TRAFFIC SIGN CLASSIFICATION

ISM 6251 Data Science Programming

Prof. Balaji Padmanabhan

Team members:

Mandeep Singh

Neti Sheth

Pranjal Shrivastava

Pranali Kanade

Index

1. INTRODUCTION:

It is easy to classify between different objects for humans, but complex for machines, hence image classification has become an important part in training the machines to recognize and classify between objects. Deep learning techniques are used to train these machines. Deep learning techniques have received vast popularity in recent years and have serious real-time applications which could automate most of the day-to-day activities. But to achieve this, it requires large amount of data and it is very time consuming and requires heavy computation for machines to learn this data.

The convolutional neural network (CNN) is a class of deep learning neural network. CNNs represent a huge breakthrough in image recognition. They're most used to analyze visual imagery and are frequently working behind the scenes in image classification. They can be found at the core of everything from Facebook's photo tagging to self-driving cars. They're working hard behind the scenes in everything from healthcare to security.

2. MOTIVATION:

Traffic signs are very important part of the road infrastructure. They provide critical information for safety of road user and provides compelling recommendations to road user/driver to adjust to their driving behavior to avoid accidents by following the traffic rules. Recognitions of traffic signs has received good attention in recent years in the field of intelligent vehicles. Car companies like Mercedes Benz, Tesla, BMW have heavily invested in these intelligent systems. Such systems can be really helpful in cases like missing traffic signs due to driving fatigue, low visibility due to critical weather conditions, or something as trivial as looking out for something in the car while driving. These activities could lead to accidents, but an intelligent car system which is able to detect the traffic signs and alert the driver at right time will help to avoid the risk of accidents.

Hence, we decided to explore more on classifying different traffic signs for such intelligent car systems.

3. METHODOLOGY:

Traffic signs may be divided to different categories based on the function and there are multiple subclasses in each of these categories. Here, we are classifying traffic signs for Florida area. There are several categories in which these road signs are being classified. Here we chose three main categories which are STOP, WARNING, and REGULARTORY signs to begin with.

For classification purpose, we decided to move forward in 2 stages.

STAGE 1:

The first level was to classify between STOP and Non-STOP signs (which may include WARNING, REGULARTORY or any other signs other than STOP). We used a binary classifier for this classification since there were only two categories.

STAGE 2:

The second level was to classify the traffic signs in STOP, WARNING and REGULARTORY signs. For this classification, we used a multi-class classifier.

4. DATA COLLECTION:

For creating classification models to recognize the traffic signs, we created our own dataset. Since our idea was to classify between the mentioned three main categories, we collected images accordingly. There were total 738 images collected which were as follows:

a. Stop sign – 245 images
b. Warning sign – 210 images
c. Regulatory sign – 283 images

These images were collected from google images, where we tried to include images of all types such as the one which were distinct, blur or images that had low resolution.

These are the few images we used in the dataset for classification.

SPEED
LIMIT
45

5. DATA PRE-PROCESSING:

As discussed earlier, our approach for classification was divided in two stages.

a. We created dataset as per the data required to work in these two stages.

- For STAGE 1 Classification i.e., STOP/Non-STOP Classification, we created a dataset from the images collected like below:
  STOP Sign: 241 images
  NON-STOP Signs (WARNING + REGULARTORY) Signs: 227 images
- For STAGE 2 Classification i.e., STOP, WARNING, REGULARTORY Signs, we created a dataset from the images collected like below:
  Stop sign – 245 images
  Warning sign – 210 images
  Regulatory sign – 283 images

b. We used OpenCV for converting BGR channel to RGB color channel.
c. For standardization, we resized images to resolution of 150 x 150 size to make all images consistent.
d. Converted pixel representations to numpy array.
e. We used ImageDataGenerator object for rescaling by using rescale = 1. /255 which normalizes the image pixel values to have zero mean and standard deviation of 1.
f. Data augmentation was done to add diversity to the dataset and to scale the dataset to avoid overfitting in the model. Each image was augmented by rescaling, rotating, height shifting, width shifting and zooming in and out.
g. Train-Test Split: The dataset was split into training and test set in 80:20 ratio.

