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

```
Mounted at /content/drive
```

```
import os
for dirname, _, filenames in os.walk('/content/drive/MyDrive/augmented data'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/content/drive/MyDrive/augmented data/no/aug_23 no_0_4438.jpg
/content/drive/MyDrive/augmented data/no/aug_23 no_0_5026.jpg
/content/drive/MyDrive/augmented data/no/aug_23 no_0_5432.jpg
/content/drive/MyDrive/augmented data/no/aug_23 no_0_6543.jpg
/content/drive/MyDrive/augmented data/no/aug_23 no_0_7084.jpg
/content/drive/MyDrive/augmented data/no/aug_23 no_0_8834.jpg
/content/drive/MyDrive/augmented data/no/aug_23 no_0_9109.jpg
/content/drive/MyDrive/augmented data/no/aug_24 no_0_1576.jpg
/content/drive/MyDrive/augmented data/no/aug_24 no_0_2171.jpg
/content/drive/MyDrive/augmented data/no/aug_24 no_0_3335.jpg
/content/drive/MyDrive/augmented data/no/aug_24 no_0_3666.jpg
/content/drive/MyDrive/augmented data/no/aug_24 no_0_3936.jpg
/content/drive/MyDrive/augmented data/no/aug_24 no_0_4650.jpg
/content/drive/MyDrive/augmented data/no/aug_24 no_0_4869.jpg
/content/drive/MyDrive/augmented data/no/aug_24 no_0_6273.jpg
/content/drive/MyDrive/augmented data/no/aug_24 no_0_7532.jpg
/content/drive/MyDrive/augmented data/no/aug_24 no_0_9627.jpg
/content/drive/MyDrive/augmented data/no/aug_25 no_0_211.jpg
/content/drive/MyDrive/augmented data/no/aug_25 no_0_2280.jpg
/content/drive/MyDrive/augmented data/no/aug_25 no_0_3207.jpg
/content/drive/MyDrive/augmented data/no/aug_25 no_0_3423.jpg
/content/drive/MyDrive/augmented data/no/aug_25 no_0_450.jpg
/content/drive/MyDrive/augmented data/no/aug_25 no_0_6282.jpg
/content/drive/MyDrive/augmented data/no/aug_25 no_0_6883.jpg
/content/drive/MyDrive/augmented data/no/aug_25 no_0_8689.jpg
/content/drive/MyDrive/augmented data/no/aug_25 no_0_8766.jpg
/content/drive/MyDrive/augmented data/no/aug_25 no_0_8833.jpg
/content/drive/MyDrive/augmented data/no/aug_26 no_0_3428.jpg
/content/drive/MyDrive/augmented data/no/aug_26 no_0_3441.jpg
/content/drive/MyDrive/augmented data/no/aug_26 no_0_3754.jpg
/content/drive/MyDrive/augmented data/no/aug_26 no_0_379.jpg
/content/drive/MyDrive/augmented data/no/aug_26 no_0_5268.jpg
/content/drive/MyDrive/augmented data/no/aug_26 no_0_7466.jpg
/content/drive/MyDrive/augmented data/no/aug_26 no_0_8162.jpg
/content/drive/MyDrive/augmented data/no/aug_26 no_0_9004.jpg
/content/drive/MyDrive/augmented data/no/aug_26 no_0_9106.jpg
/content/drive/MyDrive/augmented data/no/aug_26 no_0_9397.jpg
/content/drive/MyDrive/augmented data/no/aug_27 no_0_1521.jpg
/content/drive/MyDrive/augmented data/no/aug_27 no_0_1689.jpg
/content/drive/MyDrive/augmented data/no/aug_27 no_0_3315.jpg
/content/drive/MyDrive/augmented data/no/aug_27 no_0_5320.jpg
/content/drive/MyDrive/augmented data/no/aug_27 no_0_6550.jpg
/content/drive/MyDrive/augmented data/no/aug_27 no_0_7571.jpg
/content/drive/MyDrive/augmented data/no/aug_27 no_0_7667.jpg
/content/drive/MyDrive/augmented data/no/aug_27 no_0_9373.jpg
/content/drive/MyDrive/augmented data/no/aug_27 no_0_9639.jpg
