9 | 12 | 15 |
| 4 | 4 | 8 | 12 | 16 | 20 |
| 5 | 5 | 10 | 15 | 20 | 25 |

|  |  |  |
| --- | --- | --- |
| KEY | | Description |
|  | Acceptable | These risks would have small impact on the whole project, they are acceptable to happen. |
|  | Tolerable | These risks may interrupt the developing process, but still under control and it can be fixed. |
|  | High | These risks can bring the process to the start, it is serious mistakes and have huge impact. These are the risks that should always avoid to happen. |

Table 1. Risk Severity Matrix

The following table 2 show the potential risks could encounter in this project throughout the development process.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Risk Type | Risk ID | Potential Risks | possibility | Impact | Risks | mitigation |
| Missed Deadline | R1.1 | Illness | 1 | 2 | 2 | Finish the work to be done at least 3 days earlier |
| R1.2 | Forget to submit | 1 | 2 | 2 | Set a clock in case forget |
| R1.3 | Poor time management | 3 | 2 | 6 | Make Gantt chart early and follow the time schedule exactly |
| Technical | R2.1 | Over ambitious project plan | 3 | 2 | 6 | Research the feasibility of project early and discuss it with supervisor before started |
|  | R2.2 | Cannot run the neural networks | 4 | 3 | 12 | Check the memory of GPU in the computer before run the code, if it is too small, rent a server early. |
|  | R2.3 | The accuracy of model is lower than 80% | 4 | 4 | 16 | If the memory of GPU is no problem, check the parameter such as learning rate, epoch, batch size, or change the activation function. |
| Loss of data | R3.1 | Loss of code | 4 | 4 | 16 | Implement version control solution at start like GitHub. |

Table 2. Risk Table

## **Professional Issues**

### Legal Issue

As for the project focus on Chinese traffic signs images, it is inevitably for the images to include surrounding environment. People with ulterior motives may use these images for illegal trading. The consequences are unpredictable. However, the dataset should fairly, lawfully and transparently use for specified, explicit purposes and in a way that is adequate, relevant and limited to only what is necessary [4]. This means that the Chinese Traffic Signs dataset should only be used to build traffic sign recognition system, and cannot be used for other purposes. Therefore, the traffic signs data need to be well protected.

### Social Issue

The social issues is to improve the understanding of technology, appropriate application, and potential consequences. Although there is neural network to help the drivers to recognize the traffic signs, it is still an auxiliary tool. The accuracy of the system is not 100 percent. Therefore, it is not sensible to rely entirely on the system to make decisions.

### Ethnical Issue

The correctness of the model recognition of traffic lights is the ethnical concern for this project. This is important since the model automatically identify traffic signs for divers, if the identification is wrong, the consequence is serious. Therefore, members should accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment [8]. Therefore, it is the developer’s duty to increase the accuracy of the model constantly, since the result of prediction would affect decision of users and may put them in danger.

### Environmental Issue

As for the environmental issues, if the model was put into business, there would be facilities to hold the computing system, therefore would use large amount of electricity. The government would bind company with law and strategy to solve that is renewable energy.

# **Conclusion**

The Process of developing this project takes time since it requires the basic knowledge of deep learning and Python. The biggest challenge that this project encounter is the GPU of computer is too small to run convolutional neural networks. Therefore, the model accuracy and time cost is not ideal at first. Changing parameter like learning rate and batch size to improve accuracy does not solve the problem. Although the solution of renting a GPU server eventually alleviated the problem, the additional time cost was not predicted. However, background review really played an important role in project development. The former researches showed how to build a convolutional neural network and gave inspiration to this project about lightweight architectural. The core deep learning model this project utilize is MobileNetV2. To better process the dataset, the attached classifiers are added. And image data generator is used to reach data augmentation. More importantly, the model achieves the training accuracy of 88.10% and the validation accuracy of 93.75%. And the training loss of 40.78%, validation loss of 17.05%. Lastly, GUI for the TSR system is built successfully by PyCharm library Tkinter. In conclusion, the project went well and more importantly, a lot of valuable knowledge and experience is learned, which would build the foundation for future research.

