NATIONAL UNIVERSITY OF SINGAPORE
School of Computing
IS5126 - Hands-on with Applied Analytics
Semester 2 – AY 2023/24
Final Project
Group Submission

DETECTING DISEASE ON PLANTS - IMAGE CLASSIFICATION

Group 8 - NeuroNest

Group Members –
Putri Darmawan (A0200717R)
Premi Jeevarathinam (A0268262Y)
Cai Zhi Hao (A0290917W)
Date –
19th April 2024

1. Motivation & Objective

The motivation for our project is to address the critical need for early detection of crop diseases in agriculture. By leveraging machine learning models, the objective is to accurately identify diseases from images, empowering farmers to monitor crop health in real-time and take timely action to prevent outbreaks. This approach aims to enhance food security, maximize crop yield, and enable informed decision-making in disease management strategies.

2. Problem Statement

The Crop Leaves dataset presents a challenging task of **classifying 38 different classes of crop leaves into healthy and diseased crop leaves based on RGB images**. This data enables the development and assessment of a variety of plant phenotyping models aimed at identifying desirable traits. Our focus is on the question: "**Does this plant exhibit signs of diseases?**' and "**Which type of disease does the plant have?**". We will develop machine learning models that can distinguish between the various classes to aid in the early detection of crop diseases and ensure the health and yield of agricultural crops. By leveraging this dataset, we can explore techniques for image classification, and disease identification in crops, contributing to advancements in precision agriculture and sustainable farming practices.

3. Dataset

The dataset available on Kaggle, titled "Plant Disease" (https://www.kaggle.com/datasets/saroz014/plant-disease/data), contains images of various crop plants affected by different diseases. It comprises images of healthy plants as well as those afflicted by diseases such as bacterial leaf blight, brown spot, and yellow leaf curl virus. The dataset comprises 54,305 images distributed across 38 distinct classes, with 43,456 images allocated for training purposes and 10,849 images reserved for testing and evaluation.

4. Exploratory Data Analysis

Data distribution among 38 classes:

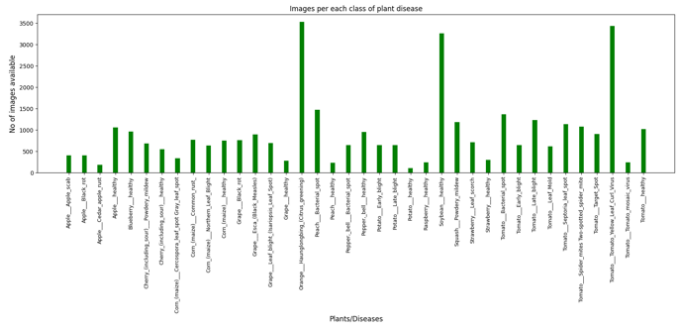

Image 1: Data distribution on original data

The exploratory data analysis of the plant disease image dataset reveals a significant class imbalance, with certain disease classes having a substantially larger number of images compared to others. This imbalance in the dataset may necessitate the application of techniques such as augmentation and undersampling to mitigate potential biases.

5. Data Preprocessing

5.1. Resizing and Normalizing Images

To prepare the dataset for training, we undertook various preprocessing measures for consistency and optimal data handling. Initially, we resized all images to a standard size of 224x224 pixels, ensuring uniform processing and decreasing computational demands during model training. Following this, we normalized pixel values by dividing them by 255, transforming the pixel range from [0, 255] to [0, 1]. This normalization not only simplified calculations but also improved the model's learning efficiency during training.

5.2. PCA

Dealing with big datasets in KNN can be tough. When training KNN on such data, we often encounter memory issues because processing large, flattened image data consumes a lot of memory. The dataset is simply too big to fit into memory, leading to frequent crashes during training on our computer. To address this challenge, we opted to use a smaller subset of the dataset for training, employed dimensionality reduction using the PCA method, and reduced the batch size, which helped alleviate the memory constraints.

We set the parameter n_components=0.95 in PCA, indicating that PCA will automatically select enough principal components to preserve 95% of the total variance in the data. This approach allows us to determine the number of new features after dimensionality reduction without explicitly specifying the exact number of principal components. PCA is then computed based on the training dataset, during which the method calculates the mean and covariance matrix of the training data to identify the principal components that capture the most variance. Finally, we transform the original image data into a new space composed of these principal components, effectively achieving dimensionality reduction while maintaining the essence of the original data.

5.3. Augmentation and Under sampling

During our dataset analysis, we discovered a significant class imbalance. To address this issue, we utilized data augmentation methods specifically targeted at the minority classes. These techniques included zooming, shearing, horizontal flipping, rotating, and shifting images, which added diversity to the minority class data and enhanced the model's ability to generalize effectively. Additionally, for classes with more than 1000 images, we implemented undersampling, reducing the sample size in the overrepresented classes to achieve a more balanced dataset. This combined strategy of data augmentation and undersampling resulted in a more even distribution of data, leading to improved model performance in handling imbalanced class distributions during training.

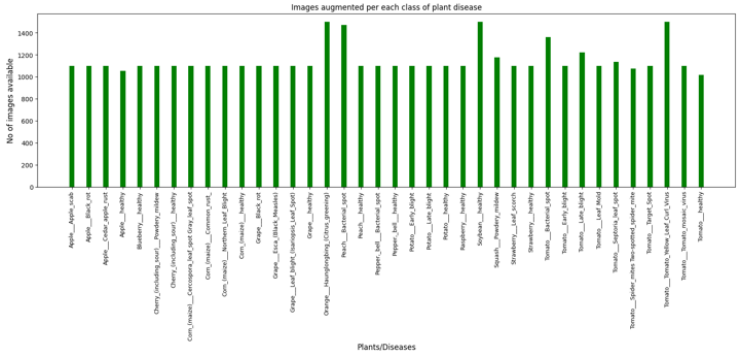

Image 2: Data distribution on training data after augmentation

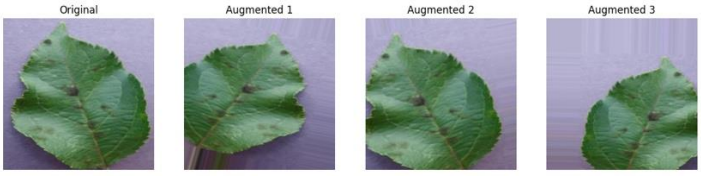

Image 3: Augmented image sample

6. Model Selection and Tuning

The overall flow comprises five main steps, each of which is detailed below.

1. **Split Training and Validation Data**: The first step involves splitting the available data into two subsets: 80% for training and 20% for validation.
2. **Data Augmentation on training data**: After the initial split, data augmentation techniques are applied to the training data. This approach aimed to enhance data diversity and quantity, potentially boosting the model's performance and generalization abilities. Performing data augmentation after the split was crucial to prevent data leakage into the validation set, ensuring unbiased model evaluation and accurate performance assessment.

3. **Fit the model with training data and tunes it based on validation performance**: The augmented training data is used to fit and train the machine learning model. During this process, the model's hyperparameters or configurations are tuned and adjusted based on the performance observed on the validation data subset.
4. **Train the best model with the augmented training and validation data**: After determining the optimal hyperparameters based on the validation performance, the best-performing model is trained again using the augmented training and validation data.
5. **Test model with test set**: Finally, the trained and optimized model is evaluated on an independent test set, which was not used during the training or validation phases. This step provides an unbiased assessment of the model's performance on unseen data, allowing for a comprehensive evaluation of its generalization capabilities and overall effectiveness.

6.1. KNN

In this project, we firstly employ the K-Nearest Neighbors (KNN) algorithm, a simple yet effective method used widely in pattern recognition and machine learning for classification tasks. KNN classifies new cases based on a similarity measure (e.g., distance functions). Our project leverages KNN to classify plant disease images after reducing the dimensionality of the original image data using Principal Component Analysis.

6.2. SVM

The process of selecting and tuning models with Support Vector Machines (SVMs) for image classification involved a meticulous approach aimed at optimizing performance and accuracy. Various SVM kernels, including linear, RBF, polynomial, and sigmoid, were explored, each offering unique characteristics. Despite their potential, SVMs faced challenges with high-dimensional image data, leading to prolonged training times and computational complexity. Additionally, SVMs exhibited limited sensitivity to hyperparameters, making effective fine-tuning difficult, especially with the diminishing effectiveness of distance-based non-linear kernels like RBF. These limitations underscored SVMs' inferior suitability for this image classification task.

