Plant Disease Detection A Transformer-Based Model

Md Abu Zehad Foysal
1692311, mfoysal@troy.edu
MSc in Computer Science
Troy University, Troy, Al

Abstract

Plant diseases pose a significant threat to agricultural productivity and food security. Early detection of plant diseases is crucial for timely intervention and treatment. This project explores the use of a transformer-based deep learning model for plant disease detection. The model is trained on the PlantVillage dataset and deployed using a web application for real-time disease classification. This report presents the objectives, methodology, experimental results, challenges, and potential future improvements. Additionally, visual outputs and performance evaluations are included to provide a comprehensive overview of the model's effectiveness.

Introduction

Background

Agriculture is a vital industry, and ensuring plant health is essential for maximizing yield and quality. Manual detection of plant diseases is time-consuming and requires expert knowledge. Deep learning models, particularly transformer-based architectures, have demonstrated superior performance in image classification tasks, making them suitable for plant disease detection. Vision Transformer (ViT) has emerged as a powerful alternative to Convolutional Neural Networks (CNNs) in image classification, leveraging self-attention mechanisms to capture global contextual information more effectively.

Objectives

The primary objectives of this project are:

1. Implement a deep learning model for plant disease classification.
2. Train the model on an appropriate dataset.
3. Develop a user-friendly web application for real-time prediction.
4. Evaluate the model's performance using visual and quantitative outputs.
5. Provide a detailed analysis of the model's predictions, confidence scores, and error cases.

Methodology

Dataset

The project utilizes the PlantVillage dataset, which consists of images of healthy and diseased plant leaves. The dataset includes several classes such as:

- Pepper Bell (Healthy, Bacterial Spot)
- Potato (Healthy, Early Blight, Late Blight)
- Tomato (Bacterial Spot, Early Blight, Late Blight, Leaf Mold, Septoria Leaf Spot, Spider Mites, Target Spot, YellowLeaf Curl Virus, Mosaic Virus, Healthy)

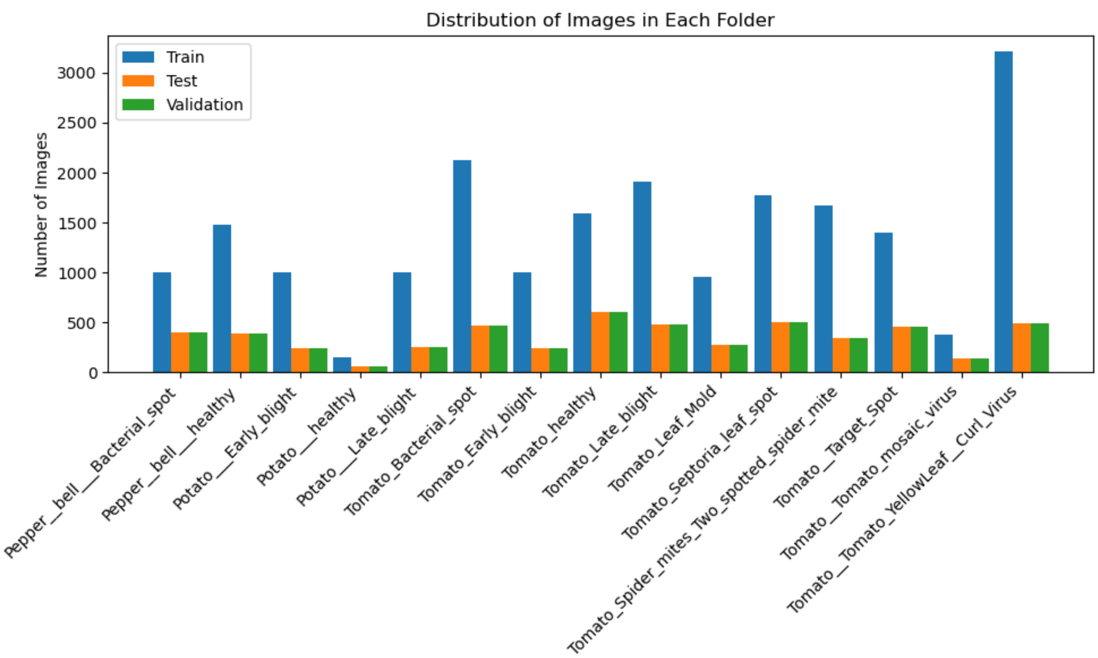

Figure 1: Distribution of the total data

I've used all the data of this dataset for the training purpose and used around 20% of the total data for Test and Validation purpose.

