Traffic Sign Image Classification

CMPE 255 - Project Report

SAN JOSÉ STATE UNIVERSITY

By,

Poojitha Amin (011811306)

Krutika Mude (012535094)

Suraj Khurana (011464427)

Professor: Dr. Gheorghi Guzun

1. Introduction

1.1. Motivation

Traffic sign detection and classification in the outdoor environment has real world applications and it has received a lot of attention due to its contribution to driver assistance, safety and autonomous driving. Sometimes the road users ignore the traffic signs due to lack of visibility or attention and this can lead to potentially dangerous situations. In such scenarios, a fully automatic road sign detection system can be useful. However, traffic sign recognition is a constrained and a challenging problem, as the algorithms have to cope with variabilities like the lighting conditions, motion blur, complex dynamic environments and high accuracy demands. Additionally, getting insights into computer vision and machine learning is also a driver for selecting this topic.

1.2. Objective

In this project, we propose to build an image recognizer to classify the traffic signs from images. It recognizes and interprets various traffic signs using image specific information. The goal of the detection is to try different approaches to build the model and optimize it for accuracy and efficiency. Solving this problem is most fundamental for the autonomous cars to operate on roads. Academically, the objective of this project is to explore data mining techniques in image classification. Not only does this project offer us insight in to the data and attributes, but also the preprocessing and standardization techniques used in image classification.

1.3. Literature/Market Review

Various traffic sign recognition systems have been proposed in the past. However, most of these systems work on good quality images. Many vehicle manufacturers such as Tesla, Inc., Continental AG, BMW and Hanover are researching and developing this technology. It was also introduced by Volkswagen in Audi A8. In this project, we aim to develop an effective, efficient traffic sign recognition system to deal with images having the following properties:

- Different shapes
- Different signs
- Different positions
- Different sizes
- Blurred images

2. System Design and Implementation

2.1. Algorithms Considered

From our market research we concluded that there is no one classifier that works best with image classification. However, we experimented with the following techniques and compared their results.

- **Random Forest -** Ensemble learning method that constructs multiple decision trees.
- **Linear SVM -** Supervised discriminative classifier known to achieve good accuracy with image classification.
- **K nearest neighbours -** Simple classification algorithm which relies on the distance between the feature vectors for prediction.
- **MLP -** Since neural networks is a popular classifier for image classification, we included multi layer perceptron , which in essence is a neural network based algorithm.
- **Keras CNN -** Keras is a high-level neural networks wrapper API which allows for easy prototyping and experimentation with neural network.

2.2. Technologies and Tools

- **OpenCV** - Image preprocessing and object detection.
- **Scikit-learn** - Image preprocessing, dimensionality reduction and classification.
- **Keras** - Neural network API.
- **Numpy** - Standard python library used with Scikit-Learn and OpenCV.
- **Pickle** - Store trained models which are later used to classify a given image.
- **HTML and JQuery** - Develop the user interface that accepts image as input, sends it to the backend and displays the class it belongs to.
- **Flask** - Processes the image obtained from UI, classifies it and returns the class name of image.
- **Python 2.7/3.6 -** Programming languages used.

2.3. Architecture Decisions

- Use of HPC to process and train classifiers with large datasets.
- Use of pickled files to load the models for creating the web application.
- Dimensionality reduction using resize, grayscaling and other scikit-learn dimensionality reduction techniques.
- Use of Keras as an API for implementing neural network.

2.4. System Design and Data Flow

2.4.1. System Design

The figure below shows the high level design of the system. We begin with the object detection, image preprocessing followed by the classifier learning phase. In the classifier learning phase we experimented with various machine learning algorithms and listed some of the best performing models.

