SYSEN-6888 Deep Learning

Final Project Report

Plant Disease Detection

Instructor: Fengqi You

By: Yen-Hsing Li <yl2924>

Yimian Liu <yl996>

Zechen Wang <zw652>

Date: Dec. 10, 2023

Abstract

This project explores the use of deep learning for early detection and classification of plant diseases using the PlantVillage dataset. It focuses specifically on tomato leaves, evaluating various convolutional neural network (CNN) models. The DenseNet121 model proved to be the most suitable due to its balance of accuracy and inference time. Data augmentation and normalization addressed imbalances in the dataset and enhanced model training. When testing with real-world images, we discovered that the performance of our model varied for different tasks, which suggests that further improvements are needed in order to adapt for real-world applications. The project suggests one of the exciting future applications of deep learning in the field of agriculture, which can be used to help farmers continuously monitor their crops with the increased number of IoT devices and network coverage.

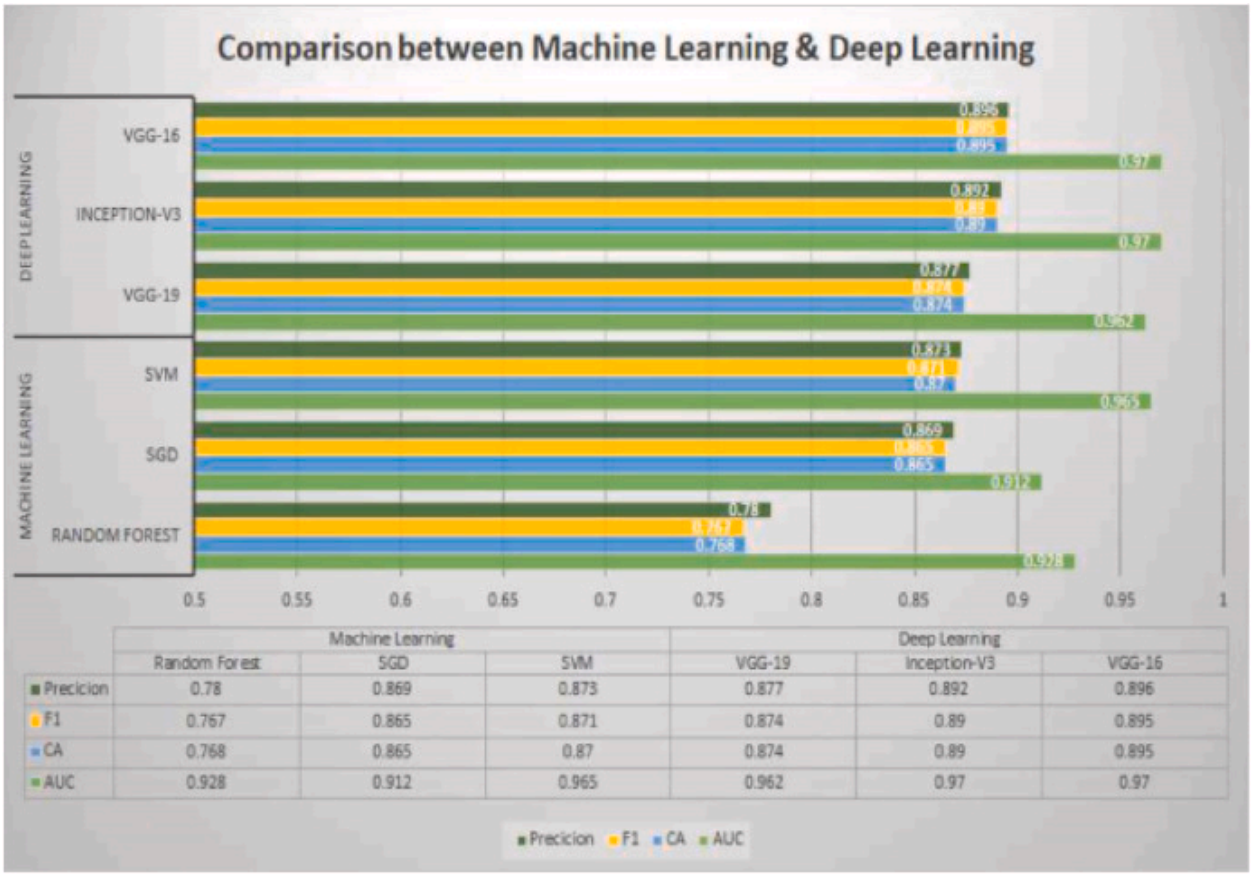
Introduction and Background

In agriculture, it is a vital issue to discover plant disease at an early stage, or the production may decrease significantly. To improve productivity, early detection technology is critical to farmers for managing and decision making. Studies show that the climate changes and host resistance may vary pathogenic stage and rate[1]. However nowadays, diseases are spread more freely and complicate the situation. New symptoms may emerge in areas where they were previously unidentified and these symptoms can be challenging to detect through observation.

Disease-affected plants often display distinct signs on their leaves, stems, flowers, or fruits. Typically, each disease or pest condition exhibits a specific visual pattern, which can be utilized for precise diagnosis of the abnormality. In most cases, plant leaves serve as the primary indicators for disease identification, as they are frequently the first site where symptoms of various diseases manifest.

Previous studies[2][3] have reviewed several papers for detecting plant disease by both neural based methods or non-neural methods. The non-neural approaches usually used machine learning methods such as Naive Bayes , Decision Tree, Nearest Neighbor, Vector Machine Support (SVM) and Random Forest. The neural approaches use different Convolutional Neural

Network(CNN) frameworks such as InceptionV3, VGG-16 to learn the features of the disease on the plant leaves.



	Machine Learning			Deep Learning		
	Random Forest	SGD	SVM	VGG-19	Inception-V3	VGG-16
Precicion	0.78	0.869	0.873	0.877	0.892	0.896
F1	0.767	0.865	0.871	0.874	0.89	0.895
CA	0.768	0.865	0.87	0.874	0.89	0.895
AUC	0.928	0.912	0.965	0.962	0.97	0.97

Fig.1 Referenced from paper[4], this diagram signifies the comparative estimation of machine and deep learning techniques.

Notably, paper [4] presents evidence suggesting that deep learning-based methods tend to surpass machine learning methods in terms of performance(Figure 1). In this project, our objective is to further investigate and analyze the performance of various CNN models in the context of plant disease detection. By conducting a series of experiments and evaluations, we aim to determine which CNN models exhibit the most promising results in accurately identifying plant diseases at early stages, thereby contributing to the broader field of agricultural technology and plant pathology.

Problem Statement

Today modern technologies make a huge impact on global food production, substantially increasing the yield of different crops across the globe. However, there still exist challenges to food security due to climate change, pollinator decline, and the threat of plant diseases. Plant diseases not only affect global food supplies but also bring severe problems to individual farmers, who contribute a great portion of agricultural production in the developing world. Within these farmers, those who lack the proper equipment or infrastructure will face significant difficulties while monitoring acres of lands with their bare eyes.

The prevalence of IoT devices presents an opportunity for innovative disease identification solutions, leveraging hardware such as phone cameras and computing power to bring solutions to farmers through mobile devices. This project aims to take advantage of the PlantVillage dataset [5] of leaf images of commonly seen plant diseases to create a classification model to identify plant diseases at the earliest possible stages with minimal effort, so that the farmers could be easily aware of the situation with only their mobile devices in hand. Through the application of deep learning techniques, specifically convolutional neural networks and transfer learning, our primary goal is to develop an automated image recognition system capable of accurately identifying diseases in crops. The availability of a substantial dataset of 54,306 images across multiple plant species with diseases, generously provided by the PlantVillage project [6], made training deep neural networks much more feasible.

