Peter the Great St. Petersburg Polytechnic University

Institute of Computer Sciences and Technologies

School of Cyber-Physical Systems and Control

COURSE WORK

Traffic Sign Recognition using CNN Algorithm

on the subject of Neuroinformatics and Neurotechnologies

Done By :-Abhijeet Sachin Chougule

Group No. :- 3540901/11701

Approved by :- Shkodyrev V.P

CONTENT

INTRODUCTION AND OBJECTIVE

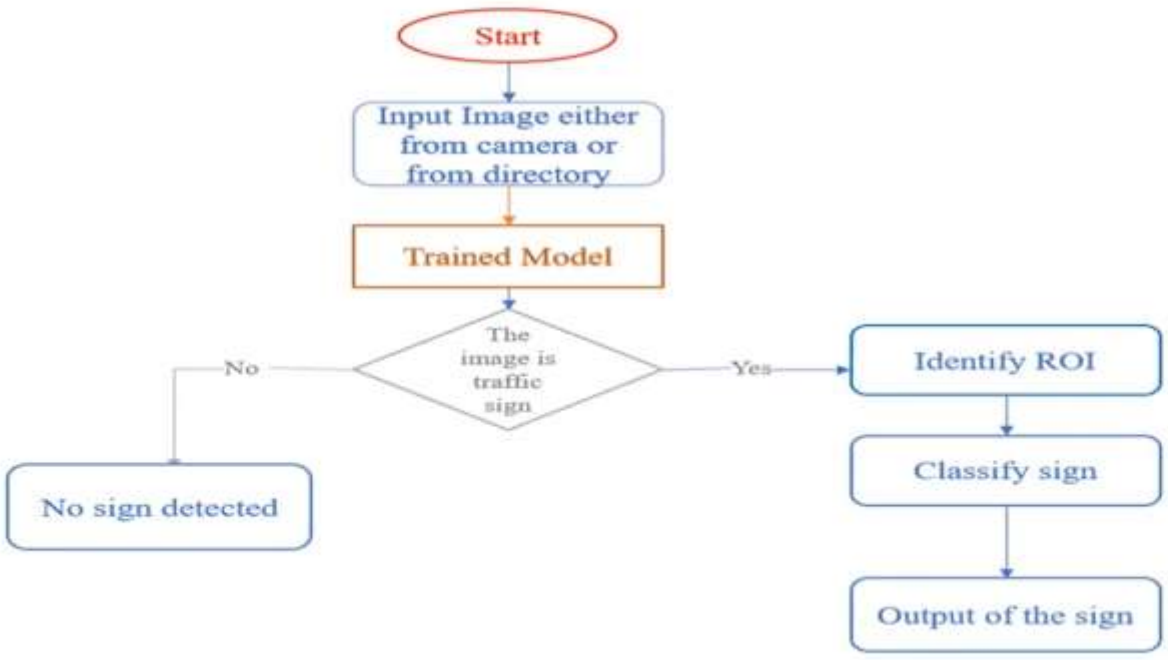
Recently the number of road vehicles has increased enormously thanks to the technological achievements in the motor industry and very precisely the availability of low rates. With this remarkable growth, the number of accidents is as well in an infinite raise year after year, due to different causes, in which the ignorance of traffic signs is considered as a major cause of these lasts. Developing automated traffic sign recognition systems helps assisting the driver in different ways in order to guarantee his/her safety, which preserves as well the safety of other drivers and pedestrians. These systems have one main goal: detecting and recognizing traffic signs during the driving process. With these functionalities the system can guide and alert the drivers to prevent danger. 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.

The benefits are generally focused around driver convenience, which for many is great news. As mentioned above, it should take the stress off attempting to keep on top of speed signs when driving somewhere new. Also, for those who spend a lot of time on the motorway where variable speed limits are becoming all the more frequent, many systems, such as Ford's traffic sign recognition software, will identify speed limits displayed on motorway gantries. More advanced versions, such as that offered by Mercedes, will offer you the chance to jump to the new speed limit (with the press of a button) if you are driving using cruise control. There are even systems which will jump for you should you wish them to, so you never miss a speed change again - or in theory at least.

