```
import matplotlib.pyplot as plt
import numpy as np
import PIL
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from keras.layers import Dense, Activation, Dropout, Flatten, Conv2D, MaxPooling2D,GlobalAveragePooling2D
from tensorflow.keras.layers import BatchNormalization
np.random.seed(1000)
import pathlib
```

```
dataset_url = "https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz"
data_dir = tf.keras.utils.get_file('flower_photos.tar', origin=dataset_url, extract=True)
print(data_dir)
data_dir = pathlib.Path(data_dir).with_suffix('')
print(data_dir)
```

```
Downloading data from https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz
228813984/228813984 [==============================] - 2s 0us/step
/root/.keras/datasets/flower_photos.tar
/root/.keras/datasets/flower_photos
```

VGG16

```
batch_size=12
img_height=224
img_width=224
```

```
train_ds=tf.keras.utils.image_dataset_from_directory(
    data_dir,validation_split=0.2,
    subset="training",
    seed=123, #the same set of images always go into the training, test, validation set.
    image_size=(img_height,img_width),
    batch_size=batch_size
)
```

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

```
val_ds=tf.keras.utils.image_dataset_from_directory(
    data_dir,validation_split=0.2,
    subset="validation",
    seed=123, #the same set of images always go into the training, test, validation set.
    image_size=(img_height,img_width),
    batch_size=batch_size
)
```

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

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

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

```
num_classes=len(class_names)
```

```
#VGG16
model=Sequential()
#layer 1
model.add(Conv2D(filters=64,input_shape=(224,224,3),strides=(1,1),kernel_size=(3,3)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(3,3),strides=(2,2)))

#layer2
model.add(Conv2D(filters=128,input_shape=(112,112,128),kernel_size=(3,3),strides=(1,1)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(3,3),strides=(2,2)))

#layer3
model.add(Conv2D(filters=256,input_shape=(56,56,256),kernel_size=(3,3),strides=(1,1)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(3,3),strides=(2,2)))
```

```
#layer4
model.add(Conv2D(filters=512,input_shape=(28,28,512),kernel_size=(3,3),strides=(1,1)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(3,3),strides=(2,2)))

#layer5
model.add(Conv2D(filters=512,input_shape=(14,14,512),kernel_size=(3,3),strides=(1,1)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(3,3),strides=(2,2)))

#fully connected
model.add(Flatten())

#1st FC
model.add(Dense(25088))
model.add(Activation('relu'))

#2nd FC
model.add(Dense(4096))
model.add(Activation('relu'))
#dropout
#model.add(Dropout(0.4))

#3rd FC
model.add(Dense(4096))
model.add(Activation('relu'))
#dropout
#model.add(Dropout(0.4))

#output layer
model.add(Dense(1000))
model.add(Activation('softmax'))


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

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 222, 222, 64)      1792      
                                                                 
 activation (Activation)     (None, 222, 222, 64)      0         
                                                                 
 max_pooling2d (MaxPooling2D  (None, 110, 110, 64)     0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (None, 108, 108, 128)     73856     
                                                                 
 activation_1 (Activation)   (None, 108, 108, 128)     0         
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 53, 53, 128)      0         
 2D)                                                             
                                                                 
 conv2d_2 (Conv2D)           (None, 51, 51, 256)       295168    
                                                                 
 activation_2 (Activation)   (None, 51, 51, 256)       0         
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 25, 25, 256)      0         
 2D)                                                             
                                                                 
 conv2d_3 (Conv2D)           (None, 23, 23, 512)       1180160   
                                                                 
 activation_3 (Activation)   (None, 23, 23, 512)       0         
                                                                 
 max_pooling2d_3 (MaxPooling  (None, 11, 11, 512)      0         
 2D)                                                             
                                                                 
 conv2d_4 (Conv2D)           (None, 9, 9, 512)         2359808   
                                                                 
 activation_4 (Activation)   (None, 9, 9, 512)         0         
                                                                 
 max_pooling2d_4 (MaxPooling  (None, 4, 4, 512)        0         
 2D)                                                             
                                                                 
 flatten (Flatten)           (None, 8192)              0         
                                                                 
 dense (Dense)               (None, 25088)             205545984 
                                                                 
 activation_5 (Activation)   (None, 25088)             0         
 dense_1 (Dense)             (None, 4096)              102764544

 activation_6 (Activation)   (None, 4096)              0

 dense_2 (Dense)             (None, 4096)              16781312

 activation_7 (Activation)   (None, 4096)              0

 dense_3 (Dense)             (None, 1000)              4097000

 activation_8 (Activation)   (None, 1000)              0

=================================================================
```

```
#model fit
epochs=4
history10 = model.fit(train_ds, validation_data=val_ds, epochs=epochs)
```

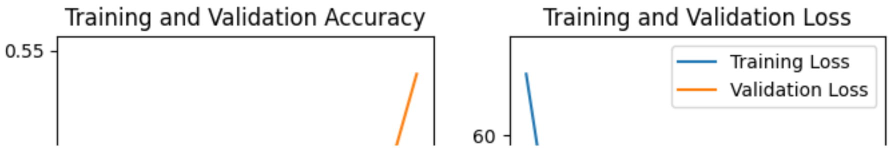
```
Epoch 1/4
/usr/local/lib/python3.10/dist-packages/keras/backend.py:5612: UserWarning: "`sparse_categorical_crossentropy` received `from_logit
  output, from_logits = _get_logits(
245/245 [==============================] - 46s 123ms/step - loss: 65.2881 - accuracy: 0.2027 - val_loss: 1.6309 - val_accuracy: 0.2
Epoch 2/4
245/245 [==============================] - 29s 118ms/step - loss: 1.5841 - accuracy: 0.2790 - val_loss: 1.5096 - val_accuracy: 0.30
Epoch 3/4
245/245 [==============================] - 29s 117ms/step - loss: 1.4455 - accuracy: 0.3546 - val_loss: 1.3658 - val_accuracy: 0.35
Epoch 4/4
245/245 [==============================] - 29s 118ms/step - loss: 1.2831 - accuracy: 0.4414 - val_loss: 1.1601 - val_accuracy: 0.53
```

accuracy: 67.30%

```
acc = history10.history['accuracy']
val_acc = history10.history['val_accuracy']
loss = history10.history['loss']
val_loss = history10.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()
```



