```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization, Activation
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
import os
import sys

# --- 0. Setup and Unpack Data ---
# This cell handles unpacking your .rar file.
# IMPORTANT: Make sure your RAR file is named 'PlantDoc.rar' and is uploaded
# to the Colab session's file storage before you run this script.

RAR_FILE_NAME = 'Plantdoc.rar'

# Check if the file has been uploaded
if not os.path.exists(RAR_FILE_NAME):
    print(f"--- ERROR ---")
    print(f"File '{RAR_FILE_NAME}' not found.")
    print(f"Please upload your .rar file to the Colab session files before running.")
    # Stop the script if the file isn't there.
    sys.exit()

# Install the 'unrar' utility (runs a shell command in Colab)
print("Installing unrar...")
get_ipython().system('apt-get install -y -qq unrar')
print("Unrar utility installed.")

# Unpack the RAR file into the Colab environment.
# The '-o+' flag ensures it overwrites any existing files if you run it again.
print(f"Unpacking '{RAR_FILE_NAME}'...")
get_ipython().system('unrar x -o+ "{RAR_FILE_NAME}"')
print("Unpacking complete.")


# --- Configuration ---
# The directories are now in the local Colab environment, not Google Drive
TRAIN_DIR = '/content/Plantdoc/train'
VALID_DIR = '/content/Plantdoc/valid'

# Define image and batch parameters
IMG_HEIGHT = 224
IMG_WIDTH = 224
BATCH_SIZE = 32
EPOCHS = 20 # You can increase this for better accuracy, e.g., 50 or 100

# --- 1. Data Preparation and Augmentation ---

# Create an ImageDataGenerator for the training set with data augmentation
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

# Create an ImageDataGenerator for the validation set (only rescaling)
valid_datagen = ImageDataGenerator(rescale=1./255)

# Create data generators that read images from the directories
train_generator = train_datagen.flow_from_directory(
    TRAIN_DIR,
    target_size=(IMG_HEIGHT, IMG_WIDTH),
```

```
    batch_size=BATCH_SIZE,
    class_mode='categorical'
)

validation_generator = valid_datagen.flow_from_directory(
    VALID_DIR,
    target_size=(IMG_HEIGHT, IMG_WIDTH),
    batch_size=BATCH_SIZE,
    class_mode='categorical'
)

# Get the number of classes from the generator
num_classes = len(train_generator.class_indices)
print(f"Found {train_generator.samples} images belonging to {num_classes} classes in the training set.")
print(f"Found {validation_generator.samples} images belonging to {num_classes} classes in the validation set.")


# --- 2. Build the Deep CNN Model ---

model = Sequential([
    # Input Layer and First Convolutional Block
    Conv2D(32, (3, 3), padding='same', input_shape=(IMG_HEIGHT, IMG_WIDTH, 3)),
    BatchNormalization(),
    Activation('relu'),
    MaxPooling2D(pool_size=(2, 2)),

    # Second Convolutional Block
    Conv2D(64, (3, 3), padding='same'),
    BatchNormalization(),
    Activation('relu'),
    MaxPooling2D(pool_size=(2, 2)),

    # Third Convolutional Block
    Conv2D(128, (3, 3), padding='same'),
    BatchNormalization(),
    Activation('relu'),
    Conv2D(128, (3, 3), padding='same'),
    BatchNormalization(),
    Activation('relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Dropout(0.25),

    # Fourth Convolutional Block
    Conv2D(256, (3, 3), padding='same'),
    BatchNormalization(),
    Activation('relu'),
    Conv2D(256, (3, 3), padding='same'),
    BatchNormalization(),
    Activation('relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Dropout(0.25),

    # Classifier Head
    Flatten(),
    Dense(512),
    BatchNormalization(),
    Activation('relu'),
    Dropout(0.5),
    Dense(num_classes, activation='softmax')
])

# Print a summary of the model architecture
model.summary()


# --- 3. Compile the Model ---

model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
```

```
    loss='categorical_crossentropy',
    metrics=['accuracy']
)


# --- 4. Train the Model ---

history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // BATCH_SIZE,
    epochs=EPOCHS,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // BATCH_SIZE
)


# --- 5. Save the Trained Model ---
# The model will be saved to the Colab session storage.
model.save('apple_disease_classifier_model.h5')
print(f"\nModel saved successfully as 'apple_disease_classifier_model.h5' in the session storage.")
print("IMPORTANT: You must download this file from the Colab 'Files' tab before the session ends.")


# --- 6. Evaluate and Visualize Training ---

# The plot will also be saved to the session storage.
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')

plt.tight_layout()
plt.savefig('training_history.png')
plt.show()

print(f"\nTraining complete. Plot saved as 'training_history.png'.")
print("IMPORTANT: Remember to download the plot and the model file!")
```