SYSTEM AND ITS SOULUTION

Existing system problem Despite the advantages of traffic sign recognition, there are drawbacks. For starters, in more rural areas it is fairly common to see a traffic sign which has been engulfed by a hedge. Which could mean you and your car plough into a 30mph zone way above the speed limit, endangering other motorists and pedestrians. Next time you are on the road, take a look at how many vans and lorries have little speed limit stickers on them. There have been reports of cars thinking these small stickers are the speed limits - not helped of course if motorists are driving too close to the rear of a van or lorry. Normally these stickers say what the vehicle is limited to, which is fine on a motorway, but if you are in a built-up area, chances are you don't want your car thinking the speed limit is 60 or 70mph. The worst case here is cars that automatically change to the new speed limit. That said, most cars of that capability will have adaptive cruise control as well, so will know there is a vehicle in front and not fly off to 70mph.

UML DAIGRAM


Start
Input Image either from camera or from directory
Trained Model
The image is traffic sign
No
Yes
No sign detected
Identify ROI
Classify sign
Output of the sign

METHODOLOGY USED

Algorithm that used in creating this mode is Convolutional Neural Network (CNN). Deep Neural Networks have greater capabilities for image pattern recognition and are widely used in Computer Vision algorithms and Convolutional Neural Network (CNN) is a class of Deep Neural Networks which is most commonly applied to analysing visual imagery. Traffic sign classification and detection are one of the major tasks in self-driving as it gives the input of what sign is in the image to decision making.

- **To Build CNN Model :-**

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

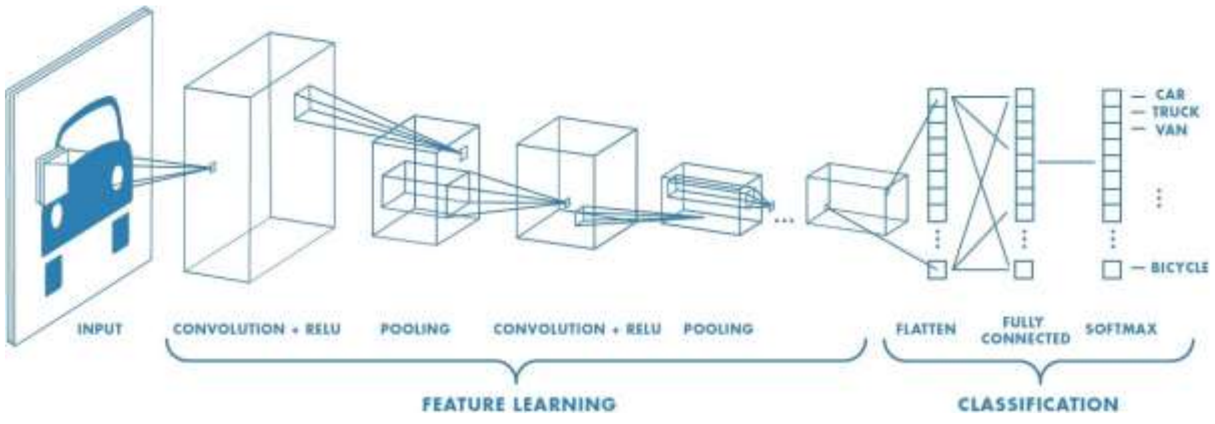
Fully connected layer (256 nodes, activation=”relu”)

Dropout layer (rate=0.5)

layer (43 nodes, activation=”softmax”)

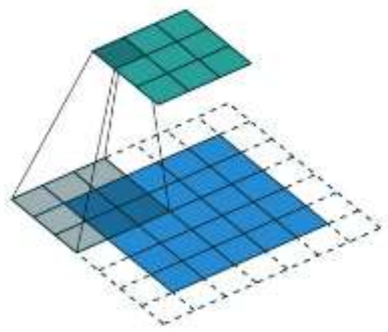
We compile the model with Adam optimizer which performs well and loss is “categorical_crossentropy” because we have multiple classes to categorise.