31.5.23

alexnet

```
batch_size=128
img_height=227
img_width=227
```

```
train_ds1=tf.keras.utils.image_dataset_from_directory(
    data_dir,validation_split=0.2,
    subset="training",
    seed=123, #the same set of images always go into the training, test, validation set.
    image_size=(img_height,img_width),
    batch_size=batch_size
)
```

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

```
class_names = train_ds1.class_names
print(class_names)
```

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

```
val_ds1=tf.keras.utils.image_dataset_from_directory(
    data_dir,validation_split=0.2,
    subset="validation",
    seed=123, #the same set of images always go into the training, test, validation set.
    image_size=(img_height,img_width),
    batch_size=batch_size
)
```

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

```
num_classes=len(class_names)
```

```
model0=Sequential()
#layer 1
model0.add(Conv2D(filters=96,input_shape=(227,227,3),kernel_size=(11,11),strides=(4,4),padding='valid'))
model0.add(Activation("relu"))
model0.add(MaxPooling2D(pool_size=(3,3),strides=(2,2),padding='valid'))

#layer2
model0.add(Conv2D(filters=96,input_shape=(227,227,3),kernel_size=(5,5),strides=(1,1),padding='valid'))
model0.add(Activation("relu"))
model0.add(MaxPooling2D(pool_size=(3,3),strides=(2,2),padding='valid'))

#layer3
model0.add(Conv2D(filters=384,kernel_size=(3,3),strides=(1,1),padding='valid'))
model0.add(Activation("relu"))


#layer4
model0.add(Conv2D(filters=384,kernel_size=(3,3),strides=(1,1),padding='valid'))
model0.add(Activation("relu"))

#layer5
model0.add(Conv2D(filters=256,kernel_size=(3,3),strides=(1,1),padding='valid'))
model0.add(Activation("relu"))

#final pooling
model0.add(MaxPooling2D(pool_size=(2,2),strides=(2,2),padding='valid'))

#fully connected
model0.add(Flatten())

#1st FC
model0.add(Dense(4096, input_shape=(227*227*3,)))
model0.add(Activation('relu'))
```

```
#dropout to prevent overfitting
model0.add(Dropout(0.4))

#2nd FC
model0.add(Dense(4096))
model0.add(Activation('relu'))
#dropout
model0.add(Dropout(0.4))

#3rd FC
model0.add(Dense(1000))
model0.add(Activation('relu'))
#dropout
model0.add(Dropout(0.4))

#output layer
model0.add(Dense(num_classes))
model0.add(Activation('relu'))


#model compile
model0.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True), #categorical clasification
              metrics=['accuracy'])


model0.summary()
```

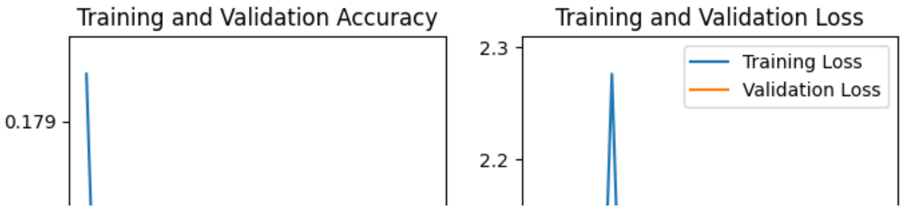
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_5 (Conv2D)           (None, 55, 55, 96)        34944     
                                                                 
 activation_9 (Activation)   (None, 55, 55, 96)        0         
                                                                 
 max_pooling2d_5 (MaxPooling  (None, 27, 27, 96)       0         
 2D)                                                             
                                                                 
 conv2d_6 (Conv2D)           (None, 23, 23, 96)        230496    
                                                                 
 activation_10 (Activation)  (None, 23, 23, 96)        0         
                                                                 
 max_pooling2d_6 (MaxPooling  (None, 11, 11, 96)       0         
 2D)                                                             
                                                                 
 conv2d_7 (Conv2D)           (None, 9, 9, 384)         332160    
                                                                 
 activation_11 (Activation)  (None, 9, 9, 384)         0         
                                                                 
 conv2d_8 (Conv2D)           (None, 7, 7, 384)         1327488   
                                                                 
 activation_12 (Activation)  (None, 7, 7, 384)         0         
                                                                 
 conv2d_9 (Conv2D)           (None, 5, 5, 256)         884992    
                                                                 
 activation_13 (Activation)  (None, 5, 5, 256)         0         
                                                                 
 max_pooling2d_7 (MaxPooling  (None, 2, 2, 256)        0         
 2D)                                                             
                                                                 
 flatten_1 (Flatten)         (None, 1024)              0         
                                                                 
 dense_4 (Dense)             (None, 4096)              4198400   
                                                                 
 activation_14 (Activation)  (None, 4096)              0         
                                                                 
 dropout (Dropout)           (None, 4096)              0         
                                                                 
 dense_5 (Dense)             (None, 4096)              16781312  
                                                                 
 activation_15 (Activation)  (None, 4096)              0         
                                                                 
 dropout_1 (Dropout)         (None, 4096)              0         
                                                                 
 dense_6 (Dense)             (None, 1000)              4097000   
                                                                 
 activation_16 (Activation)  (None, 1000)              0         
                                                                 
 dropout_2 (Dropout)         (None, 1000)              0         
                                                                 
 dense_7 (Dense)             (None, 5)                 5005      
                                                                 
 activation_17 (Activation)  (None, 5)                 0         
                                                                 
=================================================================
#model fit
epochs=15
history0 = model0.fit(train_ds1, validation_data=val_ds1, epochs=epochs)
```

