*Python for Deep. Learning Project Increment-2*

**Traffic Signs Recognition**

**Team 7 contribution:**

Avinash Ganguri (ID – 6), *Model Evaluation with different hyperparameters*

Sri Sai Nikhil Kantipudi (ID – 10), *Model Analysis with callback functions and accuracy*

Dileep Reddy Peddakam (ID – 19), *Plotting and creating GUI in Tkinter*

**Done so far:**

***Increment -1***

1. *Preprocessed the Data,*

* Resizing the images
* Scaling the data
* One hot encoding using to\_categorial

1. *Building our CNN Model using keras.*
2. *Plotted the loss and accuracy using history object.*
3. *Displayed a sample image from the test data set*
4. *Predicting the accuracy of our model with test labels data file*

***Increment -2***

1. *Modifying our CNN Model with different hyperparameters to improve the accuracy.*
2. *Implemented callback functions such as Early stopping in our model*
3. *Created a Desktop Application using Tkinter python library to upload the image and classify it.*

**Remaining Parts to be Done for Final Increment:**

* Make sure to classify the images on real world images too, other than from the test data set.
* Creating a Web application using flask and will classify the images. If possible, will try to host the site so that it can be accessible to the public.

**Challenges faced:**

* Running the environment in local machine is quite hard with package dependency issues.
* Even with Google Co-Lab’s gpu and tpu, to train the large dataset is tedious and time taking as it requires the Colab pro, so we preferred running it on Jupyter Notebook instead.
* Running more epochs again and again when changing the model features is such a drag.

**Screenshots:**

***Importing the Libraries,***

**A screenshot of a cell phone

Description automatically generated**

***Loading the data and resizing it,*  
  
A screenshot of a social media post

Description automatically generated**

**A picture containing window, green, white

Description automatically generated**

***Preprocessing the data,***

* ***Splitting and scaling the data***
* ***One hot encoding***

**A screenshot of a cell phone

Description automatically generated**

***Building the CNN Model and Compiling it,***

***For Increment 1:***

**A screenshot of a social media post

Description automatically generated**

***Training the model with 10 epochs,***

**A screenshot of a social media post

Description automatically generated**

**For Increment 2:**

***Modified the CNN Model with different Hyperparameters,***

**Model 1**

**A screenshot of a social media post

Description automatically generated**

**Training Model 1**

**A screenshot of a cell phone

Description automatically generated**

*The Accuracy is 86.5% and the Validation Accuracy is 97.7% for Model 1.*

**Model 2**

**A screenshot of a social media post

Description automatically generated**

**Training Model 2**

**A screenshot of a cell phone

Description automatically generated**

*The Accuracy is 96.1% and the Validation Accuracy is 99.5% for Model 2.*

**Model 3**

**A screenshot of a social media post

Description automatically generated**

**Training Model 3**

**A screenshot of a cell phone

Description automatically generated**

*The Accuracy is 97.9% and the Validation Accuracy is 99.4% for Model 2.*

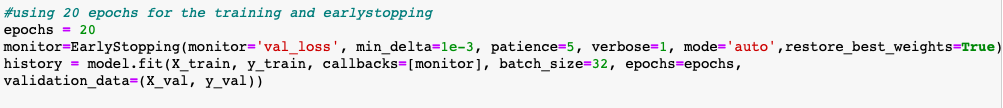
**Model 4**

**A screenshot of a social media post

Description automatically generated**

**Training Model 4**

* *With Early Stopping as Callback Function*

****

When the model is trained with 20 Epochs, it is observed that Epoch 16 is the best Epoch and it stopped further and saved it the best accuracy to get.

**A screenshot of a cell phone

Description automatically generated**

*The Accuracy is 98.5% and the Validation Accuracy is 99.7% for Model 4.*

After trying out different Hyperparameters and changing several combinations of layers and activation functions with different optimizers, we preferred **Model 4** as our best fit.

***Plotting the Loss and Accuracy using the History Object,*  
  
A screenshot of text

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

***Displaying a Test Image,***

**A screenshot of a cell phone

Description automatically generated**

***Predicting the model with Test data file and displaying the accuracy of the model,***

A screenshot of a social media post

Description automatically generated

*The Accuracy with the test data is 97.4%*

**Saving the Model**

****

**Creating GUI (Desktop Application)**

* *Using Tkinter Python Library*
* *Loading the ‘model.h5’ binary file to the Python script to classify*

*A screenshot of a computer

Description automatically generated*

*A screenshot of a computer

Description automatically generated*

*A picture containing sky

Description automatically generated*

*A screenshot of a cell phone

Description automatically generated*

*A screenshot of a cell phone

Description automatically generated*