We believe our model can be implemented in a real-life setting to help the farmers monitor their crops. As illustrated below, we plan to use our model as a connection between the data collection endpoint and the farmers to provide automatic plant disease detections. This model has the potential to replace human labor, which has inevitable manual errors and cannot keep a continuous surveillance 24/7. Our proposed system can act as a real-time alerting tool to increase overall production for the farmers.

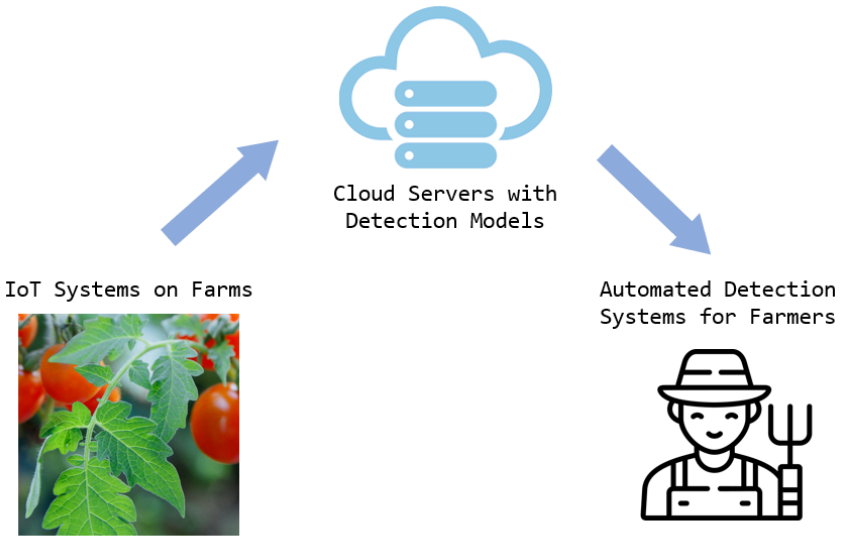


Fig.2 Sample workflow

The IoT systems on Farms in our design will act as data collection sources, which continuously captures leaves of tomatoes on different sections of the farms. These images or video streams will be then transmitted to cloud-based servers, equipped with our deep learning detection models, for disease detection. Once the detection is done, if any suspected infected plant is found, our system can send a notification to the farmers to alert them. For the scope of this project, we specifically focus on the construction of our deep learning model, and we designed this system to prove that our model does bring a meaningful impact to the problem we care about.

The data we use is available on Kaggle, a public platform for dataset and machine learning model sharing. This easily accessible data consists of 54,306 individual 256×256 RGB images of leaves, labeled according to the plant species and whether they are infected with a disease or not. Following are examples of how images would look like in the dataset:

Fig.3 leaf image of healthy and infected plants

Using the PlantVillage dataset, we plan to first to perform data preprocessing, and then either develop our own convolutional neural network to detect diseases in these leaves, or to use techniques such as transfer learning for classification. Specifically, we aim to focus on the tomato species and the detection of various different types of diseases according to their leaves' images due to the characteristics of the dataset we use, which will be discussed in a later section. Multiple models will be trained and their individual performance will be evaluated.

Data Source

PlantVillage Dataset

The PlantVillage dataset [5], found on Kaggle, is a robust collection of leaf images designed for creating and testing machine learning models for plant disease detection. This dataset includes 54,306 high-resolution RGB images (256×256 pixels), divided into 15 classes according to plant species and disease status. The accurate labeling of these images makes them a valuable resource for training and validating deep learning models. The dataset's public accessibility on the Kaggle website made it easy to download directly.

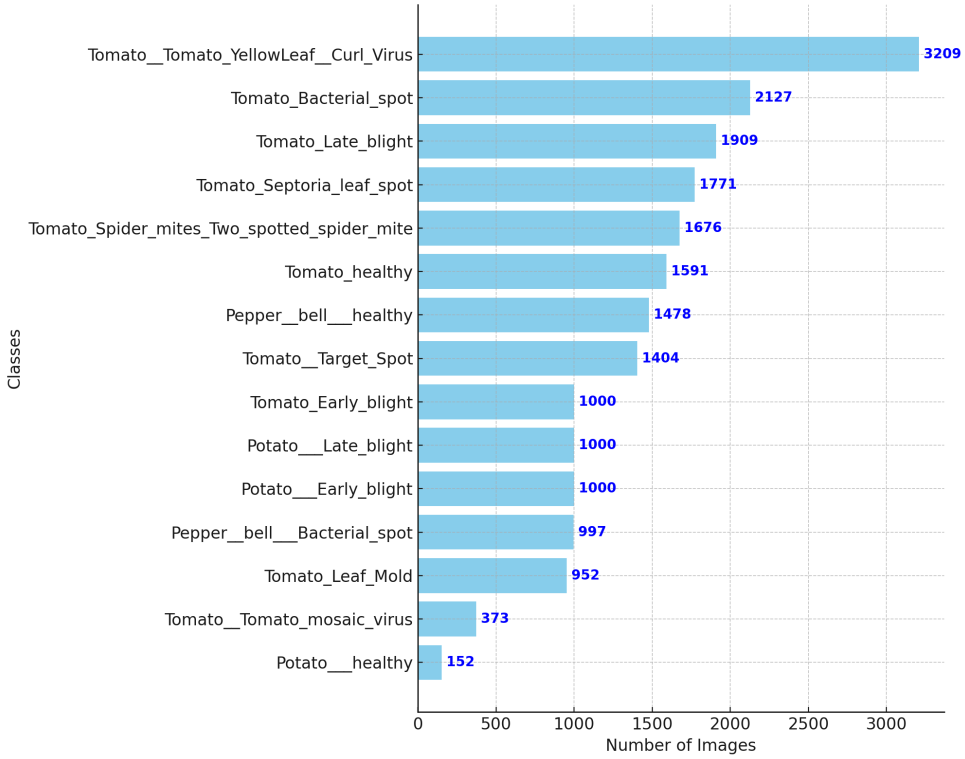


Fig.4 Distribution of Image Counts per Class in PlantVillage Dataset

The PlantVillage dataset, as depicted in the bar chart, contains features relevant to our project. It includes images of diverse plant diseases along with healthy plants, helping us to establish a comprehensive model capable of identifying an array of diseases. However, there's a notable disparity in image quantities across different classes. For instance, the

"Tomato__Tomato_YellowLeaf__Curl_Virus" class holds 3209 images, while classes like "Tomato__Tomato_mosaic_virus" only include 373. This imbalance could potentially introduce bias towards classes with a higher amount of data, affecting our model's performance. Moreover, the scarcity of images in some classes could also influence the accuracy of our model. One way to balance the dataset is to use data augmentation, by randomly including rotated, scaled, or mirrored images using the original dataset, we could supplement the data for each single class and therefore make them numerically more balanced.

Plant Disease Expert Dataset

We've discovered another dataset on Kaggle called Plant Disease Expert [7] that we can use to improve the accuracy of our model. This dataset complements the PlantVillage dataset by offering more images and labels, which will enhance our model's performance. It consists of 199,665 RGB images sized 256x256, categorized into 58 different classes. By incorporating this dataset, we will create a strong and comprehensive training environment for our deep learning algorithms.

