```
# -*- coding: utf-8 -*-
"""Assignment_3.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1__XnMuSoGz0trTTXp7BW5Ezu0G0tVZiF

# **Building a CNN model for classification of Flowers**

# **Load the dataset**
"""

!unzip '/content/drive/MyDrive/Flowers-Dataset.zip'

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

#importing required libraries to build a CNN classification model with
accuracy

import numpy as np
import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
import matplotlib.pyplot as plt
batch_size = 32
img_height = 180
img_width = 180
data_dir = "/content/flowers"

"""# **Image Augmentation**"""

from tensorflow.keras.preprocessing.image import ImageDataGenerator

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

x_train = train_datagen.flow_from_directory(r"/content/flowers",
target_size = (64,64) , class_mode = "categorical", batch_size = 100)

#Image Augumentation accuracy
data_augmentation = Sequential(
  [
    layers.RandomFlip("horizontal",input_shape=(img_height, img_width,
3)),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1),
  ]
)

"""# **Model Building and also Split dataset into training and testing
sets**"""
```

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

train_ds = tf.keras.utils.image_dataset_from_directory(
  data_dir,
  validation_split=0.2,
  subset="training",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)

val_ds = tf.keras.utils.image_dataset_from_directory(
  data_dir,
  validation_split=0.2,
  subset="validation",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)

class_names = train_ds.class_names
print(class_names)

plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
  for i in range(9):
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(images[i].numpy().astype("uint8"))
    plt.title(class_names[labels[i]])
    plt.axis("off")

"""# **Adding the layers (Convolution,MaxPooling,Flatten,Dense-
(HiddenLayers),Output)**"""

model.add(Convolution2D(32, (3,3), activation = "relu", input_shape =
(64,64,3) ))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Flatten())
model.add(Dense(300, activation = "relu"))
model.add(Dense(150, activation = "relu")) #mulitple dense layers
model.add(Dense(5, activation = "softmax")) #output layer

#Adding the layers for accuracy
num_classes = len(class_names)

model = Sequential([
  data_augmentation,
  layers.Rescaling(1./255, input_shape=(img_height, img_width, 3)),
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])

"""# **Compile The Model**"""

model.compile(loss = "categorical_crossentropy", metrics = ["accuracy"],
optimizer = "adam")
len(x_train)

#Compile the model for further accuracy
model.compile(optimizer='adam',

loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
epochs=10
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)

#To find the Training and Validation- Accuracy & Loss (Visualization)

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()

"""#**Fit The Model** """

model.fit(x_train, epochs = 15, steps_per_epoch = len(x_train))

"""# **Save The Model**"""
```

```
model.save("flowers.h1")

model.save("flowers.m5")#another model to show the accuracy

"""# **Test The Model**"""

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

model = load_model("/content/flowers.h1")

#Testing with a random rose image from Google
img =
image.load_img('/content/drive/MyDrive/flowers/rose/11233672494_d8bf0a3db
f_n.jpg',target_size=(64,64))

img

x = image.img_to_array(img)
x.ndim

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

x = image.img_to_array(img)
x = np.expand_dims(x,axis=0)
x

labels = ['daisy','dandelion','roses','sunflowers','tulips']

#Testing the alternative model with accuracy

sunflower_url =
"https://storage.googleapis.com/download.tensorflow.org/example_images/59
2px-Red_sunflower.jpg"
sunflower_path = tf.keras.utils.get_file('Red_sunflower',
origin=sunflower_url)

img = tf.keras.utils.load_img(
    sunflower_path, target_size=(img_height, img_width)
)
img_array = tf.keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch

predictions = model.predict(img_array)
score = tf.nn.softmax(predictions[0])

print(
    "This image most likely belongs to {} with a {:.2f} percent
confidence."
    .format(class_names[np.argmax(score)], 100 * np.max(score))
)
```