/content/drive/MyDrive/augmented data/no/aug_27 no_0_9914.jpg
/content/drive/MyDrive/augmented data/no/aug_28 no_0_119.jpg
/content/drive/MyDrive/augmented data/no/aug_28 no_0_1395.jpg
/content/drive/MyDrive/augmented data/no/aug_28 no_0_5487.jpg
/content/drive/MyDrive/augmented data/no/aug_28 no_0_5806.jpg
/content/drive/MyDrive/augmented data/no/aug_28 no_0_6093.jpg
/content/drive/MyDrive/augmented data/no/aug_28 no_0_7484.jpg
/content/drive/MyDrive/augmented data/no/aug_28 no_0_7751.jpg
/content/drive/MyDrive/augmented data/no/aug_28 no_0_8398.jpg
/content/drive/MyDrive/augmented data/no/aug_28 no_0_9133.jpg
/content/drive/MyDrive/augmented data/no/aug_28 no_0_9753.jpg
/content/drive/MyDrive/augmented data/no/aug_29 no_0_1670.jpg
/content/drive/MyDrive/augmented data/no/aug_29 no_0_2763.jpg
```

```
train_dir="/content/drive/MyDrive/augmented data"
```

```
Classes = ['no','yes']
```

```
import cv2
import numpy as np
train_data = []
img_size=224
def get_training_data():
    for label in  Classes:
        path=os.path.join(train_dir, label)
        class_num = Classes.index(label)
        for img in os.listdir(path):
            try:
                img_arr = cv2.imread(os.path.join(path, img))
                resized_arr = cv2.resize(img_arr, (img_size, img_size))
                train_data.append([resized_arr, class_num])
            except Exception as e:
                pass


get_training_data()


print(len(train_data))
```

```
2065
```

```
for label in Classes:
    print(Classes.index(label))
```

```
0
1
```

```
x=[]
y=[]
for i,j in train_data:
  x.append(i)
  y.append(j)
x=np.array(x).reshape(-1,img_size, img_size,3)


x.shape
```

```
(2065, 224, 224, 3)
```

```
x=x/255.0


y=np.array(y)


print(y.shape)
print(y)
```

```
(2065,)
[0 0 0 ... 1 1 1]
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=True,test_size=0.2)


print(x_train.shape)
```

```
(1652, 224, 224, 3)
```

```
print(y_test)
```

```
[0 1 0 0 1 0 0 0 1 1 0 1 1 1 0 0 1 1 0 1 1 0 0 0 1 0 0 1 1 1 0 0 1 1 1 1 0
 1 0 0 0 1 0 0 0 1 0 0 1 1 0 1 1 0 1 1 1 0 0 0 1 0 0 1 0 1 1 1 1 1 0 0 1 1
 1 1 1 1 1 0 1 1 1 0 0 0 1 0 0 0 0 0 1 1 0 0 0 1 1 0 0 0 0 1 0 0 0 0 0 0 1
 1 1 0 1 1 1 1 1 0 1 1 0 1 0 1 1 1 1 1 1 0 0 0 0 0 1 0 1 1 1 1 0 1 1 1 1 1
 1 1 1 1 1 1 1 0 1 0 1 0 1 0 0 1 0 0 0 0 0 1 1 1 1 1 0 1 0 0 0 0 1 0 1 1 0
 0 0 1 0 0 0 1 0 1 0 1 1 1 1 0 0 0 1 1 1 1 0 0 1 0 1 0 0 1 0 0 0 1 0 1 0 0
 0 0 1 1 1 1 1 0 1 1 0 0 1 1 1 1 1 0 1 1 1 0 1 1 0 1 1 0 0 1 0 0 1 0 1 1 1
 0 0 0 0 0 1 0 1 0 1 0 0 1 0 0 0 1 1 0 0 1 1 1 0 0 1 1 0 1 0 0 0 1 0 0 0 0
 1 0 0 0 0 1 1 1 0 1 0 0 0 1 0 1 1 0 0 1 1 0 0 1 1 1 1 1 0 0 1 0 1 1 1 1 0
 0 1 1 0 1 0 1 0 1 1 0 1 0 1 1 0 0 0 1 1 0 1 0 0 0 1 0 0 1 1 1 1 1 1 1 0 1
 0 1 0 0 0 1 1 0 1 1 1 1 0 0 0 0 0 0 1 0 1 1 1 0 1 0 1 0 1 1 1 1 0 1 1 0 0
 1 0 1 1 1 1]
```

```
from tensorflow.keras.utils import to_categorical
y_train= to_categorical(y_train, num_classes=6)
```

```
from tensorflow.keras.utils import to_categorical
y_test= to_categorical(y_test, num_classes=6)
```

```
print(x_train.shape,y_test.shape)
```

```
(1652, 224, 224, 3) (413, 6)
```

MobileNet

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from tensorflow import keras
from tensorflow.keras import utils
import os
from keras.layers import Flatten, Dense
from keras.models import Model
from tensorflow.keras.utils import img_to_array,load_img
from keras.preprocessing.image import  ImageDataGenerator
from keras.applications.mobilenet import MobileNet, preprocess_input
from keras.losses import categorical_crossentropy
base_model = MobileNet( input_shape=(224,224,3), include_top= False )

for layer in base_model.layers:
  layer.trainable = False
```

```
x = Flatten()(base_model.output)
```

```
x = Dense(units=6 , activation='softmax' )(x)

# creating our model.
model = Model(base_model.input, x)
#model = Model(inputs=base_model.input, outputs=output_layer)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet/mobilenet_1_0_224_tf_no_top.h5
17225924/17225924 [==============================] - 0s 0us/step
```

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

```
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 conv1 (Conv2D)              (None, 112, 112, 32)      864

 conv1_bn (BatchNormalizati  (None, 112, 112, 32)      128
 on)

 conv1_relu (ReLU)           (None, 112, 112, 32)      0

 conv_dw_1 (DepthwiseConv2D  (None, 112, 112, 32)      288
 )

 conv_dw_1_bn (BatchNormali  (None, 112, 112, 32)      128
 zation)

 conv_dw_1_relu (ReLU)       (None, 112, 112, 32)      0

 conv_pw_1 (Conv2D)          (None, 112, 112, 64)      2048

 conv_pw_1_bn (BatchNormali  (None, 112, 112, 64)      256
 zation)

 conv_pw_1_relu (ReLU)       (None, 112, 112, 64)      0

 conv_pad_2 (ZeroPadding2D)  (None, 113, 113, 64)      0
```

```
conv_dw_2 (DepthwiseConv2D  (None, 56, 56, 64)        576
)

conv_dw_2_bn (BatchNormali  (None, 56, 56, 64)        256
zation)

conv_dw_2_relu (ReLU)       (None, 56, 56, 64)        0

conv_pw_2 (Conv2D)          (None, 56, 56, 128)       8192

conv_pw_2_bn (BatchNormali  (None, 56, 56, 128)       512
zation)

conv_pw_2_relu (ReLU)       (None, 56, 56, 128)       0

conv_dw_3 (DepthwiseConv2D  (None, 56, 56, 128)       1152
)

conv_dw_3_bn (BatchNormali  (None, 56, 56, 128)       512
zation)

conv_dw_3_relu (ReLU)       (None, 56, 56, 128)       0

conv_pw_3 (Conv2D)          (None, 56, 56, 128)       16384

conv_pw_3_bn (BatchNormali  (None, 56, 56, 128)       512
zation)
```

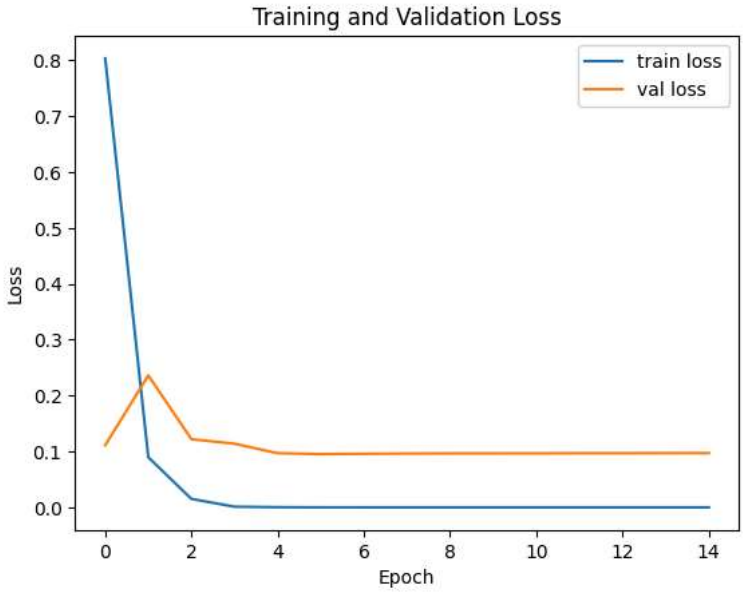
```
mnet= model.fit(x_train, y_train, epochs=15, validation_data=(x_test, y_test))
```

```
Epoch 1/15
52/52 [==============================] - 11s 133ms/step - loss: 0.8035 - accuracy: 0.8729 - val_loss: 0.1111 - val_accuracy: 0.9709
Epoch 2/15
52/52 [==============================] - 3s 50ms/step - loss: 0.0896 - accuracy: 0.9752 - val_loss: 0.2361 - val_accuracy: 0.9613
Epoch 3/15
52/52 [==============================] - 2s 47ms/step - loss: 0.0151 - accuracy: 0.9958 - val_loss: 0.1219 - val_accuracy: 0.9685
Epoch 4/15
52/52 [==============================] - 2s 47ms/step - loss: 0.0012 - accuracy: 1.0000 - val_loss: 0.1139 - val_accuracy: 0.9685
Epoch 5/15
52/52 [==============================] - 3s 50ms/step - loss: 2.8877e-04 - accuracy: 1.0000 - val_loss: 0.0969 - val_accuracy: 0.9879
Epoch 6/15
52/52 [==============================] - 3s 49ms/step - loss: 4.3402e-05 - accuracy: 1.0000 - val_loss: 0.0953 - val_accuracy: 0.9879
Epoch 7/15
52/52 [==============================] - 3s 51ms/step - loss: 2.4463e-05 - accuracy: 1.0000 - val_loss: 0.0958 - val_accuracy: 0.9879
Epoch 8/15
52/52 [==============================] - 2s 47ms/step - loss: 2.0191e-05 - accuracy: 1.0000 - val_loss: 0.0963 - val_accuracy: 0.9879
Epoch 9/15
52/52 [==============================] - 3s 51ms/step - loss: 1.8158e-05 - accuracy: 1.0000 - val_loss: 0.0966 - val_accuracy: 0.9879
Epoch 10/15
52/52 [==============================] - 3s 52ms/step - loss: 1.6073e-05 - accuracy: 1.0000 - val_loss: 0.0966 - val_accuracy: 0.9879
Epoch 11/15
52/52 [==============================] - 3s 48ms/step - loss: 1.4613e-05 - accuracy: 1.0000 - val_loss: 0.0967 - val_accuracy: 0.9879
Epoch 12/15
52/52 [==============================] - 2s 48ms/step - loss: 1.3329e-05 - accuracy: 1.0000 - val_loss: 0.0969 - val_accuracy: 0.9879
Epoch 13/15
52/52 [==============================] - 2s 48ms/step - loss: 1.2344e-05 - accuracy: 1.0000 - val_loss: 0.0969 - val_accuracy: 0.9879
Epoch 14/15
52/52 [==============================] - 3s 51ms/step - loss: 1.1577e-05 - accuracy: 1.0000 - val_loss: 0.0970 - val_accuracy: 0.9879
Epoch 15/15
52/52 [==============================] - 3s 52ms/step - loss: 1.0576e-05 - accuracy: 1.0000 - val_loss: 0.0971 - val_accuracy: 0.9879
```

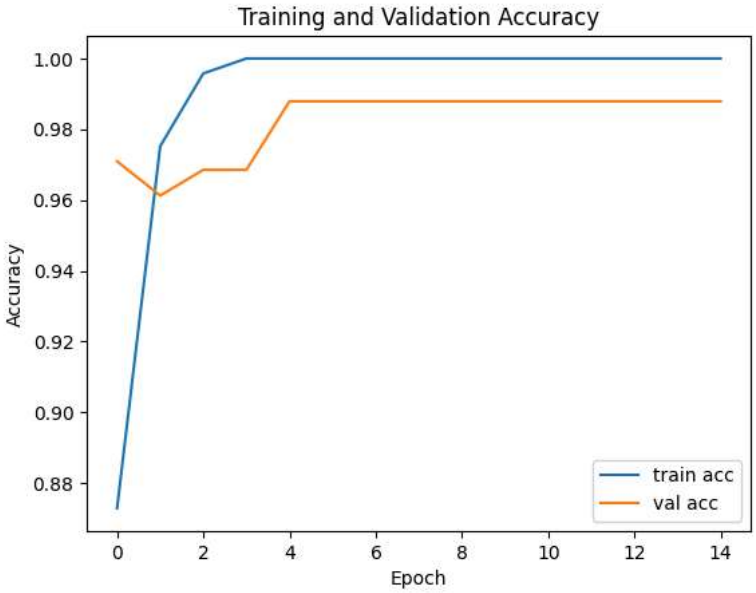
```
#Evaluate the model on your test data.
test_loss, test_accuracy = model.evaluate(x_test, y_test)
print(f"Test accuracy: {test_accuracy*100:.2f}%")
```

```
13/13 [==============================] - 1s 41ms/step - loss: 0.0971 - accuracy: 0.9879
Test accuracy: 98.79%
```

```
# Plot the loss
import matplotlib.pyplot as plt
plt.plot(mnet.history['loss'], label='train loss')
plt.plot(mnet.history['val_loss'], label='val loss')
plt.legend()
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training and Validation Loss')
plt.savefig('LossVal_loss.png')
plt.show()
```



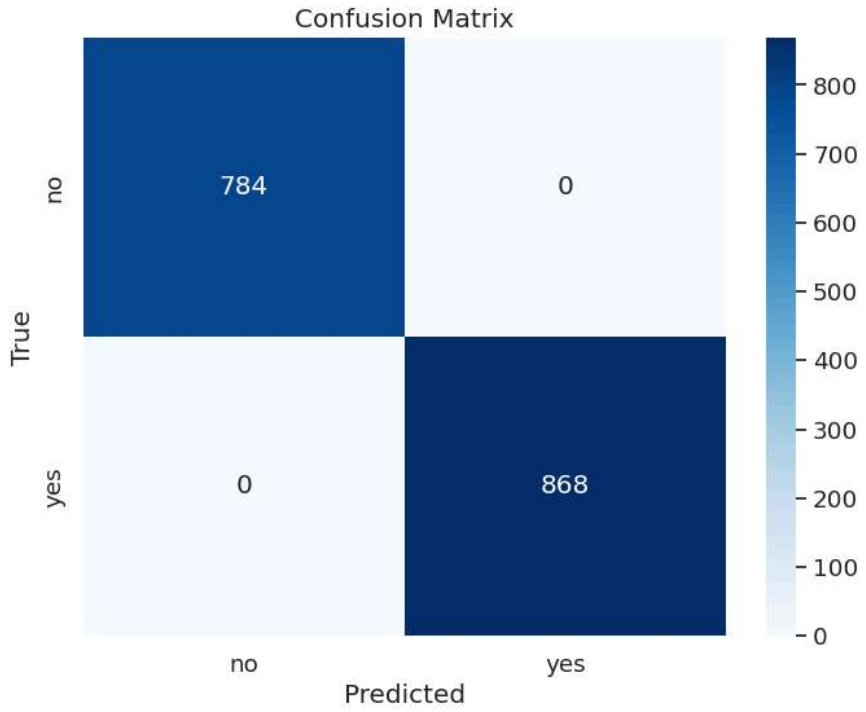

```
# Plot the accuracy
plt.plot(mnet.history['accuracy'], label='train acc')
plt.plot(mnet.history['val_accuracy'], label='val acc')
plt.legend()
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Training and Validation Accuracy')
plt.savefig('AccVal_acc.png')
plt.show()
```

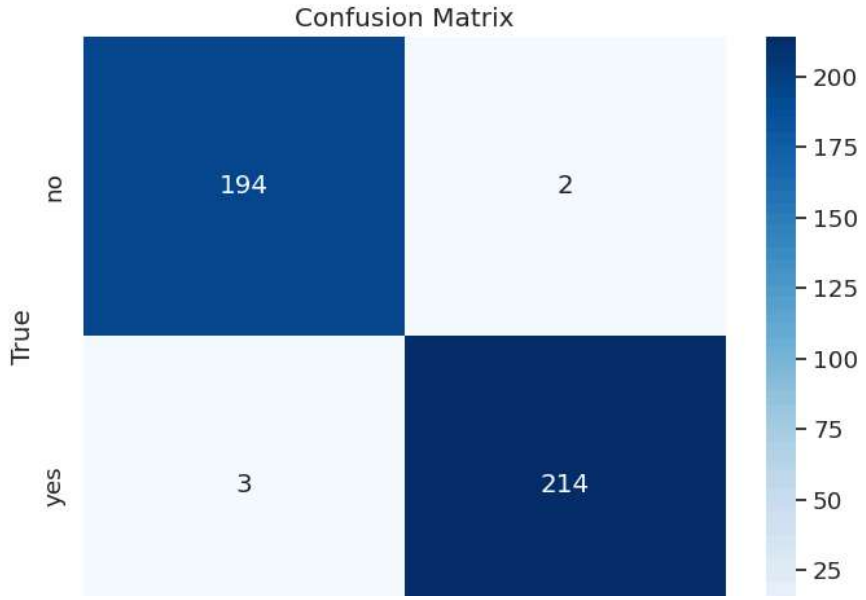


