**ASSIGNMENT 2**

|  |  |
| --- | --- |
| Assignment Date | 06 November 2022 |
| Student Name | Mr.K.Mahadevan |
| Student Roll Number | 715319106007 |
| Maximum Marks | 2 Marks |

Traffic Signs Recognition – About the Python Project

In this Python project example, we will build a deep neural network model that can classify traffic signs present in the image into different categories. With this model, we are able to read and understand traffic signs which are a very important task for all autonomous vehicles.

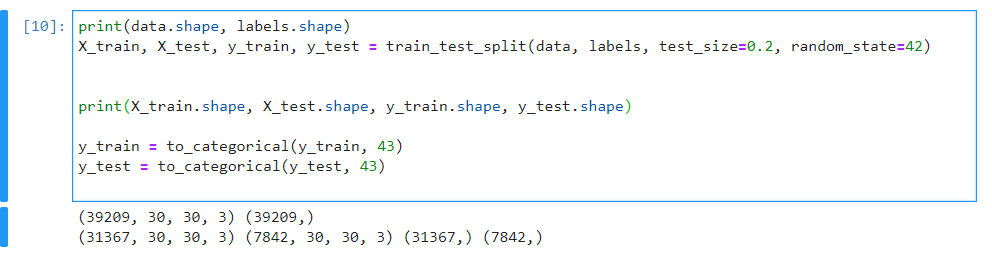
[Traffic Signs Dataset](https://www.kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign)

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 300 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.

**Step 1: Explore the dataset**

Our ‘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.

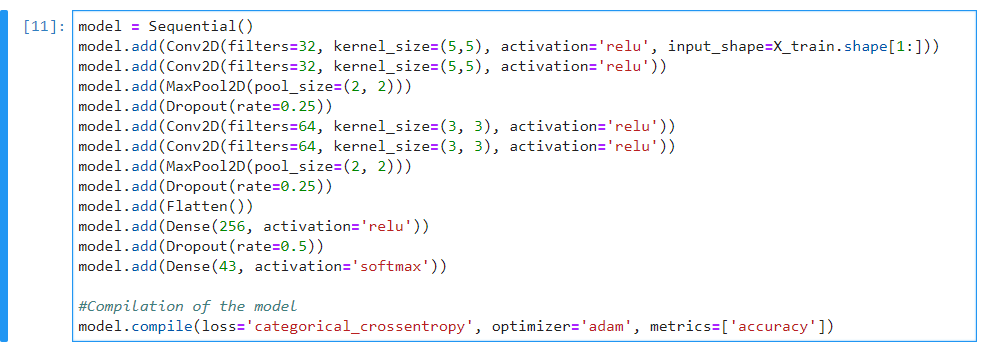
The PIL library is used to open image content into an array.

**Step 2: Build a CNN model**

To classify the images into their respective categories, we will build a CNN. CNN is best for image classification purposes.

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”)



**Steps 3: 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 After building the model architecture, we then train the model using 