```
from google.colab import drive

drive.mount('/content/drive')

Mounted at /content/drive
```

Image Augmentation

```
# Importing Library

from tensorflow.keras.preprocessing.image import ImageDataGenerator

# expanding training and testing variable

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

test_d=ImageDataGenerator(rescale=1./255)

#Data augmentation on testing data

vtrain =
train_d.flow_from_directory('/content/drive/MyDrive/flowers/Testing',target_size=(76,76),class_mode='categorical',batch_size=200)

Found 4334 images belonging to 5 classes.

#Data augmentation on training data

vtest =
test_d.flow_from_directory('/content/drive/MyDrive/flowers/Training',target_size=(76,76),class_mode='categorical',batch_size=200)

Found 4372 images belonging to 5 classes.
```

Creating CNN Model

```
#Importing Libraries

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Convolution2D, MaxPooling2D, Flatten, Dense

#Building a CNN block

model = Sequential()

model.add(Convolution2D(32,(3,3),activation='relu',input_shape=(76,76,3)))

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

model.add(Flatten())

model.add(Dense(500,activation='relu'))

model.add(Dense(250,activation='relu'))

model.add(Dense(5,activation='softmax'))

#Compiling the model
```

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

#Fittting the model


model.fit_generator(vtrain,steps_per_epoch=len(vtrain),epochs=15,validation_data=vtest,validation_steps=len(vtest))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  This is separate from the ipykernel package so we can avoid doing imports until
Epoch 1/15
22/22 [==============================] - 1419s 66s/step - loss: 2.4632 - accuracy: 0.2995 - val_loss: 1.2851 - val_accuracy: 0.4376
Epoch 2/15
22/22 [==============================] - 64s 3s/step - loss: 1.2076 - accuracy: 0.5074 - val_loss: 1.1575 - val_accuracy: 0.5320
Epoch 3/15
22/22 [==============================] - 62s 3s/step - loss: 1.0800 - accuracy: 0.5743 - val_loss: 1.0376 - val_accuracy: 0.5958
Epoch 4/15
22/22 [==============================] - 63s 3s/step - loss: 0.9855 - accuracy: 0.6214 - val_loss: 0.9492 - val_accuracy: 0.6414
Epoch 5/15
22/22 [==============================] - 63s 3s/step - loss: 0.8937 - accuracy: 0.6622 - val_loss: 0.9133 - val_accuracy: 0.6530
Epoch 6/15
22/22 [==============================] - 63s 3s/step - loss: 0.8337 - accuracy: 0.6751 - val_loss: 0.7866 - val_accuracy: 0.7091
Epoch 7/15
22/22 [==============================] - 63s 3s/step - loss: 0.7875 - accuracy: 0.7037 - val_loss: 0.7907 - val_accuracy: 0.7100
Epoch 8/15
22/22 [==============================] - 65s 3s/step - loss: 0.7410 - accuracy: 0.7220 - val_loss: 0.6903 - val_accuracy: 0.7434
Epoch 9/15
```

22/22 [==============================] - 65s 3s/step - loss: 0.7011 - accuracy: 0.7323 - val_loss: 0.6207 - val_accuracy: 0.7699

Epoch 10/15

22/22 [==============================] - 66s 3s/step - loss: 0.6562 - accuracy: 0.7575 - val_loss: 0.6067 - val_accuracy: 0.7793

Epoch 11/15

22/22 [==============================] - 63s 3s/step - loss: 0.6345 - accuracy: 0.7637 - val_loss: 0.7020 - val_accuracy: 0.7381

Epoch 12/15

22/22 [==============================] - 63s 3s/step - loss: 0.6324 - accuracy: 0.7649 - val_loss: 0.5490 - val_accuracy: 0.8008

Epoch 13/15

22/22 [==============================] - 63s 3s/step - loss: 0.6061 - accuracy: 0.7695 - val_loss: 0.5225 - val_accuracy: 0.8118

Epoch 14/15

22/22 [==============================] - 65s 3s/step - loss: 0.5382 - accuracy: 0.8032 - val_loss: 0.4787 - val_accuracy: 0.8255

Epoch 15/15

22/22 [==============================] - 66s 3s/step - loss: 0.5271 - accuracy: 0.8050 - val_loss: 0.5410 - val_accuracy: 0.8001

<keras.callbacks.History at 0x7f9e94a95e90>

save model

model.save('flowers.h5')

Testing model

from tensorflow.keras.preprocessing import image

import numpy as np

Testing 1.1(daisy)

img = image.load_img('/content/drive/MyDrive/flowers/Testing/daisy/10993818044_4c19b86c82.jpg',target_size=(76,76))

x = image.img_to_array(img)

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

prediction = np.argmax(model.predict(x))

