Traffic Sign Recognition

By **Mahadasu Gowtham Kishore**

Introduction:

In the modern world, everything is advanced in its place, technology is one such advancement that lead to the 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 learning project which classifies the traffic signs such as U-turn, stop, walk, speed limit so on.

Dataset 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 that 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 the test dataset

<u>Explore the dataset:</u>

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 colored 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.

<u>Build a CNN model</u>

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

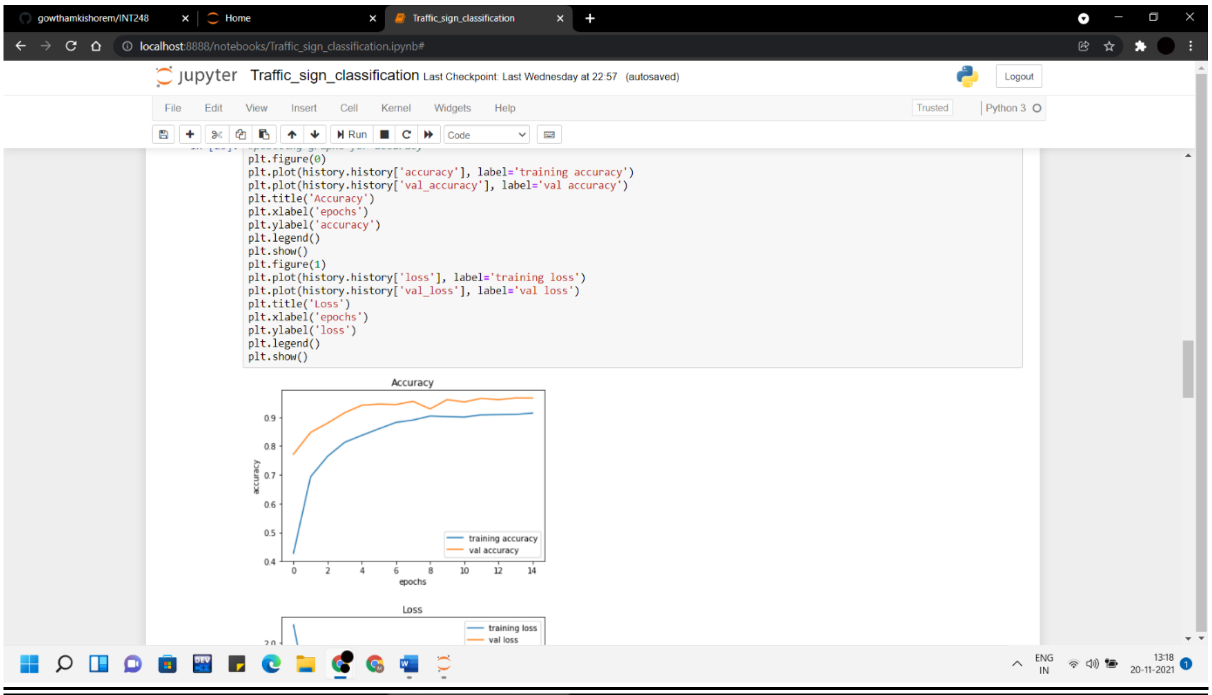
After building the model architecture, we then train the model using the model. fit(). I tried batch sizes 32 and 64. Our model performed better with a 64 batch size. And after 15 epochs the accuracy was stable.

