# Assignment -3

## Convolutional Neural Networks

| | |
|---|---|
| Date | 16 October 2022 |
| Team ID | PNT2022TMID39355 |
| Project Name | A Novel Method for Handwritten Digit Recognition System |
| Maximum Marks | 2 Marks |

**#Import necessary libraries**

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras.layers import Convolution2D

from tensorflow.keras.layers import MaxPooling2D

from tensorflow.keras.layers import Flatten

**#Image augmentation**

from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True, vertical_flip=True)

test_datagen = ImageDataGenerator(rescale=1./255)

```
In [1]: #Import necessary libraries
        from tensorflow.keras.models import Sequential
        from tensorflow.keras.layers import Dense
        from tensorflow.keras.layers import Convolution2D
        from tensorflow.keras.layers import MaxPooling2D
        from tensorflow.keras.layers import Flatten

In [2]: #Image augmentation
        from tensorflow.keras.preprocessing.image import ImageDataGenerator
        train_datagen = ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True,vertical_flip=True)
        test_datagen = ImageDataGenerator(rescale=1./255)
```

**#data set**

x_train = train_datagen.flow_from_directory(r"E:\Flowers\Training",target_size=(128,128),batch_size =32,class_mode="categorical")

x_test =
test_datagen.flow_from_directory(r"E:\Flowers\Testing",target_size=(128,128),batch_size=32,class_mode="categorical")

x_train.class_indices

model = Sequential()

```
In [2]: #Image augmentation
        from tensorflow.keras.preprocessing.image import ImageDataGenerator
        train_datagen = ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True,vertical_flip=True)
        test_datagen = ImageDataGenerator(rescale=1./255)

In [3]: x_train = train_datagen.flow_from_directory(r"E:\Flowers\Training",target_size=(128,128),batch_size=32,class_mode="categorical")
        x_test = test_datagen.flow_from_directory(r"E:\Flowers\Testing",target_size=(128,128),batch_size=32,class_mode="categorical")
        x_train.class_indices

        Found 3003 images belonging to 5 classes.
        Found 1325 images belonging to 5 classes.

Out[3]: {'daisy': 0, 'dandelion': 1, 'rose': 2, 'sunflower': 3, 'tulip': 4}
```

**#Add layers**

**#Convolution layer**

model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu'))

**#Maxpooling layer**

model.add(MaxPooling2D(pool_size=(2,2)))

**#flatten layer**

model.add(Flatten())

**#hidden layer**

model.add(Dense(units=300,kernel_initializer="random_uniform",activation="relu"))

model.add(Dense(units=200,kernel_initializer="random_uniform",activation="relu"))

model.add(Dense(units=5,kernel_initializer="random_uniform",activation="softmax"))

model.summary()

In [4]:
```
model = Sequential()
#Add layers
#Convolution layer
model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu'))
#Maxpooling layer
model.add(MaxPooling2D(pool_size=(2,2)))
#flatten layer
model.add(Flatten())
#hidden layer
model.add(Dense(units=300,kernel_initializer="random_uniform",activation="relu"))
model.add(Dense(units=200,kernel_initializer="random_uniform",activation="relu"))
model.add(Dense(units=5,kernel_initializer="random_uniform",activation="softmax"))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 126, 126, 32)      896

 max_pooling2d (MaxPooling2D  (None, 63, 63, 32)       0
 )

 flatten (Flatten)           (None, 127008)            0

 dense (Dense)               (None, 300)               38102700

 dense_1 (Dense)             (None, 200)               60200

 dense_2 (Dense)             (None, 5)                 1005

=================================================================
Total params: 38,164,801
Trainable params: 38,164,801
Non-trainable params: 0
_________________________________________________________________
```

**#compile the model**

model.compile(loss="categorical_crossentropy",optimizer="adam",metrics=["accuracy"])