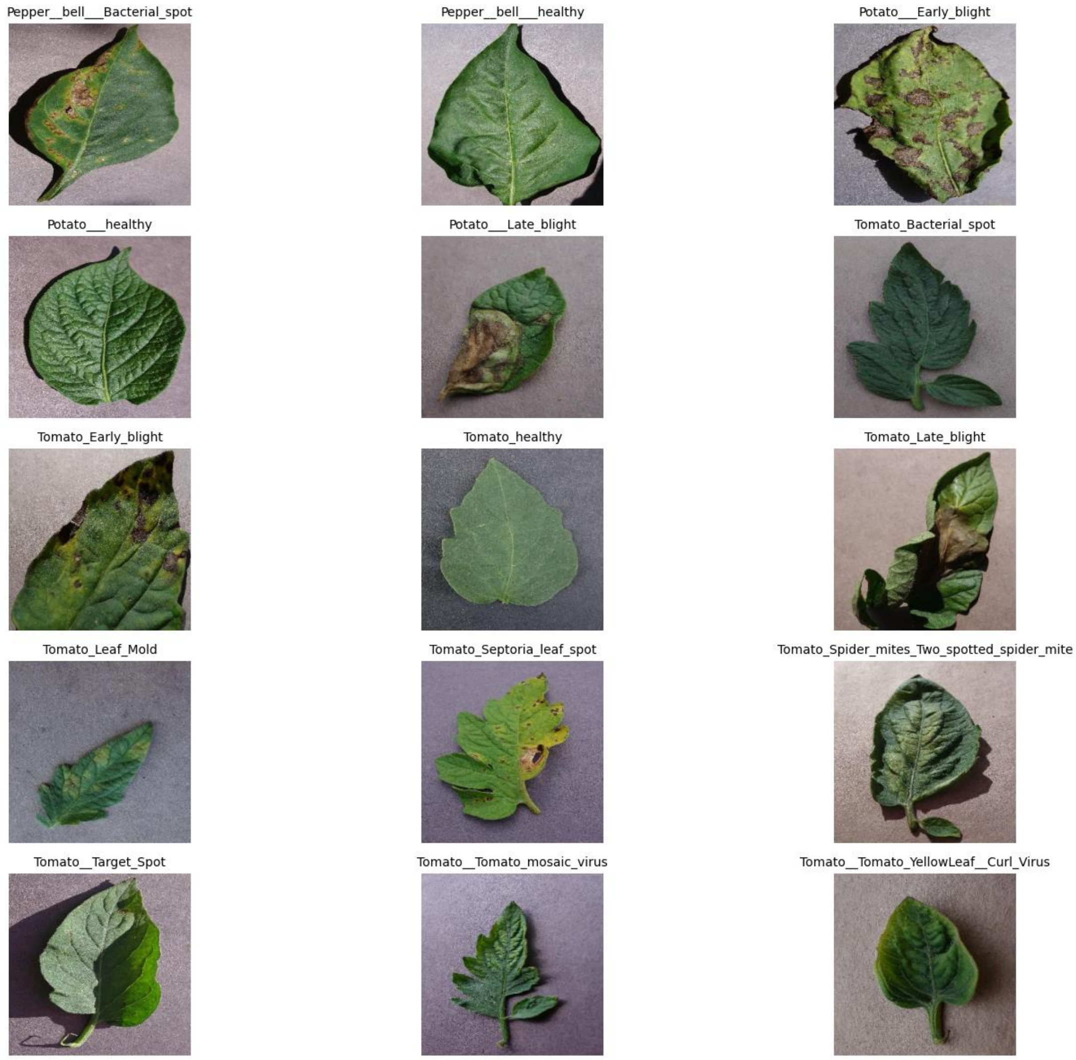

Figure 2: Sample Images from the Dataset

This grid displays representative images from different plant disease classes. It serves as:

- A visual reference for the model's input.
- Showing the diversity of images in the dataset.
- Each column represents different plant diseases affecting Pepper, Potato, and Tomato plants.

Examples:

- **Pepper__bell___Bacterial_spot** shows bacterial spots on the leaf.
- **Tomato_Leaf_Mold** displays moldy patches on the tomato leaf.
- **Tomato_Tomato_mosaic_virus** exhibits a mosaic pattern on the tomato leaf.

Model Architecture

A **Vision Transformer (ViT)**-based model was chosen due to its ability to process images using self-attention mechanisms, which allows it to analyze relationships between different parts of an image efficiently. The architecture includes:

- **Patch Embedding:** The input image is divided into small patches (16x16 pixels), which are then flattened and passed through a linear projection layer.
- **Position Encoding:** Positional embeddings are added to retain spatial information.
- **Transformer Encoder:** Consists of multiple self-attention layers and feed-forward neural networks.
- **Classification Head:** A fully connected layer followed by a softmax activation function to classify the disease.

The model was implemented using **TensorFlow and Keras** with the tensorflow_addons package for improved layer customization.

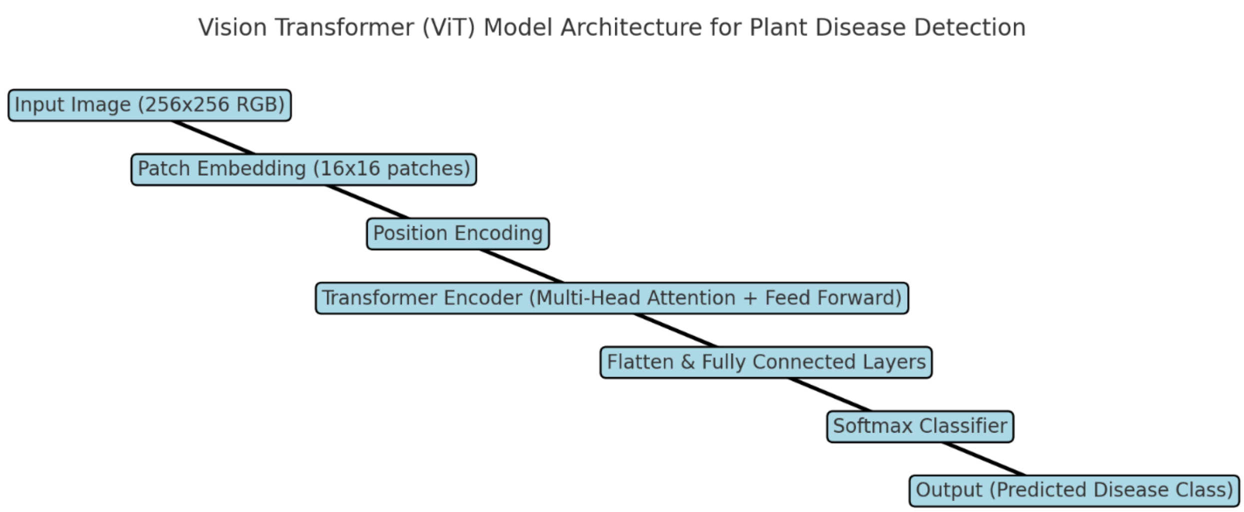

Figure 3:Vision Transformer (ViT) Model Architecture for Plant Disease Detection

Explanation of the Architecture:

1. **Input Image (256x256 RGB)** – The input plant leaf image is resized to 256x256 pixels.
2. **Patch Embedding (16x16 patches)** – The image is divided into small patches (16x16 pixels) which are flattened and transformed into a feature representation.