Hyperparameter tuning result:

Kernel	C	Gamma	Accuracy
Linear	0.1	0.01	0.738
	10	0.1	0.73
	10	1	0.72
	10	10	0.719
Poly	10	0.1	0.63
	0.1	10	0.62
Rbf	10	1	0.61
	0.1	10	0.60
Sigmoid	0.01	1	0.51
	0.1	0.1	0.49

Despite these challenges, the intricacies of SVM tuning were further explored by meticulously optimizing crucial parameters such as C and gamma. While the model exhibited limited sensitivity to C and gamma, significant improvements in accuracy were observed with changes in kernel. By ranking the accuracy from various kernels, the linear kernel yielded the best results compared to others. The results of the experiments were analyzed, evaluating the model's performance across various metrics such as accuracy, precision, recall, and F1-score.

Table 1: SVM Hyperparameter tuning result

6.3. CNN

We opted for a Convolutional Neural Network (CNN) approach for image classification due to its proven efficacy in handling image data. CNNs excel at automatically extracting crucial features while considering the spatial aspects of images, making them an ideal choice for our task. Our CNN model architecture is as follows:

Image 4: CNN architecture

Several factors influenced our choice of parameters:

- **Maxpooling**: Maxpooling helps to make images smaller as they move through the network. It works by picking the biggest value in a certain area of the image, keeping important details and getting rid of less important parts. This helps the network concentrate on the important parts, making it easier to process and find key features needed for accurate classification.
- **ReLU Activation**: Utilized for its computational simplicity and effectiveness in capturing nonlinearities.
- **Dropout**: Dropout tackles this problem by randomly turning off some neurons during each training step, making the network use different paths and not rely too much on specific details. This helps the network learn more versatile patterns, improving how it works with new data and lowering the chance of overfitting.
- **Output Layer**: Configured with 38 units and SoftMax activation, aligning with our 38-class multiclass classification problem.
- **Adam compiler with 0.001 learning rate**: Selecting the Adam optimizer with a learning rate of 0.001 to train the CNN is based on its ability to adapt and adjust learning rates for each parameter. This adaptability helps the model converge faster and generalize well to new data. Adam's use of momentum further speeds up convergence, especially useful for complex CNN structures, as it smoothens updates and prevents fluctuations during training.
- **Categorical cross entropy loss**: Using categorical cross entropy loss fits well with multiclass classification problems, making it compatible with softmax activation to produce precise class predictions.

7. Evaluation

7.1. Model Comparisons

Results on validation data:

Model	F1	Precision	Recall	Accuracy
KNN	0.38	0.54	0.39	0.392
KNN with Data preprocessing	0.36	0.56	0.38	0.377
SVM - Linear	0.70	0.68	0.68	0.73
SVM - Poly	0.61	0.58	0.56	0.63
SVM - RBF	0.57	0.54	0.55	0.61
CNN	0.82	0.85	0.81	0.87
CNN with Data preprocessing	0.85	0.87	0.85	0.89

Table 2: Validation evaluation results

Following thorough tuning and validation testing, we have selected a Convolutional Neural Network (CNN) with data preprocessing as our final model. The outcome of this selection, tested with test data subset, reflected in the classification scores showing an accuracy of 0.89 along with consistent macro-average and weighted-average precision, recall, and F1-score values of 0.88, 0.85, and 0.85, and 0.90, 0.89, and 0.89, respectively, over the same dataset, demonstrates the model's reliable performance.

7.2. Model Evaluations

KNN presents several advantages and disadvantages for image classification. On the positive side, KNN's inherent simplicity makes it easy to implement and comprehend, offering clear interpretability through its majority vote-based classification approach. Additionally, leveraging Principal Component Analysis (PCA) for feature reduction can significantly enhance KNN's performance by addressing the curse of dimensionality issue, thereby improving the classifier's accuracy and efficiency. However, there are notable drawbacks to consider. KNN faces challenges in image classification due to its computational intensity when querying large datasets, especially after dimensionality reduction. Additionally, KNN's reliance on local information and susceptibility to noise and outliers can limit its ability to capture global patterns and make accurate predictions in complex image data.

SVMs face challenges with high-dimensional image data, leading to prolonged training times and computational complexity. SVMs exhibit limited sensitivity to hyperparameters, making effective fine-tuning difficult. Moreover, the curse of dimensionality is diminishing the effectiveness of distance-based non-linear kernels like RBF. These limitations highlight SVMs' inferior suitability for this image classification task.

CNN works well with image classification tasks because of their ability to effectively capture and learn hierarchical representations of visual patterns. CNNs use the spatial and local correlation present in images, making them highly effective in recognizing and distinguishing different objects, textures, and features. Through their convolutional layers, CNN can extract important features. Furthermore, CNN is translation invariant, enabling them to recognize the same object or pattern regardless of its position within the image, enhancing their robustness and generalization capabilities.

