Visual Diagnostics: Detecting Tomato Plant Diseases through Leaf Image Analysis

Team Info:

Team ID: SWTID1720076593

Members:

1. INTURU VENKATA SAI SPURTHI (22BCE9510)
2. NADENDLA POORNIMA HARSHINI (22BCE9537)
3. VEJANDLA CHAKRISH (22BCE8330)
4. CHEETI SRIMAN (22BCE7220)

Introduction:

Project overviews:

This project aims to develop a deep learning model capable of automatically detecting tomato plant diseases by analyzing images of leaves. This technology has the potential to revolutionize how farmers diagnose and manage these diseases, leading to improved crop yields and sustainability in agriculture. It has many benefits like early identification of tomato plant diseases which allows farmers to take timely action, minimizing crop losses and data generated by the model can help decision-making regarding crop rotation, disease prevention strategies, and plant breeding for resistance.

Objectives:

- Develop a robust deep learning model to classify tomato leaf images into different disease categories.
- Enhance the dataset with preprocessing techniques to improve model performance.
- Evaluate the model's accuracy and reliability using metrics.
- Provide actionable insights to farmers and agronomists for better crop management.

- Design or choose a model architecture (e.g., CNN) that can effectively extract relevant features from the leaf images that are crucial for disease classification.

Project Initialization and Planning Phase

The "Project Initialization and Planning Phase" marks the project's outset, defining goals, scope, and key participants. This crucial phase establishes project parameters, identifies key team members, allocates resources, and outlines a realistic timeline. It also involves risk assessment and mitigation planning. Successful initiation sets the foundation for a well-organized and efficiently executed machine learning project, ensuring clarity, alignment, and proactive measures for potential challenges.

Define Problem Statement

Farmers, especially those in rural and under-resourced areas, face significant challenges in diagnosing tomato plant diseases accurately and promptly. Lack of access to agricultural experts and resources further exacerbates the issue, leading to reduced crop yields and financial losses.

Problem statement report: Click here

Proposal (Proposed solution)

The proposed project aims to develop a user-friendly, web-based platform that enables farmers to predict the disease affecting their tomato plants by simply uploading a picture of the affected plant. This platform aims to provide a quick, accurate, and accessible solution for disease diagnosis, helping farmers take timely and appropriate action to manage and treat their crops.

Project proposal template: Click here

Initial Project Planning

Initial Project Planning involves outlining key objectives, defining scope, and identifyingkey participants for a tomato plant disease detection system. It encompasses setting timelines, allocating resources, and determining the overall project strategy. During this phase, the team establishes a clear understanding of the dataset, formulates goals for analysis, and plans the workflow for data processing. Effective initial planning lays the foundation for a systematic and well-executed project, ensuring successful outcomes.

Project planning template: Click here

Data collection and Preprocessing Phase

The Data Collection and Preprocessing Phase involves executing a plan to gather relevant tomato plant disease data from Kaggle, ensuring data quality through verification and addressing missing values. Preprocessing tasks include cleaning, encoding, and organizing the dataset for subsequent exploratory analysis and machine learning model development.

Data Collection Plan & Raw Data Sources Identification

Detecting and diagnosing tomato plant diseases at an early stage is crucial for effective crop management and yield optimization. This project aims to develop a deep learning model to accurately identify various diseases in tomato plants using leaf images. Leveraging a comprehensive dataset from Kaggle, this project will ensure meticulous data curation and integrity for informed decision-making.

Data Collection Plan & Raw Data Sources Identification report: Click here

Data Quality Report

The dataset for "Visual Diagnostics: Detecting Tomato Plant Diseases through Leaf Image Analysis" is sourced from Kaggle. Data quality is ensured through thorough verification, addressing missing values, and maintaining adherence to ethical guidelines, establishing a reliable foundation for predictive modeling.

Data quality report: Click here

Data Preprocessing

The images will be preprocessed by resizing, normalizing, augmenting, denoising, adjusting contrast, detecting edges, converting color space, cropping, batch normalizing, and whitening data. These steps will enhance data quality, promote model generalization, and improve convergence during neural network training, ensuring robust and efficient performance across various computer vision tasks.

