National College of Ireland

Plant Disease Detection Using Machine Learning in a Serverless Environment

MSc Research Project
MSc in Cloud Computing

Peng Yu
Student ID: 22196242

School of Computing
National College of Ireland

Supervisor: Jorge Mario Cortes Mendoza

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Peng Yu
Student ID:	22196242
Programme:	MSc in Cloud Computing
Year:	2024
Module:	MSc Research Project
Supervisor:	Jorge Mario Cortes Mendoza
Submission Due Date:	12/12/2024
Project Title:	Plant Disease Detection Using Machine Learning in a Serverless Environment
Word Count:	6178
Page Count:	23

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Peng Yu
Date:	29th January 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Plant Disease Detection Using Machine Learning in a Serverless Environment

Peng Yu
22196242

Abstract

Plant disease is a significant threat to food security, the crop loss caused by pests and diseases is up to 40% according to the United Nations. Compared to on-site checking by agricultural experts, Artificial Intelligence (AI) technologies especially Convolutional Neural Networks (CNN) have a great performance to detect plant disease. However, the seasonal and fluctuating characteristics of agricultural production especially for family farms require a more flexible, scalable, cost-effective solution. In this study, we proposed a plant disease detection system using Deep Learning (DL) in a serverless environment, which potentially helps smallholders recognize crop disease accurately and maintain an affordable budget. We trained and compared three models on top of three CNNs including VGG19, MobileNet-V2 and ResNet50. The evaluation result shows that the MobileNet-V2 has the best performance to detect diseases on vegetables, reaching a 98.82% accuracy. Then we compared the latency and cost by deploying fine-tuned MobileNet-V2 on AWS EC2 and AWS Lambda. The experiment allows us to define a threshold to estimate the type of service that reduces cost according to the number of requests and confirmed that the annual cost of Lambda is 88% lower than that of EC2 within the acceptable range of maintaining latency for the fluctuating agricultural requirement. Finally, our system has a mobile application, a MobileNet-V2 model deployed on the Lambda, providing accuracy, flexible, scalable, and low cost plant disease detection service for smallholder farmers. Theoretically, this system will contribute to food security, especially family farms that account for 90% of all farms globally.

1 Introduction

In the background of global climate change and population growth, maintaining food security and agricultural sustainability is one of the most crucial challenges. Plant disease is a significant threat to food security, the crop loss caused by pests and diseases is up to 40% according to the Food and Agriculture Organization of the United Nations. This can lead to catastrophic results for smallholders who live on farming, as 90% of farms are family-owned, and they produce 80% of the world's food, as highlighted by Lowder et al. (2021). Thus, detecting plant disease in time and accurately is the key to maintaining food security and agricultural sustainability.

Traditionally, plant detection relies on on-site checking and the experience of agricultural experts. As the rapid development of Machine Learning (ML) especially the breakthrough advancements of Deep Learning (DL) technologies in computer vision, plant

disease detection based on image recognition posed a great potential opportunity to solve this problem.

In recent years, a lot of research has shown that DL models based on Convolutional Neural Networks (CNN) have a great performance in plant disease detection. According to Shoaib et al. (2023), these models not only can classify the known diseases but also detect early diseases of plant, which provides very important support for farmers to take action as early as possible.

At the same time, the rise of cloud computing especially the serverless computing paradigm enables the flexible and scalable deployment and serving of those detection models. Allowing developers to build and run their applications without managing the server, serverless computing has the advantages of on-demand provisioning, auto-scaling and low operating costs, which are very suitable for farmers especially smallholders because of the seasonal and fluctuating requirements of agricultural production. It laid a technical foundation for developing scalable and low-cost plant disease detection services.

Even though existing research presented the potential capabilities of plant disease detection using DL, and the deployment of models on serverless computing, there are still some limitations. On one hand, most research focus on the development of algorithms and processes to train models, it lacks the consideration of deploying models in practical agricultural environments. On the other hand, the rise of plant disease has characteristics of seasonality and suddenness, the detection system needs to be able to adapt to load change and maintain a low cost.

Research Question. How can we design and implement a system using DL and serverless computing to achieve accurate, efficient, scalable and economic plant disease detection?

To answer this question, this project will focus on solving the following key challenges:

- Which transfer learning model among VGG19, and MobileNet-V2, ResNet50 is the best for plant disease detection?
- Is the serverless environment a cost-efficient solution than the cloud virtual machine?
- How to deploy plant disease detection models in the resource-constrained serverless environment?

Through solving these challenges, the following contributions are expected:

- Choose a suitable transfer learning model that can detect plant diseases with high accuracy;
- Implement a system for plant disease detection based on DL in a serverless environment and maintain cost-efficiency and acceptable latency;
- Develop a user-friendly mobile application for farmers to diagnose plant disease conveniently.

These contributions would not only push the boundaries of serverless computing and DL in agricultural applications but also provide a reliable, scalable technical solution for

farmers especially smallholders to reduce the loss of plant disease, ultimately contributing to the global food security.

The remainder of this report is structured as follows. Section II reviews related studies, ML in plant disease detection, serverless computing, and ML models in serverless computing environments. Section III discusses the research methodology. Section IV talks about the design specification of this study, the application of transfer learning, and the implementation of serverless architecture. Section V details the implementation of this project. Section VI represents experimental results of three CNN models and the costs and latency of serving plant disease models in AWS Lambda and AWS EC2. Section VII concludes and discusses feature works of this study.

2 Related Work

In this section, we review the latest developments in the scoop of ML technologies to detect plant diseases in a serverless environment. Through the review and analysis of existing literature, we establish a solid foundation for this study and identify the research gaps and opportunities.

We focus on three aspects: Firstly, we explore the application of ML in plant disease detection, including the advantages and disadvantages of different algorithms and their challenges. Secondly, we review the development of serverless computing, examine the deployment practices of ML models in the serverless environment, and analyze the advantages and disadvantages of existing methods. Lastly, we identify the research gaps and innovations of this study.

2.1 ML Application in Plant Disease Detection

In recent years, the application of ML technologies has had a significant advancement in plant disease detection. Among them, DL especially CNN has become the focus of related research due to its powerful feature extraction, pattern recognition capabilities and high accuracy. Furthermore, because of the lack of labelled plant leaf datasets, pre-trained models such as Visual Geometry Group (VGG) by Simonyan and Zisserman (2014), GoogLeNet by Szegedy et al. (2015), Residual Network (ResNet) by He et al. (2016) and transfer learning are widely used to improve the performance of CNN models on limited training data.

Mohameth et al. (2020) compared CNNs, VGG16, GoogLeNet, and ResNet50, on the Plant Village dataset for plant disease detection, finding that VGG16 has the best accuracy of 97.92, while GoogLeNet has a faster execution time. Additionally, researchers found that extracting features using the best classifier is more efficient than transfer learning as it has greater accuracy and less execution time.

For a cassava dataset from Tusubira et al. (2022), Alford and Tuba (2024) explored using pre-trained CNN models and transfer learning to detect plant diseases. Among ResNet101V2, ResNet50V2, EfficientNetB2, VGG16, VGG19, and MobileNet-V2, they found that VGG19 has better improvements and finally results in a training accuracy of 99.31% and a test accuracy of 80.27%. However, this research only focuses on cassava with an unbalanced dataset, and how to maintain performance across a wider range of plant types and datasets needs to be further studied.

