! wget https://data.mendeley.com/public-files/datasets/tywbtsjrjv/files/b4e3a32f-c0bd-4060-81e9-6144231f2520/file\_downloaded -O plant\_disease\_dataset.zip

! 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/

# 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()

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

# 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)

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()