**GERMAN TRAFFIC SIGNS CLASSIFICATION**

Contents

[1. PROBLEM STATEMENT AND SIGNIFICANCE 2](#_Toc119533415)

[2. DATA SOURCE 2](#_Toc119533416)

[3. METHODOLOGY 4](#_Toc119533417)

[4. MODELING 4](#_Toc119533418)

[5. RESULTS 10](#_Toc119533419)

[6. FUTURE SCOPE 10](#_Toc119533420)

# PROBLEM STATEMENT AND SIGNIFICANCE

As technology is advancing humans are relying more and more on machines to make decisions. Multinational companies like Google, Tesla, Uber, Ford, Mercedes-Benz, and many others are working on automating vehicles due to advancements in technology. self-driving cars are making real-time decisions based on data gathered from surroundings.

Traffic signs are an integral part of road infrastructure. Drivers must abide by a variety of traffic signs, including traffic signals, turn left or right, speed limits, no passing of heavy vehicles, no entrance, children crossing, etc. for a safe drive. They give drivers vital information, which forces them to modify their driving styles in order to comply with the rules of the road that are in effect at the moment. Every year, almost 1.3 million individuals die on roads. Without road signs, this figure would be significantly higher. Similarly, autonomous vehicles should follow traffic regulations, interpret, recognize, and comprehend traffic signs, and make decisions that are accurate and accident-free.

This project aims at training a deep convolutional neural network, neural network(MLP), and traditional machine learning models like SVM and random forest to classify traffic signs using the [German Traffic Sign Dataset](https://nam04.safelinks.protection.outlook.com/?url=http%3A%2F%2Fbenchmark.ini.rub.de%2F%3Fsection%3Dgtsrb%26subsection%3Ddataset&data=05%7C01%7Cbatchu500%40usf.edu%7Ce0afe792045846220f1608dac81efdbc%7C741bf7dee2e546df8d6782607df9deaa%7C0%7C0%7C638042332710013663%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=IDhen6XYBFXBzUheaL%2FS%2BFjS5RgALpgBVJnqyBf9owE%3D&reserved=0) so that the system using it can make the right decisions and maximize the safety of passengers and other participants on the road. The dataset has 43 different classes with unbalanced class frequencies. This dataset was developed for a competition in 2013. In the real world, there are more than 300 different traffic signs, and building a classifier on those many categories will need a lot of training data. In our case, we are only working with 43 categories.

# DATA SOURCE

The German Traffic Sign Benchmark is a 43 class dataset sourced from Kaggle. It consists of images from speed limit, Danger, Mandatory, Prohibitory, Derestriction and Unique subclasses of images. This dataset has 39209 train images and 12630 test images totaling 51839 images. Images consists of the following different traffic signs

Speed limit (20km/h), Speed limit (30km/h), Speed limit (50km/h), Speed limit (60km/h), Speed limit (70km/h), Speed limit (80km/h), End of speed limit (80km/h), Speed limit (100km/h), Speed limit (120km/h), No passing, No passing veh over 3.5 tons, Right-of-way at intersection, Priority road, Yield, Stop, No vehicles, Veh > 3.5 tons prohibited, No entry, General caution, Dangerous curve left, Dangerous curve right, Double curve, Bumpy road, Slippery road, Road narrows on the right, Road work, Traffic signals, Pedestrians, Children crossing, Bicycles crossing, Beware of ice/snow, Wild animals crossing, End speed + passing limits, Turn right ahead, Turn left ahead, Ahead only, Go straight or right, Go straight or left, Keep right, Keep left, Roundabout mandatory, End of no passing, End no passing veh > 3.5 tons.

Data Source – [GTSRB Dataset](https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign)

A screenshot of a game

Description automatically generated with medium confidence

**Fig: Images from dataset**

Chart, bar chart, histogram

Description automatically generated

**Fig: Class distribution plot**

# METHODOLOGY

Traditional models like Random Forest, Support Vector Machines and neural network models like Artificial Neural Networks, and Convolutional Neural Networks are used in this project. GridSearch and 10-fold cross-validation techniques are used for hyperparameter tuning of the models’ hyperparameters.

As hyperparameter tuning is a computation and time-intensive process we used 4k images for Random Forest and SVM and 1k images for ANN model. For Random Forest and SVM models, the features from raw pixel values of each image are generated using various filters like Gabor, Prewitt, Scharr, Gaussian, and Median. Each filter extracts certain information like edges, de-blurs the images, and reduces noise from the images.

Using the best hyperparameters obtained for each model type from the grid search technique, models are trained on train data, validated on validation data, and tested using test data.