Data Preprocessing report: Click here

Model development Phase

The Model Development Phase entails crafting a predictive model for tomato plant disease detection. It encompasses strategic feature selection, evaluating and selecting models (Random Forest, Decision Tree, KNN, XGB), initiating training with code, and rigorously validating and assessing model performance for informed decision-making in disease management.

Model Selection Report

The Model Selection Report details the rationale behind choosing Random Forest, Decision Tree, KNN, and XGB models fortomato plant disease detection. It considers each model's strengths in handling complex relationships, interpretability, adaptability, and overall predictive performance, ensuring an informed choice aligned with project objectives.

Model selection report: Click here

Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include a summary and training and validation performance metrics for multiple models, presented through respective screenshots.
Initial Model Training Code, Model Validation and Evaluation Report: Click here

Model optimization and tuning Phase

The Model Optimization and Tuning Phase involves refining neural network models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

Hyperparameter Tuning Documentation

The Gradient Boosting model was selected for its superior performance, exhibiting high accuracy during hyperparameter tuning. Its ability to handle complex relationships, minimize overfitting, and optimize predictive accuracy aligns with project objectives, justifying its selection as the final model.

Performance Metrics Comparison Report

The Performance Metrics Comparison Report contrasts the baseline and optimized metrics for various models, specifically highlighting the enhanced performance of the Gradient Boosting model. This assessment provides a clear understanding of the refined predictive capabilities achieved through hyperparameter tuning.

Final Model Selection Justification

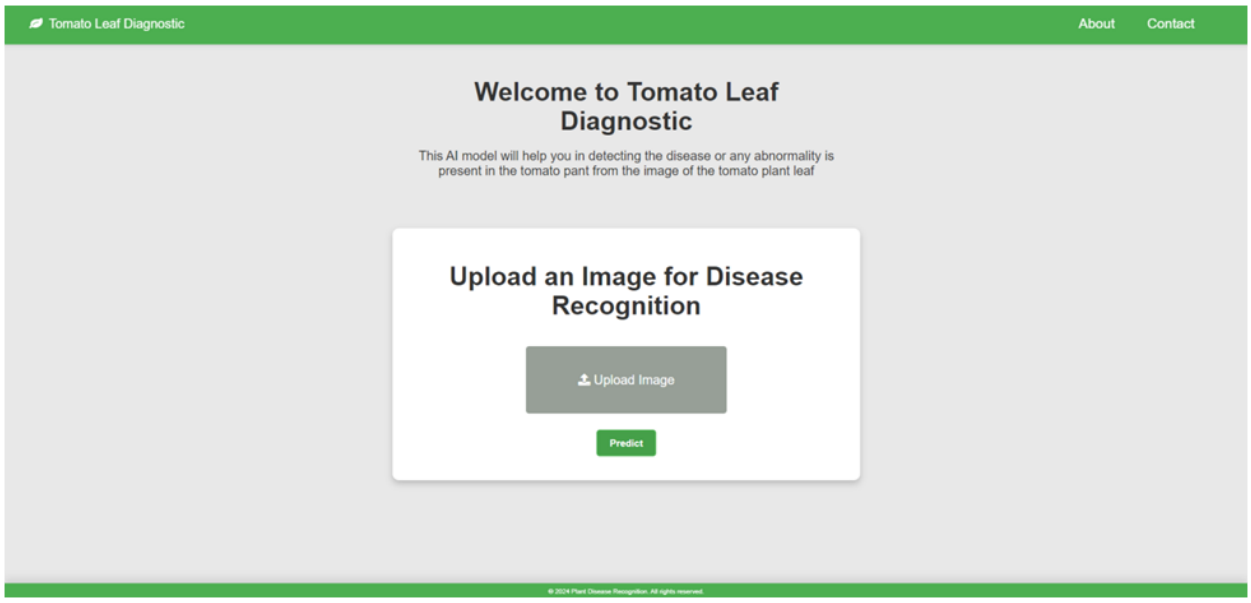
The Final Model Selection Justification articulates the rationale for choosing Gradient Boosting as the ultimate model. Its exceptional accuracy, ability to handle complexity, and successful hyperparameter tuning align with project objectives, ensuring optimaltomato plant disease detection.

