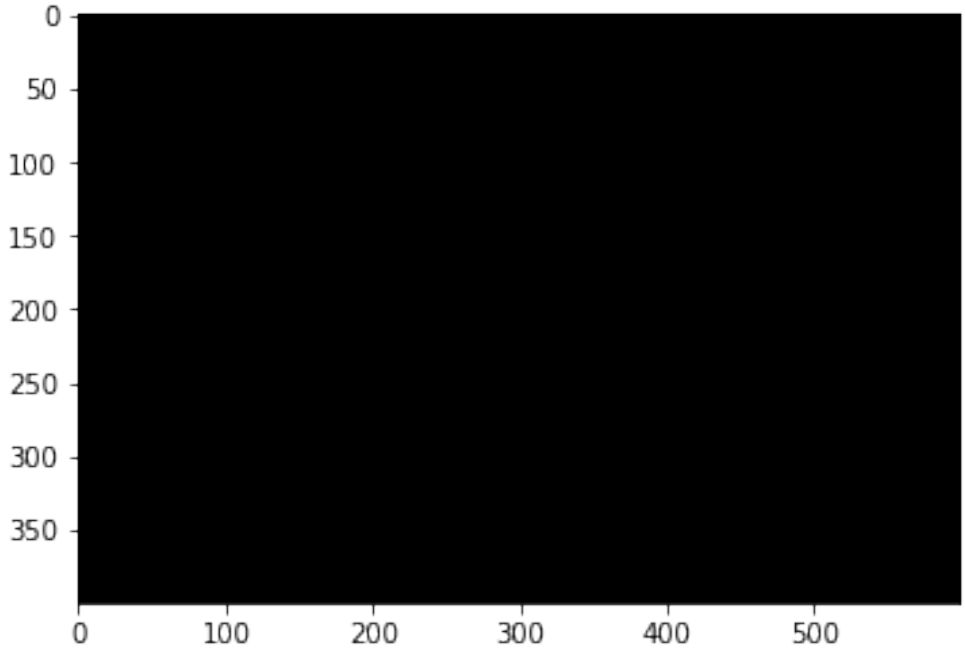
IMPORTING LIBRARIES

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

CREATE BLACK IMAGE

```
img1=np.zeros((400,600,3),np.uint8)
plt.imshow(img1)
```

```
<matplotlib.image.AxesImage at 0x7f51a2ce61d0>
```



kaggle

```
#Import  libraries
import matplotlib.pyplot as plt
import numpy as np
import os
import PIL
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential

import pathlib
data_url =
"https://storage.googleapis.com/download.tensorflow.org/example_images
/flower_photos.tgz"
data_dir = tf.keras.utils.get_file('flower_photos', origin=data_url,
```

```
untar=True)
data_dir = pathlib.Path(data_dir)
```

```
Downloading data from
https://storage.googleapis.com/download.tensorflow.org/example_images/
flower_photos.tgz
228818944/228813984 [==============================] - 3s 0us/step
228827136/228813984 [==============================] - 3s 0us/step
```

```
image_count = len(list(data_dir.glob('*/*.jpg')))
print(image_count)
```

```
3670
```

```
roses = list(data_dir.glob('roses/*'))
PIL.Image.open(str(roses[25]))
```

```
PIL.Image.open(str(roses[30]))
```

```
tulips = list(data_dir.glob('tulips/*'))
PIL.Image.open(str(tulips[0]))
```

```
PIL.Image.open(str(tulips[5]))
```

```
PIL.Image.open(str(tulips[2]))
```

Create a dataset

```
batch_size = 32
img_height = 180
img_width  = 180

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,
)
```

```
Found 3670 files belonging to 5 classes.
Using 2936 files for training.
```

```
test_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,
)
```

```
Found 3670 files belonging to 5 classes.
Using 734 files for validation.
```