```
Epoch 1/15
23/23 [==============================] - 19s 430ms/step - loss: 1.8579 - accuracy: 0.1795 - val_loss: 1.6094 - val_accuracy: 0.1757
Epoch 2/15
23/23 [==============================] - 9s 307ms/step - loss: 1.6094 - accuracy: 0.1717 - val_loss: 1.6094 - val_accuracy: 0.1757
Epoch 3/15
23/23 [==============================] - 12s 436ms/step - loss: 1.6094 - accuracy: 0.1717 - val_loss: 1.6094 - val_accuracy: 0.1757
Epoch 4/15
23/23 [==============================] - 8s 270ms/step - loss: 2.2763 - accuracy: 0.1717 - val_loss: 1.6094 - val_accuracy: 0.1757
Epoch 5/15
23/23 [==============================] - 8s 266ms/step - loss: 1.6094 - accuracy: 0.1717 - val_loss: 1.6094 - val_accuracy: 0.1757
Epoch 6/15
23/23 [==============================] - 9s 308ms/step - loss: 1.6094 - accuracy: 0.1717 - val_loss: 1.6094 - val_accuracy: 0.1757
Epoch 7/15
23/23 [==============================] - 9s 321ms/step - loss: 1.6094 - accuracy: 0.1717 - val_loss: 1.6094 - val_accuracy: 0.1757
Epoch 8/15
23/23 [==============================] - 9s 322ms/step - loss: 1.6094 - accuracy: 0.1717 - val_loss: 1.6094 - val_accuracy: 0.1757
Epoch 9/15
23/23 [==============================] - 9s 316ms/step - loss: 1.6094 - accuracy: 0.1717 - val_loss: 1.6094 - val_accuracy: 0.1757
Epoch 10/15
23/23 [==============================] - 9s 271ms/step - loss: 1.6094 - accuracy: 0.1717 - val_loss: 1.6094 - val_accuracy: 0.1757
Epoch 11/15
23/23 [==============================] - 9s 318ms/step - loss: 1.6094 - accuracy: 0.1717 - val_loss: 1.6094 - val_accuracy: 0.1757
Epoch 12/15
23/23 [==============================] - 9s 311ms/step - loss: 1.6094 - accuracy: 0.1717 - val_loss: 1.6094 - val_accuracy: 0.1757
Epoch 13/15
23/23 [==============================] - 8s 269ms/step - loss: 1.6094 - accuracy: 0.1717 - val_loss: 1.6094 - val_accuracy: 0.1757
Epoch 14/15
23/23 [==============================] - 9s 300ms/step - loss: 1.6094 - accuracy: 0.1717 - val_loss: 1.6094 - val_accuracy: 0.1757
Epoch 15/15
23/23 [==============================] - 9s 305ms/step - loss: 1.6094 - accuracy: 0.1717 - val_loss: 1.6094 - val_accuracy: 0.1757
```

```
acc = history0.history['accuracy']
val_acc = history0.history['val_accuracy']
loss = history0.history['loss']
val_loss = history0.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()
```



5.06.2023

INCEPTION PRETRAINED

```
batch_size=12
img_height=224
img_width=224
```

```
train_ds=tf.keras.utils.image_dataset_from_directory(
    data_dir,validation_split=0.2,
    subset="training",
    seed=123, #the same set of images always go into the training, test, validation set.
    image_size=(img_height,img_width),
    batch_size=batch_size
)
```

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

```
val_ds=tf.keras.utils.image_dataset_from_directory(
    data_dir,validation_split=0.2,
    subset="validation",
    seed=123, #the same set of images always go into the training, test, validation set.
    image_size=(img_height,img_width),
    batch_size=batch_size
)
```

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

```
#cheching the batch_size of training dataset
for image_batch, labels_batch in train_ds:
  print(image_batch.shape)
  print(labels_batch.shape)
  break
```

```
     (12, 224, 224, 3)
     (12,)
```

```
normalization_layer = layers.Rescaling(1./255) #255 is constant pixel value
```

```
num_classes=print(len(class_names))
```

```
     5
```

```
#model create (
#num_classes = len(class_names)
#model = Sequential([
#    layers.Rescaling(1./255, input_shape=(img_height, img_width, 3)), #input layer includes normalisation, ,sizes, color indication
#   layers.Conv2D(16, 3, padding='same', activation='relu'), #convolution layer    #16 - no. of filters,  3 - 3X3 filter size
#   layers.MaxPooling2D(), #pooling layer
#    layers.Conv2D(32, 3, padding='same', activation='relu'), #convolution layer    #32 - no. of filters,  3 - 3X3 filter size
#    layers.MaxPooling2D(), #pooling layer   #it is "2D" based on the dimention of the input image    #"1D" Pooling-text,  #"3D" pooling-
#    layers.Conv2D(64, 3, padding='same', activation='relu'), #convolution layer    #64 - no. of filters,  3 - 3X3 filter size
#   layers.MaxPooling2D(), #pooling layer
#   layers.Flatten(), #converting the data into single-vector
#   layers.Dense(128, activation='relu'), #hidden layer1 with 128 neurons
#   layers.Dense(256, activation='relu'), #hidden layer2 with 256 neurons
#   layers.Dense(num_classes) #output layer  #num_classes gives me how many output layer i want.  i.,e = 5 .
#])
```

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

