In [1]:
```
import tensorflow as tf
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications import InceptionV3
from tensorflow.keras import models,layers
from tensorflow.keras.regularizers import l2
import math
```

In [2]:
```
df = tf.keras.preprocessing.image_dataset_from_directory(r"C:\Users\soura\OneDrive\Des
                                                         shuffle=True ,
                                                         image_size = (256,256),
                                                         batch_size = 32)
```

```
Found 10000 files belonging to 2 classes.
```

In [3]:
```
train_size = 0.7
len(df)*train_size
```

Out[3]:
```
219.1
```

In [4]:
```
train_ds = df.take(219)
len(train_ds)
```

Out[4]:
```
219
```

In [5]:
```
test_ds = df.skip(219)
len(test_ds)
```

Out[5]:
```
94
```

In [6]:
```
val_ds = test_ds.take(94)
len(val_ds)
```

Out[6]:
```
94
```

In [7]:
```
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
val_ds = val_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
test_ds = test_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
```

In [8]:
```
resize_and_rescale = tf.keras.Sequential([
  layers.experimental.preprocessing.Resizing(224,224),
  layers.experimental.preprocessing.Rescaling(1./255),
])
```

In [9]:
```
@tf.function
def preprocess_data(x, y):
    x = tf.image.random_flip_left_right(x)
    x = tf.image.random_flip_up_down(x)
    k = tf.random.uniform(shape=[], minval=0, maxval=4, dtype=tf.int32)
    x = tf.image.rot90(x, k=k)
    x = tf.image.random_brightness(x, max_delta=0.1)
    x = tf.image.random_contrast(x, lower=0.8, upper=1.2)
    x = tf.image.random_hue(x, max_delta=0.1)
    x = tf.image.per_image_standardization(x)
```

```
    return x, y
batch_size = 32

preprocessed_train_ds = train_ds.map(preprocess_data).prefetch(buffer_size=tf.data.AU
preprocessed_val_ds = val_ds.map(preprocess_data).prefetch(buffer_size=tf.data.AUTOTU
```

In [10]:
```
num_classes = 2
input_shape = (256,256, 3)
initial_learning_rate = 0.001
decay_rate = 0.1
decay_steps = 10
```

In [11]:
```
base_model = InceptionV3(weights='imagenet', include_top=False, input_shape=input_sha

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu', kernel_regularizer=l2(0.01))(x)
x = Dropout(0.5)(x)
x = Dense(512, activation='relu', kernel_regularizer=l2(0.01))(x)
x = Dropout(0.5)(x)
x = Dense(256, activation='relu', kernel_regularizer=l2(0.01))(x)
predictions = Dense(num_classes, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=predictions)
for layer in base_model.layers:
    layer.trainable = False
```

In [12]:
```
def exponential_decay(epoch, initial_lr=0.001, decay_rate=0.1, decay_steps=10):
    return initial_lr * math.pow(decay_rate, epoch / decay_steps)
lr_scheduler = tf.keras.callbacks.LearningRateScheduler(exponential_decay)
```

In [13]:
```
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),  # Set the init
              loss='sparse_categorical_crossentropy',
               metrics=['accuracy'])
```

In [14]:
```
BATCH_SIZE = 32
history = model.fit(
    preprocessed_train_ds,
    batch_size=BATCH_SIZE,
    validation_data=preprocessed_val_ds,
    epochs=10,
    callbacks=[lr_scheduler],
    verbose=1
)
```

```
Epoch 1/10
219/219 [==============================] - 788s 3s/step - loss: 6.7140 - accuracy: 0.
6962 - val_loss: 1.8385 - val_accuracy: 0.8055 - lr: 0.0010
Epoch 2/10
219/219 [==============================] - 681s 3s/step - loss: 1.3147 - accuracy: 0.
7944 - val_loss: 0.9154 - val_accuracy: 0.8265 - lr: 7.9433e-04
Epoch 3/10
219/219 [==============================] - 679s 3s/step - loss: 0.8095 - accuracy: 0.
8138 - val_loss: 0.6854 - val_accuracy: 0.8342 - lr: 5.0119e-04
Epoch 4/10
219/219 [==============================] - 663s 3s/step - loss: 0.6592 - accuracy: 0.
8269 - val_loss: 0.5891 - val_accuracy: 0.8389 - lr: 2.5119e-04
Epoch 5/10
219/219 [==============================] - 664s 3s/step - loss: 0.5897 - accuracy: 0.
8352 - val_loss: 0.5663 - val_accuracy: 0.8339 - lr: 1.0000e-04
Epoch 6/10
219/219 [==============================] - 657s 3s/step - loss: 0.5611 - accuracy: 0.
8408 - val_loss: 0.5573 - val_accuracy: 0.8399 - lr: 3.1623e-05
Epoch 7/10
219/219 [==============================] - 651s 3s/step - loss: 0.5541 - accuracy: 0.
8393 - val_loss: 0.5580 - val_accuracy: 0.8326 - lr: 7.9433e-06
Epoch 8/10
219/219 [==============================] - 657s 3s/step - loss: 0.5467 - accuracy: 0.
8467 - val_loss: 0.5433 - val_accuracy: 0.8479 - lr: 1.5849e-06
Epoch 9/10
219/219 [==============================] - 645s 3s/step - loss: 0.5441 - accuracy: 0.
8457 - val_loss: 0.5546 - val_accuracy: 0.8376 - lr: 2.5119e-07
Epoch 10/10
219/219 [==============================] - 649s 3s/step - loss: 0.5434 - accuracy: 0.
8485 - val_loss: 0.5456 - val_accuracy: 0.8416 - lr: 3.1623e-08
```

