ASSIGNMENT 3

ASSIGNMENT DATE	01 OCTOBER 2020
STUDENT NAME	SHAKEEL MOHAMMED G
STUDENT ROLL NUMBER	2019504584
MAXIMUM MARKS	2 MARKS

Problem Statement:- Build CNN Model for Classification Of Flowers

Perform Below Tasks to complete the assignment: -

1. Download the Dataset
2. Image Augmentation
3. Create Model
4. Add Layers (Convolution, MaxPooling, Flatten,Dense-(Hidden Layers), Output)
5. Compile The Model
6. Fit The Model
7. Save The Model
8. Test The Model

In [ ]:
```python
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [ ]:
```python
!pip install split_folders
import splitfolders

input_folder = "/content/drive/MyDrive/IBM/Flowers" #Enter Input Folder
output = "/content/drive/MyDrive/IBM/Flowers/data" #Enter Output Folder

splitfolders.ratio(input_folder, output=output, seed=42, ratio=(0.8,0.2))
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheel
s/public/simple/
Requirement already satisfied: split_folders in /usr/local/lib/python3.7/dist-pack
ages (0.5.1)
Copying files: 4317 files [02:04, 34.67 files/s]
```

Importing the libraries

In [ ]:
```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

In [ ]:
```python
from tensorflow.keras.layers import Convolution2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Flatten
```

In [ ]:
```
#import the preprocess library of image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

Image Augmentation

In [ ]:
```
train_datagen = ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,ho
#rescale =  pixel value rescaling to 0 to 1 from 0 to 255
#shear_range => counter clock wise rotation(anti clock)
```

In [ ]:
```
test_datagen  = ImageDataGenerator(rescale=1./255)
```

In [ ]:
```
#load your images data
```

In [ ]:
```
#load your images data
```

In [ ]:
```
x_train = train_datagen.flow_from_directory('/content/drive/MyDrive/IBM/Flowers/dat
```

```
Found 3452 images belonging to 5 classes.
```

In [ ]:
```
x_test = test_datagen.flow_from_directory(r"/content/drive/MyDrive/IBM/Flowers/data
```

```
Found 865 images belonging to 5 classes.
```

In [ ]:
```
x_train.class_indices
```

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

Create Model

In [ ]:
```
#initialize the model
model = Sequential()
```

Add Layers (Convolution,MaxPooling,Flatten,Dense-(Hidden Layers),Output)

In [ ]:
```
#add convlution layer
model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu'))
# 32 => no of feature detectors
#(3,3)=> kernel size(feature detector size => 3*3 matrix)
```

In [ ]:
```
#add maxpooling layer
model.add(MaxPooling2D(pool_size=(2,2)))
```

In [ ]:
```
# you can add more convolutiona and pooling layers
model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
```

In [ ]:
```
#flatten layer => input layer to your ANN
model.add(Flatten())
```

In [ ]:
```
#hidden layers
model.add(Dense(units=500,kernel_initializer="random_uniform",activation="relu"))
model.add(Dense(units=200,kernel_initializer="random_uniform",activation="relu"))
model.add(Dense(units=300,kernel_initializer="random_uniform",activation="relu"))
model.add(Dense(units=400,kernel_initializer="random_uniform",activation="relu"))
```

In [ ]:
```
#output layer
model.add(Dense(units=5,kernel_initializer="random_uniform",activation="softmax"))
```

Compile The Model

In [ ]:
```
#compile the model
model.compile(loss="categorical_crossentropy",optimizer="adam",metrics=["accuracy"]
```

Fit The Model

In [ ]:
```
#train the model
model.fit(x_train,steps_per_epoch=len(x_train),epochs=10,validation_data=x_test,val
#steps_per_epoch = no of train images/batch size
#validation_steps = no of test images/batch size
```