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

Test our model with the test dataset

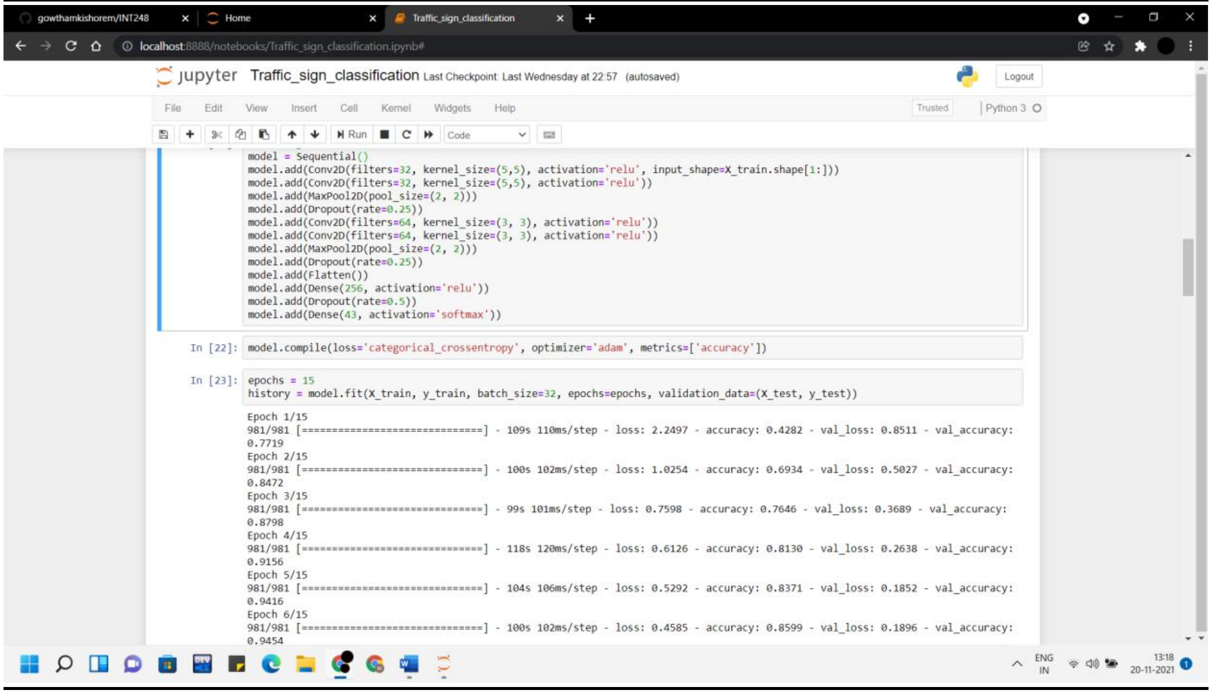
The dataset contains a test folder and in a test.csv file, the details related to the image path and their respective class labels are available. We extract the image path and labels using pandas. Then to predict the model, we have to resize our images to 30×30 pixels and make a NumPy array containing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

Screenshots



jupyter Traffic_sign_classification Last Checkpoint: Last Wednesday at 22:57 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
plt.figure(0)
plt.plot(history.history['accuracy'], label='training accuracy')
plt.plot(history.history['val_accuracy'], label='val accuracy')
plt.title('Accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.legend()
plt.show()
plt.figure(1)
plt.plot(history.history['loss'], label='training loss')
plt.plot(history.history['val_loss'], label='val loss')
plt.title('Loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```



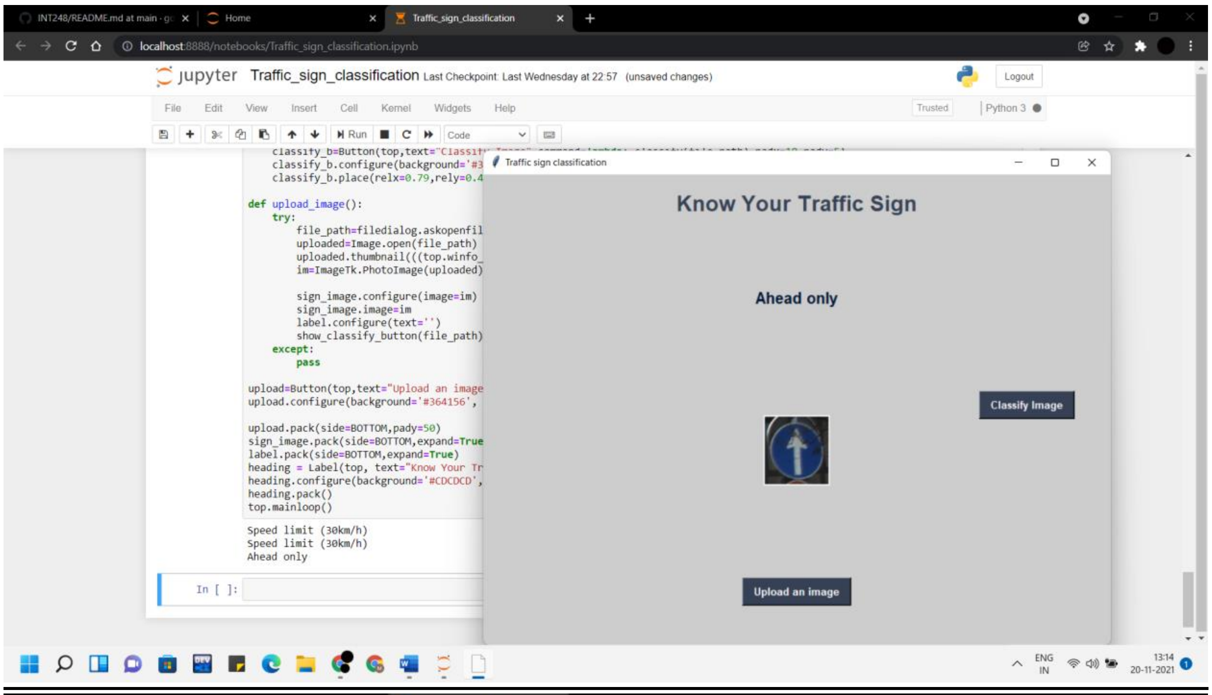
jupyter Traffic_sign_classification Last Checkpoint: Last Wednesday at 22:57 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu', input_shape=X_train.shape[1:]))
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(43, activation='softmax'))
```