```
op = ['daisy','dandelion','rose','sunflower','tulip']

op[prediction]

'daisy'

# Testing 1.2(daisy)


img =
image.load_img('/content/drive/MyDrive/flowers/Testing/daisy/525780443_bba812c26a_m.jpg',tar
get_size=(76,76))

x = image.img_to_array(img)

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

prediction = np.argmax(model.predict(x))

op = ['daisy','dandelion','rose','sunflower','tulip']

op[prediction]

'daisy'

# Testing 2.1(dandelion)


img =
image.load_img('/content/drive/MyDrive/flowers/Testing/dandelion/1195255751_d58b3d3076.jpg'
,target_size=(76,76))

x = image.img_to_array(img)

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

prediction = np.argmax(model.predict(x))

op = ['daisy','dandelion','rose','sunflower','tulip']

op[prediction]

'sunflower'

# Testing 2.2(dandelion)


img =
image.load_img('/content/drive/MyDrive/flowers/Testing/dandelion/1297972485_33266a18d9.jpg',
target_size=(76,76))

x = image.img_to_array(img)

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

prediction = np.argmax(model.predict(x))
```

```
op = ['daisy','dandelion','rose','sunflower','tulip']

op[prediction]

'daisy'

# Testing 3.1(rose)


img =
image.load_img('/content/drive/MyDrive/flowers/Testing/rose/7456887736_54e4ebac03_n.jpg',tar
get_size=(76,76))

x = image.img_to_array(img)

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

prediction = np.argmax(model.predict(x))

op = ['daisy','dandelion','rose','sunflower','tulip']

op[prediction]

'rose'

# Testing 3.2(rose)


img =
image.load_img('/content/drive/MyDrive/flowers/Testing/rose/33411423082_8150d9254e_n.jpg',t
arget_size=(76,76))

x = image.img_to_array(img)

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

prediction = np.argmax(model.predict(x))

op = ['daisy','dandelion','rose','sunflower','tulip']

op[prediction]

'rose'

# Testing 4.1(sunflower)


img =
image.load_img('/content/drive/MyDrive/flowers/Testing/sunflower/7012364067_5ffc7654c9_m.jp
g',target_size=(76,76))

x = image.img_to_array(img)

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

prediction = np.argmax(model.predict(x))
```

```
op = ['daisy','dandelion','rose','sunflower','tulip']

op[prediction]

'sunflower'

# Testing 4.2(sunflower)


img =
image.load_img('/content/drive/MyDrive/flowers/Testing/sunflower/2720698862_486d3ec079_m.j
pg',target_size=(76,76))

x = image.img_to_array(img)

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

prediction = np.argmax(model.predict(x))

op = ['daisy','dandelion','rose','sunflower','tulip']

op[prediction]

'sunflower'

# Testing 5.1(tulip)


img =
image.load_img('/content/drive/MyDrive/flowers/Testing/tulip/8892851067_79242a7362_n.jpg',tar
get_size=(76,76))

x = image.img_to_array(img)

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

prediction = np.argmax(model.predict(x))

op = ['daisy','dandelion','rose','sunflower','tulip']

op[prediction]

'tulip'

# Testing 5.2(tulip)


img =
image.load_img('/content/drive/MyDrive/flowers/Testing/tulip/5546723510_39a5a10d3a_n.jpg',tar
get_size=(76,76))

x = image.img_to_array(img)

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

prediction = np.argmax(model.predict(x))
```

```
op = ['daisy','dandelion','rose','sunflower','tulip']

op[prediction]

'tulip'
```