CS409 Class project

Result of Enhancing Potato Leaf Disease Detection Using Neural Networks

s/18/904

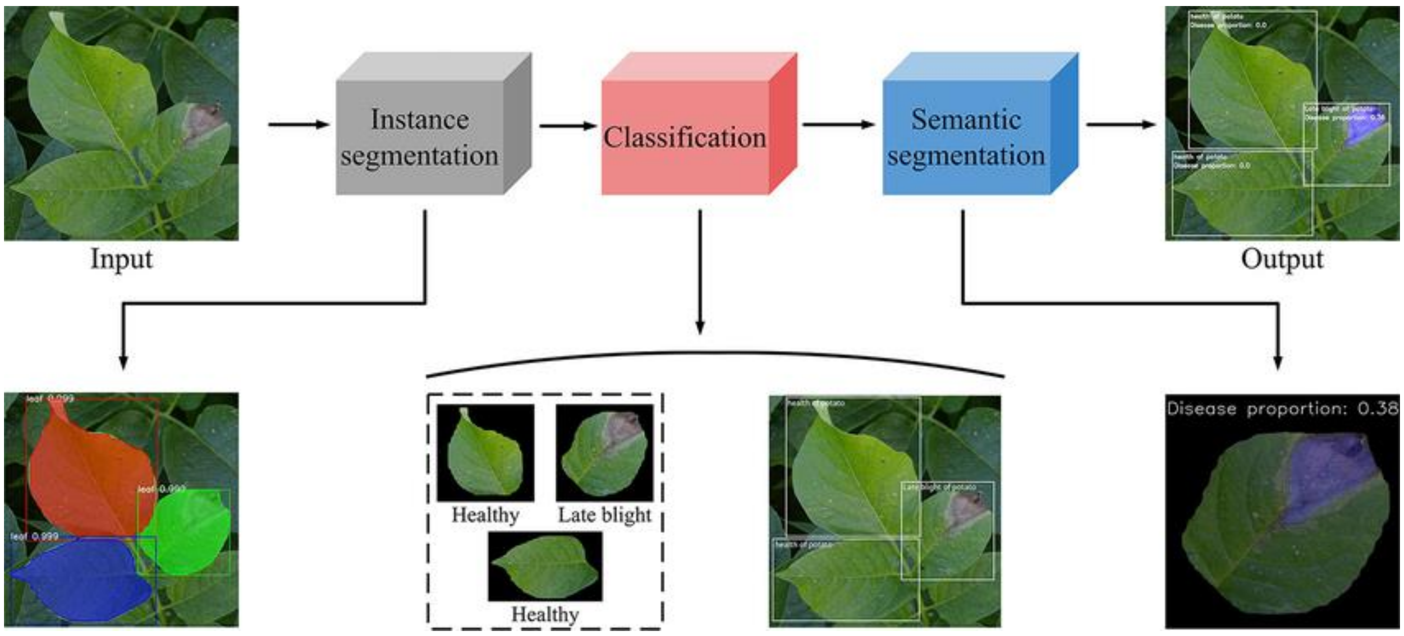
Introduction

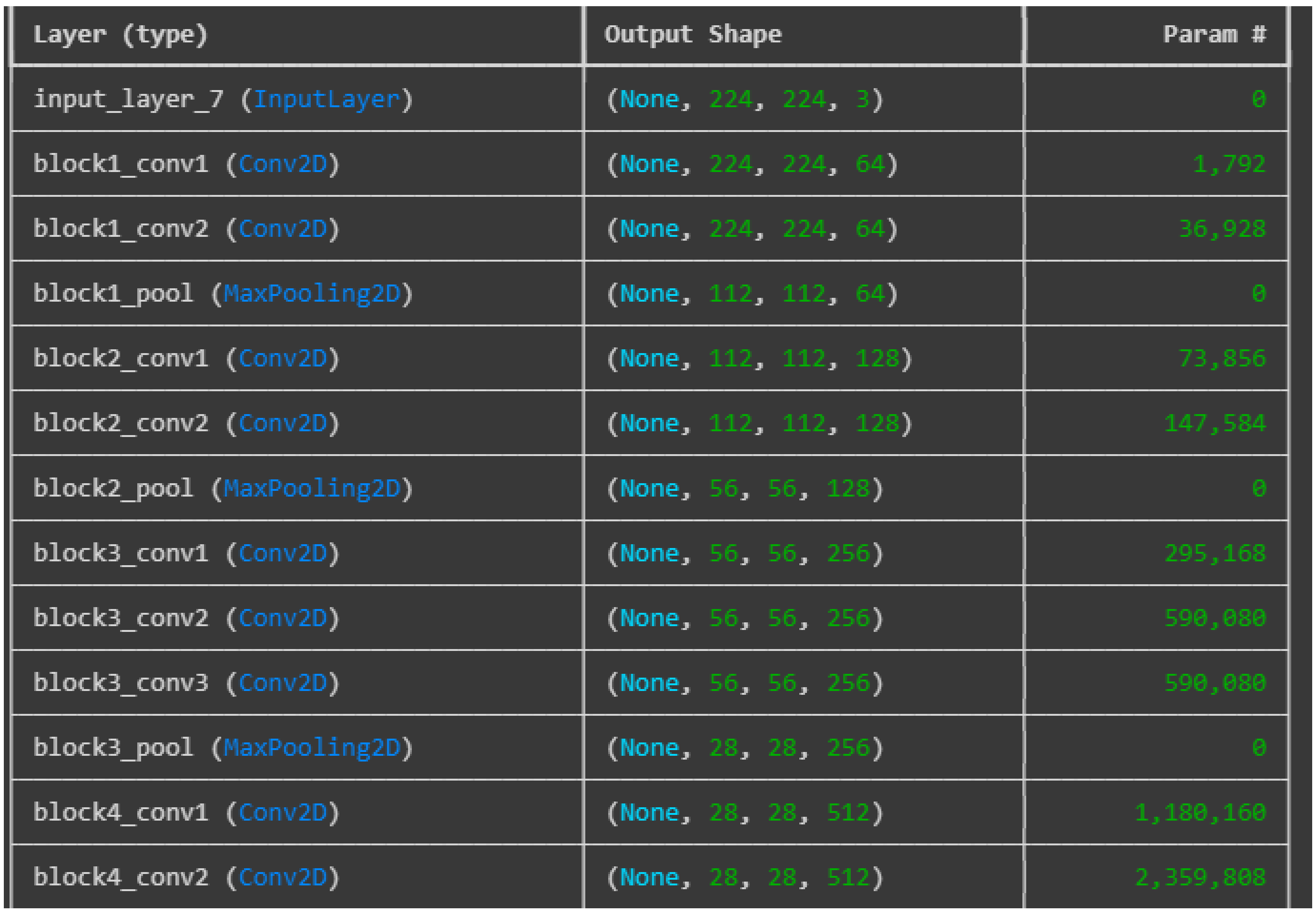
The productivity of agriculture is greatly impacted by potato leaf diseases, although early detection can minimize damage and improve crop management. This project suggests using TensorFlow to detect potato leaf diseases using a neural network-based method. The model attempts to increase accuracy in identifying different leaf diseases by utilizing contemporary machine learning techniques, including Linear Discriminant Analysis (LDA) for feature extraction and deep learning for classification.

Method

The model makes use of the sparse categorical crossentropy loss function, which works well for jobs involving the classification of many classes that have integer labels. The Adam optimizer, which excels at optimizing complex models due to its versatility, is used to guarantee effective gradient descent. The main assessment criterion is accuracy, which is monitored to keep an eye on the model's performance during training. Using LDA-transformed features and flattened label arrays, the model is trained for 10 epochs to improve its detection accuracy of several potato leaf diseases. This architecture is specifically made to strike a compromise between computational efficiency and detection accuracy, which will ultimately help farmers implement disease management measures that are timelier and more effective.

