Plant Disease Classification using KNN and CNN Features

1. Abstract

This project details the development of a K-Nearest Neighbors (KNN) based system for classifying plant diseases from leaf images. To overcome the limitations of KNN with high-dimensional image data, features were extracted using a pre-trained MobileNetV2 Convolutional Neural Network (CNN), followed by dimensionality reduction via Principal Component Analysis (PCA). Trained on the PlantVillage dataset (15 classes, 20,638 images), and with hyperparameters optimized using GridSearchCV, the system achieved a test accuracy of 88.57%. A Python script (app.py) demonstrates the system's predictive utility. The report outlines the methodology, results, and discusses potential enhancements, including the integration of environmental data.

2. Introduction

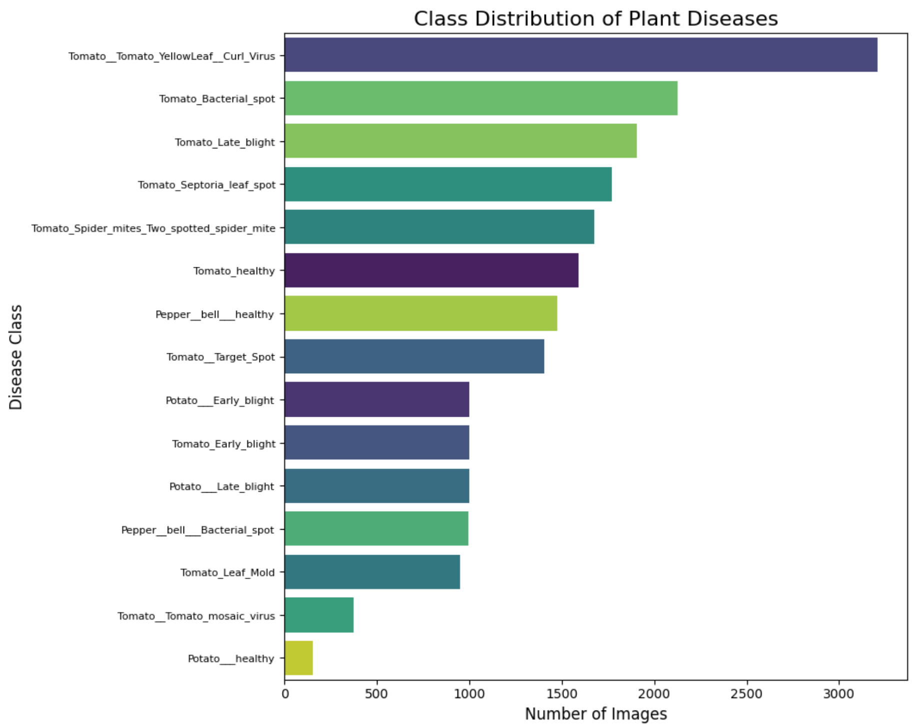
Rapid and accurate plant disease diagnosis is critical for mitigating crop losses in agriculture. This project aimed to develop an automated classification system using leaf imagery. The core challenge was to adapt the K-Nearest Neighbors (KNN) algorithm, traditionally unsuited for raw image data, by employing robust feature engineering techniques. The "PlantVillage" dataset (https://www.kaggle.com/datasets/emmarex/plantdisease) provided the image data for 15 distinct plant-disease categories. The project also considered the potential, though un-implementable due to data constraints, of incorporating environmental features like temperature and humidity.

