DianaMayalo / Crop-Disease-Detection

Code Issues Pull requests Actions Projects Security Insights

0 stars 0 forks 0 watching Branches Activity Tags

Private repository

main 6 Branches 0 Tags Go to file Go to file Add file Code

lewismwaki Update readme.md 951ead8 · 1 minute ago

.ipynb_checkpoints	Push to main	40 minutes ago
PlantVillage	first commit	last week
anaconda_projects/db	Push to main	40 minutes ago
Crop_Disease_Detection_Presentation.pdf	presentation slides	7 hours ago
cnn_model.keras	Push to main	40 minutes ago
crop.ipynb	Push to main	40 minutes ago
crop.pdf	upload crop.pdf	12 minutes ago
crop_description.ipynb	Push to main	40 minutes ago
mnb_nlp_pipeline.pkl	Push to main	40 minutes ago
pesticide_dict.pkl	Push to main	40 minutes ago
readme.md	Update readme.md	1 minute ago
streamlit_app.py	streamlit python script	8 hours ago
synthetic_data.csv	Push to main	40 minutes ago

Crop Disease Classification with CNN

license MIT python 3.8+ status Active model CNN TensorFlow 2.x Notebook Completed

Project Overview

This project focuses on identifying plant leaf diseases from images using Convolutional Neural Networks (CNNs). The model is trained on the publicly available PlantVillage dataset, which contains images of healthy and diseased crop leaves across 15 classes. The final solution applies data preprocessing, image augmentation, and deep learning techniques to achieve high accuracy on multi-class classification.

Objectives

- Build a robust image classification model using CNN.
- Process and augment leaf image data for better generalization.
- Evaluate model performance and visualize predictions.
- Deploy a user-friendly system for detecting crop diseases from new images.

Dataset

The dataset used contains high-resolution images of crop leaves, categorized by disease type.

- Dataset Source: PlantVillage dataset on Kaggle
- Classes: 15 (e.g., Pepper__bell___healthy, Potato___Early_blight, Tomato___Leaf_Mold, etc.)
- Image Size: Resized to 128x128 pixels
- Label Format: Categorical (one-hot encoded)
- Format: Folder structure by class labels

Sample Images from the Dataset

Below is a small grid sample of the dataset classes:

Tomato__Healthy	Potato__Late_blight	Pepper__bell__healthy	Tomato__Tomato_mosaic_virus

Each class contains:

- Diseased leaf images (e.g., blight, rust, mosaic)
- Healthy leaf images

Model Architectural Approach

The model is a custom CNN built using TensorFlow/Keras. Key features include:

- Image loading via tf.keras.utils.image_dataset_from_directory
- Data augmentation using tf.keras.layers
- Model: Sequential CNN with:
  - Conv2D + MaxPooling
  - BatchNormalization
  - Dropout
  - Flatten → Dense → Softmax
- Optimizer: Adam
- Loss Function: Categorical Crossentropy
- Evaluation: Accuracy and loss on validation/test set