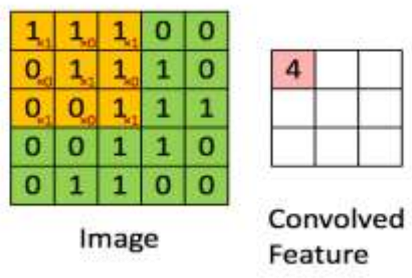
- **CNN Layers :-**

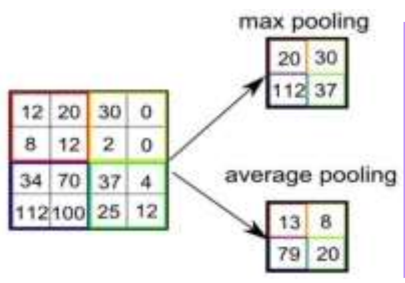


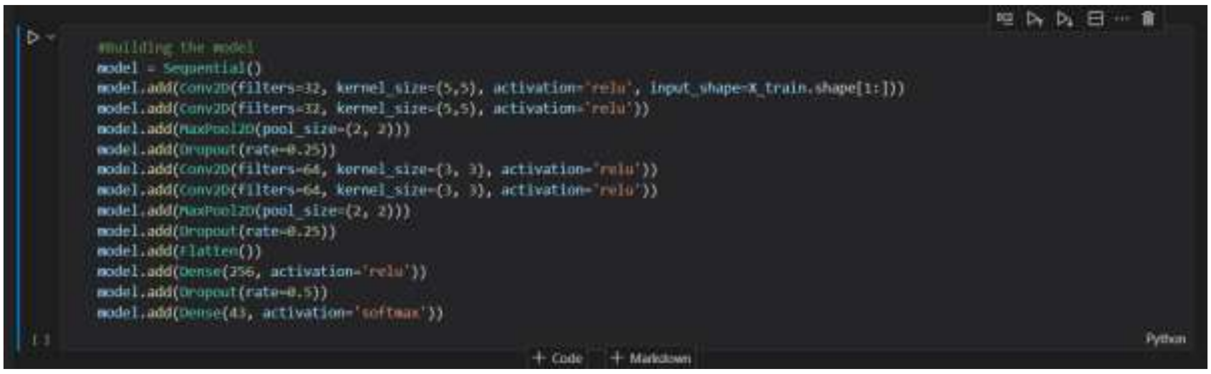
INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
CAR
TRUCK
VAN
BICYCLE
FEATURE LEARNING
CLASSIFICATION

MATHEMATICAL TASK FORMULATION









```
#Building the model
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'))
```

CODE

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cv2
import tensorflow as tf
from PIL import Image
import os
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout

data = []
labels = []
classes = 43
cur_path = os.getcwd()

#Retrieving the images and their labels
for i in range(classes):
    path = os.path.join(cur_path,'train',str(i))
    images = os.listdir(path)

    for a in images:
        try:
            image = Image.open(path + '\\'+ a)
            image = image.resize((30,30))
            image = np.array(image)
            #sim = Image.fromarray(image)
            data.append(image)
            labels.append(i)
        except:
            print("Error loading image")

#Converting lists into numpy arrays
data = np.array(data)
labels = np.array(labels)

print(data.shape, labels.shape)
#Splitting training and testing dataset
X_train, X_test, y_train, y_test = train_test_split(data, labels,
test_size=0.2, random_state=42)

print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
#Converting the labels into one hot encoding
y_train = to_categorical(y_train, 43)
y_test = to_categorical(y_test, 43)

#Building the model
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'))

#Compilation of the model
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

epochs = 15
history = model.fit(X_train, y_train, batch_size=32, epochs=epochs,
validation_data=(X_test, y_test))
model.save("my_model.h5")

#plotting graphs for accuracy
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()
```