**#Fit the model**

model.fit_generator(x_train,steps_per_epoch=75,epochs=15,validation_data=x_test,validation_steps=80)

```
In [6]: #compile the model
model.compile(loss="categorical_crossentropy",optimizer="adam",metrics=["accuracy"])
#Fit the model
model.fit_generator(x_train,steps_per_epoch=75,epochs=15,validation_data=x_test,validation_steps=80)
```

```
C:\Users\hp\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: UserWarning: `Model.fit_generator` is deprecated and will be r
emoved in a future version. Please use `Model.fit`, which supports generators.
  after removing the cwd from sys.path.
Epoch 1/15
75/75 [==============================] - ETA: 0s - loss: 1.0726 - accuracy: 0.5791WARNING:tensorflow:Your input ran out of dat
a; interrupting training. Make sure that your dataset or generator can generate at least `steps_per_epoch * epochs` batches (in
this case, 80 batches). You may need to use the repeat() function when building your dataset.
75/75 [==============================] - 83s 1s/step - loss: 1.0726 - accuracy: 0.5791 - val_loss: 1.2372 - val_accuracy: 0.504
9
Epoch 2/15
75/75 [==============================] - 68s 906ms/step - loss: 0.9907 - accuracy: 0.6125
Epoch 3/15
75/75 [==============================] - 69s 917ms/step - loss: 0.8981 - accuracy: 0.6489
Epoch 4/15
75/75 [==============================] - 70s 922ms/step - loss: 0.8850 - accuracy: 0.6522
Epoch 5/15
75/75 [==============================] - 73s 962ms/step - loss: 0.8177 - accuracy: 0.6789
Epoch 6/15
75/75 [==============================] - 75s 997ms/step - loss: 0.8101 - accuracy: 0.6917
Epoch 7/15
75/75 [==============================] - 73s 966ms/step - loss: 0.8099 - accuracy: 0.6868
Epoch 8/15
75/75 [==============================] - 72s 957ms/step - loss: 0.7574 - accuracy: 0.7229
Epoch 9/15
75/75 [==============================] - 70s 926ms/step - loss: 0.7146 - accuracy: 0.7215
Epoch 10/15
```

```
75/75 [==============================] - 69s 911ms/step - loss: 0.6867 - accuracy: 0.7446
Epoch 11/15
75/75 [==============================] - 69s 920ms/step - loss: 0.6735 - accuracy: 0.7404
Epoch 12/15
75/75 [==============================] - 70s 931ms/step - loss: 0.6735 - accuracy: 0.7562
Epoch 13/15
75/75 [==============================] - 69s 919ms/step - loss: 0.6310 - accuracy: 0.7595
Epoch 14/15
75/75 [==============================] - 75s 995ms/step - loss: 0.6296 - accuracy: 0.7724
Epoch 15/15
75/75 [==============================] - 75s 988ms/step - loss: 0.6024 - accuracy: 0.7775
```

**#Save the model**

model.save("flower.h5")

from tensorflow.keras.models import load_model

from tensorflow.keras.preprocessing import image

import numpy as np

model = load_model("Flower.h5")

```
In [7]: #Save the model
        model.save("flower.h5")
```

**#Test the model:**

img = image.load_img(r"C:\Users\hp\Downloads\rose.jpg",target_size=(128,128))

img

type(img)

x = image.img_to_array(img)

x

x.shape

x = np.expand_dims(x,axis=0)

x.shape

pred_prob = model.predict(x)

pred_prob

```
In [8]: from tensorflow.keras.models import load_model
        from tensorflow.keras.preprocessing import image
        import numpy as np
        model = load_model("Flower.h5")

In [10]: #Testing with the image
         img = image.load_img(r"C:\Users\hp\Downloads\rose.jpg",target_size=(128,128))
         img
         type(img)
Out[10]: PIL.Image.Image

In [11]: x = image.img_to_array(img)
         x
         x.shape
         x = np.expand_dims(x,axis=0)
         x.shape
Out[11]: (1, 128, 128, 3)

In [12]: pred_prob = model.predict(x)
         pred_prob
Out[12]: array([[0., 0., 1., 0., 0.]], dtype=float32)
```

class_name = ["daisy","dandelion","rose","sunfower","tulip"]

pred_id = pred_prob.argmax(axis=1)[0]

pred_id

print("Predicted flower is",str(class_name[pred_id]))

In [13]:
```
class_nane = [”daisy’,”dandelion”,”rose’,”sunfower’,”tulip”]
```

In [15]:
```
pred_id = predgrob.argmax(axis=1)[B]
pred_1d
print(’Predicted flower is’,str(class_name[pred_id]))
```

```
Predicted flower is rose
```