A comparison of InceptionV3, CNN and YOLOv5 on soybean disease detection is carried out by Tirkey et al. (2023). Experimental results show that all models' accuracy

ranges from 85% to 99%, and YOLOv5 performs better compared to the other two. At the same time, the YOLOv5 has a fast execution time and is suitable for real-time detection. The authors also found that YOLOv5 has less computational complexity and a small size, making it possible for mobile applications. This is handy information for us as we also want to deploy our models on a resource-restricted serverless environment. However, they did not extend the experiments to other crops.

In the investigation, we found that most research focuses on enhancing the performance of ML models of plant disease detection, while little research is involved in putting forward a complete solution for disease detection in cloud computing or serverless environments. Joshi et al. (2023) presented an Android application that can detect plant diseases using an embedded CNN model within the application. It can achieve lower cost and inference speed but at the cost of model accuracy. Due to the limitations of phones' performance, embedded models need to be compressed. Furthermore, this solution is not suitable for frequent model updates. In the research, Khan et al. (2020) presented a DL model for automated plant disease classification deployed on an AWS DeepLens device, however, AWS DeepLens has already reached the end of its lifecycle. Neelaveni et al. (2023) proposed a cloud-based website that can detect plant diseases achieving an accuracy of 98.12%. Elijah et al. (2022) presented a system named chili Decision Support Platform (chili-DSP) which combined Internet of Things (IoT), cloud computing, and data analytics technologies to help farmers detect diseases. However, both of them do not explain cloud services and model deployment.

Through these studies, we found that there are potential gaps in design and implementation of a complete plant disease system using ML and cloud computing technologies, especially to deploy plant disease detection models at low cost, suitable for seasonal and fluctuating agricultural production. In the next section, we will explore the possibility of deploying ML models in a serverless environment, which may be able to meet our requirements.

2.2 The Deployment of Deep Learning Models in Serverless Environments

As a new computing paradigm, serverless computing attracts wide attention in both academia and industry. Castro et al. (2019) provided a comprehensive review of the concept, characteristics and challenges of serverless computing. They pointed out that serverless computing lowers the bar in application deployment, and promises real pay-as-you-go billing through on-demand provisioning. For seasonal and fluctuating requirements of agricultural production, this is an attractive reason. However, it also faces problems such as cold start delays and resource constraints.

With the increasing use of ML, cloud providers have all developed managed ML services for training and serving ML models, for example, AWS SagerMaker, Azure ML, and Google Vertex AI. However, according to the research of Wu et al. (2022), compared to managed ML services, serverless ML inference have better performance and is more cost-effective in most cases. They designed and implemented a benchmark framework to compare serverless and managed solutions. Take MobileNet model inferencing for instance, results show that the average latency of AWS SagerMaker is 71.6 times slower than AWS Lambda, while its cost is 8.56 times higher. From this point, it can be seen that serverless computing is more suitable for applications seeking the lowest cost.

Due to the natural characteristics of serverless computing such as constrained re-

sources, non-GPUs enabled, high latency of cold start, etc., it's not easy to serve ML models especially large-size deep learning models on serverless environments. Christidis et al. (2020) proposed some techniques to reduce the size of AI codebases so that it conforms to the limit of codebase package size of serverless environment, for example up to 250MB for AWS Lambda, without sacrificing model performance. The steps include trimming the dependent Python libraries, loading pre-trained ML models in runtime, using lightweight frameworks for inference, and performance tuning for data updates and reads. They successfully deploy their model and the evaluation results present that the response time for different volumes of prediction requests remains almost constant. Another method to push the limits of codebase package size is to use container images to create AWS Lambda functions, the size limit has been increased to 2G, which may result in a slower cold start.

In the research, Chahal et al. (2020) proposed their methods and implementation to migrate pre-trained large-size ML and DL models from cloud Virtual Machines (VMs) to serverless environments. They slimed unnecessary dependencies so that the codebase could fit the storage limit of AWS Lambda. On the other hand, they utilized Amazon Elastic File System (EFS) mounted on Amazon Elastic Compute Cloud (EC2) to store large models to reduce the latency of model loading. The experimental results show that the cost per inference in AWS Lambda is less than VMs while obtaining higher scalability and faster response. Although the cost per unit of inference has decreased according to this study, one EC2 instance is required to keep running all the time. For seasonal and fluctuating agricultural production especially smallholders, that's not an ideal solution.

Researchers also tried to use model partitioning to overcome the limitation of serverless resource constraints and computing capabilities. Jarachanthan et al. (2021) proposed a framework named Automatic Model Partitioning for Serverless Inference (AMPS-Inf) which can automatically perform model partitioning and serverless resource provisioning. It leads to a 98% cost saving without degrading the performance in comparison with Amazon SageMaker. This solution is suitable for large-size models but increases cost as it uses multiple serverless functions compared with a single point of deployment.

To test the performance of serverless for inference, Ishakian et al. (2018) deployed their MxNet DL model in AWS Lambda. Experimental results showed although execution latency could be within an acceptable range, longer delays due to cold start might lead to the risk of violating service-level agreements. Ali et al. (2020) believe that ML inference cannot benefit from serverless computing without batching. Then they proposed a framework named BATCH to identify the best parameter configuration such as memory size and batch size to improve the performance of serverless batch inference atop AWS Lambda. Results show that it decreases the cost by 50% while meeting user-defined Service Level Objectives. Another option to solve the latency of cold start is keeping the serverless service active. Ravi et al. (2023) proposed the use of a scheduler in Lambda, which sends dummy requests to Lambda to keep it warm. They compared both deploying AI models on AWS Lambda and EC2, finding that Lambda has an overall better performance in terms of cost, latency, and response time.

In this section, we investigated the potential capabilities and challenges of deploying ML models especially DL models in serverless environments. Although they do not focus on plant disease detection models, their works provide great insights for our research. In the next section, we will identify our research gaps and the expected contribution of our research.

2.3 Research Niche

Despite the potential of DL in plant disease detection, existing research mainly focuses more on improving the performance of models but less on deploying them in practical environments, especially in serverless platforms, to satisfy the requirements of strong seasonality and fluctuation of agriculture. Table 1 presents the comparison of existing literature in the aspect of plant disease detection, including models, datasets, evaluation metrics, and deployment environments. Those research focus less on the deployment of models.

Although serverless provides the benefits of flexible computing resources and an elastic payment mode, resource constraints and cold start latency restrict the use of plant disease detection applications, deploying models in a serverless environment requires further study. Table 2 compares studies in different literatures on the deployment of DL models in serverless environments, including model types, serving runtimes, sererless platforms, deployment optimization mesures, and evaluation metrics. A variety of model types were used in the studies, including MobileNet, VGG, ResNet50, ALBERT, Recurrent Neural Networks (RNNs), Deep Reinforcement Learning (DRL), Counter Terrorism Preparedness Network (CTPN), Inception-V3, Xception, Res-Net152V2, etc. Deployment platform involves AWS Lambda (Lambda), Google Cloud Functions (GCF), Amazon SagerMaker and Google AI Platform (GAP). Serving Runtimes includes TensorFlow (TF), and Open Neural Network Exchange (ONNX). Some of the studies optimizes deployment by reducing dependency libraries, loading models at runtime, and automatic partitioning to cope with resource constraints and cold start delays in serverless environments. The evaluation metrics mainly include latency and cost. These studies illustrates a variety of approaches to optimize the deployment of DL models on different serverless platforms, providing a reference for this study on how to achieve efficient inference in a resource-constrained serverless environment.

