



Name: Waqas Kureshy

SID: 015421575

Course: Data Mining

Title: Project Report

Potato Crop Disease Classifier

Description:

In this Project a Potato Disease Classifier was implemented using Tensorflow, a User Interface was implemented using React JS.

Dataset:

The dataset is taken from Kaggle datasets, the Dataset is called "**Plant Village**". The dataset includes images of various Vegetable plants including Peppers, Tomatoes and Potatoes. The Model designed, focuses on detecting the health of the Potato crop.

Dataset URL:<https://www.kaggle.com/datasets/emmarex/plantdisease>

The dataset includes three classes and the dataset comprises of 2152 images and the distribution is given as:

1. Healthy Plant (152 images)
2. Late Blight (1000 images)
3. Early Blight (1000 images)

Globally The potato crop diseases are classified into two types: **Early Blight** and **Late Blight**, the treatments for these diseases differ slightly, so it is essential to determine what disease the crop is suffering from.

Samples:

- a. **Healthy Crop:** A healthy potato plant leaf has no discoloration, it is bright green in color and is free from any sort of specs.



- b. **Early Blight Crop:** Early Blight is a disease in the Potato plant crop, its leaf has small dark specs that are distinct and small.



- c. **Late Blight Crop:** Late Blight is a Potato crop disease that also has black specs, but they are considerably large and connected as compared to the Early Blight disease.



Model:

The model designed for this task is a sequential model with 17 layers, a model summary is given below:

Layer (type)	Output Shape	Param #
=====	=====	=====
sequential_5 (Sequential)	(32, 256, 256, 3)	0
sequential_6 (Sequential)	(32, 256, 256, 3)	0
conv2d_24 (Conv2D)	(32, 254, 254, 32)	896
max_pooling2d_23 (MaxPooling2D)	(32, 127, 127, 32)	0
conv2d_25 (Conv2D)	(32, 125, 125, 64)	18496
max_pooling2d_24 (MaxPooling2D)	(32, 62, 62, 64)	0
conv2d_26 (Conv2D)	(32, 60, 60, 64)	36928
max_pooling2d_25 (MaxPooling2D)	(32, 30, 30, 64)	0
conv2d_27 (Conv2D)	(32, 28, 28, 64)	36928
max_pooling2d_26 (MaxPooling2D)	(32, 14, 14, 64)	0
conv2d_28 (Conv2D)	(32, 12, 12, 64)	36928
max_pooling2d_27 (MaxPooling2D)	(32, 6, 6, 64)	0
conv2d_29 (Conv2D)	(32, 4, 4, 64)	36928
max_pooling2d_28 (MaxPooling2D)	(32, 2, 2, 64)	0
flatten_3 (Flatten)	(32, 256)	0
dense_6 (Dense)	(32, 64)	16448
dense_7 (Dense)	(32, 3)	195

Tensorflows Keras library was used to make this model, Data Augmentation techniques such as Rescaling, Resizing, RandomFlip and Rotation have been used to make a robust model. Some important parameters for the model are given below:

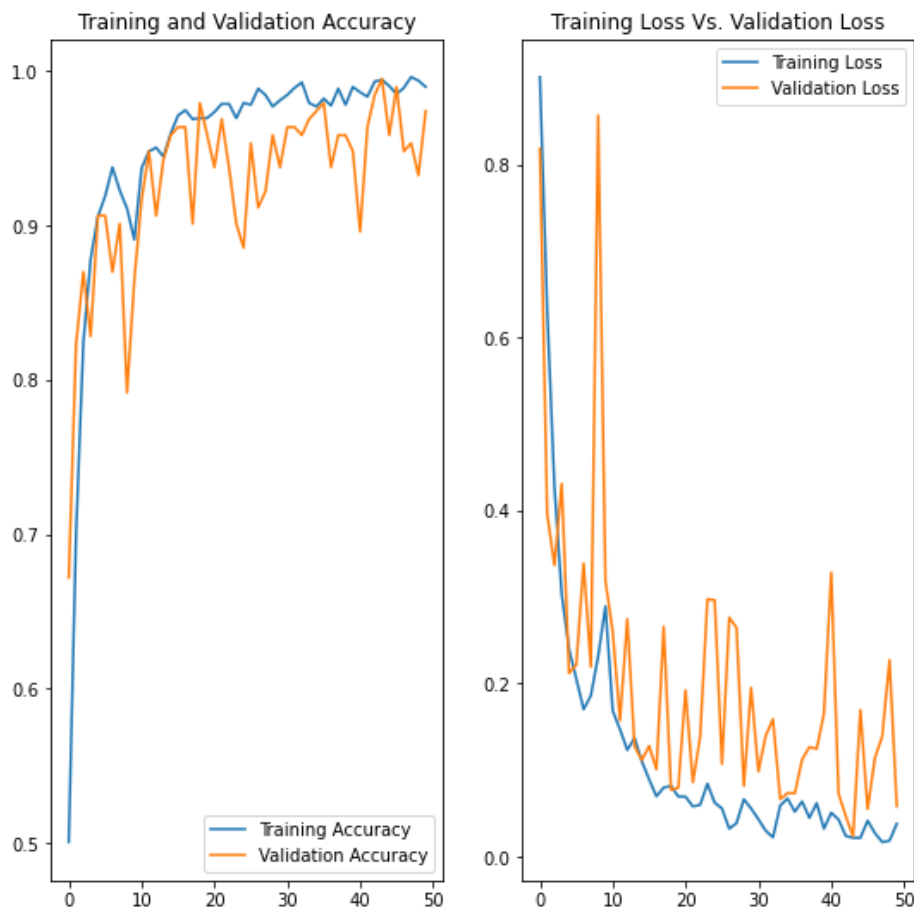
- Optimizer: Adam
- Loss Function: Sparse Categorical Cross Entropy
- Epochs: 50

On evaluation the following loss and accuracy were obtained:

Loss: 0.0141

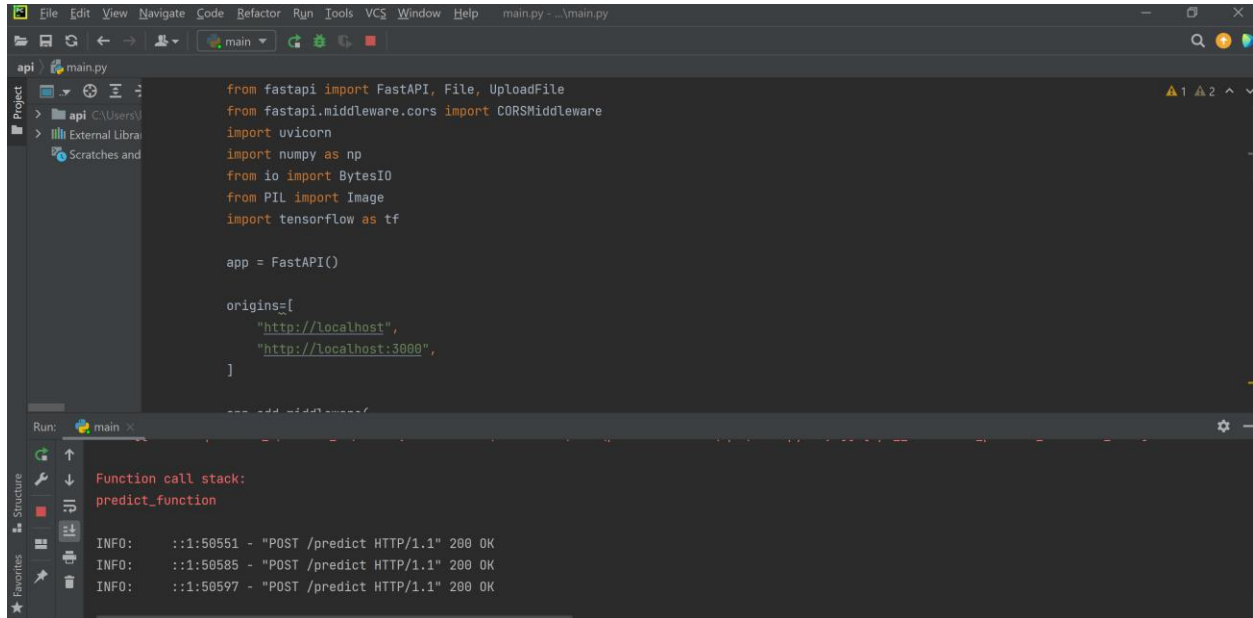
Accuracy: 0.9961

The Graph below highlights the model's Accuracy and Loss.



Interface:

FastAPI, Uvicorn were used to cast results onto the static application designed for the classifier, the FastAPI presented an easy solution to implement and show detector outputs of the classifier to the User interface. Uvicorn was used to connect the detector to the application. Postman was used to confirm result outputs. The image below shows output of the backend and a Postman request to the Classifier.



```
from fastapi import FastAPI, File, UploadFile
from fastapi.middleware.cors import CORSMiddleware
import uvicorn
import numpy as np
from io import BytesIO
from PIL import Image
import tensorflow as tf

app = FastAPI()

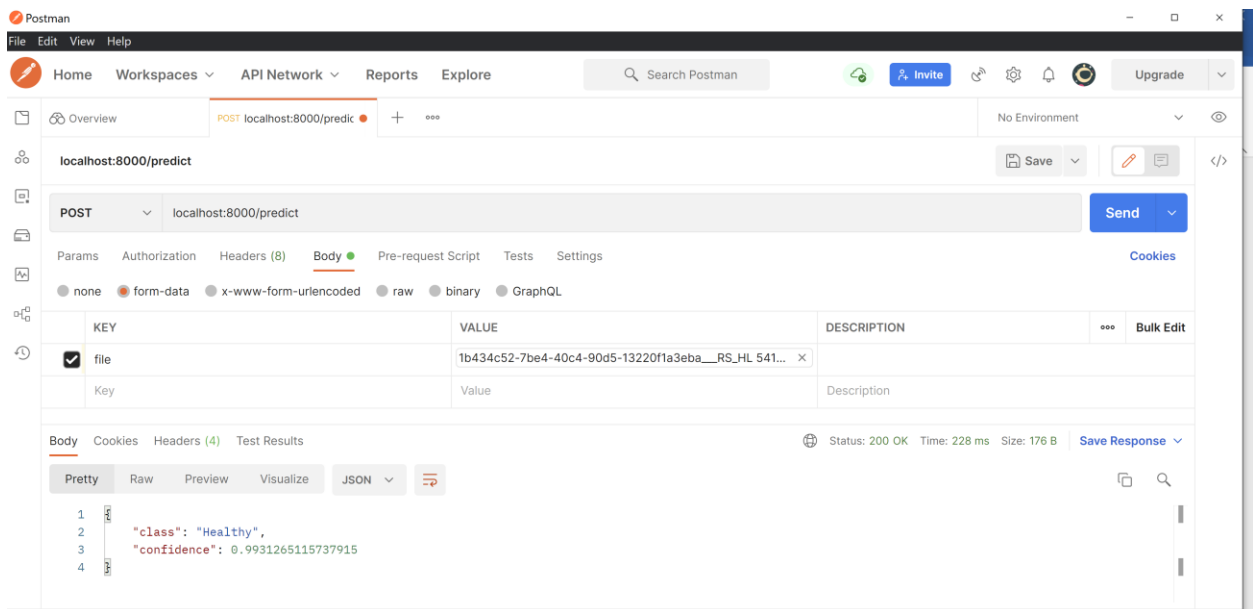
origins=[
    "http://localhost",
    "http://localhost:3000",
]
```

Run: main x

Function call stack:
predict_function

INFO: ::1:50551 - "POST /predict HTTP/1.1" 200 OK
INFO: ::1:50585 - "POST /predict HTTP/1.1" 200 OK
INFO: ::1:50597 - "POST /predict HTTP/1.1" 200 OK

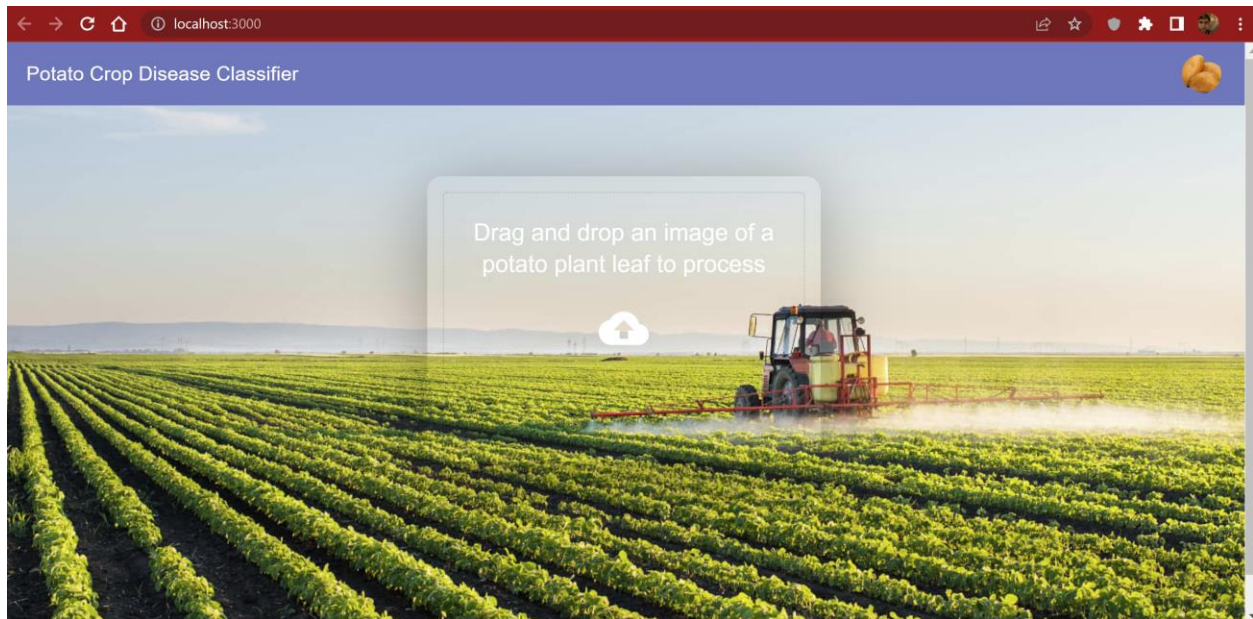
Postman request:



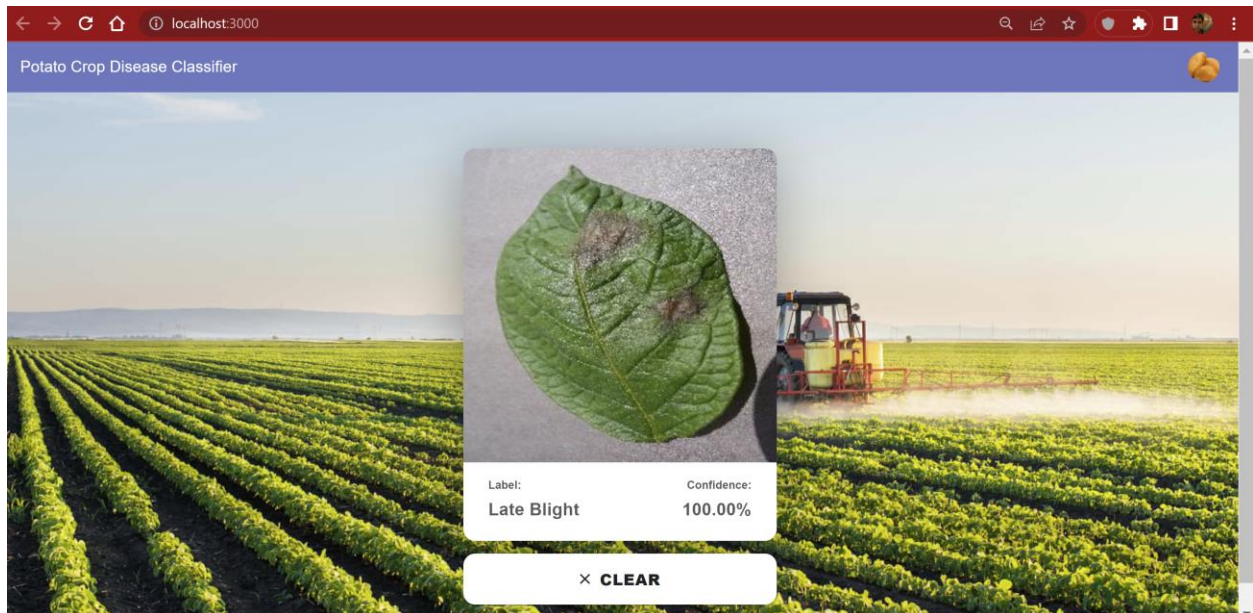
The detector's output shows the detected class for the image and gives out a confidence score for the prediction.

Frontend: For the frontend React JS was used, it's a simple interface where the user can add the image to be classified and the app makes an HTTP call to the backend and presents a class and a confidence score.

The figure below shows the Interface and an example output.



Output:



Issues and Future Developments:

- The results of the model's accuracy and loss values suggest that the model performs very well.
- The dataset contains images which show single leaves and if the crop is plagued by a disease it is shown clearly, this produced difficulties when the model when tested on random images from the internet that were not scaled right or had multiple leaves in one single image.
- If a more diverse dataset was available, the detector could have been more robust.
- As a future development task, the app could be deployed on cloud services and then the detector could also be used as an app where it could help farmers or enthusiasts or even students to determine if a crop is healthy. The model can also be extended to include other crops as well.