Build CNN Model for Classification Of Flowers

#TEAM ID : PNT2022TMID04039

```
import os
import zipfile
import random
import shutil
import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator
from os import getcwd
from os import listdir
import cv2
from keras.layers import Conv2D, Input, ZeroPadding2D,
BatchNormalization, Activation, MaxPooling2D, Flatten, Dense
from keras.models import Model, load_model
from keras.callbacks import TensorBoard, ModelCheckpoint
from sklearn.model_selection import train_test_split
import imutils
import numpy as np
import matplotlib.pyplot as plt
```

1. Download the dataset

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

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

```
from zipfile import ZipFile
file_name = "/content/drive/MyDrive/Flowers-Dataset (assignment
3).zip"

with ZipFile(file_name, 'r') as zip:
  zip.extractall()
```

Number of Images in each class

```
print(len(os.listdir('/content/flowers/daisy')))
print(len(os.listdir('/content/flowers/dandelion')))
print(len(os.listdir('/content/flowers/rose')))
print(len(os.listdir('/content/flowers/sunflower')))
print(len(os.listdir('/content/flowers/tulip')))
```

```
764
1052
784
733
984
```

2.Image Augumentation

```
TRAINING_DIR = "/content/accdetection/training"

train_datagen = ImageDataGenerator(rescale=1./255,
      rotation_range=40,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True,
      fill_mode='nearest')
train_generator = train_datagen.flow_from_directory(TRAINING_DIR,
                                                    batch_size=100,

class_mode='binary',
                                                    target_size=(150,
150))


VALIDATION_DIR = "/content/accdetection/testing"

validation_datagen = ImageDataGenerator(rescale=1./255)
validation_generator =
validation_datagen.flow_from_directory(VALIDATION_DIR,

batch_size=100,

class_mode='binary',

target_size=(150, 150))
```

```
Found 4292 images belonging to 5 classes.
Found 1243 images belonging to 5 classes.
```

3.Create model

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

4.Add layers

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(100, (3,3), activation='relu',
input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Conv2D(100, (3,3), activation='relu'),
```

```
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(50, activation='relu'),
    tf.keras.layers.Dense(50, activation='softmax')
])
print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 148, 148, 100)     2800

 max_pooling2d (MaxPooling2D  (None, 74, 74, 100)      0
 )

 conv2d_1 (Conv2D)           (None, 72, 72, 100)       90100

 max_pooling2d_1 (MaxPooling  (None, 36, 36, 100)      0
 2D)

 flatten (Flatten)           (None, 129600)            0

 dropout (Dropout)           (None, 129600)            0

 dense (Dense)               (None, 50)                6480050

 dense_1 (Dense)             (None, 50)                2550

=================================================================
Total params: 6,575,500
Trainable params: 6,575,500
Non-trainable params: 0
_________________________________________________________________
None
```

5.Compile the model

```
opt=tf.keras.optimizers.Adam(learning_rate=0.001)

model.compile(optimizer=opt,
              loss='sparse_categorical_crossentropy',
              metrics=['acc'])
```

6.Fit the model

```
history = model.fit(
            train_generator,
            validation_data=validation_generator,
            epochs=10,
            verbose=1)
```

```
Epoch 1/10
43/43 [==============================] - 439s 10s/step - loss: 1.5641
- acc: 0.3663 - val_loss: 1.1328 - val_acc: 0.5294
Epoch 2/10
43/43 [==============================] - 435s 10s/step - loss: 1.1847
- acc: 0.5156 - val_loss: 1.0994 - val_acc: 0.5414
Epoch 3/10
43/43 [==============================] - 431s 10s/step - loss: 1.1077
- acc: 0.5571 - val_loss: 0.9780 - val_acc: 0.6227
Epoch 4/10
43/43 [==============================] - 428s 10s/step - loss: 1.0248
- acc: 0.5939 - val_loss: 0.9686 - val_acc: 0.6275
Epoch 5/10
43/43 [==============================] - 436s 10s/step - loss: 0.9705
- acc: 0.6233 - val_loss: 0.9084 - val_acc: 0.6492
Epoch 6/10
43/43 [==============================] - 432s 10s/step - loss: 0.9435
- acc: 0.6323 - val_loss: 0.9555 - val_acc: 0.6372
Epoch 7/10
43/43 [==============================] - 425s 10s/step - loss: 0.9300
- acc: 0.6398 - val_loss: 0.9788 - val_acc: 0.6122
Epoch 8/10
43/43 [==============================] - 433s 10s/step - loss: 0.9085
- acc: 0.6470 - val_loss: 0.8271 - val_acc: 0.6766
Epoch 9/10
43/43 [==============================] - 433s 10s/step - loss: 0.8956
- acc: 0.6573 - val_loss: 0.8002 - val_acc: 0.6718
Epoch 10/10
43/43 [==============================] - 434s 10s/step - loss: 0.8556
- acc: 0.6759 - val_loss: 0.8035 - val_acc: 0.7056
```

7.Save the model

