Traffic Sign Classification Using CNN

[1]Ms.KSrilaxmi ,[2]B.R.Ashrith , [3]Moveena Niteshini ,[4]Rohith Dabla

[1]Assistant Professor in Department of IT, Sreenidhi Institute of Science and Technology

kslaxmi@sreenidhi.edu.in

[2,3,4]UG scholar In Department of IT, Sreenidhi Institute of Science and Technology.

18311a12c7@sreenidhi.edu.in

18311a12d0@sreenidhi.edu.in

18311a12q3@sreenidhi.edu.in

Abstract

This project presents an effective solution to detecting traffic signs on road by first classifying the traffic sign images using Convolutional Neural Network (CNN) on the German Traffic Sign Recognition Benchmark (GTSRB)and then detecting the images of Indian Traffic Signs using the Indian Dataset which will be used as testing dataset while building classification model. Therefore this system helps electric cars or self driving cars to recognize the traffic signs efficiently and correctly. The system involves two parts, detection of traffic signs from the environment and classification based on CNN thereby recognizing the traffic sign. The classification involves building a CNN model of different filters of dimensions 3 × 3, 5 × 5, 9 × 9, 13 × 13, 15 × 15,19 × 19, 23 × 23, 25 × 25 and 31 ×31 from which the most efficient filter is chosen for further classifying the image detected.

1. INTRODUCTION

Humans are getting increasingly reliant on technology in this age of Artificial Intelligence. Multinational corporations such as Google, Tesla, Uber, Ford, Audi, Toyota, Mercedes- Benz, and others are working on automating automobiles thanks to advancements in technology. They're attempting to develop more precise autonomous or self-driving automobiles. You may have heard about self-driving vehicles, in which the vehicle acts like a driver and does not require human intervention to operate on the road. It's not unreasonable to consider the safety implications—the possibility of serious machine mishaps. However, no machine is more precise than humans. Many algorithms are being tested by researchers in order to assure complete road safety and accuracy. When driving on the road, you will encounter numerous traffic signs such as traffic lights, turn left or right, speed restrictions, no passing of heavy vehicles, no entering, children crossing, and so on, which you must obey in order to drive safely.

In order to reach accuracy, autonomous cars must also analyse these indicators and make judgments. Traffic signs categorization is the process of determining which class a traffic sign belongs to. We will use a convolutional neural network (CNN) and the Keras library to create a model for classifying traffic signs in an image into many categories in this Deep Learning project.

1.1 Scope

The ultimate goal is to create a system that can be used to catalogue traffic signs. By automatically recognising and categorising one or more traffic signs from a complicated image acquired by a vehicle's camera, this technology can aid local or national authorities in the work of maintaining and upgrading its road and traffic signs. The key approach is to identify the perfect colour combination in the scene so that one colour is inside the convex hull of another colour, and then combine that with the appropriate form. If a contender is identified, the system attempts to identify the item using the rim pictogram combination and returns the classification result. The following are the goals: To get a better understanding of the attributes of road and traffic signs and how they affect image processing for the recognition challenge. To grasp the concepts of colour, colour spaces, and colour space conversion. To provide reliable colour segmentation algorithms that can be employed in a variety of environments. Using invariant shape measurements, create a recognizer that is invariant to in-plane transformations such as translation, rotation, and scaling. To determine the best method for extracting features from traffic signs. To create a traffic sign categorization algorithm that is adequate. To assess the robustness of the aforementioned approaches under a variety of weather, illumination geometry, and sign situations.

2 Existing System

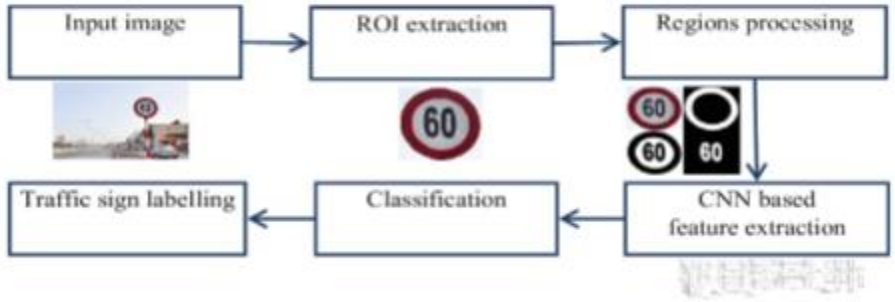
• To solve the target recognition problem, most of the previous works use hand-crafted feature extracting methods like Scale-Invariant Feature Transform (SIFT) , Histogram of Oriented Gradient (HOG). Those conventional feature extraction methods, over-reliant on the designer’s experience, meet some restriction in feature expression. On the contrary, the deep network model based on CNN is more powerful in feature expression. CNN method, which possessing the characteristics of rotation, translation and scaling invariance, is able to realize weight sharing through local receptive fields. It’s widely used in the sub-area of target recognition e.g., image classification, face recognition and pedestrian detection.

