

University of Hertfordshire

Department of Physics, Astronomy and Mathematics

MSc Data Science

Project report: 7PAM2002 – Data Science Project

TRAFFIC SIGN BOARD RECOGNITION SYSTEM USING DEEP LEARNING AND MACHINE LEARNING

VIJAY GOVARDHAN PASULA

Supervisor: Carolyn Devereux

MSc Final Project Declaration

This report is submitted in partial fulfilment of the requirement for the degree of Master of Data science at the University of Hertfordshire (UH).

ABSTRACT

One of the most important features in a self-driving car is to detect and identify the traffic sign boards for right guidance and instruction. This ability to recognize the traffic sign board accurately by self-driven car is very crucial because this effects the safety and security of humans. In this paper I have built three models for developing a traffic sign board recognition system namely - Convolutional Neural Network (CNN) model, Support Vector Machines (SVM) and Random Forest. I am using a data set from Kaggle called GTSRB - German Traffic Sign Recognition Benchmark to develop these models. My aim for this project is to compare all the three models in terms of accuracy, training speed, and testing speed. Out of these three models, the performance of CNN is highly acceptable with an accuracy of around 95% on the validation data and 37 minutes of training speed and 3 seconds of testing speed. Even though the Support vector machines gives an accuracy around 96%, but the testing speed is 2 minutes which is not suitable in applications like this, though the training speed is 8 minutes. Unfortunately, the performance of random forest model could not be traced because google colab sessions are getting disconnected due to longer training speeds. However, when I compare the results of CNN and SVM, I am using CNN model for traffic sign board recognition system because of its testing speed (3 seconds) and accuracy (95%).

TABLE OF CONTENTS

Chapter 1: Background	4
Chapter 2: Literature review.....	7
Chapter 3: Data set description.....	10
3.1 Source of Data set	10
3.2 Size of Data set	10
Chapter 4: Methodologies	15
4.1 The Convolutional neural network	15
4.1.1 Layers in CNN model.....	15
4.2 Support Vector Machines.....	17
4.2.1 Parameter of SVM model.....	17
4.3 Random Forest.....	19
4.3.1 Parameter of Random Forest.....	20
4.3.2 Advantages and disadvantages of RF.....	21
Chapter 5: Analysis of all models.....	22
5.1 Convolutional Neural Network (CNN).....	22
5.1.1 Compiling the CNN model.....	23
5.1.2 Training the model.....	25
5.2 Support Vector Machines.....	25
5.3 Random Forest.....	26
Chapter 6: Results.....	28
6.1 Convolutional Neural Network (CNN).....	28
6.2 Support Vector Machines.....	30
6.3 Random Forest.....	32
6.4 Comparison of all 3 models.....	33
Conclusion.....	34
References.....	35

CHAPTER 1: BACKGROUND

Nothing is more important than human life in this world. And humans are more dependent on technology and automation in this generation. In olden days, people used to drive cars which has no automation system in-built in their vehicles but things have changed over time. Big companies like Mercedes-Benz, Tesla, Audi, Toyota, and Ford have developed self-driving cars in which no human effort is required. Everything is taken care by machines in self-driven vehicles. Speed control, vehicle recognition, and traffic sign recognition system are the key features in self driving vehicles. They are aiming to make such great machines which high accuracies keeping human life and safety as their at most priority.

When it comes to safety and security in self driving cars, one of the most important features of the automated vehicles is the ability to detect the traffic sign boards on roads. As humans, we are able to identify the traffic sign board with our naked eye and take decisions accordingly within fraction of seconds. Imagine – am driving a car and couldn't recognize the stop board on the road, and went ahead. I could either end up meeting with an accident or at least I would be asked for the penalty for my mistake. Either way, it is too risky and dangerous if I can't drive according to traffic rules and do not follow the sign boards.

If recognizing traffic sign boards is such an important factor for manual driving, then how much more it is important in self driven vehicles?



Fig 1: self-driving cars recognizing traffic sign boards on road.

Source: https://images.wapcar.my/file1/7ea571f7ee1e488baa14273469fdbeb4_800.jpg

Generally speaking, there are many traffic signs in real world, like speed limits, no passing, stop boards, turn left or right, school junction, work in progress, cycling, children crossing and zebra crossing etc. It is highly important for a self-driven car, to recognize these traffic sign boards and follow them to prevent accidents and penalties on road.

Few Examples of traffic sign boards:



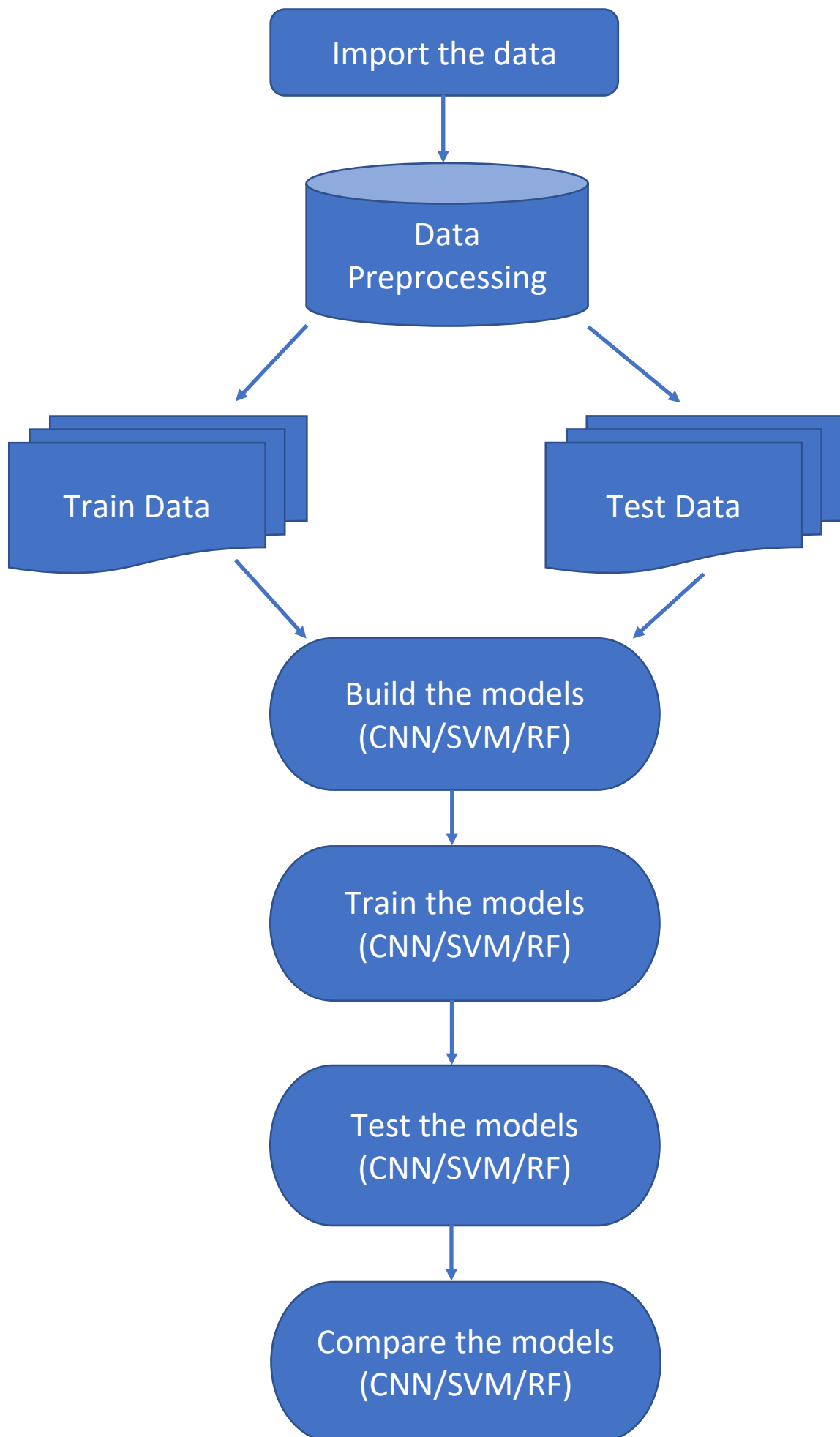
For this purpose of developing a traffic sign board recognition system in self-driving cars, I have chosen three image classification models in Data Science application field – one model from deep learning (Convolutional Neural Networks) and two models from machine learning (Support Vector Machines and Random Forest).

Three models:

1. Convolutional Neural Networks model (CNN).
2. Support Vector Machines model (SVM).
3. Random forest model.

After building all the models, comparing their accuracies, training speeds, and performance of these models on test data and finally plotting the accuracy plots is the main aim of this project.

PROJECT PLAN



CHAPTER 2: LITERATURE REVIEW

In my study and research, Nazmul Hasan and team ⁽²⁰²⁰⁾, have come up with a proposal that the traffic sign recognition system has two parts in it. First part is localization and second part is recognition. In the first localization part, the sign board region is location and it is identifies by established a rectangular area. Whereas in second part i.e., recognition part, the rectangular box provided the result of traffic sign located in that particular region.

Bulla Premamayudu ^(Bulla, et al, 2022), in recent years, has published his experiment on available public traffic sign data set, where he defined a CNN model and gained ~94% precision value from classification report and also ~80% recall value for the same CNN model. But in my paper, I also calculated the f1 score in my experiment which is the harmonic mean value of both precision and recall values.

Other research done by Keshav M and his team ⁽²⁰²¹⁾, have developed a system called – ADAS, which stands for Advanced Driving Assistance System, which is used to assist the drivers to detect and classify the natural objects, speed regulations and also by detecting the traffic sign boards that can enhance the safety of vehicle and the driver. This system can hugely revolutionize this technology and using this kind of ADAS system can highly reduce numerous accidents on roads.

According to Lobo V.B ^(Lobo, et al, 2022), in his research paper, he has built a CNN model and trained this model on different data set from German, Indian and hybrid and came up with interesting results where German data set gives an accuracy of ~99%, whereas Indian data set has given accuracy of ~91% and a hybrid data set has given an accuracy of ~95% respectively, this concludes that the accuracy is not only dependant on the model building but also different on different data sets.

Not only the accuracy changes when tested on different data sets. But the number of epochs also will affect the accuracy. I found that Fernando W.H.D ⁽²⁰²¹⁾, has come up with an interesting results where he

concludes that the even in the increase of number of epochs will have no effect on test accuracy. In fact, the research estimates the high accuracy in this traffic sign recognition system by using different skip connections feature extraction.

There are many regularization techniques that can be performed on the model. But out of all different regularization techniques, L1 & L2, dropout, and batch normalization techniques are common. Gyanchandini M ⁽²⁰²²⁾, have used all the regularization techniques that are mentioned above and they have good influence in building the model. In other words, using different regularization techniques will have its own impact in training the model, and their future work is focused on analysis of CNN architecture design to focus on further increase of the model performance. And also they are planning to use image annotation methods to improve the frame data lead to a better traffic sign board recognition system.

The other popular technique called transfer learning in which a model will be pre-trained using a data set and then it is tuned over the other data set. The normal procedure of pre-processing and regularization techniques are used to improve the overall performance of the model. This kind of experiment is done by Samad A. and his team ⁽²⁰¹⁸⁾, they first pre-trained on German data set and then tuned on Pakistani data set. And achieved a training accuracy of minimal 41%, which is not suitable in real world applications.

Apart from standard image classification model, (CNN), Zeniarja J. and her team^(Zeniarja et al, 2016) have also performed the unsupervised models in traffic sign recognition system, based on a technique called bag-of-visual words. First, by clustering the key points and then clustering the images by using SURF. Cluster key points of images into many groups and taking the centroid of every group as a visual word. Therefore, number of visual words is same as the number of clusters. And then construct a histogram of each image and cluster them.

Along traffic sign board system, the voice alert system within the vehicle is also very important because few self-driving cars have only a display of traffic sign board but doesn't have the ability to respond according to the sign board. In such cases, it is highly essential for a self-driving car to have a voice alert system within the vehicle, that can be implemented using CNN. In his paper (Shah k, et al 2021), he developed a system where a voice message will be sent to the driver whenever the traffic is detected and alerts him of the symbol. A map will be displayed to see the traffic sign and follow them accordingly to take appropriate decisions by driver.







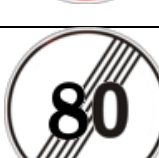
I found an interesting research paper by Vinh T.Q (Vinh T.Q. et al, 2015) in which he presents a particular design and precise implementation on a real time traffic sign board recognition system which is purely based on a technique called Friendly ARM Tiny4412 board. He along with his team, have developed three main techniques based on shape matching, colour segmentation and SVM. He is able bring an accuracy around 90% at fifteen frames per second on a board called Friendly ARM Tiny4412.