Based on the above analysis, this study is committed to filling these research gaps by proposing a plant disease detection system using DL in a serverless environment. As shown in the last row of Table 2, we focus on the transfer learning and performance comparison of three CNN models, the comparison of model serving in serverless (Amazon Lambda) and VM (AWS EC2) environments in terms of latency and cost. Additionally, we explore deployment optimizations, including ONNX runtime, library trimming, and the use of AWS Lambda layers. The ultimate goal is to provide a scalable and cost-effective solution for family farmers to detect plant diseases.

Table 1: Literature Comparison (Plant Disease Detection Models)

No.	Algorithm	Dataset	Metrics	Deployment	Author
1	CNNs, VGG16, GoogLeNet, and ResNet 50	Plant Village	Accuracy, F-score, Execution Time	No	Mohameth et al.
2	ResNet101V2, ResNet50V2, EfficientNetB2, VGG16, VGG19, and MobileNet2	Cassava Disease Classification	Accuracy, Precision, Recall, F1-Score, Support	No	Alford & Tuba
3	InceptionV3, CNN and YOLOV5	Self-Collected	Accuracy, mAP, Recall, and Precision	No	Tirkey et al.
4	ResNet-50, AlexNet, VGG16, VGG-19, DenseNet, SqueezeNet and DarkNet	Local Farmlands & Plant Village	Precision, Recall, Accuracy, and F1-Score	Amazon SageMaker	Khan et al.
5	MobileNet V2	Self-Collected	Accuracy and Loss	Not Mentioned	Neelaveni et al.
6	CNN	Self-Collected	Accuracy, Precision, Recall, and F1-Score	Cloud PaaS	Elijah et al.

3 Methodology

This study follows Cross-Industry Standard Process for Data Mining (CRISP-DM) method, ensuring that the project is implemented in a scientific and orderly manner. There are six steps in CRISP-DM workflow, including Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment.

3.1 Business Understanding

In the Business Understanding phase, the critical importance of early identification of plant diseases to improve crop yields was clarified. Agricultural production, especially smallholders, is suffering from severe food losses due to pests and diseases. According to the Food and Agriculture Organization of the United Nations, about 40 percent of global food loss each year is caused by pests and diseases, leading to over $220 billion cost. Traditional plant disease diagnosis heavily relies on on-site assessments by agricultural experts, which is time-consuming and costly, with limited support for smallholder farmers. Therefore, this study aims to implement a DL and serverless architecture-based plant disease detection system, which enables smallholder farmers to upload plant leaf photos through mobile application to detect crop diseases in a low-cost and efficient way, thereby reducing crop losses and improving food production efficiency.

Table 2: Literature Comparison (Deploying DL Models in Serverless Environments)

No.	Model Type	Serving Runtimes	Serverless Platform	Deployment Optimization	Evaluation Metrics	Author
1	MobileNet, ALBERT, VGG, MobileNet	TF and ONNX	Lambda, GCF, SageMaker, GAP	No	Cost and Latency	Wu et al.
2	RNNs and DRL	ONNX	AWS Lambda	Trimming dependent libraries and Loading models in runtime	Latency	Christidis et al.
3	CTPN	Not Mentioned	EC2 instances, AWS Lambda	Slimming unnecessary dependencies, Loading models in runtime	Latency and Cost	Chahal et al.
4	ResNet50, Inception-V3 and Xception	Not Mentioned	Lambda, SageMaker	Partitioning model	Latency and Cost	Jarachanthan et al
5	VGG19, MobileNet-V2, ResNet50	ONNX	Lambda, EC2	Trimming libraries, using AWS Lambda layers	Latency and Cost	Peng Yu

3.2 Data Understanding

PlantVillage dataset is chosen as the training data for our study. There are 14 kinds of plants, 38 categories of diseases and 54,305 instances in total. Among these samples, there are 8 kinds of fruits and 6 kinds of vegetables. Due to hardware resource constrains, we decided to use vegetables as our dataset. There is only 1 category of disease for soybean and squash, so we finally chose corn, pepper, potato and tomato, which is still a meaningful application for farmers to detect these vegetable diseases.

In order to better understand the characteristic of data and provide evidence for the following data preprocessing and model training, we conducted Exploratory Data Analysis (EDA), and the analysis content including dataset distribution, image size and format, brightness and contrast distribution. The results of the analysis are as follows:

Data Distribution. The dataset contains 26,639 samples across 19 disease categories and 4 types of vegetables. Figure 1 displays sample images from each category. Figure 2 illustrates the category distribution, revealing a significant imbalance. In particular, the 'Potato Healthy' and 'Tomato Mosaic Virus' are underrepresented. This imbalance suggests the need for data balancing techniques to ensure more robust model training.

Image Size and Format. The dimensions of all images in the dataset are 256x256

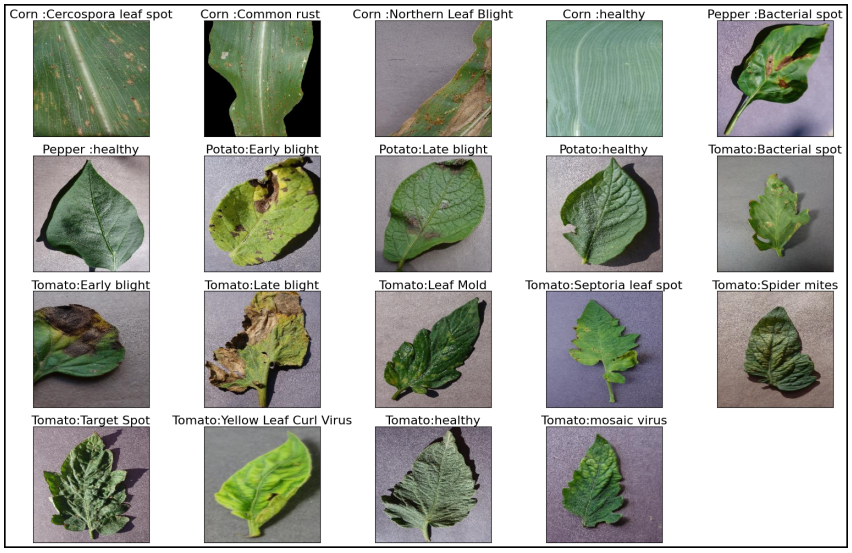

Figure 1: Samples per Class

pixels and are in JPEG format. Uniform image sizes and formats simplify the data preprocessing process, which means we don't need extra adjustment and format conversion to ensure the consistency of data entry.

Brightness Distribution. The average brightness of the image is concentrated between 75 and 150, as shown in Figure 3, showing an approximate normal distribution. This indicates that the overall brightness of the image is moderate, but there are some brightness fluctuations.

