1. Import Libraries

```
import os
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Define the path to your dataset
dataset_dir = '../Cucumber_dataset/'
```

```
!ls ../Cucumber_dataset/
```

```
Anthracnose       Belly Rot         Fresh Cucumber    Gummy Stem Blight
Bacterial Wilt    Downy Mildew      Fresh Leaf        Pythium Fruit Rot
```

2. Loading dataset

```
# Create an instance of the ImageDataGenerator for training with data augmentation
train_datagen = ImageDataGenerator(
    rescale=1.0/255.0,      # Rescale pixel values (normalization)
    validation_split=0.2    # Split for validation (20% of data used for validation)
)

# Load the dataset from the directory and split into train and validation sets
train_generator = train_datagen.flow_from_directory(
    dataset_dir,
    target_size=(150, 150),     # Resize images to 150x150
    batch_size=32,
    class_mode='categorical',   # For multi-class classification
    subset='training'           # Load the training set
)
```

```
validation_generator = train_datagen.flow_from_directory(
    dataset_dir,
    target_size=(150, 150),
    batch_size=32,
    class_mode='categorical',
    subset='validation'         # Load the validation set
)
```

```
Found 5120 images belonging to 8 classes.
Found 1280 images belonging to 8 classes.
```

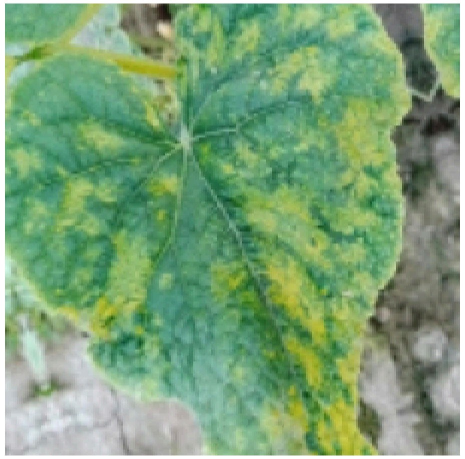
3. Dataset representation

```
# Get the class indices
class_labels = train_generator.class_indices
print("Class labels:", class_labels)

# Get a batch of images and labels from the training data
images, labels = next(train_generator)

# Plot several images with labels
plt.figure(figsize=(10, 10))
for i in range(9):  # Show 9 images
    plt.subplot(3, 3, i + 1)
    plt.imshow(images[i])
    plt.title(list(class_labels.keys())[list(labels[i]).index(1)])  # Show class label
    plt.axis('off')
plt.show()
```

```
Class labels: {'Anthracnose': 0, 'Bacterial Wilt': 1, 'Belly Rot': 2, 'Downy Mildew': 3, 'Fresh Cucumber': 4, 'Fresh
```