```
In [22]: model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
In [23]: epochs = 15
         history = model.fit(X_train, y_train, batch_size=32, epochs=epochs, validation_data=(X_test, y_test))
```

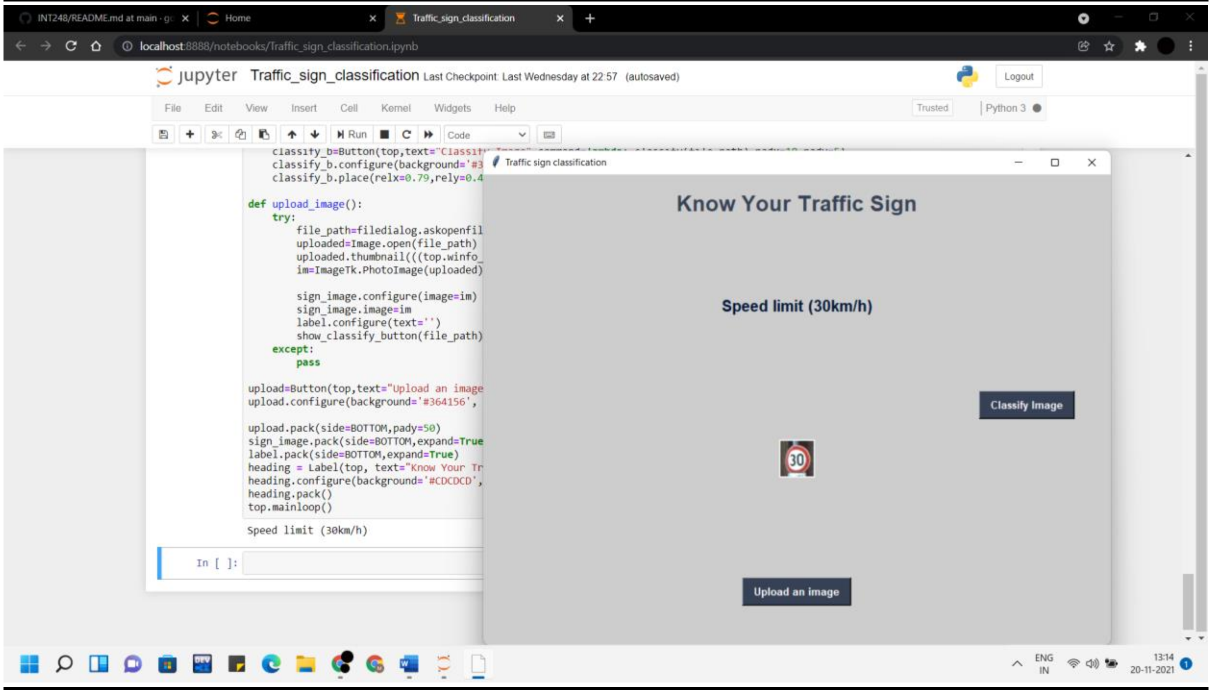
```
Epoch 1/15
981/981 [==============================] - 109s 110ms/step - loss: 2.2497 - accuracy: 0.4282 - val_loss: 0.8511 - val_accuracy:
0.7719
Epoch 2/15
981/981 [==============================] - 100s 102ms/step - loss: 1.0254 - accuracy: 0.6934 - val_loss: 0.5027 - val_accuracy:
0.8472
Epoch 3/15
981/981 [==============================] - 99s 101ms/step - loss: 0.7598 - accuracy: 0.7646 - val_loss: 0.3689 - val_accuracy:
0.8798
Epoch 4/15
981/981 [==============================] - 118s 120ms/step - loss: 0.6126 - accuracy: 0.8130 - val_loss: 0.2638 - val_accuracy:
0.9156
Epoch 5/15
981/981 [==============================] - 104s 106ms/step - loss: 0.5292 - accuracy: 0.8371 - val_loss: 0.1852 - val_accuracy:
0.9416
Epoch 6/15
981/981 [==============================] - 100s 102ms/step - loss: 0.4585 - accuracy: 0.8599 - val_loss: 0.1896 - val_accuracy:
0.9454
```


INT248/README.md at main · gi
Home
Traffic_sign_classification
localhost:8888/notebooks/Traffic_sign_classification.ipynb
jupyter
Traffic_sign_classification Last Checkpoint: Last Wednesday at 22:57 (unsaved changes)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Code
Traffic sign classification
Know Your Traffic Sign
Ahead only
Classify Image
Upload an image
ENG IN
13:14 20-11-2021