Model optimization and tuning Phase report: Click here

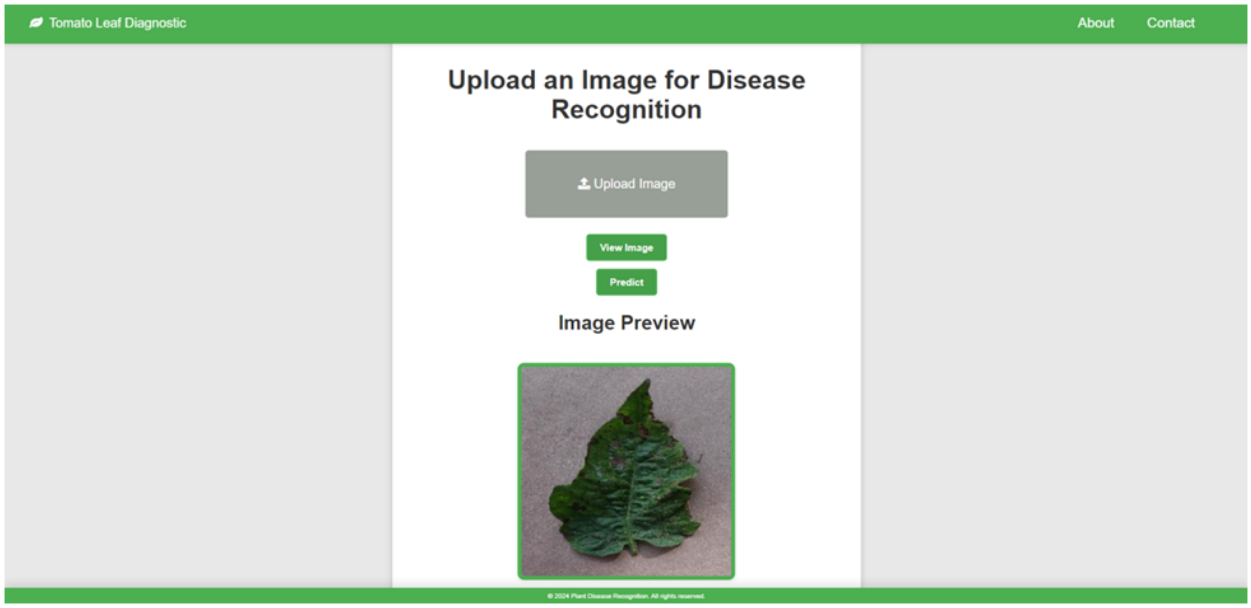
Result:

Output Screenshots:

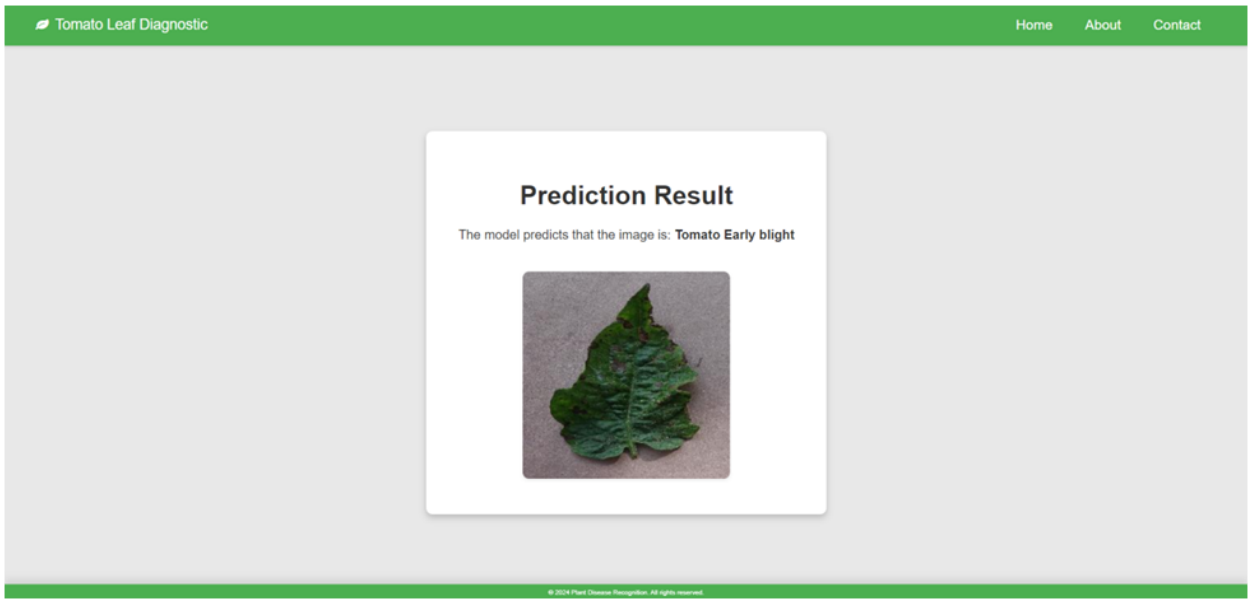
The below picture is the home page of our website



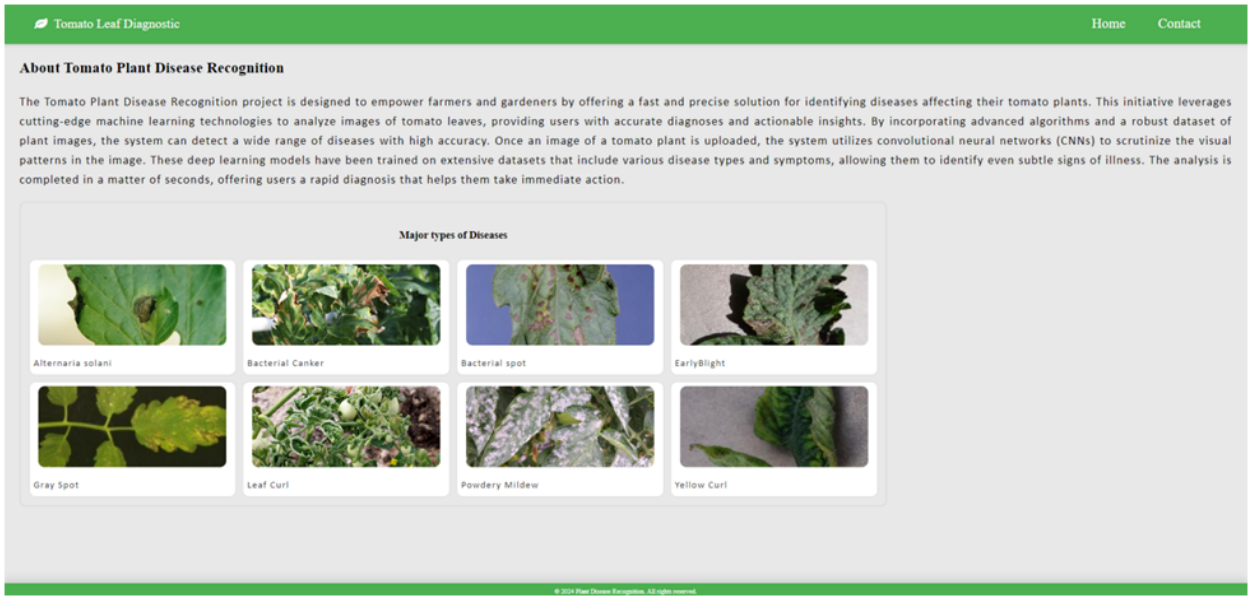
This is the preview of image which we uploaded.