Anthracnose

Downy Mildew

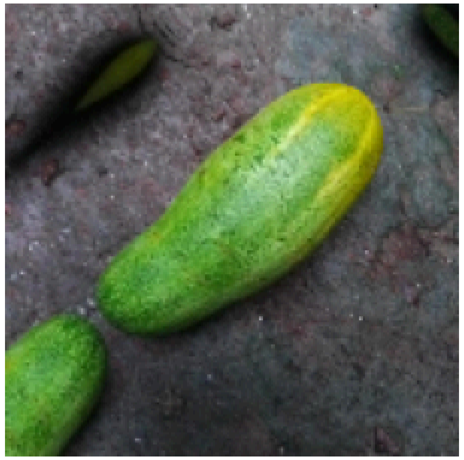


Gummy Stem Blight

Bacterial Wilt

Anthracnose

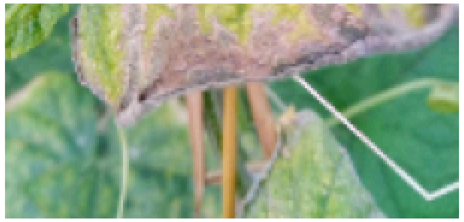
Fresh Cucumber



Gummy Stem Blight

Pythium Fruit Rot

Gummy Stem Blight



4.Deep Learning Model

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(len(class_labels), activation='softmax')
])
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

```
/Users/oybekeraliev/Documents/AI_agranom/DL_model/my_env/lib/python3.10/site-packages/keras/src/layers/convolutional,
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
2024-11-15 10:36:14.827284: I metal_plugin/src/device/metal_device.cc:1154] Metal device set to: Apple M1 Pro
2024-11-15 10:36:14.827308: I metal_plugin/src/device/metal_device.cc:296] systemMemory: 16.00 GB
2024-11-15 10:36:14.827319: I metal_plugin/src/device/metal_device.cc:313] maxCacheSize: 5.33 GB
2024-11-15 10:36:14.827353: I tensorflow/core/common_runtime/pluggable_device/pluggable_device_factory.cc:305] Could
2024-11-15 10:36:14.827368: I tensorflow/core/common_runtime/pluggable_device/pluggable_device_factory.cc:271] Creat
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                     ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ conv2d (Conv2D)                  │ (None, 148, 148, 32)   │           896 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d (MaxPooling2D)     │ (None, 74, 74, 32)     │             0 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_1 (Conv2D)                │ (None, 72, 72, 64)     │        18,496 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_1 (MaxPooling2D)   │ (None, 36, 36, 64)     │             0 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ flatten (Flatten)                │ (None, 82944)          │             0 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                    │ (None, 128)            │    10,616,960 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                  │ (None, 8)              │         1,032 │
└──────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 10,637,384 (40.58 MB)
 Trainable params: 10,637,384 (40.58 MB)
 Non-trainable params: 0 (0.00 B)
```

5.Model training

```
#Training
history = model.fit(train_generator,epochs=5, validation_data=validation_generator)
```

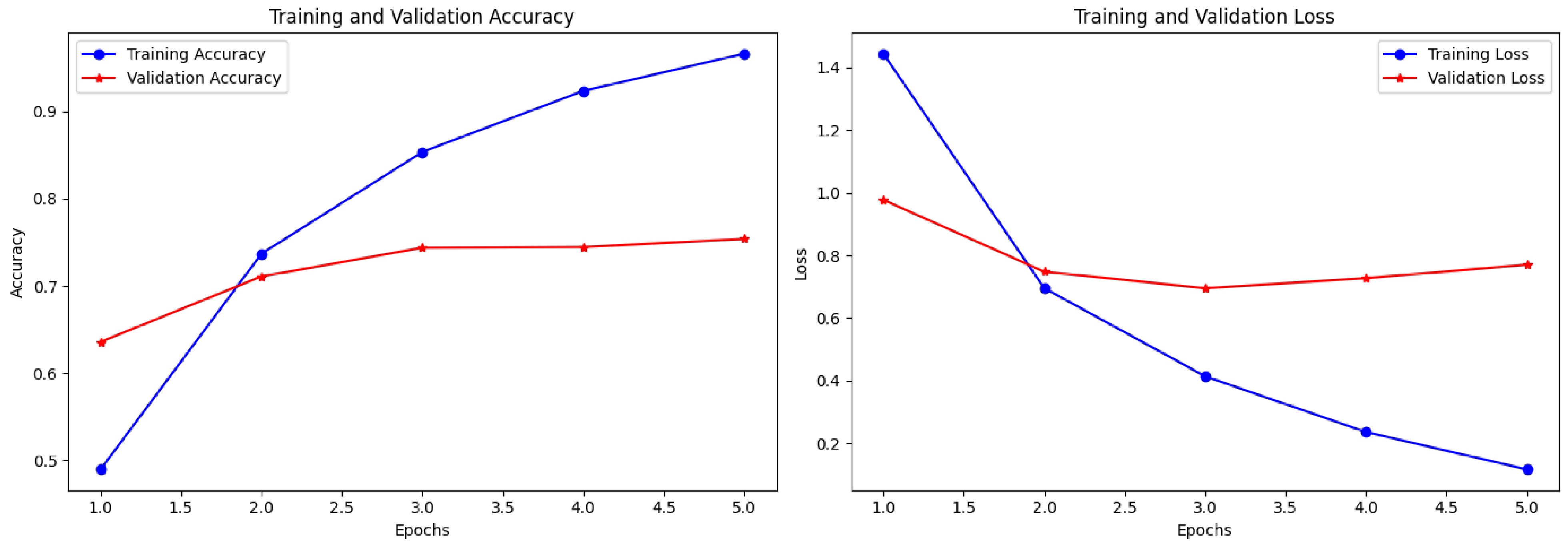
```
Epoch 1/5
2024-11-15 10:36:55.170165: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:117] Plugin opt
/Users/oybekeraliev/Documents/AI_agranom/DL_model/my_env/lib/python3.10/site-packages/keras/src/trainers/data_adapte
```

```
  self._warn_if_super_not_called()
160/160 ━━━━━━━━━━━━━━━━━━━━ 62s 368ms/step - accuracy: 0.3373 - loss: 2.0744 - val_accuracy: 0.6359 - val_loss: 0.9
Epoch 2/5
160/160 ━━━━━━━━━━━━━━━━━━━━ 58s 352ms/step - accuracy: 0.7287 - loss: 0.7170 - val_accuracy: 0.7109 - val_loss: 0.7
Epoch 3/5
160/160 ━━━━━━━━━━━━━━━━━━━━ 57s 347ms/step - accuracy: 0.8482 - loss: 0.4291 - val_accuracy: 0.7437 - val_loss: 0.6
Epoch 4/5
160/160 ━━━━━━━━━━━━━━━━━━━━ 58s 347ms/step - accuracy: 0.9339 - loss: 0.2223 - val_accuracy: 0.7445 - val_loss: 0.7
Epoch 5/5
160/160 ━━━━━━━━━━━━━━━━━━━━ 58s 349ms/step - accuracy: 0.9678 - loss: 0.1171 - val_accuracy: 0.7539 - val_loss: 0.7
```