```
    classify_b=Button(top,text="Classif
    classify_b.configure(background='#3
    classify_b.place(relx=0.79,rely=0.4

def upload_image():
    try:
        file_path=filedialog.askopenfil
        uploaded=Image.open(file_path)
        uploaded.thumbnail(((top.winfo_
        im=ImageTk.PhotoImage(uploaded)

        sign_image.configure(image=im)
        sign_image.image=im
        label.configure(text='')
        show_classify_button(file_path)
    except:
        pass

upload=Button(top,text="Upload an image
upload.configure(background='#364156',

upload.pack(side=BOTTOM,pady=50)
sign_image.pack(side=BOTTOM,expand=True
label.pack(side=BOTTOM,expand=True)
heading = Label(top, text="Know Your Tr
heading.configure(background='#CDCDCD',
heading.pack()
top.mainloop()
```

```
Speed limit (30km/h)
Speed limit (30km/h)
Ahead only
```

```
In [ ]:
```


INT248/README.md at main · gi
Home
Traffic_sign_classification
localhost:8888/notebooks/Traffic_sign_classification.ipynb
jupyter
Traffic_sign_classification Last Checkpoint: Last Wednesday at 22:57 (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Code
Traffic sign classification
Know Your Traffic Sign
Speed limit (30km/h)
Classify Image
30
Upload an image
ENG IN
13:14 20-11-2021

```
    classify_b=Button(top,text="Classif
    classify_b.configure(background='#3
    classify_b.place(relx=0.79,rely=0.4

def upload_image():
    try:
        file_path=filedialog.askopenfil
        uploaded=Image.open(file_path)
        uploaded.thumbnail(((top.winfo_
        im=ImageTk.PhotoImage(uploaded)

        sign_image.configure(image=im)
        sign_image.image=im
        label.configure(text='')
        show_classify_button(file_path)
    except:
        pass

upload=Button(top,text="Upload an image
upload.configure(background='#364156',

upload.pack(side=BOTTOM,pady=50)
sign_image.pack(side=BOTTOM,expand=True
label.pack(side=BOTTOM,expand=True)
heading = Label(top, text="Know Your Tr
heading.configure(background='#CDCDCD',
heading.pack()
top.mainloop()
```

```
Speed limit (30km/h)
```

```
In [ ]:
```

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 change with time, which is pretty good from a simple CNN model.

References

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