```
model.save('Flower Classification.h5')
```

8.Test the model

```
import numpy as np
from tensorflow.keras.preprocessing import image
img = image.load_img('/content/accdetection/testing/sunflower
img/18876985840_7531dc8e6a.jpg',target_size=(150,150))
s = image.img_to_array(img)
```

```
s = np.expand_dims(s,axis=0)
s
```

```
array([[[[ 28.,  55.,  10.],
         [ 20.,  33.,  13.],
         [ 18.,  26.,  13.],
         ...,
         [ 63., 106.,  35.],
         [ 62., 107.,  26.],
         [ 46.,  87.,  21.]],

        [[ 31.,  59.,  11.],
         [ 22.,  36.,  13.],
         [ 16.,  24.,   9.],
         ...,
         [ 61., 103.,  39.],
         [ 61., 105.,  26.],
         [ 49.,  90.,  22.]],

        [[ 31.,  61.,  11.],
         [ 22.,  38.,  12.],
         [ 16.,  27.,  10.],
         ...,
         [ 58., 100.,  36.],
         [ 58., 102.,  25.],
         [ 48.,  91.,  20.]],

        ...,

        [[ 58.,  87.,  33.],
         [ 50.,  88.,  31.],
         [ 42.,  81.,  28.],
         ...,
         [ 40.,  78.,  27.],
         [ 40.,  78.,  29.],
         [ 32.,  63.,  19.]],

        [[ 53.,  79.,  32.],
         [ 52.,  85.,  28.],
         [ 45.,  80.,  24.],
         ...,
         [ 41.,  79.,  28.],
         [ 38.,  76.,  27.],
         [ 31.,  64.,  17.]],

        [[ 51.,  77.,  29.],
         [ 50.,  83.,  26.],
         [ 48.,  83.,  27.],
         ...,
         [ 41.,  79.,  28.],
```

```
         [ 37.,  75.,  26.],
         [ 30.,  67.,  13.]]]], dtype=float32)
```

```
model.predict(s)
```

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

```
output = ['daisy','dandelion','rose','sunflower','tulip']
pred = np.argmax(model.predict(s))
output[pred]
```

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

```
img = image.load_img('/content/accdetection/testing/sunflower
img/18876985840_7531dc8e6a.jpg',target_size=(150,150))
s = image.img_to_array(img)
s = np.expand_dims(s,axis=0)
pred = np.argmax(model.predict(s))
output[pred]
```

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

```
img = image.load_img('/content/accdetection/testing/daisy
img/12601254324_3cb62c254a_m.jpg',target_size=(150,150))
s = image.img_to_array(img)
s = np.expand_dims(s,axis=0)
pred = np.argmax(model.predict(s))
output[pred]
```

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

```
img = image.load_img('/content/accdetection/testing/tulip
img/14046760909_0c73e84a1f_n.jpg',target_size=(150,150))
s = image.img_to_array(img)
s = np.expand_dims(s,axis=0)
pred = np.argmax(model.predict(s))
output[pred]
```

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

```
img = image.load_img('/content/accdetection/testing/rose
img/12395698413_c0388278f7.jpg',target_size=(150,150))
s = image.img_to_array(img)
s = np.expand_dims(s,axis=0)
pred = np.argmax(model.predict(s))
output[pred]
```

{"type":"string"}