**Evaluation Metrics:**

* Accuracies and confusion matrix are used for all models
* Precision, recall, f1-score for the best model – Convolutional Neural Networks

**Feature Engineering:**

As Random Forest and Support Vector Machines models cannot extract features from raw pixel values of each image the following filters were used to extract features for training these models

1. **Gabor**: In image classification, Gabor filters are used for edge detection, texture categorization, feature extraction, and disparity estimation.
2. **Scharr**: These filters are used to identify and highlight gradient edges/features
3. **Prewitt**: Prewitt operator is used for edge detection in an image
4. **Sobel**:  Sobel filter is used for edge detection
5. **Gaussian (sigma = 3 and 7)**: Gaussian Filter is a low pass filter used for reducing noise (high frequency components) and blurring regions of an image.
6. **Median**:  Median filter is the filtering technique used for noise removal from images and signals

These filters use different techniques to extract features like edges, de-blur, noise reduction, etc. from each image.

# MODELING

Support Vector Machines:

Using hyperparameter tuning following parameters were found as the best for SVM model

**C: 100**

**coef0: 0.5**

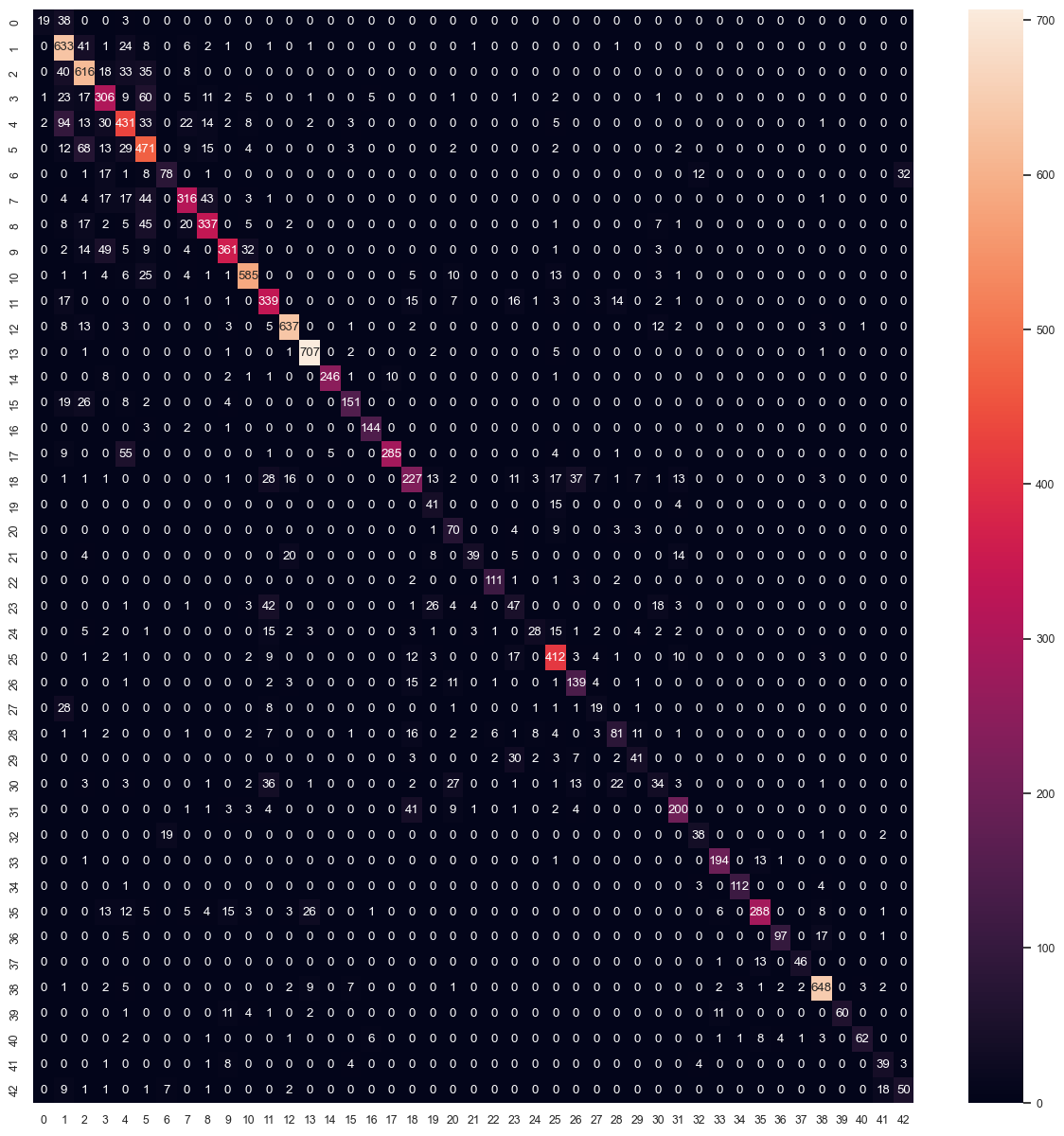
**degree: 3**

**kernel: rbf**

With computation and time constraint in view only 19k images were used for training the model and validated on 5.8k images. Also, the model was tested on 12k images test data. For all the datasets different features mentioned earlier were extracted. Accuracies obtained are