3. Proposed System

In this project we will make two contributions; one is to develop a new dataset for Traffic and Road Signs and the other is develop and design a deep CNN architecture for Traffic sign recognition. The collected data set is given as an input to the proposed CNN architecture for training, validation and testing. The detailed explanation of the CNN architecture is provided in the next slide. Once the CNN is trained, it is ready to be used for classifying new images which were not part of the collected dataset. A Deep CNN architecture is also proposed for traffic sign

recognition Generally, CNNs consist of multiple hidden layers between the input and output layers. The design of the proposed CNN is implemented using Python.

4. Architectural Design



construct a deep CNN architecture for traffic sign identification. The high-level perspective of the system is shown in For training, validation, and testing, the acquired data set is fed into the suggested CNN architecture. In the next part, we'll go through the CNN architecture in depth. Once the CNN has been trained, it may be used to categorise fresh pictures that were not included in the dataset. This is the first time a comprehensive database on Traffic has been created. For traffic sign identification, a Deep CNN architecture is also proposed. Between the input and output layers, CNNs often include numerous hidden layers. Python is used to implement the suggested CNN's design.

5. Implementation

1.Os:

This module allows you to use operating system-dependent functions on the go. If you merely want to read or write a file, use open(), os.pathmodule if you want to alter paths, and fileinput if you want to read all the lines in all the files on the command line. See the tempfile module for producing temporary files and directories, and the shuttle module for managing high-level files and directories.

2.numpy:

NumPy is a Python library used for working with arrays. It also has functions for working in the domain of linear algebra, fourier transform, and matrices.

3. keras.models.load_model():

It is a method in keras model library to load the weights of the saved model.

6. OUTPUT SCREENS

Output Screens of various functionalities in our application are shown over here, along with the description.

```
Epoch 1/15
696/696 [==============================] - 122s 176ms/step - loss: 1.4509 - accuracy: 0.5940 - val_loss: 0.3094 - val_accuracy: 0.9139
Epoch 2/15
696/696 [==============================] - 123s 176ms/step - loss: 0.4240 - accuracy: 0.8696 - val_loss: 0.1284 - val_accuracy: 0.9616
Epoch 3/15
696/696 [==============================] - 121s 174ms/step - loss: 0.2894 - accuracy: 0.9120 - val_loss: 0.1081 - val_accuracy: 0.9670
Epoch 4/15
696/696 [==============================] - 117s 169ms/step - loss: 0.2205 - accuracy: 0.9312 - val_loss: 0.0825 - val_accuracy: 0.9749
Epoch 5/15
696/696 [==============================] - 122s 175ms/step - loss: 0.1905 - accuracy: 0.9384 - val_loss: 0.0536 - val_accuracy: 0.9859
Epoch 6/15
696/696 [==============================] - 121s 174ms/step - loss: 0.1657 - accuracy: 0.9489 - val_loss: 0.0675 - val_accuracy: 0.9845
Epoch 7/15
696/696 [==============================] - 119s 171ms/step - loss: 0.1581 - accuracy: 0.9509 - val_loss: 0.0559 - val_accuracy: 0.9842
Epoch 8/15
696/696 [==============================] - 119s 171ms/step - loss: 0.1367 - accuracy: 0.9564 - val_loss: 0.0457 - val_accuracy: 0.9865
Epoch 9/15
696/696 [==============================] - 119s 171ms/step - loss: 0.1152 - accuracy: 0.9626 - val_loss: 0.0392 - val_accuracy: 0.9894
Epoch 10/15
696/696 [==============================] - 117s 168ms/step - loss: 0.1172 - accuracy: 0.9635 - val_loss: 0.0359 - val_accuracy: 0.9904
Epoch 11/15
696/696 [==============================] - 119s 171ms/step - loss: 0.1056 - accuracy: 0.9657 - val_loss: 0.0308 - val_accuracy: 0.9907
Epoch 12/15
696/696 [==============================] - 117s 169ms/step - loss: 0.0968 - accuracy: 0.9692 - val_loss: 0.0348 - val_accuracy: 0.9908
Epoch 13/15
696/696 [==============================] - 118s 169ms/step - loss: 0.0916 - accuracy: 0.9726 - val_loss: 0.0399 - val_accuracy: 0.9907
Epoch 14/15
696/696 [==============================] - 120s 172ms/step - loss: 0.0860 - accuracy: 0.9726 - val_loss: 0.0300 - val_accuracy: 0.9925
Epoch 15/15
696/696 [==============================] - 122s 175ms/step - loss: 0.0848 - accuracy: 0.9743 - val_loss: 0.0329 - val_accuracy: 0.9914
```