```
import numpy as np
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix

# Assuming you have already trained your model and obtained predictions for both train and test sets
train_predictions = model.predict(x_train)  # Replace 'model' with your trained model
test_predictions = model.predict(x_test)

# Convert predictions to class labels
train_pred_labels = np.argmax(train_predictions, axis=1)
test_pred_labels = np.argmax(test_predictions, axis=1)

# True labels
train_true_labels = np.argmax(y_train, axis=1)
test_true_labels = np.argmax(y_test, axis=1)

# Calculate metrics for the training dataset
train_accuracy = accuracy_score(train_true_labels, train_pred_labels)
train_precision = precision_score(train_true_labels, train_pred_labels, average='weighted')
train_recall = recall_score(train_true_labels, train_pred_labels, average='weighted')
train_f1_score = f1_score(train_true_labels, train_pred_labels, average='weighted')
train_confusion = confusion_matrix(train_true_labels, train_pred_labels)

# Calculate metrics for the testing dataset
test_accuracy = accuracy_score(test_true_labels, test_pred_labels)
test_precision = precision_score(test_true_labels, test_pred_labels, average='weighted')
test_recall = recall_score(test_true_labels, test_pred_labels, average='weighted')
test_f1_score = f1_score(test_true_labels, test_pred_labels, average='weighted')
test_confusion = confusion_matrix(test_true_labels, test_pred_labels)

# Print the metrics
print("Training Metrics:")
print(f"Accuracy: {train_accuracy:.4f}")
print(f"Precision: {train_precision:.4f}")
print(f"Recall: {train_recall:.4f}")
print(f"F1-Score: {train_f1_score:.4f}")
print("Confusion Matrix:")
print(train_confusion)

print("\nTesting Metrics:")
print(f"Accuracy: {test_accuracy:.4f}")
print(f"Precision: {test_precision:.4f}")
print(f"Recall: {test_recall:.4f}")
print(f"F1-Score: {test_f1_score:.4f}")
print("Confusion Matrix:")
print(test_confusion)
```

```
52/52 [==============================] - 3s 39ms/step
13/13 [==============================] - 0s 33ms/step
Training Metrics:
Accuracy: 1.0000
Precision: 1.0000
Recall: 1.0000
F1-Score: 1.0000
Confusion Matrix:
[[784   0]
 [  0 868]]

Testing Metrics:
Accuracy: 0.9879
Precision: 0.9879
Recall: 0.9879
F1-Score: 0.9879
Confusion Matrix:
[[194   2]
 [  3 214]]
```

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix

# ... (previous code for predictions and metrics calculation) ...

# Define class names
class_names = ['no','yes'] # Replace with your actual class names

# Create confusion matrix for training dataset
train_confusion = confusion_matrix(train_true_labels, train_pred_labels)

# Create confusion matrix for testing dataset
test_confusion = confusion_matrix(test_true_labels, test_pred_labels)

# Function to plot confusion matrix
def plot_confusion_matrix(cm, labels):
    plt.figure(figsize=(8, 6))
    sns.set(font_scale=1.2)
    sns.heatmap(
        cm,
        annot=True,
        cmap='Blues',  # You can choose other colormaps like 'viridis', 'coolwarm', etc.
        xticklabels=labels,
        yticklabels=labels,
        fmt='g'
    )
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.title('Confusion Matrix')
    plt.show()

# Plot confusion matrix for training dataset
plot_confusion_matrix(train_confusion, class_names)

# Plot confusion matrix for testing dataset
plot_confusion_matrix(test_confusion, class_names)
```





SAVE MODEL

```
from tensorflow.keras.models import load_model
model.save('model_mobilenet.h5')
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `m
  saving_api.save_model(
```

```
import tensorflow as tf
from tensorflow.keras.preprocessing import image
import numpy as np
import matplotlib.pyplot as plt
```