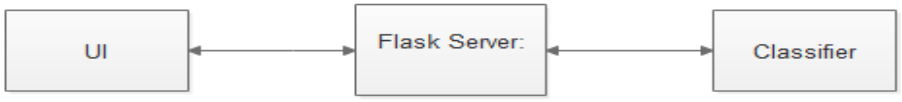

Figure 1. High level architecture

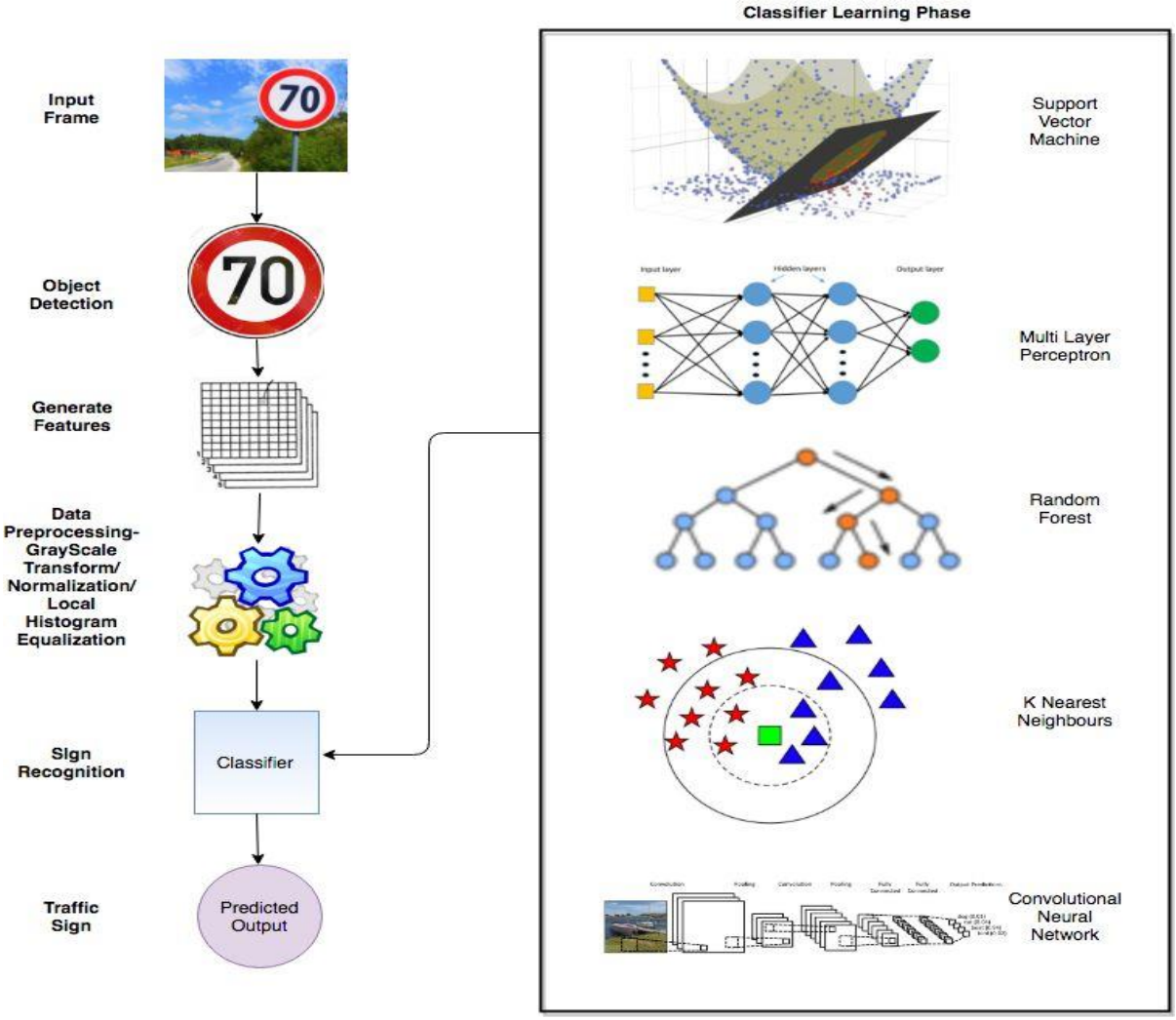

Figure 2. Components and Data flow

2.5. Component Details

2.5.1. Image Pre-Processing

The aim of image pre-processing is to improve the image by suppressing unwanted distortions and enhance the image features necessary for further processing.

