```
import os
import os.path
import pandas as pd
import zipfile
import numpy as np
import tensorflow as tf
import glob
from tensorflow import keras
from tensorflow.keras import layers
```

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
## Labelling the two folders
```

```
!cp -r "/content/drive/MyDrive/Capstone_project/COVID" .
```

```
df = pd.DataFrame({"File_path":['/content/drive/MyDrive/Capstone_project/COVID/Covid (1).png','/content/drive/MyDrive/Capstone_project/COVID/
                "Labels":['covid','covid']})
```

```
df
```

	File_path	Labels
0	/content/drive/MyDrive/Capstone_project/COVID/...	covid
1	/content/drive/MyDrive/Capstone_project/COVID/...	covid

```
!cp -r "/content/drive/MyDrive/Capstone_project/non-COVID" .
```

```
df = pd.DataFrame({
    "File_path": ['/content/drive/MyDrive/Capstone_project/non-COVID/Non-Covid (1).png','/content/drive/MyDrive/Capstone_project/non-COVID/Nc
    "Labels": ['non_covid', 'non_covid']})
```

```
df
```

	File_path	Labels
0	/content/drive/MyDrive/Capstone_project/non-CO...	non_covid
1	/content/drive/MyDrive/Capstone_project/non-CO...	non_covid

```
file_nameC = []
labelC = []

for i in (glob.glob('/content/drive/MyDrive/Capstone_project/COVID'+'*.png')[:5]):
  file_nameC.append(i)
  labelC.append(i.split('/')[-2])
```

```
file_nameNC = []
labelNC = []

for i in (glob.glob('/content/drive/MyDrive/Capstone_project/non-COVID'+'*.png')[:5]):
  file_nameNC.append(i)
  labelNC.append(i.split('/')[-2])
```

```
file_nameC = []
labelC = []
for i in glob.glob('/content/drive/MyDrive/Capstone_project/'+'*/*.png'):
  file_nameC.append(i)
  labelC.append(i.split('/')[-2])
```

```
master_data = pd.DataFrame({"File_Name":file_nameC,"Label":labelC})
```

```
master_data
```

	File_Name	Label
0	/content/drive/MyDrive/Capstone_project/non-CO...	non-COVID
1	/content/drive/MyDrive/Capstone_project/non-CO...	non-COVID
2	/content/drive/MyDrive/Capstone_project/non-CO...	non-COVID
3	/content/drive/MyDrive/Capstone_project/non-CO...	non-COVID
4	/content/drive/MyDrive/Capstone_project/non-CO...	non-COVID
...	...	...
2475	/content/drive/MyDrive/Capstone_project/COVID/...	COVID
2476	/content/drive/MyDrive/Capstone_project/COVID/...	COVID
2477	/content/drive/MyDrive/Capstone_project/COVID/...	COVID
2478	/content/drive/MyDrive/Capstone_project/COVID/...	COVID
2479	/content/drive/MyDrive/Capstone_project/COVID/...	COVID

2480 rows × 2 columns

```
master_data = master_data.sample(frac=1)
master_data = master_data.sample(frac=1).reset_index(drop=True)
```

```
filename=[]
for i in os.listdir('/content/drive/MyDrive/Capstone_project/COVID')[:5]:
  print(i)
  filename.append("/content/drive/MyDrive/Capstone_project/COVID"+i)
```

```
Covid (215).png
Covid (124).png
Covid (26).png
Covid (187).png
Covid (203).png
```

```
master_data['Label']=master_data['Label'].replace({'COVID':0,'non-COVID':1})
```

```
from sklearn.model_selection import train_test_split
train_df,test_df = train_test_split(master_data,test_size=0.25,shuffle=True,random_state=42)
```

```
train_df
```

	File_Name	Label
1234	/content/drive/MyDrive/Capstone_project/non-CO...	1
2126	/content/drive/MyDrive/Capstone_project/COVID/...	0
1299	/content/drive/MyDrive/Capstone_project/non-CO...	1
2012	/content/drive/MyDrive/Capstone_project/COVID/...	0
1123	/content/drive/MyDrive/Capstone_project/COVID/...	0
...	...	...
1638	/content/drive/MyDrive/Capstone_project/COVID/...	0
1095	/content/drive/MyDrive/Capstone_project/COVID/...	0
1130	/content/drive/MyDrive/Capstone_project/COVID/...	0
1294	/content/drive/MyDrive/Capstone_project/COVID/...	0
860	/content/drive/MyDrive/Capstone_project/non-CO...	1

1860 rows × 2 columns

```
test_df
```

	File_Name	Label
767	/content/drive/MyDrive/Capstone_project/non-CO...	1
259	/content/drive/MyDrive/Capstone_project/non-CO...	1
1068	/content/drive/MyDrive/Capstone_project/non-CO...	1
1769	/content/drive/MyDrive/Capstone_project/COVID/...	0
56	/content/drive/MyDrive/Capstone_project/COVID/...	0
...	...	...
1546	/content/drive/MyDrive/Capstone_project/COVID/...	0
834	/content/drive/MyDrive/Capstone_project/COVID/...	0
2036	/content/drive/MyDrive/Capstone_project/non-CO...	1
230	/content/drive/MyDrive/Capstone_project/COVID/...	0

```
print("This is my train data size --",train_df.shape)
print("This is my test data size --",test_df.shape)
```

```
This is my train data size -- (1860, 2)
This is my test data size -- (620, 2)
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
train_generator = ImageDataGenerator(
    rescale = 1./255,
    horizontal_flip=True,
    width_shift_range = 0.2,
    height_shift_range = 0.2,
    validation_split=0.2
)
```

```
train_generator
```

```
<keras.preprocessing.image.ImageDataGenerator at 0x7f75c731c8b0>
```

```
test_generator = ImageDataGenerator(
    rescale=1./255
)
```

```
train_df.head(10)
```

	File_Name	Label
1234	/content/drive/MyDrive/Capstone_project/non-CO...	1
2126	/content/drive/MyDrive/Capstone_project/COVID/...	0
1299	/content/drive/MyDrive/Capstone_project/non-CO...	1
2012	/content/drive/MyDrive/Capstone_project/COVID/...	0
1123	/content/drive/MyDrive/Capstone_project/COVID/...	0
1058	/content/drive/MyDrive/Capstone_project/non-CO...	1
374	/content/drive/MyDrive/Capstone_project/non-CO...	1
1994	/content/drive/MyDrive/Capstone_project/COVID/...	0
1502	/content/drive/MyDrive/Capstone_project/non-CO...	1
1222	/content/drive/MyDrive/Capstone_project/non-CO...	1

```
train_images = train_generator.flow_from_dataframe(
    dataframe=train_df,
    x_col ='File_Name',
    y_col = 'Label',
    target_size=(120,120),
    color_mode='rgb',
    class_mode='raw',
    batch_size=32,
```

```
    shuffle=True,
    subset='training'
)
```

```
Found 1488 validated image filenames.
```

```
val_images = train_generator.flow_from_dataframe(
    dataframe=train_df,
    x_col ='File_Name',
    y_col = 'Label',
    target_size=(120,120),
    color_mode='rgb',
    class_mode='raw',
    batch_size=32,
    shuffle=True,
    subset='validation'
)
```

```
Found 372 validated image filenames.
```

```
test_images = test_generator.flow_from_dataframe(
    dataframe = test_df,
    x_col='File_Name',
    y_col='Label',
    target_size=(120,120),
    color_mode='rgb',
    class_mode='raw',
    batch_size=8,
    shuffle=True
)
```

```
Found 620 validated image filenames.
```

```
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam

# Load the ResNet50 model with pre-trained weights and without top layers
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(120, 120, 3))

# Add new top layers to the ResNet50 model
x = base_model.output
x = Flatten()(x)
x = Dense(256, activation='relu')(x)
x = Dense(1, activation='sigmoid')(x)
model = Model(inputs=base_model.input, outputs=x)

# Freeze the pre-trained layers in the ResNet50 model
for layer in base_model.layers:
    layer.trainable = False

# Compile the model with Adam optimizer and binary cross-entropy loss
model.compile(optimizer=Adam(learning_rate=0.0001), loss='binary_crossentropy', metrics=['accuracy'])

# Train the model on the augmented images
history = model.fit(train_images, epochs=10, validation_data=test_images)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_n
94765736/94765736 [==============================] - 1s 0us/step
Epoch 1/10
47/47 [==============================] - 36s 486ms/step - loss: 0.6974 - accuracy: 0.5786 - val_loss: 0.7035 - val_accuracy: 0.5871
Epoch 2/10
47/47 [==============================] - 20s 426ms/step - loss: 0.6431 - accuracy: 0.6277 - val_loss: 0.6386 - val_accuracy: 0.6339
Epoch 3/10
47/47 [==============================] - 21s 449ms/step - loss: 0.6415 - accuracy: 0.6163 - val_loss: 0.7053 - val_accuracy: 0.5952
Epoch 4/10
47/47 [==============================] - 21s 441ms/step - loss: 0.6301 - accuracy: 0.6142 - val_loss: 0.6111 - val_accuracy: 0.6210
Epoch 5/10
47/47 [==============================] - 21s 446ms/step - loss: 0.6340 - accuracy: 0.6317 - val_loss: 0.6396 - val_accuracy: 0.6419
Epoch 6/10
47/47 [==============================] - 20s 419ms/step - loss: 0.6339 - accuracy: 0.6230 - val_loss: 0.6020 - val_accuracy: 0.6726
Epoch 7/10
47/47 [==============================] - 26s 550ms/step - loss: 0.6132 - accuracy: 0.6405 - val_loss: 0.5720 - val_accuracy: 0.7210
Epoch 8/10
47/47 [==============================] - 21s 442ms/step - loss: 0.6028 - accuracy: 0.6519 - val_loss: 0.6833 - val_accuracy: 0.6258
Epoch 9/10
```