Fig 6.1 Accuracy and Validation accuracy for our model

The above figure shows the accuracy and validation accuracy of our model. An epoch means training the neural network with all the training data for one cycle. In an epoch, we use all of the data exactly once. A forward pass and a backward pass together are counted as one pass: An epoch is made up of one or more batches, where we use a part of the dataset to train the neural network. Our model has an accuracy of 97.43%, and validation accuracy is 99.14%

PLOT THE TRAINING ACCURACY AND VALIDATION ACCURACY

```
In [21]: plt.plot(history.history['accuracy'],'r-*',label='training accuracy')
         plt.plot(history.history['val_accuracy'],label='validation accuracy')
         plt.xlabel('epochs')
         plt.ylabel('loss')
         plt.legend()
         plt.show()
```

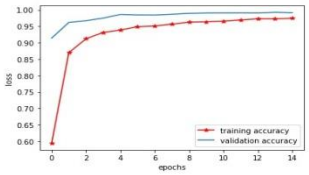


Fig 6.2 Plot of training and validation accuracy

From the curves of training and validation accuracy, we can see that the model performs well on the training data compared to the validation data. With this, we can proceed with the project.

PLOT THE TRAINING LOSS AND VALIDATION LOSS

```
In [20]: plt.plot(history.history['loss'],'r-*',label='training loss')
         plt.plot(history.history['val_loss'],label='validation loss')
         plt.xlabel('epochs')
         plt.ylabel('loss')
         plt.legend()
         plt.show()
```

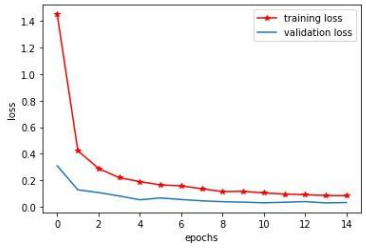


Fig 6.3 Plot of Training and Validation loss

From the curves of training and validation loss, we can see that validation loss is greater than the training loss.

From the previous two plots, we can say that the system is trained on the training data and tested on the validation data, which it has not seen during the training; it's new for the model and hence the reason for this behavior.

With these results, we can proceed with using the model in our application. If the results were not satisfactory, we would have to perform hyperparameter tuning on our model. In this case, it is not required.