Contrast Distribution. The contrast of images is concentrated between 20 and 80, as shown in Figure 3, and the distribution is relatively balanced. The balanced contrast helps the model to stably extract effective information during the feature learning process without additional contrast enhancement.

3.3 Data Preparation

In the data preparation phase, image resizing and normalization operations need to be conducted according to the EDA results. Following the standard practice of pre training ImageNet models by Deng et al. (2009), we normalized the RGB channels using the mean ([0.485, 0.456, 0.406]) and standard deviation ([0.229, 0.224, 0.225]) of the ImageNet dataset to ensure compatibility with pre trained network weights and maintain consistent input distributions.

Due to the imbalance of the dataset categories, we implemented weighted sampling strategy through PyTorch's WeightedRandomSampler. As demonstrated by Zhang et al. (2021), oversampling can lead to overfitting of duplicate samples and increase the size of the dataset. On the other hand, undersampling discards valuable training data from the majority class. In contrast, weighted random sampling preserves the original dataset while achieving balanced class representation through probability-based sampling. By assigning higher sampling weights to the categories with small sample sizes, the model can fully learn the characteristics of these minority samples during training, avoiding the model biasing towards the majority sample categories.

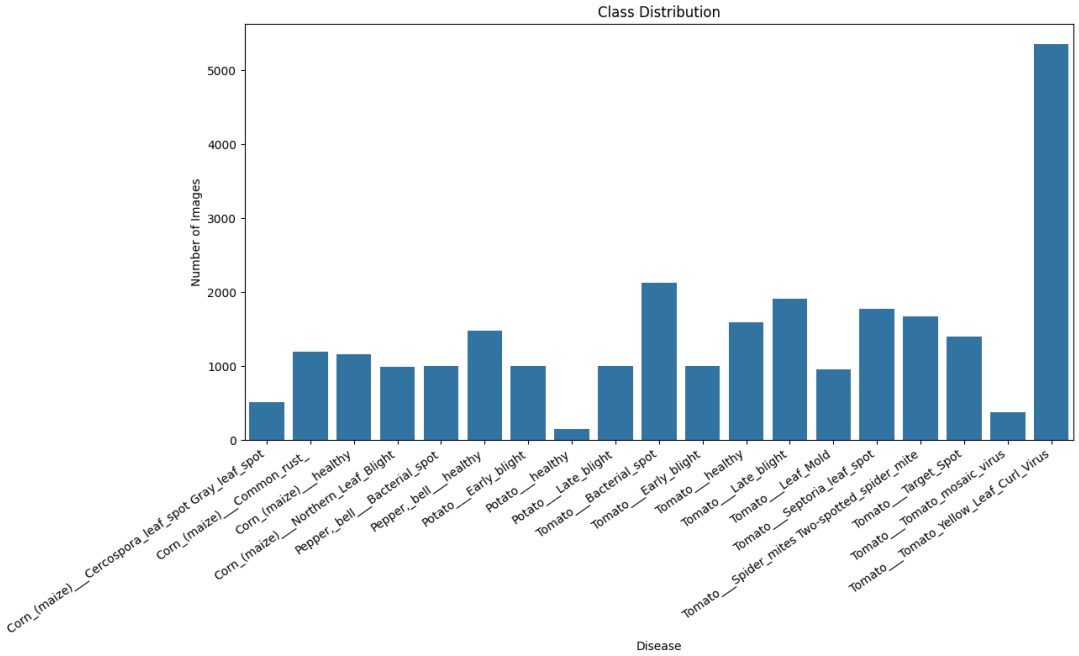

Figure 2: Class Distribution

3.4 Modeling

Due to the time and hardware resource limitation, this study selected three models representing deep traditional (VGG19), mobile-optimized (MobileNet-V2), and residual learning (ResNet50) network designs to provide a diverse representation of CNN architectures. These models are pre-trained on ImageNet dataset, so they can be directly applied to the PlantVillage dataset through transfer learning to conduct plant disease detection. As defined by Gupta et al. (2022), transfer learning is a way to use pre-trained models as a starting point for developing new models on different datasets.

3.5 Evaluation

There are two types of evaluation we need to conduct, including model performance and deployment environment evaluation.

3.5.1 Model Performance Evaluation

To comprehensively evaluate the performance of models, we adopted standard evaluation metrics, including Accuracy, Precision, Recall, F1-Score and Confusion Matrix. The notations used in these metrics are defined in Table 3. These metrics evaluate the model performance in different dimensions, helping us better understand the accuracy and robustness of models in plant disease detection.

Accuracy. Used to measure the overall correctness of model predictions by calculating the proportion of correct predictions out of all predictions.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

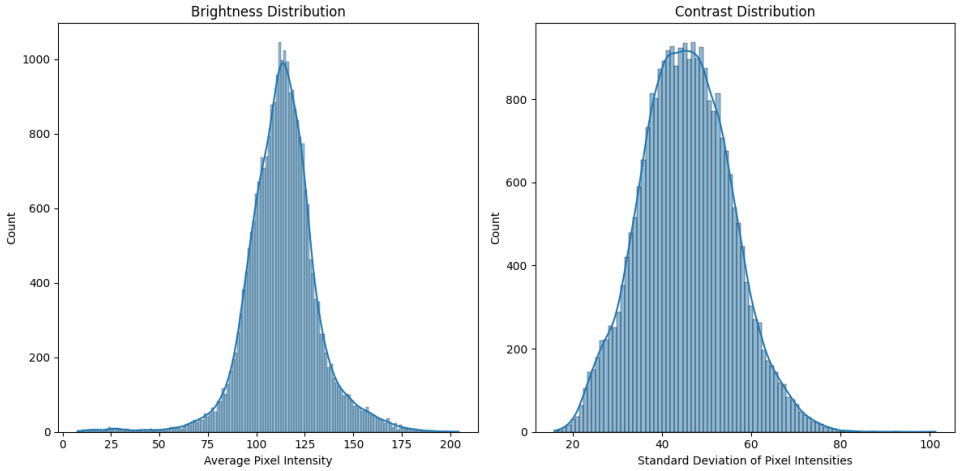

Figure 3: Brightness and Contrast Distribution

Notation	Definition
True Positives (TP)	The number of samples correctly predicted as a specific disease type.
True Negatives (TN)	The number of samples correctly predicted as not belonging to a specific disease type.
False Positives (FP)	The number of samples incorrectly predicted as a specific disease type when they belong to other types.
False Negatives (FN)	The number of samples incorrectly predicted as not belonging to a specific disease type when they actually do.

Table 3: Notation used in Evaluation Metrics for Multi-Class Plant Disease Detection

Precision. It indicates the proportion of diseases that the model predicts to be a disease that actually belong to that class, reflecting the accuracy of the model for positive classes. Higher precision means few false positives.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

Recall. It measures the proportion of correct positive recognition out of all actual positive samples, reflecting the sensitivity of the model. A high Recall presents that the model is able to identify disease samples more comprehensively.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

F1-Score. It's the average of Precision and Recall, used to comprehensively evaluate the performance of models on different categories, especially suitable for evaluations where there is a trade-off between accuracy and recall.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

Confusion Matrix. It displays the detailed prediction results of a model on all classes, including the correct prediction samples and misclassified samples for each category. We can understand categories in which our model tends to make mistakes through

the Confusion Matrix, which provides specific directions for subsequent model optimization.