6. DATA MODELLING:

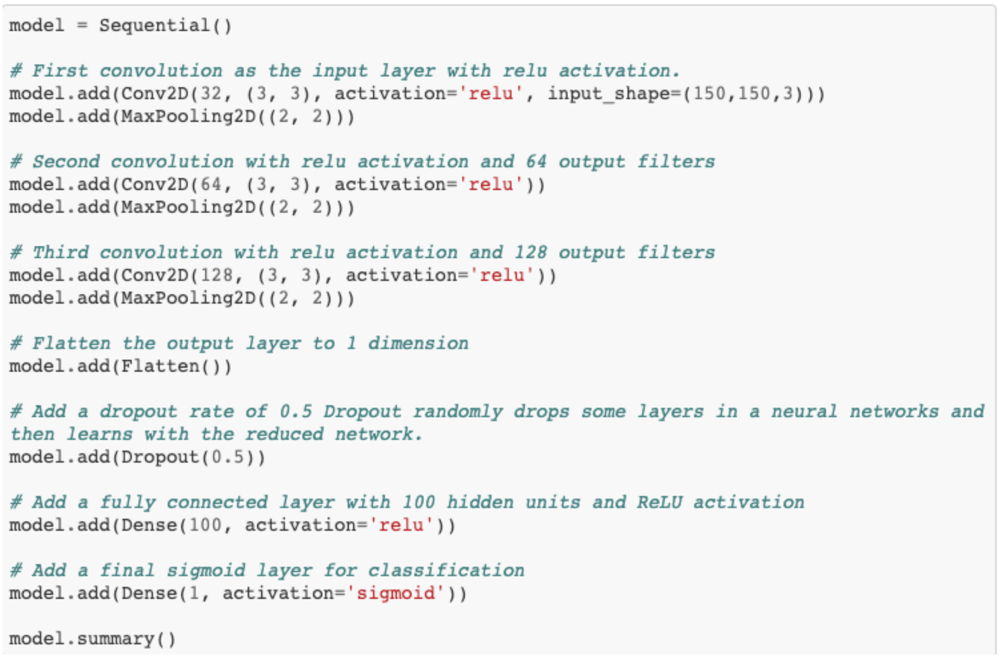
A. STAGE 1 Model: Binary Classification using CNN Classifier:

   1. For STAGE 1 Model, our aim was to distinguish between STOP and Non-STOP Signs.

   2. After processing the data as described in Data preprocessing above, we created a sequential model and specified CNN models various layers.

      a. We initialized three sets of convolution layers with a (3 x 3) kernel to help the model identify features and shapes with the traffic signs.
      b. These layers were then passed to the “relu” activation function.
      c. The layers were then passed through a MaxPooling function in order to reduce the dimensionality and let the model makes assumptions about features within the sub-regions.
      d. To generate the output vectors, we flattened the data before passing it through the dense layers ahead.
      e. A dropout layer was added to this model in order to prevent overfitting, as we had a small dataset.
      f. The activation function used to get a two-class output was “**sigmoid**” function which was connected to the final dense layer.

Model Code:



```
model = Sequential()

# First convolution as the input layer with relu activation.
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(150,150,3)))
model.add(MaxPooling2D((2, 2)))

# Second convolution with relu activation and 64 output filters
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

# Third convolution with relu activation and 128 output filters
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

# Flatten the output layer to 1 dimension
model.add(Flatten())

# Add a dropout rate of 0.5 Dropout randomly drops some layers in a neural networks and
then learns with the reduced network.
model.add(Dropout(0.5))

# Add a fully connected layer with 100 hidden units and ReLU activation
model.add(Dense(100, activation='relu'))

# Add a final sigmoid layer for classification
model.add(Dense(1, activation='sigmoid'))

model.summary()
```

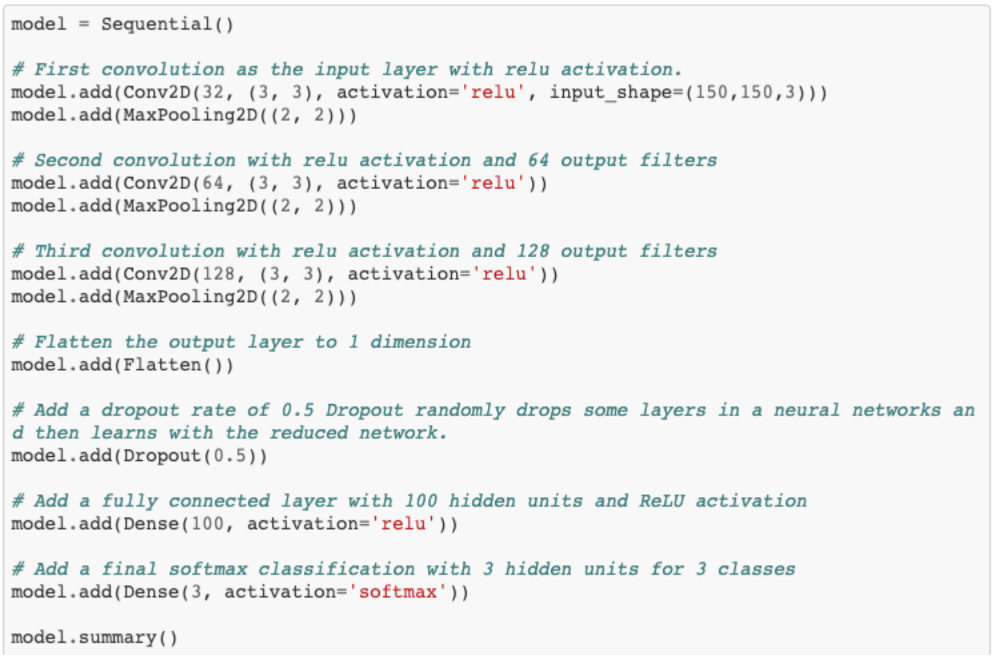
B. STAGE 2 Model: Multi-class Classification using CNN Classifier:

1. For STAGE 2 Model, our aim was to distinguish between STOP, WARNING and REGULATORY Signs.

2. After processing the data as described in Data preprocessing above, we created a sequential model and specified CNN models various layers.

   a. We initialized three sets of convolution layers with a (3 x 3) kernel to help the model identify features and shapes with the traffic signs.
   b. These layers were then passed to the “relu” activation function.
   c. The layers were then passed through a MaxPooling function in order to reduce the dimensionality and let the model makes assumptions about features within the sub-regions.
   d. To generate the output vectors, we flattened the data before passing it through the dense layers ahead.
   e. A dropout layer was added to this model in order to prevent overfitting, as we had a small dataset.
   f. The activation function used to get a multi-class output was “**softmax**” function which was connected to the final dense layer.

Model Code:



```
model = Sequential()

# First convolution as the input layer with relu activation.
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(150,150,3)))
model.add(MaxPooling2D((2, 2)))

# Second convolution with relu activation and 64 output filters
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

# Third convolution with relu activation and 128 output filters
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

# Flatten the output layer to 1 dimension
model.add(Flatten())

# Add a dropout rate of 0.5 Dropout randomly drops some layers in a neural networks an
d then learns with the reduced network.
model.add(Dropout(0.5))

# Add a fully connected layer with 100 hidden units and ReLU activation
model.add(Dense(100, activation='relu'))

# Add a final softmax classification with 3 hidden units for 3 classes
model.add(Dense(3, activation='softmax'))

model.summary()
```

7.MODEL EVALUTAION:

Model evaluation was done with the help of Accuracy and F1-Score. Different combination of neural network models was tried, and best model was picked with best accuracy and F1 Score amongst all the tried combination.

A. Binary classification model:

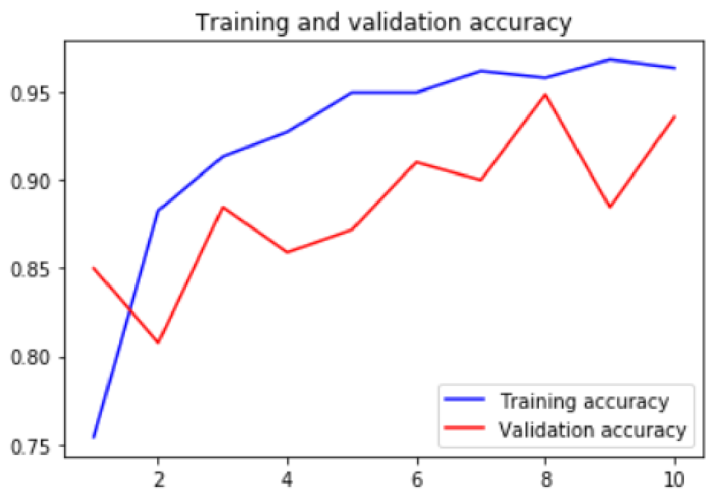
   Accuracy: 0.861702
   F1_score: 0.839506

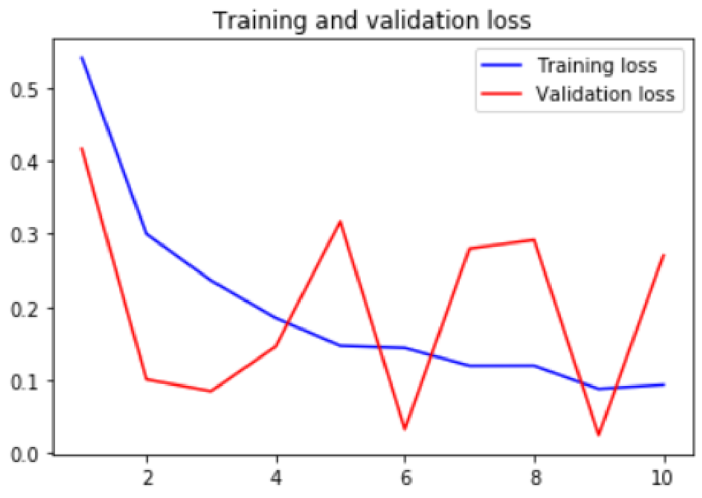
B. Multiclass classification model:

