Traffic Sign Recognition Using Convolutional Neural Networks

by

Bita Houshmand, Moeen Bagheri

A dissertation
presented to Ryerson University
in partial fulfillment of
the requirements for the degree of
Master of Science (MSc)
in the Program of
Data Science and Analytics

Toronto, Ontario, Canada, 2020

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A THESIS

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my thesis may be made electronically available to the public

TRAFFIC SIGN RECOGNITION USING
CONVOLUTIONAL NEURAL NETWORKS

Bita Houshmand, Moeen Bagheri

Master of Science (MSc)

Data Science and Analytics

Ryerson University, 2020

Abstract

In this project, we have developed a Convolutional Neural Network (CNN) to recognize and classify German traffic signs. We compared our results with an Artificial Neural Network, as well as Random Forest (RF) and Support Vector Machine (SVM) models, and explored the effects of dropout and image augmentation on the accuracy and generalization of the model. The results of our project showed the power of convolutional neural networks in recognizing patterns within images, as well as, the importance of image augmentation and dropout in preventing the overfitting of the model. The CNN model was able to achieve an outstanding accuracy of 98.14% on new images obtained using Google Maps Street View, which was much better than 72.56%, the highest accuracy achieved by the baseline models. Moreover, we have identified pairs of traffic signs that the models are unable to confidently distinguish from each other.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The design of traffic signs makes them very easy to read for humans. However, for computers, recognizing traffic signs is a challenging pattern recognition problem. In practice a large number of traffic signs must be classified with very high accuracy [1]. Traffic sign recognition is a crucial part of autonomous vehicles. The performance of autonomous driving systems depends highly on their ability to detect and recognize traffic signs. Any mistakes in recognizing traffic signs can lead to a catastrophic outcome. Another application for traffic sign recognition is in driver-assistance systems (ADAS), where the system warns the driver of upcoming traffic signs [2]. A study found that human error was the sole cause in 57% and a contributing factor in 90% of all road accidents [3].

In this paper, we will propose a Convolutional Neural Network model for recognizing German traffic signs. Convolutional Neural Networks are known for their ability to recognize images, as they are able to learn from the spatial relationship of pixels within images. We will then compare our CNN model with three other baseline models, which include Artificial Neural Network, Random Forest, and Support Vector Machines. Additionally, we will explore the effects of dropout and image augmentation on the performance of the CNN model.

Chapter 2

Literature Review

Automatic traffic sign detection and recognition (TSDR) has been widely studied due to its importance in enhancing road safety and its potential in various applications in Driver Assistance Systems (ADAS), Autonomous Driving Systems (ADS) and maintenance of traffic signs.

However the task is still challenging because of different factors such as similarity between signs in the same category, sign regulations may vary in different countries, color fading with time, weather condition can affect the visibility of signs, presence of objects with similar color or shape in a same frame with the traffic sign, as the image is usually taken from a moving car, it often suffers from motion blur, signs may be found disoriented, damaged or occluded, color information is sensitive to different illuminations and highlights [4].

2.1 Traffic Sign Definition

Road signs are placed along the roads to regulate traffic, inform drivers about the conditions and restrictions of front road and assist drivers to find their way. The main types of traffic signs are regulatory, warning, temporary condition and information and direction signs. Traffic signs may have different structures and appearances in different countries which pose a challenge on the way of TSDR systems.

2.2 Benchmarks

Some countries have road signs that are unique to their country. Also, around the world, there are traffic signs that have the same meaning but different appearances. Thus, it is difficult to compare the TSDR methods designed for different countries. The public datasets provide benchmarks for comparison.

German Traffic Sign Benchmark (GTSB), consists of two different datasets including the German Traffic Sign Recognition Benchmark (GTSRB) and the German Traffic Sign Detection Benchmark (GTSDB), which are a large multi-category classification and detection benchmarks. There are 900 images in GTSDB including 600 training and 300 evaluation images. The GTSRB includes more than 50000 images and more than 40 classes [5].

European traffic sign dataset (ETSD), is composed of traffic signs from 6 different countries including Belgium, Croatia, France, Germany, Netherlands, and Sweden. It is composed of more than 80000 images divided in 164 classes that at the same time belong to four main categories following the Vienna Convention of Road Signs [6].

KUL Belgium Traffic Sign Dataset (BTSD), consist of two datasets BelgiumTSC and BelgiumTSD which are comprehensive datasets for classification and detection respectively. BelgiumTSC includes 4591 training and 2534 testing images. BelgiumTSD includes 25634 images, it is divided into 5905 training and 3101 testing instances [7].

Swedish Traffic Signs Dataset (STSD), consist of sequences from highways and cities recorded from more that 350 km of Swedish roads. It includes more than 20000 images, 3488 traffic signs and 20 percent of instances are labeled [8].

Mapping and Assessing the State of Traffic Infrastructure (MASTIF), consists of three datasets TS2009, TS2010, and TS2011. TS2009 includes 6000 cropped images, TS2010 includes 3000 signs in the form of full source images with annotation, and TS2011 includes around 1000 signs in the form of full source images with annotation. All images have resolution of 720×576 pixels [9].

RUG Traffic Sign Image dataset contains 48 images (360 $\times$ 270 pixels) of three traffic sign types pedestrian crossing, intersection and compulsory for bikes which captured in Netherland [10].

LISA Traffic Sign dataset is a set of videos and annotated frames containing US traffic signs. It is released in two stages, one with only the pictures and one with both pictures and videos. There are 7855 images with 6610 signs. This dataset can be used to verify detection or tracking methods [11].

Tsinghua-Tencent 100k (TT100K) is a large traffic dataset consists of 100000 images including 30000 traffic-sign instances. The dataset covers large variations of images in different illuminance and weather conditions. Each traffic sign in the dataset is annotated with a class label, its bounding box and pixel mask [12].

Stereopolis database is made of 847 images 9601080 containing 251 road signs. The images were acquired in Paris, France, with the IGN moving van, grabbing a picture every 5 meter [13].

2.3 Traffic Sign Detection and Recognition Approaches

The procedure of TSDR system can be divided into two key steps, TSD and TSR. TSD refers to a process of detecting and locating traffic signs in the image or video frame. At first, background objects and unimportant information in the image are eliminated. The resulted image from this step is a binary image containing the traffic sign and any other objects similar to the color of the traffic sign. The noise and unrelated objects in the binary image are cleaned by the object selector process. Then, in TSR step, the detected traffic signs are utilized as inputs to a classification module which classifies the detected traffic signs into a particular class.

2.3.1 Traffic sign detection

The first step in any TSDR system is identifying the location of potential traffic sign from an input image which is called detection module. In other words a ROI which points to a traffic sign is identified and localized [14]. Traffic signs usually have a strict color scheme (red, blue, and white) and specific shapes (round, square, and triangular). These inherent characteristics distinguish them from other outdoor objects making them suitable to be processed by a computer vision system automatically, thus, allow the TSDR system to distinguish traffic signs from the background scene [15]. Traffic sign detection methods can be classified into color-based, shape-based and hybrid (color-shape-based) detection.

Color based detection

Color-based techniques use color as the main feature to identify the Region of Interest (ROI) which in TSDR case is the region containing the traffic sign in the image or video frame to make the image ready for classification stage. The process eliminates the unnecessary objects and hence it reduces the search area. Color-based approaches usually transform images into a specific color space where the signs are more distinct and then the ROI is extracted based on color segmentation. Different color spaces have been used including red-green-blue (RGB), hue-saturation-value (HSV) and hue-saturation-intensity (HSI) [16]. There are various approaches which employ color-based detection techniques, the most common ones are color thresholding segmentation, color distance transform, detection in HIS/HSV space and recently YCbCr space.

Color thresholding segmentation is based on the assumption that adjacent pixels whose values are in a specific range belong to the same class. The reason of thresholding is to segment pixels of an image into object pixels or background pixels. This can be done by putting definite threshold for one of the categories, a pixel falls on object pixel category if its color is close enough to a reference level, if not it will be a background pixel [14, 17].

In color distance transform, color distance is similar to the Euclidean distance between two points and it is calculated by taking the difference between the two colors. As the color distance decreases the similarity increases [18]. Detection in HSI space separates achromatic and chromatic components, the detection can be performed even in the challenging lighting conditions. The image in RGB space is converted to HSI color space and the segmentation is done based on the Hue and Saturation. in this approach usually images are transformed to binary by setting the pixels of interest to 255 and others to 0. Compared with RGB and HSI color spaces, the HSV color space has a faster detection speed, less affected by illumination, and has a preferable segmentation advantage [19]. YCbCr color space has been considered in recent approaches. Different from other common color spaces, it represents color as brightness and two-color difference signals. In this method red color which is a common color in traffic signs is obtained from the YCbCr color space by applying the dynamic thresholding on the Cr component value. Then the objects with smaller area than some predefined value are eliminated as they are less likely to be the traffic signs [20].

Although color-based approaches are computationally efficient and fast which almost meet the real-time requirements of TSDR systems, these methods are largely impacted by illumination, color fading, cluttered environment and different weather conditions which may result in lower detection rate [21, 22].

Shape-based detection

The prohibitory, danger or mandatory traffic signs often have standard shapes, such as circle, triangle and rectangle, hence shape-based detection approaches are a common way to use geometric information to locate traffic signs which reduces the search area for a sign from a whole image to small number of pixels [15, 23].

The most common shape-based approach is Hough Transformation which usually isolates features of a particular shape such as circle, triangle and rectangle within a given

image using shape's boundary points by a voting process. In order to apply hough transformation, the image is converted to grayscale as edge detector which is part of detecting the contours demands for grayscale image [24]. The main advantage of the Hough transformation technique is its tolerance of gaps in feature boundary descriptions and is relatively unaffected by image noise. However, its main disadvantage is the dependency on input data [4].

Distance transform matching is another approach, in this technique, edges in the original image are found and then a distance transform (DT) image is built. A DT image is an image in which each pixel represents the distance to the nearest edge. Its advantage over other techniques is that it is capable of detecting objects of arbitrary shapes when dealing with non-rigid objects. It also uses a template hierarchy to capture the different shapes of object [25].

Template matching is a simple methodology for shape-based object detection within images. It identifies part of an image that matches a predefined template. It can be easily performed on greyscale or edge images. Usually a sub-area of size N $\times$ N is selected, and pixel values are normalised by the maximum and minimum values of the selected sub-area. The size is also normalised depending on the template to be matched, and the closeness with the template is calculated. The main advantage of this technique is its effectiveness with respect to the detection of low-textured objects or objects characterized mainly by shape [25, 26].

Another approach is Haar-like features recognizes the target objects based on the Haar wavelet. The main advantage is its calculating speed, where any size of images can be calculated in a constant time. However, its weakness is the requirement of a large number of training images and high false positive rates [15, 23].

As previously discussed, shape-based techniques reduce the search area from a whole image to small number of pixels. However, damaged, partially obscured, faded and blurred traffic signs may cause difficulties in detecting traffic signs accurately.

Hybrid detection

In order to overcome the drawbacks of aforementioned methods and improve the traffic sign detection accuracy, hybrid method takes both color and shape information into account. Hybrid methods usually adopt two step strategy. First color segmentation is done in any color space to narrow the search space and then shape-based detection is performed only to the segmented areas [27].

2.3.2 Traffic sign recognition

Traffic sign recognition step in TSDR is mainly applied on the detected signs from previous step to analyze and classify the traffic signs and accurately obtain their actual meaning [19].

Template matching uses predefined templates to search the whole image pixel by pixel or to perform on a small segment. First step is to slide already learned template image over a detected traffic sign. Sliding means moving a patch pixel by pixel from left to right and top to bottom. A metric is calculated at each location to see how similar the patch is to that particular area of detected image and the result is saved in a similarity matrix for each location. Each element in the matrix shows the match metric. Then normalized cross-correlation can be used to find the match between the detected traffic signs and the template. It has the advantages of being fast, straightforward and accurate. However, template matching method requires a different template for each scale and orientation. It is also sensitive to noise and occlusions [28, 29].