3.5.2 Deployment Environments Evaluation

In the deployment phase, in order to evaluate the application effect of serverless architecture in plant disease detection, we deployed the best CNN model in AWS Lambda and AWS EC2 environments, respectively. Then we compared the cost and latency performance of two different serving environments by simulating a real-world agricultural production application scenario which includes off-peak season, regular season and peak season of use.

4 Design Specification

The plant disease detection system this study designed aims to provide a simple, economic and efficient disease detection tool for smallholder family farms. The design specification of this system focuses on two core factors: the application of transfer learning, and the implementation of serverless-based system. These factors can ensure the balance among performance, cost, and user usability, which helps farmers obtain accuracy and convenient plant disease detection service under the limited resource situations.

4.1 The Application of Transfer Learning

This system adopts transfer learning to implement the core functionality of plant disease detection. The choice of transfer learning can take advantage of models that are pre-trained on large-scale datasets such as ImageNet, from where we can obtain rich image features and then implement effective plant disease detection on limited agricultural dataset. This study chose three pre-trained VGG19, MobileNet-V2 and RestNet50, which can provide flexible options under different resource conditions and demand scenarios.

VGG19. Trained by Simonyan and Zisserman (2014), it has relatively strong feature extraction capabilities while has a higher computational cost and it's suitable for scenarios with sufficient computing resources.

MobileNet-V2. Trained by Sandler et al. (2018), it's a light weight network structure and designed for mobile devices for serverless environments. It has high computing efficiency and low memory footprint, is the preferred model for the deployment of this system.

ResNet50. Trained by He et al. (2016). By introducing the residual blocks, ResNet50 improved the stability of the deep network training and is suitable for scenarios that require high classification accuracy.

The application of transfer learning reduces the training time, as described by Gupta et al. (2022). Furthermore, by fine-tuning the model we can adapt its structure to the specific task of plant disease detection, avoiding the data and resource consumption of training from scratch.

4.2 Serverless-based System Architecture

In the design of the system architecture, the application of serverless architecture enables the system to gain significant advantages in terms of elastic scalability and cost con-

trol. As a serverless computing service, AWS Lambda supports scalability and billing on demand, which is particularly suitable for applications with large seasonal demand fluctuations in agriculture. Through the elastic scalability of AWS Lambda, the system can quickly scale the number of instances during peak demand periods to meet the large-scale demands while automatically scaled back to avoid unnecessary costs during low demand periods.

There are three layers in our design, Data Persistence layer, Business Logic layer and Presentation layer, as shown in Figure 4. The main components among different layers include the following:

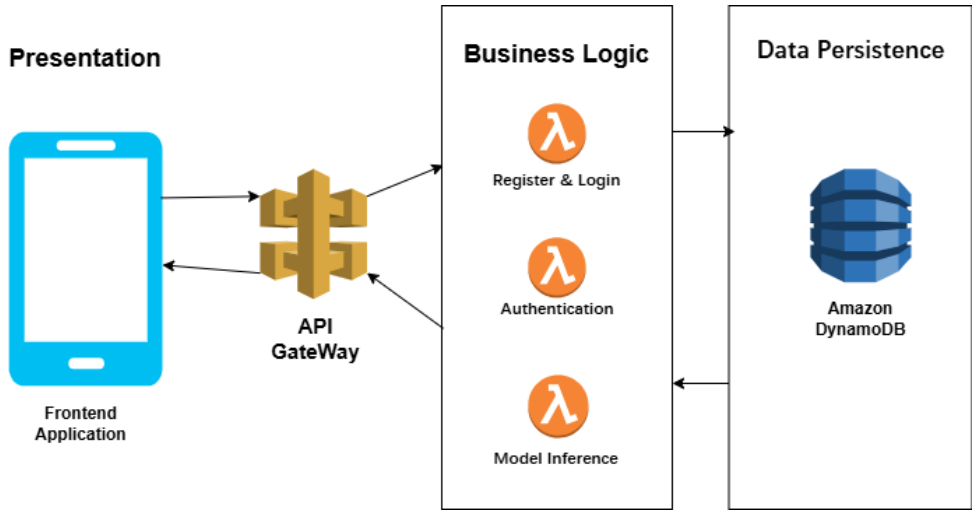

Figure 4: System Architecture

Frontend Application. The frontend of the system adopts the mobile application based on React Native to provide convenient user interaction methods.

Amazon API Gateway. As a unified entry point to the system, API Gateway receives HTTP requests from frontend clients and routes them to the corresponding Lambda functions. It also provides authentication and access control features to ensure the security of system.

AWS Lambda. Lambda functions are the computing cores of the system and there are three Lambda function deployed. Register & Login function is used to implement the user register and login feature, it combines the JSON Web Tokens (JWT) technology to generate and verify tokens to ensure the legitimacy of user identity. Model Inference function is used to serve the ONNX format plant disease detection model, it processes the images uploaded by users and infers quickly then send the results back. Authentication function is called by API Gateway before it routes requests to Model Inference function. It performs validity checks on the JWT token provided by users provide security for the protected API.

Amazon DynamoDB. This system uses DynamoDB as the database to store user data including username and hashed password. DynamoDB is high-performance and serverless which can cope with the read and write requirements in the high concurrency scenarios while maintaining a low cost.

5 Implementation

The implementation of this system consists of the following key steps: model training and evaluation, model deployment and frontend application development. Each step is designed and implemented in strict accordance with the design specification, ensuring the efficiency and stability of the system in practical applications.

5.1 Model Training and Evaluation

In the model training phase, this study fine-tuned and evaluated three transfer learning models, VGG19, MobileNet-V2 and RestNet50.

Transfer Learning Model Loading. All three CNN models use weights pre-trained on the ImageNet dataset. Most of the early layers are frozen and only the last layer is fine-tuned to apply them on the plant disease classification tasks. This transfer learning method makes efficient use of the feature extraction capabilities of pre-trained models and reduces the dependency on large-scale dataset.

Hyperparameter Setting. During the model training process, the cross-entropy is used as the loss function as it is the standard choice for classification tasks, particularly in multi-class classification problems, according to Goodfellow et al. (2016). Adam is used as the optimizer. According to Kingma (2014), Adam combines the advantages of momentum and adaptive scaling, resulting in faster convergence and robustness across various optimization problems. They also suggested a default learning rate of 0.001, which has been shown to work effectively for a wide range of deep learning tasks. A batch size of 32 is chosen to balance computational efficiency and model generalization. According to Keskar et al. (2017), small-batch training introduces noise in gradient estimation, helping models escape sharp minima and converge to flat minima, which improves generalization. Additionally, a batch size of 32 is computationally efficient for our constrained GPU resource. The value for the early stop was selected based on the unimproved performance (UP) criterion, as described by Prechelt (1998). Specifically, the method triggers early stopping when the validation loss fails to improve for 5 consecutive evaluations.