```
#testing accuracy on test dataset
from sklearn.metrics import accuracy_score

y_test = pd.read_csv('Test.csv')

labels = y_test["ClassId"].values
imgs = y_test["Path"].values

data=[]

for img in imgs:
    image = Image.open(img)
    image = image.resize((30,30))
    data.append(np.array(image))

X_test=np.array(data)

predict_x=model.predict(X_test)
# pred = model.predict_classes(X_test)

#Accuracy with the test data
from sklearn.metrics import accuracy_score
print(accuracy_score(labels, predict_x, normalize=True, sample_weight=None))

#print(model.predict(X_test[20]))
```

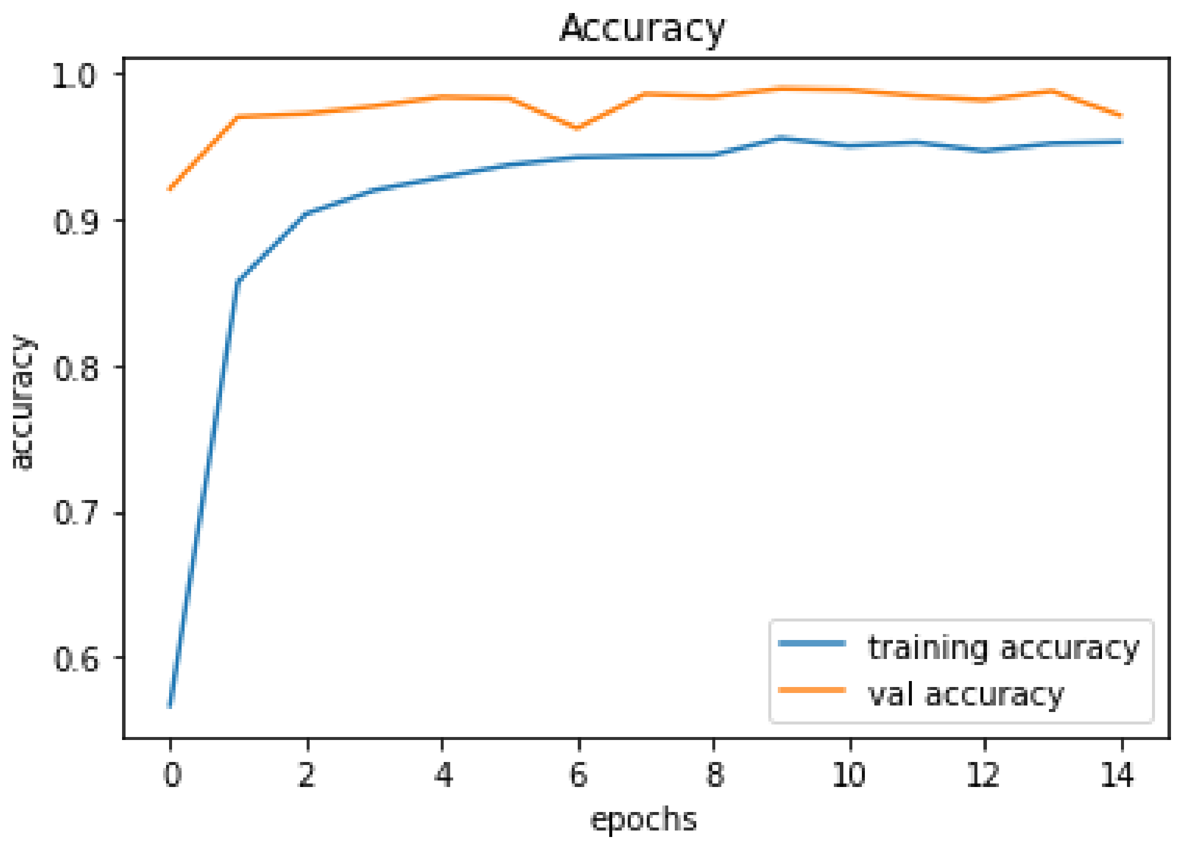
GUI:-

```
import tkinter as tk
from tkinter import filedialog
from tkinter import *
from PIL import ImageTk, Image

import numpy
#load the trained model to classify sign
from keras.models import load_model
model = load_model('traffic_classifier.h5')

#dictionary to label all traffic signs class.
classes = { 1:'Speed limit (20km/h)',
            2:'Speed limit (30km/h)',
            3:'Speed limit (50km/h)',
            4:'Speed limit (60km/h)',
            5:'Speed limit (70km/h)',
            6:'Speed limit (80km/h)',
            7:'End of speed limit (80km/h)',
            8:'Speed limit (100km/h)',
            9:'Speed limit (120km/h)',
           10:'No passing',
           11:'No passing veh over 3.5 tons',
           12:'Right-of-way at intersection',
           13:'Priority road',
           14:'Yield',
           15:'Stop',
           16:'No vehicles',
           17:'Veh > 3.5 tons prohibited',
           18:'No entry',
           19:'General caution',
           20:'Dangerous curve left',
           21:'Dangerous curve right',
           22:'Double curve',
           23:'Bumpy road',
           24:'Slippery road',
           25:'Road narrows on the right',
           26:'Road work',
           27:'Traffic signals',
           28:'Pedestrians',
           29:'Children crossing',
           30:'Bicycles crossing',
           31:'Beware of ice/snow',
           32:'Wild animals crossing',
           33:'End speed + passing limits',
           34:'Turn right ahead',
           35:'Turn left ahead',
           36:'Ahead only',
           37:'Go straight or right',
           38:'Go straight or left',
           39:'Keep right',
           40:'Keep left',
           41:'Roundabout mandatory',
           42:'End of no passing',
           43:'End no passing veh > 3.5 tons' }

#initialise GUI
top=tk.Tk()
top.geometry('800x600')
top.title('Traffic sign classification')
top.configure(background='#CDCDCD')

label=Label(top,background='#CDCDCD', font=('arial',15,'bold'))
sign_image = Label(top)

def classify(file_path):
```

```python
    global label_packed
    image = Image.open(file_path)
    image = image.resize((30,30))
    image = numpy.expand_dims(image, axis=0)
    image = numpy.array(image)
    print(image.shape)
    pred = model.predict_classes([image])[0]
    sign = classes[pred+1]
    print(sign)
    label.configure(foreground='#011638', text=sign)


def show_classify_button(file_path):
    classify_b=Button(top,text="Classify Image",command=lambda:
classify(file_path),padx=10,pady=5)
    classify_b.configure(background='#364156',
foreground='white',font=('arial',10,'bold'))
    classify_b.place(relx=0.79,rely=0.46)

def upload_image():
    try:
        file_path=filedialog.askopenfilename()
        uploaded=Image.open(file_path)
        uploaded.thumbnail(((top.winfo_width()/2.25),(top.winfo_height()/2.25)
))
        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",command=upload_image,padx=10,pady=5)
upload.configure(background='#364156',
foreground='white',font=('arial',10,'bold'))