CHAPTER 3: DATASET DESCRIPTION










3.1 Source of dataset: The data set is chosen from Kaggle (Data source, et al 2011) where a competition held at International Joint Conference on Neural Networks (IJCNN) 2011.










3.2 Size of dataset:










This dataset contains more than 50,000 images and in 43 different classes (traffic sign boards) as below. And the data set has 43 subfolders from a train folder and test folder as new data of same categories.

1. Image of Speed limit (20km/h)	
2. Image of Speed limit (30km/h)	
3. Image of Speed limit (50km/h)	
4. Image of Speed limit (60km/h)	
5. Image of Speed limit (70km/h)	
6. Image of Speed limit (80km/h)	
7. Image of End of speed limit (80km/h)	

8. Image of Speed limit (100km/h)	
9. Image of Speed limit (120km/h)	
10. Image of No passing	
11. Image of No passing vehicle over 3.5 tons	
12. Image of Right-of-way at intersection	
13. Image of priority road	
14. Image of Yield	
15. Image of stop	
16. Image of No vehicles	

17. Image of Vehicle > 3.5 tons prohibited	
18. Image of No entry	
19. Image of General caution	
20. Image of Dangerous curve left	
21. Image of Dangerous curve right	
22. Image of Double curve	
23. Image of Bumpy Road	
24. Image of Slippery Road	
25. Image of Road narrows on the right	

26. Image of Road work	
27. Image of Traffic signals	
28. Image of Pedestrians	
29. Image of Children crossing	
30. Image of Bicycles crossing	
31. Image of Beware of ice/snow	
32. Image of Wild animals crossing	
33. Image of End speed + passing limits	
34. Image of Turn right ahead	

35. Image of Turn left ahead	
36. Image of Ahead only	
37. Image of Go straight or right	
38. Image of Go straight or left	
39. Image of Keep right	
40. Image of Keep left	
41. Image of Roundabout mandatory	
42. Image of End of no passing	
43. Image of End no passing vehicle > 3.5 tons	

CHAPTER 4: METHODOLOGIES

4.1 CONVOLUTIONAL NEURAL NETWORK:

The Convolutional Neural Network (CNN), is a special type of the neural networks in the field of data science that is primarily used for different applications like image classification, image captioning and speech recognition. The concept of CNN was first introduced by “Yann le cun” in 1998 for the purpose of digit classification where he used only one convolutional layer. And then it became popular in 2012 by Alexnet which used many convolutional layers to achieve the state of art on image net.

Since my project is about image classification, I'll give more information about it. Image classification involves in extracting the features of image to observe few patterns in the given data set. When I use ANN instead of CNN for image classification purposes, then I end up being very expensive in terms computational process since the trainable parameters become extremely large in the method.

CNNs are built with many layers inside it, which takes in the input image and process it and then gives the right prediction of the image.

4.1.1 The following are the layers in Convolutional Neural Network (CNN):

- a. Convolutional layer: The most important layer in the CNN are convolutional layers because this layer operates the convolutional operation on the input images by using kernel – of different sizes, filters – of different sizes and activation functions with an aim of extracting the features of an image.
- b. Pooling layer: From a computational point of view, the same process happens in pooling layer just as in convolutional layer. But the difference is that I either can take the average or maximum value form the result, depending on my application. I chose to apply max pooling in my CNN model in order to reduce the dimensions of the features, which reduces the no. of parameters to learn and the also the computational process within the network.

- c. Dense layer: In CNN architecture, this is a simple layer of neurons, in which each neuron receives the input from all the neurons of the previous layer in CNN model, hence it called as Dense. This layer is used in classifying the image based on the output from the previous convolutional layers.
- d. Dropout: The term drop out means dropping out units from a neural network. By dropping out units, which means am temporarily dropping them or removing them from the network along with its incoming and outgoing connections. By using dropout technique in CNN model, I avoid the problem of overfitting as well.
- e. Flatten layer: This layer will convert the data into a one dimensional array as it gives input to the next layer. This performs the flattening operation of the output of convolutional layers to create a single long feature vector in the CNN model. This layer is finally connected to fully connected layer. And the input of the final layer is the output of the flatten layer.
- f. Fully connected layer: So far, by using the pervious layer in CNN model, image was greatly reduced in terms of dimensions and now I have a tightly meshed layers. With all the individual sub-images that are formed now will be linked again in order to recognize the connections between them and finally it helps in predicting the image and classifying it correctly.

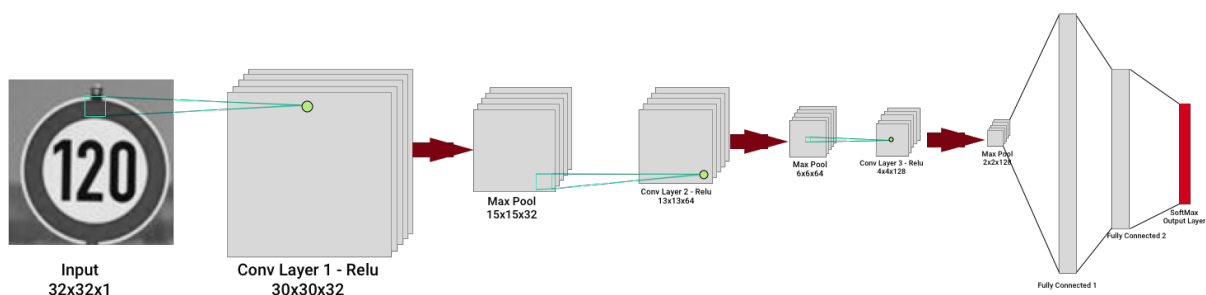


Fig 2: This image explains the layers in CNN model

Source: <https://github.com/Mini-Project-Jain-University/Traffic-Sign-Recognition>

4.2 SUPPORT VECTOR MACHINES

The second method am using in this project is Support Vector Machines (SVM). This SVM is a supervised machine learning algorithm, which can be used for both classification and regression analysis. Few applications of SVM includes – face detection, hand writing detection, image classifications, and text categorizations. But my project is mainly focused on image classification problem since am dealing with the traffic sign board recognition system.

However, since the SVM is widely used for classification problems. In this algorithm, the data is usually plotted in n-dimensional space, where 'n' represents the number of features that I have. Then I will perform classification by drawing the hyper plane that differentiate the classes between each other.

In computer vision world, the image will be in form of array of pixels. Suppose if the size of the array is 100 x 100 x 3 – then the first 100 represents the width and second 100 represents the height and next 3 represents the RGB channel values. The values in this array are arranged from 0-255 which actually represents the intensity of pixels at every point.

4.2.1 Parameters in SVM classifier:

a. *Gamma:*

This parameter will define the influence of the single training samples reach the values leads to biased results. Low values of gamma means far and distant while high values of gamma mean close and near. This is also used to give curvature weight of the decision boundary.

b. *C-parameter:*

This C parameter controls the cost of misclassifications. In other words, this parameter informs us how much to avoid the misclassifying each training sample. The larger the C value, the high the cost of misclassification. The lower the C value, the low the cost

of misclassification. The only problem with this parameter is that it can take any value and will not have any direct interpretation. Therefore, it is very difficult to choose the correct value and I have to rely on cross validation technique in order to find the suitable value for my application.

c. *Nu-parameter:*

Since it is difficult to choose the right c value, it was regulated to new regularization parameter ν . This parameter is bounded between 0 and 1 and has a direct interpretation. There are margin errors and support vectors in SVM and this ν parameter describes the upper boundary on the fraction of the margin errors and on the other side the lower boundary on the fraction of support vectors which are relative to the number of training samples. For example – I choose ν value to be 0.05 in this project, which means at least 5% of my training samples are at the cost of a smaller margin and at least 5% of my training samples are support vectors. Both C parameter and ν parameter in SVM are equal in regards to their power of classification. The regularization of ν parameter is easier to understand and interpret when compared to C .

d. *Kernel in SVM:*

SVMs use mathematical functions that can be defined as the kernel. There are different types of kernels in SVM:

- i. Linear
- ii. RBF (Radial Basis Function)
- iii. Polynomial kernel

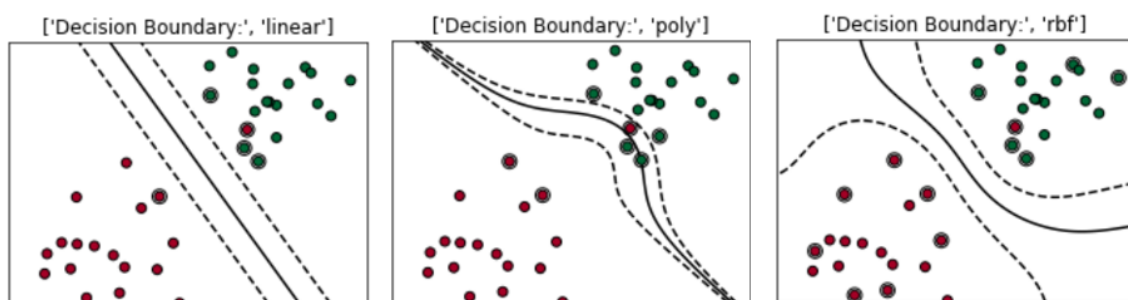


Fig 3: Kernels in SVM model

Source: <https://www.merahealthgenie.com/kzcc.aspx?iid=56660689-python+svm+kernel+types&cid=7>

4.3 **RANDOM FOREST:**

The third method that I am using for developing a traffic sign board recognition system is random forest. Before I give reasons for why I have chosen random forest, let me talk about decision tree and its limitations.

Decision tree will try to simulate the thinking of human process by binarizing every step in the decision. So, for example, at each step in the process, the algorithm will choose between true or false to move forward. This decision tree algorithm is very simple and easy to understand and widely used in machine learning models. However, one of the main concerns with decision trees is that its difficulty in generalizing a particular problem. That means the model learns any given dataset so well that it cannot be reliable on a new unseen data set. To avoid this problem, a new type of decision tree algorithm is designed by gathering many trees trained over variations of same dataset and using a voting system which combines them and then decides the best result for every data sample observed in the data set. This concept is called Random Forest.

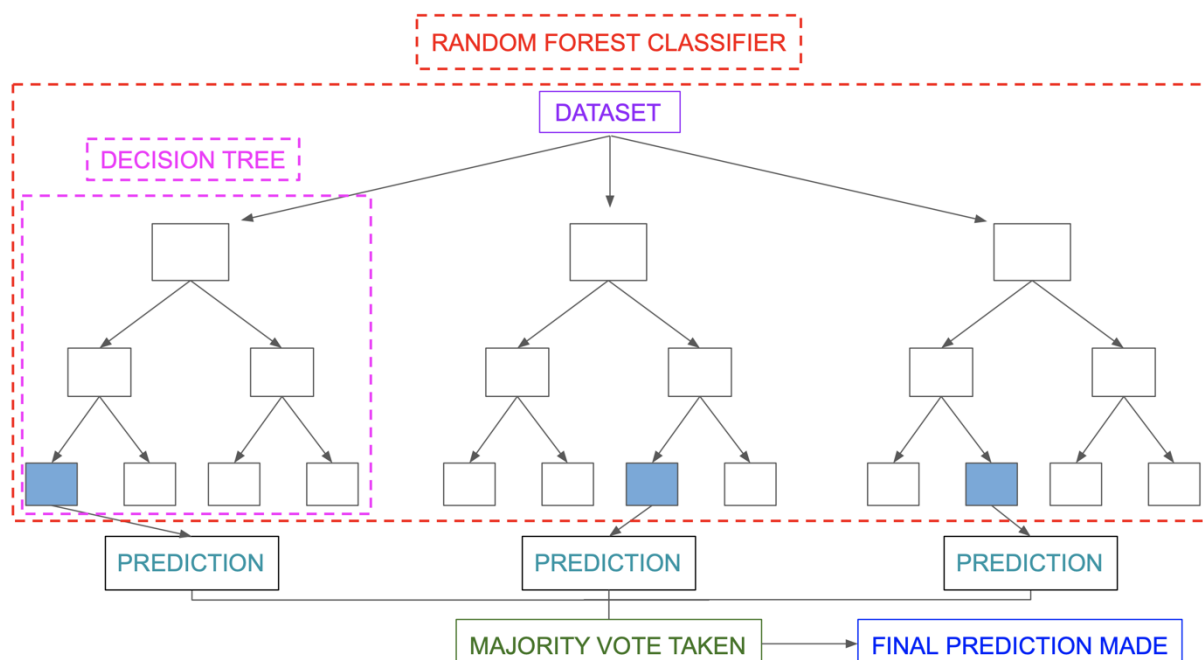


Fig 4: Random Forest classifier model

Source: <https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/>

The random forest algorithm will create a set of decision trees from a randomly selected subset of the training data set. That's why it is called random forest, because it randomly selects and then take decision based on majority. Basically, the set of decision trees is a subset of training set and then it collects the votes or follow the majority principle from different decision trees for final prediction.