```
#inceptionv3 model - pretrained
InceptionV3_model=tf.keras.applications.InceptionV3(weights='imagenet',include_top=False,input_shape=(224,224,3))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_weights_tf_dim_orderin
87910968/87910968 [==============================] - 0s 0us/step
```

```
from keras.models import Model
```

```
for layer in InceptionV3_model.layers[:-15]:
  layer.trainable = False
x=InceptionV3_model.output
x=GlobalAveragePooling2D()(x)
x=Flatten()(x)
x=Dense(units=512,activation='softmax')(x)
x=Dropout(0.3)(x)
x=Dense(units=512,activation='softmax')(x)
x=Dropout(0.3)(x)
output=Dense(units=5,activation='softmax')(x)
model=Model(InceptionV3_model.input, output)
```

```
model.summary()
```

```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to                     
==================================================================================================
 input_1 (InputLayer)           [(None, 224, 224, 3  0           []                               
                                )]                                                                
                                                                                                  
 conv2d_10 (Conv2D)             (None, 111, 111, 32  864         ['input_1[0][0]']                
                                )                                                                 
                                                                                                  
 batch_normalization (BatchNorm  (None, 111, 111, 32  96         ['conv2d_10[0][0]']              
 alization)                     )                                                                 
                                                                                                  
 activation_18 (Activation)     (None, 111, 111, 32  0           ['batch_normalization[0][0]']    
                                )                                                                 
                                                                                                  
 conv2d_11 (Conv2D)             (None, 109, 109, 32  9216        ['activation_18[0][0]']          
                                )                                                                 
                                                                                                  
 batch_normalization_1 (BatchNo  (None, 109, 109, 32  96         ['conv2d_11[0][0]']              
 rmalization)                   )                                                                 
                                                                                                  
 activation_19 (Activation)     (None, 109, 109, 32  0           ['batch_normalization_1[0][0]']  
                                )                                                                 
                                                                                                  
 conv2d_12 (Conv2D)             (None, 109, 109, 64  18432       ['activation_19[0][0]']          
                                )                                                                 
                                                                                                  
 batch_normalization_2 (BatchNo  (None, 109, 109, 64  192        ['conv2d_12[0][0]']              
 rmalization)                   )                                                                 
                                                                                                  
 activation_20 (Activation)     (None, 109, 109, 64  0           ['batch_normalization_2[0][0]']  
                                )                                                                 
                                                                                                  
 max_pooling2d_8 (MaxPooling2D)  (None, 54, 54, 64)  0           ['activation_20[0][0]']          
                                                                                                  
 conv2d_13 (Conv2D)             (None, 54, 54, 80)   5120        ['max_pooling2d_8[0][0]']        
                                                                                                  
 batch_normalization_3 (BatchNo  (None, 54, 54, 80)  240         ['conv2d_13[0][0]']              
 rmalization)                                                                                     
                                                                                                  
 activation_21 (Activation)     (None, 54, 54, 80)   0           ['batch_normalization_3[0][0]']  
                                                                                                  
 conv2d_14 (Conv2D)             (None, 52, 52, 192)  138240      ['activation_21[0][0]']          
                                                                                                  
 batch_normalization_4 (BatchNo  (None, 52, 52, 192)  576        ['conv2d_14[0][0]']              
 rmalization)                                                                                     
                                                                                                  
 activation_22 (Activation)     (None, 52, 52, 192)  0           ['batch_normalization_4[0][0]']  
                                                                                                  
 max_pooling2d_9 (MaxPooling2D)  (None, 25, 25, 192)  0          ['activation_22[0][0]']          
                                                                                                  
 conv2d_18 (Conv2D)             (None, 25, 25, 64)   12288       ['max_pooling2d_9[0][0]']        
                                                                                                  
 batch_normalization_8 (BatchNo  (None, 25, 25, 64)  192         ['conv2d_18[0][0]']              
 rmalization)                                                                                     
                                                                                                  