2. Neural Network Implementation





```
Layer (type)                     Output Shape                 Param #
input_layer_7 (InputLayer)       (None, 224, 224, 3)                0
block1_conv1 (Conv2D)            (None, 224, 224, 64)           1,792
block1_conv2 (Conv2D)            (None, 224, 224, 64)          36,928
block1_pool (MaxPooling2D)       (None, 112, 112, 64)               0
block2_conv1 (Conv2D)            (None, 112, 112, 128)         73,856
block2_conv2 (Conv2D)            (None, 112, 112, 128)        147,584
block2_pool (MaxPooling2D)       (None, 56, 56, 128)                0
block3_conv1 (Conv2D)            (None, 56, 56, 256)          295,168
block3_conv2 (Conv2D)            (None, 56, 56, 256)          590,080
block3_conv3 (Conv2D)            (None, 56, 56, 256)          590,080
block3_pool (MaxPooling2D)       (None, 28, 28, 256)                0
block4_conv1 (Conv2D)            (None, 28, 28, 512)        1,180,160
block4_conv2 (Conv2D)            (None, 28, 28, 512)        2,359,808
```

The model is constructed using a **Sequential neural network**, starting with an input layer where the size is determined by the features extracted through **Linear Discriminant Analysis (LDA)**. This ensures the model emphasizes the most discriminative features. The first dense layer consists of **128 neurons** with **ReLU activation**, designed to capture complex patterns within the input data. A second dense layer, with **64 neurons** and **ReLU activation**, further refines these features. The final output layer is a **softmax layer** with **3 output neurons**, corresponding to the three disease categories (or a healthy state), enabling effective multi-class classification.

Code

```
!pip install tensorflow
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator

image_directory = '/content/drive/MyDrive/AI/PLD_3_Classes_256'


datagen = ImageDataGenerator(rescale=1./255)


image_generator = datagen.flow_from_directory(
    image_directory,
    target_size=(256, 256),
    batch_size=32,
    class_mode='categorical'
)


model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(256, 256,
3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(3, activation='softmax')
])


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


model.fit(image_generator, epochs=10)
```

```python
datagen = ImageDataGenerator(featurewise_center=True,
featurewise_std_normalization=True)
```

```python
model.save('Naveen_model.h5')
```

```python
from tensorflow.keras.models import load_model


loaded_model = load_model('Naveen_model.h5')
```

```python
loss, accuracy = loaded_model.evaluate(features_lda, labels_flattened)
print(f'Test accuracy: {accuracy}')
print(f'Test loss: {loss}')
```

```python
import numpy as np
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg16 import preprocess_input
from collections import Counter

loaded_model = load_model('Naveen_model.h5')


uploaded = files.upload()

image_files = list(uploaded.keys())


for image_file in image_files:

  img = image.load_img(image_file, target_size=(224, 224))
  x = image.img_to_array(img)
  x = np.expand_dims(x, axis=0)
  x = preprocess_input(x)


  model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
  features = model.predict(x)
  features = features.reshape((features.shape[0], -1))

  scaler = StandardScaler()
  features_scaled = scaler.fit_transform(features)




  predictions = loaded_model.predict(features_lda)
```

```
predicted_labels = np.argmax(predictions, axis=1)

print(f"Image: {image_file}, Predicted label: {predicted_labels[0]}")
```