Another classification method is random forest. It is an ensemble of simple trees, where each tree contributes with a single vote for the assignment of the most frequent class to the input data. In standard trees, each node is split using the best split among all variables. In a Random Forest, each node is split using the best among a subset of predictors randomly chosen at that node. Usually features of the resulted images from

detection step are extracted and the feature vectors are fed into random forest classifier. The approach performs well in comparison to many other classifiers and is robust against over-fitting. However, the main limitation of a random forest is that a large number of trees can make the algorithm slow and ineffective for real-time predictions [30].

Genetic algorithm is another classification method. It improves the recognition accuracy of traffic signs under adverse conditions such as if the sign has some shape loss or illumination problem. GA can be incorporated with template matching. First similarity between the detected image and a template can be calculated by cross-correlation. Then the reciprocal of this similarity is used as the fitness of GA and the generation evolving is repeated until the fitness got smaller than a given value. The disadvantage of the genetic algorithm is non-deterministic work time and there is no guarantee in finding of the best solution [25, 31].

One of the most common methods for traffic sign recognition is convolutional neural network (CNN). Simultaneous detection and classification of traffic signs can be achieved using CNN method which results in improved performance, boosted training and testing speeds. The only drawbacks of convolutional neural network is that it requires a large number of training samples for real world applications [25].

Support vector machine (SVM) is another classification method used in traffic sign classification. The basic concept of SVM is to transform the input vectors to a higher dimensional space by a nonlinear transform, and then a hyperplane separates data. SVM is designed to solve binary classification problems. Solving multi-classes problems is accomplished through combinations of binary classification problems. SVM classifier is robust, highly accurate and extremely fast which is a good choice for large amounts of training data [21, 30].

Chapter 3

Dataset

The training, validation, and test datasets were obtained from the following kaggle competition:

https://www.kaggle.com/ valentynsichkar/traffic-signs-preprocessed.

The data files used for this project are named $\mathrm{train.pickle}$, $\mathrm{validation.pickle}$, and $\mathrm{test.pickle}$. All images are RGB and have a size of 32×32. There are 34799, 4410, and 12630 samples in the training, validation, and test sets, respectively. Furthermore, the dataset includes 43 classes of traffic signs. Figure B.2 shows the name of all classes and Figure B.3 shows a single image for each class from the training set.

Moreover, for each traffic sign, 5 RGB images with a square aspect ratio and varying sizes were obtained using Google Maps Street View to be used as a final test for the models. However, due to the rarity of some traffic signs, some images were obtained from other sources by Google search, and also for a few traffic sign classes, multiple images were captured of the same traffic sign from different angles in order to create multiple samples. Additionally, images for some classes were obtained by flipping the images of another similar class. For example, some of the images for the $\mathrm{Turn\ right\ ahead}$ traffic sign were obtained by flipping the images of $\mathrm{Turn\ left\ ahead}$ traffic sign. For simplicity, we will refer to these images as Google Maps images. The Google Maps images can be found in the following link:

https://drive.google.com/drive/folders/1-BDp5ljNP4okonNZe5Nmq_uon2YmeHQ4.

Chapter 4

Methodology

All models were developed using python, as well as the packages Keras and Sklearn. Additionally, all visualization were obtained using the Matplotlib and Seaborn libraries. Following data preprocessing, we trained each model and obtained their accuracies on the test and Google Maps images. Afterwards, we compared the performance of the CNN model to the performance of the three baseline models. The rest of this chapter introduces the other tools used in the project, and explains the preprocessing done on the data, as well as the structure of the proposed CNN and the structure of the three baseline methods: (1) ANN, (2) Random Forest, and (3) SVM.

4.1 Data Preprocessing

All training, validation, and test images were normalized by dividing all values by 255, which is the maximum possible value of a pixel. All target values were also OneHotEncoded by first fitting the encoder on the training target set, and then using the fitted encoder to transform all target values.

Moreover, in order to avoid overfitting of the CNN and ANN models and help the models generalize better, we performed image augmentation on the training set using the ImageDataGenerator class from the keras.preprocessing.image package. Image augmentation allows us to create various modified versions of the images, which can improve the performance of the model, by making it insensitive to, for example, the angle or size of the

objects. This in turn helps the model generalize better and recognize new images.

4.2 Convolutional Neural Network

A summary of the structure of the CNN is shown in Figure B.4. The proposed CNN includes four convolution layers, as well as two fully-connected layers and an output layer. All convolution and fully-connected layers used a relu activation function. Furthermore, all convolution layers were followed by a batch-normalization layer after activation in order to enhance the learning speed of the model. We also applied dropout with a rate of 0.1 for convolution layers and 0.2 for fully-connected layers. Moreover, all convolution layers were followed by a max pooling layer after dropout. Finally, the model was compiled using a cross-entropy loss, accuracy as its evaluation metric, and Adam as its optimizer with a learning rate of 0.0001.

Furthermore, Figure B.1 shows the distribution of the number of samples per class. It can be seen that some classes have a lot less samples than other classes, which causes an imbalance in the dataset and can affect the performance of the model by making it biased towards the majority classes. To overcome this issue and balance the dataset, we used the $\mathrm{sklearn.utils.class\_weight.compute\_class\_weight}$ function to compute a weight for each class. These weights were passed to the fit function, which makes the model pay more attention to the under-represented classes. In addition to this, we also used image augmentation to help the model generalize better. To generate augmented images, we used a batch size of 64 and explored a width and height shift of 0.1, a zoom range of 0.2, a shear range of 0.1, and a rotation range of 30 degrees.

The model contained 6,334,811 trainable parameters and was fit to the generator obtained from image augmentation for 20 epochs and 1000 steps per epoch. In order to find the model with minimum loss on the validation set, we used the $\mathrm{ModelCheckpoint}$ class from keras to save checkpoints of the model when the model reaches a new minimum

loss on the validation set.

Following training, we loaded the model with minimum loss on the validation set and obtained its accuracy on the original training set. We also tested the trained model on the test and Google maps images. Finally, we examined the results and identified the traffic signs that are unrecognizable by the model, as well as, pairs of traffic signs that are mistaken or indistinguishable by the model.

In the rest of this section, we will discuss the hyperparameters used in the convolution, pooling, fully-connected, and output layers.

Convolution Layers

Our model used a total of four convolution layers. The first convolution layer used 128 filters with a filter size of (5, 5). The purpose of this layer is to capture the general patterns in the image, hence, we chose a larger filter size and also strides of (2, 2) for this layer. We also used same as padding for this layer in order to preserve information at the borders of the images. The second convolution layer also used 128 filters. However, we chose a smaller filter size of (3, 3), valid as padding, and strides of (1, 1) for this layer. The third and fourth convolution layers used 256 filters, with a filter size of (3, 3) and strides of (1, 1).

Pooling Layers

All convolution layers were followed by a pooling layer in order to reduce the complexity and the number of parameters. All pooling layers used the max as the pooling method. We decided to use max pooling over average pooling in order to capture the most important features at every step. Moreover, all pooling layers used a pool size of (2, 2) and strides of (1, 1) in order to minimize information loss after pooling.

Fully-connected and Output Layers

Following convolution, the output was flattened for the fully-connected layers. The first fully-connected layer contained 512 nodes and the second fully-connected layer contained 1028 nodes. Finally, the output layer contained 43 nodes, which is equal to the number of classes, and used a $\mathrm{softmax}$ activation function.

4.3 Artificial Neural Network

An ANN was used as the first baseline for comparison with CNN. The ANN was fully-connected and used 5 hidden layers. The first two hidden layers contained 128 nodes, the third hidden layer contained 256 nodes, and the last two hidden layers contained 512 nodes. Additionally, the input layer contained 3072 nodes, which is the length of the vectorized images, and the output layer contained 43 nodes, which is the number of classes, with a $\mathrm{softmax}$ activation function. We decided to use a $\tanh$ activation function for all hidden layers, since we found that the model performs better with a $\tanh$ activation function compared to a relu or $\mathrm{sigmoid}$. Additionally, we used a batch-normalization layer after each hidden layer in order to speed up the training process. Moreover, following batch-normalization, we added dropout with a rate of 0.2 to the second, third, and fourth hidden layers to prevent overfitting. Finally, the model was compiled using cross-entropy as the loss function, accuracy as the evaluation metric, and Adam as its optimizer with a learning rate of 0.0001. Figure B.8 shows a summary of the ANN structure, which contained 862,251 trainable parameters.

Similar to the CNN model, in order to deal with the data imbalance, we computed weights for each class based on their abundance in the training set and passed the weights to the fit function to prevent the model from over-fitting on the over-represented classes. Additionally, we also performed image augmentation to avoid overfitting, however, for the ANN model, we explored a smaller range compared to the CNN model, since higher ranges

significantly hindered the learning ability of the model. We used a batch size of 64 and explored a zoom range of 0.1, a shear range of 0.1, and a rotation range of 10 degrees.

Subsequently, we fit the model to the image augmentation generator for 20 and 1000 steps per epoch. Similar to the CNN model, we used the $\mathrm{ModelCheckpoint}$ class to save the model with the minimum loss on the validation set. We then obtained the best trained model with minimum validation loss and calculated its accuracy on the original training images, as well as the test and Google Maps images. Finally, we examined the results and identified the traffic signs that are unrecognizable or indistinguishable to the model.

4.4 Random Forest

As a second baseline for comparison with CNN, random forest is selected as an ensemble method with the goal of combining the predictions of several base estimators to improve generalizability and robustness over a single estimator.

In order to find the optimized number of trees in the forest, 100, 200, 300, 400 and 500 trees are tested using grid search. Based on grid search result, we set the number of RF estimators to 500. In learning step, first we fitted the model with all features using vectorized version of the input images. The input images have the dimension of 32x32 pixels and 3 channels thus the number of features used is 3072 in the vectorized version.

To achieve better accuracy and generalization on google map images, we applied different feature extraction methods before fitting the model including PCA, scaling the input image and then applying PCA and extracting Histogram of Oriented Gradients (HOG) of input image.

One of the techniques that is used for extracting the most important features is PCA. Generally, we want the explained variance to be between 95% to 99% thus we set the number of components of PCA to 0.95 [33]. To improve the accuracy further, we also scaled the input images to the range between 0 and 1 and then applied PCA with the same

setting with previous one is applied.

As local object appearance and shape can be characterized using the distribution of local intensity gradients, histogram of gradients or HOG feature descriptors are selected as another approach for feature extraction to representing different shapes of the traffic signs. The most important parameters for the HOG descriptor are the orientations, pixels per cell, and the cells per block. These three hyperparameters effectively control the dimensionality of the resulting feature vector. To tune these hyperparameters we tested different settings and measured the accuracy of RF considering each setting.

In the first setting the number of orientation bin is set to 9, size of a cell in pixel is set to (2, 2), number of cells in each block is also set to (2,2), block-normalization is set to L2-Hys and multichannel is set to true as we are feeding color images to HOG extractor. In the second setting, the number of orientation bin is set to 9, size of a cell in pixel is set to (4, 4), number of cells in each block is set to (2,2) and the rest of parameters are set fixed as previous setting. In the third setting, the number of orientation bin is set to 9, size of a cell in pixel is set to (8, 8), number of cells in each block is set to (2,2) and the rest of parameters are set fixed as previous settings. In the fourth setting, we set the number of orientations to 5, size of a cell in pixel to (4, 4), number of cells in each block to (2,2) and the rest of parameters are set fixed as previous settings.

Based on resulted accuracies of different implemented random forest models, RF fitted with the extracted Hog features with its parameters set to setting 1 is selected as baseline to comparison with CNN. We discuss the results in detail in section 5.3.

4.5 Support Vector Machine

Support Vector Machine is selected as a third baseline for comparison with CNN. Same as previous baseline first, HOG features are extracted from the images as it widely used for object detection in computer vision literature. As we discussed previously hog descrip-

tors represent the occurrence of gradient orientations in the image. Gradient histograms measure the orientations and strengths of image gradients within an image region. There are two benefits in using HOG features. First, traffic signs have distinctive shapes, when converting the image to grayscale and use one channel, different layers may have strong edges and the gradient can efficiently capture these features. Second, using HOG can help to achieve scale invariance [32].