 activation_26 (Activation)     (None, 25, 25, 64)   0           ['batch_normalization_8[0][0]']  
```

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

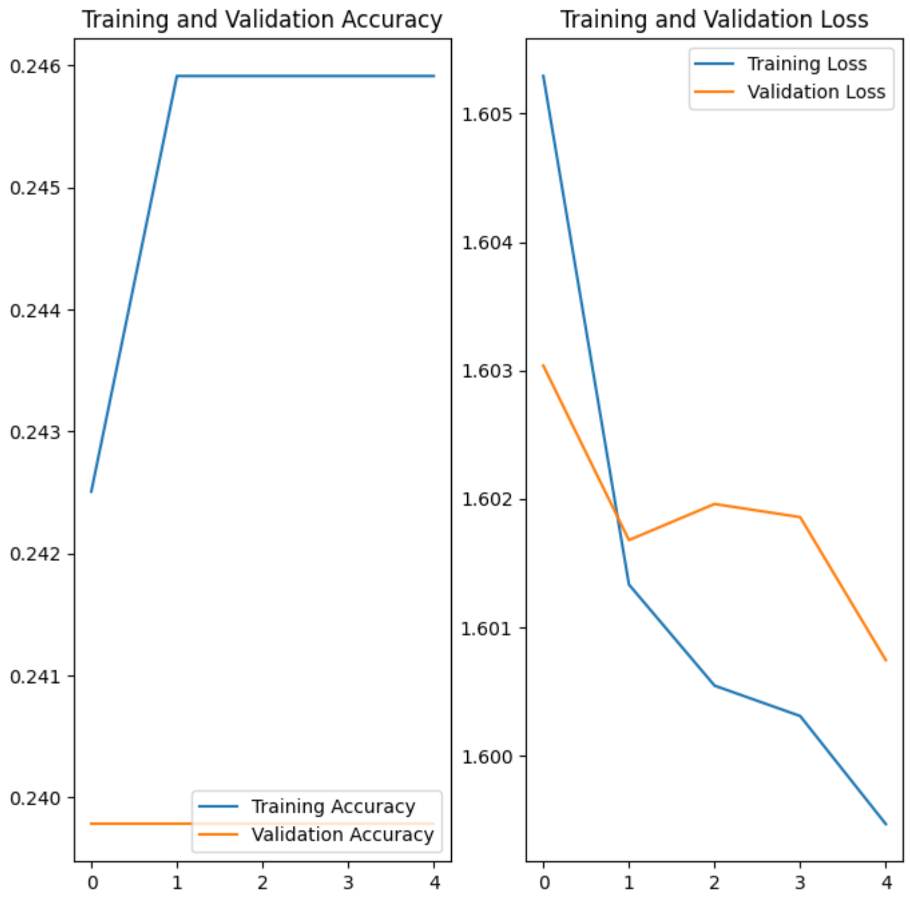
```
epochs=5
history=model.fit(train_ds,validation_data=val_ds,epochs=epochs)
```

```
Epoch 1/5
245/245 [==============================] - 26s 77ms/step - loss: 1.6053 - accuracy: 0.2425 - val_loss: 1.6030 - val_accuracy: 0.239
Epoch 2/5
245/245 [==============================] - 12s 48ms/step - loss: 1.6013 - accuracy: 0.2459 - val_loss: 1.6017 - val_accuracy: 0.239
Epoch 3/5
245/245 [==============================] - 12s 48ms/step - loss: 1.6005 - accuracy: 0.2459 - val_loss: 1.6020 - val_accuracy: 0.239
Epoch 4/5
245/245 [==============================] - 13s 53ms/step - loss: 1.6003 - accuracy: 0.2459 - val_loss: 1.6019 - val_accuracy: 0.239
Epoch 5/5
245/245 [==============================] - 12s 48ms/step - loss: 1.5995 - accuracy: 0.2459 - val_loss: 1.6007 - val_accuracy: 0.239
```

```
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()
```



VGG16

```
pretrained_model=tf.keras.applications.VGG16(input_shape = (224,224, 3),
                        include_top = False,
                        weights ='imagenet')
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_n
58889256/58889256 [==============================] - 0s 0us/step
```

```
for layer in pretrained_model.layers[:-15]:
  layer.trainable = False
x=pretrained_model.output
x=GlobalAveragePooling2D()(x)
x=Flatten()(x)
x=Dense(units=512,activation='softmax')(x)
x=Dropout(0.3)(x)
x=Dense(units=512,activation='softmax')(x)
x=Dropout(0.3)(x)
output=Dense(units=5,activation='softmax')(x)
model=Model(pretrained_model.input, output)
```

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

```
model.summary()
```

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_2 (InputLayer)        [(None, 224, 224, 3)]     0         
                                                                 
 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792      
                                                                 
 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928     
                                                                 
 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0         
                                                                 
 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856     
                                                                 
 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584    
                                                                 
 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0         
                                                                 
 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168    
                                                                 
 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080    
                                                                 
 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080    
                                                                 
 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0         
                                                                 
 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160   
                                                                 
 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808   
                                                                 
 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808   
                                                                 
 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0         
                                                                 
 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808   
                                                                 
 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808   
                                                                 
 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808   
                                                                 
 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0         
                                                                 
 global_average_pooling2d_2   (None, 512)              0         
 (GlobalAveragePooling2D)                                        
                                                                 
 flatten_4 (Flatten)         (None, 512)               0         
                                                                 
 dense_14 (Dense)            (None, 512)               262656    
                                                                 
 dropout_7 (Dropout)         (None, 512)               0         
                                                                 
 dense_15 (Dense)            (None, 512)               262656    
                                                                 
 dropout_8 (Dropout)         (None, 512)               0         
                                                                 
 dense_16 (Dense)            (None, 5)                 2565      
                                                                 
=================================================================
```

```
epochs=5
history1=history=model.fit(train_ds,validation_data=val_ds,epochs=epochs)
```

