```
! wget https://data.mendeley.com/public-files/datasets/tywbtsjrjv/files/b4e3a32f-c0bd-4060-81e9-6144231f2520/file_downloaded -O plant_disease_datase
! unzip plant_disease_dataset.zip


! pwd
! ls
! rm -rf Plant_leave_diseases_dataset_with_augmentation/B*
! rm -rf Plant_leave_diseases_dataset_with_augmentation/C*
! rm -rf Plant_leave_diseases_dataset_with_augmentation/D*
! rm -rf Plant_leave_diseases_dataset_with_augmentation/E*
! rm -rf Plant_leave_diseases_dataset_with_augmentation/F*
! rm -rf Plant_leave_diseases_dataset_with_augmentation/G*
! rm -rf Plant_leave_diseases_dataset_with_augmentation/H*
! rm -rf Plant_leave_diseases_dataset_with_augmentation/I*
! rm -rf Plant_leave_diseases_dataset_with_augmentation/J*
! rm -rf Plant_leave_diseases_dataset_with_augmentation/K*
! rm -rf Plant_leave_diseases_dataset_with_augmentation/L*
! rm -rf Plant_leave_diseases_dataset_with_augmentation/M*
! rm -rf Plant_leave_diseases_dataset_with_augmentation/N*
! rm -rf Plant_leave_diseases_dataset_with_augmentation/O*
! rm -rf Plant_leave_diseases_dataset_with_augmentation/P*
! rm -rf Plant_leave_diseases_dataset_with_augmentation/Q*
! rm -rf Plant_leave_diseases_dataset_with_augmentation/R*
! rm -rf Plant_leave_diseases_dataset_with_augmentation/S*
! rm -rf Plant_leave_diseases_dataset_with_augmentation/T*
! rm -rf Plant_leave_diseases_dataset_with_augmentation/U*
! rm -rf Plant_leave_diseases_dataset_with_augmentation/V*
! rm -rf Plant_leave_diseases_dataset_with_augmentation/W*
! rm -rf Plant_leave_diseases_dataset_with_augmentation/X*
! rm -rf Plant_leave_diseases_dataset_with_augmentation/Y*
! rm -rf Plant_leave_diseases_dataset_with_augmentation/Z*
! pwd
! ls Plant_leave_diseases_dataset_with_augmentation/
```

```
/content
plant_disease_dataset.zip  Plant_leave_diseases_dataset_with_augmentation  sample_data
/content
Apple___Apple_scab  Apple___Black_rot  Apple___Cedar_apple_rust  Apple___healthy
```

```
# Importing required libraries
import os
import numpy as np
import matplotlib.pyplot as plt
import cv2
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout


# Path to the dataset folder
data_dir = '/content/Plant_leave_diseases_dataset_with_augmentation'

# Creating a list of all the images and labels
images = []
labels = []
disease_types = os.listdir(data_dir)

for disease_type in disease_types:
    label = disease_types.index(disease_type)
    disease_folder_path = os.path.join(data_dir, disease_type)
    for img_path in os.listdir(disease_folder_path):
        img = cv2.imread(os.path.join(disease_folder_path, img_path))
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        img = cv2.resize(img, (224, 224))
        images.append(img)
        labels.append(label)

# Converting the lists into numpy arrays
images = np.array(images)
labels = np.array(labels)

# Splitting the dataset into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(images, labels, test_size=0.2, stratify=labels, random_state=42)

# Normalizing the pixel values of the images
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255

# Converting the labels into one-hot encoded vectors
y_train = to_categorical(y_train, num_classes=len(disease_types))
y_test = to_categorical(y_test, num_classes=len(disease_types))
```

```
# Defining the CNN model
model = Sequential([
  Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)),
  MaxPooling2D((2, 2)),

  Conv2D(64, (3, 3), activation='relu'),
  MaxPooling2D((2, 2)),

  Conv2D(128, (3, 3), activation='relu'),
  MaxPooling2D((2, 2)),

  Conv2D(128, (3, 3), activation='relu'),
  MaxPooling2D((2, 2)),

  Flatten(),

  Dense(512, activation='relu'),
  Dropout(0.5),

  Dense(len(disease_types), activation='softmax')
])


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

# Training the model
history = model.fit(x_train, y_train, epochs=25, batch_size=32, validation_data=(x_test, y_test))

# Evaluating the model on the test set
test_loss, test_acc = model.evaluate(x_test, y_test)
print('Test accuracy:', test_acc)

# Plotting the accuracy and loss curves
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.plot(history.history['loss'], label='loss')
plt.plot(history.history['val_loss'], label='val_loss')
plt.xlabel('Epoch')
plt.ylabel('Accuracy/Loss')
plt.legend()
plt.show()
```

```
Epoch 1/25
117/117 [==============================] - 503s 4s/step - loss: 0.7713 - accuracy: 0.6876 - val_loss: 0.5423 - val_accuracy: 0.7750
Epoch 2/25
```

```
117/117 [==============================] - 475s 4s/step - loss: 0.4126 - accuracy: 0.8402 - val_loss: 0.2944 - val_accuracy: 0.8708
Epoch 3/25
117/117 [==============================] - 471s 4s/step - loss: 0.2806 - accuracy: 0.8991 - val_loss: 0.2466 - val_accuracy: 0.9139