After extraction of hog features from images in train, validation, test and google map sets. The SVM classifier is trained on hog features of training set to classify the traffic signs. SVM is a binary classifier, which classifies data between two classes. But traffic sign images cannot be classified using two classes. Hence to train the SVM classifier, the pair wise classification of training images is used. Here, the classifier is trained for each possible pair of classes. For 43 different sign classes, this results in 903 binary classifiers. We performed grid search to tune the hyperparameters and find the best parameters to train the SVM model. Different values for kernel, Kernel coefficient, degree and penalty parameter are considered.

Based on grid search result, the SVM classifier hyperparameters are set to kernel Radial Basis Function, penalty parameter 10, degree 4 and gamma. In prediction stage, a new unseen traffic sign x is classified by applying each of the binary classifiers and count how many times traffic sign x was assigned to that class label. Class label with highest count is then considered as a target label for the instance. We discuss the results in detail in section 5.4.

Chapter 5

Results

Table A.1 shows the accuracy of all models on the training, validation, test, and Google Maps images. It can be seen that the proposed CNN model performs much better than the three baseline methods, as expected, and was able to obtain a much higher accuracy on all images. Particularly, the CNN model significantly outperformed the baseline models on the Google Maps images and was able to generalize much better to new images. Moreover, the accuracy of ANN was comparable to the RF and SVM models, however, the ANN model had a slightly higher accuracy on the Google Maps images, compared to the RF and SVM models, while having a lower accuracy on the validation and test images.

In the rest of this section, we will discuss the results of each model in detail and we will point out the traffic signs that the models had a difficult time recognizing, as well as pairs of traffic signs that were indistinguishable to the models.

5.1 Convolutional Neural Network

The CNN model was fit to the image augmentation generator for 20 epochs. The accuracy and loss of the model at each epoch on the augmented training data and the validation set is shown in figure B.5. We can see that the accuracy curve plateaus around iteration 10. Moreover, the best model found had the minimum loss of 0.129562 on the validation set, which occurred at epoch 17. Furthermore, the best CNN model found had an accuracy of 99.95% on the original training images, and an accuracy of 97.73% on the validation,

95.80% on the test, and 98.14% on the Google Maps images. These accuracies highlight the ability of convolutional neural networks in recognizing patterns within images. Moreover, the high accuracies obtained by the CNN model on the test and Google Maps images show that the model was able to generalize well and correctly recognize new images.

Figure B.6 shows the precision and recall of the CNN model on the test and Google Maps images. We can see that in the test set, the model had the lowest precision and recall for Double curve, Pedestrians traffic signs. Moreover, the model had a low recall for Dangerous curve to the left and Beware of ice/snow traffic signs, and a low precision for the Go straight or right traffic sign. We can further investigate the model's inaccuracies by examining the heatmap of the wrong predictions made by the model in Figure B.7. We can see that in the test set, the model incorrectly classified 33 instances of the Speed limit (50km/h) traffic sign as Speed limit (30km/h) traffic sign, 30 instances of General caution traffic sign as Double curve traffic sign, and 29 instances of Priority road traffic sign as Go straight or right traffic sign. These results show that the model is unable to confidently distinguish between these pairs of traffic signs. In general, the model seems to have a bit of difficulty distinguishing between some pairs of speed limit traffic signs. In addition to the Speed limit (50km/h) and Speed limit (30km/h) traffic signs mentioned before, the model seems to mistaken Speed limit (70km/h) for Speed limit (30km/h) traffic sign, and Speed limit (80km/h) for Speed limit (50km/h) traffic sign. Similarly, we can further examine the heatmap to identify other traffic sign pairs that are indistinguishable to the model.

5.2 Artificial Neural Network

The Accuracy and loss of the ANN after each epoch on the augmented training images is shown in Figure B.9. It can be seen that the accuracies plateaus around epoch 13, and the best model with minimum loss of 0.436524 on the validation set was found at epoch 15. The best model was able to achieve an accuracy of 96.78% on the original training

set, and an accuracy of 88.41% on the validation, 85.78% on the test, and 72.56% on the Google Maps images. The ANN model was unable to generalize as well as the CNN model, which further highlights the power of convolutional neural networks in recognizing and classifying images.

From the precision and recall of the CNN model on the test and Google Maps images, shown in figure B.10, we can see that the model is unable to confidently recognize many traffic signs, specially the Speed limit (20km/h), Dangerous curve to the right, Road narrows on the right, Pedestrians, and Beware of ice/snow traffic signs. Moreover, by examining the heatmap of wrong predictions in the test set, shown in Figure B.11a, we can see that the model is unable to distinguish between several pairs of traffic signs. In particular, similar to the CNN model, the ANN model is unable to distinguish between different speed limit traffic signs. The model misclassified 88 samples of Speed limit (100km/h) as Speed limit (120km/h), 70 samples of Speed limit (60km/h) as as Speed limit (80km/h), and 54 samples of Speed limit (50km/h) as Speed limit (80km/h). However, this issue is much more predominant for ANN, not only for speed limit traffic signs, but also for other traffic sign pairs.

5.3 Random Forest

As table A.2 shows Random Forest model fitted with all features of input image achieved 100% accuracy on training, 74% accuracy on validation, 76% accuracy on test and 54% accuracy on google map set. The validation accuracy of the model for traffic sign recognition task is not acceptable as it is critical to classify all signs almost perfectly. And also, the model does not generalize well on random images captured from google map.

RF fitted with the extracted features using PCA achieved 100% accuracy on training, 61% accuracy on validation, 64% accuracy on test and 35% accuracy on google map images. The result shows that the model does not perform well on validation and test sets

and also poorly generalize on unseen real images. As we discussed in previous sections to improve the accuracy of RF, before fitting the model, we scaled the data and then applied PCA. The model achieved 100% accuracy on training, 62% accuracy on validation, 64% accuracy on test and 34% accuracy on google map set which performs worse than previous model.

As mentioned earlier to extract important features, we represented traffic signs with histogram of gradients and then fitted the model using the extracted features. The RF fitted with hog features with its parameters set to setting 1, achieved 100% accuracy on training, 93% accuracy on validation, 90% accuracy on test and 71% accuracy on google map set. The model performs relatively well on all 3 sets. The RF fitted with hog features with its parameters set to setting 2, achieved 100% accuracy on training, 91% accuracy on validation, 88% accuracy on test and 70% accuracy on google map set. As we can see accuracies on all datasets declined. The RF fitted with hog features with its parameters set to setting 3, achieved 100% accuracy on training, 79% accuracy on validation, 77% accuracy on test and 61% accuracy on google map set which does not generalize well. The RF fitted with hog features with its parameters set to setting 4, achieved 100% accuracy on training, 92% accuracy on validation, 88% accuracy on test and 66% accuracy on google map set which shows that the number of orientation of the image is affected the generalization of the model and caused a decline in a google map accuracy in comparison with setting 1 with the same parameter except the orientation.

Based on the resulted accuracies of different implemented random forest models, RF fitted with the extracted Hog features with its parameters set to setting 1 is selected as baseline to comparison with CNN as it achieved higher score on all 3 datasets and generalizes better on unseen random data captured from google map.

However, from figures B.12 and B.13 we can see that the model has difficulty to correctly classify couple of classes on google map images, mainly Speed limit (50km/h), Speed limit (80km/h), Speed limit (120km/h), Right-of-way at the next intersection and

Road-work signs. It is clear that the model usually misclassify the signs that are visualy very similar to each other for example it misclassify Speed limit (50km/h) with Speed limit (80km/h) or Speed limit (120km/h) with Speed limit (100km/h).

5.4 Support Vector Machine

The accuracies achieved by SVM classifier, fitted with hog feature descriptors are 100% accuracy on training set, 91% accuracy on test set, 90% accuracy on validation set and 70% accuracy on google map images. Although the SVM model is completely accurate on training set and perform relatively well on validation and test sets but the challenge is it does not perform well on real-world random and unseen data which means it does not generelize well enough in the task of traffic sign recognition which requires high accuracy in classifying all signs. From figures B.14 we can see that the model has difficulty to correctly classify couple of classes on test and google map images including Speed limit (30km/h), Speed limit (50km/h), Right-of-way at the next intersection, Priority-road and Road-work signs.

Chapter 6

Discussion

An existing approach for classifying German traffic signs used two convolution layers with 12 and 25 filters, respectively, and three fully-connected layers including the output layer with 300, 100, and 43 nodes, respectively [34]. Moreover, they did not perform image augmentation and only included dropout after the flattening layer with a rate of 0.2. Their model was able to achieve an accuracy of 93.37% on the test set and 63.6% on new images, which is much lower than the accuracies achieved by our proposed model, suggesting that our model was able to generalize better to new images. In another study, they used three convolution layers with 70, 110, and 180 filters, as well as two fully-connected layers, including the output layer, with 200 and 43 nodes [35]. They also performed image augmentation but did not include dropout. Their model was able to achieve an accuracy of 99.65% on the test set. Even though they were able to achieve a higher accuracy on the test set compared to our model, they did not test their model on a new image source, such as our Google Maps images, and it is uncertain how their model will perform on these images. Note that our model could have achieved a higher accuracy on the validation and test images using different hyperparameter values, at the cost of a lower accuracy on the Google Maps images.

Particularly, the dropout layers were crucial for generalization of the CNN and ANN models and significantly increased the accuracy of the models on the Google Maps images, which shows the ability of dropout in preventing the overfitting of the models. The

performance of the CNN model on the Google Maps images was highly sensitive to the dropout rates. For example, increasing the dropout rate of the fully-connected layers to 0.5 decreased the accuracy of the model to about 70% on the Google Maps images. On the other hand, removing dropout layers caused the accuracy on the Google Maps images to drop to 95.81%. Additionally, image augmentation was a significant factor in preventing the model from overfitting as well. Even though the CNN model was able to achieve almost 100% accuracy on the training set without image augmentation, the accuracy of the model was lower on the test and Google Maps images. Without image augmentation, the CNN model was only able to achieve an accuracy of 86.98% on the Google Maps images, which is much lower than 98.14%, which was the accuracy achieved with image augmentation. This shows the importance of image augmentation in preventing overfitting and making the model insensitive to factors such as the angle or distance of an object. Besides this, the batch-normalization layers significantly increased the learning speed of the models. This is due to the fact that batch normalization reduces the number of steps needed for the model to reach its optimal state, which in turn speeds up training.

On a separate note, from the heatmap in figure B.7, which shows the number of wrong predictions for each pair of target and predicted values, we can see that the CNN model has a hard time distinguishing between different speed limit traffic signs. Specifically, the model is unable to distinguish the numbers 50 and 70 from the number 30, which is potentially due to the high resemblance between these numbers. A possible method to improve the accuracy of the model on speed limit signs could be to train the model on close-up images of speed limit signs, in which only the number on the sign is visible. This will help the model distinguish between different numbers. Note that this will be acceptable for the current dataset, since numbers only convey speed in the dataset. However, if there were other traffic signs with numbers in the dataset that conveyed a different meaning (such as distance), this approach will not be suitable. In general, this approach can also be used for other traffic signs, provided that the traffic sign does not share the same patterns with

other traffic sign classes.

Furthermore, we can clearly acknowledge the importance of convolutional neural networks and their ability to recognize patterns within images by comparing the performance of the CNN model with the baseline models. The CNN model significantly outperforms the ANN model, which highlights the importance of spatial information in recognizing traffic signs. Since the images are vectorized for the ANN model before training, this spatial information is lost.