```
Epoch 1/10
35/35 [==============================] - 106s 3s/step - loss: 1.4203 - accuracy:
0.3543 - val_loss: 1.1715 - val_accuracy: 0.4775
Epoch 2/10
35/35 [==============================] - 99s 3s/step - loss: 1.2220 - accuracy: 0.
4615 - val_loss: 1.1513 - val_accuracy: 0.4902
Epoch 3/10
35/35 [==============================] - 100s 3s/step - loss: 1.1233 - accuracy:
0.5159 - val_loss: 1.0047 - val_accuracy: 0.5653
Epoch 4/10
35/35 [==============================] - 98s 3s/step - loss: 1.0440 - accuracy: 0.
5744 - val_loss: 0.9395 - val_accuracy: 0.6266
Epoch 5/10
35/35 [==============================] - 100s 3s/step - loss: 1.0361 - accuracy:
0.5759 - val_loss: 0.9530 - val_accuracy: 0.6162
Epoch 6/10
35/35 [==============================] - 99s 3s/step - loss: 1.0050 - accuracy: 0.
6037 - val_loss: 0.8916 - val_accuracy: 0.6439
Epoch 7/10
35/35 [==============================] - 101s 3s/step - loss: 0.9490 - accuracy:
0.6295 - val_loss: 0.8843 - val_accuracy: 0.6416
Epoch 8/10
35/35 [==============================] - 98s 3s/step - loss: 0.8924 - accuracy: 0.
6550 - val_loss: 0.8537 - val_accuracy: 0.6590
Epoch 9/10
35/35 [==============================] - 99s 3s/step - loss: 0.8786 - accuracy: 0.
6602 - val_loss: 0.8756 - val_accuracy: 0.6451
Epoch 10/10
35/35 [==============================] - 101s 3s/step - loss: 0.8491 - accuracy:
0.6790 - val_loss: 0.8410 - val_accuracy: 0.6486
```

Out[ ]:
```
<keras.callbacks.History at 0x7fc4e65a0310>
```

Save The Model

```
In [ ]: model.save("/flowers.h5")
```

Test The Model

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

```
In [ ]: model = load_model("/flowers.h5")
```

```
In [ ]: img = image.load_img("/content/drive/MyDrive/IBM/sunflower.jpg",target_size=(128,12
```

```
In [ ]: img
```

```
Out[ ]:
```

```
In [ ]: x = image.img_to_array(img)
```

```
In [ ]: x
```

```
Out[ ]: array([[[ 94., 127.,  56.],
                [ 92., 125.,  54.],
                [ 90., 123.,  52.],
                ...,
                [ 96., 128.,  52.],
                [104., 135.,  59.],
                [112., 140.,  65.]],

               [[106., 133.,  64.],
                [109., 136.,  67.],
                [109., 136.,  67.],
                ...,
                [101., 132.,  54.],
                [111., 139.,  62.],
                [115., 142.,  65.]],

               [[129., 150.,  85.],
                [130., 151.,  86.],
                [132., 153.,  88.],
                ...,
                [108., 137.,  53.],
                [112., 141.,  59.],
                [120., 144.,  66.]],

               ...,

               [[141., 159., 111.],
                [134., 153.,  98.],
                [125., 145.,  86.],
                ...,
                [ 62.,  96.,   2.],
                [ 55.,  88.,   7.],
                [ 48.,  82.,   8.]],

               [[141., 158., 113.],
                [138., 155., 110.],
                [132., 150., 102.],
                ...,
                [ 62.,  96.,   2.],
                [ 55.,  88.,   7.],
                [ 47.,  81.,   7.]],

               [[133., 152., 106.],
                [128., 150., 101.],
                [116., 140.,  88.],
                ...,
                [ 61.,  94.,   3.],
                [ 57.,  89.,   6.],
                [ 50.,  80.,  10.]]], dtype=float32)
```

```
In [ ]: x.shape
```

```
Out[ ]: (128, 128, 3)
```

```
In [ ]: #(1,64,64,3) to expand the dims
```

```
In [ ]: x = np.expand_dims(x,axis=0)
        x.shape
```

```
Out[ ]: (1, 128, 128, 3)
```

In [ ]:
```
pred_prob = model.predict(x)
```
```
1/1 [==============================] - 0s 168ms/step
```
In [ ]:
```
pred_prob
```
Out[ ]:
```
array([[0., 0., 0., 1., 0.]], dtype=float32)
```
In [ ]:
```
class_name=["DAISY","DANDELION",'ROSE','SUNFLOWER',"TULIP",]
pred_id = pred_prob.argmax(axis=1)[0]
```
In [ ]:
```
pred_id
```
Out[ ]:
```
3
```
In [ ]:
```
print("Predicted Flower is",str(class_name[pred_id]))
```
```
Predicted Flower is SUNFLOWER
```