Training on balanced data offers several advantages, including unbiased learning across all classes and improved model interpretability. Balanced data ensures equal representation of each class during training, preventing biases towards dominant classes. This balanced exposure helps in mitigating the risk of the model being skewed towards predicting the majority class and facilitates better discrimination between classes, leading to a more reliable and generalizable model. Furthermore, when combined with data augmentation techniques, training on balanced data enhances the model's variation and generality, further improving its performance on unseen data.

8. Difficulties Faced

The challenge of handling high-dimensional image data, especially with 38 classes, was pronounced, as both SVMs and KNN struggled due to computational demands and prolonged training times. Non-linear SVMs were computationally expensive due to the dataset's complexity and struggled with large datasets like ours, leading to prolonged training times. Similarly, Convolutional Neural Networks (CNNs) also required significant computational resources and longer training times, albeit offering better performance in capturing intricate patterns and features within images compared to SVMs.

9. Conclusion

CNN emerges as the superior model for image classification tasks due to its ability to capture hierarchical representations of visual patterns and exploit spatial correlations within images. SVMs and KNN face challenges with high-dimensional data and limited sensitivity to hyperparameters. Training CNN on balanced data offers significant advantages by ensuring unbiased learning across classes, improving interpretability, and enhancing model generalization. Future improvements could include utilizing pre-trained models, GPU acceleration, and advanced optimization techniques. Overall, CNN outperforms SVMs and KNN, making it the optimal choice for image classification tasks.

10. Further Improvements

In addition to the current strategies, further improvements can be achieved through the following approaches:

- **Transfer Learning**: Pre-trained models like ResNet, VGG, or Inception can be fine-tuned on the project's dataset. This technique allows the model to leverage knowledge learned from a large dataset, potentially boosting performance on the smaller dataset used in this project.
- **Model Interpretability**: Techniques such as Grad-CAM, SHAP values, or LIME can interpret and visualize the classification model's predictions. These methods provide insights into the model's decision-making process, helping identify areas for improvement and better understand the factors contributing to the distinction between human-generated text and language model outputs.
- **Model Ensemble**: Multiple models with diverse architectures or trained on different data subsets can be combined into an ensemble. Their predictions can be combined using techniques like averaging or stacking. Ensembles can enhance the overall system's robustness and generalization performance by leveraging the strengths of individual models and mitigating their weaknesses.

11. References

[1] Kim, Jinho, Byung-Soo Kim, and Silvio Savarese. "Comparing image classification methods: K-nearest-neighbor and support-vector-machines." Proceedings of the 6th WSEAS international conference on Computer Engineering and Applications, and Proceedings of the 2012 American conference on Applied Mathematics. 2012.

[2] Chaganti, S. Y., Prudhvith, T. G. N., Nanda, I., Kumar, N., Pandi, K. R., et al. (March 2020). "Image Classification using SVM and CNN." DOI: 10.1109/ICCSEA49143.2020.9132851. Retrieved from ResearchGate: Link

[3] "How To Implement Image Classification Using SVM In Convolution Neural Network". Retrieved from Youtube: Link

[4] Purwono, Purwono & Ma'arif, Alfian & Rahmaniar, Wahyu & Imam, Haris & Fathurrahman, Haris Imam Karim & Frisky, Aufaclav & Haq, Qazi Mazhar Ul. (2023). Understanding of Convolutional Neural Network (CNN): A Review. 2. 739-748. 10.31763/ijrcs.v2i4.888.

12. Annex

Step-by-step process:

1. Download data from Kaggle (https://www.kaggle.com/datasets/saroz014/plant-disease/data). If using the code in CNN juptyer notebook, need to download kaggle.json from Kaggle (API key)
2. Conduct Exploratory Data Analysis (EDA) to comprehensively analyze and understand the characteristics, patterns, and distributions within the dataset.
3. Perform dataset splitting, dividing it into distinct training and validation subsets, ensuring independence between these sets to avoid data leakage and maintain robust evaluation metrics.
4. Apply data augmentation exclusively to the training data subset, enhancing its diversity and aiding the model's generalization capabilities without altering the validation data.
5. Train the K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Convolutional Neural Network (CNN) models on both the original dataset and the augmented dataset to compare their performance and evaluate the impact of data augmentation.
6. Conduct hyperparameter tuning based on the validation set's performance, optimizing parameters such as learning rates, regularization factors, and model architectures to enhance predictive accuracy and generalization. SVM is tuned with grid search with cross validation, CNN is tuned manually (by changing number of layers and neurons), while KNN is tuned manually by changing the value of k.
7. Utilize the augmented training and validation datasets to train the best-performing model (in this case, the CNN) further, leveraging the augmented data's enriched features and diversity.
8. Evaluate the final CNN model's performance using the independent test dataset to assess its generalization capabilities and validate its effectiveness in real-world scenarios.