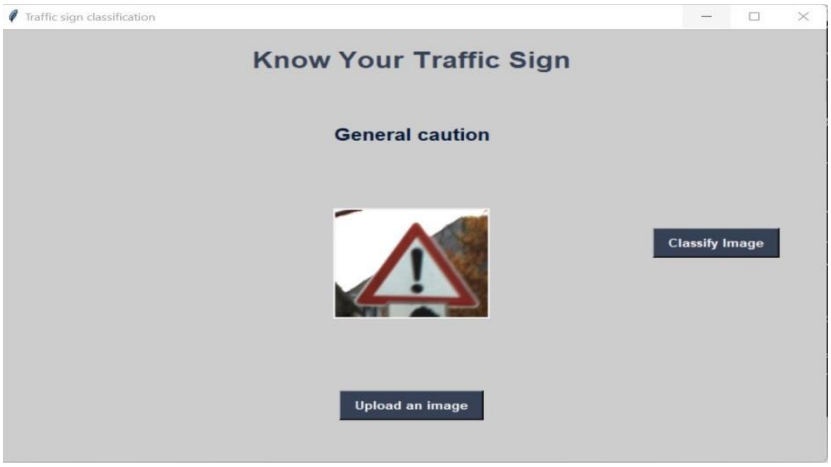
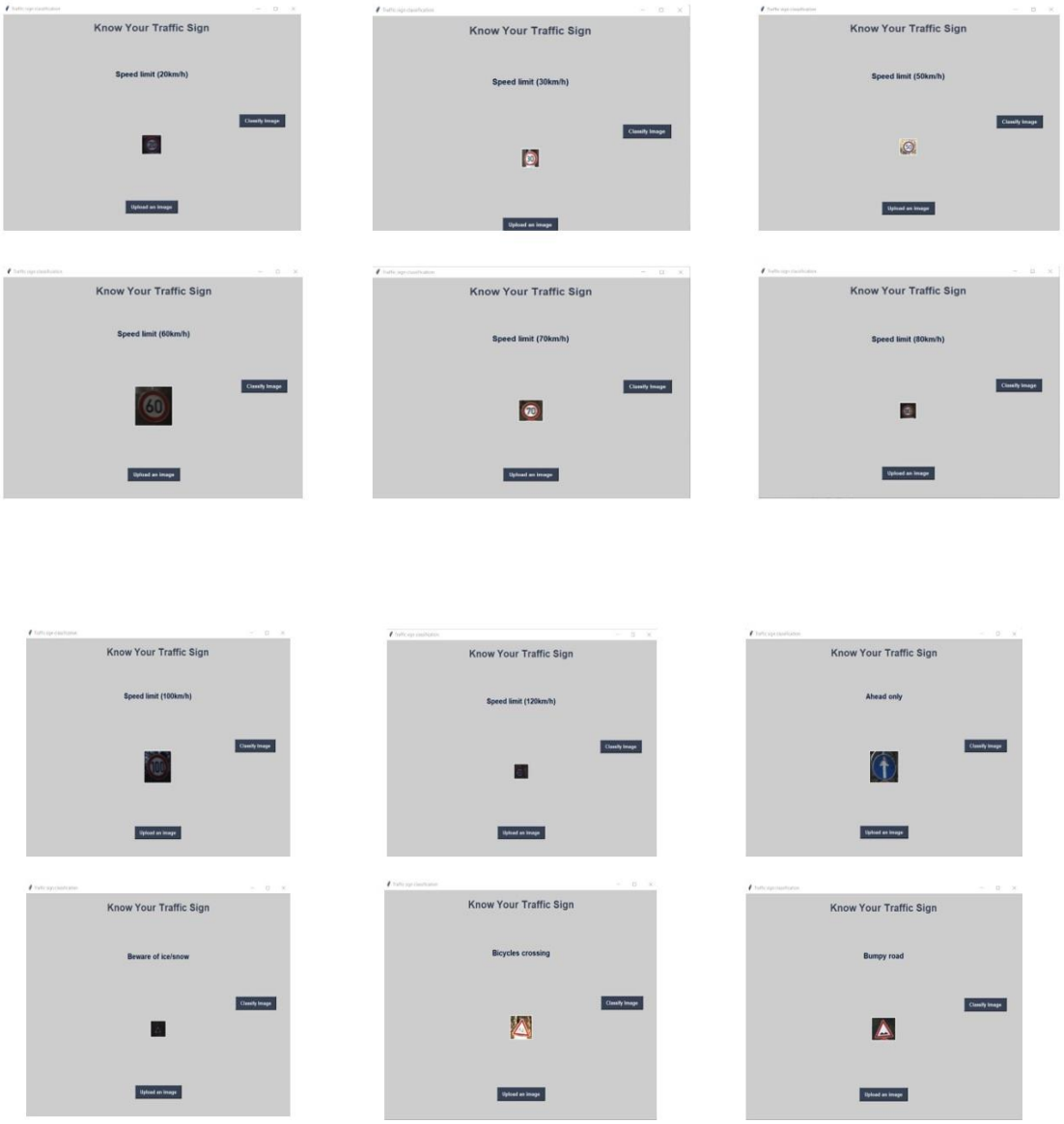


Fig 6.4 Traffic Sign Classification using GUI

In this case, the user gives the application permission to upload an image and classify image. The image is preprocessed and given as input to the model we have created. The model classifies the traffic sign. Based on the classification, the appropriate result is displayed on the screen. For example: If we upload an image and classify image. It displayed “General Caution” on the screen.




