ASSIGNMENT 3

Assignment Date	06 November 2022
Student Name	V.Kaalisankar
Student Roll Number	715319106302
Maximum Marks	2 Marks

Real-time Traffic Monitoring System using Python

Test our model with test dataset

Our dataset contains a test folder and in a test.csv file, we have the details related to the image path and their respective class labels. 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 num array containing all image data. From the sk learn. metrics, we imported the accuracy_ score and observed how our model predicted the actual labels. We achieved a 95% accuracy in this model.

[14]:
```
from sklearn.metrics import accuracy_score
import pandas as pd
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)

pred = model.predict_classes(X_test)

#Accuracy with the test data
from sklearn.metrics import accuracy_score
accuracy_score(labels, pred)
```

[14]:
```
0.9532066508313539
```

Traffic Signs Classifier GUI

Now we are going to build a graphical user interface for our traffic signs classifier with Tkinter. kinter sis a GUI toolkit in the standard python library. Make a new file in the project folder and copy the below code. Save it as gui.py and you can run the code by typing python gui.py in the command line.

In this file, we have first loaded the trained model 'traffic_classifier.h5' using Keras. And then we build the GUI for uploading the image and a button is used to classify which calls the classify() function. The classify() function is converting the image into the dimension of shape (1, 30, 30, 3). This is because to predict the traffic sign we have to provide the same dimension we have used when building the model. Then we predict the class, the model. predict_ classes(image) returns us a number between (0-42) which represents the class it belongs to. We use the dictionary to get the information about the class. Here's the code for the gui.py file.

```
1. import numpy as np
2. import pandas as pd
3. import matplotlib.pyplot as plt
4. import cv2
5. import tensorflow as tf
6. from PIL import Image
7. import os
8. from sklearn.model_selection import train_test_split
9. from keras.utils import to_categorical
10.     from keras.models import Sequential, load_model
11.     from keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout
12.     data = []
13.     labels = []
14.     classes = 43
15.     cur_path = os.getcwd()
16.     #Retrieving the images and their labels
17.     for i in range(classes):
18.     path = os.path.join(cur_path,'train',str(i))
19.     images = os.listdir(path)
20.     for a in images:
21.     try:
22.     image = Image.open(path + '\\'+ a)
23.     image = image.resize((30,30))
24.     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)
```

```
61.     plt.plot(history.history['accuracy'], label='training accuracy')
62.     plt.plot(history.history['val_accuracy'], label='val accuracy')
63.     plt.title('Accuracy')
64.     plt.xlabel('epochs')
65.     plt.ylabel('accuracy')
66.     plt.legend()
67.     plt.show()
68.     plt.figure(1)
69.     plt.plot(history.history['loss'], label='training loss')
70.     plt.plot(history.history['val_loss'], label='val loss')
71.     plt.title('Loss')
72.     plt.xlabel('epochs')
73.     plt.ylabel('loss')
74.     plt.legend()
75.     plt.show()
76.     #testing accuracy on test dataset
77.     from sklearn.metrics import accuracy_score
78.     y_test = pd.read_csv('Test.csv')
79.     labels = y_test["ClassId"].values
80.     imgs = y_test["Path"].values
81.     data=[]
82.     for img in imgs:
83.     image = Image.open(img)
84.     image = image.resize((30,30))
85.     data.append(np.array(image))
86.     X_test=np.array(data)
87.     pred = model.predict_classes(X_test)
88.     #Accuracy with the test data
89.     from sklearn.metrics import accuracy_score
90.     print(accuracy_score(labels, pred))
91.     model.save('traffic_classifier.h5')
```