The random forest algorithm is less likely to overfit under 3 scenarios:

- By training the model on the random subset of the data.
- By creating many decision trees, each of which is again based on the random subset of the data.
- By averaging the results of multiple decision trees.

4.3.1 Parameters in Random Forest model:

- Criterion:* the supported criteria for the random forest are 'Gini', which is used for Gini impurity. And the other one is 'entropy' for the information gain. Definition of Gini impurity – the sum of the squared probabilities of each class. Definition of information gain – the decrease in entropy. In case of random forest, a decrease in entropy can be interpreted as increase in the purity of the node. In other words, the random forest algorithm will try to maximize the information gain at every node in the model.
- Random state:* The random state in random forest algorithm operates 2 randomized processes — one is bootstrapping: bootstrapping of the samples when creating decision tress and another one is - getting a random subset of features to search for the best feature during the node splitting process when creating each decision tree.
- n estimators:* The no. of decision trees in the random forest model. I defined with a tree list of 50, 100, 200, 300, 500.

4.3.2 Advantages & disadvantages of Random forest classifier:

Advantages:

- Random forests are more accurate than the decision tree algorithm because random forests reduce the variance, and, they are very less likely to overfit the model.
- This algorithm can also handle missing values and outliers better than decision trees algorithms.
- They are also easier to tune than decision trees and they are easy to tune with hyperparameter tuning.

Disadvantages:

- Random forest classifiers typically very slow to train. However, the accuracy and flexibility of this model make it worth the extra time investing in training the model.
- Sometimes, random forest models can be difficult to interpret.

CHAPTER 5: ANALYSIS OF ALL MODELS

5.1 CONVOLUTIONAL NEURAL NETWORK:

Before I start building the convolutional neural network (CNN) model, I imported the necessary the libraries for building the model. Numpy and pandas are basic libraries to import. I imported matplotlib for plotting the results of the convolutional neural network (CNN) model. And libraries like 'TensorFlow', 'Image', and 'os' are required for few operations in building the model. From "sklearn.model_selection", I imported a library called 'train_test-split' to split the data into training and testing. To convert all the labels into one-hot encoding, I need to_categorical library. After importing to_categorical library, I imported preprocessing library for performing the data preprocessing before I build or train the convolutional neural network (CNN) model. And then finally, I imported 'sequential' from keras.model to build the sequential convolutional neural network (CNN) model. Since we build the convolutional neural network (CNN) model with layers, I imported convolutional layers, max pooling layers, dense layers, flatten layers, and also dropouts.

At the next stage in building the convolutional neural network (CNN) model, I need to import or download the data from Kaggle platform to the google colab for working in python environment. For this, I need to download kaggle.json file. Using –"! pip install -q Kaggle", this means that the tool will allow me to install and manage additional libraries and also the dependencies that are not distributed as part of the standard library. And then imported files library for importing the files and upload them using - files.upload(). After that make the directory for the files to store in the google colab. And then finally unzipping the content and extracting them to the images and making them available to use for coding in the later part.

In the data preprocessing stage, first I set the variables for later use in the code – like data, labels, classes and current path. And then I retrieve the images and their labels. Resizing the images and then appending them is also done in this preprocessing stage. Now, at this point, am

converting all the lists to Numpy arrays and storing them in two variables namely – data and labels.

After doing some preprocessing steps, I checked the data shape and labels shape. And found them as - (39209, 30, 30, 3) (39209,). And then using the 'train_test_split' library, I split the data into training data set and testing data set by passing the test size argument as 0.2 by setting random state as 42. To cross verify the shape of the data and labels after splitting, I displayed the shape after split to make sure that am working on the data split. And then finally, I converted the labels to one hot encoding using to_categorical function.

Now, I build a CNN model which can classify the images into respective categories. The architecture of my model consists of 4 convolutional layers in total, with different kernel size and activation is ReLu. And also, few max pooling layers, flatten layers, drop outs, and dense layer as well. In the final dense layer, I used SoftMax activation function.

5.1.1 Compiling the model:

I compiled my CNN model using model.compile by passing the below arguments:

a. loss='categorical_crossentropy'

I used loss function in my CNN model to compute the quantity that my CNN model should be seeking to minimize during the training phase. The lower the loss, the better the model performs. Generally for classification problems, either categorical cross entropy is used or binary cross entropy is used depending on the problem statement.

When training a neural network for classification, Keras provides 3 different types of loss functions – binary cross entropy, categorical cross entropy and the third one is sparse categorical cross entropy. I used categorical cross entropy in my CNN model because this loss function is used for multi class classification models, in which there are two or more classes in output labels. I have 43 different classes of traffic sign boards.

b. optimizer='Adam'

I used Adam optimiser in my CNN model because it is computational powerful than the other optimizers like stochastic gradient descent since Adam optimizer is used instead of the classical standard stochastic gradient descent optimizer in order to update network weights. Adam optimizer is well suited for large data or parameters, in my case I have large data to train. And this very suitable for noisy gradients and require less tuning parameters. And well recommended for non-stationary objects like my traffic sign boards on roads.

c. metrics=['accuracy']

I used accuracy as my metrics in CNN model because the accuracy gives the good results when the data is balanced for all the classes. I don't have a perfect balanced data set because not all classes have equal number of images of traffic sign boards. But they are almost roughly the same (approx.. 300-400) on each class.

Accuracy is the metrics which describes how well the model is performing across all the classes. To better understand the accuracy metrics, I must understand the confusion matrix below:

		Actual Value	
		Positive	Negative
Predicted Value	Positive	TP (True Positive)	FP (False Positive)
	Negative	FN (False Negative)	TN (True Negative)

- True Positive (TP) : Observation is positive, and is predicted to be positive.
- False Negative (FN) : Observation is positive, but is predicted negative.
- True Negative (TN) : Observation is negative, and is predicted to be negative.
- False Positive (FP) : Observation is negative, but is predicted positive.