**Validation accuracy**: 97.33 %

**Test Accuracy**: 77.47%



**Fig: Confusion Matrix of SVM model’s output on test data**

Random Forest:

Using hyperparameter tuning following parameters were found as the best for Random Forest model

**max\_depth: 35**

**n\_estimators: 500**

With computation and time constraint in view only 19k images were used for training the model and validated on 5.8k images. Also, the model was tested on 12k images test data. For all the datasets different features mentioned earlier were extracted. Accuracies obtained are

**Validation accuracy**: 97.44 %

**Test Accuracy**: 77.71%

A picture containing text, outdoor, light, plant

Description automatically generated

**Fig: Confusion Matrix of Random Forest model’s output on Test data**

Neural Networks using Keras:

Architecture:

* Input Layer
* Dense fully connected layer (128 neurons, activation = ‘relu’)
* Dropout layer (rate = 0.2)
* Dense layer (43 nodes, activation= ‘softmax’)

Using hyperparameter tuning the following parameters were found as the best for Neural Nets model.

**batch\_size: 36**

**epochs: 500**

**optimizer: sgd**

Using the 39k images dataset 10% of the data was held out for validation and the model was trained. Then the model was tested on a test set of 12k images.

**Validation accuracy**: 95.72 %

**Test Accuracy**: 85.68 %

A picture containing text, outdoor, red, light

Description automatically generated

**Fig: Confusion Matrix of Random Forest model’s output on Test data**

Convolutional Neural Networks:

* Input layer
* 2 Conv2D layers (filter=32, kernel\_size=(5,5), activation= ”relu”)
* MaxPool2D layer ( pool\_size=(2,2))
* Dropout layer (rate = 0.25)
* 2 Conv2D layers (filter=64, kernel\_size = (3,3), activation = ”relu”)
* MaxPool2D layer ( pool\_size = (2,2))
* Dropout layer (rate = 0.25)
* Flatten layer to squeeze the layers into 1 dimension
* Dense Fully connected layer (256 nodes, activation=”relu”)
* Dropout layer (rate=0.5)
* Dense layer (43 nodes, activation=”softmax”)

Following hyper parameters were used

batch\_size = 32

epochs = 30

Using the 39k images dataset 10% of the data was held out for validation and the model was trained. Then the model was tested on test set of 12k images.

**Validation accuracy**: 99.72 %

**Test Accuracy**: 96.36 %

A picture containing table

Description automatically generatedA picture containing table

Description automatically generated

From the above precision and recall output for the CNN model on the test data we can see that for the images of class 21 and 27 we got less precision. As maximum precision is needed for the applications of this problem the precision of low precise classes can be improved by some of the methods like Data augmentation, overcoming the problem of class imbalance using under sampling or over sampling techniques etc.

A picture containing text, outdoor, light, red

Description automatically generated  
**Fig: Confusion matrix for CNN Model**

# RESULTS

|  |  |  |
| --- | --- | --- |
| **Model** | **Validation Accuracy (%)** | **Test Accuracy (%)** |
| Support Vector Machines | 97.33 | 77.47 |
| Random Forest | 97.44 | 77.71 |
| Neural Networks | 95.72 | 85.68 |
| Convolutional Neural Networks | 99.72 | 96.36 |

We can see that SVM, Random Forest and Neural network models have high validation accuracy but they failed on the test dataset implying that the models have overfitted the train data. As the CNN model has highest accuracy and also generalizes well on the test data, we are considering this as the best model.

# FUTURE SCOPE

For future work, we would like to consider below steps:

1. Perform data augmentation to address the imbalanced classes problem
2. Utilize layers of the current CNN model trained with RGB channels and apply transfer learning with gray scaling to recognize photos with variations in lighting and nighttime conditions.
3. Use autoencoders to further interpret traffic signs partially hidden, broken, damaged, etc.
4. From the application point of future scope this project can be applied in vehicles where the device with this model will detect traffic signs and caution the driver (probably vibration feedback through the steering wheel) when the driver is driving rashly or beyond the limits.