ASSIGNMENT 3

ASSIGNMENT DATE	**12 OCTOBER 2022**
TEAM ID	**PNT2022TMID17273**
PROJECT NAME	**A NOVEL METHOD FOR HANDWRITTEN DIGIT RECOGNITION SYSTEM**
MAXIMUM MARKS	**2 marks**

QUESTION: WRITE A PYTHON CODE FOR BLINKING LED AND TRAFFIC LIGHT FOR RASPBERRY PI.

PROJECT :

HANDWRITTEN DIGIT RECOGNITION SYSTEM

#Unzipping

#Data Augmentation

[ ]:

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_gen    = ImageDataGenerator(rescale        =1. / 255 ,
                                  zoom_range   =0.2 ,
                                  horizontal_flip      =True)
test_gen    = ImageDataGenerator(rescale        =1. / 255 )
```

[ ]:

```
xtrain    = train_gen   . flow_from_directory(       ' /content/flowers      ' ,
                                                      target_size     =( 64 , 64 ) ,
                                                      class_mode    =' categorical       ' ,
                                                      batch_size    =100 )
```

Found 4317 images belonging to 5 classes.

#Train

[ ]:

```
from tensorflow.keras.models import Sequential from tensorflow.keras.layers
import Convolution2D, MaxPooling2D, Dense, Flatten from keras.callbacks import
EarlyStopping, ReduceLROnPlateau
```

[ ]:

```
model = Sequential() model.add(Convolution2D(32,(3,3),activation='relu',input_shape=(64,64,3)))
model.add(MaxPooling2D((2,2))) model.add(Flatten())

model.add(Dense(400,activation='relu')) model.add(Dense(200,activation='relu'))
model.add(Dense(100,activation='relu')) model.add(Dense(5,activation='softmax'))
```

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

```
[ ]: early_stopping = EarlyStopping(monitor ='accuracy' ,
                              patience =3)
     reduce_lr = ReduceLROnPlateau(monitor ='accuracy' ,
                              patience =5,
                              factor =0.5 ,min_lr =0.00001 )

     callback = [ reduce_lr,early_stopping ]
```

```
[ ]: model . fit_generator(xtrain,
                        steps_per_epoch = len ( xtrain),
                        callbacks =callback,
                        epochs =100 )
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: UserWarning: `Model.fit_generator` is
deprecated and will be removed in a future version.
Please use `Model.fit`, which supports generators. after removing the
   cwd from sys.path.

Epoch 1/100
44/44 [==============================] - 31s 703ms/step - loss: 0.6074 accuracy: 0.7751 - lr:
0.0010
Epoch 2/100
44/44 [==============================] - 31s 702ms/step - loss: 0.5491 accuracy: 0.7973 - lr:
0.0010
Epoch 3/100
44/44 [==============================] - 31s 696ms/step - loss: 0.5417 accuracy: 0.8043 - lr:
0.0010
Epoch 4/100
44/44 [==============================] - 31s 693ms/step - loss: 0.4930 accuracy: 0.8156 - lr:
0.0010
Epoch 5/100
44/44 [==============================] - 31s 692ms/step - loss: 0.4616 accuracy: 0.8293 - lr:
0.0010
Epoch 6/100
44/44 [==============================] - 31s 695ms/step - loss: 0.4350 accuracy: 0.8392 - lr:
0.0010
Epoch 7/100
44/44 [==============================] - 31s 700ms/step - loss: 0.4190 accuracy: 0.8469 - lr:
0.0010
Epoch 8/100
44/44 [==============================] - 31s 692ms/step - loss: 0.3975 accuracy: 0.8568 - lr:
0.0010
Epoch 9/100
44/44 [==============================] - 31s 689ms/step - loss: 0.4207 accuracy: 0.8432 - lr:
0.0010
Epoch 10/100
44/44 [==============================] - 31s 696ms/step - loss: 0.3674 accuracy: 0.8687 - lr:
0.0010
Epoch 11/100
44/44 [==============================] - 31s 689ms/step - loss: 0.3267 accuracy: 0.8888 - lr:
```