Fig 5: Confusion matrix

Source: <https://www.kdnuggets.com/2020/04/performance-evaluation-metrics-classification.html>

The ratio of true positive and true negative to all the predictions is the measure of accuracy. The formula is as below:

$$\text{Accuracy} = \frac{(TP + TN)}{(TP + FP + TN + FN)}$$

Fig 6: Accuracy formula

Source:<https://www.mydatamodels.com/learn/how-good-is-your-machine-learning-algorithm>

5.1.2 Training the model:

a. For 15 epochs

I chose to train the model for 15 epochs. Generally, to avoid the problem of overfitting and also to increase the generalization gap of the neural network model, it should be trained with reasonable or optimal no. of epochs. I checked the performance of the CNN model after each epoch while training it.

After training the model for certain epochs, I observed whether the model's performance and constantly checked whether it is overfitting or not. After experimenting with few epochs, I come up with an optimal number of 15 where my model might start overfitting after this optimal number.

b. Batch size of 64

The batch size is the number of training samples in one forward pass. I higher the number, the more memory I need.

5.2 SUPPORT VECTOR MACHINES (SVM)

In Support vector machines, the first few steps of importing the data from Kaggle to google colab and also data preprocessing steps are almost similar to the CNN model. After doing some preprocessing steps, I checked the data shape and labels shape. And found them as - (39209, 30, 30, 3) (39209,). And then using the library called 'train_test_split', I split the data into training and testing by passing the test size argument as 0.2 by setting random state as 42. To cross verify the shape of the data and labels after splitting, I displayed the shape after split to make sure that am working on the data split.

Resizing both the features of training data and test data. I normalized the data for working with SVM's using .scale function. And then I created a blank data frames to store the model scores.

I defined the SVM model with below arguments:

- a. nu=0.05,
- b. kernel='rbf',
- c. gamma=0.00001,
- d. random state = 121

⇒ And then I trained the model with .fit method by passing X_train and y_train arguments.

⇒ Using .predict method, I predicted the values on the test data.

⇒ I Imported the metrics from SKLEARN library for Calculating recall, precision, f1 score and accuracy of SVM.

⇒ And then finally I printed the classification report for SVM(more on this in results section) which clearly states the recall, precision, f1 score and accuracy of all 43 different traffic sign boards or categories.

5.3 RANDOM FOREST ALGORITHM:

For random forest model, I create a list of number of trees like – 50,100,200,300,500 to see the results for different results for different number of decision trees in this application.

I imported Random Forest Classifier from `sklearn.ensemble` and then created few variables like `y_pred_list` (Predicting values for test data), `time_rf_list` (Calculating time taken), and `rf_accuracy`(for calculating accuracy) to use them later in code.

But unfortunately, couldn't track the training speed for this model since it is taking ages to train the model. My google colab sessions are getting disconnected after a long wait of 3-4hrs. sadly, I couldn't bring the results for this random forest model although all my settings, parameters and arguments are given correctly.

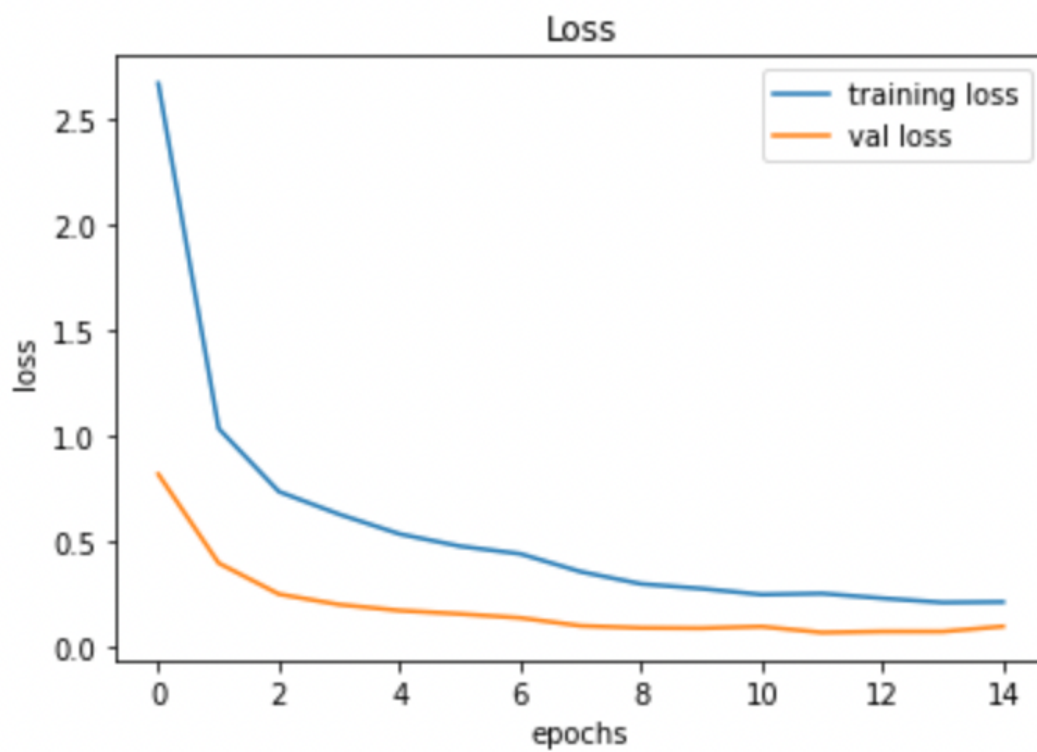
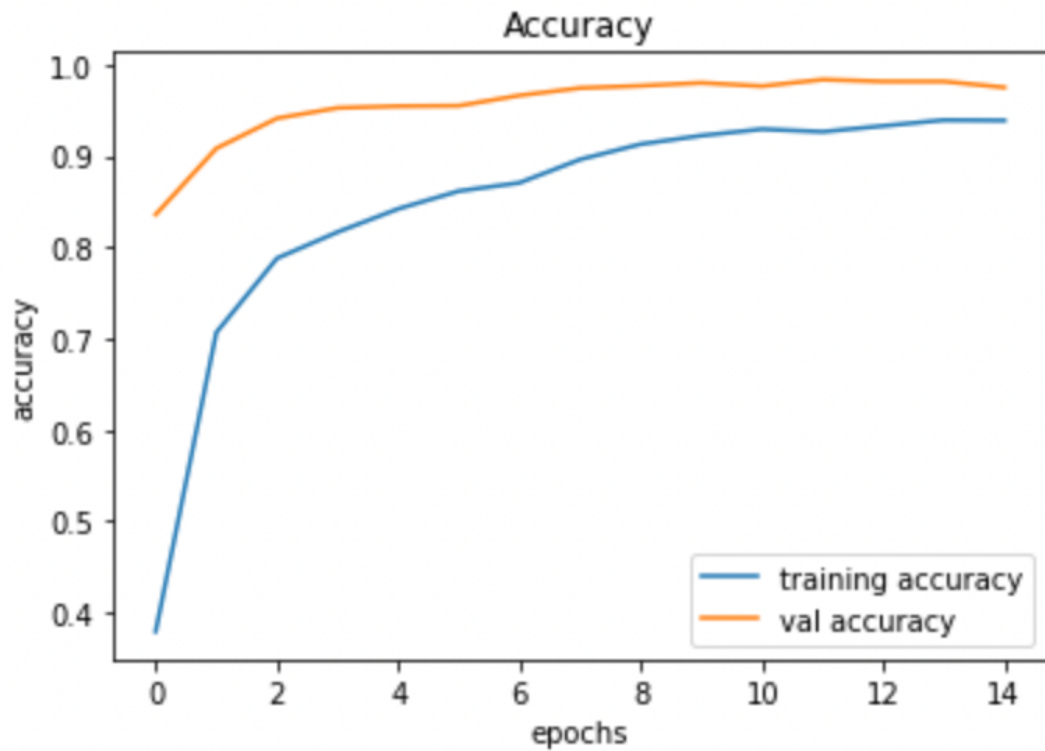
I have decided to compare all the three models – Convolutional neural networks (CNN), Support vector machines (SVM) and random forest, but unfortunately, I couldn't finish random forest algorithm and hence I can only compare 2 models now.