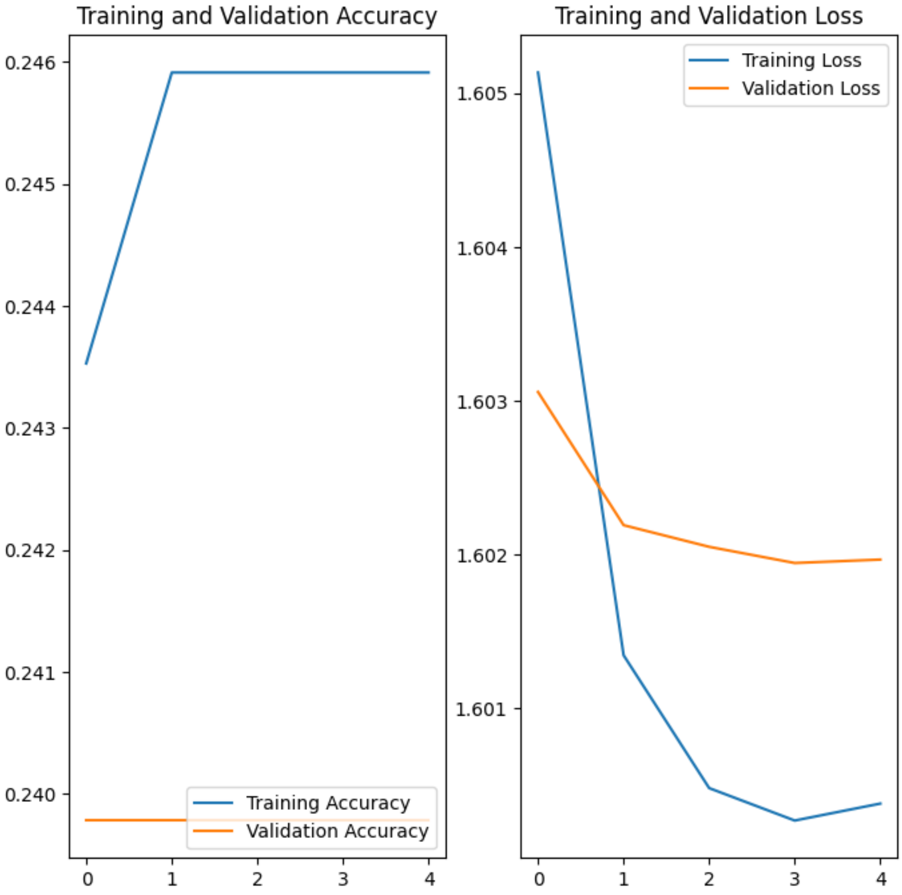
```
Epoch 1/5
245/245 [==============================] - 53s 169ms/step - loss: 1.6051 - accuracy: 0.2435 - val_loss: 1.6031 - val_accuracy: 0.23
Epoch 2/5
```

```
245/245 [==============================] - 39s 157ms/step - loss: 1.6013 - accuracy: 0.2459 - val_loss: 1.6022 - val_accuracy: 0.23
Epoch 3/5
245/245 [==============================] - 39s 158ms/step - loss: 1.6005 - accuracy: 0.2459 - val_loss: 1.6021 - val_accuracy: 0.23
Epoch 4/5
245/245 [==============================] - 39s 160ms/step - loss: 1.6003 - accuracy: 0.2459 - val_loss: 1.6019 - val_accuracy: 0.23
Epoch 5/5
245/245 [==============================] - 40s 162ms/step - loss: 1.6004 - accuracy: 0.2459 - val_loss: 1.6020 - val_accuracy: 0.23
```

```
acc = history1.history['accuracy']
val_acc = history1.history['val_accuracy']
loss = history1.history['loss']
val_loss = history1.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()
```



VGG19

```
pretrained_model1=tf.keras.applications.VGG19(input_shape = (224,224,3),
                        include_top = False,
                        weights ='imagenet')
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg19/vgg19_weights_tf_dim_ordering_tf_kernels_n
80134624/80134624 [==============================] - 0s 0us/step
```

```
for layer in pretrained_model1.layers[:-15]:
  layer.trainable = False
x=pretrained_model1.output
x=GlobalAveragePooling2D()(x)
x=Flatten()(x)
x=Dense(units=512,activation='softmax')(x)
x=Dropout(0.3)(x)
x=Dense(units=512,activation='softmax')(x)
x=Dropout(0.3)(x)
output=Dense(units=5,activation='softmax')(x)
model=Model(pretrained_model1.input, output)
```

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

```
model.summary()
```

```
Model: "model_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_3 (InputLayer)        [(None, 224, 224, 3)]     0         
                                                                 
 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792      
                                                                 
 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928     
                                                                 
 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0         
                                                                 
 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856     
                                                                 
 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584    
                                                                 
 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0         
                                                                 
 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168    
                                                                 
 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080    
                                                                 
 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080    
                                                                 
 block3_conv4 (Conv2D)       (None, 56, 56, 256)       590080    
                                                                 
 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0         
                                                                 
 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160   
                                                                 
 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808   
                                                                 
 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808   
                                                                 
 block4_conv4 (Conv2D)       (None, 28, 28, 512)       2359808   
                                                                 
 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0         
                                                                 
 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808   
                                                                 
 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808   
                                                                 
 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808   
                                                                 
 block5_conv4 (Conv2D)       (None, 14, 14, 512)       2359808   
                                                                 
 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0         
                                                                 
 global_average_pooling2d_3   (None, 512)              0         
 (GlobalAveragePooling2D)                                        
                                                                 
 flatten_5 (Flatten)         (None, 512)               0         
                                                                 
 dense_17 (Dense)            (None, 512)               262656    
                                                                 
 dropout_9 (Dropout)         (None, 512)               0         
                                                                 
 dense_18 (Dense)            (None, 512)               262656    
```

```
epochs=5
history1=history=model.fit(train_ds,validation_data=val_ds,epochs=epochs)
```