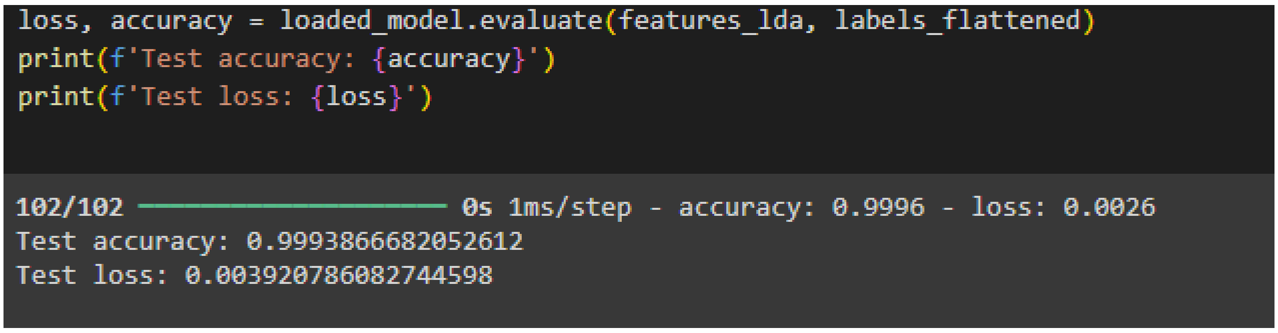
Model Performance

1.4200 images using for taring process

2.The train was set up to train 10 epochs

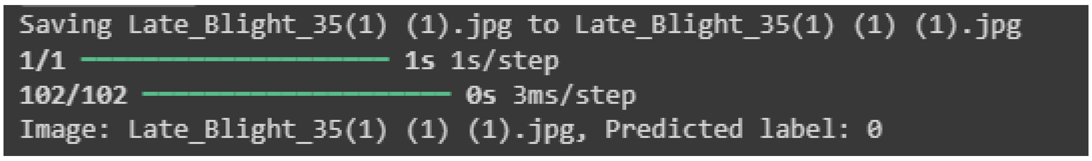
Results

This model is designed to detect potato leaf diseases with a high degree of accuracy. It classifies the leaf condition into three categories: **early blight (1)**, **late blight (0)**, and **healthy (2)**. According to the results from model training, it achieves an impressive **99% accuracy**, making it a highly reliable tool for early detection and classification of potato leaf diseases. This high level of precision ensures effective disease management, helping farmers take timely and appropriate actions to protect their crops.

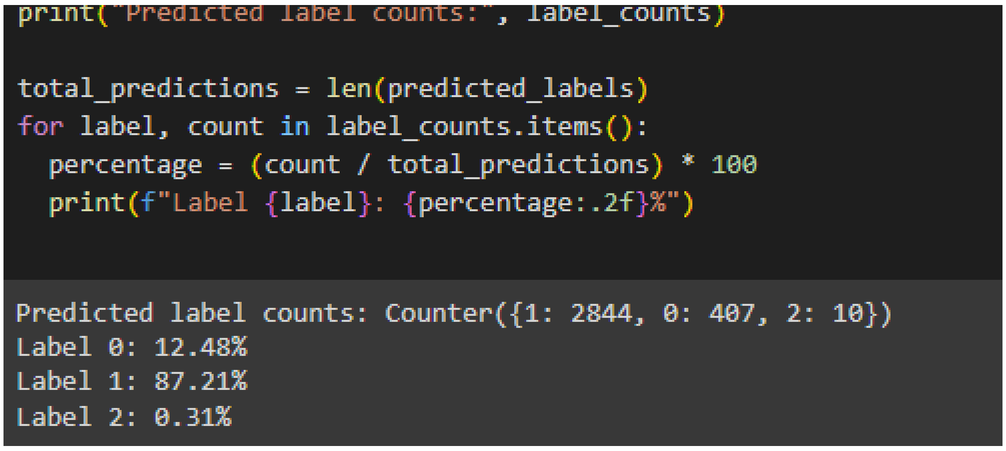


```
loss, accuracy = loaded_model.evaluate(features_lda, labels_flattened)
print(f'Test accuracy: {accuracy}')
print(f'Test loss: {loss}')

102/102 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9996 - loss: 0.0026
Test accuracy: 0.9993866682052612
Test loss: 0.003920786082744598
```



```
Saving Late_Blight_35(1) (1).jpg to Late_Blight_35(1) (1) (1).jpg
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step
102/102 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
Image: Late_Blight_35(1) (1) (1).jpg, Predicted label: 0
```



```python
print("Predicted label counts:", label_counts)

total_predictions = len(predicted_labels)
for label, count in label_counts.items():
  percentage = (count / total_predictions) * 100
  print(f"Label {label}: {percentage:.2f}%")
```

```
Predicted label counts: Counter({1: 2844, 0: 407, 2: 10})
Label 0: 12.48%
Label 1: 87.21%
Label 2: 0.31%
```