This is the prediction result of our uploaded image



This is the about page of our website



Advantages & Disadvantages:

Advantages:

1. **Accuracy**: This model can achieve high accuracy in identifying diseases from leaf images due to their ability to learn complex patterns.
2. **Efficiency**: Automated disease detection is faster than manual inspection, enabling timely intervention.
3. **Scalability**: Once trained, it can be deployed on various platforms and used across large farms.
4. **Cost-effective**: Reduces the need for expert human resources for disease diagnosis.
5. **Consistency**: Provides consistent results, reducing human error in disease identification.

Disadvantages:

1. **Data Dependency**: It requires large, well-labeled datasets for effective training, which can be hard to obtain.
2. **Overfitting**: Models may overfit to the training data, performing poorly on unseen data.
3. **Complexity**: Developing and fine-tuning deep learning models requires specialized

knowledge and computational resources.

4. **Interpretability**: Deep learning models are often seen as black boxes, making it hard to understand how decisions are made.
5. **Environment Sensitivity**: Variability in lighting, background, and leaf conditions can affect model performance.

Conclusion:

In this project, we've explored various aspects of deep learning relevant to our task of detecting tomato plant diseases through leaf image analysis. The goal of our project is to develop a deep learning model that can effectively detect tomato plant diseases by analyzing images of leaves. This model can be used to identify various diseases early on which allows farmers to take necessary actions for better management and potentially higher crop yields. For this, we used the the concept of Convolutional Neural Network(CNN). We selected it because it is a best choice for image classification due to its ability to automatically extract relevant features from images through convolutional layers.

Future Scope:

- We can create a mobile application that allows farmers to capture images of their tomato leaves and receive real-time disease detection results.
- We can integrate the model into larger agricultural monitoring systems for large-scale disease detection and management.
- This model can be integrated into automated monitoring systems to continuously assess the health of tomato crops, allowing farmers to proactively address any potential disease outbreaks.
- While CNNs are a good starting point, we can explore alternative architectures like EfficientNets or DenseNets that might offer better performance or efficiency for your specific dataset.

Appendix:

Source code:

Plant Disease Prediction

Importing Dataset

Dataset Link: https://www.kaggle.com/datasets/vipoooool/new-plant-diseases-dataset

Importing libraries

```
[ ]: import tensorflow as tf
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from tensorflow import keras
```

Data Preprocessing

Training Image preprocessing

```
[ ]: training_set = tf.keras.utils.image_dataset_from_directory(
    'train',
    labels="inferred",
    label_mode="categorical",
    class_names=None,
    color_mode="rgb",
    batch_size=32,
    image_size=(128, 128),
    shuffle=True,
    seed=None,
    validation_split=None,
    subset=None,
    interpolation="bilinear",
    follow_links=False,
    crop_to_aspect_ratio=False
)
```

Validation Image Preprocessing

```
[ ]: validation_set = tf.keras.utils.image_dataset_from_directory(
    'valid',
    labels="inferred",
    label_mode="categorical",
    class_names=None,
    color_mode="rgb",
    batch_size=32,
    image_size=(128, 128),
    shuffle=True,
    seed=None,
    validation_split=None,
    subset=None,
    interpolation="bilinear",
    follow_links=False,
    crop_to_aspect_ratio=False
)
```

```
training_set •••
```

```
[ ]: for x,y in training_set:
    print(x,x.shape)
    print(y,y.shape)
    break
```

