LaTeX Guidelines for Author Response

A. Abstract

Articulate on the abstract presentation of the project and what to expect by reading your report in full detail. Briefly discuss the problem, proposed methods and used data, and the achieved results. [maximum of 150 words]

B. Introduction

The agricultural industry is a vital part of the global economy, providing food and other essential resources to people all over the world. However, plant diseases can have a significant impact on crop yields and quality, leading to economic losses and food insecurity. [1] Accurate and timely detection of plant diseases is crucial for effective disease management and control. Traditional methods of plant disease diagnosis rely on visual inspection by trained experts, which can be time-consuming, subjective, and prone to errors. [2]

With the increasing availability of digital imaging technology and deep learning algorithms, there is an opportunity to develop automated plant disease classification systems that can assist farmers, researchers, and other stakeholders in the agricultural sector. In this project, we have developed a plant disease classification system that can accurately identify and classify different types of plant diseases using leaf images. The system uses various deep learning algorithms(MobileNet, ShuffleNet, and Resnet18) to learn the visual features of healthy and diseased leaves and classify them into different disease categories.

By providing a fast and accurate diagnosis of plant diseases, this system has the potential to revolutionize the way we manage plant diseases and increase crop productivity, leading to a more sustainable and food-secure future. The development of this plant disease classification system is of great importance to the agricultural industry, as it has the potential to significantly improve the accuracy, speed, and cost-effectiveness of plant disease diagnosis.By enabling farmers to detect and manage plant diseases more efficiently, this system can help reduce crop losses and increase yields, leading to a more sustainable and secure food supply for the growing global population.

There are several challenges associated with plant disease classification using image-based analysis. One of the main challenges is the high variability in the appearance of healthy and diseased leaves, which can make it difficult to distinguish between them accurately. Moreover, environmental factors such as lighting conditions, camera angles, and leaf position can also affect the image quality and accuracy of disease diagnosis. Another challenge is the availability and quality of labeled datasets for training and testing deep learning models. Collecting and annotating a large and diverse dataset of plant images is time-consuming and requires expertise in plant pathology. Additionally, the quality and consistency of labeling can vary depending on the annotator's expertise, leading to errors and inconsistencies in the dataset.

To address the high variability in the appearance of healthy and diseased leaves, several studies have used data augmentation techniques to increase the size and diversity of the training dataset. This can include techniques such as random rotation, cropping, and flipping, which can simulate different lighting and viewing conditions and improve the model's robustness to variations in the input images. Another approach to address the challenge of dataset availability is transfer learning, where a pre-trained deep learning model on a large and diverse dataset such as ImageNet is fine-tuned on the target plant disease dataset. This approach can significantly reduce the amount of training data required and improve the performance of the model, especially when the target dataset is small.

Deep learning algorithms have emerged as a promising solution for plant disease classification from leaf images. However, these approaches also have their own set of pros and cons. On the positive side, deep learning algorithms offer high accuracy in disease identification and classification, and with the availability of lightweight models and hardware accelerators, they can be trained and deployed efficiently on various devices. Additionally, transfer learning and data augmentation techniques allow deep learning models to be trained with small datasets, which can be beneficial when large datasets are not available. Furthermore, these approaches can reduce the cost of plant disease diagnosis by replacing the need for manual inspection by experts.

On the negative side, the accuracy of deep learning models is heavily dependent on the quality and size of the dataset used for training. A biased or incomplete dataset can result in poor generalization and performance of the model, and deep learning models may have difficulty generalizing to new and unseen plant diseases that were not present in the training dataset. Moreover, the use of large models and hardware accelerators can require significant computational resources and investment.