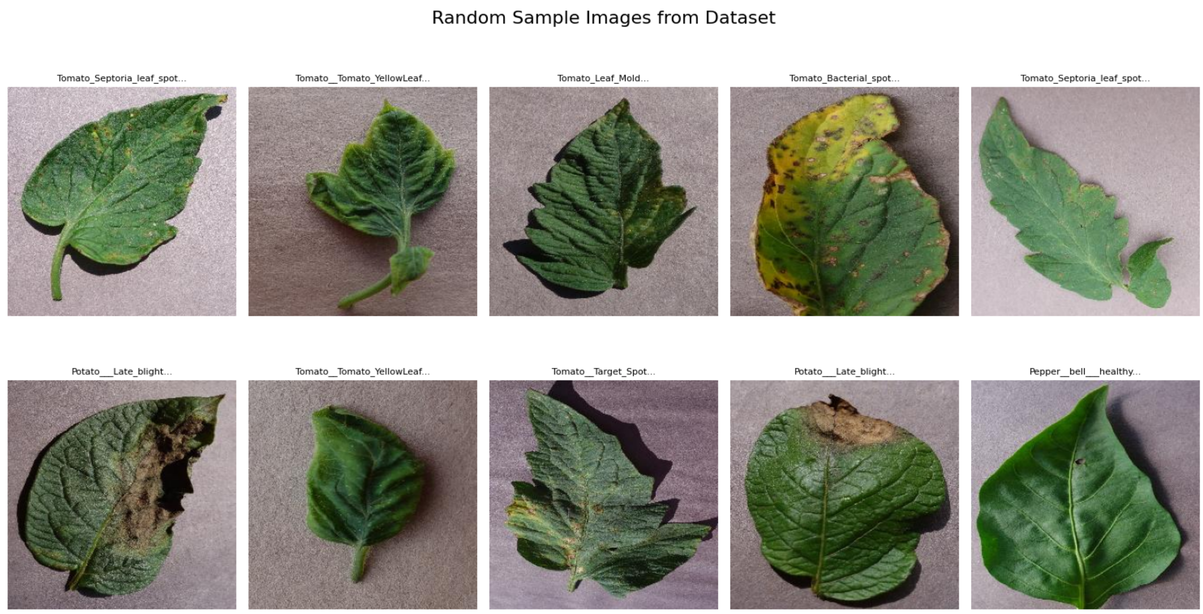
3. Methodology

The project followed a systematic machine learning pipeline implemented in Python using Scikit-learn, TensorFlow/Keras, Pandas, NumPy, Matplotlib, and Seaborn.

3.1. Data Preparation and Exploratory Data Analysis (EDA)

Dataset images and corresponding labels (derived from folder names) were loaded into a Pandas DataFrame. EDA (Figure 1: Class Distribution) revealed significant class imbalance, underscoring the need for stratified sampling and careful, per-class metric evaluation. Images were standardized to 224x224 pixels, the input requirement for the MobileNetV2 CNN. Sample images were visually inspected (Figure 2).