In [16]:

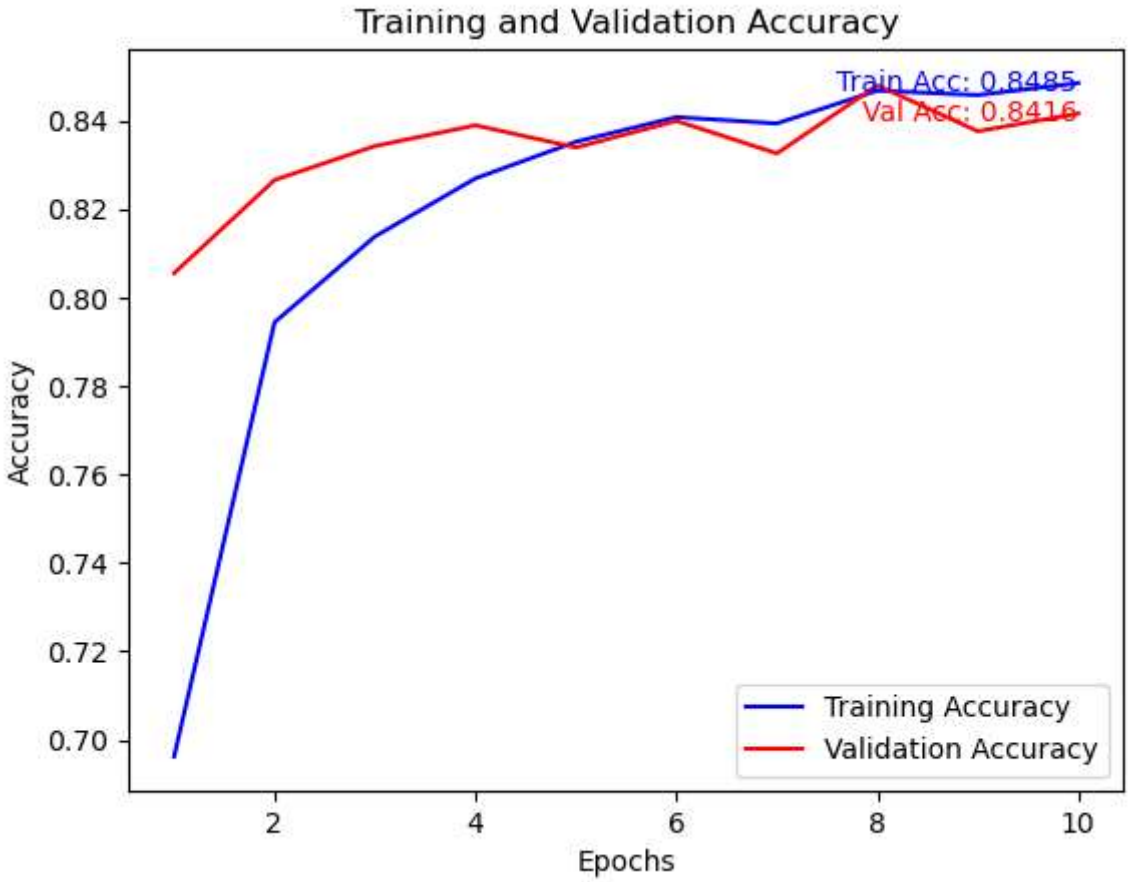
```
import matplotlib.pyplot as plt
train_acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

final_train_acc = train_acc[-1]
final_val_acc = val_acc[-1]

epochs = range(1, len(train_acc) + 1)
plt.plot(epochs, train_acc, 'b', label='Training Accuracy')
plt.plot(epochs, val_acc, 'r', label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.text(epochs[-1], final_train_acc, f'Train Acc: {final_train_acc:.4f}', ha='right',
plt.text(epochs[-1], final_val_acc, f'Val Acc: {final_val_acc:.4f}', ha='right', va='

plt.show()
```


Training and Validation Accuracy
Train Acc: 0.8485
Val Acc: 0.8416
Accuracy
0.84
0.82
0.80
0.78
0.76
0.74
0.72
0.70
2
4
6
8
10
Epochs
Training Accuracy
Validation Accuracy

In [ ]: