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

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

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNet
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.models import Model

# Define the parameters
img_width, img_height = 224, 224
batch_size = 32
num_classes = 4
```

```
# Define the path to your dataset
train_data_dir = '/content/drive/MyDrive/ECG_Images/train'
val_data_dir = '/content/drive/MyDrive/ECG_Images/val'
```

```
# Create an ImageDataGenerator for training and validation
train_datagen = ImageDataGenerator(rescale=1./255)
val_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical')

val_generator = val_datagen.flow_from_directory(
    val_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical')
```

```
Found 648 images belonging to 4 classes.
Found 183 images belonging to 4 classes.
```

```
# Load MobileNet model without the top (fully connected) layers
base_model = MobileNet(weights='imagenet', include_top=False)

# Add custom top layers
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
predictions = Dense(num_classes, activation='softmax')(x)

# Combine the base model with custom top layers
model = Model(inputs=base_model.input, outputs=predictions)

for layer in model.layers[:80]:
    layer.trainable = False
```

```
WARNING:tensorflow:`input_shape` is undefined or non-square, or `rows` is not in [128, 160, 192, 224]. Weights for input
```

```
model.summary()
```

```
 conv_pw_11_bn (BatchNormal  (None, None, None, 512)   2048
 ization)

 conv_pw_11_relu (ReLU)      (None, None, None, 512)   0

 conv_pad_12 (ZeroPadding2D  (None, None, None, 512)   0
 )

 conv_dw_12 (DepthwiseConv2  (None, None, None, 512)   4608
 D)

 conv_dw_12_bn (BatchNormal  (None, None, None, 512)   2048
 ization)

 conv_dw_12_relu (ReLU)      (None, None, None, 512)   0

 conv_pw_12 (Conv2D)         (None, None, None, 1024   524288
                             )

 conv_pw_12_bn (BatchNormal  (None, None, None, 1024   4096
 ization)                    )

 conv_pw_12_relu (ReLU)      (None, None, None, 1024   0
                             )

 conv_dw_13 (DepthwiseConv2  (None, None, None, 1024   9216
 D)                          )

 conv_dw_13_bn (BatchNormal  (None, None, None, 1024   4096
 ization)                    )

 conv_dw_13_relu (ReLU)      (None, None, None, 1024   0
                             )

 conv_pw_13 (Conv2D)         (None, None, None, 1024   1048576
                             )
```

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

```
# Train the model
model.fit(train_generator,
          steps_per_epoch=train_generator.samples // batch_size,
          epochs=19,
          validation_data=val_generator,
          validation_steps=val_generator.samples // batch_size)
```

```
Epoch 1/19
20/20 [==============================] - 20s 1s/step - loss: 0.1203 - accuracy: 0.9545 - val_loss: 3.3274 - val_accuracy
Epoch 2/19
20/20 [==============================] - 21s 1s/step - loss: 0.1089 - accuracy: 0.9562 - val_loss: 2.7674 - val_accuracy
Epoch 3/19
20/20 [==============================] - 21s 1s/step - loss: 0.0501 - accuracy: 0.9821 - val_loss: 3.4933 - val_accuracy
Epoch 4/19
20/20 [==============================] - 20s 1s/step - loss: 0.0363 - accuracy: 0.9838 - val_loss: 0.4977 - val_accuracy
Epoch 5/19
20/20 [==============================] - 21s 1s/step - loss: 0.0088 - accuracy: 0.9968 - val_loss: 2.4260 - val_accuracy
Epoch 6/19
20/20 [==============================] - 20s 1s/step - loss: 0.0067 - accuracy: 0.9968 - val_loss: 0.4704 - val_accuracy
Epoch 7/19
20/20 [==============================] - 22s 1s/step - loss: 0.0071 - accuracy: 0.9968 - val_loss: 0.6100 - val_accuracy
Epoch 8/19
20/20 [==============================] - 21s 1s/step - loss: 0.0536 - accuracy: 0.9773 - val_loss: 0.5148 - val_accuracy
Epoch 9/19
20/20 [==============================] - 25s 1s/step - loss: 0.0234 - accuracy: 0.9935 - val_loss: 0.2216 - val_accuracy
Epoch 10/19
20/20 [==============================] - 26s 1s/step - loss: 0.0061 - accuracy: 0.9984 - val_loss: 0.1384 - val_accuracy
Epoch 11/19
20/20 [==============================] - 21s 1s/step - loss: 0.0021 - accuracy: 1.0000 - val_loss: 0.1120 - val_accuracy
Epoch 12/19
20/20 [==============================] - 22s 1s/step - loss: 0.0014 - accuracy: 1.0000 - val_loss: 0.0842 - val_accuracy
Epoch 13/19
20/20 [==============================] - 21s 1s/step - loss: 3.9734e-04 - accuracy: 1.0000 - val_loss: 0.0944 - val_accu
Epoch 14/19
20/20 [==============================] - 19s 998ms/step - loss: 8.6477e-04 - accuracy: 1.0000 - val_loss: 0.1294 - val_a
Epoch 15/19
20/20 [==============================] - 22s 1s/step - loss: 3.9108e-04 - accuracy: 1.0000 - val_loss: 0.0912 - val_accu
Epoch 16/19
20/20 [==============================] - 20s 1s/step - loss: 3.6688e-04 - accuracy: 1.0000 - val_loss: 0.1044 - val_accu
Epoch 17/19
20/20 [==============================] - 22s 1s/step - loss: 2.3560e-04 - accuracy: 1.0000 - val_loss: 0.0632 - val_accu
Epoch 18/19
20/20 [==============================] - 26s 1s/step - loss: 3.0488e-04 - accuracy: 1.0000 - val_loss: 0.0908 - val_accu
Epoch 19/19
20/20 [==============================] - 21s 1s/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.1121 - val_accuracy
<keras.src.callbacks.History at 0x7ca183db5750>
```