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 Traffic Sign",pady=20,
font=('arial',20,'bold'))
heading.configure(background='#CDCDCD',foreground='#364156')
heading.pack()
top.mainloop()
```

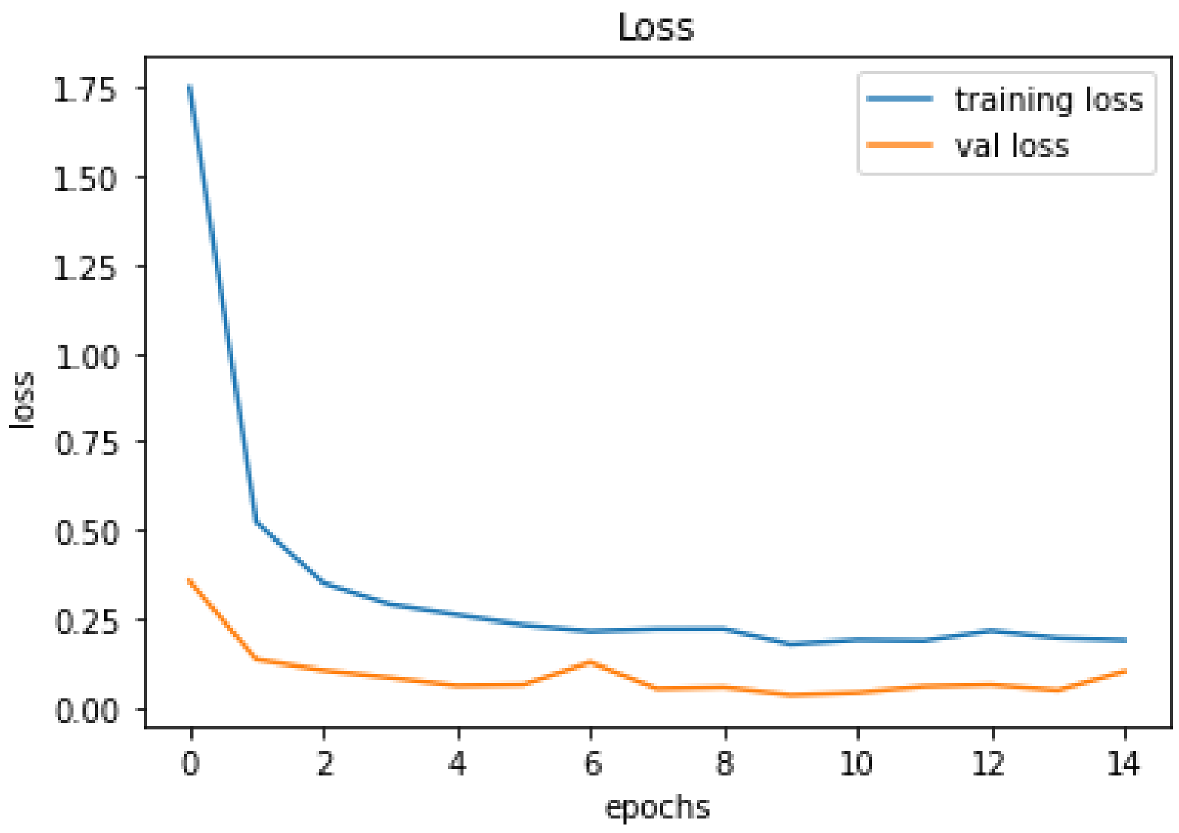
RESULTS AND OUTPUT

- **MODEL ACCURACY**



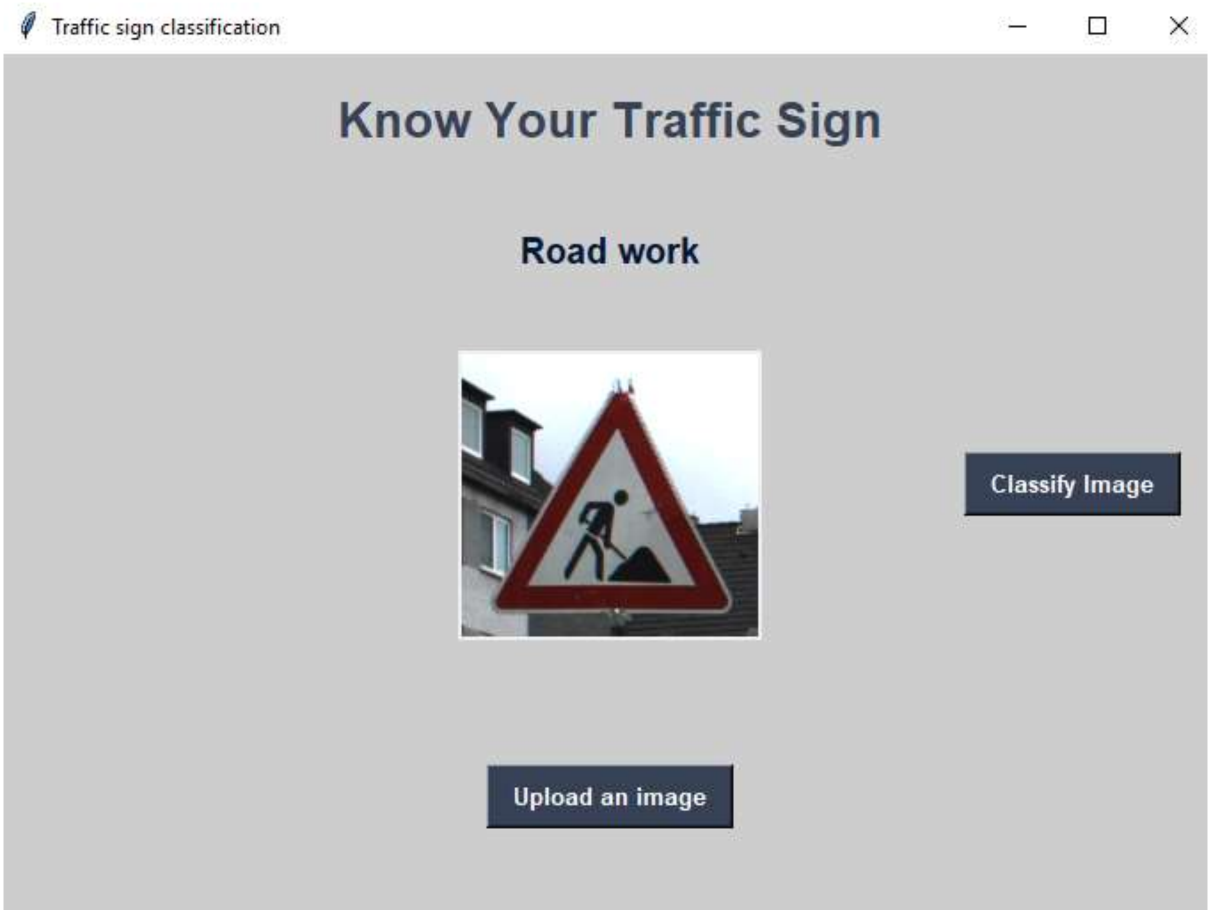
Accuracy Plot of the trained model with respect to no. of epochs given

- **MODEL LOSS**



Loss Plot of the trained model with respect to no. of epochs given

OUTPUT :-


Traffic sign classification
Know Your Traffic Sign
Road work
Classify Image
Upload an image

CONCLUSION

This project considers an implementation of the classification algorithm for the traffic signs recognition task. Combined with preprocessing and localization steps from previous works, the proposed method for traffic signs classification shows very good results: 95.94% of correctly classified images. The proposed classification solution is implemented using the TensorFlow framework. The use of our TSR algorithms allows processing of video streams in real-time with high resolution, and therefore at greater distances and with better quality than similar TSR systems have. Full HD resolution makes it possible to detect and recognize a traffic sign at a distance up to 50 m. Also, we plan to use a CNN not only for classification but for object detection too.

FUTURE SCOPE

Another direction for further research is to develop a real time traffic sign recognition system which captures a video by a camera mounted on the vehicle, detects and recognises the traffic signs in real time and gives the result to the driver within a sufficient time frame in order to take the right action.

The crucial issue in real time applications is the time spent to recognise the traffic sign. This should be reduced to the 169 minimum by choosing the proper techniques for real time applications and by optimizsing the code. The methods presented in this thesis can be modified to fit the real time requirements.

REFERENCES

- https://www.sciencedirect.com/science/article/pii/S1877705817341231
- https://link.springer.com/chapter/10.1007/978-3-319-23989-7_28
- International Conference on Intelligent Science and Big Data Engineering
- IScIDE 2015: Intelligence Science and Big Data Engineering. Image and Video Data Engineering pp 272-280
- https://www.analyticsvidhya.com/blog/2021/12/traffic-signs-recognition-using-cnn-and-keras-in-python/
- Data refreance :- https://www.kaggle.com/andrewmvd/road-sign-detection
- https://www.youtube.com/watch?v=SWaYRyi0TTs
- https://www.youtube.com/watch?v=qahpZkPlTRM&t=430s