Training speed, testing speed and accuracy are the main important for any model that am working with. A model with good accuracy and reasonable training speed which can quickly work (high testing speed) is a good and reliable model for our traffic sign recognition system. But when a model which gives best results on the test data, and yet slow in terms of testing speed is not a good reliable model for our application.

Since am working on traffic sign board recognition system, I should have a working model which gives best testing speeds along with good training speed and high accuracy. So, in this case, I should either rely on SVM or CNN rather than non-working random forest model which is not successful in my project due to sessions disconnections because of high training speed.

CHAPTER 6: RESULTS

6. 1 CONVOLUTIONAL NEURAL NETWORKS:



One of the main things that I want to avoid is the problem of overfitting in the model. This overfitting happens when my model learns the training data well, but it fails to perform well and generalize and make accurate predictions for data that is unseen by the model. To find out whether the model is overfitting or not, I must follow a technique called cross validation, where I split the data into two parts – training set and validation set. The training data set is used for training the model and validation data set is used for evaluating the model's performance. That's the reason I calculated accuracy and loss on the training data and also on the validation data and plotted them in graphs above.

Models' accuracy and loss on training set and validation set are as follows:

- Training accuracy: ~95%
- Validation accuracy: ~99%
- Training loss: ~16%
- Validation loss: ~3%

Based on the validation accuracy, which is ~99%, I can expect my model to perform well and be ~99% accurate on a new unseen data.

The training loss indicates how well my CNN model is fitting the training data and on the other hand the validation loss indicates how well my model fits new unseen data. My aim is to make the validation loss as low as possible.

Results of CNN model:

- Training speed: 37 minutes
- Testing speed: 3 seconds
- Training accuracy: ~95%
- Validation accuracy: ~99%
- Accuracy on new data: ~95%
- Training loss: ~16%
- Validation loss: ~3%

6.2 SUPPORT VECTOR MACHIINES

Classification report of Support Vector Machines (SVM):

	precision	recall	f1-score	support
0	0.97	0.97	0.97	38
1	0.95	0.96	0.96	496
2	0.94	0.96	0.95	450
3	0.89	0.95	0.92	280
4	0.93	0.98	0.96	418
5	0.91	0.95	0.93	364
6	0.94	0.98	0.96	59
7	0.94	0.94	0.94	278
8	0.98	0.92	0.95	301
9	0.98	0.96	0.97	268
10	0.98	0.98	0.98	370
11	0.98	0.98	0.98	236
12	0.99	0.99	0.99	450
13	0.99	0.98	0.98	452
14	0.99	0.95	0.97	162
15	0.96	0.97	0.96	120
16	0.98	0.99	0.98	90
17	0.98	0.99	0.98	219
18	0.96	0.99	0.98	231
19	0.97	0.79	0.87	43
20	0.93	0.91	0.92	78
21	0.97	0.92	0.94	63
22	1.00	0.99	0.99	86
23	0.94	0.94	0.94	109
24	0.96	0.78	0.86	59
25	0.97	0.96	0.96	298
26	0.95	0.94	0.95	122
27	0.98	0.94	0.96	47
28	0.94	0.97	0.96	99
29	0.97	0.97	0.97	59
30	0.99	0.88	0.93	95
31	0.94	0.96	0.95	160
32	0.93	1.00	0.96	41
33	1.00	0.94	0.97	138
34	0.99	0.95	0.97	88
35	0.99	0.96	0.98	224
36	0.99	0.97	0.98	80
37	0.96	0.94	0.95	47
38	0.96	0.99	0.97	418
39	1.00	1.00	1.00	58
40	0.98	0.90	0.94	60
41	1.00	0.87	0.93	47
42	0.95	0.90	0.92	41
accuracy			0.96	7842
macro avg	0.96	0.95	0.96	7842
weighted avg	0.96	0.96	0.96	7842

The classification report presents the scores of precision, recall, F1, and support values for a given model.