Moreover, In comparison with ANN with average accuracy of 85.88%, Random Forest achieved average accuracy of 88.23% which is higher than ANN by 2.35%. It is clear that considering average accuracy Random Forest as an ensemble method outperformed ANN. However, A.1 shows that ANN generalizes better on unseen images captured from Google Maps. Also, as we can see from B.11 and B.13 RF has difficulty classify signs that include very similar numbers like $\mathrm{Speed\ limit\ (80km/h)}$ and $\mathrm{Speed\ limit\ (60km/h)}$ but ANN mostly classify these signs correctly which is an indication that as we discussed earlier an ensemble of ANNs can be a method for further improvement.

In addition, the third baseline, SVM model acheived 87.62% on average accuracy which is very close to RF model as their difference is less than 1%. However the SVM model suffers from the same generalization problem as RF.

Chapter 7

Conclusion

We explored the performance of four different models on recognizing German traffic signs. The CNN model significantly outperformed all baseline models, which shows the power of convolutional neural networks in classifying images. Additionally, the CNN model performed and generalized much better than the ANN model, which indicates the ability of convolution layers in recognizing patterns in images. Moreover, we highlighted the importance of data augmentation and dropout to prevent overfitting of the model, and the importance of batch-normalization for speeding up the training process.

On the other hand, it is uncertain how the model will perform on traffic signs obtained from other countries. It is likely that the model will perform poorly in recognizing foreign traffic signs, since there can be many variations in the shape, color, and patterns of traffic signs in different countries. Future studies can focus on creating a model that can generalize to other countries. However, this will require extensive data collection, since the model will need to be trained on traffic sign images from many countries.

In conclusion, we learned the importance and ability of convolutional neural networks in recognizing images. Moreover, we learned about the potential of batch normalization in speeding up the training process, and the potential of dropout and image augmentation in preventing a model from overfitting.

Chapter 8

Contributions

Moeen Bagheri

- Architecture design and optimization of all models.
- Implementation of CNN and ANN models.
- Writing the methodology and results of CNN and ANN models.
- Comparison of the CNN model with existing work.
- Writing the abstract, introduction, dataset, part of discussion, and conclusion of the report.
- Collection of Google Maps images.

Bita Houshmand

- Architecture design and optimization of all models.
- Implementation of RF and SVM models.
- Writing the methodology and results of RF and SVM models.
- Writing the literature review and part of discussion of the report.
- Collection of Google Maps images.

Appendices

Appendix A

Tables

Table A.1: The accuracy of the models on the training, validation, test, and Google Maps images.

Model	Training	Validation	Test	Google Maps
CNN	99.95%	97.73%	95.80%	98.14%
ANN	96.78%	88.41%	85.78%	72.56%
RF	100%	92.61%	89.62%	70.70%
SVM	100%	90.82%	89.42%	70.23%

Table A.2: The accuracy of the random forest models on the training, validation, test, and Google Maps images.

Model	Training	Validation	Test	Google Maps
Full features+Random Forest	100%	73.47%	76.37%	54.42%
PCA+Random Forest	100%	61%	63.86%	35.35%
Scaling+PCA+Random Forest	100%	61.77%	63.75%	35.95%
HOG(setting 1)+Random Forest	**100%**	**92.61%**	**89.62%**	**70.70%**
HOG(setting 2)+Random Forest	100%	91.34%	88.30%	69.30%
HOG(setting 3)+Random Forest	100%	78.32%	77.24%	60.47%
HOG(setting 4)+Random Forest	100%	91.70%	87.93%	66.05%

Appendix B

Figures

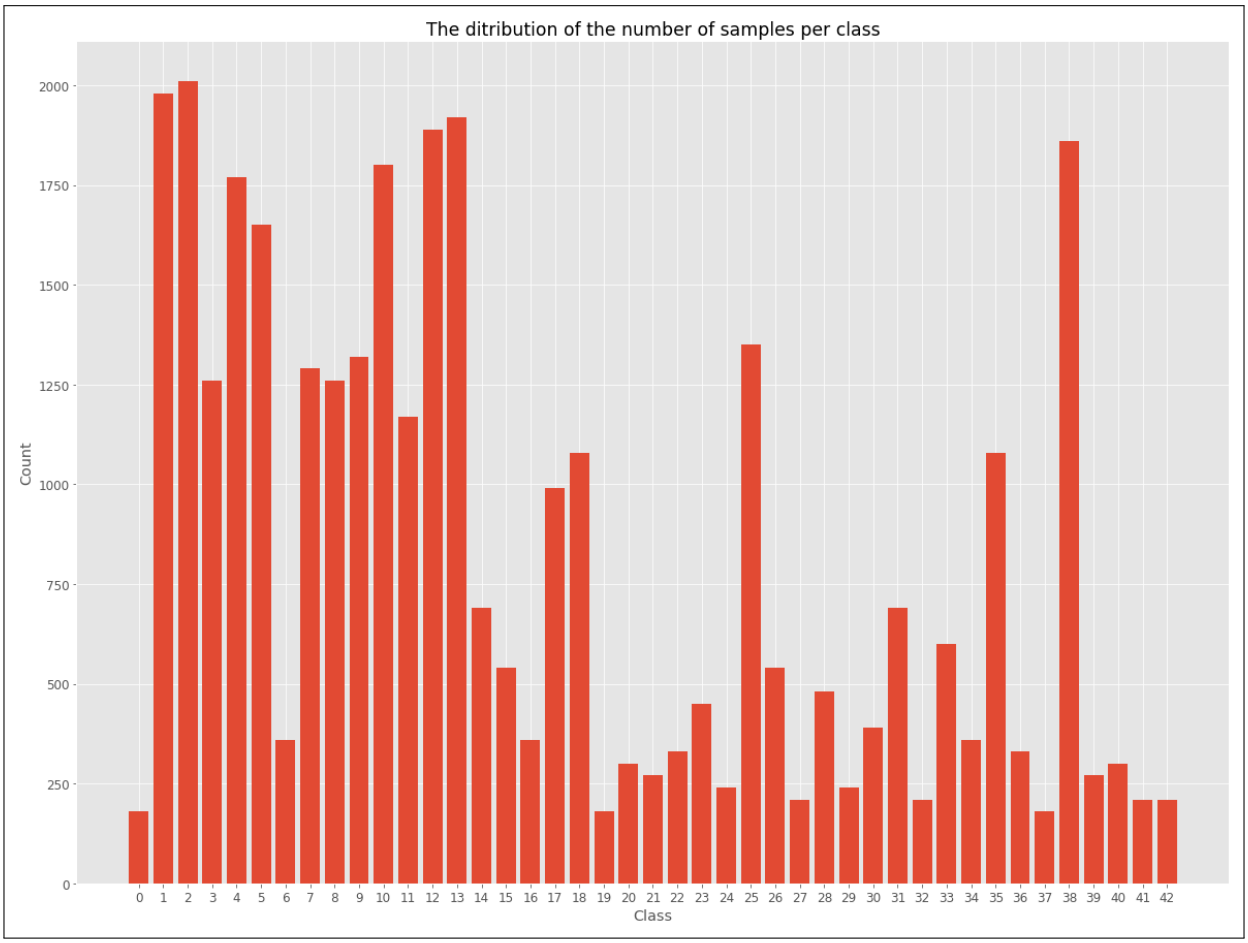


Figure B.1: The distribution of the samples per class.

	Traffic Sign Name
0	Speed limit (20km/h)
1	Speed limit (30km/h)
2	Speed limit (50km/h)
3	Speed limit (60km/h)
4	Speed limit (70km/h)
5	Speed limit (80km/h)
6	End of speed limit (80km/h)
7	Speed limit (100km/h)
8	Speed limit (120km/h)
9	No passing
10	No passing for vehicles over 3.5 metric tons
11	Right-of-way at the next intersection
12	Priority road
13	Yield
14	Stop
15	No vehicles
16	Vehicles over 3.5 metric tons prohibited
17	No entry
18	General caution
19	Dangerous curve to the left
20	Dangerous curve to the right
21	Double curve
22	Bumpy road
23	Slippery road
24	Road narrows on the right
25	Road work
26	Traffic signals
27	Pedestrians
28	Children crossing
29	Bicycles crossing
30	Beware of ice/snow
31	Wild animals crossing
32	End of all speed and passing limits
33	Turn right ahead
34	Turn left ahead
35	Ahead only
36	Go straight or right
37	Go straight or left
38	Keep right
39	Keep left
40	Roundabout mandatory
41	End of no passing
42	End of no passing by vehicles over 3.5 metric tons

Figure B.2: The names of the traffic sign classes.

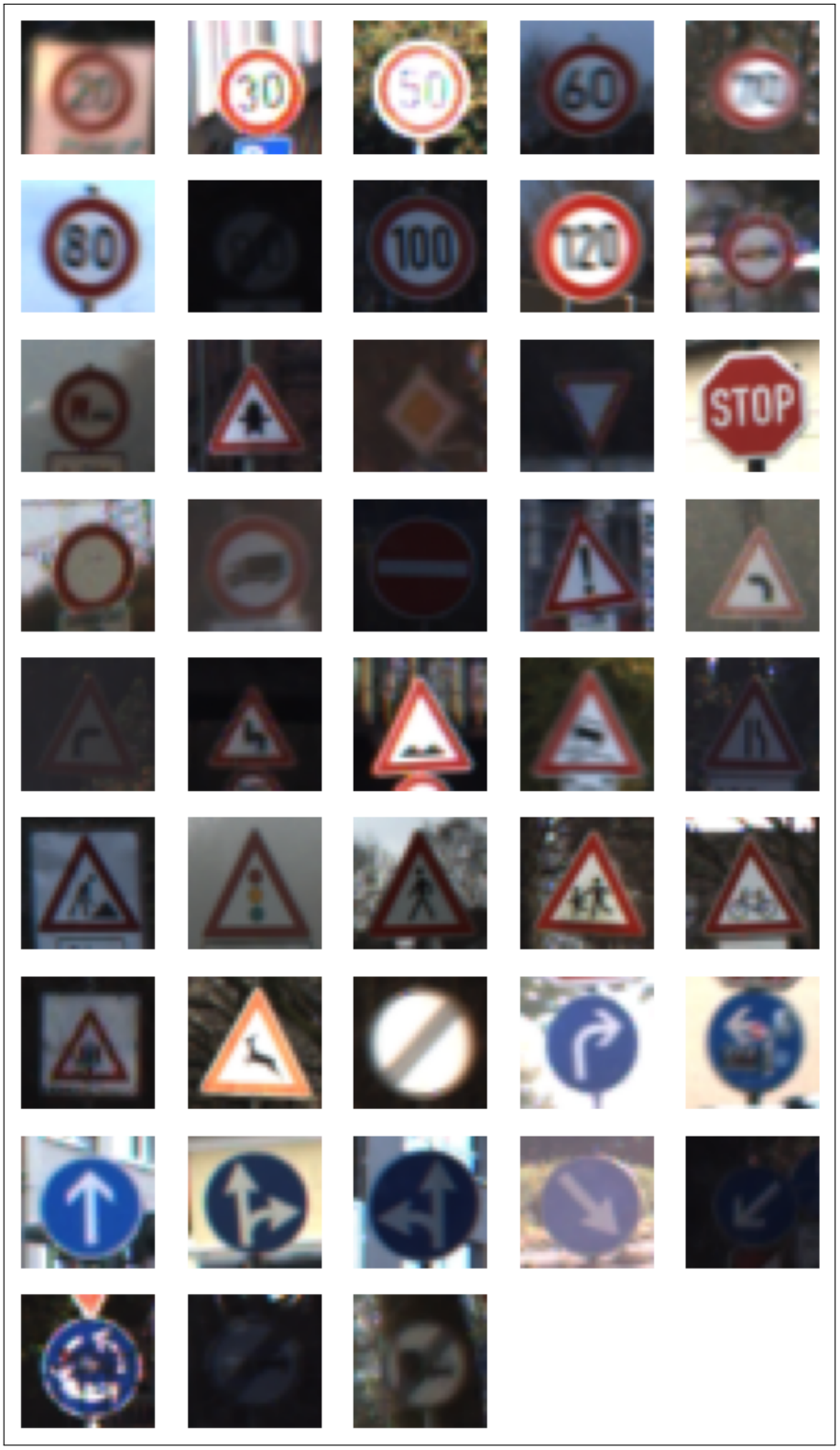
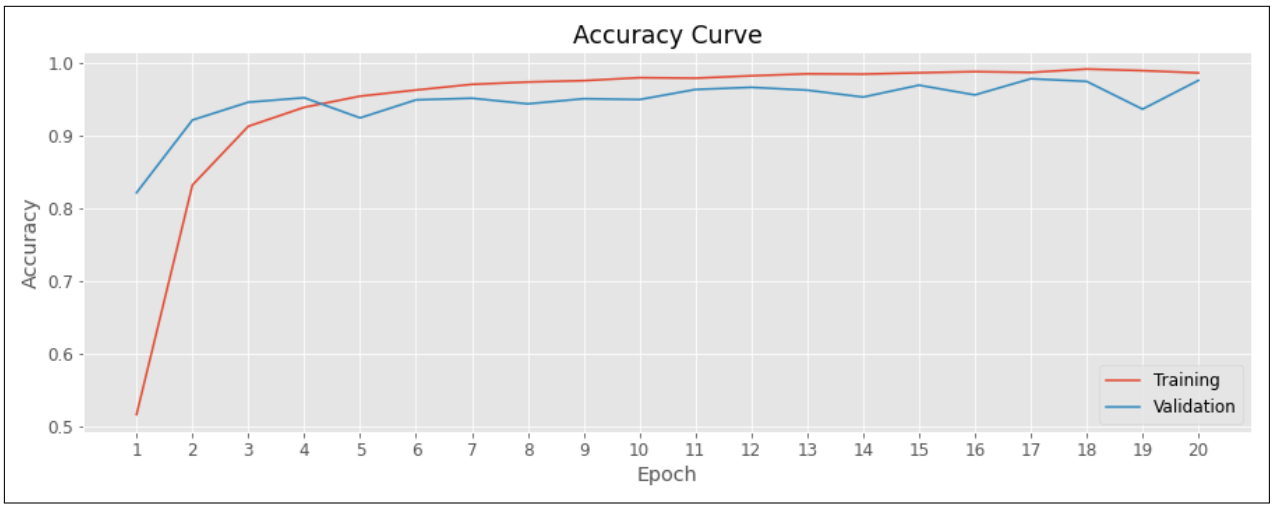


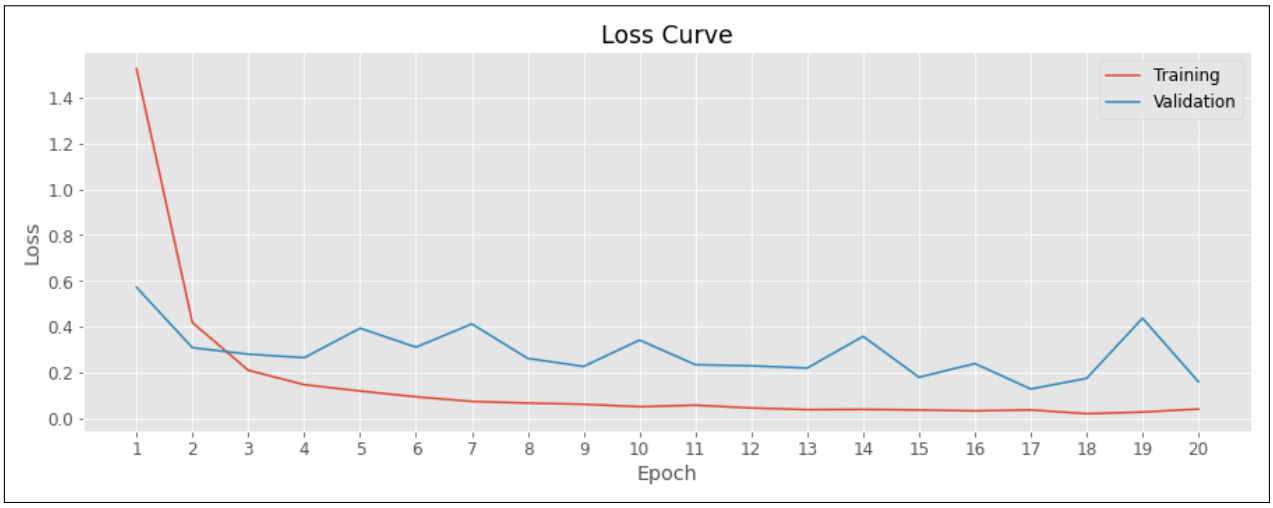
Figure B.3: A single training sample from each traffic sign class.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 16, 16, 128)       9728
_________________________________________________________________
batch_normalization_1 (Batch (None, 16, 16, 128)       512
_________________________________________________________________
dropout_1 (Dropout)          (None, 16, 16, 128)       0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 15, 15, 128)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 13, 13, 128)       147584
_________________________________________________________________
batch_normalization_2 (Batch (None, 13, 13, 128)       512
_________________________________________________________________
dropout_2 (Dropout)          (None, 13, 13, 128)       0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 12, 12, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 10, 10, 256)       295168
_________________________________________________________________
batch_normalization_3 (Batch (None, 10, 10, 256)       1024
_________________________________________________________________
dropout_3 (Dropout)          (None, 10, 10, 256)       0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 9, 9, 256)         0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 7, 7, 256)         590080
_________________________________________________________________
batch_normalization_4 (Batch (None, 7, 7, 256)         1024
_________________________________________________________________
dropout_4 (Dropout)          (None, 7, 7, 256)         0
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 6, 6, 256)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 9216)              0
_________________________________________________________________
dropout_5 (Dropout)          (None, 9216)              0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               4719104
_________________________________________________________________
dropout_6 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1028)              527364
_________________________________________________________________
dropout_7 (Dropout)          (None, 1028)              0
_________________________________________________________________
dense_3 (Dense)              (None, 43)                44247
=================================================================
Total params: 6,336,347
Trainable params: 6,334,811
Non-trainable params: 1,536
```

Figure B.4: A summary of the CNN structure.

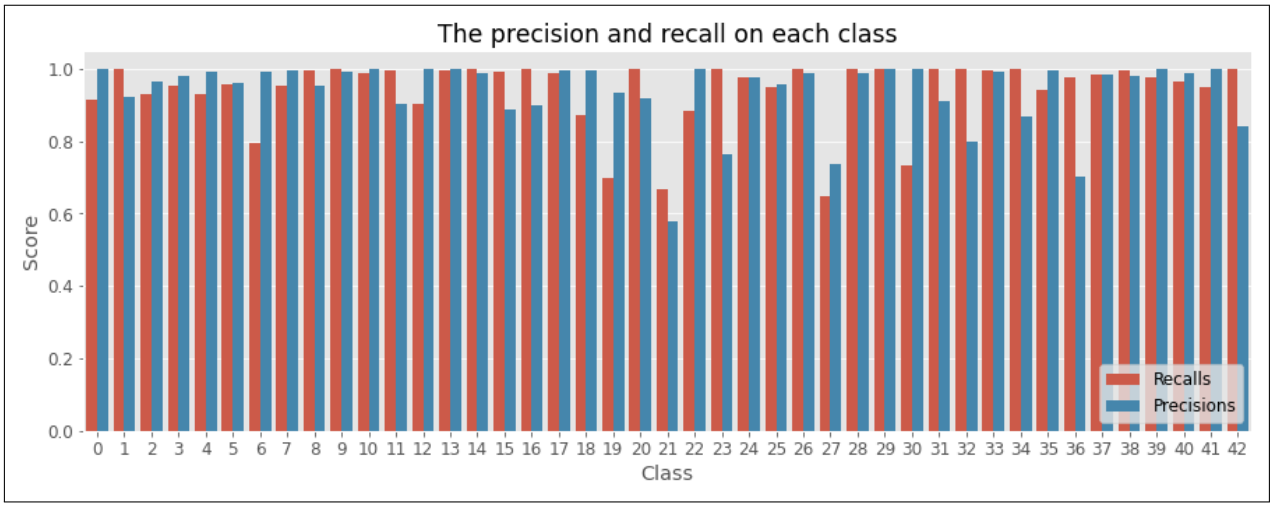


(a) Accuracy

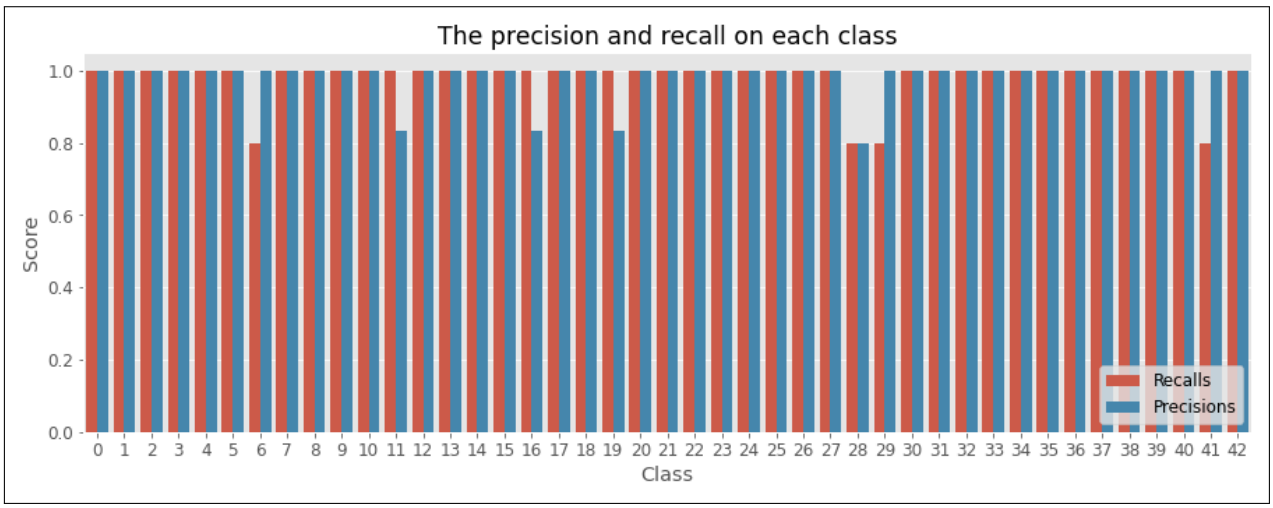


(b) Loss

Figure B.5: The accuracy and loss vs. epoch for the CNN model.

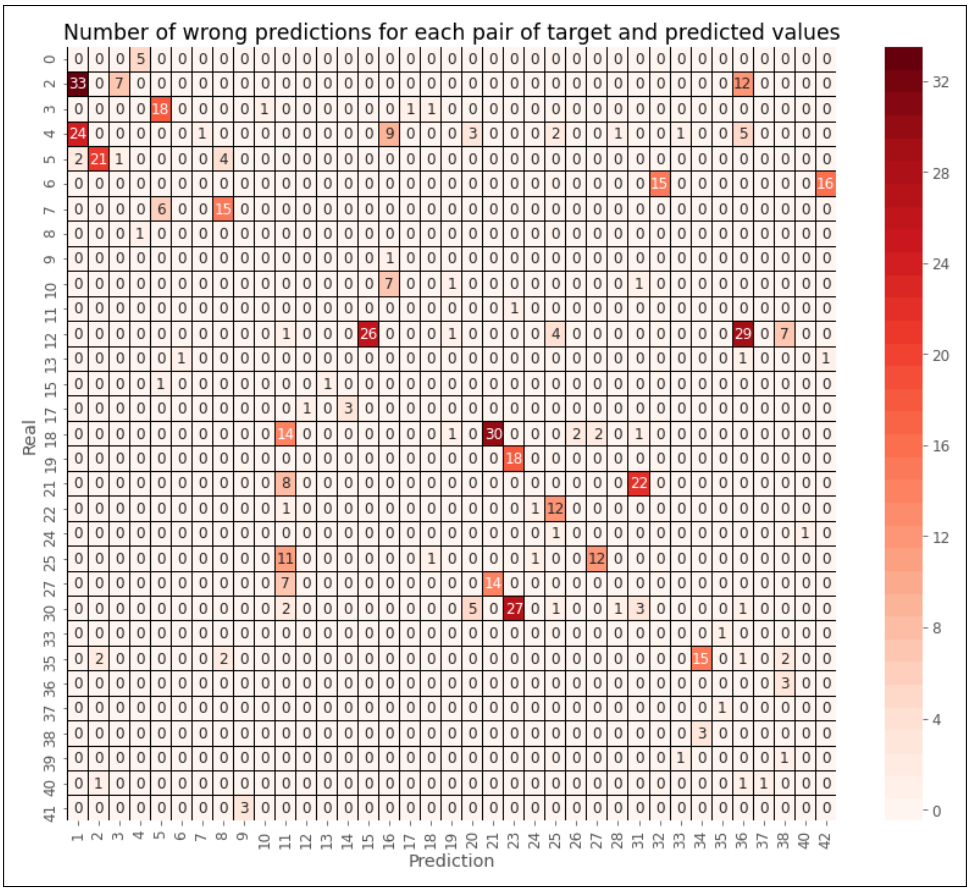


(a) Test images

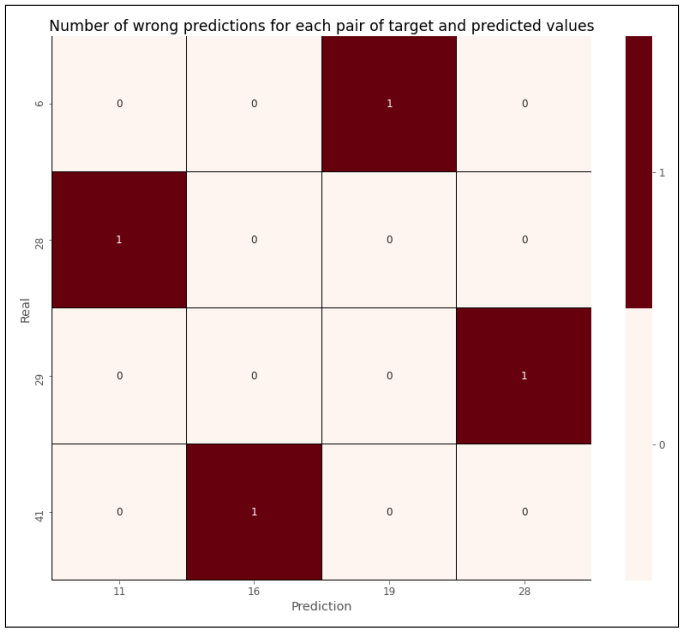


(b) Google Maps images

Figure B.6: The precision and recall of the CNN model per class on the test and Google Maps images.



(a) Test images

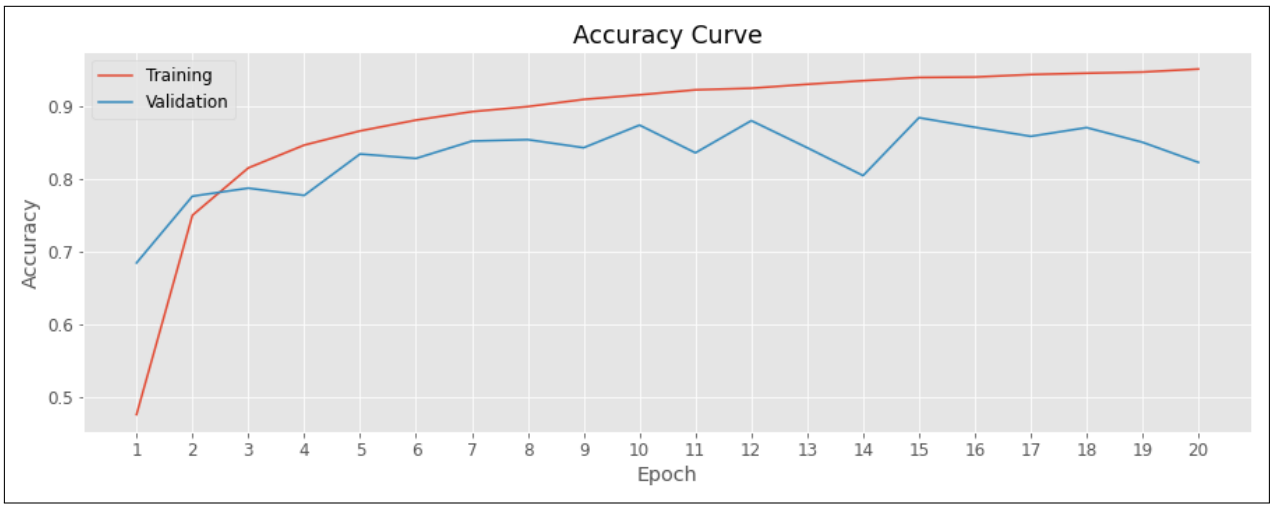


(b) Google Maps images

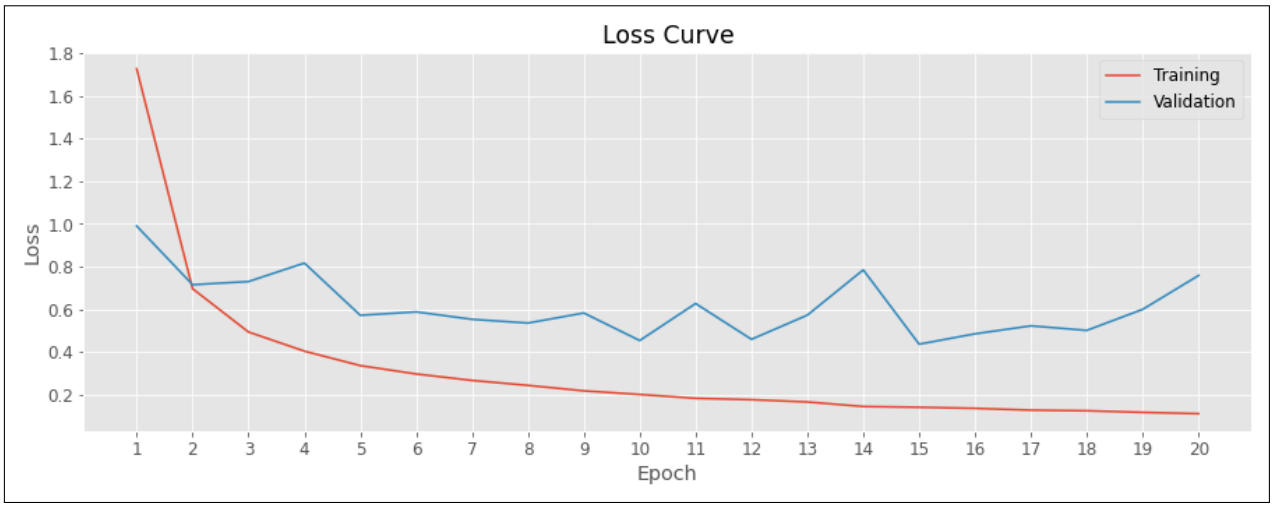
Figure B.7: The number of misclassifications for each pair of traffic signs for the CNN model on the test and Google Maps images.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 128)               393344
_________________________________________________________________
batch_normalization_1 (Batch (None, 128)               512
_________________________________________________________________
dense_2 (Dense)              (None, 128)               16512
_________________________________________________________________
batch_normalization_2 (Batch (None, 128)               512
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 256)               33024
_________________________________________________________________
batch_normalization_3 (Batch (None, 256)               1024
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_4 (Dense)              (None, 512)               131584
_________________________________________________________________
batch_normalization_4 (Batch (None, 512)               2048
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_5 (Dense)              (None, 512)               262656
_________________________________________________________________
batch_normalization_5 (Batch (None, 512)               2048
_________________________________________________________________
dense_6 (Dense)              (None, 43)                22059
=================================================================
Total params: 865,323
Trainable params: 862,251
Non-trainable params: 3,072
```