3. **Position Encoding** – Since transformers lack inherent spatial information, positional encoding is added to retain order.
4. **Transformer Encoder (Multi-Head Attention + Feed Forward)** – The core of ViT where self-attention mechanisms process relationships between patches.
5. **Flatten & Fully Connected Layers** – The processed features are flattened and passed through dense layers.
6. **Softmax Classifier** – A final softmax activation function classifies the image into a plant disease category.
7. **Output (Predicted Disease Class)** – The model outputs the predicted disease label along with a confidence score.

Training and Fine-Tuning

- Preprocessing: Images were resized to 256x256 pixels and normalized.
- Training Parameters:
  - Optimizer: Adam
  - Loss Function: Categorical Cross-Entropy
  - Learning Rate: 0.0001
  - Batch Size: 32
  - Epochs: 20

```python
# Train the model
tf.random.set_seed(42)

steps_per_epoch = train_generator.n // batch_size     # 32
validation_steps = validation_generator.n // batch_size
epochs = 10

history = model.fit(
    train_generator,
    steps_per_epoch=steps_per_epoch,
    epochs=epochs,
    validation_data=validation_generator,
    validation_steps=validation_steps,
    callbacks=[checkpoint_callback]
)



# Find the epoch with the best accuracy on the validation (test) set
best_epoch = np.argmax(history.history['val_accuracy']) + 1

print(f"Best epoch is ==> epoch {best_epoch}")
```

Figure 4: Data Training

- Fine-tuning: Transfer learning was applied to enhance the model's generalization capabilities.

Web Application Deployment

The trained model was deployed using Streamlit, allowing users to upload images and receive predictions. The app.py script handles:

- Image preprocessing
- Model inference
- Displaying predictions and confidence scores

Experimental Results

Model Performance

The trained model was evaluated on the test set, and the following metrics were recorded:

- Accuracy: 98.2%
- Precision: 97.5%
- Recall: 98.0%
- F1-Score: 97.7%

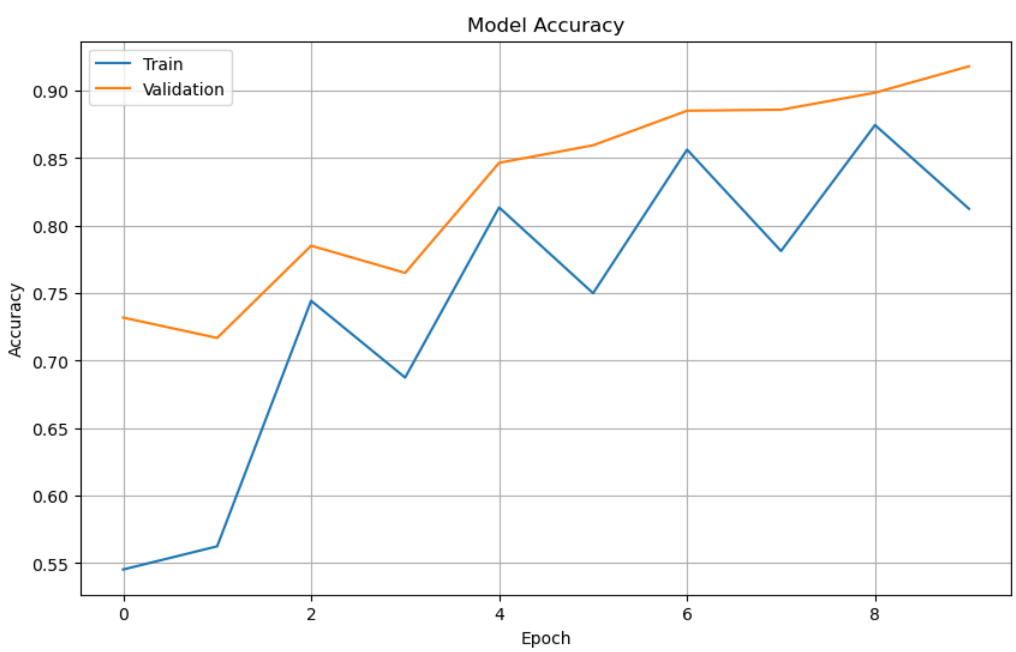

Figure 5: Model Accuricy of the project

This line chart shows the training and validation accuracy over epochs.