5.2 Model Deployment

In the deployment phase, we deployed the MobileNet-V2 in AWS Lambda and AWS EC2 two environments to compare the effect of serverless architecture with the traditional VM environments for agricultural application.

AWS Lambda Deployment. The mobileNet-V2 model is converted to ONNX format and is loaded by AWS Lambda function to infer. There are many benefits to use ONNX format such as the cross-platform compatibility, framework-agnostic transformations, easy deployment. For serverless environments, it can also reduce the dependency library size and thus decrease the cold start time. Lambda layers are also used in the system to manage and reuse the dependencies. To accept requests from users, AWS API Gateway is used to provide an HTTP interface, which can be called by users through the frontend application to do the real-time plant disease detection. Specifically, API Gateway routes user request to the Lambda function and Lambda function returns the detection result.

AWS EC2 Deployment. The MobileNet-V2 model with ONNX format is deployed as a RESTful API through the FastAPI web framework. EC2 provides an always-on

computing environment that is suitable for frequently invoked scenarios. The purpose is to compare the inference latency and cost efficiency of AWS Lambda and AWS EC2 for seasonal requirements of agricultural applications.

Base64 Encoding. Because AWS API Gateway automatically encodes the binary image file to Base64 format for Lambda to use, we decided to transfer Base64 encoded images from the frontend to the backend, which also makes it easier to just handle string format data. To ensure the model inference results will not be affected, this study conducted a comparison between model inferences on regular image and Base64 format image and the experiment shows that results in terms of the prediction, confidence score and accuracy are totally identical. Figure 5 presents the confidence score between regular image and Base64 formats.

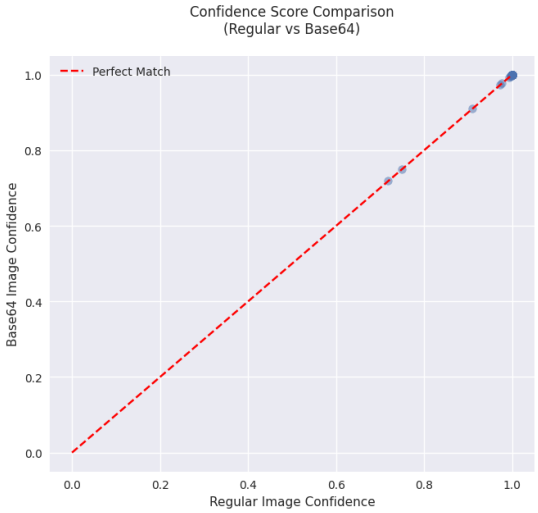

Figure 5: Comparison of Regular and Base64

5.3 Mobile Application

An React Native-based mobile application is implemented for farmers to use. Users can register and login through the application. After logging in, users can upload plant images to trigger the disease detection process. The detection result is presented in a simple way, including the predicted disease name and confidence level, helping farmers stay up-to-date on plant health. Figure 6 is the screenshot of the application.

6 Evaluation

There are two parts of evaluation in this study: the model performance evaluation and the deployment environment evaluation. The evaluation of model performance includes the comparison of Accuracy, Precision, Recall and F1-Score, while deployment environments evaluation focuses on the inference latency and cost between AWS Lambda and AWS EC2.

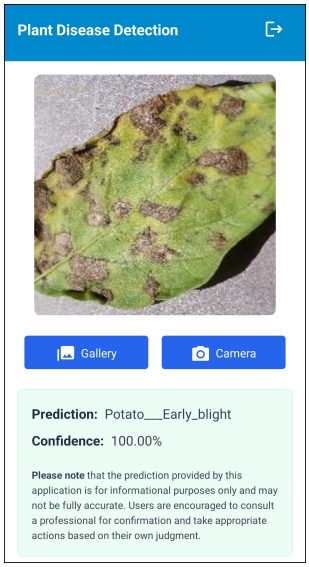

Figure 6: Mobile Application

6.1 Model Performance Evaluation

In the process of model training and validation, we use three classical CNNs, VGG19, MobileNet-V2 and ResNet50. The experiment results shows that the MobileNet-V2 excels in multiple performance metrics. The accuracy of the validation set reached 98.82%, and the training process completed in 24m 14s, which is significantly better than that of the other two models. Table 4 shows the results of the three models (values are percentages with % symbol omitted). 7 and 8 display the loss and accuracy line charts across epochs. 9 , 10 and 11 present the Confusion Matrix.

Table 4: Model Performance Comparison

No.	Algorithm	Accuracy	Precision	Recall	F1-Score	Training Time	Inference Time
1	VGG19	3.75	0.14	3.75	0.27	40m 21s	0.29ms
2	MobileNet-V2	98.82	98.85	98.82	98.82	24m 14s	0.56ms
3	ResNet50	97.94	98	97.94	97.91	54m 53s	0.53ms

The results show that the MobileNet-V2's performance is slightly higher than ResNet50 in terms of Accuracy, Precision, Recall and F1-Score. However, VGG19 failed to learn meaningful features and collapsed to predicting a single class (Potato Late blight), as shown in Figure 9, resulting in extremely poor performance across all metrics. At the same time, the mobileNet-V2 has a faster training speed compared to the other models.

6.2 Deployment Environments Evaluation

In order to validate the practical effect of serverless architecture in plant disease detection, this study deployed MobileNet-V2 model on both AWS Lambda and AWS EC2 environments and mainly focused on the performance of inference delay and cost. To ensure a fair comparison, t2.micro EC2 instances and Lambda with the same 1GB of memory

Figure 7: Loss and Accuracy Line Charts Across Epochs - MobileNet-V2

Figure 8: Loss and Accuracy Line Charts Across Epochs - ResNet50

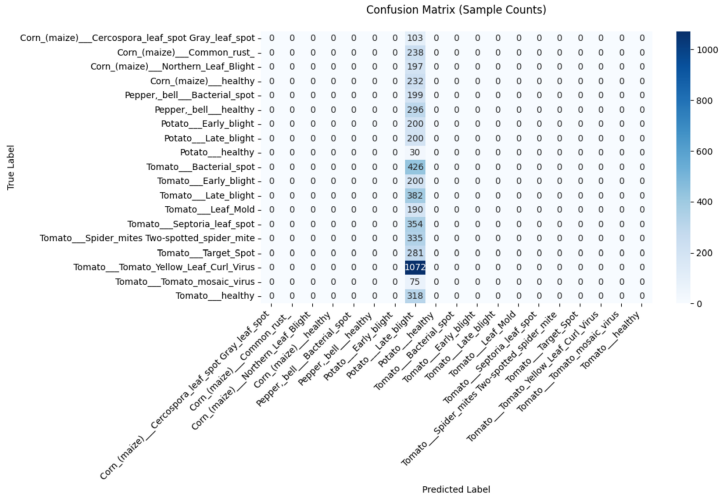

Figure 9: Confusion Matrix - VGG19

were selected. For **t2.mocro EC2**, it has 1 vCPU, 1G RAM, and its hourly Rate is $0.0126. For **Lambda**, it is configured to 1G RAM, its compute rate is $0.0000166667 per second and request rate is $0.00000020 per request.