   Accuracy: 0.842

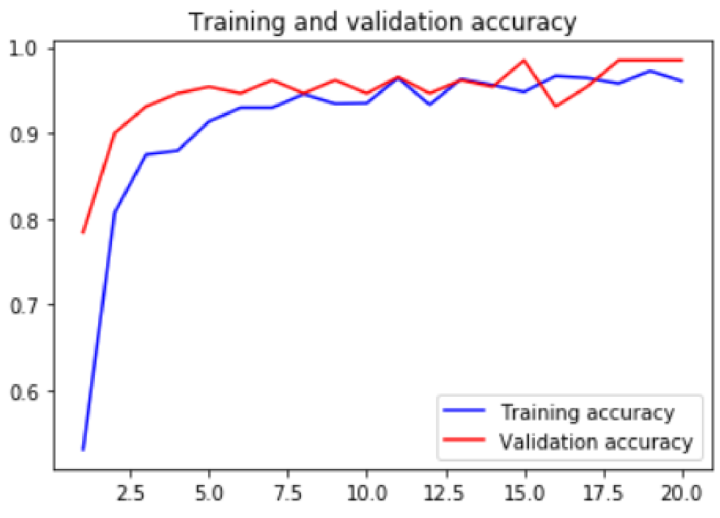
Training and validation accuracy and loss curve is shown below for binary and multiclass classification model.

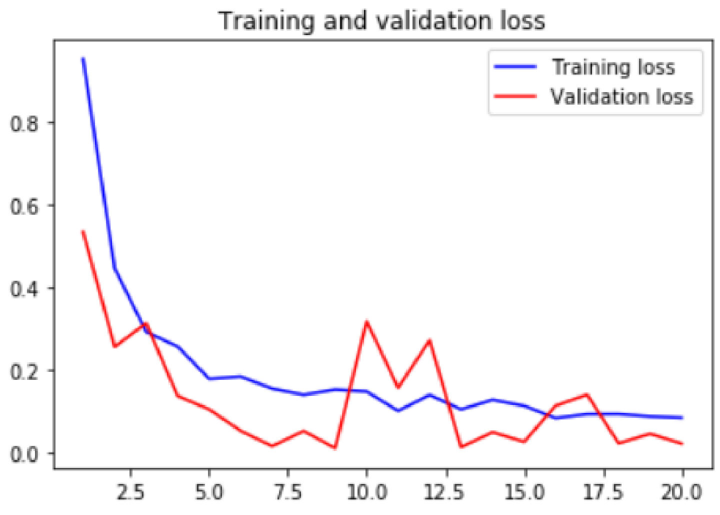
A. <u>Binary classification Training and validation accuracy and loss graph:</u>





B. <u>Multiclass classification Training and validation accuracy and loss graph:</u>



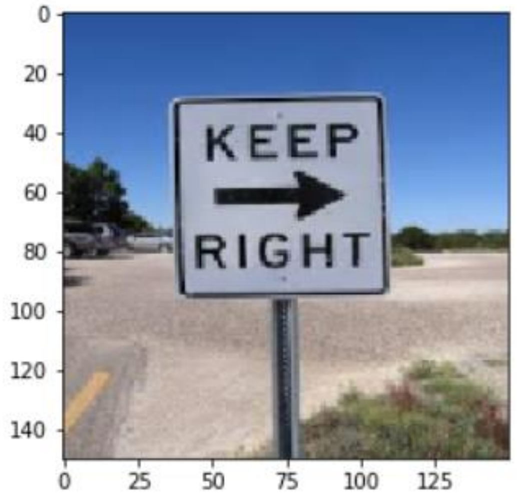


From above graphs we can infer that training and validation accuracy and loss curve are very much inclined with each. Hence models are not overfitting and they are performing well.