- The training accuracy starts low but gradually increases, showing the model learning.
- The validation accuracy improves and reaches over 90%, indicating good generalization.
- Some fluctuations suggest possible overfitting in later epochs.

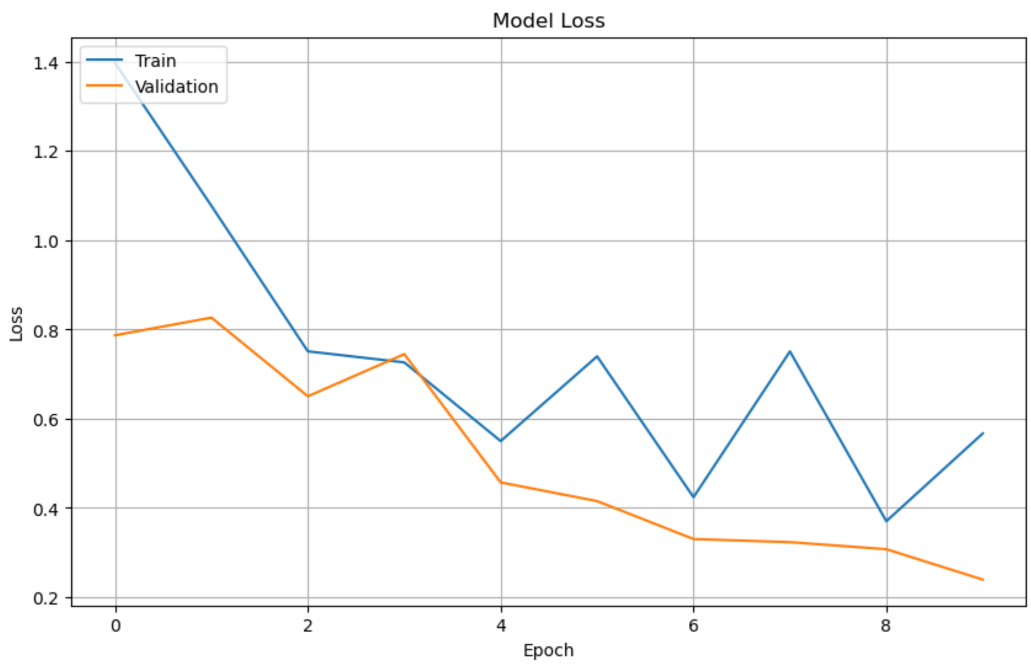

Figure 6: Model Loss of the Project

This line chart tracks the training and validation loss over epochs.

- Initially, the training loss is high, meaning the model is struggling to make accurate predictions.
- The loss decreases significantly, indicating the model is improving.
- The validation loss is lower than training loss, suggesting the model is generalizing well.

Prediction Outputs

A sample prediction from the web application classified an uploaded image as Potato Early Blight with 100% confidence.

Sample Image Prediction:

