```
!pip install tensorflow
```

```
Looking in indexes: https://pypi.org/simple, https://us-
python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: tensorflow in
/usr/local/lib/python3.7/dist-packages (2.8.2+zzzcolab20220929150707)
Requirement already satisfied: tensorboard<2.9,>=2.8 in
/usr/local/lib/python3.7/dist-packages (from tensorflow) (2.8.0)
Requirement already satisfied: keras<2.9,>=2.8.0rc0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow) (2.8.0)
Requirement already satisfied: libclang>=9.0.1 in
/usr/local/lib/python3.7/dist-packages (from tensorflow) (14.0.6)
Requirement already satisfied: typing-extensions>=3.6.6 in
/usr/local/lib/python3.7/dist-packages (from tensorflow) (4.1.1)
Requirement already satisfied: astunparse>=1.6.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: wrapt>=1.11.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow) (1.14.1)
Requirement already satisfied: keras-preprocessing>=1.1.1 in
/usr/local/lib/python3.7/dist-packages (from tensorflow) (1.1.2)
Requirement already satisfied: google-pasta>=0.1.1 in
/usr/local/lib/python3.7/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/usr/local/lib/python3.7/dist-packages (from tensorflow) (1.49.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
/usr/local/lib/python3.7/dist-packages (from tensorflow) (0.27.0)
Requirement already satisfied: setuptools in
/usr/local/lib/python3.7/dist-packages (from tensorflow) (57.4.0)
Requirement already satisfied: opt-einsum>=2.3.2 in
/usr/local/lib/python3.7/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: h5py>=2.9.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow) (3.1.0)
Requirement already satisfied: gast>=0.2.1 in
/usr/local/lib/python3.7/dist-packages (from tensorflow) (0.5.3)
Requirement already satisfied: six>=1.12.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow) (1.15.0)
Requirement already satisfied: tensorflow-estimator<2.9,>=2.8 in
/usr/local/lib/python3.7/dist-packages (from tensorflow) (2.8.0)
Requirement already satisfied: flatbuffers>=1.12 in
/usr/local/lib/python3.7/dist-packages (from tensorflow) (22.9.24)
Requirement already satisfied: numpy>=1.20 in
/usr/local/lib/python3.7/dist-packages (from tensorflow) (1.21.6)
Requirement already satisfied: termcolor>=1.1.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow) (2.0.1)
Requirement already satisfied: protobuf<3.20,>=3.9.2 in
/usr/local/lib/python3.7/dist-packages (from tensorflow) (3.17.3)
Requirement already satisfied: absl-py>=0.4.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow) (1.2.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/usr/local/lib/python3.7/dist-packages (from astunparse>=1.6.0-
>tensorflow) (0.37.1)
Requirement already satisfied: cached-property in
/usr/local/lib/python3.7/dist-packages (from h5py>=2.9.0->tensorflow)
(1.5.2)
Requirement already satisfied: google-auth<3,>=1.6.3 in
/usr/local/lib/python3.7/dist-packages (from tensorboard<2.9,>=2.8-
>tensorflow) (1.35.0)
Requirement already satisfied: requests<3,>=2.21.0 in
/usr/local/lib/python3.7/dist-packages (from tensorboard<2.9,>=2.8-
>tensorflow) (2.23.0)
Requirement already satisfied: markdown>=2.6.8 in
/usr/local/lib/python3.7/dist-packages (from tensorboard<2.9,>=2.8-
>tensorflow) (3.4.1)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in
/usr/local/lib/python3.7/dist-packages (from tensorboard<2.9,>=2.8-
>tensorflow) (1.8.1)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in
/usr/local/lib/python3.7/dist-packages (from tensorboard<2.9,>=2.8-
>tensorflow) (0.4.6)
Requirement already satisfied: werkzeug>=0.11.15 in
/usr/local/lib/python3.7/dist-packages (from tensorboard<2.9,>=2.8-
>tensorflow) (1.0.1)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0
in /usr/local/lib/python3.7/dist-packages (from tensorboard<2.9,>=2.8-
>tensorflow) (0.6.1)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/usr/local/lib/python3.7/dist-packages (from google-auth<3,>=1.6.3-
>tensorboard<2.9,>=2.8->tensorflow) (0.2.8)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from google-auth<3,>=1.6.3-
>tensorboard<2.9,>=2.8->tensorflow) (4.2.4)
Requirement already satisfied: rsa<5,>=3.1.4 in
/usr/local/lib/python3.7/dist-packages (from google-auth<3,>=1.6.3-
>tensorboard<2.9,>=2.8->tensorflow) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
/usr/local/lib/python3.7/dist-packages (from google-auth-
oauthlib<0.5,>=0.4.1->tensorboard<2.9,>=2.8->tensorflow) (1.3.1)
Requirement already satisfied: importlib-metadata>=4.4 in
/usr/local/lib/python3.7/dist-packages (from markdown>=2.6.8-
>tensorboard<2.9,>=2.8->tensorflow) (5.0.0)
Requirement already satisfied: zipp>=0.5 in
/usr/local/lib/python3.7/dist-packages (from importlib-metadata>=4.4-
>markdown>=2.6.8->tensorboard<2.9,>=2.8->tensorflow) (3.8.1)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in
/usr/local/lib/python3.7/dist-packages (from pyasn1-modules>=0.2.1-
>google-auth<3,>=1.6.3->tensorboard<2.9,>=2.8->tensorflow) (0.4.8)
Requirement already satisfied: idna<3,>=2.5 in
/usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.9,>=2.8->tensorflow) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in
```

```
/usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.9,>=2.8->tensorflow) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1
in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.9,>=2.8->tensorflow) (1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.9,>=2.8->tensorflow) (2022.9.24)
Requirement already satisfied: oauthlib>=3.0.0 in
/usr/local/lib/python3.7/dist-packages (from requests-oauthlib>=0.7.0-
>google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.9,>=2.8->tensorflow)
(3.2.1)
```

```
import numpy as np
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
train_path = '/content/drive/MyDrive/flowers/train'
valid_path = '/content/drive/MyDrive/flowers/train'
```

```
train_datagen = ImageDataGenerator(rescale =
1./255,shear_range=0.2,zoom_range=0/2,horizontal_flip=True)
```

