```
import os
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
```

```
# Set random seed for reproducibility
np.random.seed(42)
tf.random.set_seed(42)
```

```
# Define paths to your dataset
train_path = '/content/drive/MyDrive/Code alpha/tomato/train'
test_path = '/content/drive/MyDrive/Code alpha/tomato/val'

# Define image dimensions and batch size
img_width, img_height = 128, 128
batch_size = 32
```

```
# Create data generators with data augmentation for training set
train_datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True
)
```

```
train_generator = train_datagen.flow_from_directory(
    train_path,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='binary'
)
```

```
Found 950 images belonging to 10 classes.
```

```
# Create data generator for test set
test_datagen = ImageDataGenerator(rescale=1./255)

test_generator = test_datagen.flow_from_directory(
    test_path,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='binary'
)
```

```
Found 1010 images belonging to 10 classes.
```

```
# Build a simple CNN model
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(img_width, img_height, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(units=128, activation='relu'))
model.add(Dense(units=1, activation='sigmoid'))
```

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

# Train the model
model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // batch_size,
    epochs=5,
    validation_data=test_generator,
    validation_steps=test_generator.samples // batch_size
)
```

```
Epoch 1/5
29/29 [==============================] - 342s 12s/step - loss: -12589.9775 - accuracy: 0.0000e+00 - val_loss: -28521.3359 - val_accu
Epoch 2/5
29/29 [==============================] - 13s 442ms/step - loss: -151708.7188 - accuracy: 0.0000e+00 - val_loss: -216782.8125 - val_a
Epoch 3/5
29/29 [==============================] - 11s 396ms/step - loss: -712838.7500 - accuracy: 0.0000e+00 - val_loss: -822441.6875 - val_a
Epoch 4/5
29/29 [==============================] - 13s 445ms/step - loss: -2164749.7500 - accuracy: 0.0000e+00 - val_loss: -2174265.0000 - val
Epoch 5/5
```

```
29/29 [==============================] - 11s 390ms/step - loss: -5022241.0000 - accuracy: 0.0000e+00 - val_loss: -4544939.0000 - val
<keras.src.callbacks.History at 0x7ba574657640>
```

```
# Save the model for later use
model.save('plant_disease_detection_model.h5')
```

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

Double-click (or enter) to edit

```
# Make predictions on new images
def predict_image(image_path):
    img = tf.keras.preprocessing.image.load_img(image_path, target_size=(img_width, img_height))
    img_array = tf.keras.preprocessing.image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array /= 255.0
    prediction = model.predict(img_array)
    if prediction[0] < 0.5:
        print("Healthy")
    else:
        print("Infected")
```

```
# Example usage
test_image_path = '/content/drive/MyDrive/Code alpha/tomato/val/Tomato___Bacterial_spot/01a3cf3f-94c1-44d5-8972-8c509d62558e___GCREC_Bact
predict_image(test_image_path)
```

```
1/1 [==============================] - 0s 162ms/step
Infected
```