Figure B.8: A summary of the ANN structure.

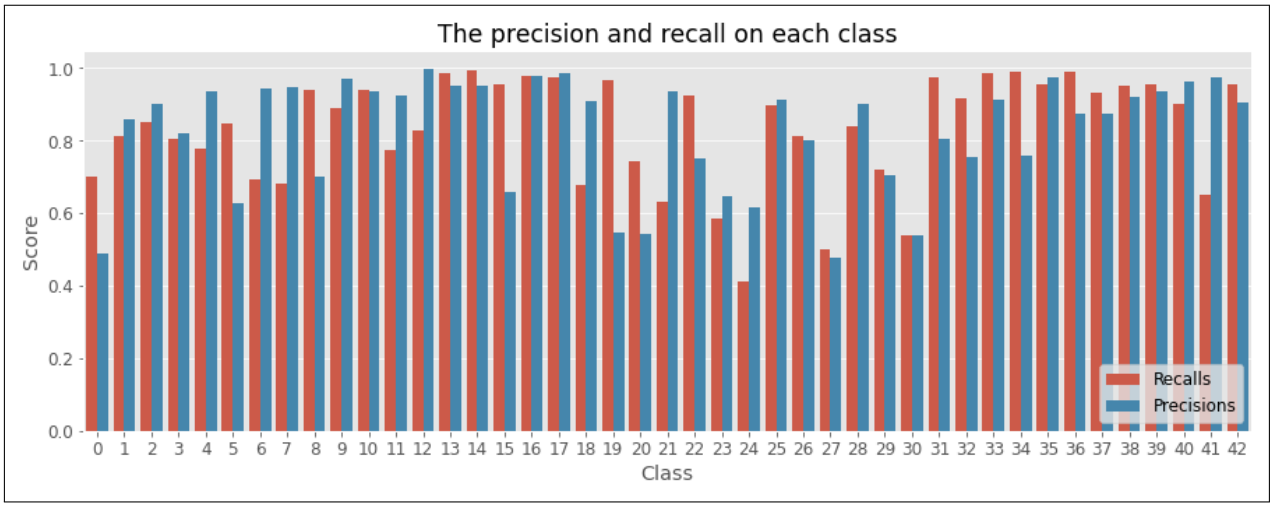


(a) Accuracy

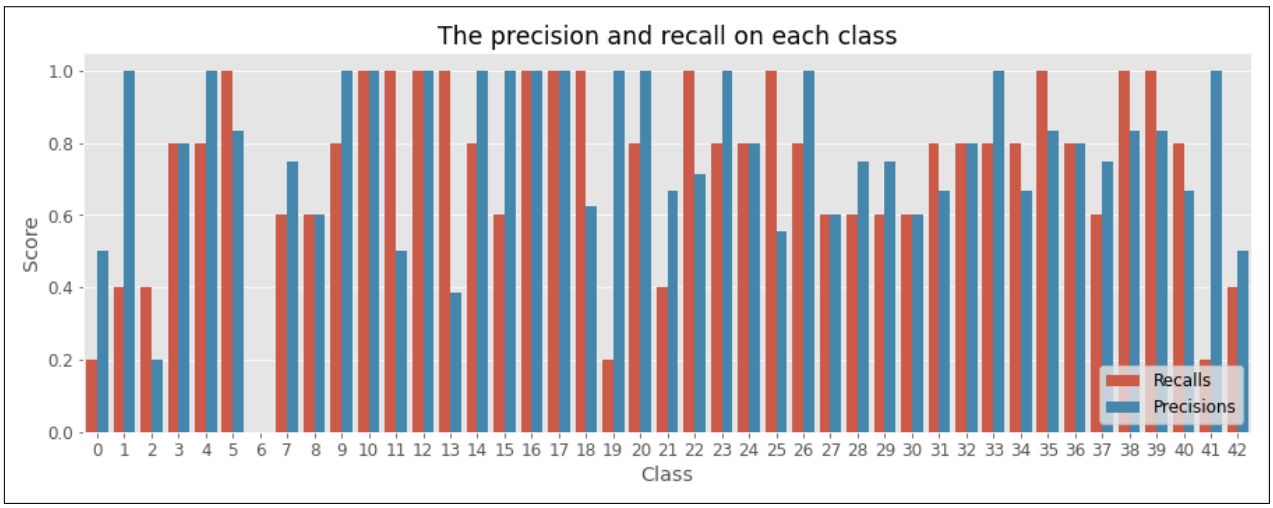


(b) Loss

Figure B.9: The accuracy and loss vs. epoch for the ANN model.