Building Model

```
from keras.layers import Dense,Conv2D,MaxPooling2D,Input,Flatten,Dropout
from keras.models import Sequential
```

```
model = Sequential()
```

```
model.add(Conv2D(filters=32, kernel_size=3, padding='same', activation='relu', input_shape=[128, 128, 3]))
model.add(Conv2D(filters=32, kernel_size=3, activation='relu'))
model.add(MaxPooling2D(pool_size=2, strides=2))
```

```
model.add(Conv2D(filters=64, kernel_size=3, padding='same', activation='relu'))
model.add(Conv2D(filters=64, kernel_size=3, activation='relu'))
model.add(MaxPooling2D(pool_size=2, strides=2))
```

```
model.add(Conv2D(filters=128, kernel_size=3, padding='same', activation='relu'))
model.add(Conv2D(filters=128, kernel_size=3, activation='relu'))
model.add(MaxPooling2D(pool_size=2, strides=2))
```

```
model.add(Conv2D(filters=256,kernel_size=3,padding='same',activation='relu'))
model.add(Conv2D(filters=256,kernel_size=3,activation='relu'))
model.add(MaxPooling2D(pool_size=2,strides=2))
```

```
model.add(Conv2D(filters=512,kernel_size=3,padding='same',activation='relu'))
model.add(Conv2D(filters=512,kernel_size=3,activation='relu'))
model.add(MaxPooling2D(pool_size=2,strides=2))
```

```
model.add(Dropout(0.25))
```

```
model.add(Flatten())
```

```
model.add(Dense(units=1500,activation='relu'))
```

```
model.add(Dropout(0.4))
```

```
#Output Layer
model.add(Dense(units=10,activation='softmax'))
```

Compiling and Training Phase

```
!pip install tf_keras
import os
os.environ['TF_USE_LEGACY_KERAS'] = 'True'
```

```
model.compile(optimizer=tf.keras.optimizers.Adam(
    learning_rate=0.0001),loss='categorical_crossentropy',metrics=['accuracy'])
```

```
model.summary()
```

Training Model

```
training_history = model.fit(x=training_set,validation_data=validation_set,epochs=15)
```

Evaluating Model

```
#Training set Accuracy
train_loss, train_acc = model.evaluate(training_set)
print('Training accuracy:', train_acc)
```

```
#Validation set Accuracy
val_loss, val_acc = model.evaluate(validation_set)
print('Validation accuracy:', val_acc)
```

Saving Model

```
model.save('trained_plant_disease_model.keras')
```

```
training_history.history #Return Dictionary of history
```

```
#Recording History in json
import json
with open('training_hist.json','w') as f:
  json.dump(training_history.history,f)
```

```
print(training_history.history.keys())
```

Accuracy Visualization

```
epochs = [i for i in range(1,16)]
plt.plot(epochs,training_history.history['accuracy'],color='red',label='Training Accuracy')
plt.plot(epochs,training_history.history['val_accuracy'],color='blue',label='Validation Accuracy')
plt.xlabel('No. of Epochs')
plt.ylabel('Accuracy Result')
plt.title('Visualization of Accuracy Result')
plt.legend()
plt.show()
```

Some other metrics for model evaluation

```
class_name = validation_set.class_names
```

```
test_set = tf.keras.utils.image_dataset_from_directory(
    'valid',
    labels="inferred",
    label_mode="categorical",
    class_names=None,
    color_mode="rgb",
    batch_size=1,
    image_size=(128, 128),
    shuffle=False,
    seed=None,
    validation_split=None,
    subset=None,
    interpolation="bilinear",
    follow_links=False,
    crop_to_aspect_ratio=False
)
```

```

```

```
y_pred = model.predict(test_set)
y_pred.shape
predicted_categories = tf.argmax(y_pred, axis=1)
predicted_categories
```

```
true_categories = tf.concat([y for x, y in test_set], axis=0)
true_categories
Y_true = tf.argmax(true_categories, axis=1)
Y_true
```

```
Y_true
```

```
predicted_categories
```

```

```

```
from sklearn.metrics import confusion_matrix,classification_report
cm = confusion_matrix(Y_true,predicted_categories)
cm.shape
```

```
# Precision Recall Fscore
print(classification_report(Y_true,predicted_categories,target_names=class_name))
```

Confusion Matrix Visualization

```

```

```
plt.figure(figsize=(40, 40))
sns.heatmap(cm,annot=True,annot_kws={"size": 10})

plt.xlabel('Predicted Class',fontsize = 20)
plt.ylabel('Actual Class',fontsize = 20)
plt.title('Plant Disease Prediction Confusion Matrix',fontsize = 25)
plt.show()
```

Git link: https://github.com/Spurthi7904/Detecting-Tomato-plant-diseases-through-leaf-image-analysis.git

Video link: https://youtu.be/nVNdFd3mo6Y?feature=shared