- **Gray Scale Transformation:** It is the process of converting a color image to a grayscale image by converting the RGB values to grayscale values. It lowers the complexity of the model by compressing the images. We achieved this using different libraries such as OpenCV library - cv2.COLOR_RGB2GRAY and scikit-image.
- **Local Histogram Equalization:** It is the process of enhancing images of low contrast, by spreading out the most frequent intensity values in an image. This method can lead to better image views when background and foreground are both dark or bright in images. We used the function skimage.filters.rank.equalize(image, selem) from the scikit image to implement this.
- **Image Normalization:**We normalised the images in order to scale the pixel intensity values in the range of [0,1]. This process is also known as contrast stretching or histogram stretching.
- **Image Blurring:** We used image blurring for transforming a grainy image obtained by color thresholding, to a smoother image. This was helpful in creating a bounding box around the Region of Interest.
- **Anti Aliasing:** We have used anti aliasing on training images, to smoothen out jaggedness in low resolution images.
- **Other Dimensionality Reduction techniques:** We also explored other dimensionality reduction techniques like PCA and SVD.

2.5.2. Object Detection

2.5.2.1. Approach 1

The images in the test set are not centered in all cases. They may be found in the top-left or bottom-right corner of the image. The classifiers from sklearn may not be able to classify such images correctly. This problem is solved by the convolutional neural network. We performed convolution operation by sliding a filter over the entire image. With this we tried to achieve translation invariance in the model and be able to detect the object from the image irrespective of where it is.

2.5.2.2. Approach 2

As sign boards usually have a peculiar color to draw attention of users, for example, a red colored background or border, and since this color is starkly different from the background (ie noise), we color thresholded images, to obtain sign board blocks. To refine grainy images

obtained by color thresholding, we used blurring(bilateral blurring) to obtain a smooth block in image. We then crop the original image with dimension of the smooth block obtained by blurring.

2.5.3. Image Augmentation

We tried data augmentation to reduce overfitting. This is the process of creating more images, by changing the size and orientation of the image. This was done by using ImageDataGenerator instance of the Keras library. However, this did not contribute to the accuracy.

2.6. UI Screenshot

3. Experiments

3.1. Dataset

Our image dataset came from the traffic sign dataset provided in the link below- http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset. It contains over 50,000 images in total including the train, test and validation dataset. Each image is a photo of a traffic sign and the images are divided into 43 different classes. The images are not of fixed dimensions and are not necessarily centered. We have images of 15x15 to 250x250 pixels. We plotted the training dataset to see the distribution of classes and we could see that the data set was clearly imbalanced in nature.

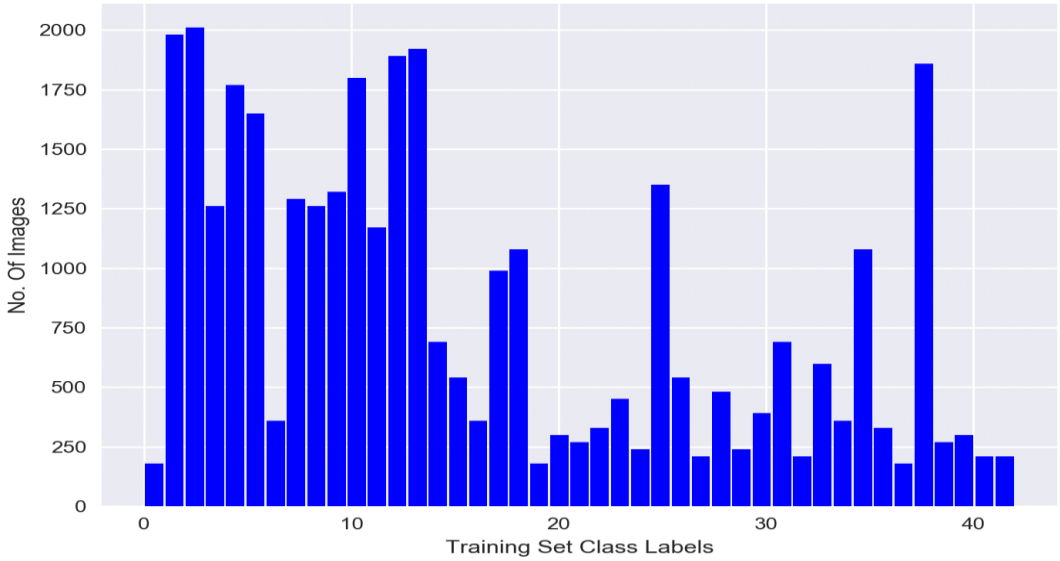

Figure 3. Class Distribution

Below are some of the sample images from the data set:

End of No Passing

Turn Right Ahead

Pedestrians

Figure 4. Sample Images

3.2. Methodology

We have used n-fold cross validation and train-test-validation split as required. For classifiers, which were quick to build, for example, Multilayer Perceptron(MLP), Random forest we have used n-fold cross validation with n = 6. For CNN, Linear SVM and KNN , we have used train-test-split with the ratio 80 : 20. We experimented with different ratios of train-test split and n, but did not achieve significant difference in F1 or accuracy scores.

3.3. Graphs and Plots

A normal color graph contains a lot of data points, and is not very useful for detecting misclassified points. Hence, we use log normalized confusion matrix.

3.3.1. Confusion matrix and F1 score plots for sklearn classifiers

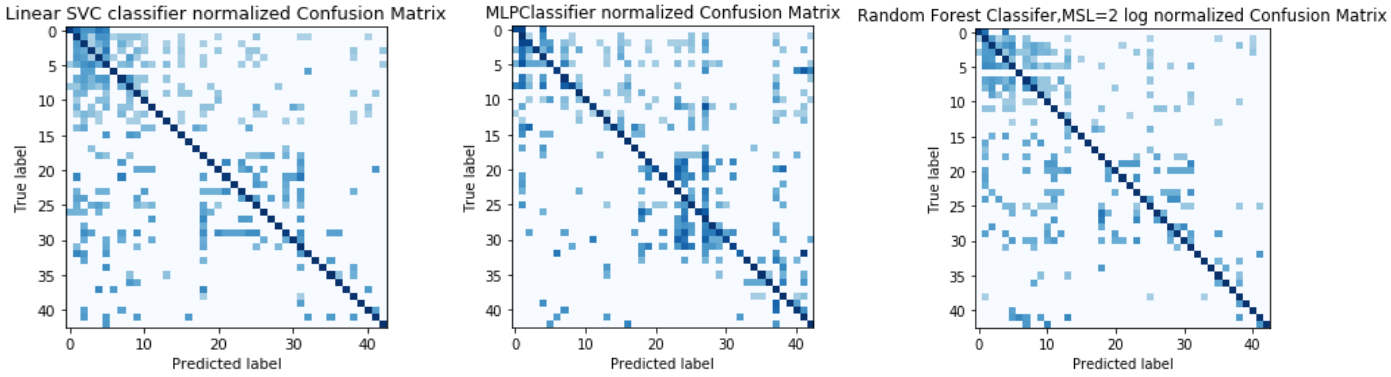

Figure 4. Confusion Matrix

In MLP Classifier, 3 hidden layers gave us the best F1 score. Beyond 4 hidden layers, we could observe overfitting. In Random Forest Classifier F1-Score increases with decrease in min_sample_leaf parameter.

