IBM NALAIYA THIRAN

Assignment -3

Team ID	PNT2022TMID33638
Project Name	Real-Time Communication System Powered by AI For Specially Abled
Student Name	Sivaranjani.A
Student Roll Number	922519106153
Maximum Marks	2 Marks

Import all necessary libraries:

```
!pip install split-folders
import splitfolders
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting split-folders
  Downloading split_folders-0.5.1-py3-none-any.whl (8.4 kB)
Installing collected packages: split-folders
Successfully installed split-folders-0.5.1
```

```
import numpy as np
```

1. Download and Load Dataset:

```
from google.colab import drive
drive.mount('/content/drive')
```

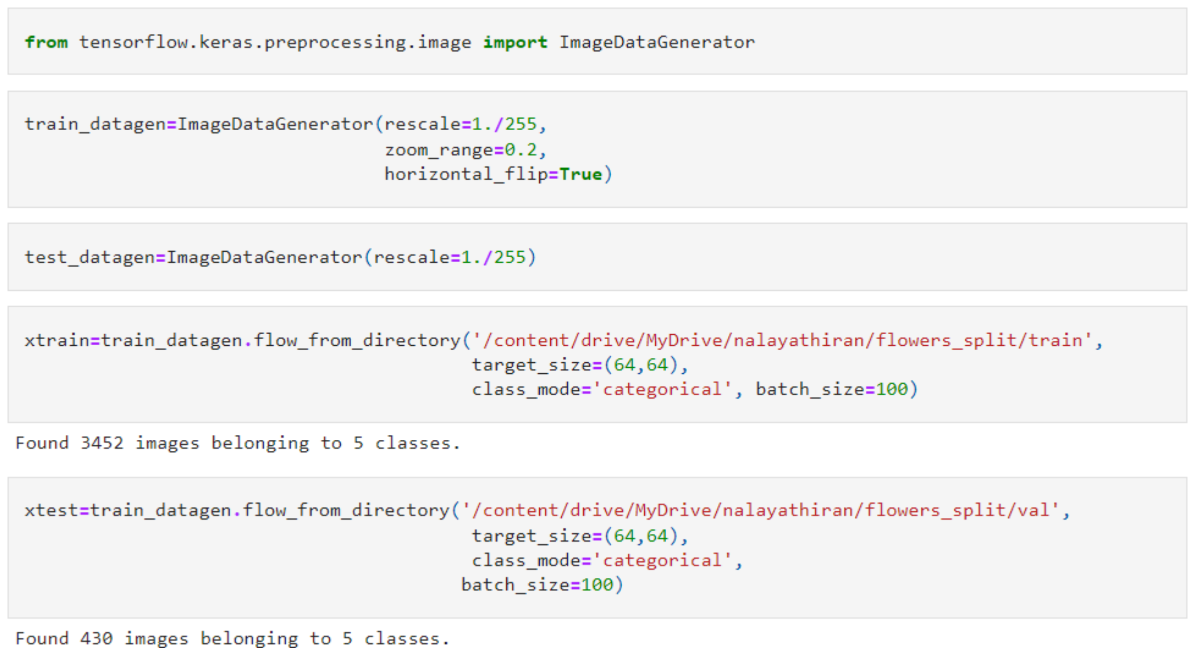
```
Mounted at /content/drive
```

Split Dataset to Training Data, Validation Data and Testing Data

```
splitfolders.ratio('/content/drive/MyDrive/nalayathiran/flowers', output="/content/drive/MyDrive/nalayathir
```

```
Copying files: 4317 files [01:37, 44.32 files/s]
```

2. Image Augmentation:



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

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

```
test_datagen=ImageDataGenerator(rescale=1./255)
```

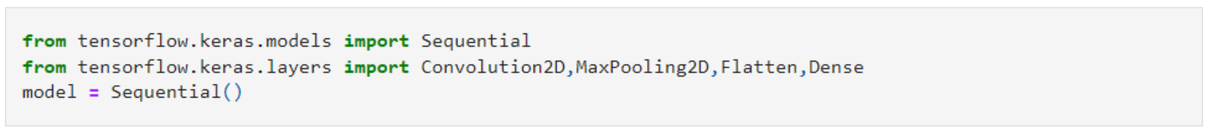
```
xtrain=train_datagen.flow_from_directory('/content/drive/MyDrive/nalayathiran/flowers_split/train',
                                         target_size=(64,64),
                                         class_mode='categorical', batch_size=100)
```

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

```
xtest=train_datagen.flow_from_directory('/content/drive/MyDrive/nalayathiran/flowers_split/val',
                                        target_size=(64,64),
                                        class_mode='categorical',
                                       batch_size=100)
```

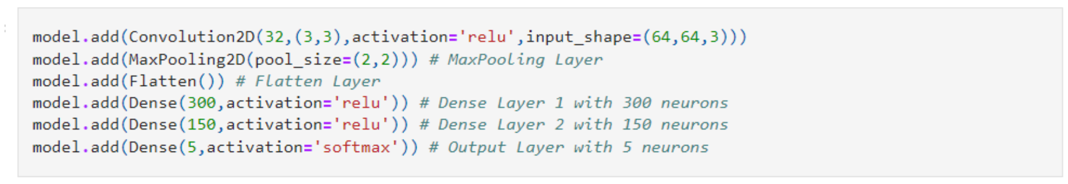
```
Found 430 images belonging to 5 classes.
```