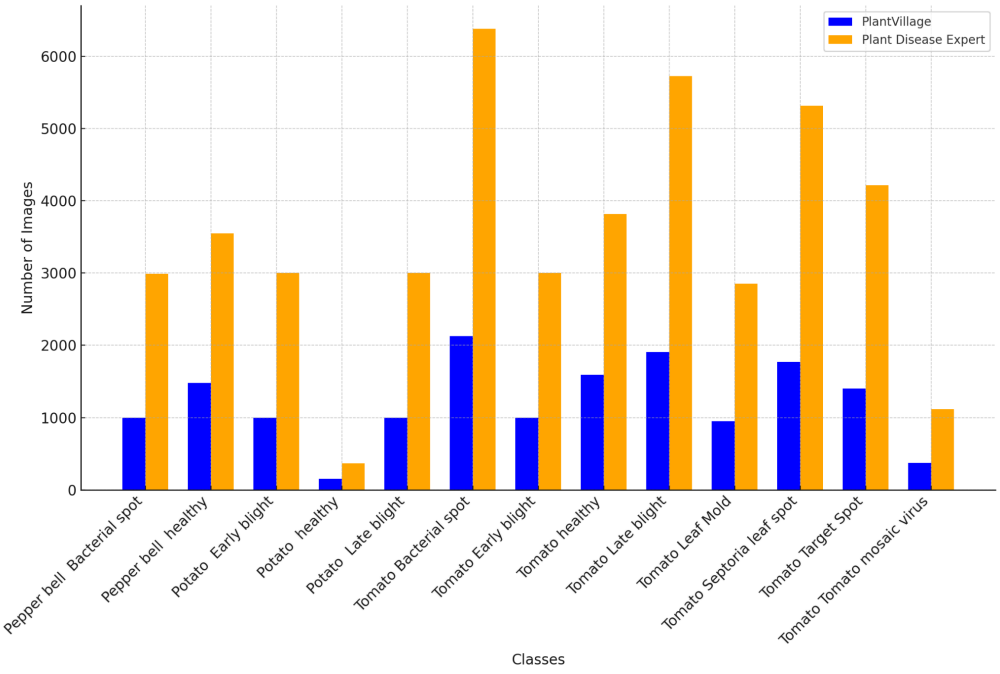


Fig.5 Comparison of Common Classed in PlantVilage and Plant Disease Expert Datasets

This bar chart effectively compares the number of images for 13 common classes between the PlantVillage and Plant Disease Expert datasets. It's clear that the image count for each class is higher in the Plant Disease Expert dataset. However, the Plant Disease Expert dataset includes some repeated images that are merely flipped or rotated versions of the same image, which is partially the data augmentation process we mentioned earlier. For this reason, we've chosen to use the PlantVillage dataset as our primary source instead of the Plant Disease Expert dataset.

Fig.6 Example Image of Tomato Bacterial spot class in PlantVilage (left) and Plant Disease Expert (right)

The above figure showcases two randomly chosen images from the Tomato Bacterial Spot class in both the PlantVillage and Plant Disease Expert datasets. These images have commonalities such as similar backgrounds, lighting, and photographic conditions. Yet, the shadow in the Plant Disease Expert image is more pronounced than in the PlantVillage dataset. This implies that merging these two datasets could potentially yield a more generalized model.

Metrics

We introduced two different metrics to evaluate the performance of our model in our project. We first created a basic metric, which is the accuracy for healthy/unhealthy classification. With this, we can measure the model's capabilities in the most straightforward way – detecting infected plants from the healthy plants. However, in order to get a better understanding on the performance of our models, we also wanted to introduce a second metric, which is the accuracy metric for specific disease classification. We created the second metric because we not only want our model to tell whether a plant is infected or not, but also want it to be able to classify the disease a specific plant is infected with.

These metrics are crucial for us to understand the effectiveness of the models we train and how exactly they perform on different tasks. From simpler tasks such as identifying if a plant is healthy, to more difficult tasks such as pointing out what specific kind of disease is present on the plant, the model is expected to have different performances. Therefore, the combination of these two metrics may offer a comprehensive view of how these models perform in real-world scenarios, where accurate disease identification is essential.

Methods and Tools

Data Preprocessing

Data preprocessing is a very important step in the development of our deep learning model for plant disease detection. The PlantVillage dataset we choose to use contains a diverse set of images with an imbalanced number of images. Therefore, we need to implement augmentation so that each class can have a similar amount of data. We utilized data augmentation techniques to artificially expand and enhance our dataset, which included operations such as random rotations, zooming, horizontal flipping, and shifting. This has not only helped us in preventing overfitting but also helped our model to adapt to generalized and unseen samples.

Fig.7 Augmented images from PlantVillage

Fig. 7 includes images of our augmented data, we can see that by the previously mentioned methods, we were able to create a diverse set of images. Since the original dataset contained all leaves placed in an upward direction at the center, these variations will greatly help the generalization of the model. Also, the transformations have introduced additional noise in the background. Since most images in the same class share similar background colors and lighting conditions, it is important to introduce such noise so that our model does not specifically focus on learning the background patterns. We aimed to use this model in a generalized real-life case, where we assume the background of input images can be a lot noisier, and the lighting conditions and the leaf locations will not likely be the same as the training models.

Another key aspect of our preprocessing is normalization. We made sure that the pixel values of each image are normalized to the same scale. By making sure that the input data has a uniform scale, we reduce the complexity of the optimization and aim to correct the potential error from different image formats or scales, making it easier for the model to learn patterns effectively and correctly.

Deep Learning Frameworks

For this project, we used TensorFlow/Keras as our primary deep learning frameworks. We found these tools contained very useful functions and models that we used to build our transfer learning models and performance analysis systems.

Transfer Learning

Alongside developing our own custom CNNs, we plan to explore transfer learning using pre-trained models. These models which have been trained on vast datasets already contain understanding of visual features from images, which is very helpful for our image classification problem. By using transfer learning, we can take advantage of this pre-acquired knowledge to further fine-tune it to our specific task of plant disease classification. Given the limited size of the PlantVillage dataset, this approach could be really useful since it reduces the amount of training data required. With that being said, we decided not to construct our own CNN models but to fully focus on exploring the existing architectures using transfer learning.

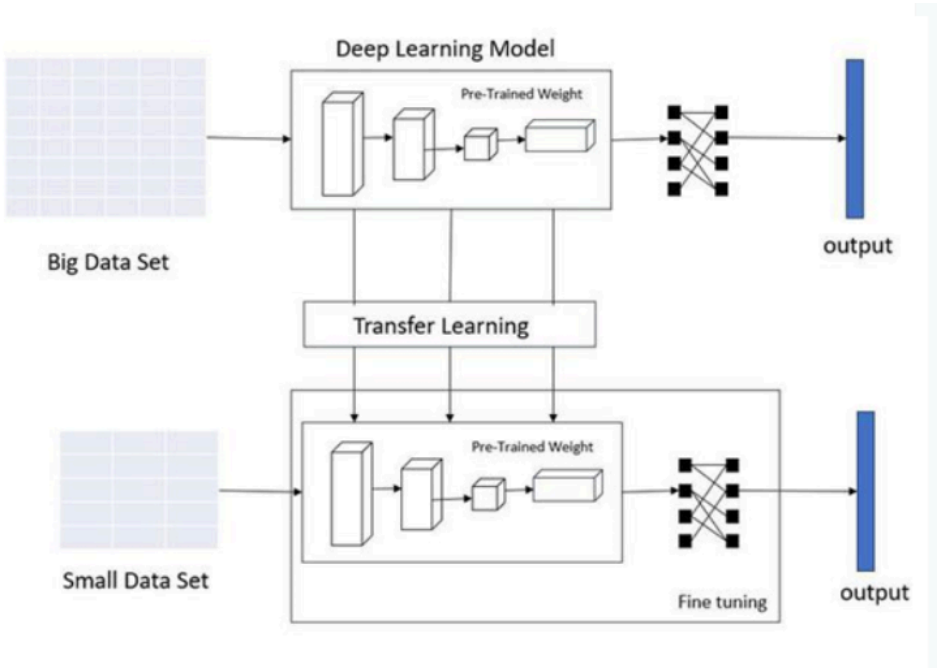


Fig.8 Transfer Learning Process

The models we explored are listed below, each with their own distinct architectures:

- **ResNet**, is great for handling the vanishing gradient problem with its architecture and residual connections, making it ideal for learning from complex datasets like ours. [8]
- **DenseNet**, notable for its dense connections between layers, contains and preserves much more complex information flow, which is crucial for capturing subtle disease indicators in plant leaves.
- **EfficientNetB1**, known for its balance between accuracy and efficiency, can be a powerful yet resource-efficient choice for our application.
- **VGG16**, known for its simplicity and depth, uses a series of convolutional layers followed by max pooling layers. This architecture is particularly useful for capturing complicated features in images.
- **Inception V3**, an advancement in the inception series, is distinguished by its 'network within a network' design and the use of asymmetric convolutions. This structure allows it to efficiently manage computational resources while maintaining high accuracy.
- **InceptionResNet V2**, it combines the Inception architecture with residual connections. This hybrid model performs well in handling deep networks, which typically requires the extraction of intricate features from images.

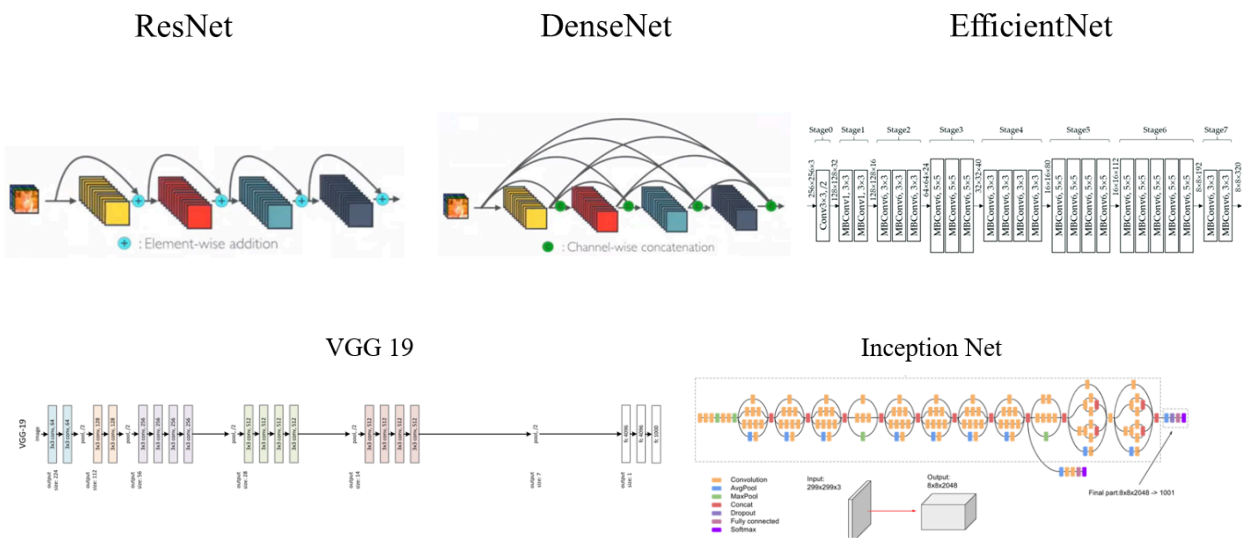


Fig.9 Architectures of the selected models

With each of these models having their own strengths and characteristics, we wanted to fully explore their capabilities by training with different parameters while fine-tuning them on our own PlantVillage datasets. We also aim to find out the differences between these models' performances and by evaluating them on testing datasets and additional datasets of online images.

Training Architecture

In this section, we will describe in detail the methodology and configuration of our deep learning model designed for the classification of tomato plant diseases. We prepared our data with a batch size of 128 and an image size of 256x256 pixels, with consistency across all images. A seed value of 42 was used for reproducible results across different runs. Additionally, we randomly selected 15% of the dataset for the validation use.

Before sending the input data into CNN models, data augmentation is used for improving model robustness and performance. We implemented the ImageDataGenerator class for this purpose, using transformations such as rotation, width and height shifts, shear, zoom, and brightness adjustments. By applying variations in the original dataset, the model is less likely to overfit the dataset and more likely to perform well on the unseen data in the real world. Figure 7.shows the augmentation result performed on the dataset.

For each of the models we explored, we use the base architecture with a pre-trained model weights with proven efficacy in image classification tasks. This allowed us to benefit from transfer learning, where we fine-tuned a model already trained on a large and diverse dataset (ImageNet). We added a GlobalAveragePooling2D layer and a Dense layer with 9 units, and a softmax activation function to tailor the model to our classification task. An option to freeze layers of the base model was also included, this option could offer flexibility in training strategies. We could freeze the specific layers to let the model learn the feature adaptively.

We used Adam optimizer through the training process. Adam has proven to perform well on various datasets and large models in the past. It adjusts the learning rate for each parameter

individually, based on the first and second moments of the gradients. This means it scales the learning rate by how quickly or slowly a parameter is changing, which makes training more efficient.

The models were trained for 50-200 epochs with early stopping, this balanced the need for accuracy with computational efficiency. We recorded and plotted both accuracy and loss using Matplotlib during training to monitor model performance and make necessary adjustments. Additionally, we evaluated the model on different test sets by measuring accuracy and inference time. The confusion matrix is also plotted to visualize the testing accuracy for our additional test sets consisting of several plant images we found online. We will introduce it later in the following section.

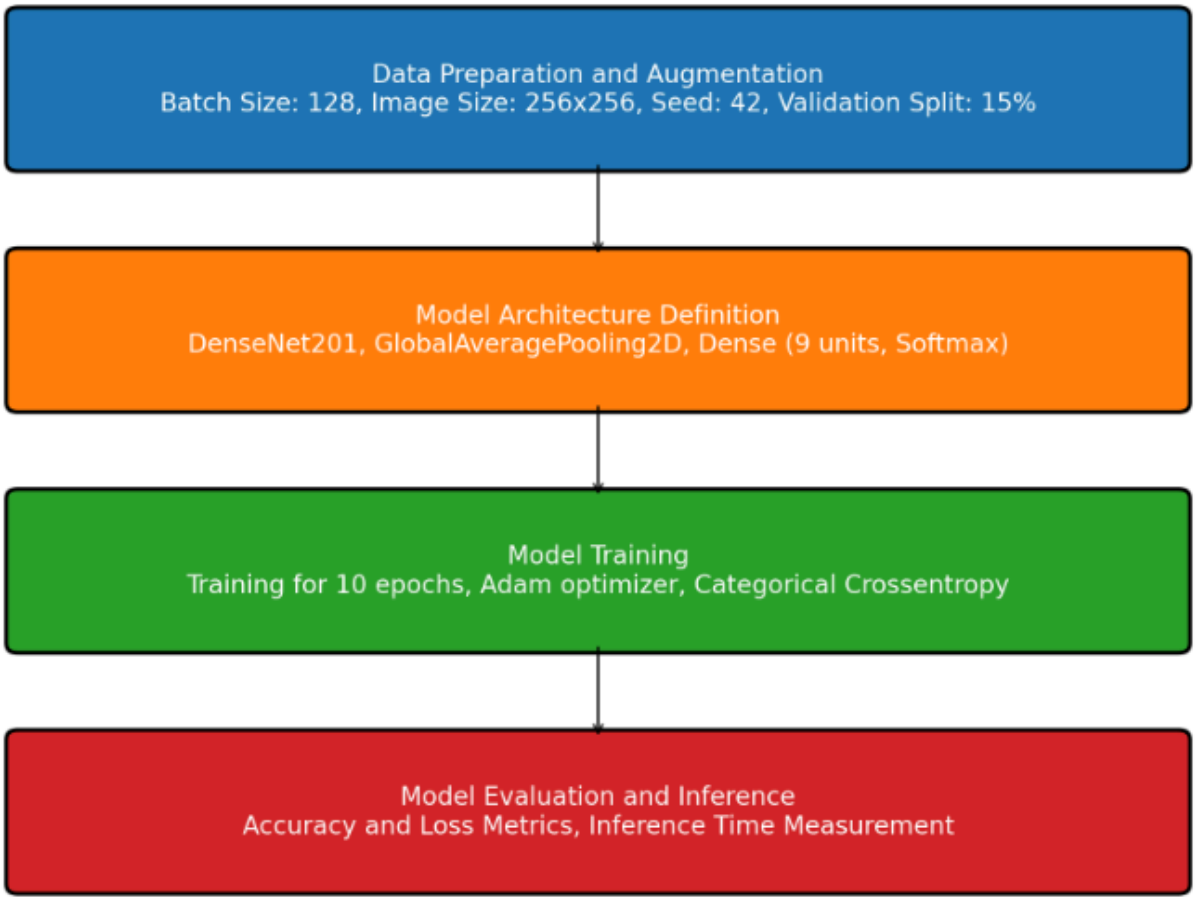


Fig.10 Training Architecture of DenseNet201 for Plant Disease Classification

Additional Online Image Testing Dataset

Alongside the traditional split of training, validation and test split of data. We further introduced a manually created testing dataset using images we searched from online public sources, and we call it the online image testing dataset. We created this dataset with the purpose of testing the generalizability of our trained model – how well is it able to perform on inputs that could potentially be very different from the training set, and more similar to real life inputs.

Fig.11 Samples from the online image testing dataset

This dataset illustrated in Fig. 11, created from various online public sources, contains a diverse range of images that differ in lighting, background, and image quality compared to the standard PlantVillage dataset. It can be seen that the background of such images can either be dark, light, blurred, or with mixed objects. Some images even contain watermarks that may mimic some noise that we may see from real life data (water partially covering the leaf, etc.). By testing our models on this dataset, we assess their ability to generalize to more realistic scenarios like those

we experience in practical agricultural settings. This evaluation is critical in ensuring that our model is not just theoretically accurate but also practically usable when conditions are often far from ideal and vary significantly in real-world settings. We also expect to have comparatively worse performance on this dataset compared to the original test dataset when evaluating our models due to such differences. However, the results from this experiment will provide valuable insights into the model's adaptability in real life settings and suggest directions where we need to further work on.

Results

Base Model Selection - Different architectures

We initially examined six transfer learning models: EfficientNetB1, ResNet101, DenseNet121, VGG16, InceptionV3, and InceptionResNetV2. To identify the model that performed best for our plant disease detection task and enhance training speed, we resized the input shape of the image data from 256 to 75, and unfreeze all layers. We employed a batch size of 256 and utilized an Adam optimizer to expedite the training process. Each model included an average pooling layer followed by a dense layer for output. The training duration for each model was set at 50 epochs.

The table below shows the training results of these three models using our training dataset. The training data is composed of randomly arranged samples from 72.25% of the plant-village dataset, while the validation data consists of randomly arranged samples from 12.75% of this dataset. The remaining 15% is designated as test data. Additionally, the trained models are evaluated on another dataset called Plant Disease Expert, which contains different but similar images and labels compared to the Plant Village dataset used for training. This evaluation aims to assess the model's ability to generalize.

Table.1 Training Result of Various Architectures Base Model

Model	**Training Accuracy**	**Training Loss**	**Test Accuracy**	**Inference Time (s)**	**Test Accuracy on Plant Disease Expert**
EfficientB1	0.999	0.003	98.5%	0.26	92.9%
ResNet101	1.000	1.1e-4	94.9%	0.36	88.0%
DenseNet121	1.000	9e-6	99.0%	0.28	95.1%
VGG16	0.991	0.035	92.4%	0.65	84.0%
InceptionV3	1.000	9e-6	98.3%	0.54	87.3%
InceptionResNetV2	1.000	2e-6	96.8%	0.49	86.2%

As shown in the table, all models, except VGG16, exhibit remarkably high training accuracies ranging from 99% to 100%. This indicates their excellent grasp of the training data. The training losses are also exceptionally low across all models, especially for DenseNet121, InceptionV3, and InceptionResNetV2 with losses on the order of 1e-6. This further validates these models' proficiency in effectively fitting the training data.

The test accuracies vary among the models, with DenseNet121 achieving the highest accuracy of 99.0% and VGG16 performing the worst at 92.4%. It is worth noting that even though some models (such as ResNet101, InceptionV3, and InceptionResNetV2) have perfect training accuracy, their test accuracies are lower. This difference implies that these models may not generalize well to unseen data. On the other hand, DenseNet121 demonstrates excellent generalization ability due to its high test accuracy despite also having a perfect training accuracy.

When evaluated on the Plant Disease Expert dataset, all models show a noticeable decrease in accuracy compared to the training dataset. DenseNet121 achieves the highest accuracy of 95.1%, demonstrating strong generalization abilities.

EfficientB1 has the shortest inference time per sample (0.26 seconds), closely followed by DenseNet121 (0.28 seconds). The slightly longer inference time of VGG16 may be attributed to its deeper architecture. Despite its high accuracy, DenseNet121 maintains a relatively low inference time, which is advantageous.

Overall, DenseNet121 seems to be the top-performing model in terms of accuracy and the balance between training loss and inference time. It has achieved the highest test accuracy with a minimal training loss, while its inference time remains competitive.

Base Model Selection - Different DenseNet size

We then evaluated three types of DenseNet: DenseNet121, DenseNet169, and DenseNet201.

Table.2 Training Result of Different DenseNet Size Base Model

Model	**Training Accuracy**	**Training Loss**	**Test Accuracy**	**Inference Time (s)**	**Test Accuracy on Plant Disease Expert**
DenseNet121	1.000	9e-6	99.0%	0.28	95.1%
DenseNet169	1.000	2e-5	98.9%	0.30	93.4%
DenseNet201	1.000	1.9e-5	99.0%	0.36	94.5%

As indicated in the table above, all three models have attained a flawless training accuracy of 1.000 or 100%. This implies that they have acquired an exceptional understanding of the training dataset. The training losses are remarkably low (ranging from 1e-5 to 9e-6), indicating a strong alignment with the training data.

