PYTHON PROJECT

Problem Statement:

In the world there are several traffic signs such as turn left, stop, bicycle crossing etc. So in this project we are going to deal with the Classification of Traffic Signal Signs present in the image into different categories using Tensor-flow and keras and placing them accordingly. After which a GUI is built for the classification of the traffic signs using Tkinter.

Dataset:

The dataset being used in this project is the publicly available dataset present in kaggle

link: https://www.kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign

The Dataset contains 39,209 images of various traffic related signs which is classified into 43 i.e. 0 to 42 different classes. The dataset is not quite satisfying as some classes have more images while some have only less. There are 2 main folders which are Train and Test. The Train folder is used to train the model with the 43 classes of images and the Test folder is used to test out the model.

Explanation:

There are 5 *Main* steps in this project:

- Exploration of the Dataset
- Classification of Training data and testing data
- Building the Model
- Training and Testing out the Model
- Testing the model with the Test Dataset

Then we classify the different sign using a GUI using Tkinter.

1. Exploration of the Dataset:

The dataset consists of 43 Different folders which contain different classes of the traffic signals i.e. 0 to 42. Using the OS module we iterate over each and every classes and append the images with its respective labels in the labels list and the data list which is done by the PILLOW module (PIL), to convert the image into an array.

After storing all the images and labels into the data and labels list, the lists are fed into the model after converting the list into the numpy array. The shape of the train data is (29406, 30, 30, 3) which represents the 39,209 images of 30×30 image pixels. The 3 represents the RGB value. The test data is (9803,30,30,3).

2. Classification of Train and Test:

Using the train_test_split from sklearn.model_selection, we split the training and testing data at a 75:25 ratio with a random state of 43. Then we use the categorical method from 'keras.utils' to convert the data or labels present in the y_train and y_test into ONE HOT ENCODING.

3. Building the Model:

For the classification of the traffic sign images we use Convolutional Neural Network Model (CNN) as CNN is used in image classification purposes.
The architecture of the model is:

- Convolution Layer of filter=32, kernel_size=5x5, Rectified linear unit activation and again a 2D convolutional layer of filter=64, kernel_size=3x3, Rectified linear unit activation
- Max pooling operation for spatial data of – pool size=2×2
- Dropout layer to prevent over-fitting of data of rate 0.25
- Flatten layer to convert the n Dimensional layers into 1 dimension

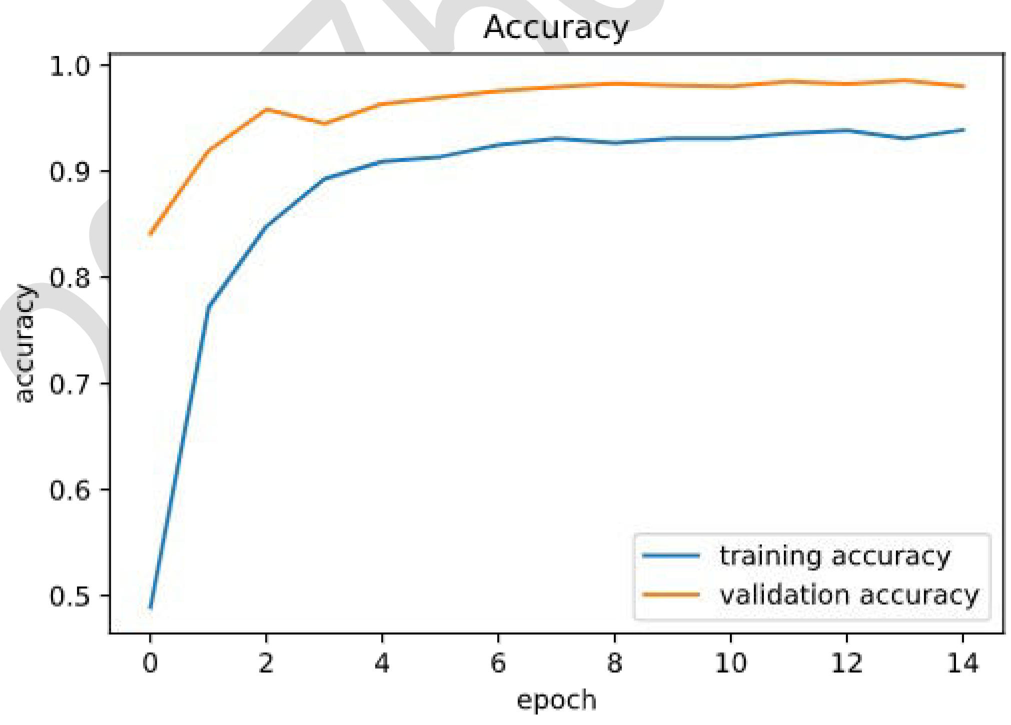
- Dense Fully connected layer of a total of 256 nodes and activation is RELU
- a dropout layer to avoid over fitting with a rate of 0.5
- Again a dense layer of 43 with an activation of soft-max activation for the output layers as there are total classes of 0 to 42 -> 43

Then we compile the model with ADAM optimizer and the loss used is categorical cross entropy as we have multiple data's to be classified.

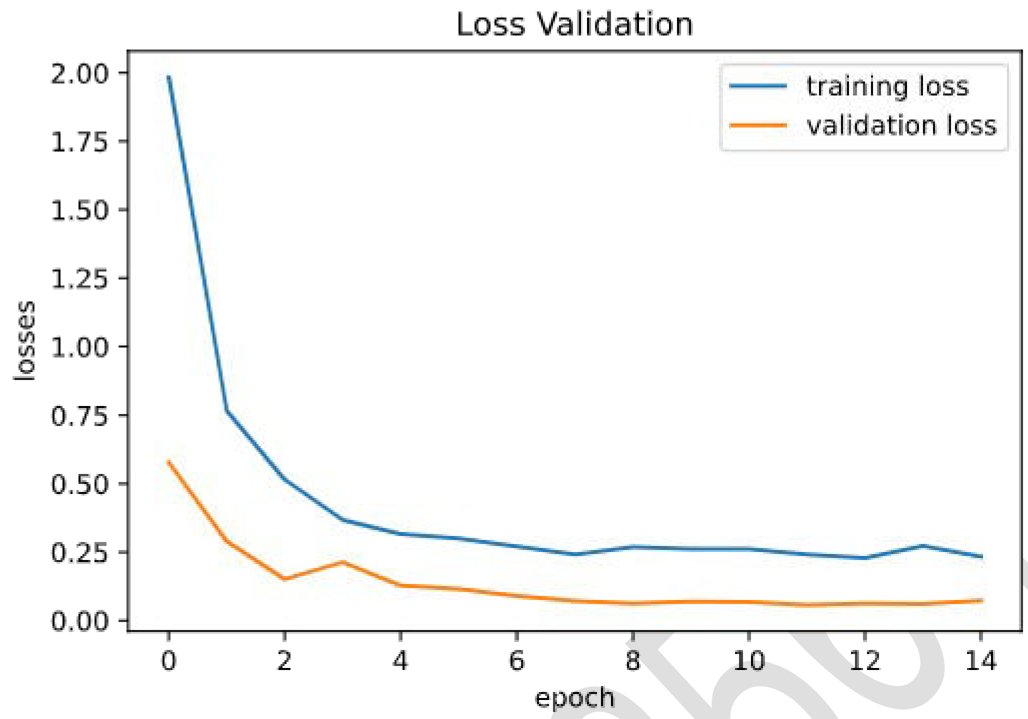
4.Training and Testing:

The model is trained using model.fit() with a batch-size of 64 and epochs = 15. The model was stabilized after 15 epochs and the model got an accuracy of 94%.

Using Matplotlib and Seaborn the training accuracy and validation accuracy is plotted.



Training Accuracy and Validation Accuracy



Training Loss and Validation Loss

5.*Testing the Model:*

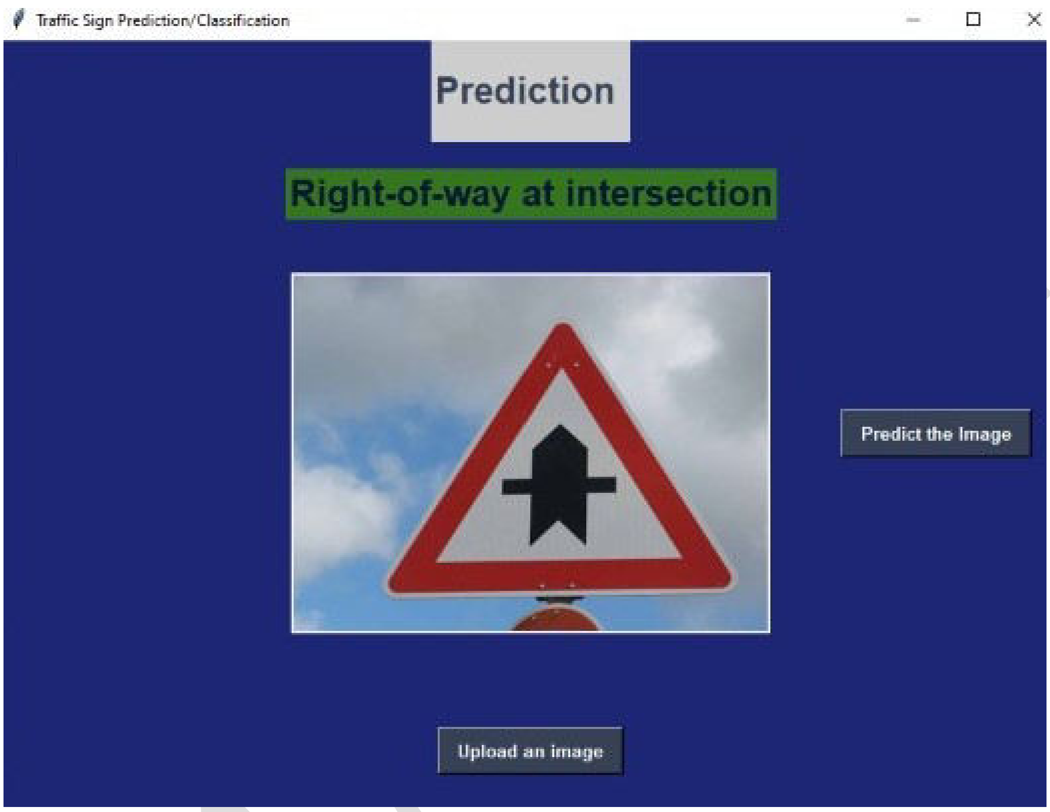
The *Test.csv* contains the details related to the image, their path and the respective class labels. The image path and the labels are extracted using pandas. Then we resize the images into 30x30 pixels and convert the image data into a numpy array.

