```
import tensorflow as ts
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.preprocessing.image import ImageDataGenerator

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).
```

```
IMG_SIZE=224
BATCH_SIZE=32

train_datagen=ImageDataGenerator(rescale=1.225,validation_split=0.2)

train_generator=train_datagen.flow_from_directory(target_size=(IMG_SIZ
E,IMG_SIZE),batch_size=BATCH_SIZE,class_mode='binary',subset='training
',directory='/content/drive/MyDrive/findleaf/leaf')
```

```
Found 345 images belonging to 2 classes.
```

```
val_generator=train_datagen.flow_from_directory(target_size=(IMG_SIZE,
IMG_SIZE),batch_size=BATCH_SIZE,class_mode='binary',subset='validation
',directory='/content/drive/MyDrive/findleaf/leaf')
```

```
Found 85 images belonging to 2 classes.
```

```
model=keras.Sequential([

layers.Conv2D(32,kernel_size=(3,3),activation='relu',input_shape=(IMG_
SIZE,IMG_SIZE,3)),
    layers.MaxPooling2D(pool_size=(2,2)),
    layers.Conv2D(64,kernel_size=(3,3),activation='relu'),
    layers.MaxPooling2D(pool_size=(2,2)),
    layers.Conv2D(128,kernel_size=(3,3),activation='relu'),
    layers.MaxPooling2D(pool_size=(2,2)),
    layers.Flatten(),
    layers.Dense(128,activation='relu'),
    layers.Dense(1,activation='sigmoid')
])
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
```

```
model.summary()
```

```
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃         Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ conv2d (Conv2D)                      │ (None, 222, 222, 32)        │             896 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d (MaxPooling2D)         │ (None, 111, 111, 32)        │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_1 (Conv2D)                    │ (None, 109, 109, 64)        │          18,496 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d_1 (MaxPooling2D)       │ (None, 54, 54, 64)          │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_2 (Conv2D)                    │ (None, 52, 52, 128)         │          73,856 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d_2 (MaxPooling2D)       │ (None, 26, 26, 128)         │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ flatten (Flatten)                    │ (None, 86528)               │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense (Dense)                        │ (None, 128)                 │      11,075,712 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_1 (Dense)                      │ (None, 1)                   │             129 │
└──────────────────────────────────────┴─────────────────────────────┴─────────────────┘

 Total params: 11,169,089 (42.61 MB)

 Trainable params: 11,169,089 (42.61 MB)

 Non-trainable params: 0 (0.00 B)
```

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

model.fit(train_generator,epochs=3,validation_data=val_generator,batch_size=BATCH_SIZE)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()

Epoch 1/3
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 11s/step - accuracy: 0.6277 - loss: 720.5594

/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()

 11/11 ━━━━━━━━━━━━━━━━━━━━ 164s 15s/step - accuracy: 0.6295 - loss: 694.2172 - val_accuracy: 0.8471 - val_loss: 25.8115
Epoch 2/3
11/11 ━━━━━━━━━━━━━━━━━━━━ 44s 4s/step - accuracy: 0.8027 - loss: 19.2492 - val_accuracy: 0.9765 - val_loss: 0.3079
Epoch 3/3
11/11 ━━━━━━━━━━━━━━━━━━━━ 81s 4s/step - accuracy: 0.9883 - loss: 0.1317 - val_accuracy: 0.9882 - val_loss: 0.2830

<keras.src.callbacks.history.History at 0x7fdd71164990>
```

```
model.save('/content/drive/MyDrive/findleaf/leaf.h5')
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
```

```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import matplotlib.pyplot as plt
import numpy as np
model=load_model('/content/drive/MyDrive/findleaf/leaf.h5')
print("Model Loaded Successfully")
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.

Model Loaded Successfully
```

```
test_image_path='/content/drive/MyDrive/findleaf/leaf/Healthy/Healthy (183).jpg'
img=image.load_img(test_image_path,target_size=(224,224))
plt.imshow(img)
plt.axis('off')
plt.show()
```

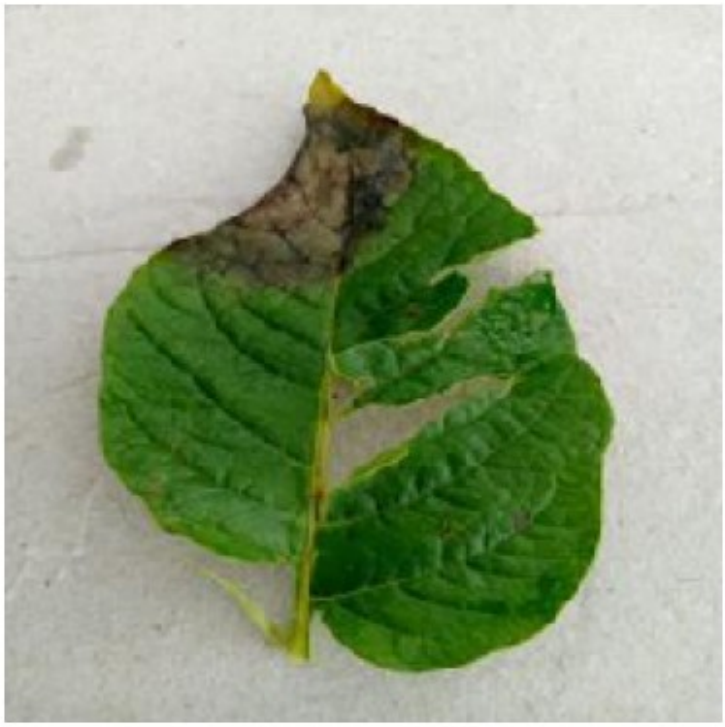
```
immg_array=image.img_to_array(img)
immg_array=np.expand_dims(immg_array,axis=0)
immg_array=immg_array/255.0
prediction=model.predict(immg_array)
print(prediction)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 143ms/step
[[0.45673764]]
```

```
if prediction>=0.5:
  print("Leaf is Diseased")
else:
  print("Leaf is Healthy")
```

```
Leaf is Healthy
```

```
test_image_path='/content/drive/MyDrive/findleaf/leaf/Late
Blight/Late_Blight (458).jpg'
img=image.load_img(test_image_path,target_size=(224,224))
plt.imshow(img)
plt.axis('off')
plt.show()
```



```
immg_array=image.img_to_array(img)
immg_array=np.expand_dims(immg_array,axis=0)
immg_array=immg_array/255.0
prediction=model.predict(immg_array)
print(prediction)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 72ms/step
[[0.5217044]]
```

```
if prediction>=0.5:
  print("Leaf is Diseased")
else:
  print("Leaf is Healthy")
```

```
Leaf is Diseased
```