Know Your Traffic Sign
Children crossing
Classify Image
Upload an Image


Know Your Traffic Sign
Dangerous curve left
Classify Image
Upload an Image


Know Your Traffic Sign
Dangerous curve right
Classify Image
Upload an Image


Know Your Traffic Sign
Double curve
Classify Image
Upload an Image


Know Your Traffic Sign
End of no passing
Classify Image
Upload an Image


Know Your Traffic Sign
End of speed limit (80km/h)
Classify Image
Upload an Image


Know Your Traffic Sign
End speed + passing limits
Classify Image
Upload an Image


Know Your Traffic Sign
General caution
Classify Image
Upload an Image


Know Your Traffic Sign
Go straight or left
Classify Image
Upload an Image


Know Your Traffic Sign
Go straight or right
Classify Image
Upload an Image


Know Your Traffic Sign
Keep left
Classify Image
Upload an Image


Know Your Traffic Sign
Keep right
Classify Image
Upload an Image


Know Your Traffic Sign
No entry
Classify Image
Upload an Image


Know Your Traffic Sign
No passing veh over 3.5 tons
Classify Image
Upload an Image


Know Your Traffic Sign
No passing
Classify Image
Upload an Image


Know Your Traffic Sign
No vehicles
Classify Image
Upload an Image


Know Your Traffic Sign
Pedestrians
Classify Image
Upload an Image


Know Your Traffic Sign
Priority road
Classify Image
Upload an Image


Know Your Traffic Sign
Right-of-way at intersection
Classify Image
Upload an Image


Know Your Traffic Sign
Road narrows on the right
Classify Image
Upload an Image


Know Your Traffic Sign
Road work
Classify Image
Upload an Image


Know Your Traffic Sign
Roundabout mandatory
Classify Image
Upload an Image


Know Your Traffic Sign
Slippery road
Classify Image
Upload an Image

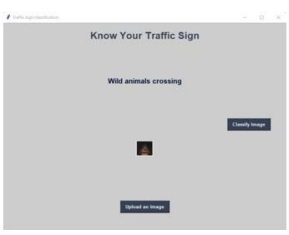

Know Your Traffic Sign
Stop
STOP
Classify Image
Upload an Image


Know Your Traffic Sign
Traffic signals
Classify Image
Upload an Image


Know Your Traffic Sign
Turn left ahead
Classify Image
Upload an Image


Know Your Traffic Sign
Turn right ahead
Classify Image
Upload an Image


Know Your Traffic Sign
Veh > 3.5 tons prohibited
Classify Image
Upload an Image


Know Your Traffic Sign
Wild animals crossing
Classify Image
Upload an Image


Know Your Traffic Sign
Yield
Classify Image
Upload an Image

7. CONCLUSION

A method is proposed that combines colour transformation and Convolutional Neural Networks (CNN). The CNN with fixed and learnable layers achieved good results when working on the image preprocessed by colour transformation. The CNN we used has the following advantages: For starters, a fixed layer can limit the amount of locations that the classifier must deal with, thereby speeding up detection. Second, because the ROIs formed by the fixed filter are extremely close to the borders of traffic signs, the alignment problem is avoided; otherwise, the supervised convolution network's performance would suffer. Third, it has been demonstrated that a CNN with a proper architecture taught in a supervised manner may extract features for traffic sign classification.The outcomes of our trial backed up our conclusion. The best results obtained for training the dataset with the modified CNN 8-layers model were 97.43% accuracy.

8. Future Scope

The traffic signs focus on reduction of traffic load on existing road network through various travel demand management measures. Traffic signs should remove encroachment, congestion, and improve the traffic signal. Road signs notify road user of regulation and provide warning and guidance needed for safe, uniform and efficient operation.

9.BIBLIOGRAPHY

[1] Keras Documentation: https://keras.io/guides/

[2] Research Paper: https://ieeexplore.ieee.org/document/7859723

[3] Dataset: https://www.kaggle.com/valentynsichkar/traffic-signs-classification-with-cnn

[4]https://wwwresearchgate.net/publication/352393724_Traffic_Sign_Classificati

on_and_Detection_of_Indian_Traffic_Signs_using_Deep_Learning