3. Create Model:



```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense
model = Sequential()
```

4. Add Layers (Convolution, Max Pooling, Flatten, Dense- (Hidden Layers), Output):



```
model.add(Convolution2D(32,(3,3),activation='relu',input_shape=(64,64,3)))
model.add(MaxPooling2D(pool_size=(2,2))) # MaxPooling Layer
model.add(Flatten()) # Flatten Layer
model.add(Dense(300,activation='relu')) # Dense Layer 1 with 300 neurons
model.add(Dense(150,activation='relu')) # Dense Layer 2 with 150 neurons
model.add(Dense(5,activation='softmax')) # Output Layer with 5 neurons
```

5. Compile the Model:

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

```
model.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_8 (Conv2D)           (None, 62, 62, 32)        896       
                                                                 
 max_pooling2d_5 (MaxPooling  (None, 31, 31, 32)       0         
 2D)                                                             
                                                                 
 flatten_5 (Flatten)         (None, 30752)             0         
                                                                 
 dense_15 (Dense)            (None, 300)               9225900   
                                                                 
 dense_16 (Dense)            (None, 150)               45150     
                                                                 
 dense_17 (Dense)            (None, 5)                 755       
                                                                 
=================================================================
Total params: 9,272,701
Trainable params: 9,272,701
Non-trainable params: 0
_________________________________________________________________
```

6. Fit the Model:

```
model.fit_generator(xtrain,steps_per_epoch=len(xtrain),epochs=100,validation_data=xtest,validation_steps=le
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: UserWarning: `Model.fit_generator` is depre
cated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  """Entry point for launching an IPython kernel.
Epoch 1/100
35/35 [==============================] - 16s 462ms/step - loss: 0.9243 - accuracy: 0.6443 - val_loss: 0.965
1 - val_accuracy: 0.6372
Epoch 2/100
35/35 [==============================] - 17s 486ms/step - loss: 0.8995 - accuracy: 0.6521 - val_loss: 0.983
3 - val_accuracy: 0.6209
Epoch 3/100
35/35 [==============================] - 18s 506ms/step - loss: 0.8339 - accuracy: 0.6857 - val_loss: 0.905
7 - val_accuracy: 0.6581
Epoch 4/100
35/35 [==============================] - 16s 462ms/step - loss: 0.7864 - accuracy: 0.6979 - val_loss: 0.938
5 - val_accuracy: 0.6628
Epoch 5/100
35/35 [==============================] - 18s 504ms/step - loss: 0.7368 - accuracy: 0.7167 - val_loss: 0.883
1 - val_accuracy: 0.6953
Epoch 6/100
35/35 [==============================] - 17s 483ms/step - loss: 0.7570 - accuracy: 0.7065 - val_loss: 0.914
7 - val_accuracy: 0.6721
Epoch 7/100
35/35 [==============================] - 16s 468ms/step - loss: 0.6869 - accuracy: 0.7416 - val_loss: 0.878
6 - val_accuracy: 0.6651
Epoch 8/100
35/35 [==============================] - 17s 490ms/step - loss: 0.6629 - accuracy: 0.7520 - val_loss: 0.910
4 - val_accuracy: 0.6558
Epoch 9/100
35/35 [==============================] - 16s 467ms/step - loss: 0.6273 - accuracy: 0.7683 - val_loss: 0.904
6 - val_accuracy: 0.6884
Epoch 10/100
35/35 [==============================] - 16s 463ms/step - loss: 0.5810 - accuracy: 0.7856 - val_loss: 0.924
8 - val_accuracy: 0.6907
Epoch 11/100
35/35 [==============================] - 16s 466ms/step - loss: 0.6014 - accuracy: 0.7749 - val_loss: 0.905
7 - val_accuracy: 0.6651
Epoch 12/100
35/35 [==============================] - 16s 462ms/step - loss: 0.5347 - accuracy: 0.8021 - val_loss: 0.859
7 - val_accuracy: 0.7000
```

7. Save The Model:

```
# Model was trained with accuracy of 98%
model.save('/content/drive/MyDrive/nalayathiran/Flowers.h5')
```

8. Test The Model:

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

```
# Loading the Test Image
img=image.load_img('/content/drive/MyDrive/nalayathiran/flowers_split/test/rose/18486124712_17ebe7559b_n.jp
img # Image belonging to the class label Rose
```

```
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0) # Adding extra dimension to image as it is in RGB
model.predict(x)
```

```
array([[0., 0., 1., 0., 0.]], dtype=float32)
```

```
op=['Daisy','Dandelion','Rose','Sunflower','Tulip']
pred=np.argmax(model.predict(x))
# Predicting the output
op[pred]
```

```
'Rose'
```