The test accuracies of DenseNet121 and DenseNet201 are equally high, both at 99.0%. However, DenseNet169 has a slightly lower accuracy of 98.9%. These minimal differences suggest that all models perform well when faced with unseen data. The fact that the test accuracies closely match the training accuracies and are exceptionally high is a positive indication of the models' ability to generalize effectively.

When the Plant Disease Expert dataset was evaluated, which is assumed to be different from the training and initial test datasets, all models experienced a noticeable decrease in performance. This drop in performance is commonly observed when models are exposed to new data distributions. Among the three models, DenseNet121 achieves the highest accuracy of 95.1% on this external dataset, indicating that it has superior generalization capabilities. On the other hand, DenseNet169 exhibits the most significant decline in performance (to 93.4%), suggesting that it may not generalize as effectively as the others in this particular context.

The inference times increase as the models become more complex. DenseNet121 is the fastest, taking 0.28 seconds, followed by DenseNet169 at 0.30 seconds, and DenseNet201 being the slowest with a time of 0.36 seconds. This increase in inference time is expected because DenseNet169 and DenseNet201 have more layers than DenseNet121, making them computationally heavier and slower in processing.

Overall, in terms of accuracy, the performance of all three models is quite similar. However, DenseNet121 stands out as it offers the best generalization and strikes a good balance between high accuracy and lower inference time. Therefore, we ultimately chose it as our base model.

Model Architecture - Output Layer Design

In this section, various designs between the base model and the output layer were explored, and the results are shown in the table below.

Table.3 Training Results of Different Output Layer Exploration

Layer After Basemodel Before Output Layer	**Training Accuracy**	**Training Loss**	**Test Accuracy**	**Inference Time (s)**	**Test Accuracy on Plant Disease Expert**
None	1.000	9e-6	99.0%	0.28	95.1%
Dense256	1.00	9e-6	99.3%	0.43	93.2%
Dense256+Dense64	1.00	2e-6	98.9%	0.27	93.5%
dropout0.5	1.00	6e-6	98.8%	0.27	94.0%
dropout0.3	1.00	4e-6	98.8%	0.28	94.7%
dropout0.1	1.00	2e-6	98.9%	0.28	95.3%
dense256+dropout0.5	0.998	0.005	94.6%	0.28	89.7%
dense256+droupout0.3	1.00	4e-6	99.0%	0.28	94.7%
dense256+dropout0.1	1.00	6e-6	99.2%	0.29	94.7%
dense128+dropout0.3	1.00	2.6e-5	98.5%	0.29	94.8%

The bare model performs exceptionally well with perfect training accuracy and very high test accuracy on both the standard test set and the Plant Disease Expert dataset. It also has a fast inference time. Adding dense layers (Dense256, Dense256+Dense64) generally maintains high performance but slightly increases inference time. However, the test accuracy on the Plant Disease Expert dataset drops, indicating potential overfitting to the training data. Incorporating dropout improves generalization as seen in increased test accuracy on the Plant Disease Expert dataset, especially with dropout rates of 0.3 and 0.1. This suggests that dropout helps prevent overfitting. Combining dense and dropout layers results in varied performance. While some configurations like dense256+dropout0.1 maintain high accuracy, others (e.g., dense256+dropout0.5) show a notable drop in performance, particularly on the Plant Disease Expert dataset.

In general, dropout layers appear to improve generalization without significantly affecting inference time. The addition of dense layers increases model complexity and can result in overfitting, as evidenced by the decreased accuracy on the Plant Disease Expert dataset. Therefore, we concluded that either not adding any extra layers or only including a dropout rate of 0.1 is a good choice for our tasks.

Evaluation on BaseModel

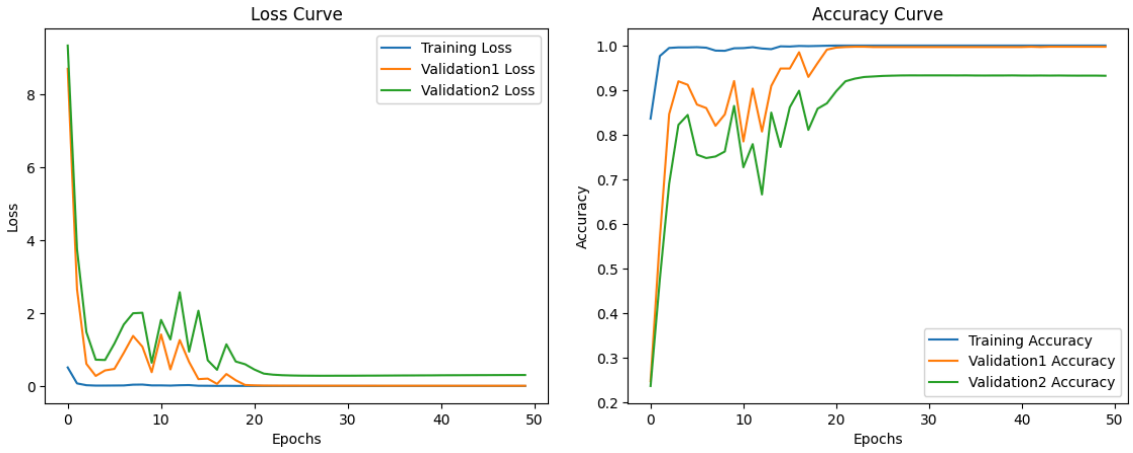


Fig12. Loss Curve and Accuracy Curve of DenseNet121

The figure above displays the accuracy and loss curves of a DenseNet121 model trained on a dataset and validated using two separate validation sets. Validation set 1 is the standard validation set derived from the training dataset prior to training, while validation set 2 is an external dataset called Plant Disease Expert that contains similar data.

As shown in the Figure, the training accuracy quickly reaches a high level and remains close to 1.0 throughout the training process, indicating that the model fits the training data well. The accuracy for Validation1 exhibits some volatility in the early epochs but stabilizes around epoch 20, suggesting that the model eventually generalizes well to the validation data used during training. The accuracy for Validation2 (Plant Disease Expert dataset) is lower than that of Validation1, as expected since Validation2 represents a different data distribution on which the model was not trained. The model's accuracy on Validation2 fluctuates more, possibly indicating

less certainty about this unseen data. However, it improves and stabilizes as training progresses, which is a positive indication of the model learning to generalize from its training data to this new dataset.

The training loss sharply decreases and remains consistently low after the initial epochs, indicating a strong fit to the training data. The high training accuracy further supports this observation. The validation loss for Validation1 also decreases and stabilizes, reflecting improvements in accuracy. However, it does not reach the same low level as the training loss, which is expected since the model is optimized specifically for the training data. On the other hand, the validation loss for Validation2 starts off high but gradually decreases over time. This suggests that the model is progressively improving its predictions on the Plant Disease Expert dataset as it learns. The higher loss compared to Validation1 could be attributed to differences in data distribution and potentially more challenging or diverse examples present in Validation2.

In general, there doesn't appear to be any noticeable overfitting to the training data as the validation losses decrease and stabilize instead of increasing or diverging from the training loss. The model demonstrates its ability to generalize to unseen data, as indicated by the improvement observed in Validation2 across epochs.

Additional Generalization Test using Online Images

In practical implementation, we utilized our trained model to predict plant images available online. We conducted an evaluation of our model's generalization accuracy using two benchmarks: Firstly, the model's capacity to determine whether plant images are healthy or unhealthy; secondly, its ability to accurately diagnose the specific disease affecting unhealthy plants and also recognize the healthy one. For this purpose, we curated a dataset of 35 plant images sourced from online public resources. These images were manually annotated and then fed into our trained model to assess its performance against both benchmarks. In the first benchmark, our model demonstrated a 100% accuracy rate in classifying all unhealthy images, while it achieved a 67% accuracy rate for healthy images. In the second benchmark, focused on identifying the correct disease types, the model has an accuracy of 51.4%.

Additional Test - Accuracy on healthy/unhealthy classification using Online Images

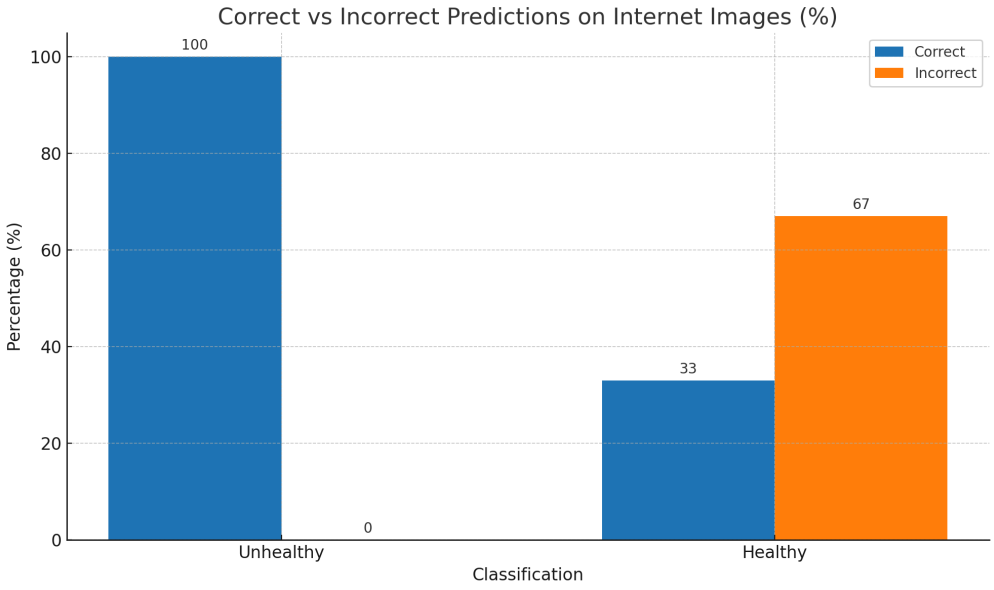


*Fig.13 Correct and Incorrect Predictions on **Online Images***

As illustrated in the Figure above, our model did not correctly classify all healthy plant images. This outcome can be attributed to several factors. Firstly, the original dataset used for training featured a consistent background across images, whereas the images sourced online for testing exhibited varied lighting conditions and backgrounds. These differences likely contributed to the misclassification issues observed in our trained model. Secondly, several of the images used in our evaluation contained watermarks and other forms of noise, which could have negatively impacted the model's predictive accuracy, resulting in a decrease in overall performance.

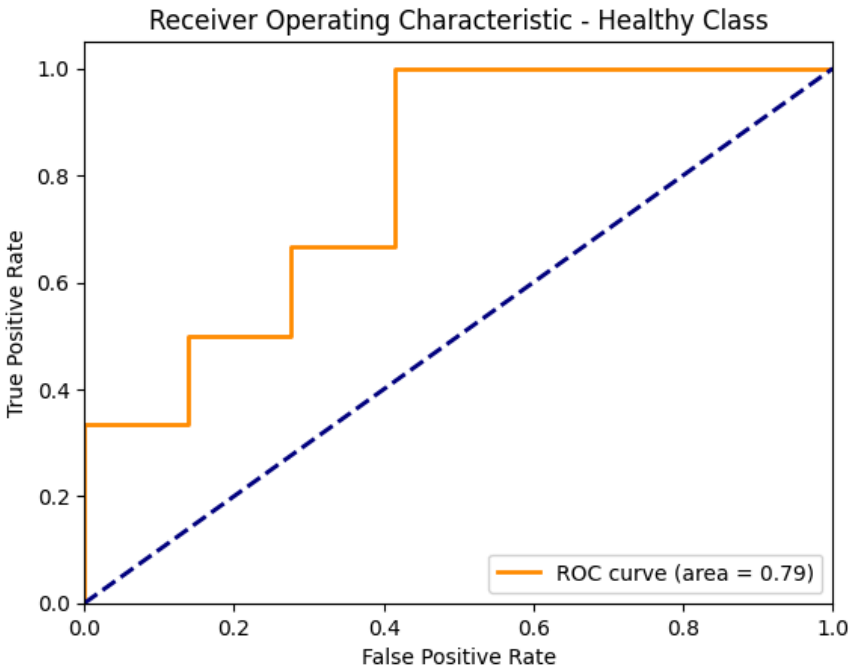


*Fig.14 ROC Curve of Healthy Class on **Online Images***

As shown in the ROC Curve above, the Area Under the Curve (AUC) is 0.79. This value ranges from 0 to 1, where 1 represents a perfect model and 0.5 represents a model with no discriminative ability (equivalent to random guessing). An AUC of 0.79 suggests that the model can effectively distinguish between the 'healthy' class and other classes on Online Images. The ROC curve appears steep at certain thresholds, indicating that the model achieves low false positive rates (1-specificity) while maintaining high true positive rates (sensitivity) for specific decision thresholds. For instance, if we intend to use this model as a quick pre-filter for identifying potential unhealthy plants, we would prioritize detecting all unhealthy cases even if it means accepting some false negatives but not tolerating any false positives. Therefore, we should set the threshold to achieve a TPR around 0.33 while keeping FPR at 0.

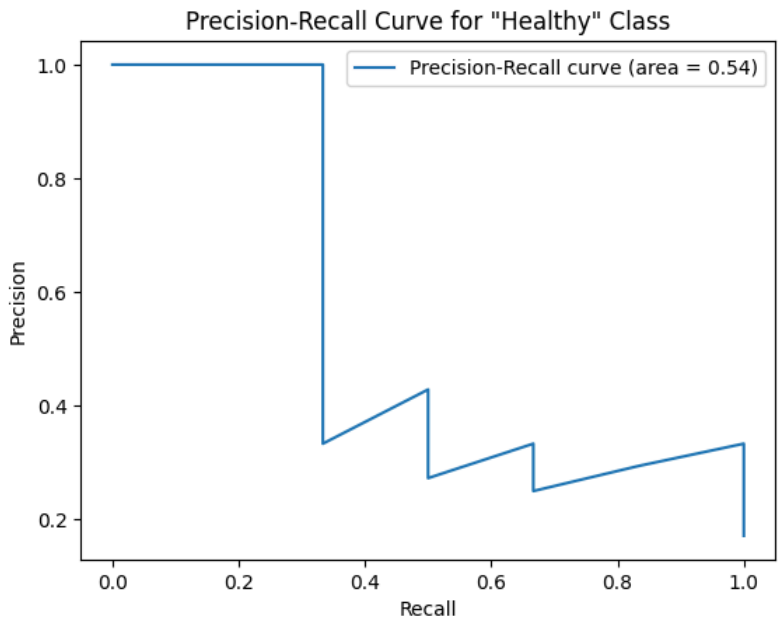


Fig.15 Precision-Recall Curve of Healthy Class on ***Online Images***

As shown in the Figure above, the AUC for this precision-recall curve is 0.54, which is relatively close to 0.5. This indicates that the model performs only slightly better than a random classifier for the 'healthy' class. At near maximum recall, there is a significant drop in precision, indicating that when the model tries to capture all healthy cases, it also makes many false positive errors. Conversely, at near maximum precision, recall is very low, suggesting that although predictions made for the 'healthy' class are highly accurate, many healthy cases are likely being missed. The shape of the curve suggests that achieving both high precision and high recall simultaneously poses a challenge for the model. This could be due to an imbalance in the training dataset where healthy samples comprise only around 1/10 of the total dataset.