# **References**

1. A. Benhamida, A. R. Várkonyi-Kóczy and M. Kozlovszky, "Traffic Signs Recognition in a mobile-based application using TensorFlow and Transfer Learning technics," 2020 IEEE 15th International Conference of System of Systems Engineering (SoSE), Budapest, Hungary, 2020, pp. 000537-000542, doi: 10.1109/SoSE50414.2020.9130519.
2. Alghmgham, D. et al. (2019) ‘Autonomous Traffic Sign (ATSR) Detection and Recognition using Deep CNN’, *Procedia Computer Science,* 163(12), pp.264-274. Available at: <https://www.sciencedirect.com/science/article/pii/S1877050919321477> (Accessed: 6 November 2022).
3. Andrew G, H. et al. (2017) ‘MobileNets: Efficient convolutional neural networks for mobile vision applications’, *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* Available at: <https://arxiv.org/abs/1704.04861v1> (Accessed: 2 January 2023).
4. GOV.UK. (2018) *Data Protection.* Available at: <https://www.gov.uk/data-protection> (Accessed: 6 November 2022).
5. He Z, Nan F, Li X, Lee SJ, Yang Y. (2020) ‘Traffic sign recognition by combining global and local features based on semi-supervised classification’. IET Intel Transport Syst 14(5), pp. 323–330. doi: <https://doi.org/10.1049/iet-its.2019.0409>.
6. Hoanh, N. (2020) ‘Fast Traffic Sign Detection Approach Based on Lightweight Network and Multilayer Proposal Network’. Sensor Physical Interpretation, Signal and Artificial Intelligence Processing, 2020(6), pp.13, doi: <https://doi.org/10.1155/2020/8844348>
7. Lim K, Hong Y, Choi Y, Byun H (2017)Real-time traffic sign recognition based on a general purpose GPU and deep-learning. PLoS One 12(3):e0173317, Doi: <https://doi.org/10.1371/journal.pone.0173317>
8. IEEE. (2020) *IEEE Code of Ethics.* Available at: <https://www.ieee.org/content/dam/ieee-org/ieee/web/org/about/corporate/ieee-code-of-ethics.pdf> (Accessed: 6 November 2022).
9. Shustanoy, A. and Yakimov, P. (2017) ‘CNN Design for Real-Time Traffic Sign Recognition’, *Procedia Engineering,* 201(9), pp.718-725. Available at: <https://www.sciencedirect.com/science/article/pii/S1877705817341231> (Accessed: 6 November 2022).
10. Saltzman, M. (2022) Self-driving cars are a thing of the future. But is that future right around the corner? Available at: <https://www.usatoday.com/story/tech/2022/08/29/self-driving-cars-future-gm-tesla/7896389001/> (Accessed: 22 March 21 2023).
11. Sandler, M. et al. (2018) ‘MobileNetV2: Inverted Residuals and Linear Bottlenecks’, *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 4510-4520.* Available at: <https://arxiv.org/abs/1801.04381> (Accessed: 3 January 2023)
12. Staddon, J. (2016) Death by Stop Sign Why US traffic signs are killing thousands. Available at: <https://www.psychologytoday.com/us/blog/adaptive-behavior/201605/death-stop-sign> (Accessed: 21 March 2023).
13. Sun, Y. and Ge, P. and Liu D. (2020) ‘Traffic Sign Detection and Recognition Based on Convolutional Neural Network’, *2019 Chinese Automation Congress (CAC)*, 2019, pp. 2851-2854. Available at: <https://ieeexplore.ieee.org/document/8997240/references> (Accessed: 6 November 2022).
14. Vennelakanti, A. et al. (2019) ‘Traffic Sign Detection and Recognition using a CNN Ensemble’, *2019 IEEE International Conference on Consumer Electronics (ICCE)*, 2019, pp. 1-4. Available at: <https://ieeexplore.ieee.org/document/8662019/references> (Accessed: 6 November 2022).
15. Wasif Arman, H. et al. (2021) ‘DeepThin: A novel lightweight CNN architecture for traffic sign recognition without GPU requirements’, *Expert Systems with Applications*, 168 (5), pp.114-127. Available at: <https://www.sciencedirect.com/science/article/pii/S0957417420311283> (Accessed: 6 November 2022).
16. Xu S, Niu D, Tao B, Li G (2018) Convolutional neural network based traffic sign recognition system. In International Conference on Systems and Informatics (ICSAI), pp 957-961
17. Yoon S. Kim E.: ‘Temporal classification error compensation of convolutional neural network for traffic sign recognition’, *J. Phys., Conf. Ser.*, 2017, **806**, (1), p. 012007. Doi: <https://doi.org/10.1088/1742-6596/806/1/012007>
18. Z. Wang, J. Wang, Y. Li and S. Wang (2022), "Traffic Sign Recognition With Lightweight Two-Stage Model in Complex Scenes," in IEEE Transactions on Intelligent Transportation Systems, 23(2), pp. 1121-1131, doi: 10.1109/TITS.2020.3020556.
19. Zaklouta, F. Stanciulescu, B. (2012) ‘Real-time traffic-sign recognition using tree classifiers’, *IEEE Trans. Intell. Transp. Syst.*,  **13**, (4), pp. 1507–1514. Doi: <https://doi.org/10.1109/TITS.2012.2225618>.
20. Zhu, Y. and Yan, W.Q. (2022) ‘Traffic sign recognition based on deep learning’, *Multimed Tools Appl,* 81(5), pp. 17779–17791. doi: <https://doi.org/10.1007/s11042-022-12163-0>.
21. Zhang J, Hui L, Lu J, Zhu Y. (2018) Attention-based neural network for traffic sign detection. International Conference on Pattern Recognition. 12(8), pp. 20-24. Doi: <https://doi.org/10.1109/ICPR.2018.8546289>

# **Appendices**