The steps can be seen in the Jupyter Notebook. It's important to note that the models incorporate randomness, so running them again may yield different results.

Data / Information	Description
Preprocessed Training Dataset	Data size: 34781 images Image_size: (224,224) Pixel values: from [0, 255] to [0, 1]
PCA Parameters	The explained variance ratio (n_components): 0.95
Augmentation Techniques	Zoom(0.2), Shear(0.2), Horizontal flip, Rotate(20), Width shift(0.2), Height shift(0.2)
Augmented Dataset	Training data: 43710 images Validation data: 8675 images (not augmented) Test data: 10849 images (not augmented)
Hyperparameters - KNN	n_neighbors= 3, 5, 7, 9, 15
Hyperparameters - SVM	C: [0.1, 1, 10, 100] gamma: [0.01, 0.1, 1, 10] kernel: ['linear', 'rbf', 'poly', 'sigmoid']
Hyperparameters - CNN	Number of layers and neurons
Validation Performance Metrics	Accuracy, Precision, Recall, and F1-score.
Testing Performance Metrics	Accuracy, Precision, Recall, and F1-score.

Table 3: Annex - Model Training and Evaluation Information

Data Samples:

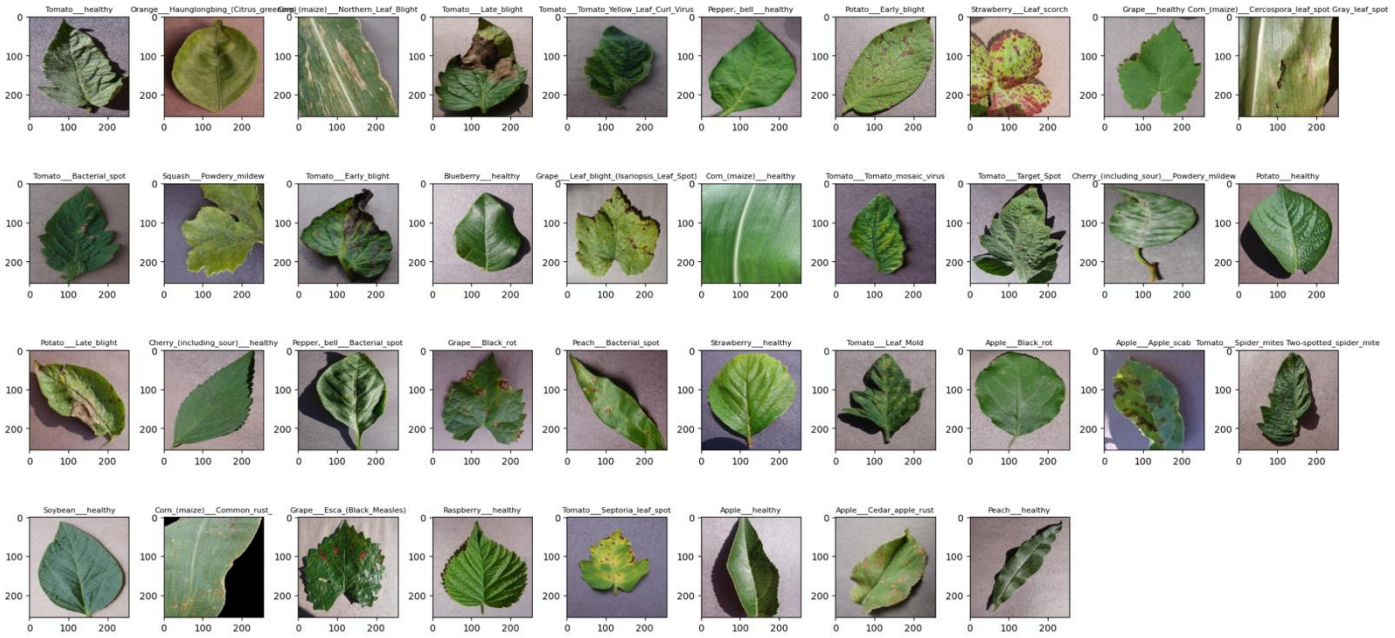

Image 5: Data Samples from Original Dataset