6.Model Evaluation

```python
# Plotting the training and validation loss and accuracy
import matplotlib.pyplot as plt

# Get the training history
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc) + 1)

# Plot training and validation accuracy
plt.figure(figsize=(14, 5))
plt.subplot(1, 2, 1)
plt.plot(epochs, acc, 'bo-', label='Training Accuracy')
plt.plot(epochs, val_acc, 'r*-', label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

# Plot training and validation loss
plt.subplot(1, 2, 2)
```

```
    plt.plot(epochs, loss, 'bo-', label='Training Loss')
    plt.plot(epochs, val_loss, 'r*-', label='Validation Loss')
    plt.title('Training and Validation Loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()

    plt.tight_layout()
    plt.show()
```

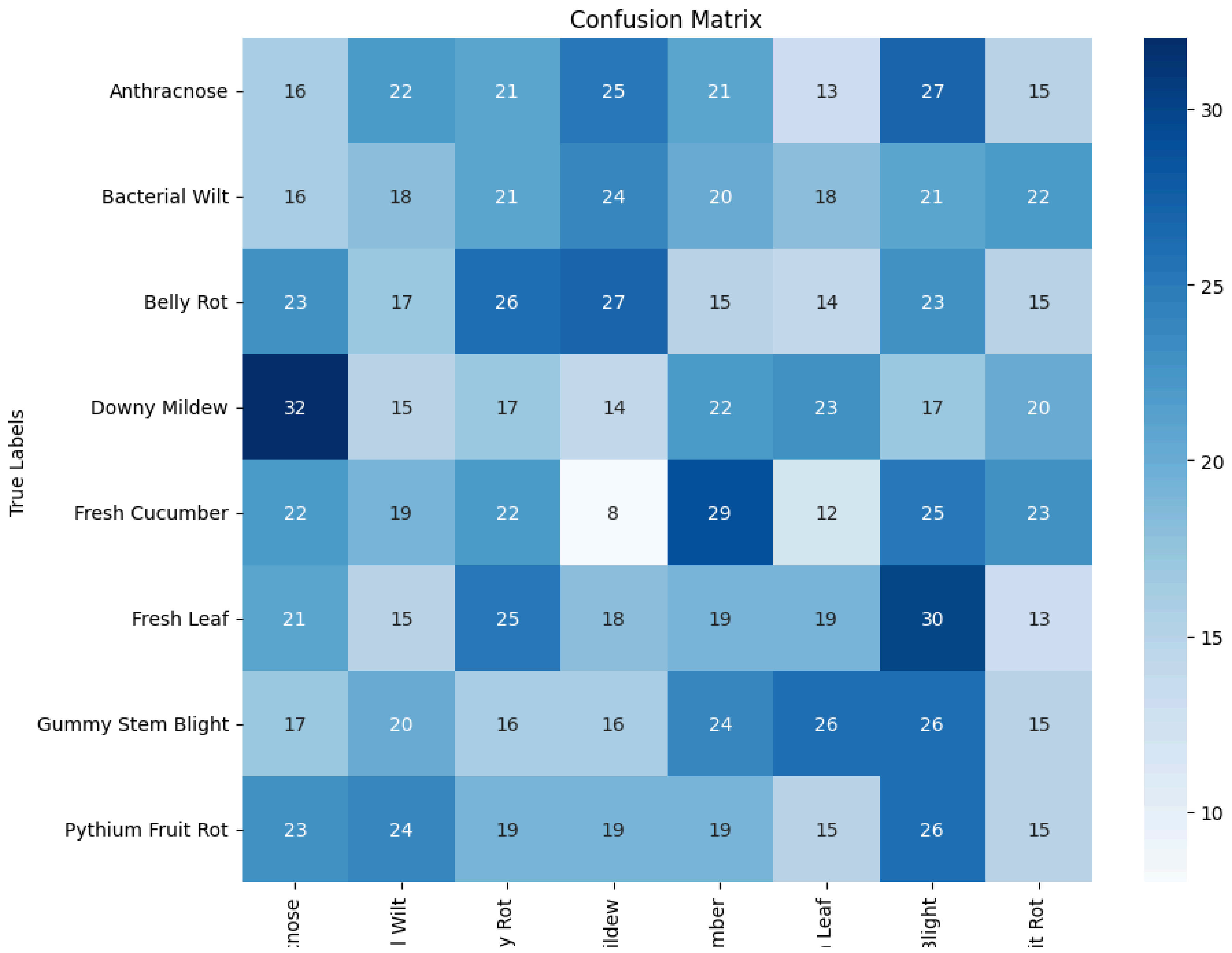


```
# Evaluate the model on the validation set
val_loss, val_accuracy = model.evaluate(validation_generator)
print(f"Validation Loss: {val_loss}")
print(f"Validation Accuracy: {val_accuracy}")
```

```
40/40 ━━━━━━━━━━━━━━━━━━━━ 11s 282ms/step - accuracy: 0.7621 - loss: 0.7640
Validation Loss: 0.7703855633735657
Validation Accuracy: 0.75390625
```

```
import numpy as np
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import seaborn as sns

# Get true labels and predictions for the validation set
validation_generator.reset()  # Reset the validation generator before predicting
predictions = model.predict(validation_generator, verbose=1)

# Convert predictions from one-hot encoded to label indices
predicted_classes = np.argmax(predictions, axis=1)

# True labels from the generator (these are also indices)
true_classes = validation_generator.classes

# Get the class labels from the generator
class_labels = list(validation_generator.class_indices.keys())

# Generate the confusion matrix
conf_matrix = confusion_matrix(true_classes, predicted_classes)

# Plot the confusion matrix using seaborn
plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=class_labels, yticklabels=class_labels)
plt.title('Confusion Matrix')
plt.ylabel('True Labels')
plt.xlabel('Predicted Labels')
plt.show()
```

```
40/40 ━━━━━━━━━━━━━━━━━━━━ 12s 287ms/step
```



True Labels	nose	l Wilt	y Rot	ildew	mber	Leaf	3light	it Rot
Anthracnose	16	22	21	25	21	13	27	15
Bacterial Wilt	16	18	21	24	20	18	21	22
Belly Rot	23	17	26	27	15	14	23	15
Downy Mildew	32	15	17	14	22	23	17	20
Fresh Cucumber	22	19	22	8	29	12	25	23
Fresh Leaf	21	15	25	18	19	19	30	13
Gummy Stem Blight	17	20	16	16	24	26	26	15
Pythium Fruit Rot	23	24	19	19	19	15	26	15

```
from sklearn.metrics import f1_score
f1 = f1_score(true_classes, predicted_classes, average='macro')
print(f"Model f_1 score: {f1}")
```

```
Model f_1 score: 0.1265325955568996
```

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.