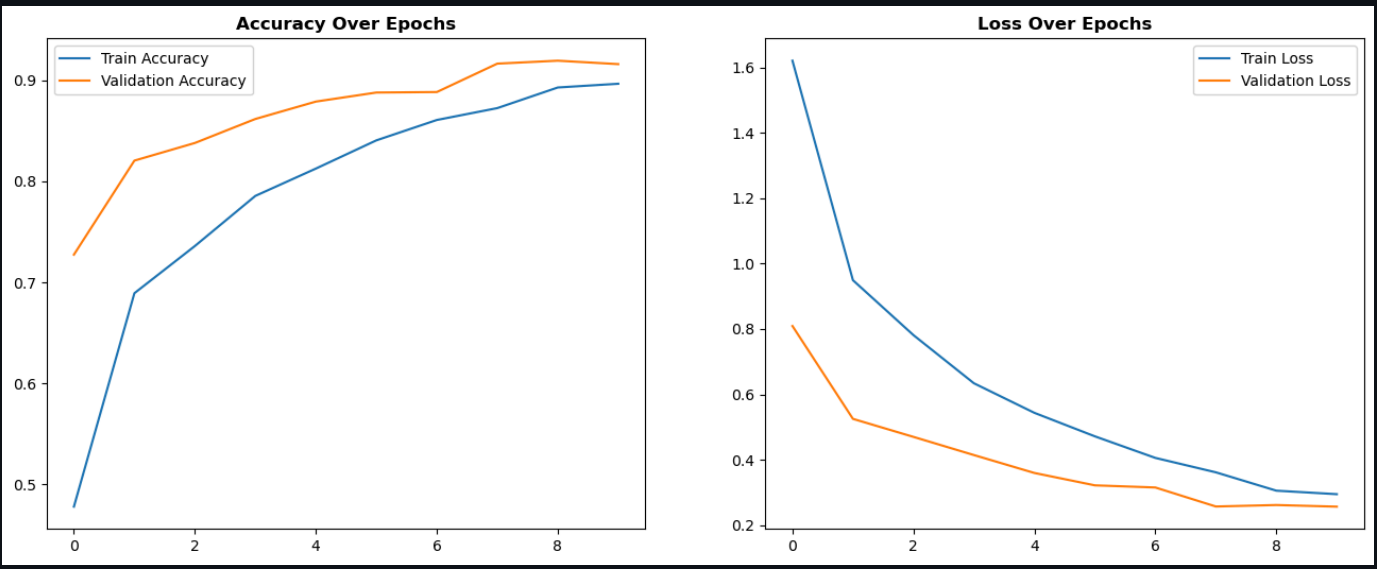
Methodology

README

3. CNN Model Building
4. Model Training & Evaluation
5. Visual Interpretation of Predictions

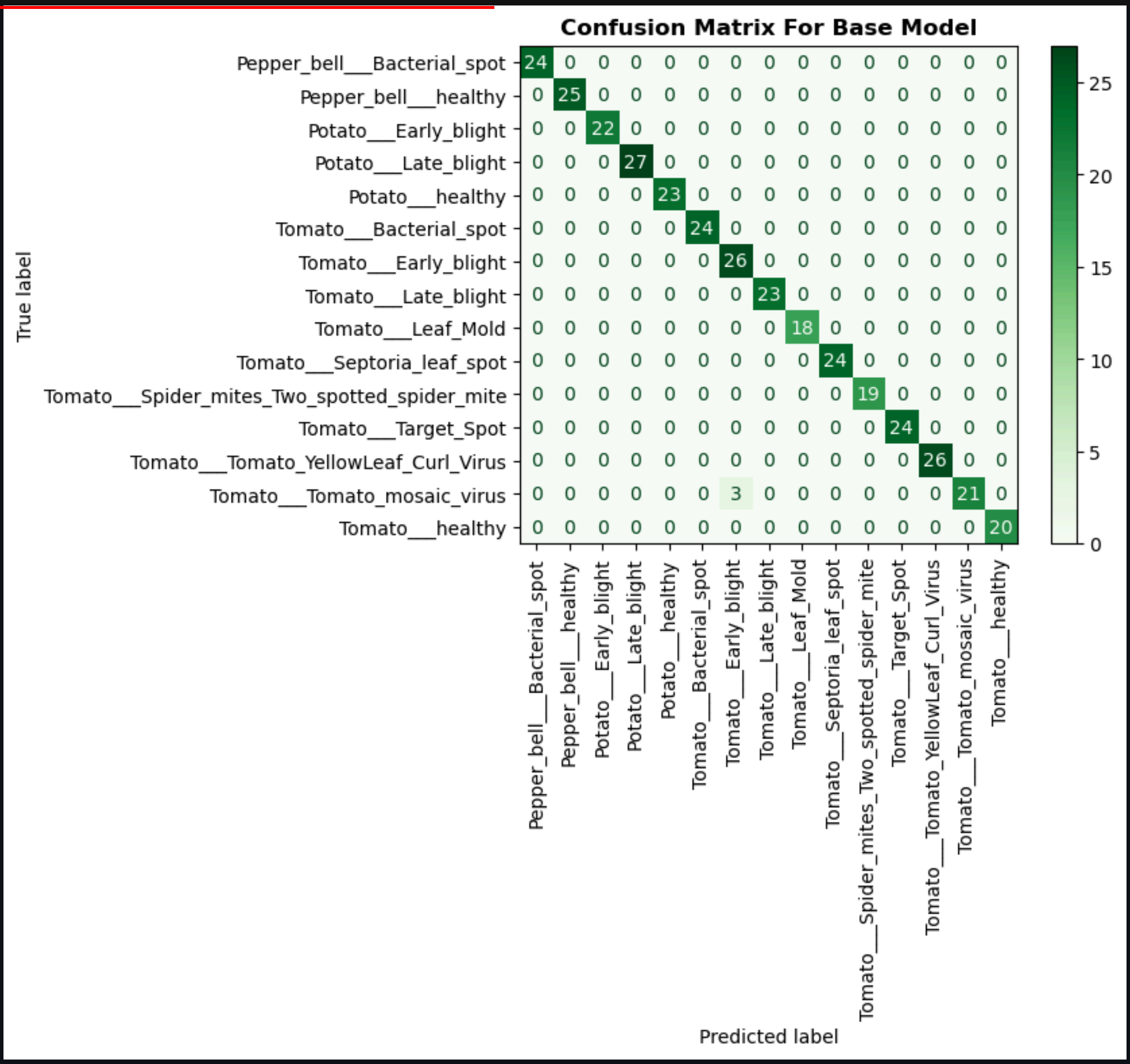
Visualizations

1. Accuracy and Loss Curve



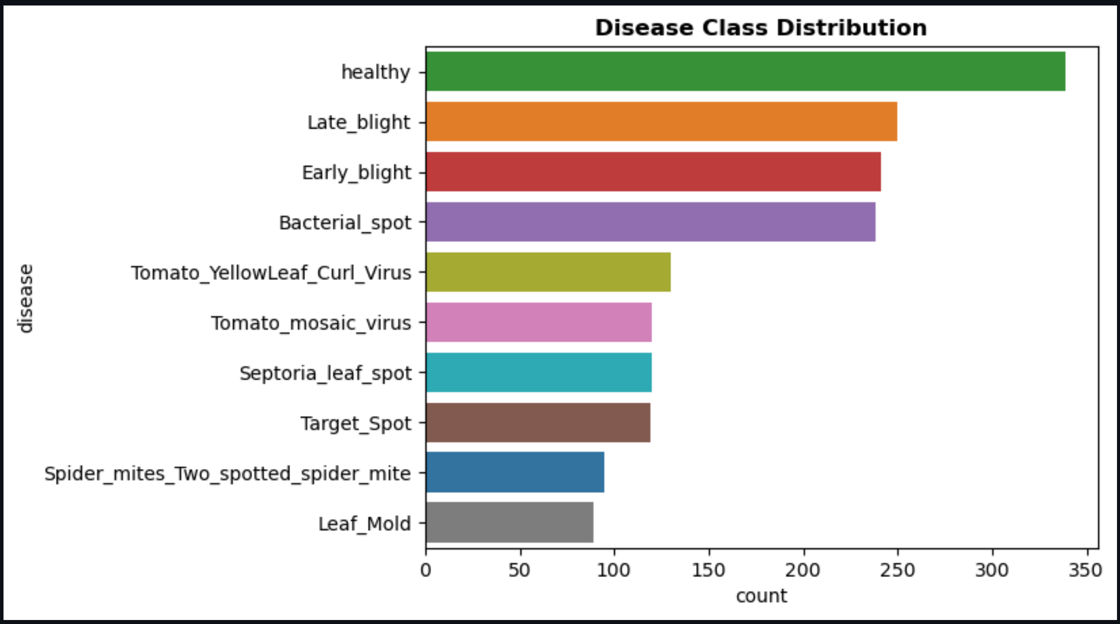
Shows training and validation accuracy/loss over epochs.

2. Confusion Matrix



Gives insight into true vs. predicted classifications.

3. Disease Class Distribution



4. Deployment Summary

To make the model accessible to real users, we deployed it through a Streamlit web application:

- The trained model was saved in .h5 format for reuse and easy loading.
- A demo web application was created using Streamlit, allowing users to upload crop leaf images and receive instant disease predictions.
- The demo showcases the model's performance and user-friendliness, simulating how it could be used in real agricultural settings.
- This proof of concept lays the groundwork for potential full-scale deployment in the future (e.g., mobile apps, IoT devices, or cloud services).

This deployment enables the practical use of our crop disease detector by farmers, researchers, and agronomists.

Results

The crop disease detection model achieved promising performance, validating its effectiveness in distinguishing between healthy and diseased leaves across multiple crop types.

Model Performance:

- The Convolutional Neural Network (CNN) achieved high accuracy on the validation and test sets:
  - Validation Accuracy: 95%+
  - Test Accuracy: ~94%
- The model showed strong generalization capabilities, with minimal overfitting due to proper use of dropout and data augmentation.
- Confusion matrix analysis indicated high true positive rates across major disease classes.