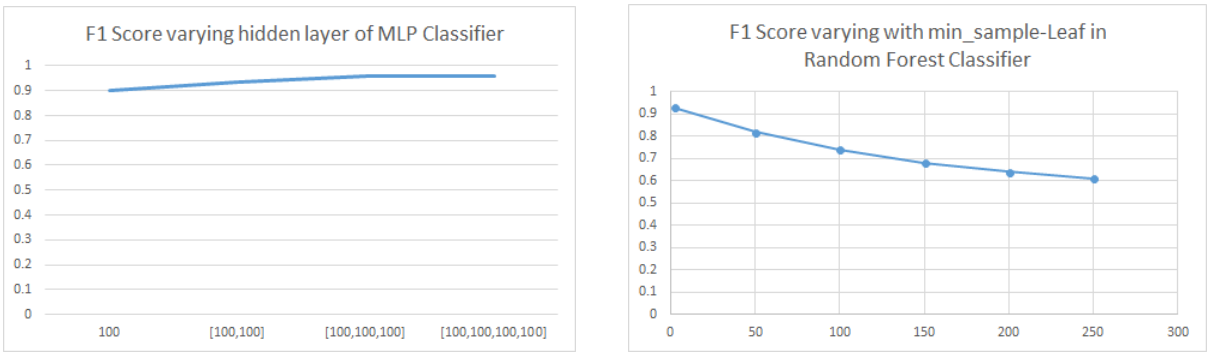

Figure 5. F1 score plots on varying parameters

3.3.2. Keras CNN plots

The plots below show the comparison in the training/validation accuracy and loss when the model was overfitting and after the measures were taken to reduce overfitting. We can see that the initial validation loss was higher than the training loss. This shows that the model had overfitted and had memorised the training data set. This was improved by introducing Dropout and multiple Convolution layers.