```
# Define the path to the test dataset
test_data_dir = '/content/drive/MyDrive/ECG_Images/test'

# Create an ImageDataGenerator for the test set
test_datagen = ImageDataGenerator(rescale=1./255)

test_generator = test_datagen.flow_from_directory(
    test_data_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    shuffle=False)  # Important: Do not shuffle for proper evaluation

# Evaluate the model on the test set
loss, accuracy = model.evaluate(test_generator)

print("Test Loss:", loss)
print("Test Accuracy:", accuracy)
```

```
Found 97 images belonging to 4 classes.
4/4 [==============================] - 2s 483ms/step - loss: 0.1333 - accuracy: 0.9691
Test Loss: 0.13326138257980347
Test Accuracy: 0.969072163105011
```

```
# Save the model
model.save("/content/drive/MyDrive/MobileNet_Tune.h5")
```

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

```
from sklearn.metrics import classification_report

# Assuming you have trained your model and stored it in the variable 'model'
# Evaluate the model on the validation data
test_loss, test_accuracy = model.evaluate(test_generator)

# Get the predictions for the validation data
test_predictions = model.predict(test_generator)
# Assuming val_predictions is in one-hot encoded format, convert it to class labels
test_pred_labels = np.argmax(test_predictions, axis=1)

# Get the true labels for the validation data
test_true_labels = test_generator.classes

# Generate the classification report
class_names = list(test_generator.class_indices.keys())
report = classification_report(test_true_labels, test_pred_labels, target_names=class_names)

print(report)

```

```
4/4 [==============================] - 3s 513ms/step - loss: 0.1333 - accuracy: 0.9691
4/4 [==============================] - 2s 488ms/step
                                                                 precision    recall  f1-score   support

      ECG Images of Myocardial Infarction Patients (240x12=2880)       1.00      1.00      1.00        25
     ECG Images of Patient that have History of MI (172x12=2064)       0.94      0.89      0.91        18
ECG Images of Patient that have abnormal heartbeat (233x12=2796)       1.00      0.96      0.98        24
                           Normal Person ECG Images (284x12=3408)       0.94      1.00      0.97        30

                                                        accuracy                           0.97        97
                                                       macro avg       0.97      0.96      0.97        97
                                                    weighted avg       0.97      0.97      0.97        97
```

```
import numpy as np
from tensorflow.keras.preprocessing import image

# Load the trained model
model_path = "/content/drive/MyDrive/MobileNet_Tune.h5"
model = tf.keras.models.load_model(model_path)

# Load and preprocess the single image
img_path = "/content/drive/MyDrive/data/ECG Images of Myocardial Infarction Patients/MI (16).jpg"  # Provide the path to
img = image.load_img(img_path, target_size=(224, 224))
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array = img_array / 255.  # Normalize the image

```

```
# Make predictions
predictions = model.predict(img_array)

# Interpret the predictions
class_names = ["MI", "PMI", "Abnormal", "Normal"]  # Define your class names
predicted_class = np.argmax(predictions)

print("Predicted class:", class_names[predicted_class])
print("Confidence:", predictions[0][predicted_class])
```

```
1/1 [==============================] - 0s 416ms/step
Predicted class: PMI
Confidence: 0.99984336
```

```
Start coding or generate with AI.
```