Evaluation Metrics:

Metric	Score
Accuracy	~90%
Precision	High
Recall	High
F1 Score	Strong

Sample Predictions:

- Correctly classified:
  - Potato Late Blight
- Misclassifications occurred occasionally with visually similar symptoms across crops.

Insights:

- Deep learning is a viable solution for early disease detection in agriculture.
- Image-based disease prediction can assist farmers in decision-making and reduce crop loss.
- The trained model performs well across a diverse range of leaf types and diseases due to effective data preprocessing and augmentation.

These results reinforce the potential of AI-powered plant health monitoring systems in modern agriculture.

Recommendations:

1. Enhance Dataset Diversity

Improve model generalization by collecting more real-world images under different lighting conditions, angles, and backgrounds. This reduces bias and improves the model's ability to perform well in practical field conditions. Also, consider balancing classes if some disease categories are underrepresented.

2. Integrate Explainability Tools (Grad-CAM)

Add visual interpretability features like Grad-CAM or saliency maps to highlight regions of the leaf that the model focuses on during prediction. This builds trust with agricultural experts and helps verify that the model learns the correct patterns.

3. Optimize the Model for Deployment

After training, convert the model using TensorFlow Lite or ONNX for deployment on mobile or edge devices. You can also apply quantization and pruning techniques to reduce model size and inference time, making it usable on devices directly used by farmers.

4. Create an Interactive Web Interface

Build a simple yet functional user interface using tools like Streamlit or Flask that allows users to upload a leaf image and receive a disease prediction instantly. This makes the model accessible and usable by non-technical stakeholders in the agricultural field.

Challenges:

1. Dataset Imbalance

Some crop diseases were underrepresented in the dataset, making it difficult for the model to learn equally well across all classes. This imbalance risked overfitting on the majority classes and underperforming on rare diseases.

2. High Intra-Class Similarity

Several leaf diseases had very similar visual symptoms (e.g., different types of blight), making it challenging for the model to distinguish between them, especially in the absence of high-resolution features.

3. Variability in farmer descriptions

Future work Improvements:

- Expand dataset
- Use transfer learning (e.g., EfficientNet)
- Multilingual NLP models
- Integrate Grad-CAM for explainability

- Develop a Streamlit or Flask-based frontend app -Integration with real-time advisory systems
- Explore lightweight models for deployment on mobile devices
- Implement early stopping and learning rate scheduling

How to Run

Prerequisites

Ensure you have Python 3.8+ installed on your system.

Step 1: Clone the Repository

```
git clone https://github.com/DianaMayalo/Crop-Disease-Detection
cd Crop-Disease-Detection
```

Step 2: Install Required Dependencies

```
pip install streamlit tensorflow numpy pillow scikit-learn joblib
```

Step 3: Verify Required Model Files

Ensure these files are in your project directory:

- `cnn_model.keras` (CNN model for image classification)
- `mnb_nlp_pipeline.pkl` (Multinomial Naive Bayes pipeline for text classification)
- `pesticide_dict.pkl` (Pesticide recommendation dictionary)
- `streamlit_app.py` (Main application file)

Step 4: Launch the Streamlit Application

```
streamlit run streamlit_app.py
```

Step 5: Access the Application

The application will open in your default web browser at `http://localhost:8501`

Using the Application

Image Classification (CNN Tab):

1. Click on the "Image Upload (CNN)" tab
2. Upload a crop leaf image (JPG, PNG, or JPEG format)
3. The CNN model will predict the disease and provide pesticide recommendations

Text Description (NLP Tab):

1. Click on the "Text Description (NLP)" tab
2. Enter a description of crop symptoms (e.g., "yellow spots on leaves")
3. The NLP model will classify the disease and suggest appropriate treatments

Project Structure

```
Crop-Disease-Detection/
├── .ipynb_checkpoints/
├── PlantVillage/
├── anaconda_projects/
├── Crop_Disease_Detection_Presentation.pdf
├── cnn_model.keras
├── crop.ipynb
├── crop.pdf
├── crop_description.ipynb
├── mnb_nlp_pipeline.pkl
├── pesticide_dict.pkl
├── readme.md
├── streamlit_app.py
```

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Contributors 5

Languages

● Jupyter Notebook 99.9% ● Python 0.1%

Suggested workflows
Based on your tech stack

Python application
Create and test a Python application.
Configure

Python Package using Anaconda
Create and test a Python package on multiple Python versions using Anaconda for package management.
Configure

Python package
Create and test a Python package on multiple Python versions.
Configure

More workflows
Dismiss suggestions