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

```
training_set = train_datagen.flow_from_directory(train_path,
                                                 target_size = (200,
200),
                                                 batch_size = 100,
                                                 class_mode =
'categorical')
```

```
Found 4317 images belonging to 5 classes.
```

```
import cv2 as cv
im =
cv.imread('/content/drive/MyDrive/flowers/test/daisy/100080576_f52e8ee
070_n.jpg')
```

```
im.shape
```

```
(263, 320, 3)
```

```
training_set.class_indices
```

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

```
test_set = test_datagen.flow_from_directory(valid_path,
                                            target_size = (200, 200),
                                            batch_size = 100,
                                            class_mode =
'categorical')
```

```
Found 4317 images belonging to 5 classes.
```

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

```
Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).
```

```
test_set.class_indices
```

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

```
model = Sequential()
model.add(Conv2D(16,(3,3),input_shape=(200,200,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(32,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dense(300, activation='relu'))
model.add(Dense(150, activation='relu'))
model.add(Dense(5, activation='softmax'))
```

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

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 198, 198, 16)      448       
                                                                 
 max_pooling2d (MaxPooling2D  (None, 99, 99, 16)       0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (None, 97, 97, 32)        4640      
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 48, 48, 32)       0         
 2D)                                                             
                                                                 
 flatten (Flatten)           (None, 73728)             0         
 dense (Dense)               (None, 300)               22118700

 dense_1 (Dense)             (None, 150)               45150

 dense_2 (Dense)             (None, 5)                 755

=================================================================
Total params: 22,169,693
Trainable params: 22,169,693
Non-trainable params: 0
_________________________________________________________________
```