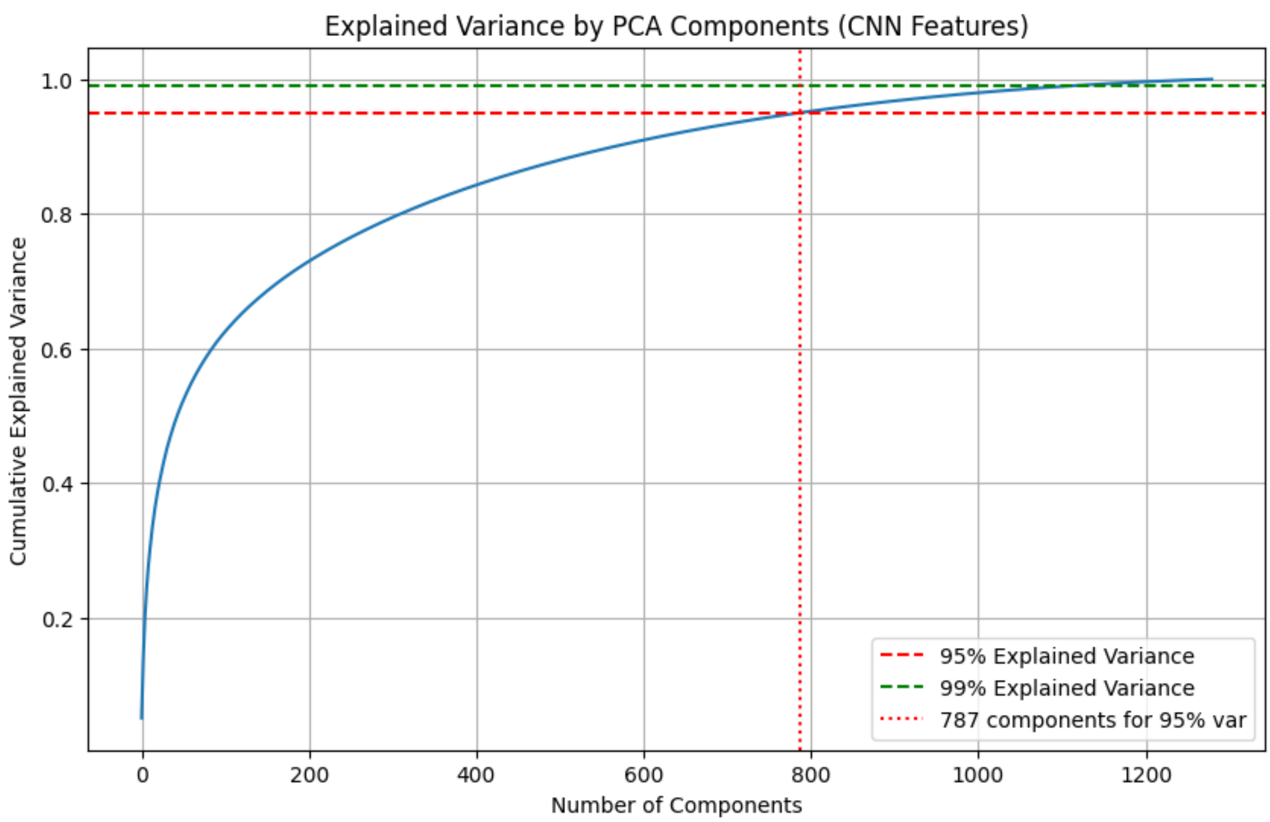




3.2. Feature Engineering

A two-stage feature engineering process was implemented:

- **CNN Feature Extraction:** MobileNetV2, pre-trained on ImageNet, was utilized as a feature extractor (include_top=False). A GlobalAveragePooling2D layer was appended to its output, yielding a 1280-dimensional feature vector for each image. Images were preprocessed using MobileNetV2's specific preprocess_input function. This step transforms raw pixel data into a more semantically rich representation.
- **Principal Component Analysis (PCA):** The 1280-D CNN features were first scaled using StandardScaler (fitted on training data only). PCA was then applied for dimensionality reduction. Analysis of the cumulative explained variance (Figure 3: PCA Variance Plot) showed that 787 components captured 94.88% of the variance. This reduced dimensionality aims to improve KNN's efficiency and potentially its generalization by removing less informative or noisy components.



3.3. Model Training and Optimization

1. **Label Encoding:** Categorical disease names were converted to numerical labels using LabelEncoder.
2. **Train-Test Split:** The dataset was divided into training (75%) and testing (25%) sets using a stratified split to preserve class proportions.
3. **KNN Classifier:** A K-Nearest Neighbors classifier was chosen.
4. **Hyperparameter Tuning:** GridSearchCV with 5-fold stratified cross-validation was employed to optimize KNN's n_neighbors ([3, 5, 7, 9, 11]), weights (['uniform', 'distance']), and metric (['euclidean', 'manhattan', 'cosine']). The optimal parameters found were {'metric': 'cosine', 'n_neighbors': 7, 'weights': 'distance'}.
5. **Persistence:** The LabelEncoder, StandardScaler, PCA model, and the optimized KNeighborsClassifier were saved using pickle for later use in the prediction application.

4. Results and Evaluation

The performance of the optimized KNN model was assessed on the unseen test set.

- **Cross-Validation Accuracy (during tuning):** 0.8797
- **Test Set Accuracy (final model):** 0.8857 (88.57%)

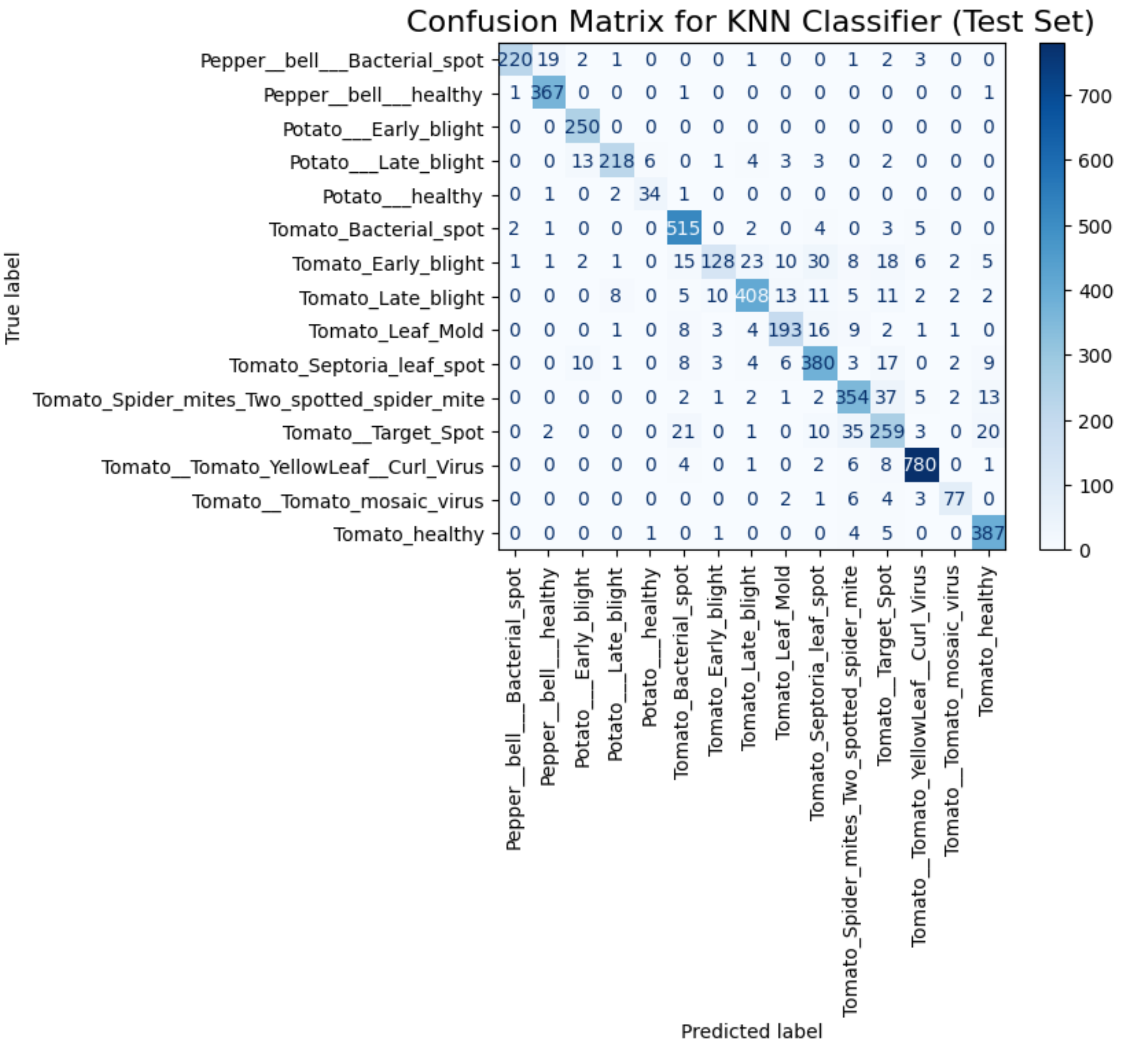
4.1. Classification Report Analysis

The detailed per-class performance on the test set is summarized below:

Class	Precision	Recall	F1-Score	Support
Pepper__bell___Bacterial_spot	0.9821	0.8835	0.9302	249
Pepper__bell___healthy	0.9386	0.9919	0.9645	370
Potato___Early_blight	0.9025	1.0000	0.9488	250
Potato___Late_blight	0.9397	0.8720	0.9046	250
Potato___healthy	0.8293	0.8947	0.8608	38
Tomato_Bacterial_spot	0.8879	0.9680	0.9263	532
Tomato_Early_blight	0.8707	0.5120	0.6448	250
Tomato_Late_blight	0.9067	0.8553	0.8803	477
Tomato_Leaf_Mold	0.8465	0.8109	0.8283	238
Tomato_Septoria_leaf_spot	0.8279	0.8578	0.8426	443
Tomato_Spider_mites_Two_spotted_spider_mite	0.8213	0.8449	0.8329	419
Tomato__Target_Spot	0.7038	0.7379	0.7204	351
Tomato__Tomato_YellowLeaf__Curl_Virus	0.9653	0.9726	0.9689	802
Tomato__Tomato_mosaic_virus	0.8953	0.8280	0.8603	93
Tomato_healthy	0.8836	0.9724	0.9258	398
Overall / Averages				
Accuracy			0.8857	5160
Macro Avg	0.8801	0.8668	0.8693	5160
Weighted Avg	0.8866	0.8857	0.8830	5160

4.2. Confusion Matrix Analysis

The confusion matrix (Figure 4) visually details the classification performance, highlighting correct predictions along the diagonal and specific misclassifications off-diagonal. For instance, further analysis of the off-diagonal elements for "Tomato_Early_blight" would reveal which other classes it was most often confused with, providing insights for potential model improvements.



5. Prediction Application (app.py)

A Python script, app.py, was developed to serve as a practical demonstration of the trained system. This script encapsulates the entire prediction pipeline:

1. **Loading Artifacts:** Loads the saved label_encoder, scaler, pca_model, and best_knn_model.
2. **Feature Extractor Re-creation:** Reconstructs the MobileNetV2 feature extractor.
3. **Prediction Process:** For a new input image, it performs:
    - CNN feature extraction.
    - Scaling of features using the loaded scaler.
    - PCA transformation using the loaded pca_model.
    - Prediction using the loaded knn_model.
    - Decoding of the numerical prediction to a disease name using the loaded label_encoder.
      The script successfully predicted a sample image as: Predicted disease: Pepper__bell___Bacterial_spot (Confidence: 99.84%).

6. Discussion

6.1. Interpretation of Results and System Efficacy

The overall test accuracy of 88.57% indicates that the chosen approach of using CNN-extracted features with PCA and KNN is effective for plant disease classification. The high precision and recall for many classes demonstrate the model's ability to learn distinguishing characteristics from the leaf images. The systematic feature engineering and hyperparameter tuning were crucial in achieving this performance.

6.2. Strengths

- **Effective Feature Representation:** Leveraging MobileNetV2 provided robust features far superior to raw pixels for KNN.

- **Optimized Model:** GridSearchCV ensured the KNN model was tuned for optimal performance on the given feature set.
- **Handling of Imbalance:** Stratified splitting and comprehensive per-class metrics provided a nuanced evaluation.
- **End-to-End Solution:** The project delivered a complete pipeline from data processing to a functional prediction script.

7. Conclusion

This project successfully developed and evaluated a plant disease classification system employing a K-Nearest Neighbors classifier on features extracted by a pre-trained MobileNetV2 and refined by PCA. The achieved test accuracy of 88.57% demonstrates the viability of this hybrid approach. The accompanying app.py script validates the system's capability for practical predictions. While certain limitations exist, primarily related to class imbalance and the dataset's scope, the project establishes a strong foundation for further development and potential deployment in agricultural applications.

Team Members

Moataz Ahmed Samir	2305223
Malak Gehad	2305249
Omar El-Sayed	2305057