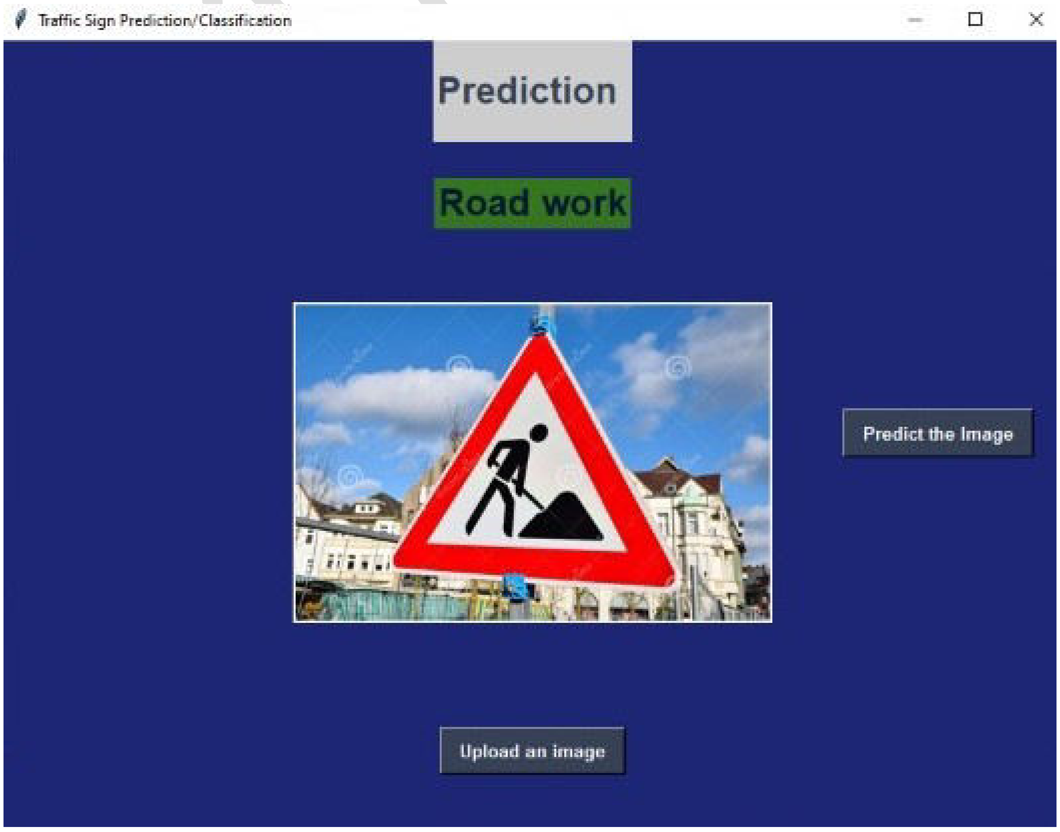
Using accuracy_score from sklearn.metrics the accuracy of the model is predicted and the accuracy is 94.4% and then the model is saved to use it in the GUI using *model.save()*.

TRAFFIC SIGN GUI:

The GUI for the Traffic Sign classification/prediction is built using Tkinter. The model is loaded using Keras. The GUI is for uploading the image and for the classification of the images. The Classify function is for the conversion of the image into the shape (a, b, c, d) where a is the image, b and c are the pixels i.e. 30x30 and d is the RGB settings. We convert it into this format as we have trained the image of this format. The prediction returns a number from 0 to 42 which represents the class it belongs to using the dictionary.

Outcome:


Traffic Sign Prediction/Classification
Prediction
Right-of-way at intersection
Predict the Image
Upload an image


Traffic Sign Prediction/Classification
Prediction
Road work
Predict the Image
Upload an image