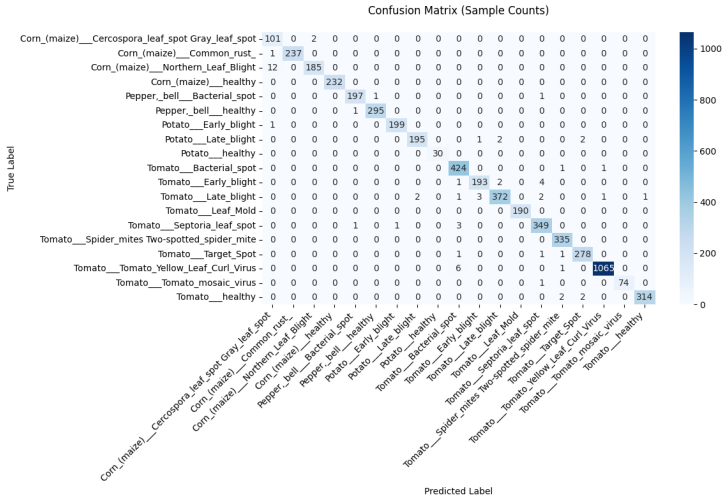

Figure 10: Confusion Matrix - MobileNet-V2

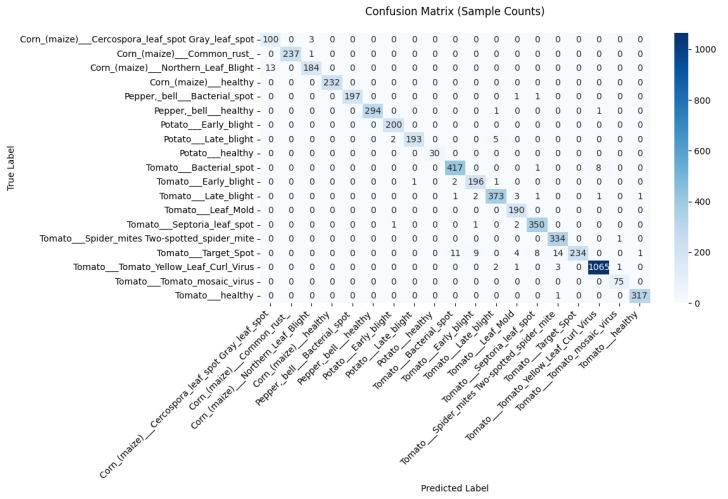

Figure 11: Confusion Matrix - ResNet50

We compared latency performance and cost by simulating seasonal loads in real-world scenarios. Based on the principles of agricultural production and our knowledge, we formulated a basic hypothesis: there are three seasons of use. During the **off-peak** season, there are 2 requests per hour, lasting for 4 months; in the **regular season**, there are 60 requests per hour, lasting for 6 months; and in the **peak season**, there are 3600 requests per hour, lasting for 2 months. Figure 12 shows the experiment results.

Cost Analysis. The annul total cost of EC2 is $108.86, because it's needed to run the server around the clock. While the annul cost of Lambda is only $12.76, approximately 88% savings over EC2. Even during the peak season (monthly cost EC2: $9.07 vs Lambda: $6.12), Lambda is more economic. However, our analysis presents that when the monthly requests exceed 3.85 million, approximately 5,277 requests per hour, EC2 becomes more cost-effective than Lambda. This threshold provides a clear guideline for choosing between the two services based on expected workload.

Latency Performance. The latency of EC2 is between 35 and 170 millisecond (ms),

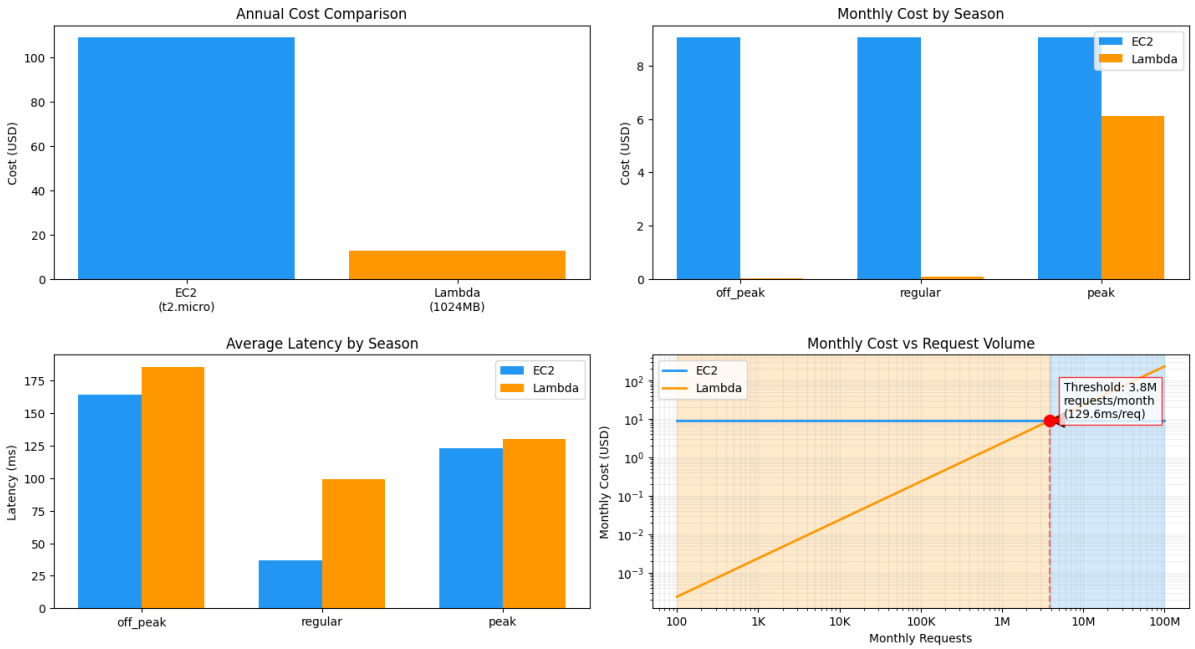

Figure 12: Latency and Cost Comparison: EC2 vs Lambda

134 ms on average. The latency of Lambda is between 100 and 180 ms, 139 ms on average, relatively higer then EC2. And both of these two solutions maintained a 100% success rate in all seasons.

Scalability. The scalability evaluates a system's ability to handle an increasing amount of work. In our test, the latency of EC2 increases gradually as the loads increase, indicating that the increased load affects performance. While Lambda, despite its high latency, is stable when the load changes.

All factors considered, Lambda is a better choice for workloads below the threshold of 5,277 requests per hour. Although Lambda has slightly higher latency, it has a significant cost advantage and scalability. For applications exceeding this threshold, EC2 provides both cost benefits and better latency performance.

7 Conclusion and Future Work

7.1 Conclusion

This study successfully built a plant disease detection system that combined transfer learning and serverless architecture, helping smallholder farmers diagnose plant diseases in a cost-effective and efficient manner to reduce the yield loss caused by plant diseases. Through the analysis and preprocessing, this study trained VGG19, MobileNet-V2 and ResNet50 three transfer learning models and comprehensively evaluated their performance on tomato disease detection tasks. MobileNet-V2 is the best choice due to its lightweight and high accuracy, whose accuracy reaches 98.57% on the validation set. It performs well in terms of the inference speed, and is suitable for the deployment in the serverless environment.