```
47/47 [==============================] - 22s 471ms/step - loss: 0.6035 - accuracy: 0.6694 - val_loss: 0.6231 - val_accuracy: 0.6597
Epoch 10/10
47/47 [==============================] - 20s 423ms/step - loss: 0.6013 - accuracy: 0.6673 - val_loss: 0.5761 - val_accuracy: 0.6823
```

```
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

# Define early stopping callback to stop training when validation loss stops improving
early_stop = EarlyStopping(monitor='val_loss', patience=3, verbose=1, mode='min', restore_best_weights=True)

# Define model checkpoint callback to save the best model during training
checkpoint = ModelCheckpoint('best_model.h5', monitor='val_loss', save_best_only=True, mode='min', verbose=1)

# Train the model with callbacks
history = model.fit(train_images, epochs=10, validation_data=test_images, callbacks=[early_stop, checkpoint])
```

```
Epoch 1/10
47/47 [==============================] - ETA: 0s - loss: 0.6180 - accuracy: 0.6277
Epoch 1: val_loss improved from inf to 0.54705, saving model to best_model.h5
47/47 [==============================] - 23s 486ms/step - loss: 0.6180 - accuracy: 0.6277 - val_loss: 0.5471 - val_accuracy: 0.7290
Epoch 2/10
47/47 [==============================] - ETA: 0s - loss: 0.5888 - accuracy: 0.6788
Epoch 2: val_loss improved from 0.54705 to 0.53749, saving model to best_model.h5
47/47 [==============================] - 22s 467ms/step - loss: 0.5888 - accuracy: 0.6788 - val_loss: 0.5375 - val_accuracy: 0.7258
Epoch 3/10
47/47 [==============================] - ETA: 0s - loss: 0.5832 - accuracy: 0.6821
Epoch 3: val_loss improved from 0.53749 to 0.53301, saving model to best_model.h5
47/47 [==============================] - 21s 451ms/step - loss: 0.5832 - accuracy: 0.6821 - val_loss: 0.5330 - val_accuracy: 0.7371
Epoch 4/10
47/47 [==============================] - ETA: 0s - loss: 0.6006 - accuracy: 0.6573
Epoch 4: val_loss did not improve from 0.53301
47/47 [==============================] - 26s 563ms/step - loss: 0.6006 - accuracy: 0.6573 - val_loss: 0.5402 - val_accuracy: 0.7274
Epoch 5/10
47/47 [==============================] - ETA: 0s - loss: 0.5846 - accuracy: 0.6774
Epoch 5: val_loss did not improve from 0.53301
47/47 [==============================] - 20s 434ms/step - loss: 0.5846 - accuracy: 0.6774 - val_loss: 0.5370 - val_accuracy: 0.7242
Epoch 6/10
47/47 [==============================] - ETA: 0s - loss: 0.6077 - accuracy: 0.6566Restoring model weights from the end of the best epoc

Epoch 6: val_loss did not improve from 0.53301
47/47 [==============================] - 21s 446ms/step - loss: 0.6077 - accuracy: 0.6566 - val_loss: 0.5957 - val_accuracy: 0.6726
Epoch 6: early stopping
```

```
import numpy as np
from sklearn.metrics import classification_report, confusion_matrix

# Load the best saved model
model.load_weights('best_model.h5')

# Make predictions on the test images
y_pred = model.predict(test_images)

# Convert predictions from probabilities to binary labels
y_pred = np.where(y_pred > 0.5, 1, 0)

# Get true labels of the test images
y_true = test_df['Label']

# Evaluate the model's performance with classification report and confusion matrix
print(classification_report(y_true, y_pred))
print(confusion_matrix(y_true, y_pred))
```

```
78/78 [==============================] - 5s 66ms/step
              precision    recall  f1-score   support

           0       0.56      0.61      0.59       316
           1       0.56      0.50      0.53       304

    accuracy                           0.56       620
   macro avg       0.56      0.56      0.56       620
weighted avg       0.56      0.56      0.56       620

[[194 122]
 [151 153]]
```

✓ 5s completed at 1:08 PM