

```

#CNN

##Download the Dataset

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

Mounted at /content/drive

cd/content/drive/MyDrive

/content/drive/MyDrive

!unzip Flowers-Dataset.zip

##Image Augmentation

    from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_datagen=ImageDataGenerator(rescale=1./255,zoom_range=0.2,horizon
tal_flip=True,vertical_flip=False)

test_datagen=ImageDataGenerator(rescale=1./255)

pip install split-folders

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 splitfolders

input_folder='/content/drive/MyDrive/flowers'

splitfolders.ratio(input_folder,output='/content/drive/MyDrive/
flowersdataset',ratio=(.8,0,.2),group_prefix=None)

Copying files: 0 files [00:00, ? files/s]

x_train=train_datagen.flow_from_directory(r"/content/drive/MyDrive/
flowersdataset/
train",target_size=(64,64),class_mode='categorical',batch_size=24)

Found 3452 images belonging to 5 classes.

x_test=test_datagen.flow_from_directory(r"/content/drive/MyDrive/
flowersdataset/
test",target_size=(64,64),class_mode='categorical',batch_size=24)

Found 865 images belonging to 5 classes.

```

```

x_train.class_indices
{'daisy': 0, 'dandelion': 1, 'rose': 2, 'sunflower': 3, 'tulip': 4}

##Create Model

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

model=Sequential()

##Add Layers(Convolution,MaxPooling,Flatten,

##Dense,Hidden,Output Layers)

#Adding Convolutional Layer
model.add(Convolution2D(32,
(3,3),input_shape=(64,64,3),activation='relu'))

#Adding Pooling Layer
model.add(MaxPooling2D(pool_size=(2,2)))

#Flatten Layer
model.add(Flatten())

model.summary()

Model: "sequential"

```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d (MaxPooling2D)	(None, 31, 31, 32)	0
flatten (Flatten)	(None, 30752)	0
Total params: 896		
Trainable params: 896		
Non-trainable params: 0		

```

#Hidden Layers
model.add(Dense(300,activation='relu'))
model.add(Dense(150,activation='relu'))

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

##Compile the model

```

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

```
len(x_train)
```

```
144
```

```
##6)Fit the Model
```

```
model.fit_generator(x_train,steps_per_epoch=len(x_train),validation_data=x_test,validation_steps=len(x_test),epochs=20)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1:
UserWarning: `Model.fit_generator` is deprecated and will be removed
in a future version. Please use `Model.fit`, which supports
generators.
```

```
"""Entry point for launching an IPython kernel.
```

```
Epoch 1/20
```

```
144/144 [=====] - 747s 5s/step - loss: 0.9629
- accuracy: 0.6234 - val_loss: 1.0152 - val_accuracy: 0.6301
```

```
Epoch 2/20
```

```
144/144 [=====] - 19s 129ms/step - loss:
0.8936 - accuracy: 0.6535 - val_loss: 1.0343 - val_accuracy: 0.6000
```

```
Epoch 3/20
```

```
144/144 [=====] - 18s 128ms/step - loss:
0.8338 - accuracy: 0.6840 - val_loss: 1.1712 - val_accuracy: 0.5769
```

```
Epoch 4/20
```

```
144/144 [=====] - 18s 126ms/step - loss:
0.7766 - accuracy: 0.7008 - val_loss: 0.9804 - val_accuracy: 0.6393
```

```
Epoch 5/20
```

```
144/144 [=====] - 19s 129ms/step - loss:
0.7430 - accuracy: 0.7164 - val_loss: 0.9958 - val_accuracy: 0.6405
```

```
Epoch 6/20
```

```
144/144 [=====] - 19s 130ms/step - loss:
0.7026 - accuracy: 0.7291 - val_loss: 1.0984 - val_accuracy: 0.6185
```

```
Epoch 7/20
```

```
144/144 [=====] - 18s 127ms/step - loss:
0.6567 - accuracy: 0.7506 - val_loss: 1.0654 - val_accuracy: 0.6185
```

```
Epoch 8/20
```

```
144/144 [=====] - 18s 126ms/step - loss:
0.6157 - accuracy: 0.7726 - val_loss: 1.0470 - val_accuracy: 0.6786
```

```
Epoch 9/20
```

```
144/144 [=====] - 18s 126ms/step - loss:
0.5689 - accuracy: 0.7836 - val_loss: 1.0108 - val_accuracy: 0.6590
```

```
Epoch 10/20
```

```
144/144 [=====] - 18s 126ms/step - loss:
0.5628 - accuracy: 0.7955 - val_loss: 1.1033 - val_accuracy: 0.6509
```

```
Epoch 11/20
```

```
144/144 [=====] - 18s 126ms/step - loss:
0.5016 - accuracy: 0.8134 - val_loss: 1.0186 - val_accuracy: 0.6728
```

```

Epoch 12/20
144/144 [=====] - 18s 127ms/step - loss:
0.4555 - accuracy: 0.8297 - val_loss: 1.1393 - val_accuracy: 0.6751
Epoch 13/20
144/144 [=====] - 18s 127ms/step - loss:
0.4503 - accuracy: 0.8410 - val_loss: 1.0704 - val_accuracy: 0.6671
Epoch 14/20
144/144 [=====] - 18s 127ms/step - loss:
0.4131 - accuracy: 0.8470 - val_loss: 1.1088 - val_accuracy: 0.6613
Epoch 15/20
144/144 [=====] - 18s 127ms/step - loss:
0.3748 - accuracy: 0.8589 - val_loss: 1.1775 - val_accuracy: 0.6636
Epoch 16/20
144/144 [=====] - 18s 126ms/step - loss:
0.3633 - accuracy: 0.8618 - val_loss: 1.2502 - val_accuracy: 0.6474
Epoch 17/20
144/144 [=====] - 18s 125ms/step - loss:
0.3283 - accuracy: 0.8815 - val_loss: 1.2621 - val_accuracy: 0.6671
Epoch 18/20
144/144 [=====] - 18s 125ms/step - loss:
0.3180 - accuracy: 0.8844 - val_loss: 1.4578 - val_accuracy: 0.6590
Epoch 19/20
144/144 [=====] - 18s 125ms/step - loss:
0.3011 - accuracy: 0.8972 - val_loss: 1.3159 - val_accuracy: 0.6578
Epoch 20/20
144/144 [=====] - 18s 124ms/step - loss:
0.2743 - accuracy: 0.9053 - val_loss: 1.3801 - val_accuracy: 0.6728

```

<keras.callbacks.History at 0x7f603e71cc50>

##Test the model

```

import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image

img=image.load_img(r"/content/drive/MyDrive/flowersdataset/test/
daisy/3706420943_66f3214862_n.jpg",target_size=(64,64))
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
y=np.argmax(model.predict(x),axis=1)
x_train.class_indices
index=['daisy','dandelion','rose','sunflower','tulip']
index[y[0]]

{"type": "string"}

img=image.load_img(r"/content/tulip.jfif",target_size=(64,64))
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
y=np.argmax(model.predict(x),axis=1)

```

```
x_train.class_indices
index=['daisy','dandellion','rose','sunflower','tulip']
index[y[0]]
{"type":"string"}
```