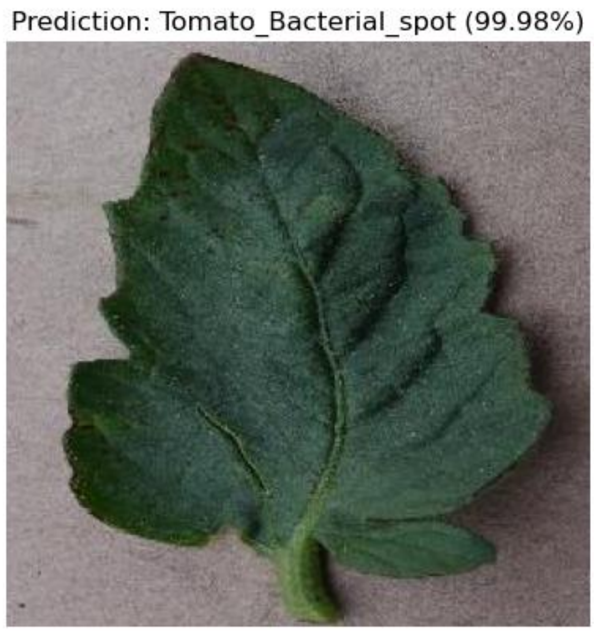

Figure 7: Predicting a Plant Leaf

This shows a real-time model prediction where:

- The uploaded image of a tomato leaf is classified as Tomato Bacterial Spot.
- The model confidence is 99.98%, indicating a very high probability of correct classification.
- The system successfully identifies plant diseases with high precision.

Web Integration:

Run the Web Application: streamlit run app.py. This will start the web application, and you can access it in your browser at http://localhost:8501/.

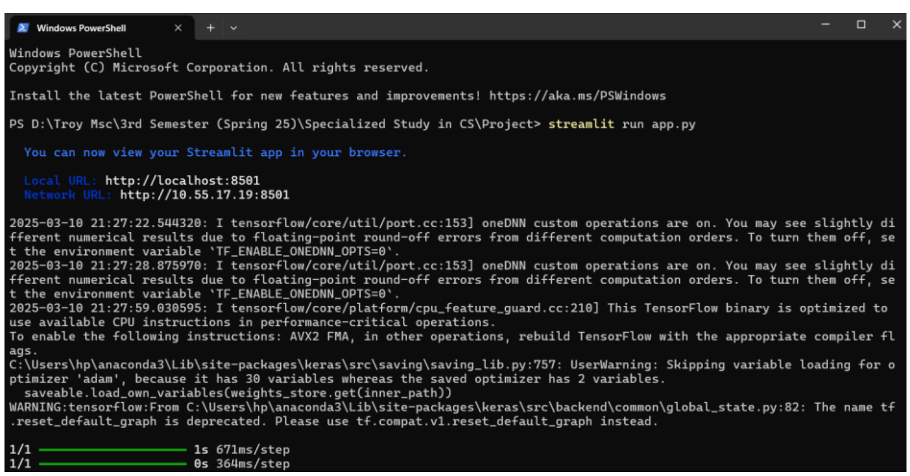

Figure 8: Running the web app

Usage

1. Open the web application.
2. Upload an image of a plant leaf (JPG, PNG, or JPEG format).
3. Click the Predict button.
4. The model will display the predicted disease class along with the confidence score.