```
r = model.fit_generator(
  training_set,
  validation_data=test_set,
  epochs=10
)
```

```
/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.
```

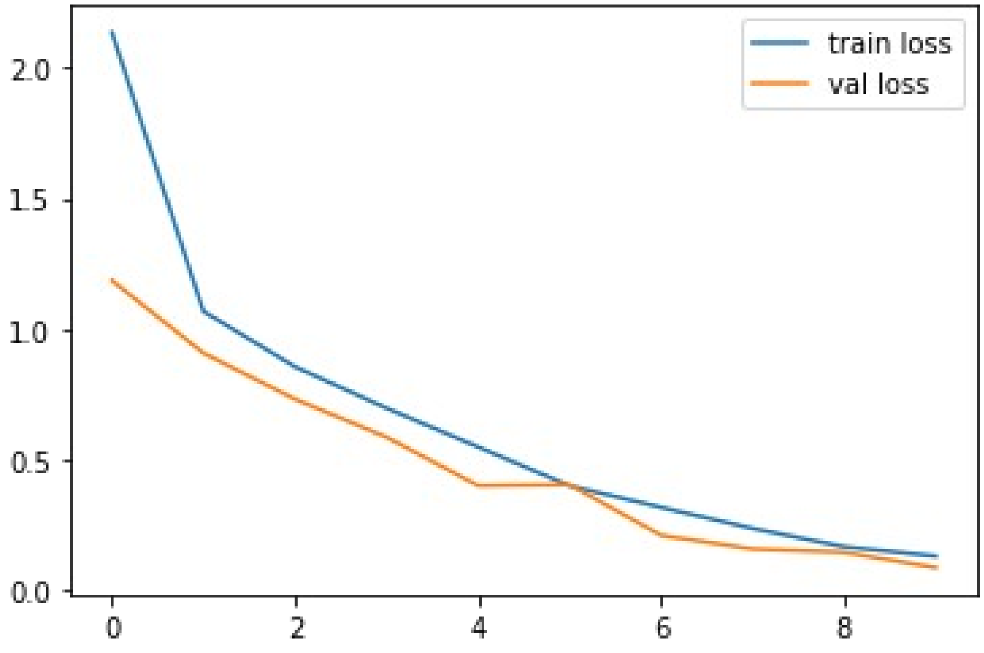
```
Epoch 1/10
44/44 [==============================] - 791s 18s/step - loss: 2.1362
- accuracy: 0.2993 - val_loss: 1.1855 - val_accuracy: 0.5022
Epoch 2/10
44/44 [==============================] - 196s 4s/step - loss: 1.0670 -
accuracy: 0.5800 - val_loss: 0.9072 - val_accuracy: 0.6514
Epoch 3/10
44/44 [==============================] - 196s 4s/step - loss: 0.8548 -
accuracy: 0.6713 - val_loss: 0.7308 - val_accuracy: 0.7470
Epoch 4/10
44/44 [==============================] - 197s 4s/step - loss: 0.6961 -
accuracy: 0.7422 - val_loss: 0.5850 - val_accuracy: 0.7806
Epoch 5/10
44/44 [==============================] - 196s 4s/step - loss: 0.5475 -
accuracy: 0.8045 - val_loss: 0.3990 - val_accuracy: 0.8705
Epoch 6/10
44/44 [==============================] - 200s 5s/step - loss: 0.3984 -
accuracy: 0.8677 - val_loss: 0.4027 - val_accuracy: 0.8543
Epoch 7/10
44/44 [==============================] - 198s 4s/step - loss: 0.3158 -
accuracy: 0.8962 - val_loss: 0.2083 - val_accuracy: 0.9402
Epoch 8/10
44/44 [==============================] - 196s 4s/step - loss: 0.2348 -
accuracy: 0.9247 - val_loss: 0.1564 - val_accuracy: 0.9553
Epoch 9/10
44/44 [==============================] - 201s 5s/step - loss: 0.1648 -
accuracy: 0.9493 - val_loss: 0.1438 - val_accuracy: 0.9560
```

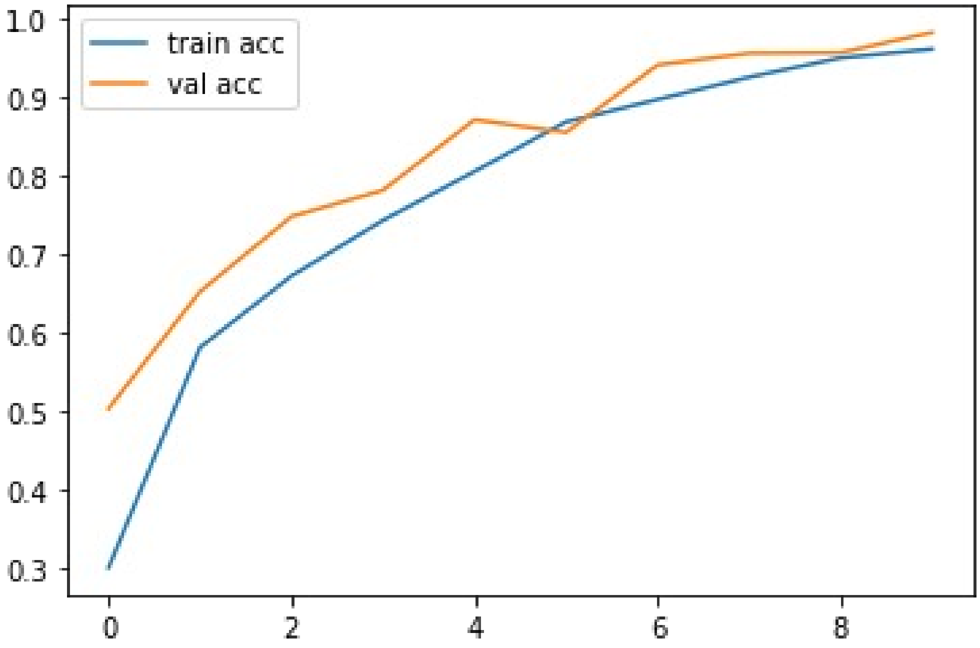
```
Epoch 10/10
44/44 [==============================] - 198s 4s/step - loss: 0.1292 -
accuracy: 0.9606 - val_loss: 0.0846 - val_accuracy: 0.9817
```

```
model.evaluate(test_set)
```

```
44/44 [==============================] - 45s 1s/step - loss: 0.0846 -
accuracy: 0.9817

[0.084574393928051, 0.9817002415657043]
```

```
import matplotlib.pyplot as plt

# plot the loss
plt.plot(r.history['loss'], label='train loss')
plt.plot(r.history['val_loss'], label='val loss')
plt.legend()
plt.show()
plt.savefig('LossVal_loss')

# plot the accuracy
plt.plot(r.history['accuracy'], label='train acc')
plt.plot(r.history['val_accuracy'], label='val acc')
plt.legend()
plt.show()
plt.savefig('AccVal_acc')
```





```
<Figure size 432x288 with 0 Axes>
```

```
model.save('flower.h5')
```

```
model_json = model.to_json()
with open("model-bw.json","w") as json_file:
  json_file.write(model_json)
```

```
from tensorflow.keras.models import load_model
from keras.preprocessing import image
model1 = load_model('/content/flower.h5')
```

```
img = image.load_img('/content/drive/MyDrive/flowers/test/daisy/2513618768_ff7c004796_m.jpg',target_size=(200,200)) # Reading image
x = image.img_to_array(img) # Converting image into array
x = np.expand_dims(x,axis=0) # expanding Dimensions
pred = np.argmax(model.predict(x)) # Predicting the higher probablity index
op = ['daisy', 'dandelion', 'rose', 'sunflower', 'tulip'] # Creating list
op[pred] # List indexing with output
```