In order to balance cost and response time, the MobileNet-V2 model is deployed both

in AWS Lambda and AWS EC2 environments. Through comparison of experiments, the cost advantages of AWS Lambda in application scenarios with strong seasonal demand are confirmed. The pay-per-call billing model and on-demand scaling features of AWS Lambda reduces the operational cost for seasonal disease detection applications.

Overall, this study achieved a good balance between accuracy and economy, providing a convenient plant disease detection tool for smallholders. Through the React Native application, farmers can simply upload the plant leaf images and then obtain a real-time disease detection results, which is supposed to improve the production decision-making ability of farmers.

7.2 Future Work

In the future work, will plan to extend and optimize our system in multiple aspects. The first priority is to increase the size and diversity of the dataset, enabling the system to identify a wider range of crop disease categories to improve the practical value and scope of the model. In terms of the model optimization, we will introduce automatic parameter tuning technology, combined with the Cross-Validation training strategy, to obtain an optimal hyper parameter combination, so as to improve the generalization ability and prediction accuracy of the model. For system improvement, we plan to develop a detection history tracking function and allow farmers to submit feedback to help us continuously improve the model performance. Furthermore, we would like to explore the possibility of Large Language Model integration to provide scientific and practical disease prevention and control suggestions and solutions, which can potentially help us build a more comprehensive and intelligent service platform for plant disease detection and control.

References

Alford, J. and Tuba, E. (2024). Cassava plant disease detection using transfer learning with convolutional neural networks, *2024 12th International Symposium on Digital Forensics and Security (ISDFS)*, pp. 1–6.

Ali, A., Pinciroli, R., Yan, F. and Smirni, E. (2020). Batch: Machine learning inference serving on serverless platforms with adaptive batching, *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, IEEE, pp. 1–15.

Castro, P., Ishakian, V., Muthusamy, V. and Slominski, A. (2019). The server is dead, long live the server: Rise of serverless computing, overview of current state and future trends in research and industry, *arXiv preprint arXiv:1906.02888* .

Chahal, D., Ojha, R., Ramesh, M. and Singhal, R. (2020). Migrating large deep learning models to serverless architecture, *2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, IEEE, pp. 111–116.

Christidis, A., Moschoyiannis, S., Hsu, C.-H. and Davies, R. (2020). Enabling serverless deployment of large-scale ai workloads, *IEEE Access* **8**: 70150–70161.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database, *2009 IEEE conference on computer vision and pattern recognition*, Ieee, pp. 248–255.

Elijah, O., Rahim, S. K., Abioye, E. A., Salihu, Y. O., Oremeyi, A. A. et al. (2022). Decision support platform for production of chili using iot, cloud computing, and machine learning approach, *2022 IEEE Nigeria 4th International Conference on Disruptive Technologies for Sustainable Development (NIGERCON)*, IEEE, pp. 1–5.

Goodfellow, I., Bengio, Y. and Courville, A. (2016). *Deep Learning*, MIT Press. `http://www.deeplearningbook.org`.

Gupta, J., Pathak, S. and Kumar, G. (2022). Deep learning (cnn) and transfer learning: a review, *Journal of Physics: Conference Series*, Vol. 2273, IOP Publishing, p. 012029.

He, K., Zhang, X., Ren, S. and Sun, J. (2016). Deep residual learning for image recognition, *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Ishakian, V., Muthusamy, V. and Slominski, A. (2018). Serving deep learning models in a serverless platform, *2018 IEEE International conference on cloud engineering (IC2E)*, IEEE, pp. 257–262.

Jarachanthan, J., Chen, L., Xu, F. and Li, B. (2021). Amps-inf: Automatic model partitioning for serverless inference with cost efficiency, *Proceedings of the 50th International Conference on Parallel Processing*, pp. 1–12.

Joshi, R. C., Patel, V. R., Mishra, A. and Kumar, S. (2023). Real-time plant leaf disease detection using cnn and solutions to cure with android app, *2023 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, IEEE, pp. 455–460.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M. and Tang, P. T. P. (2017). On large-batch training for deep learning: Generalization gap and sharp minima.
URL: *https://arxiv.org/abs/1609.04836*

Khan, A., Nawaz, U., Ulhaq, A. and Robinson, R. W. (2020). Real-time plant health assessment via implementing cloud-based scalable transfer learning on aws deeplens, *Plos one* **15**(12): e0243243.

Kingma, D. P. (2014). Adam: A method for stochastic optimization, *arXiv preprint arXiv:1412.6980* .

Lowder, S. K., Sánchez, M. V. and Bertini, R. (2021). Which farms feed the world and has farmland become more concentrated?, *World Development* **142**: 105455.
URL: *https://www.sciencedirect.com/science/article/pii/S0305750X2100067X*

Mohameth, F., Bingcai, C. and Sada, K. A. (2020). Plant disease detection with deep learning and feature extraction using plant village, *Journal of Computer and Communications* **8**(6): 10–22.

Neelaveni, R., Jeevan, P., Rao, T. S., Shinde, S. K., Ravichandran, S. and Bobade, S. D. (2023). Cloud computing based advance system for detection of plant health, *Journal of Pharmaceutical Negative Results* pp. 135–142.

Prechelt, L. (1998). Automatic early stopping using cross validation: quantifying the criteria, *Neural networks* **11**(4): 761–767.

Ravi, B. S., Madhukanthi, C., Sivasankar, P. and Prasanna, J. D. (2023). A novel technique to improve latency and response time of ai models using serverless infrastructure, *2023 International Conference on Inventive Computation Technologies (ICICT)*, IEEE, pp. 428–433.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks, *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520.

Shoaib, M., Shah, B., Ei-Sappagh, S., Ali, A., Ullah, A., Alenezi, F., Gechev, T., Hussain, T. and Ali, F. (2023). An advanced deep learning models-based plant disease detection: A review of recent research, *Frontiers in Plant Science* **14**: 1158933.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition, *arXiv preprint arXiv:1409.1556* .

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A. (2015). Going deeper with convolutions, *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9.

Tirkey, D., Singh, K. K. and Tripathi, S. (2023). Performance analysis of ai-based solutions for crop disease identification, detection, and classification, *Smart Agricultural Technology* **5**: 100238.
URL: *https://www.sciencedirect.com/science/article/pii/S2772375523000680*

Tusubira, F. J., Nakatumba-Nabende, J., Babirye, C., Okao-Okujja, G., Mutebi, C., Mugalu, B. W., Nabagereka, D. and Namanya, G. (2022). Makerere University Cassava Image Dataset.
URL: *https://doi.org/10.7910/DVN/T4RB0B*

Wu, Y., Dinh, T. T. A., Hu, G., Zhang, M., Chee, Y. M. and Ooi, B. C. (2022). Serverless data science-are we there yet? a case study of model serving, *Proceedings of the 2022 international conference on management of data*, pp. 1866–1875.

Zhang, Y., Lei, Z., Zhuang, L. and Yu, H. (2021). A cnn based method to solve class imbalance problem in sar image ship target recognition, *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, Vol. 5, IEEE, pp. 229–233.