**Traffic Sign Recognition**

**Introduction:**

In the modern world, everything is advanced in its place, technology is one such advancements which lead to invention of self-driving cars in which Traffic Sign Recognition is a crucial factor. Most of the companies like Tesla, Google, Ford and others have started their self-driving cars for their buyers to depend on driving. Traffic Sign Recognition is a python deep learnig project which classifies the traffic signs such as U-turn, stop, walk, and speed limit so on.

[**Dataset**](https://www.kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign) **used**

The dataset contains more than 50,000 images of different traffic signs. It is further classified into 43 different classes. The dataset is quite varying, some of the classes have many images while some classes have few images. The size of the dataset is around 600 MB. The dataset has a train folder which contains images inside each class and a test folder which we will use for testing our model.

**Proposed Architecture**

Steps to build our project are as follows:

* Explore the dataset
* Build a CNN model
* Train and validate the model
* Test the model with test dataset

Explore the dataset:

The ‘train’ folder contains 43 folders each representing a different class. The range of the folder is from 0 to 42. With the help of the OS module, we iterate over all the classes and append images and their respective labels in the data and labels list.

The list is converted to numpy arrays for feeding to the model.

The shape of data is (39209, 30, 30, 3) which means that there are 39,209 images of size 30×30 pixels and the last 3 means the data contains coloured images (RGB value).

With the sklearn package, the train\_test\_split() method is used to split training and testing data.

From the keras.utils package, the to\_categorical method is used to convert the labels present in y\_train and t\_test into one-hot encoding.

**Build a CNN model**

**To Classify images CNN model works best.**

The architecture of our model is:

* 2 Conv2D layer (filter=32, kernel\_
* size=(5,5), activation=”relu”)
* MaxPool2D layer ( pool\_size=(2,2))
* Dropout layer (rate=0.25)
* 2 Conv2D layer (filter=64, kernel\_size=(3,3), activation=”relu”)
* MaxPool2D layer ( pool\_size=(2,2))
* Dropout layer (rate=0.25)
* Flatten layer to squeeze the layers into 1 dimension
* Dense Fully connected layer (256 nodes, activation=”relu”)
* Dropout layer (rate=0.5)
* Dense layer (43 nodes, activation=”softmax”)

