**ASSIGNMENT 3**

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



### 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)
25. #sim = Image.fromarray(image)
26. data.append(image)
27. labels.append(i)
28. **except**:
29. print("Error loading image")
30. #Converting lists into numpy arrays
31. data = np.array(data)
32. labels = np.array(labels)
33. print(data.shape, labels.shape)
34. #Splitting training and testing dataset
35. X\_train, X\_test, y\_train, y\_test = train\_test\_split(data, labels, test\_size=0.2, random\_state=42)
36. print(X\_train.shape, X\_test.shape, y\_train.shape, y\_test.shape)
37. #Converting the labels into one hot encoding
38. y\_train = to\_categorical(y\_train, 43)
39. y\_test = to\_categorical(y\_test, 43)
40. #Building the model
41. model = Sequential()
42. model.add(Conv2D(filters=32, kernel\_size=(5,5), activation='relu', input\_shape=X\_train.shape[1:]))
43. model.add(Conv2D(filters=32, kernel\_size=(5,5), activation='relu'))
44. model.add(MaxPool2D(pool\_size=(2, 2)))
45. model.add(Dropout(rate=0.25))
46. model.add(Conv2D(filters=64, kernel\_size=(3, 3), activation='relu'))
47. model.add(Conv2D(filters=64, kernel\_size=(3, 3), activation='relu'))
48. model.add(MaxPool2D(pool\_size=(2, 2)))
49. model.add(Dropout(rate=0.25))
50. model.add(Flatten())
51. model.add(Dense(256, activation='relu'))
52. model.add(Dropout(rate=0.5))
53. model.add(Dense(43, activation='softmax'))
54. #Compilation of the model
55. model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])
56. epochs = 15
57. history = model.fit(X\_train, y\_train, batch\_size=32, epochs=epochs, validation\_data=(X\_test, y\_test))
58. model.save("my\_model.h5")
59. #plotting graphs for accuracy
60. 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’)