To tackle the problem of plant disease classification, we adopted a methodology that involved collecting various datasets of images portraying both healthy and diseased plant leaves from open sources. These datasets were then split into separate training, validation, and testing sets to facilitate accurate model training and evaluation. The images were pre-processed before training three state-of-the-art CNN models, namely MobileNet, Shufflenet, and ResNet-18 on the datasets of healthy and diseased plant leaves. The accuracy of the models was evaluated using different performance metrics. We also used hyperparameter optimization to enhance the performance of the least performing model. Additionally, we utilized transfer learning techniques to improve the accuracy of the models while using significantly fewer resources and time.

Our study achieved high accuracy rates for plant disease classification using deep learning algorithms for all three models. Specifically, our MobileNet, ShuffleNet, and ResNet models achieved accuracy rates ranging from 92.64% to 99.19% on various datasets of diseased and healthy plant leaves. These results demonstrate the potential of deep learning-based solutions for accurately identifying and classifying plant diseases, which could ultimately reduce the need for manual inspection by experts and improve the efficiency and cost-effectiveness of plant disease diagnosis.

B.1. Literature Review and Related Works

Numerous studies have been conducted in the field of plant disease classification using deep learning algorithms. Plant disease detection could be achieved by extracting shape features method. Patil and Bodhe (2011) [5] applied this technique for disease detection in sugarcane leaves where they have used threshold segmentation to determine leaf area and triangle threshold for the lesioning area, getting the average accuracy of 98.60% at the final experiments. In a study by Isharat et al. (2019) [3] , a transfer learning-based approach was employed for plant disease identification from leaf images. The authors fine-tuned the pre-trained ResNet-50 model on their dataset and achieved an accuracy of 99.80% in classifying grape diseases. Erika et al. (2016) [4] proposed a four layers CNN model which contains 7 types of diseases along with healthy cucumber leaves. They remarked good and bad condition of images and found an average accuracy of 82.3

C. Methodology

C.1. Datasets

Each of our three datasets offers a unique configuration in terms of the number of classes and the total number of images. All three datasets are publicly available on Kaggle.

C.1.1 Dataset1 - Plant Diseases Recognition Dataset

This dataset consists of 1530 images of plant leaves, each with a resolution of 4000x2672 pixels in JPG format. The images were collected using a digital camera from live plants without plucking their leaves on sunny and cloudy days. The dataset has three classes: ”Healthy”, ”Powdery”, and ”Rust”, which correspond to different plant conditions.

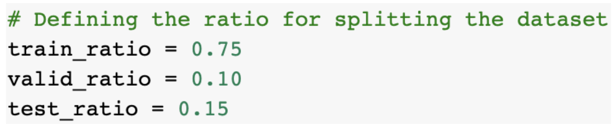
C.1.2 Dataset2 - PlantVillage Dataset

The PlantVillage dataset contains 20,600 images of tomato, potato, and pepper leaves, with each image having a default size of 256x256 pixels in JPG format. The dataset has 15 classes, consisting of healthy and diseased variants of the leaves. The images were captured outdoors using a digital camera on sunny or cloudy days, with the leaves removed from the plant and placed against a grey or black background.

C.1.3 Dataset3 - New Plant Diseases Dataset

The New Plant Diseases Dataset is a subset of the original PlantVillage dataset, consisting of 50,000 images in 23 different classes, with a default image size of 256x256 pixels in JPG format. The images were collected by photographing leaves of different plants against a grey or black background after removing them from the plant. The dataset was created using offline augmentation from the original PlantVillage dataset, which had 38 classes and 87,000 images.