There are four different ways to check if the correctness of predictions:

1. True Positive.
2. True Negative.
3. False Positive.
4. False Negative.

$$\begin{aligned}\text{Precision} &= \frac{\text{True Positive}}{\text{Actual Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \\ \text{Recall} &= \frac{\text{True Positive}}{\text{Predicted Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \\ \text{Accuracy} &= \frac{\text{True Positive} + \text{True Negative}}{\text{Total}}\end{aligned}$$

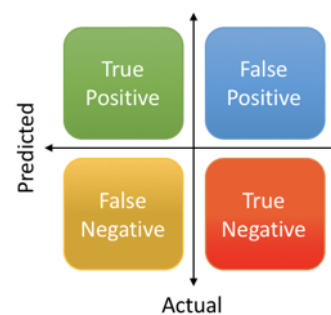


Fig 7: Precision, recall and accuracy

Source: <https://medium.com/@shrutisaxena0617/precision-vs-recall-386cf9f89488>

- *Precision*: describes how much percentage of my predictions are correct?
- *Recall* — describes how much percentage of the positive cases are correct?
- *F1 score* — F1 score is the weighted harmonic mean of precision and recall. Its range is between 0.0 and 1.0. One being the best score and zero being the worst score.

$$\text{F1 Score} = 2 * (\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision})$$

- *Support*: It simply gives the number of occurrences of each class in the data set.

→ Accuracy: Sum of True positives and true negatives to the total number of predictions.

Interpretation of SVM results:

Since the accuracy of SVM model is ~96%, which means the SVM model is able to predict any given unseen traffic sign board 96% accurately. Ideally any given model should be close 100% of accuracy, because it is very important for any self-driven car to rightly detect the traffic sign board to prevent the accident. For this right prediction I must take training speed (time taken by the model to learn the data) and testing speed (time taken by the model to predict on any unseen image of traffic sign board) into account.

- Training speed: 8 minutes
- Testing speed: 2 minutes
- Accuracy: ~96%

A model with less testing speed is dangerous because it is not ideal for a self-driving car to wait for 2 minutes to detect the traffic sign board. Hence it is better to rely on CNN model whose testing speed (~3 seconds) is less than the testing speed of SVM model (2 minutes). And also, CNN gives better accuracy than the SVM model. CNN accuracy is 99% and SVM accuracy is 96%.

6.3 RANDOM FOREST:

Runtime disconnected

Your runtime has been disconnected due to inactivity or reaching its maximum duration. [Learn more](#)

If you are interested in longer runtimes with more lenient timeouts, you may want to take a look at [Colab Pro](#).

Close

Reconnect

The training time for random forest model is forever. Session runtime is getting disconnected while training the model. Hence, I have no tracking of training speed, testing speed and accuracy for this model.

COMPARISION OF ALL THREE MODELS

MODEL NAME	TRAINING SPEED	TESTING SPEED	ACCURACY	PREFERRED?
Convolutional Neural Network	37 minutes	3 seconds	~95%	YES
Support Vector machine	8 minutes	2 minutes	~95%	NO
Random Forest Model	Forever	—	—	NO

CONCLUSION

In this paper, I developed a traffic sign board recognition system in automated cars using Convolutional Neural Network (CNN) and Support Vector Machines (SVM). The CNN model is performing well on new unseen data with an accuracy of 95% with training speed (time taken to train the model) of 37 minutes and testing speed (time takes to recognize new image) is 3 seconds. Whereas SVM model is giving a similar accuracy of 95-96% approx., with training speed of 8 minutes and testing speed of 2 minutes. Since the accuracies are almost the same for both CNN and SVM, but to choose one model over the other, the training speeds and testing speeds plays important role. I would definitely prefer CNN model over SVM model because of high testing speed (3 seconds). SVMs 2 minutes testing speed is not good suitable in traffic sign board recognition system as it might lead to an accident if signs board is recognized in 2 minutes. One of the major limitations in working with random forest is session disconnections since the data is very large. However, the CNN model is good enough, not perfect though, to use in real world application in real car. If I have to continue this project in the future, I would not only focus on improving my accuracy to close 100% but also will proceed to work on how this system can be implemented in countries that are poorly developed with only few traffic signs boards available on roads – will check what other alternatives can be used if enough sign boards are not available.

References:

Bulla, Premamayudu (2022) "Traffic Sign Detection and Recognition Based on Convolutional Neural Network", *International Journal on Recent and Innovation Trends in Computing and Communication*, 10(4), pp. 43–53. doi: 10.17762/ijritcc.v10i4.5533.

Data source (2011):

<https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign?datasetId=82373&searchQuery=svm2>

Fernando W. H. D. (2021), "Traffic Sign Recognition Using Convolutional Neural Network and Skipped Layer Architecture," 2021 IEEE 16th International Conference on Industrial and Information Systems (ICIIS), 2021, pp. 174-179, doi: 10.1109/ICIIS53135.2021.9660676.

Gyanchandani M. (2022), "Regularized CNN for Traffic Sign Recognition," 2022 International Conference on Smart Technologies and Systems for Next Generation Computing (ICSTSN), 2022, pp. 1-5, doi: 10.1109/ICSTSN53084.2022.9761341.

Keshav M, and team (2021), "Robust Drive Controlling System Based on Road Sign Detection for Low-Resolution Images," *2021 International Conference on Smart Generation Computing, communication and Networking (SMART GENCON)*, 2021, pp. 1-5, doi: 10.1109/SMARTGENCON51891.2021.9645878.

Lobo V. B. (2022), "A Real-Time Traffic Sign Detection and Recognition System on Hybrid Dataset using CNN," 2022 7th International Conference on Communication and Electronics Systems (ICCES), 2022, pp. 1354-1358, doi: 10.1109/ICCES54183.2022.9835954.

Nazmul Hasan and team (2020) 4th Int. Conf. Inventive Comms. and Comp. Tech., 'Traffic Sign Recognition System (TSRS): SVM and Convolutional Neural Network' pg.5

Samad A. and his team (2018), "A Transfer Learning based approach for Pakistani Traffic-sign Recognition; using ConvNets," 2018 International Conference on Computing, Electronic and Electrical Engineering (ICE Cube), 2018, pp. 1-6, doi: 10.1109/ICECUBE.2018.8610979.

Shah K. (2021), "Traffic Sign Board Recognition and Voice Alert System using Convolutional Neural Network," 2021 2nd International Conference for Emerging Technology (INCET), 2021, pp. 1-5, doi: 10.1109/INCET51464.2021.9456302.

Vinh T.Q. (2015), "Real-time traffic sign detection and recognition system based on friendlyARM Tiny4412 board," 2015 International Conference on Communications, Management and Telecommunications (ComManTel), 2015, pp. 142-146, doi: 10.1109/ComManTel.2015.7394276.

Zeniarja J. and her team (2016), "An unsupervised approach for traffic sign recognition based on bag-of-visual-words," 2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE), 2016, pp. 1-4, doi: 10.1109/ICITEED.2016.7863253.