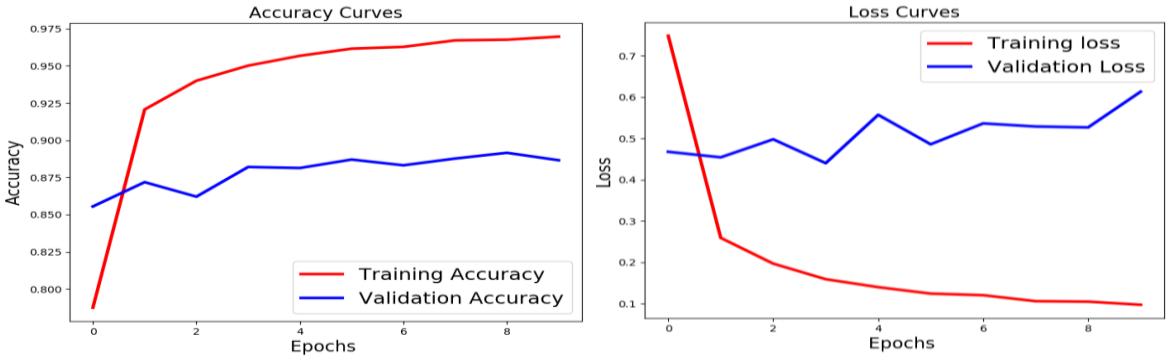

Figure 6. Loss and Accuracy Curves with overfitting

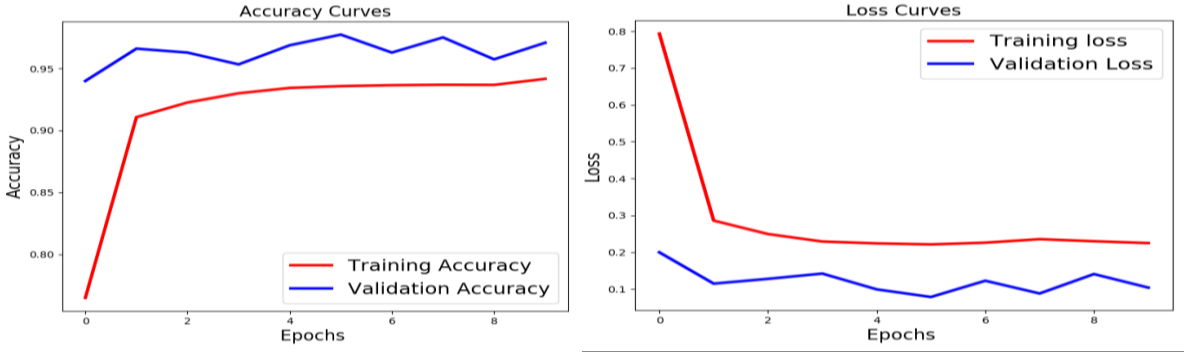

Figure 7. Loss and Accuracy Curves after overfitting reduction

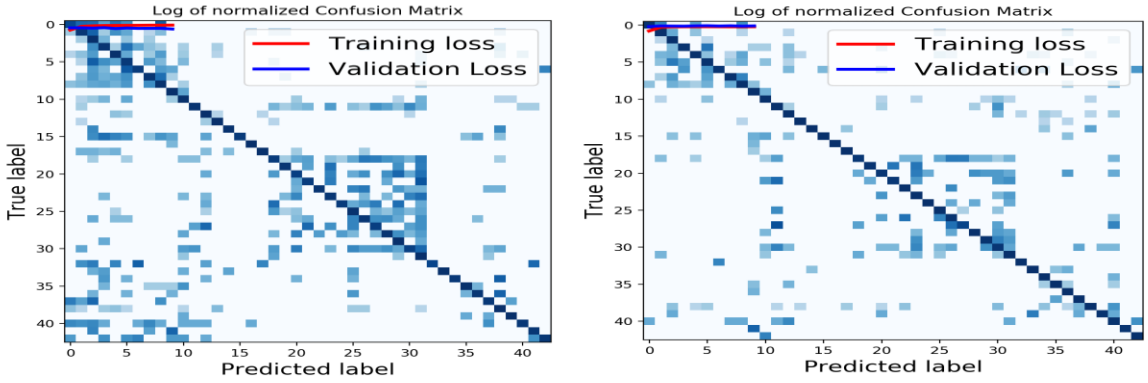

Figure 8. Confusion matrix comparison for Keras models

3.4. Analysis of results

We measured the F1 score and accuracy for various models and we could see that the MLP Classifier and Keras Convolutional Neural network outperformed the other classifiers. There is a marginal difference between the results obtained from MLP classifier and Keras CNN. The CNN architecture with EPOCHS 20 and BATCH_SIZE 10 did fairly well achieving:

- Training Accuracy : 96.71%
- Test Accuracy : 95.77%
- Validation Accuracy: 98.39%

We observe that in all classifiers, the most misclassified classes are “Speed Limit” signs and similar signs with triangular shape.

Figure 10. Misclassified Samples

Table 1. Algorithm performance comparison

Model	**F1-Score/Accuracy**	**Training Time**
MLP Classifier	0.972	0:02:31
Random Forest	0.939	0:00:07
Linear SVM	0.926	00:47:25
KNN	0.88	01:50:52
Keras CNN	0.983	00:20:00

Table 2. MLP Performance Evaluation

Number of hidden layers	**Iterations**	**Training Time**	**Performance**
100	100	00:02:31	0.972
[100,100]	200	00:01:46	0.934
[100,100,100]	200	00:02:20	0.96
[200,100]	150	00:04:20	0.963

4. Discussions and Conclusions

The optimal design of a model is always a challenging task. We tested with a variety of machine learning models as there is no single algorithm that works best in all the application areas. Our experimental results showed that the performance of the model lies with the choice of its parameters.

Difficulties faced:

- Training a few models on such large dataset.
- Exploring image processing techniques.

Things that did not work:

- Python 3 setup on HPC and hence could not integrate Keras model with the UI.
- Image augmentation did not contribute to the accuracy.
- Working with KNN classifier. It made exceptionally big models(time taken to build, size of pickle files), to be loaded, reused or tweaked for performance.
- Python notebook and Matplotlib are not available on HPC.
- Achieving very good object detection, only using color thresholding.

Conclusion

As part of an intelligent transportation system, the sign recognition model can be used to guide a driver or an autonomous car with the traffic signs on the present traffic conditions. In this project, we developed a traffic sign detection system using state-of-the-art image processing and classification techniques.

5. Task Distribution

Poojitha Amin	Suraj Khurana	Krutika Mude
● Explored image pre-processing steps. ● Implemented Keras CNN model. ● Implemented evaluation metrics. ● Tested models with various parameters.	● Implemented object detection in image. ● Implemented Random Forest, MLP models, . ● Implemented evaluation metrics. ● Tested models with various parameters	● Implemented SVM and KNN models. ● Implemented front end of the application. ● Implemented evaluation metrics. ● Tested models with various parameters