TRAFFIC SIGNS CLASSIFICATION USING CNN

PRESENTED BY

ROHIT KADLAG , SANSKRUTI JANA, SAHIL BODKE, MAYUR GAYKAR

INTRODUCTION

The project is a real-time application which focuses on analyzing traffic signs using CNN(convolution neural network) and we also created a WebApp using Flask. Here the project shows how machine learnig plays a vital part in classifying real world application. We are working on dataset named 'German Traffic Sign Recognition Benchmark' which consist of plenty of traffic sign images along with their features(height,size,img path etc) we are using different approaches to handle and analyze each image.

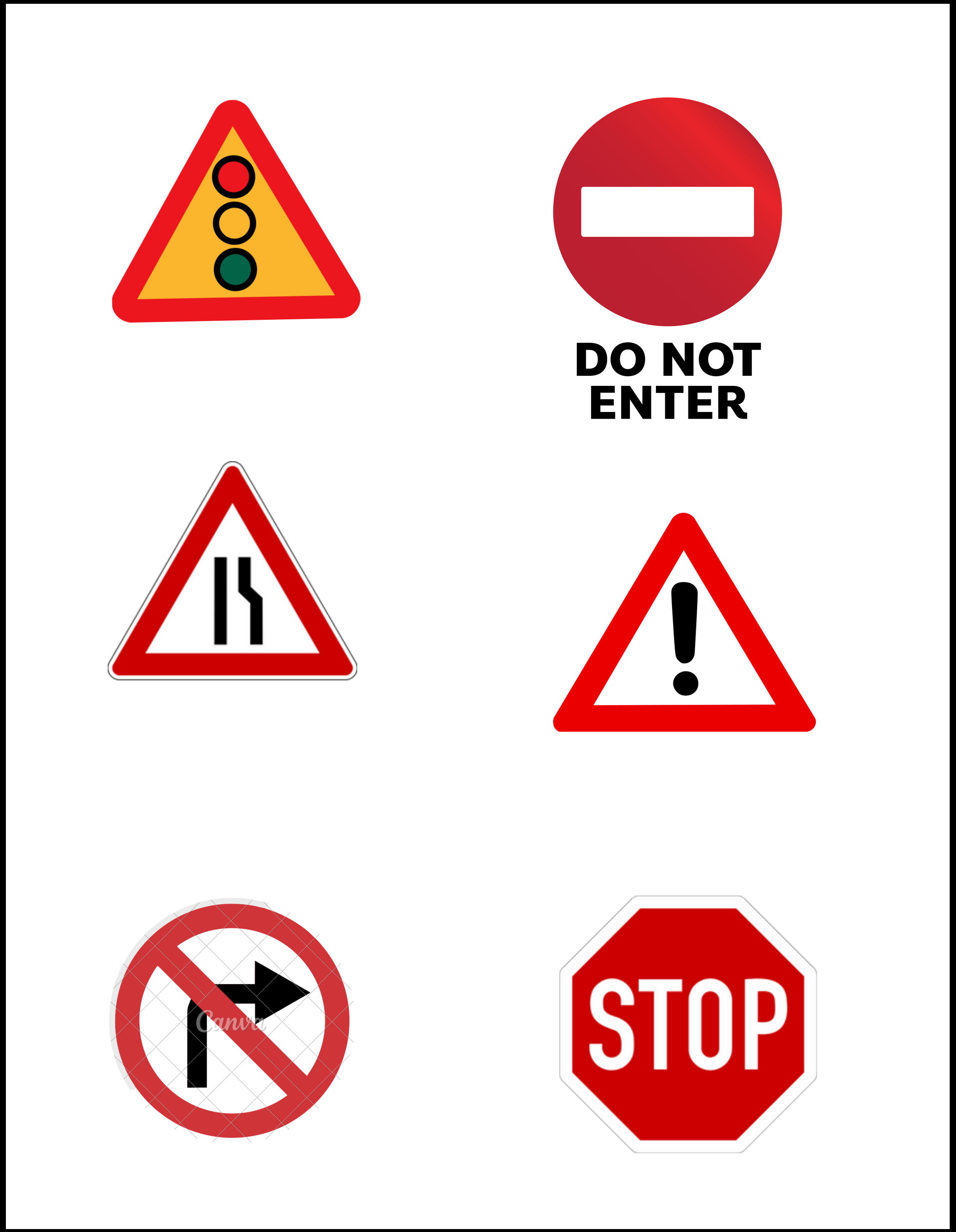
CNN is a type of artificial neural network used primarily for image recognition and processing, due to its ability to recognize patterns in images.

FLASK WEBAPP

Flask is a micro web framework written in Python. We using Flask webapp to detack the different types of class or traffic signs along with using CNN

Keras
Libraries Used

- CONVO2D: This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. If use bias is True, a bias vector is created and added to the outputs. Finally, if activation is not None, it is applied to the outputs as well.
- MAX POOLING: Max pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter. Thus, the output after max-pooling layer would be a feature map containing the most prominent features of the previous feature map.
- FLATTEN: Flatten is used to flatten the input. For example, if flatten is applied to layer having input shape as (batch_size, 2,2), then the output shape of the layer will be (batch_size, 4).
- DENSE : Keras Dense layer is the layer that contains all the neurons that are deeply connected within themselves. This means that every neuron in the dense layer takes the input from all the other neurons of the previous layer. We can add as many dense layers as required.
- DROPOUT: The Dropout layer is a mask that nullifies the contribution of some neurons towards the next layer and leaves unmodified all others.



CONCLUSION

- As we are working on training and testing data with two different test and pred variables X and Y respectively by using sklearn module we are getting accuracy of 0.87 which is highly accurate.
- Tsr.h5 file which we created makes the execution process easy as the file already contains the reports which make easy while processing the data at moment