**Train and validate the model**

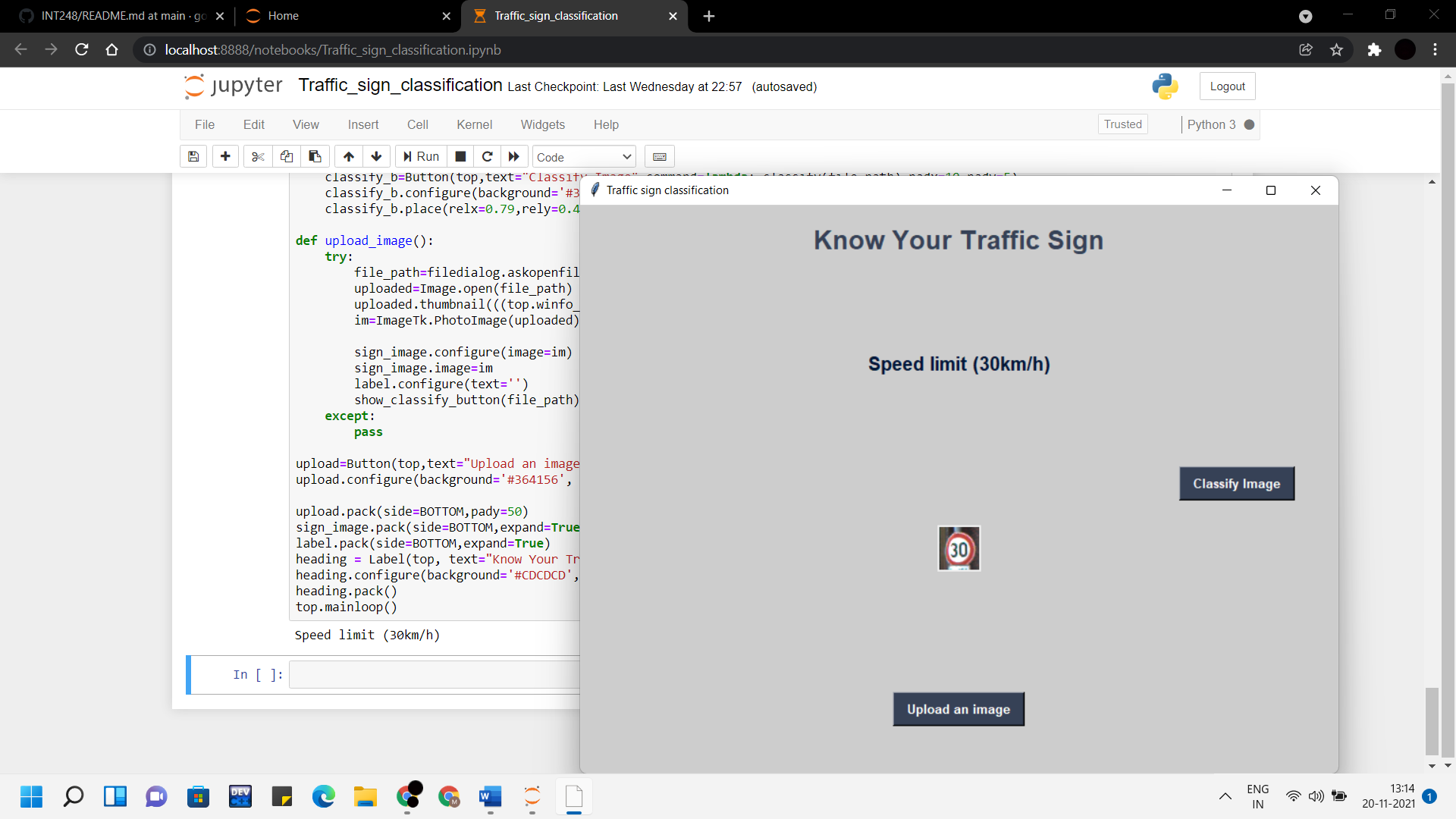
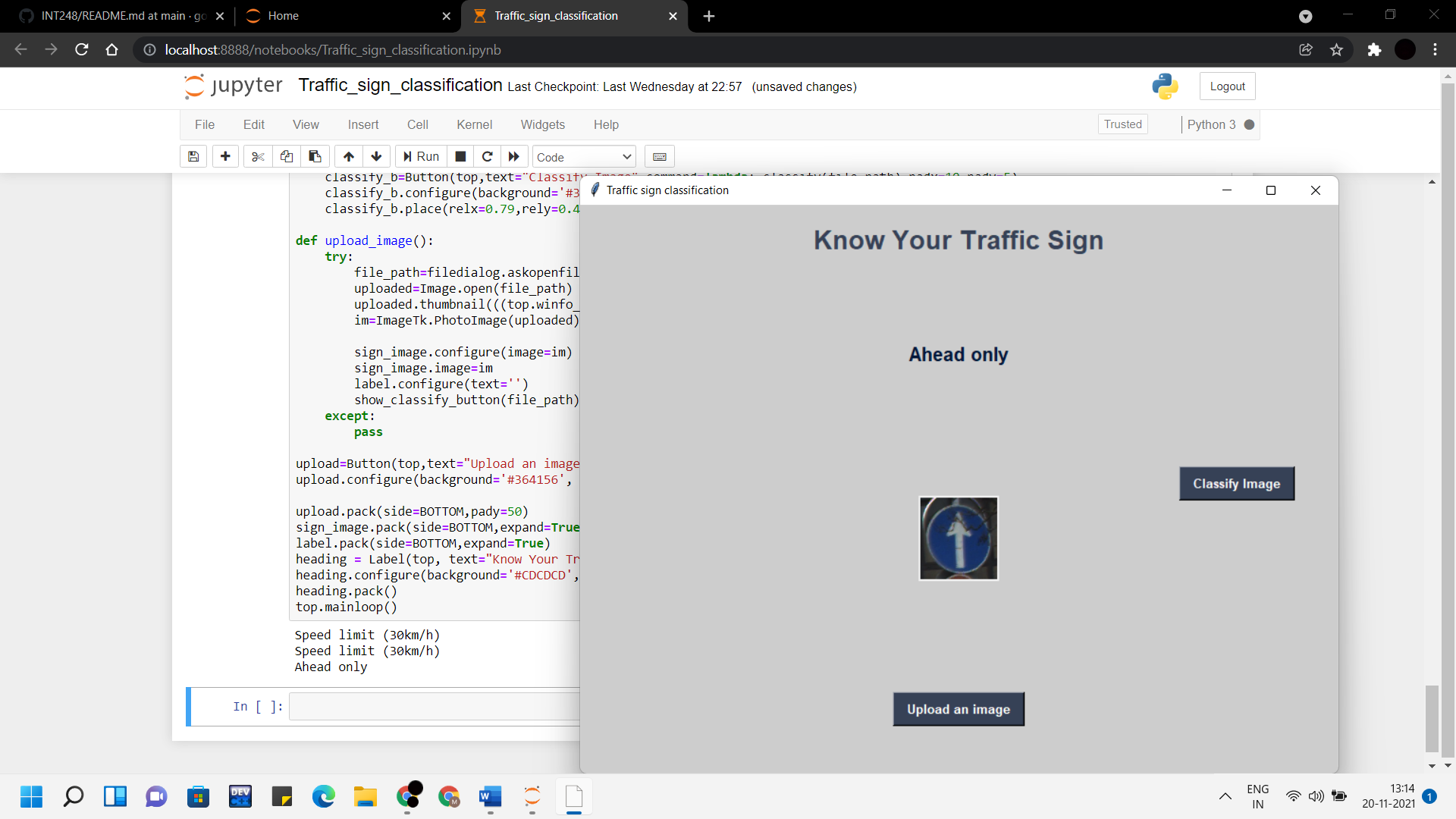
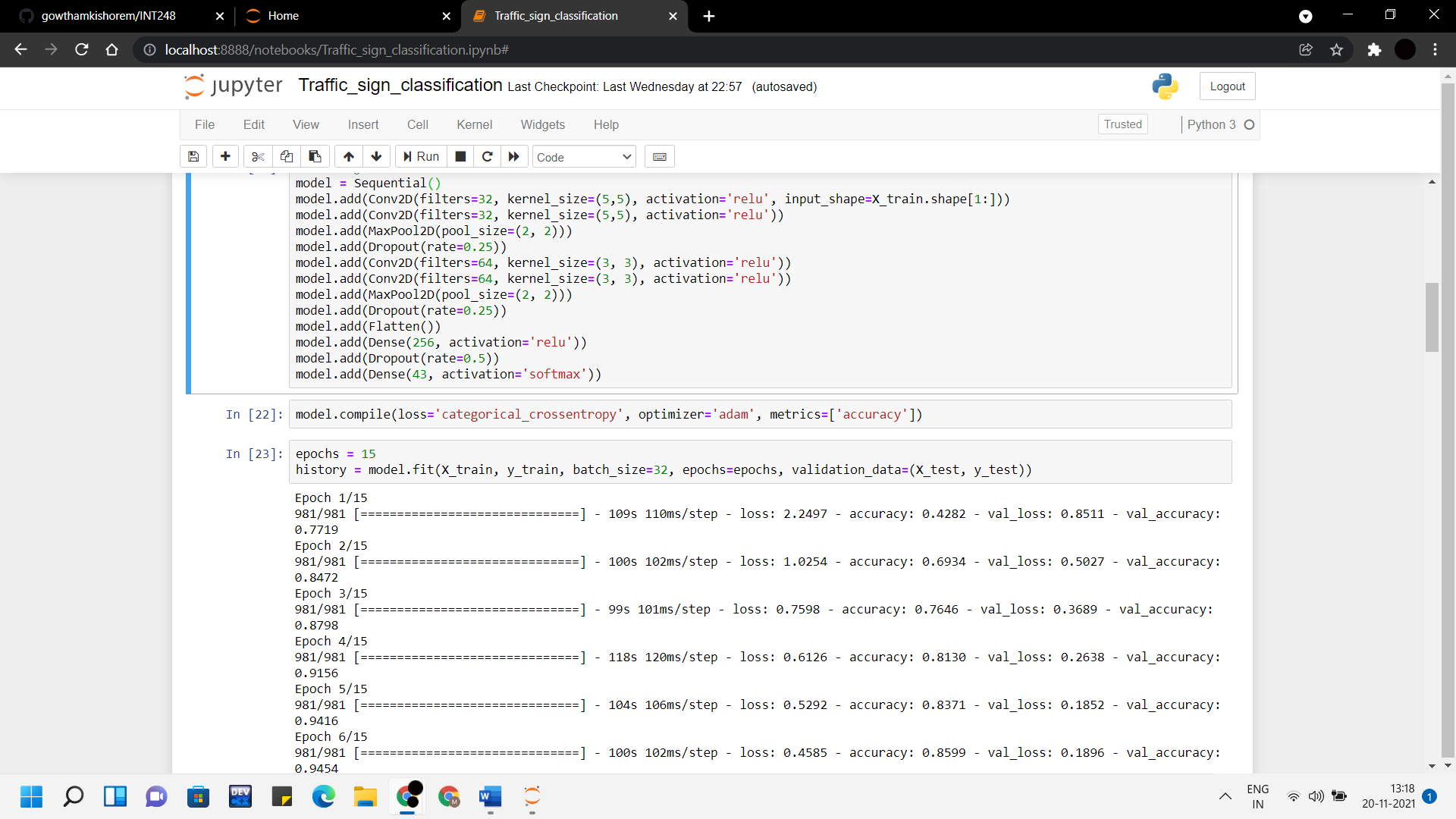
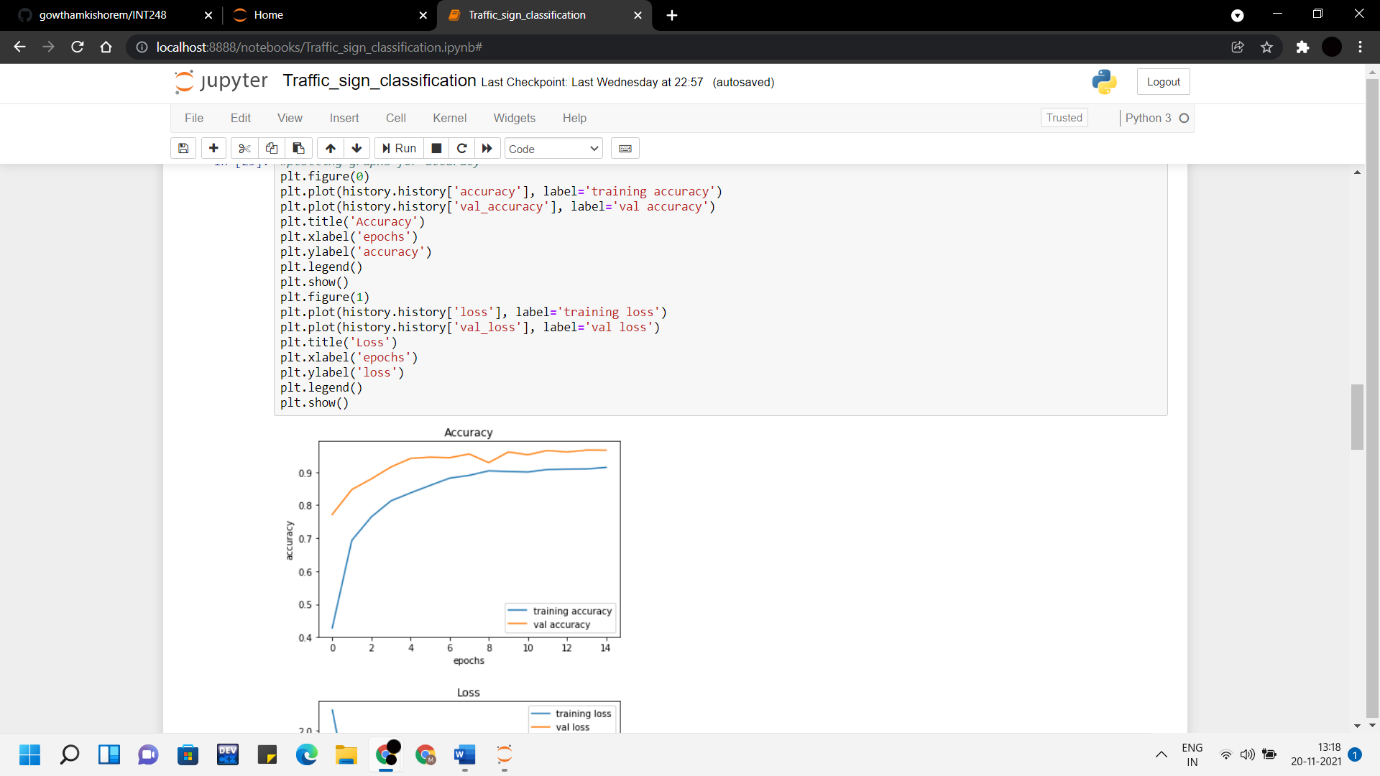
After building the model architecture, we then train the model using model.fit(). I tried with batch size 32 and 64. Our model performed better with 64 batch size. And after 15 epochs the accuracy was stable.

Our model got a 92% accuracy on the training dataset. With matplotlib, we plot the graph for accuracy and the loss.

**Test our model with test dataset**

The dataset contains a test folder and in a test.csv file, the details related to the image path and their respective class labels are available. We extract the image path and labels using pandas. Then to predict the model, we have to resize our images to 30×30 pixels and make a numpy array containing all image data. From the sklearn.metrics, the accuracy\_score is imported and observed how our model predicted the actual labels.

**Screenshots**

****

**Conclusion**

An accuracy of 92% is achieved in this model.

Finally, using the Keras model.save() function our model is saved.

In this Python project with source code, we have successfully classified the traffic signs classifier with 92% accuracy and also visualized how our accuracy and loss changes with time, which is pretty good from a simple CNN model.

**References**

https://data-flair.training/blogs/python-project-traffic-signs-recognition/