```
{"type":"string"}
```

```
img = image.load_img('/content/drive/MyDrive/flowers/test/dandelion/2469856983_fe8e36ba57.jpg',target_size=(200,200)) # Reading image
x = image.img_to_array(img) # Converting image into array
x = np.expand_dims(x,axis=0) # expanding Dimensions
pred = np.argmax(model.predict(x)) # Predicting the higher probablity
```

```
index
op = ['daisy', 'dandelion', 'rose', 'sunflower', 'tulip'] # Creating
list
op[pred] # List indexing with output
```

```
{"type":"string"}
```

```
img =
image.load_img('/content/drive/MyDrive/flowers/test/dandelion/24779863
96_19da36d557_m.jpg',target_size=(200,200)) # Reading image
x = image.img_to_array(img) # Converting image into array
x = np.expand_dims(x,axis=0) # expanding Dimensions
pred = np.argmax(model.predict(x)) # Predicting the higher probablity
index
op = ['daisy', 'dandelion', 'rose', 'sunflower', 'tulip'] # Creating
list
op[pred] # List indexing with output
```

```
{"type":"string"}
```

```
img =
image.load_img('/content/drive/MyDrive/flowers/test/rose/2682566502_96
7e7eaa2a.jpg',target_size=(200,200)) # Reading image
x = image.img_to_array(img) # Converting image into array
x = np.expand_dims(x,axis=0) # expanding Dimensions
pred = np.argmax(model.predict(x)) # Predicting the higher probablity
index
op = ['daisy', 'dandelion', 'rose', 'sunflower', 'tulip'] # Creating
list
op[pred] # List indexing with output
```

```
{"type":"string"}
```

```
img =
image.load_img('/content/drive/MyDrive/flowers/test/sunflower/27292065
69_9dd2b5a3ed.jpg',target_size=(200,200)) # Reading image
x = image.img_to_array(img) # Converting image into array
x = np.expand_dims(x,axis=0) # expanding Dimensions
pred = np.argmax(model.predict(x)) # Predicting the higher probablity
index
op = ['daisy', 'dandelion', 'rose', 'sunflower', 'tulip'] # Creating
list
op[pred] # List indexing with output
```

```
{"type":"string"}
```

```
img =
image.load_img('/content/drive/MyDrive/flowers/test/tulip/112428665_d8
f3632f36_n.jpg',target_size=(200,200)) # Reading image
x = image.img_to_array(img) # Converting image into array
x = np.expand_dims(x,axis=0) # expanding Dimensions
pred = np.argmax(model.predict(x)) # Predicting the higher probablity
```

```
index
op = ['daisy', 'dandelion', 'rose', 'sunflower', 'tulip'] # Creating
list
op[pred] # List indexing with output
```

{"type":"string"}

```
from tensorflow.keras.callbacks import EarlyStopping,
ReduceLROnPlateau

early_stop = EarlyStopping(monitor='val_accuracy',
                           patience=5)

lr = ReduceLROnPlateau(monitor='val_accuaracy',
                       factor=0.5,
                       min_lr=0.00001)

callback = [early_stop,lr]# Train model

model.fit_generator(training_set,
                    steps_per_epoch=len(training_set),
                    epochs=100,
                    callbacks=callback,
                    validation_data=test_set,
                    validation_steps=len(test_set))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6:
UserWarning: `Model.fit_generator` is deprecated and will be removed
in a future version. Please use `Model.fit`, which supports
generators.


Epoch 1/100
44/44 [==============================] - ETA: 0s - loss: 0.0983 -
accuracy: 0.9764

WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuaracy` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr

 44/44 [==============================] - 201s 5s/step - loss: 0.0983
- accuracy: 0.9764 - val_loss: 0.0587 - val_accuracy: 0.9882 - lr:
0.0010
Epoch 2/100
44/44 [==============================] - ETA: 0s - loss: 0.0512 -
accuracy: 0.9870

WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuaracy` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr
```

```
 44/44 [==============================] - 198s 4s/step - loss: 0.0512
- accuracy: 0.9870 - val_loss: 0.0503 - val_accuracy: 0.9856 - lr:
0.0010
Epoch 3/100
44/44 [==============================] - ETA: 0s - loss: 0.0535 -
accuracy: 0.9873

WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuaracy` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr

 44/44 [==============================] - 197s 4s/step - loss: 0.0535
- accuracy: 0.9873 - val_loss: 0.0428 - val_accuracy: 0.9889 - lr:
0.0010
Epoch 4/100
44/44 [==============================] - ETA: 0s - loss: 0.0357 -
accuracy: 0.9924

WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuaracy` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr

 44/44 [==============================] - 201s 5s/step - loss: 0.0357
- accuracy: 0.9924 - val_loss: 0.0212 - val_accuracy: 0.9956 - lr:
0.0010
Epoch 5/100
44/44 [==============================] - ETA: 0s - loss: 0.0257 -
accuracy: 0.9951

WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuaracy` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr

 44/44 [==============================] - 197s 4s/step - loss: 0.0257
- accuracy: 0.9951 - val_loss: 0.0224 - val_accuracy: 0.9942 - lr:
0.0010
Epoch 6/100
44/44 [==============================] - ETA: 0s - loss: 0.0240 -
accuracy: 0.9942

WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuaracy` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr

 44/44 [==============================] - 197s 4s/step - loss: 0.0240
- accuracy: 0.9942 - val_loss: 0.0246 - val_accuracy: 0.9940 - lr:
0.0010
Epoch 7/100
44/44 [==============================] - ETA: 0s - loss: 0.0218 -
accuracy: 0.9940
```

```
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuaracy` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr

 44/44 [==============================] - 196s 4s/step - loss: 0.0218
- accuracy: 0.9940 - val_loss: 0.0148 - val_accuracy: 0.9981 - lr:
0.0010
Epoch 8/100
44/44 [==============================] - ETA: 0s - loss: 0.0182 -
accuracy: 0.9977

WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuaracy` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr

 44/44 [==============================] - 195s 4s/step - loss: 0.0182
- accuracy: 0.9977 - val_loss: 0.0097 - val_accuracy: 0.9991 - lr:
0.0010
Epoch 9/100
44/44 [==============================] - ETA: 0s - loss: 0.0100 -
accuracy: 0.9984

WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuaracy` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr

 44/44 [==============================] - 199s 5s/step - loss: 0.0100
- accuracy: 0.9984 - val_loss: 0.0072 - val_accuracy: 0.9991 - lr:
0.0010
Epoch 10/100
44/44 [==============================] - ETA: 0s - loss: 0.0106 -
accuracy: 0.9979

WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuaracy` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr

 44/44 [==============================] - 196s 4s/step - loss: 0.0106
- accuracy: 0.9979 - val_loss: 0.0052 - val_accuracy: 0.9988 - lr:
0.0010
Epoch 11/100
44/44 [==============================] - ETA: 0s - loss: 0.0101 -
accuracy: 0.9984

WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuaracy` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr

 44/44 [==============================] - 195s 4s/step - loss: 0.0101
- accuracy: 0.9984 - val_loss: 0.0058 - val_accuracy: 0.9991 - lr:
0.0010
Epoch 12/100
```

```
44/44 [==============================] - ETA: 0s - loss: 0.0071 - 
accuracy: 0.9986

WARNING:tensorflow:Learning rate reduction is conditioned on metric 
`val_accuaracy` which is not available. Available metrics are: 
loss,accuracy,val_loss,val_accuracy,lr

 44/44 [==============================] - 195s 4s/step - loss: 0.0071 
- accuracy: 0.9986 - val_loss: 0.0099 - val_accuracy: 0.9975 - lr: 
0.0010
Epoch 13/100
44/44 [==============================] - ETA: 0s - loss: 0.0076 - 
accuracy: 0.9986

WARNING:tensorflow:Learning rate reduction is conditioned on metric 
`val_accuaracy` which is not available. Available metrics are: 
loss,accuracy,val_loss,val_accuracy,lr