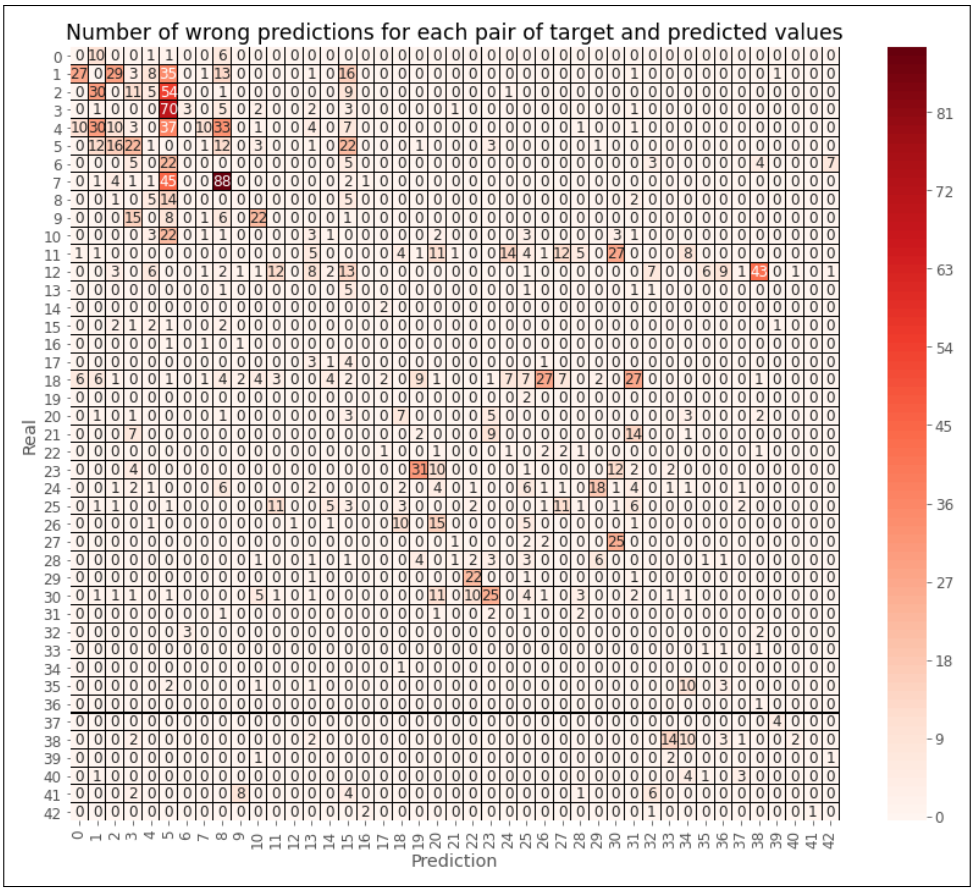


(a) Test images

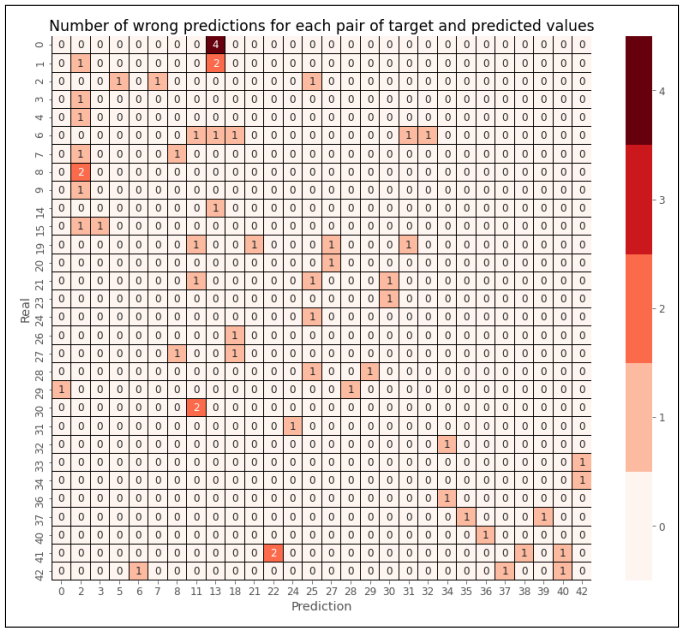


(b) Google Maps images

Figure B.10: The precision and recall of the ANN model per class on the test and Google Maps images.

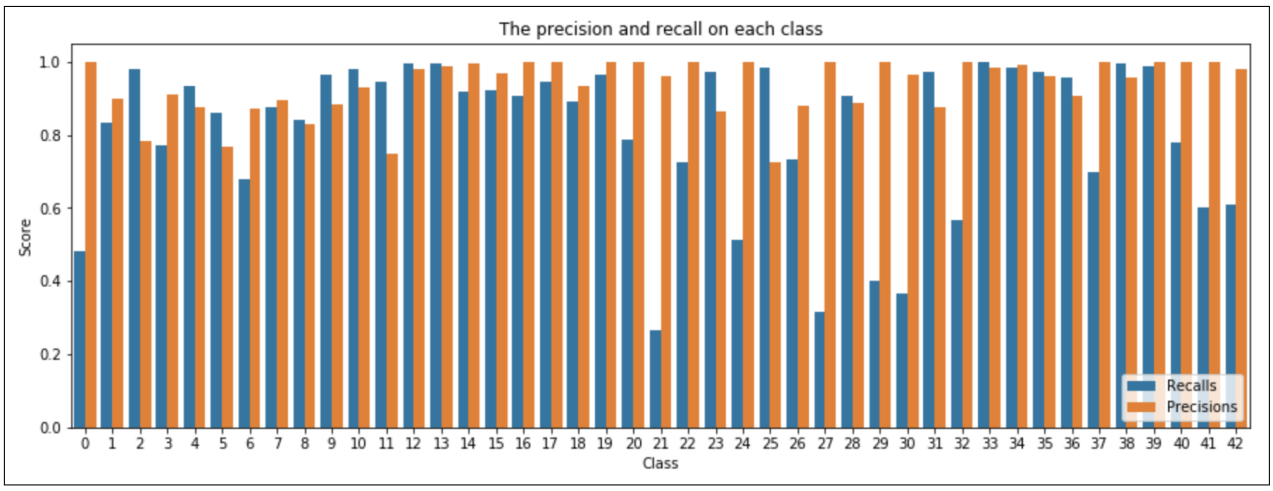


(a) Test images

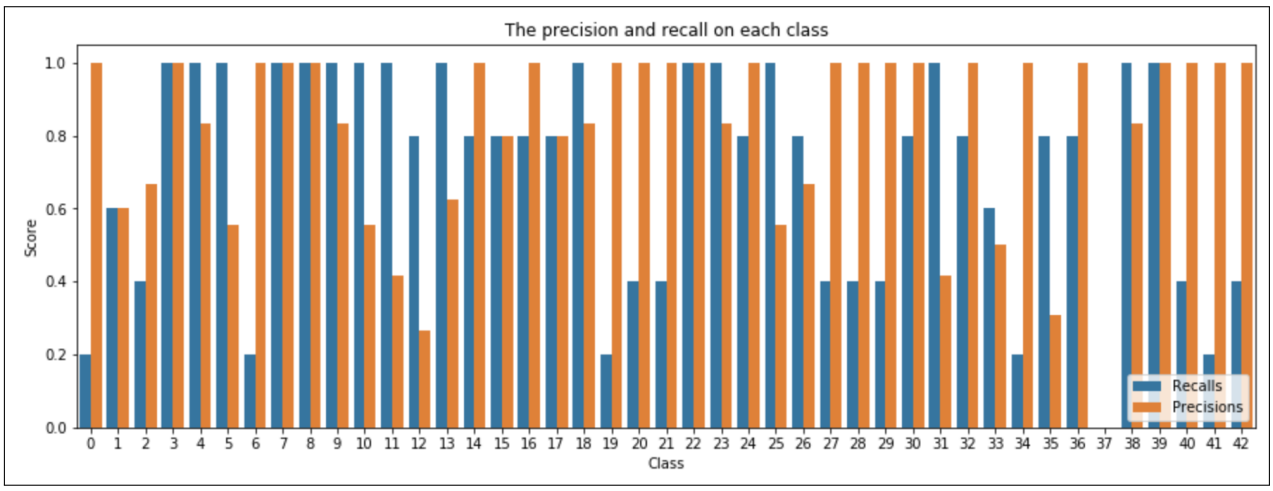


(b) Google Maps images

Figure B.11: The number of misclassifications for each pair of traffic signs for the ANN model on the test and Google Maps images.



(a) Test images



(b) Google Maps images

Figure B.12: The precision and recall of the Random Forest model per class on the test and Google Maps images.