Create a Model

```
batch_size = 32
img_height = 180
img_width  = 180
```

```
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,
)
```

```
Found 3670 files belonging to 5 classes.
Using 2936 files for training.
```

```
test_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,
)
```

```
Found 3670 files belonging to 5 classes.
Using 734 files for validation.
```

```
class_name = train_ds.class_names
print(class_name)
```

```
['daisy', 'dandelion', 'roses', 'sunflowers', 'tulips']
```

```
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
  for i in range(4):
    ax = plt.subplot(2, 2, i + 1)
    plt.imshow(images[i].numpy().astype("uint8"))
    plt.title(class_name[labels[i]])
    plt.axis("off")
```


```
for image_batch , label_batch in train_ds:
    print(image_batch.shape)
    print(label_batch.shape)
    break
```

```
(32, 180, 180, 3)
(32,)
```

```
AUTOTUNE = tf.data.AUTOTUNE

train_ds =
train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_ds = test_ds.cache().prefetch(buffer_size=AUTOTUNE)

normalization_layer = layers.Rescaling(1./255)

normalized_ds = train_ds.map(lambda x, y: (normalization_layer(x), y))
image_batch, labels_batch = next(iter(normalized_ds))
first_image = image_batch[0]
# Notice the pixel values are now in `[0,1]`.
print(np.min(first_image), np.max(first_image))
```

```
0.0 0.9995523
```

Create the model

```
num_classes = len(class_name)
model = Sequential([
    layers.Rescaling(1./255, input_shape=(img_height, img_width, 3)),
    layers.Conv2D(16, 3 ,padding='same' , activation='relu'),
    layers.MaxPool2D(),
    layers.Conv2D(32, 3 ,padding='same' , activation='relu'),
    layers.MaxPool2D(),
    layers.Conv2D(64, 3 ,padding='same' , activation='relu'),
    layers.MaxPool2D(),
    layers.Conv2D(128, 3 ,padding='same' , activation='relu'),
    layers.MaxPool2D(),
    layers.Flatten(),
    layers.Dense(256,activation='relu'),
    layers.Dense(num_classes),

])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_2 (Rescaling)     (None, 180, 180, 3)       0

 conv2d (Conv2D)             (None, 180, 180, 16)      448

 max_pooling2d (MaxPooling2D  (None, 90, 90, 16)       0
 )

 conv2d_1 (Conv2D)           (None, 90, 90, 32)        4640

 max_pooling2d_1 (MaxPooling  (None, 45, 45, 32)       0
 2D)
```

```
 conv2d_2 (Conv2D)           (None, 45, 45, 64)        18496

 max_pooling2d_2 (MaxPooling  (None, 22, 22, 64)       0
 2D)

 conv2d_3 (Conv2D)           (None, 22, 22, 128)       73856

 max_pooling2d_3 (MaxPooling  (None, 11, 11, 128)      0
 2D)

 flatten (Flatten)           (None, 15488)             0

 dense (Dense)               (None, 256)               3965184

 dense_1 (Dense)             (None, 5)                 1285

=================================================================
Total params: 4,063,909
Trainable params: 4,063,909
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(optimizer='adam',
loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
metrics=['accuracy'])

history = model.fit(train_ds,epochs=10, validation_data=test_ds)
```