 44/44 [==============================] - 199s 5s/step - loss: 0.0076 
- accuracy: 0.9986 - val_loss: 0.0055 - val_accuracy: 0.9991 - lr: 
0.0010

<keras.callbacks.History at 0x7f69aac574d0>
```

```
img = 
image.load_img('/content/drive/MyDrive/flowers/test/daisy/174131220_c8
53df1287.jpg',target_size=(200,200)) # Reading image
x = image.img_to_array(img) # Converting image into array
x = np.expand_dims(x,axis=0) # expanding Dimensions
pred = np.argmax(model.predict(x)) # Predicting the higher probablity 
index
op = ['daisy', 'dandelion', 'rose', 'sunflower', 'tulip'] # Creating 
list
op[pred] # List indexing with output
```

{"type":"string"}

```
img = 
image.load_img('/content/drive/MyDrive/flowers/test/dandelion/24698569
83_fe8e36ba57.jpg',target_size=(200,200)) # Reading image
x = image.img_to_array(img) # Converting image into array
x = np.expand_dims(x,axis=0) # expanding Dimensions
pred = np.argmax(model.predict(x)) # Predicting the higher probablity 
index
op = ['daisy', 'dandelion', 'rose', 'sunflower', 'tulip'] # Creating 
list
op[pred] # List indexing with output
```

{"type":"string"}

```
img = 
image.load_img('/content/drive/MyDrive/flowers/test/dandelion/24779863
96_19da36d557_m.jpg',target_size=(200,200)) # Reading image
x = image.img_to_array(img) # Converting image into array
x = np.expand_dims(x,axis=0) # expanding Dimensions
pred = np.argmax(model.predict(x)) # Predicting the higher probablity
index
op = ['daisy', 'dandelion', 'rose', 'sunflower', 'tulip'] # Creating
list
op[pred] # List indexing with output
```

{"type":"string"}

```
img =
image.load_img('/content/drive/MyDrive/flowers/test/rose/2682566502_96
7e7eaa2a.jpg',target_size=(200,200)) # Reading image
x = image.img_to_array(img) # Converting image into array
x = np.expand_dims(x,axis=0) # expanding Dimensions
pred = np.argmax(model.predict(x)) # Predicting the higher probablity
index
op = ['daisy', 'dandelion', 'rose', 'sunflower', 'tulip'] # Creating
list
op[pred] # List indexing with output
```

{"type":"string"}

```
img =
image.load_img('/content/drive/MyDrive/flowers/test/sunflower/27292065
69_9dd2b5a3ed.jpg',target_size=(200,200)) # Reading image
x = image.img_to_array(img) # Converting image into array
x = np.expand_dims(x,axis=0) # expanding Dimensions
pred = np.argmax(model.predict(x)) # Predicting the higher probablity
index
op = ['daisy', 'dandelion', 'rose', 'sunflower', 'tulip'] # Creating
list
op[pred] # List indexing with output
```

{"type":"string"}

```
img =
image.load_img('/content/drive/MyDrive/flowers/test/tulip/112428665_d8
f3632f36_n.jpg',target_size=(200,200)) # Reading image
x = image.img_to_array(img) # Converting image into array
x = np.expand_dims(x,axis=0) # expanding Dimensions
pred = np.argmax(model.predict(x)) # Predicting the higher probablity
index
op = ['daisy', 'dandelion', 'rose', 'sunflower', 'tulip'] # Creating
list
op[pred] # List indexing with output
```

{"type":"string"}

```
img = image.load_img('/content/drive/MyDrive/herbert-goetsch-
SGKQh9wNgAk-unsplash.jpg',target_size=(200,200)) # Reading image
```

```
x = image.img_to_array(img) # Converting image into array
x = np.expand_dims(x,axis=0) # expanding Dimensions
pred = np.argmax(model.predict(x)) # Predicting the higher probablity index
op = ['daisy', 'dandelion', 'rose', 'sunflower', 'tulip'] # Creating list
op[pred] # List indexing with output
```

```
{"type":"string"}
```