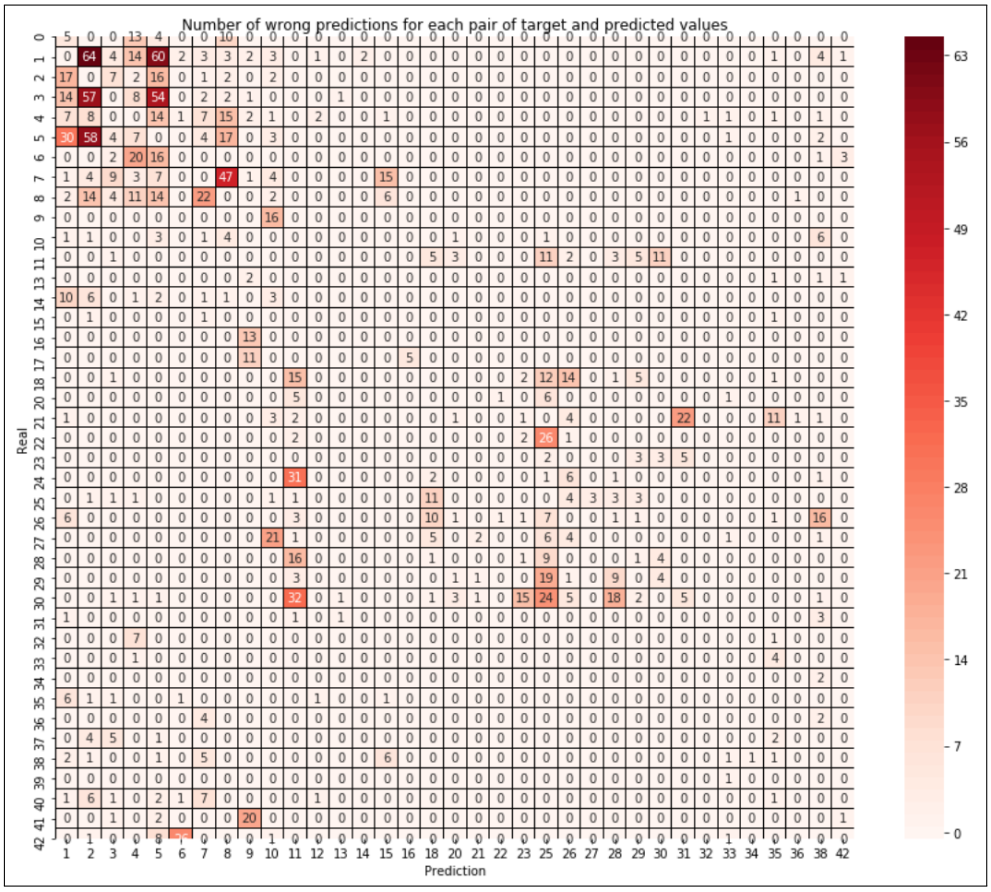
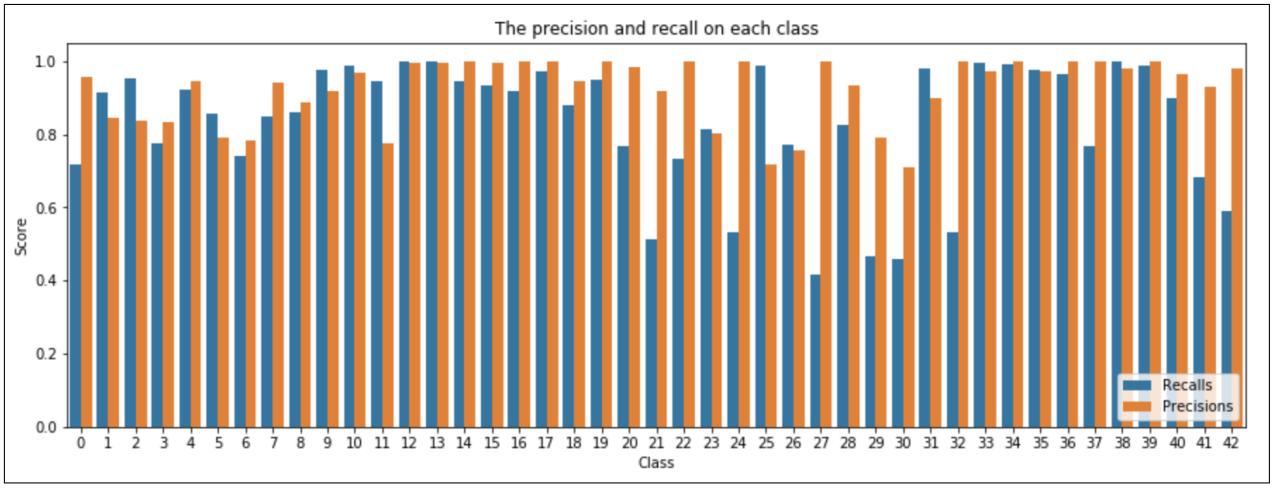
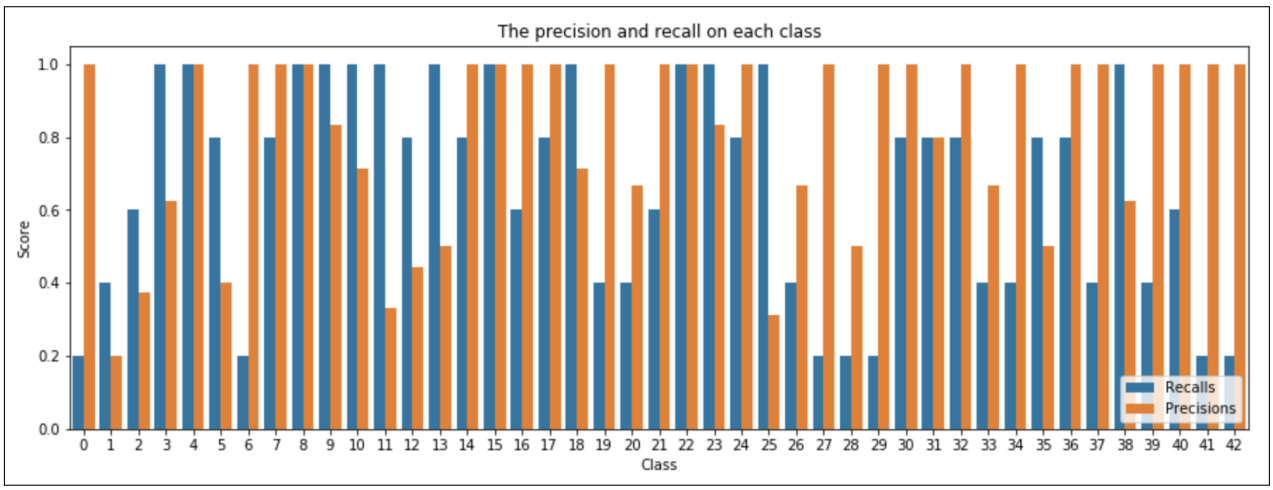


Figure B.13: The number of misclassifications for each pair of traffic signs for the RF model on the Google Maps images.



(a) Test images



(b) Google Maps images

Figure B.14: The precision and recall of the SVM model per class on the test and Google Maps images.

Bibliography

[1] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, “Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition.,” Neural Networks, vol. 32, pp. 323–332, 2012.

[2] F. Shao, X. Wang, F. Meng, T. Rui, D. Wang, and J. Tang, “Real-Time Traffic Sign Detection and Recognition Method Based on Simplified Gabor Wavelets and CNNs,” Sensors, vol. 18, no. 10, p. 3192, 2018.

[3] M. Green and J. Senders, “Human error in road accidents.,” Human Error in Road Accidents, 2004.

[4] A. Gudigar, S. Chokkadi and R. U , ”A review on automatic detection and recognition of traffic sign,” Multimedia Tools and Applications, p. 333–364, 2016.

[5] S. Houben, J. Stallkamp and J. Sa, ”Detection of Traffic Signs in Real-World Images: The German Traffic Sign Detection Benchmark,” in International Joint Conference on Neural Networks, 2013.

[6] Y. Ruicheck and C. G. Serna, ”Classification of Traffic Signs: The European Dataset,” IEEE Access, pp. 78136-78148, 2018.

[7] R. Timofte, K. Zimmermann and L. V. Gool , ”Multi-view traffic sign detection, recognition, and 3D localisation,” Machine Vision and Applications, p. 633–647, 2011.

[8] F. Larsson and M. Felsberg, ”Using Fourier Descriptors and Spatial Models for Traffic Sign Recognition,” in 17th Scandinavian Conference on Image Analysis, Ystad, 2011.

[9] S. Šegvić, K. Brkić, Z. Kalafatić and A. Pinz , ”Exploiting temporal and spatial constraints in traffic sign detection from a moving vehicle,” Machine Vision and Applications, p. 649–665, 2014.

[10] C. Grigorescu and N. Petkov, ”Distance sets for shape filters and shape recognition,” IEEE Transactions on Image Processing , pp. 1274 - 1286, 2003.

[11] S. Sivaraman and M. M. Trivedi, ”A General Active-Learning Framework for On-Road Vehicle Recognition and Tracking,” IEEE Transactions on Intelligent Transportation Systems, pp. 267 - 276, 2010.

[12] Z. Zhe, D. Liang, S. Zhang, X. Huang, B. Li and S. Hu, ”Traffic-Sign Detection and Classification in the Wild,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, 2016.

[13] R. Belaroussi, J.-P. Tarel, B. Soheilian, P. Charbonnier and N. Paparoditis, ”Road Sign Detection in Images: A Case Study,” in 20th International Conference on Pattern Recognition, Istanbul, Turkey, 2010.

[14] M. A. A. Sheikh, A. Kole and T. Maity, ”Traffic Sign Detection and Classification using Colour Feature and Neural Network,” in International Conference on Intelligent Control Power and Instrumentation (ICICPI), Kolkata, India, 2016.

[15] C. Liu, S. Li, F. Chang and Y. Wang, ”Machine Vision Based Traffic Sign Detection Methods: Review, Analyses and Perspectives,” IEEE Access, pp. 86578-86596, 2019.

[16] H. Luo, Y. Yang, B. Tong, . F. Wu and B. Fan, ”Traffic Sign Recognition Using a Multi-Task Convolutional Neural Network,” IEEE Transactions on Intelligent Transportation Systems, pp. 1100-1111, 2018.

[17] H. Gomez-Moreno, S. Maldonado-Bascon , P. Gil-Jimenez and S. Lafuente-Arroyo, ”Goal Evaluation of Segmentation Algorithms for Traffic Sign Recognition,” IEEE Transactions on Intelligent Transportation Systems, pp. 917-930, 2010.

[18] S. Kaplan Berkaya, H. Gunduz and O. Ozs, ”On circular traffic sign detection and recognition,” Expert Systems with Applications, vol. 48, pp. 67-75, 2016.

[19] J. Cao, C. Song, S. Peng, F. Xiao and S. Song, ”Improved Traffic Sign Detection and Recognition Algorithm for Intelligent Vehicles,” Sensors, vol. 19, 2019.

[20] H. Huang and L.-Y. Hou, ”Speed Limit Sign Detection Based on Gaussian Color Model and Template Matching,” in International Conference on Vision, Image and Signal Processing (ICVISP), Osaka, Japan, 2017.

[21] A. Saifuddin Saif, P. Paul, K. . M. Zubair and S. A. Shubho, ”Traffic Sign Detection based on Color Segmentation of Obscure Image Candidates: A Comprehensive Study,” International Journal of Modern Education & Computer Science, vol. 10, 2018.

[22] K. Lim, Y. Hong, Y. Choi and H. Byun, ”Real-time traffic sign recognition based on a general purpose GPU and deep-learning,” PLoS ONE, vol. 12, no. 3, 2017.

[23] T. Yang, X. Long, A. K. Sangaiah, Z. Zheng and C. Tong, ”Deep detection network for real-life traffic sign in vehicular networks,” Computer Networks, vol. 136, pp. 95-104, 2018.

[24] I. Matoš, Z. Krpić and K. Romić, ”The Speed Limit Road Signs Recognition Using Hough Transformation and Multi-Class Svm,” in International Conference on Systems, Signals and Image Processing (IWSSIP), Osijek, Croatia, 2019.

[25] P. Saxena, N. Gupta, S. Y. Laskar and P. P. Borah, ”A Study on Automatic Detection and Recognition Techniques For Road Signs,” International Journal of Computational Engineering Research (IJCER), vol. 5, no. 12, pp. 24-28, 2015.

[26] R. Qian, B. Zhang, Y. Yue and F. Coenen , ”Traffic Sign Detection by Template Matching based on Multi-Level Chain Code Histogram,” in 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), Zhangjiajie, China, 2015.

[27] Q. Hu, S. Paisitkriangkrai, C. Shen, A. v. d. Hengel and F. Porikl, ”Fast Detection of Multiple Objects in Traffic Scenes With a Common Detection Framework,” EEE Transactions on Intelligent Transportation Systems, vol. 17, pp. 1002-1014, 2016.

[28] P. Pandey and R. Kulkarni, ”Traffic Sign Detection Using Template Matching Technique,” in Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), Pune, India, 2018.

[29] M. Swathi and K. Suresh, ”Automatic traffic sign detection and recognition: A review,” in International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET), Chennai, India, 2017.

[30] A. Ellahyani, M. ElAnsari and I. ElJaafar, ”Traffic sign detection and recognition based on random forests,” Applied Soft Computing, vol. 46, pp. 805-815, 2016.

[31] M. Kobayashi, M. Baba, K. Ohtani and L. Li, ”A method for traffic sign detection and recognition based on genetic algorithm,” in IEEE/SICE International Symposium on System Integration (SII), Nagoya, Japan, 2016.

[32] T. Muthukumaresan, B. Kirubakaran, D. Kumaresan and A. Satheesan, ”Recognition of Traffic Sign using Support Vector Machine and Fuzzy Cluster,” IJSTE - International Journal of Science Technology Engineering, vol. 2, no. 10, pp. 190-197, 2016.

[33] B. Mikulski, ”Data Craft,” [Online]. Available: https://www.mikulskibartosz.name/pca-how-to-choose-the-number-of-components/. [Accessed April 2020].

[34] J. Li, “Udacity Traffic Sign Classifier,” GitHub, 21-Feb-2017. [Online]. Available: https://github.com/lijunsong/udacity-traffic-sign-classifier/blob/master/Traffic_Sign_Classifier.ipynb.

[35] J. Jin, K. Fu and C. Zhang, ”Traffic Sign Recognition With Hinge Loss Trained Convolutional Neural Networks,” in IEEE Transactions on Intelligent Transportation Systems, vol. 15, no. 5, pp. 1991-2000, Oct. 2014.