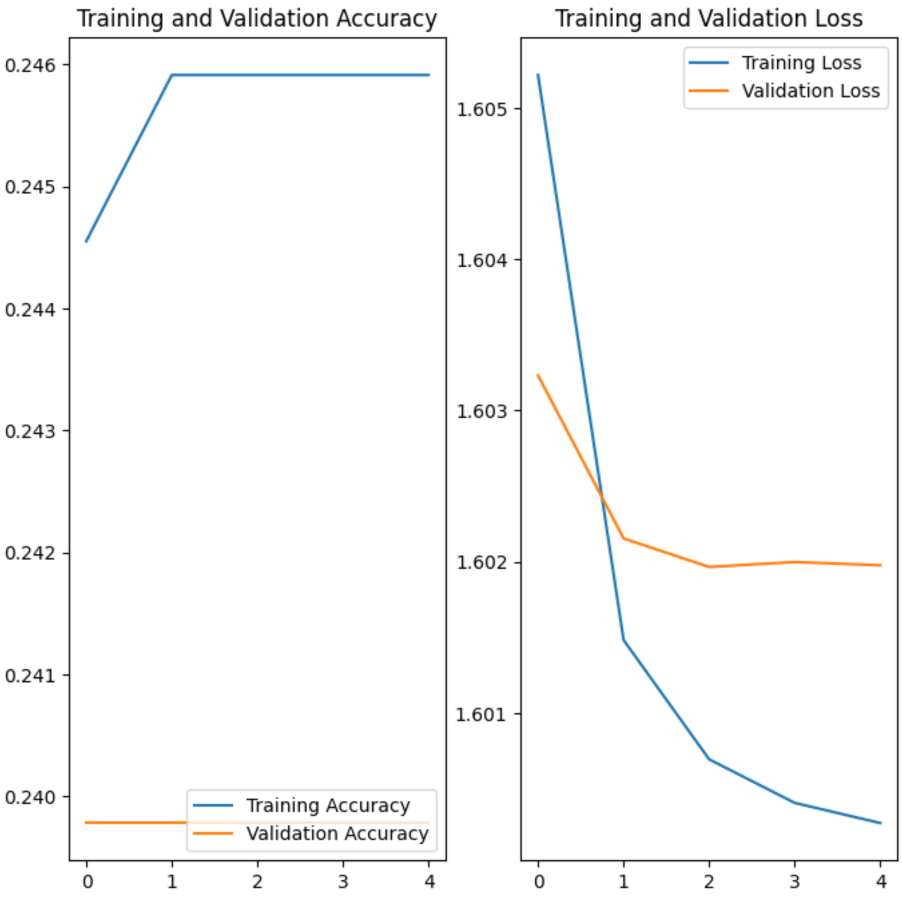
```
Epoch 1/5
245/245 [==============================] - 48s 174ms/step - loss: 1.6052 - accuracy: 0.2446 - val_loss: 1.6032 - val_accuracy: 0.23
Epoch 2/5
245/245 [==============================] - 42s 171ms/step - loss: 1.6015 - accuracy: 0.2459 - val_loss: 1.6022 - val_accuracy: 0.23
Epoch 3/5
245/245 [==============================] - 43s 175ms/step - loss: 1.6007 - accuracy: 0.2459 - val_loss: 1.6020 - val_accuracy: 0.23
Epoch 4/5
245/245 [==============================] - 43s 173ms/step - loss: 1.6004 - accuracy: 0.2459 - val_loss: 1.6020 - val_accuracy: 0.23
```

```
Epoch 5/5
245/245 [==============================] - 43s 177ms/step - loss: 1.6003 - accuracy: 0.2459 - val_loss: 1.6020 - val_accuracy: 0.23
```

```
acc = history1.history['accuracy']
val_acc = history1.history['val_accuracy']
loss = history1.history['loss']
val_loss = history1.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()
```



RESNET50

```
import cv2
import numpy as np
import os
from keras.preprocessing.image import ImageDataGenerator
from keras import backend as k
from keras.models import Model, load_model
from keras.optimizers import SGD
from keras.callbacks import EarlyStopping, ModelCheckpoint
from google.colab.patches import cv2_imshow
from keras.layers import Input, Dense,Activation, ZeroPadding2D, BatchNormalization, Flatten, Conv2D, AveragePooling2D, MaxPooling2D
from keras.preprocessing import image
from keras.initializers import  glorot_uniform
```

```
img_height,img_width = 224,224
num_classes=5
#if imgaenet weights are being loaded
#input must have a static square shape(one of (128,128),(160,160),(192,192)or(224,224))
```

```
base_model=tf.keras.applications.ResNet50(weights=None, include_top=False,input_shape=(img_height,img_width,3))
```

```
x=base_model.output
x=GlobalAveragePooling2D()(x)
x=Dropout(0.7)(x)
output=Dense(units=5,activation='softmax')(x)
model=Model(base_model.input, output)
```

