Group43-Final-Project: Plant Disease Detection Web App

A full-stack deep learning web application designed to detect plant leaf diseases specifically for Pepper, Tomato, and Potato plants using the comprehensive PlantVillage dataset.

This project leverages cutting-edge technologies to deliver accurate and actionable insights:

- Convolutional Neural Networks (CNN) with Keras / TensorFlow 2.10.0
- Flask for a dynamic web interface
- The extensive PlantVillage dataset for robust training
- Developed using Python 3.10.11

🚀 Key Features

Accurate Disease Detection

Upload a plant leaf image and instantly receive the predicted disease class, leveraging a highly trained CNN model.

Comprehensive Disease Information

Beyond detection, the app provides a detailed description of the disease, its symptoms, underlying causes, and severity.

Actionable Treatment Recommendations

Receive practical recommendations, including preventive measures, organic solutions, and chemical treatments tailored to the detected disease.

Robust Dataset & User-Friendly Interface

Trained on the extensive PlantVillage dataset and accessible via an intuitive Flask web interface with seamless file upload capabilities.

📂 Project Structure Overview

The project is meticulously organized to ensure clarity and maintainability. Below is a high-level overview of the main directories and files:

```
group43-Final-Project/
  │
  ├──── app.py # Main Flask app entrypoint
  ├──── routes.py # Flask routes and views
  ├──── ml_model.py # CNN model class and training logic
  ├──── train_model.py # Command-line training script
  ├──── evaluate_model.py # Command-line evaluation script
  ├──── disease_data.py # Disease info and treatment database
  ├──── requirements.txt # Python package requirements
  ├──── .env # Environment variables
  ├──── .gitignore
  │
  ├──── models/ # Saved Keras models (.keras) and class_names.json
  ├──── dataset/ # PlantVillage dataset (train/test splits)
  │   ├──── Pepper__bell___Bacterial_spot/
  │   ├──── Pepper__bell___healthy/
  │   ├──── Potato___Early_blight/
  │   └──── ...
  │   └──── test/ # For test/evaluation (At least 10 images each folder)
  │   ├──── Pepper__bell___Bacterial_spot/
  │   ├──── Pepper__bell___healthy/
  │   ├──── Potato___Early_blight/
  │   └──── ...
  │
  ├──── uploads/ # Uploaded images (temporary)
  ├──── static/ # Static frontend files (CSS, JS)
  │   ├──── css/
  │   └──── js/
  ├──── templates/ # HTML templates
  │   ├──── index.html
  │   ├──── results.html
  │   ├──── 404.html
  │   └──── 500.html
  │
  └──── logs/ # TensorBoard logs and training logs
```

⚙️ Setup Instructions

1

Clone the Repository

```
git clone https://github.com/MichaelWaruiru/Group43-Final-Project.git
cd Group43-Final-Project
```

Begin by cloning the project repository from GitHub to your local machine and navigate into the project directory.

2

Install Python Dependencies

```
pip install -r requirements.txt
```

It is highly recommended to use a virtual environment to manage dependencies. Install all required Python packages listed in `requirements.txt`.

3

Create Environment Variables

```
SESSION_SECRET=your_super_secret_key_here
```

Create a `.env` file in the project's root directory and define your Flask session secret for secure application operation.

🏋️ Training & Evaluating the Model

Training the Model

Ensure the PlantVillage dataset is correctly structured within the `dataset/` folder:

```
dataset/
├── Pepper__bell___Bacterial_spot/
├── Pepper__bell___healthy/
└── Potato___Early_blight/
└── ...
```

Then, initiate the training process:

```
python train_model.py
```

This script will train the CNN or MobileNetV2 transfer model, save the best model and class names, and plot the training history.

Evaluating the Model

For model evaluation, ensure your `dataset/test/` directory contains all trained classes:

```
dataset/test/
├── Pepper__bell___Bacterial_spot/
├── Pepper__bell___healthy/
└── ...
```

Then execute the evaluation script:

```
python evaluate_model.py
```

This will display test accuracy and a detailed classification report. A minimal set of 5-10 images per class is sufficient for testing.

🌐 Running the Web App

1

Start the Flask Application

```
python app.py
```

Launch the Flask web application by running the main entry point script.

2

Access in Browser

Open your web browser and navigate to the local server address:

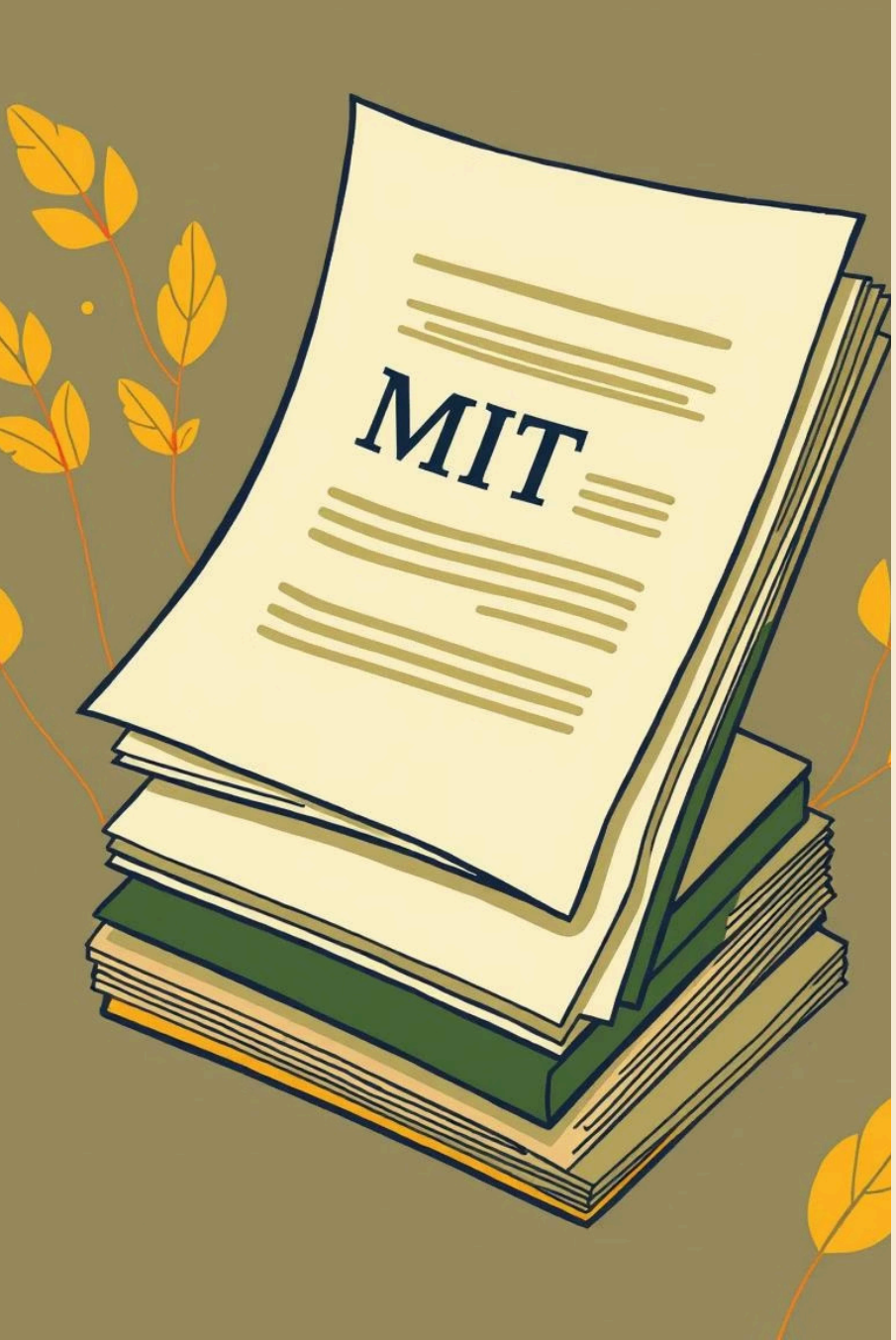
http://127.0.0.1:5000/

3

Using the Web Interface

On the homepage, upload a plant leaf image (supported formats: png, jpg, jpeg, gif, bmp, webp). The app will display the disease class, confidence score, detailed description, symptoms, causes, severity, and comprehensive treatment recommendations.

⚠ **Notes on Dataset Splits:** Your test set **must** contain all classes that the model was trained on to ensure the classification report functions correctly. Even 5-10 images per class are sufficient for evaluation, preventing errors or warnings.



📜 License Information

This project is distributed under the **MIT License**.

This license permits free use, modification, and distribution, with proper attribution. For more details, refer to the `LICENSE` file in the repository.

🙏 Special Acknowledgements

PlantVillage Dataset (Kaggle)

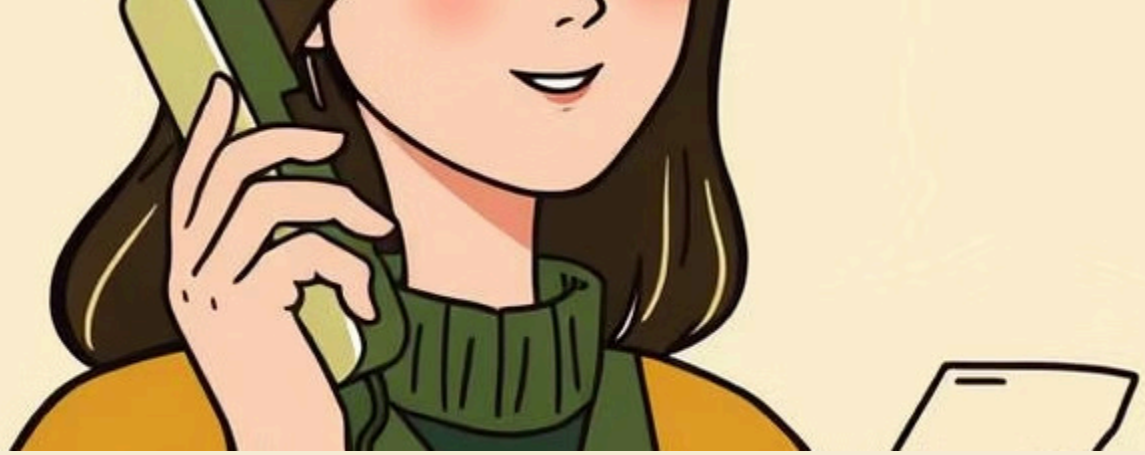
Our profound gratitude to the creators and maintainers of the PlantVillage dataset on Kaggle, which was indispensable for training our robust deep learning model.

TensorFlow / Keras

We extend our appreciation to the TensorFlow and Keras teams for providing the powerful and flexible deep learning frameworks that underpin our model development and deployment.

Flask Framework

Many thanks to the Flask development team for their lightweight and versatile web framework, which enabled the creation of an intuitive and responsive user interface for our application.



📞 Contact & Support

For any questions, issues, or collaborative inquiries regarding the Plant Disease Detection Web App, please feel free to reach out.

The most efficient way to get support or report a problem is to **open an issue directly on the GitHub repository**.

Alternatively, you can contact the project maintainers via the contact information provided in the repository's README file.