8.RESULTS AND PREDICTION:

A. Prediction result of the Binary classification model for prediction of signs.

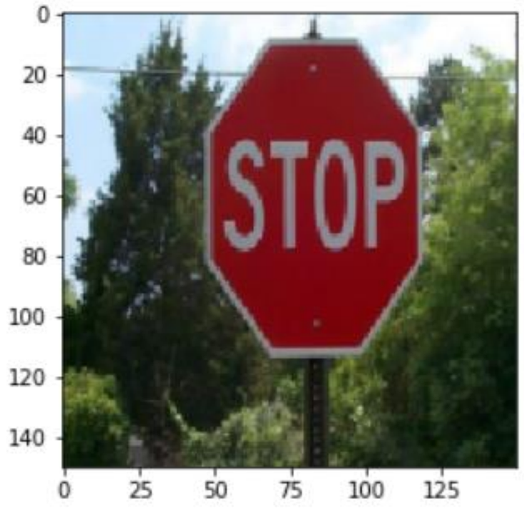
```
[0]
 This is a non-stop sign
```



B. Prediction result of the Multiclass classification model for prediction of stop sign.

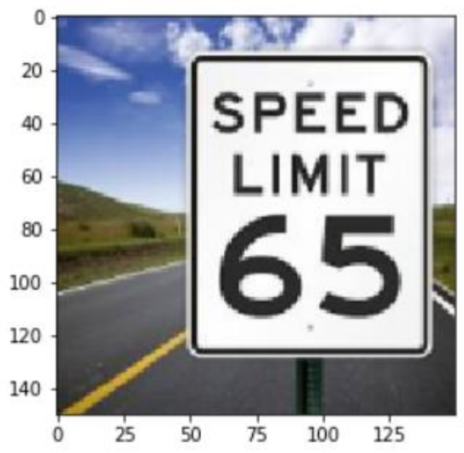
a. Stop sign

```
Predicted:  Stop sign
<matplotlib.image.AxesImage at 0x1c5d5908808>
```



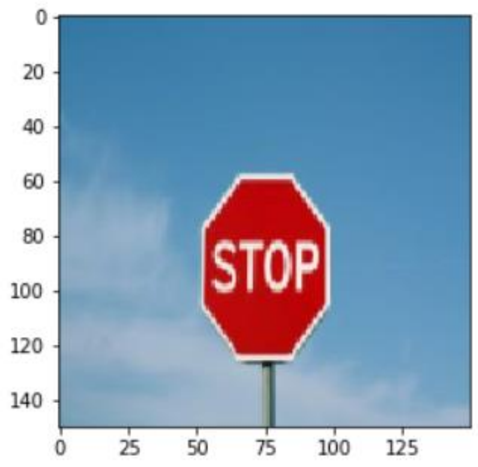
b. Regulatory sign.

```
Predicted:  Regulatory sign
<matplotlib.image.AxesImage at 0x1c5d50bf408>
```



c. Different Stop sign.

```
Predicted:  Stop sign
<matplotlib.image.AxesImage at 0x1c5d5b91448>
```



Prediction results are correct and as expected for given set of images.

9. FUTURE WORK:

The basic image classification model can be used to train self-driving cars in the classification of road signs while driving, but we would like to further enhance it with real-time complexities. We also want to further train the model for multiple sign detection in images and videos.

Currently there are many models which are implemented and being used for training self-driving cars. But we would also like to explore further given we have more computing resources and data.
Few of the features that can be implemented in the near future are:

a. Training model on a much bigger dataset for more accurate predictions
b. Detecting all unique traffic signs and not just a category
c. Detecting multiple signs correctly in a single image
d. Detecting signs correctly in blur images or far-away images.
e. Detecting unique traffic signs in a video

10. CONCLUSION:

We built different basic model for recognizing traffic road signs in an image which can be used to train a self-driving car or help drivers become more alert of the road signs ahead. The model can currently classify images as stop/non-stop and identify the category of the road sign – warning, regulatory or stop. With the use of different modelling techniques like the CNN classifiers used here, we were able to achieve results with up to 86% accuracy. Also, since the dataset used here was small, we can try to add more images which are diverse in nature to this dataset in order to improve the model accuracy.

11. REFERENCES:

a. https://towardsdatascience.com/wtf-is-image-classification-8e78a8235acb
b. https://www.pyimagesearch.com/2018/09/10/keras-tutorial-how-to-get-started-with-keras-deep-learning-and-python/
c. https://www.pyimagesearch.com/2018/05/14/a-gentle-guide-to-deep-learning-object-detection/
d. https://machinelearningmastery.com/object-recognition-with-deep-learning/
e. https://pythonprogramming.net/loading-custom-data-deep-learning-python-tensorflow-keras/
f. https://note.nkmk.me/en/python-opencv-bgr-rgb-cvtcolor/
g. https://www.geeksforgeeks.org/opencv-and-keras-traffic-sign-classification-for-self-driving-car/