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

```
model.summary()
```

```
Model: "model_4"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to                     
==================================================================================================
 input_5 (InputLayer)           [(None, 224, 224, 3  0           []                               
                                )]                                                                
                                                                                                  
 conv1_pad (ZeroPadding2D)      (None, 230, 230, 3)  0           ['input_5[0][0]']                
                                                                                                  
 conv1_conv (Conv2D)            (None, 112, 112, 64  9472        ['conv1_pad[0][0]']              
                                )                                                                 
                                                                                                  
 conv1_bn (BatchNormalization)  (None, 112, 112, 64  256         ['conv1_conv[0][0]']             
                                )                                                                 
                                                                                                  
 conv1_relu (Activation)        (None, 112, 112, 64  0           ['conv1_bn[0][0]']               
                                )                                                                 
                                                                                                  
 pool1_pad (ZeroPadding2D)      (None, 114, 114, 64  0           ['conv1_relu[0][0]']             
                                )                                                                 
                                                                                                  
 pool1_pool (MaxPooling2D)      (None, 56, 56, 64)   0           ['pool1_pad[0][0]']              
                                                                                                  
 conv2_block1_1_conv (Conv2D)   (None, 56, 56, 64)   4160        ['pool1_pool[0][0]']             
                                                                                                  
 conv2_block1_1_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2_block1_1_conv[0][0]']    
 ization)                                                                                         
                                                                                                  
 conv2_block1_1_relu (Activatio  (None, 56, 56, 64)  0           ['conv2_block1_1_bn[0][0]']      
 n)                                                                                               
                                                                                                  
 conv2_block1_2_conv (Conv2D)   (None, 56, 56, 64)   36928       ['conv2_block1_1_relu[0][0]']    
                                                                                                  
 conv2_block1_2_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2_block1_2_conv[0][0]']    
 ization)                                                                                         
                                                                                                  
 conv2_block1_2_relu (Activatio  (None, 56, 56, 64)  0           ['conv2_block1_2_bn[0][0]']      
 n)                                                                                               
                                                                                                  
 conv2_block1_0_conv (Conv2D)   (None, 56, 56, 256)  16640       ['pool1_pool[0][0]']             
                                                                                                  
 conv2_block1_3_conv (Conv2D)   (None, 56, 56, 256)  16640       ['conv2_block1_2_relu[0][0]']    
                                                                                                  
 conv2_block1_0_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2_block1_0_conv[0][0]']    
 ization)                                                                                         
                                                                                                  
 conv2_block1_3_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2_block1_3_conv[0][0]']    
 ization)                                                                                         
                                                                                                  
 conv2_block1_add (Add)         (None, 56, 56, 256)  0           ['conv2_block1_0_bn[0][0]',      
                                                                  'conv2_block1_3_bn[0][0]']      
                                                                                                  
 conv2_block1_out (Activation)  (None, 56, 56, 256)  0           ['conv2_block1_add[0][0]']       
                                                                                                  
 conv2_block2_1_conv (Conv2D)   (None, 56, 56, 64)   16448       ['conv2_block1_out[0][0]']       
                                                                                                  
 conv2_block2_1_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2_block2_1_conv[0][0]']    
 ization)                                                                                         
```

```
epochs=5
history1=history=model.fit(train_ds,validation_data=val_ds,epochs=epochs)
```

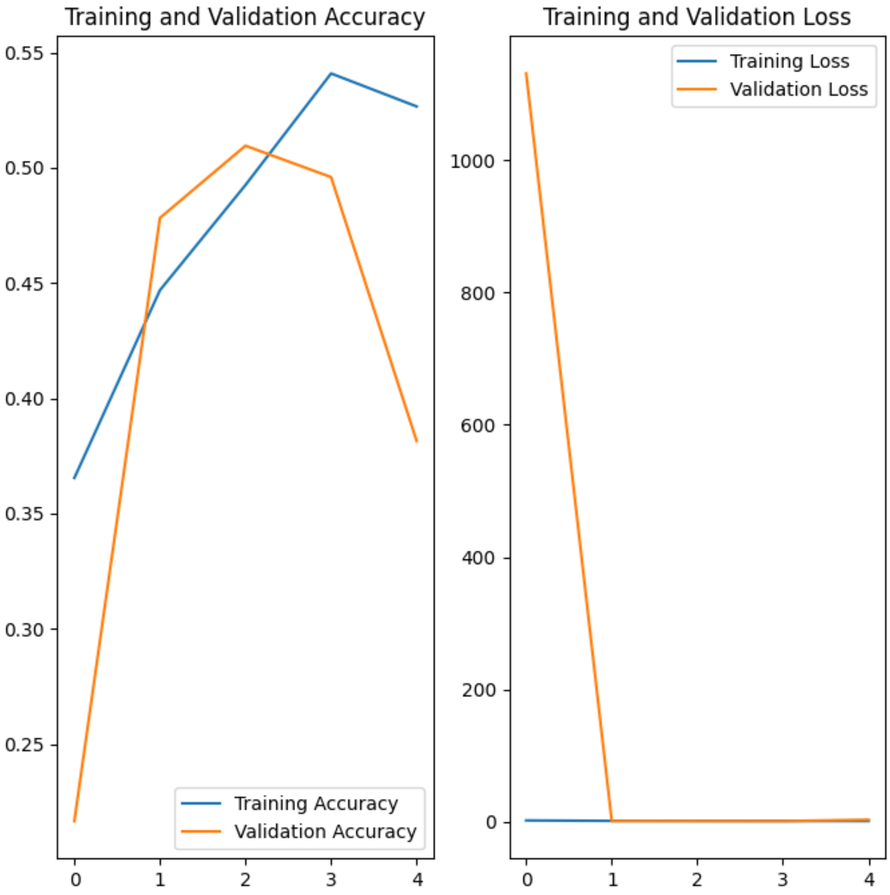
```
Epoch 1/5
245/245 [==============================] - 84s 168ms/step - loss: 2.2050 - accuracy: 0.3655 - val_loss: 1130.4174 - val_accuracy: 0
Epoch 2/5
245/245 [==============================] - 38s 153ms/step - loss: 1.5068 - accuracy: 0.4469 - val_loss: 1.3025 - val_accuracy: 0.47
Epoch 3/5
245/245 [==============================] - 40s 163ms/step - loss: 1.3604 - accuracy: 0.4925 - val_loss: 1.3428 - val_accuracy: 0.50
```

```
Epoch 4/5
245/245 [==============================] - 38s 155ms/step - loss: 1.2807 - accuracy: 0.5409 - val_loss: 1.1930 - val_accuracy: 0.49
Epoch 5/5
245/245 [==============================] - 38s 156ms/step - loss: 1.2691 - accuracy: 0.5266 - val_loss: 3.2465 - val_accuracy: 0.38
```

```
acc = history1.history['accuracy']
val_acc = history1.history['val_accuracy']
loss = history1.history['loss']
val_loss = history1.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()
```


Training and Validation Accuracy
Training Accuracy
Validation Accuracy
Training and Validation Loss
Training Loss
Validation Loss