Epoch 4/25
117/117 [==============================] - 471s 4s/step - loss: 0.1960 - accuracy: 0.9271 - val_loss: 0.3689 - val_accuracy: 0.8590
Epoch 5/25
117/117 [==============================] - 464s 4s/step - loss: 0.1863 - accuracy: 0.9314 - val_loss: 0.1173 - val_accuracy: 0.9516
Epoch 6/25
117/117 [==============================] - 466s 4s/step - loss: 0.1369 - accuracy: 0.9532 - val_loss: 0.1397 - val_accuracy: 0.9483
Epoch 7/25
117/117 [==============================] - 458s 4s/step - loss: 0.1128 - accuracy: 0.9578 - val_loss: 0.0940 - val_accuracy: 0.9602
Epoch 8/25
117/117 [==============================] - 464s 4s/step - loss: 0.0881 - accuracy: 0.9682 - val_loss: 0.1433 - val_accuracy: 0.9462
Epoch 9/25
117/117 [==============================] - 471s 4s/step - loss: 0.1360 - accuracy: 0.9537 - val_loss: 0.4824 - val_accuracy: 0.8450
Epoch 10/25
117/117 [==============================] - 460s 4s/step - loss: 0.1277 - accuracy: 0.9543 - val_loss: 0.1680 - val_accuracy: 0.9483
Epoch 11/25
117/117 [==============================] - 466s 4s/step - loss: 0.0618 - accuracy: 0.9801 - val_loss: 0.0787 - val_accuracy: 0.9666
Epoch 12/25
117/117 [==============================] - 472s 4s/step - loss: 0.0632 - accuracy: 0.9774 - val_loss: 0.0675 - val_accuracy: 0.9795
Epoch 13/25
117/117 [==============================] - 467s 4s/step - loss: 0.0433 - accuracy: 0.9844 - val_loss: 0.0521 - val_accuracy: 0.9785
Epoch 14/25
117/117 [==============================] - 476s 4s/step - loss: 0.0386 - accuracy: 0.9868 - val_loss: 0.0586 - val_accuracy: 0.9806
Epoch 15/25
117/117 [==============================] - 466s 4s/step - loss: 0.0591 - accuracy: 0.9806 - val_loss: 0.1001 - val_accuracy: 0.9634
Epoch 16/25
117/117 [==============================] - 476s 4s/step - loss: 0.0693 - accuracy: 0.9752 - val_loss: 0.0889 - val_accuracy: 0.9742
Epoch 17/25
 37/117 [========>.....................] - ETA: 5:00 - loss: 0.0431 - accuracy: 0.9856
```

```
import tensorflow as tf
from tensorflow.keras.datasets import fashion_mnist


# Load the dataset
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()

# Normalize the images
train_images = train_images / 255.0
test_images = test_images / 255.0
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
```

```
5148/5148 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 [==============================] - 0s 0us/step
```

```
# Define the CNN model
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), padding='same', activation='relu', input_shape=(28,28,1)),
    tf.keras.layers.MaxPooling2D((2,2)),
    tf.keras.layers.Conv2D(64, (3,3), padding='same', activation='relu'),
    tf.keras.layers.MaxPooling2D((2,2)),
    tf.keras.layers.Conv2D(64, (3,3), padding='same', activation='relu'),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10)
])
```

```
# Compile the model
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

# Train the model
model.fit(train_images[..., tf.newaxis], train_labels, epochs=5)

# Evaluate the model
test_loss, test_acc = model.evaluate(test_images[..., tf.newaxis], test_labels)
print('Test accuracy:', test_acc)
```

```
Epoch 1/5
1875/1875 [==============================] - 96s 50ms/step - loss: 0.4045 - accuracy: 0.8523
Epoch 2/5
1875/1875 [==============================] - 92s 49ms/step - loss: 0.2593 - accuracy: 0.9045
Epoch 3/5
1875/1875 [==============================] - 91s 49ms/step - loss: 0.2137 - accuracy: 0.9206
Epoch 4/5
1875/1875 [==============================] - 91s 49ms/step - loss: 0.1825 - accuracy: 0.9320
Epoch 5/5
1875/1875 [==============================] - 91s 48ms/step - loss: 0.1544 - accuracy: 0.9427
313/313 [==============================] - 4s 13ms/step - loss: 0.2326 - accuracy: 0.9160
Test accuracy: 0.9160000085830688
```

```
import numpy as np
import matplotlib.pyplot as plt
```

```
# Define class names
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

# Make predictions on the test set
predictions = model.predict(test_images[..., tf.newaxis])

# Plot a random sample of test images with their predicted labels
num_rows, num_cols = 5, 3
num_images = num_rows * num_cols
plt.figure(figsize=(2*num_cols, 2*num_rows))
for i in range(num_images):
    plt.subplot(num_rows, num_cols, i+1)
    plt.imshow(test_images[i], cmap='gray')
    predicted_label = np.argmax(predictions[i])
    true_label = test_labels[i]
    if predicted_label == true_label:
        color = 'green'
    else:
        color = 'red'
    plt.title('{} ({})'.format(class_names[predicted_label], class_names[true_label]), color=color)
    plt.axis('off')
plt.show()
```

```
313/313 [==============================] - 5s 16ms/step
```