```
0.0010
Epoch 12/100
44/44 [==============================] - 31s 699ms/step - loss: 0.3255 accuracy: 0.8877 - lr:
0.0010
Epoch 13/100
44/44 [==============================] - 31s 688ms/step - loss: 0.2975 accuracy: 0.8999 - lr:
0.0010
Epoch 14/100
44/44 [==============================] - 31s 690ms/step - loss: 0.3144 accuracy: 0.8927 - lr:
0.0010
Epoch 15/100
44/44 [==============================] - 31s 697ms/step - loss: 0.2544 accuracy: 0.9120 - lr:
0.0010
Epoch 16/100
44/44 [==============================] - 31s 694ms/step - loss: 0.2986 accuracy: 0.8939 - lr:
0.0010
Epoch 17/100
44/44 [==============================] - 31s 699ms/step - loss: 0.2570 accuracy: 0.9097 - lr:
0.0010
Epoch 18/100
44/44 [==============================] - 31s 691ms/step - loss: 0.2282 accuracy: 0.9205 - lr:
0.0010
Epoch 19/100
44/44 [==============================] - 31s 697ms/step - loss: 0.2180 accuracy: 0.9226 - lr:
0.0010
Epoch 20/100
44/44 [==============================] - 31s 694ms/step - loss: 0.1995 accuracy: 0.9314 - lr:
0.0010
Epoch 21/100
44/44 [==============================] - 31s 690ms/step - loss: 0.1805 accuracy: 0.9340 - lr:
0.0010
Epoch 22/100
44/44 [==============================] - 31s 697ms/step - loss: 0.1986 accuracy: 0.9307 - lr:
0.0010
Epoch 23/100
44/44 [==============================] - 31s 693ms/step - loss: 0.2023 accuracy: 0.9287 - lr:
0.0010
Epoch 24/100
44/44 [==============================] - 31s 695ms/step - loss: 0.1646 accuracy: 0.9465 - lr:
0.0010
Epoch 25/100
44/44 [==============================] - 31s 693ms/step - loss: 0.1727 accuracy: 0.9407 - lr:
0.0010
Epoch 26/100
44/44 [==============================] - 32s 719ms/step - loss: 0.1285 accuracy: 0.9569 - lr:
0.0010
Epoch 27/100
44/44 [==============================] - 31s 703ms/step - loss: 0.1705 accuracy: 0.9400 - lr:
0.0010
Epoch 28/100
44/44 [==============================] - 31s 709ms/step - loss: 0.1531 accuracy: 0.9439 - lr:
0.0010
```

```
Epoch 29/100
44/44 [==============================] - 32s 711ms/step - loss: 0.1413 accuracy: 0.9539 - lr:
0.0010
```

```
[ ]: <keras.callbacks.History at 0x7efc457f6e50>
```

```
[]:  model . save( ' flower_cnn.h5     ' )
```

#Test

```
[]:  import numpy as np
     from tensorflow.keras.preprocessing import image
```

```
[]:  img  =  image . load_img(   ' /content/test image.jpg          ' ,target_size      =( 64 , 64 ))
     img
```

```
[]:
```

```
[]:  h  =  image . img_to_array(img)
     h
```

```
[ ]: array([[[1., 1., 1.], [1., 1., 1.],
                 [1., 1., 1.],
                 ...,
                 [1., 1., 1.],
                 [1., 1., 1.],
                 [1., 1., 1.]],

                [[1., 1., 1.],
                 [1., 1., 1.],
                 [1., 1., 1.],
                 ...,
                     [1., 1., 1.],
                     [1., 1., 1.],
                     [1., 1., 1.]],

                [[1., 1., 1.],
                     [1., 1., 1.],
                     [1., 1., 1.],
                 ...,
                     [1., 1., 1.],
                     [1., 1., 1.],
                     [1., 1., 1.]],

                ...,

                [[1., 1., 1.],
            [1., 1., 1.],
            [1., 1., 1.],
        ...,
            [1., 1., 1.],
            [1., 1., 1.],
            [1., 1., 1.]],

       [[1., 1., 1.],
            [1., 1., 1.],
            [1., 1., 1.],
        ...,
            [1., 1., 1.],
            [1., 1., 1.],
            [1., 1., 1.]],

       [[1., 1., 1.],
            [1., 1., 1.],
            [1., 1., 1.],
        ...,
            [1., 1., 1.],
            [1., 1., 1.],
        [1., 1., 1.]]], dtype=float32)
```

[ ]:

```
h= np . expand_dims(h,axis = 0)
h
```

```
[ ]: array([[[[1., 1., 1.], [1., 1., 1.],
                  [1., 1., 1.],
              ...,
                  [1., 1., 1.],
                  [1., 1., 1.],
              [1., 1., 1.]],

            [[1., 1., 1.],
              [1., 1., 1.],
              [1., 1., 1.],
              ...,
              [1., 1., 1.],
              [1., 1., 1.],
              [1., 1., 1.]],

            [[1., 1., 1.],
              [1., 1., 1.],
              [1., 1., 1.],
              ...,
              [1., 1., 1.],
              [1., 1., 1.],
              [1., 1., 1.]],

            ...,
        [[1., 1., 1.],
         [1., 1., 1.],
         [1., 1., 1.],
         ...,
         [1., 1., 1.],
         [1., 1., 1.],
         [1., 1., 1.]],

        [[1., 1., 1.],
         [1., 1., 1.],
         [1., 1., 1.],
         ...,
         [1., 1., 1.],
         [1., 1., 1.],
         [1., 1., 1.]],

        [[1., 1., 1.],
         [1., 1., 1.],
         [1., 1., 1.],
         ...,
         [1., 1., 1.],
         [1., 1., 1.],
         [1., 1., 1.]]]], dtype=float32)
```

[ ]:

```
val = list(xtrain.class_indices.keys())
val
```

[ ]: ['daisy', 'dandelion', 'rose', 'sunflower', 'tulip']

[ ]:

```
val[np.argmax(model.predict(h))]
```

[ ]: 'rose'