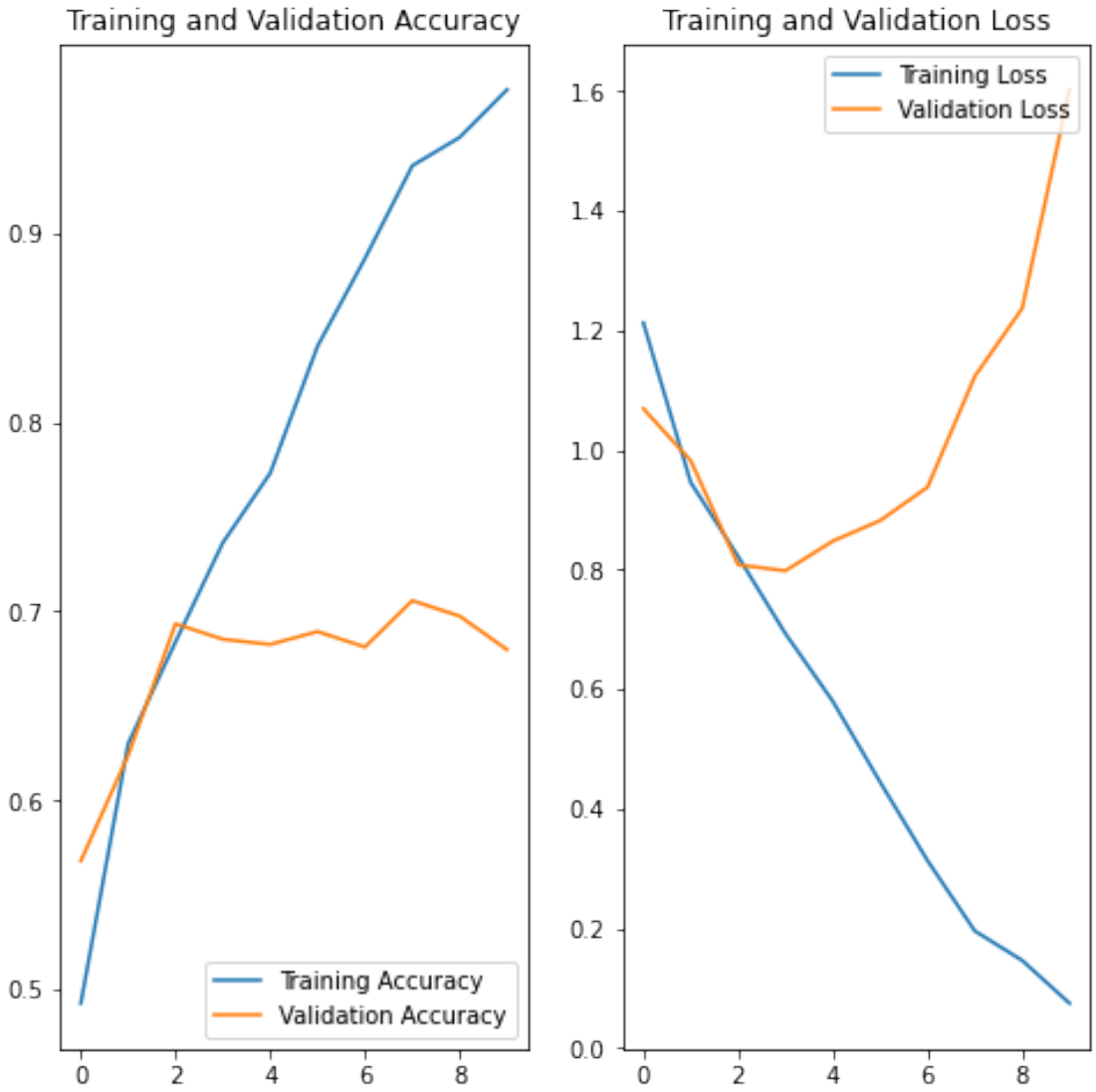
```
Epoch 1/10
92/92 [==============================] - 114s 1s/step - loss: 1.2126 -
accuracy: 0.4928 - val_loss: 1.0692 - val_accuracy: 0.5681
Epoch 2/10
92/92 [==============================] - 111s 1s/step - loss: 0.9457 -
accuracy: 0.6301 - val_loss: 0.9821 - val_accuracy: 0.6240
Epoch 3/10
92/92 [==============================] - 111s 1s/step - loss: 0.8221 -
accuracy: 0.6839 - val_loss: 0.8081 - val_accuracy: 0.6935
Epoch 4/10
92/92 [==============================] - 116s 1s/step - loss: 0.6931 -
accuracy: 0.7364 - val_loss: 0.7981 - val_accuracy: 0.6853
Epoch 5/10
92/92 [==============================] - 114s 1s/step - loss: 0.5801 -
accuracy: 0.7732 - val_loss: 0.8478 - val_accuracy: 0.6826
Epoch 6/10
92/92 [==============================] - 113s 1s/step - loss: 0.4448 -
accuracy: 0.8403 - val_loss: 0.8820 - val_accuracy: 0.6894
Epoch 7/10
92/92 [==============================] - 113s 1s/step - loss: 0.3136 -
accuracy: 0.8866 - val_loss: 0.9378 - val_accuracy: 0.6812
Epoch 8/10
```

```
92/92 [==============================] - 113s 1s/step - loss: 0.1960 -
accuracy: 0.9356 - val_loss: 1.1233 - val_accuracy: 0.7057
Epoch 9/10
92/92 [==============================] - 113s 1s/step - loss: 0.1467 -
accuracy: 0.9506 - val_loss: 1.2368 - val_accuracy: 0.6975
Epoch 10/10
92/92 [==============================] - 112s 1s/step - loss: 0.0750 -
accuracy: 0.9758 - val_loss: 1.6023 - val_accuracy: 0.6798
```

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs=10
epoch_range = range(epochs)

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

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




```
flower_url =
'https://storage.googleapis.com/download.tensorflow.org/example_images
/592px-Red_sunflower.jpg'
flower_path = tf.keras.utils.get_file('Red_flower',
origin=flower_url )
```

```
Downloading data from
https://storage.googleapis.com/download.tensorflow.org/example_images/
592px-Red_sunflower.jpg
122880/117948 [==============================] - 0s 0us/step
131072/117948 [==============================] - 0s 0us/step
```

```
PIL.Image.open(flower_path)
```