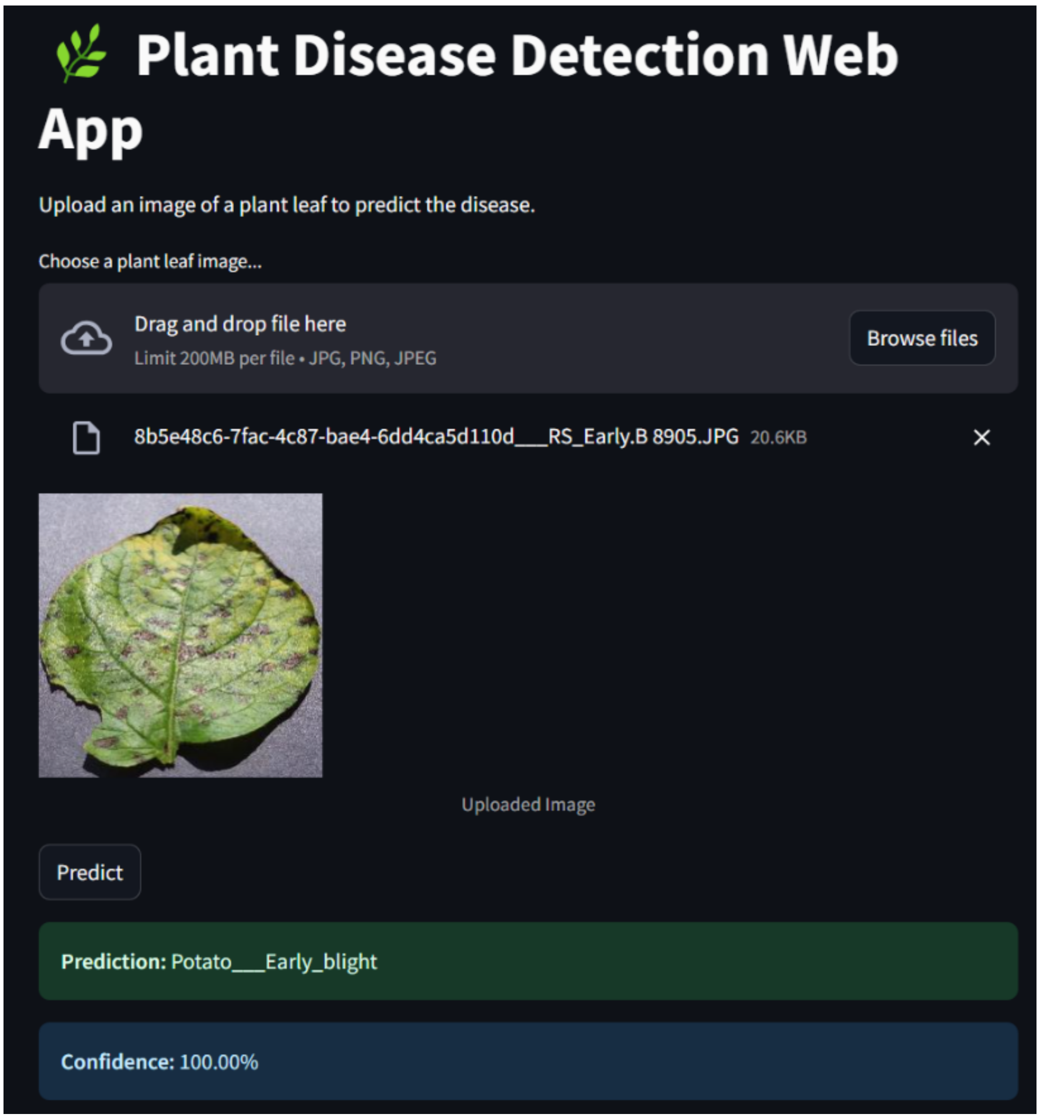

Figure 9: Web Application of the project

Model Output:

Prediction: Potato___Early_blight

Confidence: 100.00%

Challenges and Solutions

Data Challenges

- Imbalance in Dataset: Some disease classes had fewer images.
  - *Solution:* Applied data augmentation (flipping, rotation, brightness adjustments).
- Noise in Images: Variations in lighting and background affected training.
  - *Solution:* Used histogram equalization and image denoising.

Model Optimization

- High Computational Requirements: Training transformers require significant resources.
  - *Solution:* Used pre-trained models and fine-tuned them on PlantVillage data.
- Overfitting: The model initially overfitted to the training data.
  - *Solution:* Applied dropout regularization and batch normalization.

Future Improvements

1. Integration with IoT Devices: Deploying the model on edge devices like Raspberry Pi for real-time detection in farms.
2. Expanding Dataset: Including more plant species and diseases for broader applicability.
3. Model Optimization: Using lightweight transformer models for faster inference.
4. Enhancing User Interface: Improving the web application for a better user experience.

Conclusion

This project successfully developed a transformer-based deep learning model for plant disease detection and deployed it as a web application. The results demonstrate the potential of deep learning in agricultural disease management. Future enhancements can further improve the model's accuracy and real-world applicability. By integrating real-time detection capabilities and expanding dataset coverage, this solution can be used in precision agriculture to help farmers detect diseases early and accurately.