Accuracy on specific disease classification using Online Images

In our result for predicting on specific disease classification, we obtained overall 51.4% accuracy on our additional test set. As shown in the following confusion matrix, our model can identify the healthy leafs and some diseases with obvious features, including mosaic virus, target spots, bacterial spots, and spider spots correctly. On the other hand, the model does not perform well on the dark blight and early blight. The difficulty our model faces in distinguishing between these

two diseases in tomatoes can be attributed to several factors, including the inherent similarity of symptoms between the two diseases, the potential risks of overfitting, and possible manually label errors in the dataset.

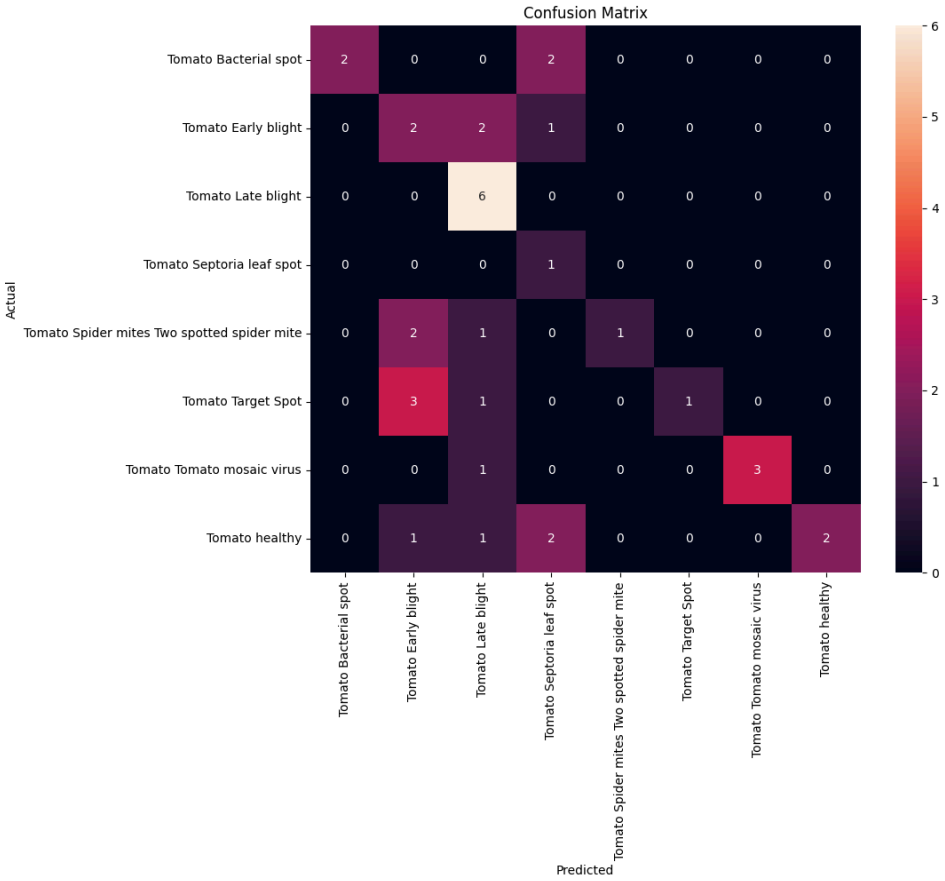


Fig.16 Confusion Matrix of Specific Disease Classification on ***Online Images***

Future Improvements

To improve the model's accuracy for real-world plant images, it is crucial to enhance the training dataset with diverse, high-quality images, ensure precise labeling, and possibly add additional contextual data using Generative Adversarial Network(GAN).

Conclusion

In conclusion, our study demonstrates a potential application of deep learning by demonstrating a potential use of CNNs and transfer learning in the early detection and classification of plant diseases. We used the PlantVillage dataset to train our models, which includes images of thousands of tomato leaves infected with different kinds of diseases. Since we found the dataset having imbalances and varying image quality, techniques like data augmentation and normalization were proven to be useful to enhance model training and generalization.

Our experiments with various existing models, including DenseNet, ResNet, and EfficientNet, revealed that DenseNet121 is the most suitable model in our use case, with a balance of high accuracy with reasonable inference time. While high accuracy is crucial to our application, inference time is also important since we expect to apply this model on a large number of images when used in agriculture settings.

However, the varying performance of our model on generalization tests have shown that we may need some additional future work. Due to the large variation of the images we found online and their inherent differences compared to our training dataset, the performance of our model is impacted during evaluation on such data. We suggest gathering more real-life based training dataset and exploring advanced techniques like GANs in order to further improve the model’s capabilities to work in real-life based settings.

In the end, our project suggests a possible application of deep learning in agriculture. There could be countless possible solutions in the future where farmers can take advantage of the technology for crop management and automated surveillance, which helps to improve food security in an increasingly unpredictable world. We sincerely hope that our project and works related to our project could be implemented in reality someday to truly make an impact.

Team Contributions

Our team distributed tasks evenly among all team members, from brainstorming, implementation, to creating this report. The percentage of work done by each team member is listed as below:

Yen-Hsing Li <yl2924>	33%
Yimian Liu <yl996>	33%
Zechen Wang <zw652>	33%

References

[1] Singh, Brajesh K., et al. "Climate change impacts on plant pathogens, food security and paths forward." Nature Reviews Microbiology (2023): 1-17.

[2] Li, Lili, Shujuan Zhang, and Bin Wang. "Plant disease detection and classification by deep learning—a review." IEEE Access 9 (2021): 56683-56698.

[3] Saleem, Muhammad Hammad, Johan Potgieter, and Khalid Mahmood Arif. "Plant disease detection and classification by deep learning." Plants 8.11 (2019): 468.

[4] Jackulin, C., and S. Murugavalli. "A comprehensive review on detection of plant disease using machine learning and deep learning approaches." Measurement: Sensors (2022): 100441.

[5] T. O. Emmanuel, "PlantVillage Dataset," Kaggle, Oct. 30, 2018. [Online]. Available: https://www.kaggle.com/datasets/emmarex/plantdisease. [Acce ssed: Nov. 10, 2023].

[6] T. O. Emmanuel, "Plant Disease Detection using Keras," Kaggle, Nov. 10, 2018. [Online]. Available: https://www.kaggle.com/emmarex/plant-disease-detection-using-keras. [Accessed: Nov. 10, 2023].

[7] S. S. Mahi, "Plant Disease Expert Dataset," Kaggle, Mar. 2023. [Online]. Available: https://www.kaggle.com/datasets/sadmansakibmahi/plant-disease-expert/. [Accessed: Nov. 10, 2023].

[8] FraMan, “Why do resnets avoid the vanishing gradient problem?,” Artificial Intelligence Stack Exchange, https://ai.stackexchange.com/questions/17764/why-do-resnets-avoid-the-vanishing-gradient-problem (accessed Dec. 10, 2023).