While two of the datasets were pre-divided into train, test, and validation sets, we merged them into one and further split them into new train, validation, and test sets with a ratio of 0.75, 0.10, and 0.15, respectively, for all three datasets. This was done to ensure consistency across all datasets and to make sure that our models were not biased towards any particular subset of the data. The split ratios were chosen to ensure that we had enough data for training and validation while still having a separate test set to evaluate the performance of our models on unseen data. By resplitting the data, we were able to improve the generalization capability of our models and ensure that they could accurately classify both healthy and diseased plant leaves.



```
# Defining the ratio for splitting the dataset
train_ratio = 0.75
valid_ratio = 0.10
test_ratio = 0.15
```

Figure 1. Split Ratio

Before feeding into the deep-neural network pipeline, the images were preprocessed using a combination of resizing, conversion to PyTorch tensors, and normalization. Specifically, we preprocessed the images by removing any distortions or flaws and standardizing the pixel values to a common range. The images were resized to 224x224 pixels, and converted to tensors. This preprocessing step was important to ensure that the input to the deep neural network pipeline was consistent and normalized, allowing the network to learn more effectively. Additionally, the images were filtered to remove any images that were not relevant to the classification task and to ensure a balance of healthy and diseased plant images in the dataset.

Below are some examples of pre-processed images. Figures 4 and 5 display the images before and after pre-processing, respectively.

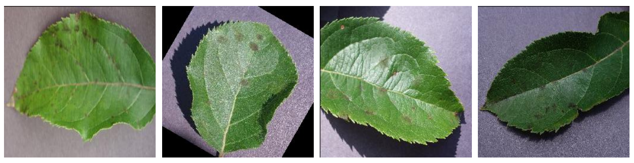


Figure 2. Before pre-processing

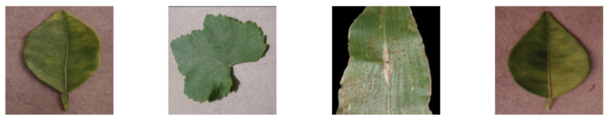


Figure 3. After pre-processing

C.2. CNN Models

For this project, we have used MobileNet [6], ShuffleNet [8], and ResNet-18 [7] which are three popular convolutional neural network architectures used in computer vision tasks.

C.2.1 ShuffleNet

Designed for mobile and embedded devices, ShuffleNet is a lightweight CNN architecture. It reduces the number of parameters while retaining excellent accuracy by combining pointwise and depthwise convolutions. The architecture also features channel shuffling operations, which enable communication between various convolutional groups. Additionally, ShuffleNet makes use of residual connections to boost model precision.

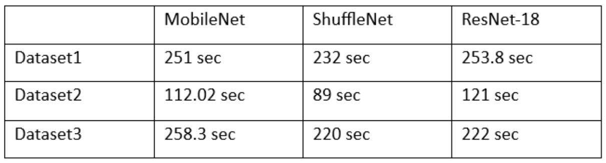
C.2.2 MobileNet

Another compact CNN design that is suitable for mobile devices is MobileNet_v2. In order to divide each convolution into a depthwise convolution and a pointwise convolution, it employs depthwise separable convolutions. This keeps the model's accuracy high while lowering its computing cost. Skip connections are another feature of MobileNet_v2 that facilitate more effective information transfer throughout the network. It also contains inverted residual blocks that make use of short-cut connections to enhance information flow throughout the network.

C.2.3 ResNet18

ResNet18 makes use of residual connections, which enable the model to learn residual information and boost model precision. ResNet18 has 18 layers, including fully connected, pooling, and convolutional layers. Additionally, the architecture has skip connections, which allow to avoid the issue of vanishing gradients and enhance model training.

The selected CNN models, ShuffleNet, MobileNet, and ResNet18, were chosen for this task because the datasets are relatively small and these models can provide good accuracy while using fewer parameters, resulting in faster training times and lower computational costs. Their use of depthwise separable convolutions, residual connections, and skip connections allows for effective information transfer and model precision.



	MobileNet	ShuffleNet	ResNet-18
Dataset1	251 sec	232 sec	253.8 sec
Dataset2	112.02 sec	89 sec	121 sec
Dataset3	258.3 sec	220 sec	222 sec

Figure 4. Time taken per Epoch

Figure 4 above shows the time taken per epoch by each of the model on all three datasets.

For Dataset1, where each image has a lot of well-defined features(most complex in terms of different images), it takes longer for the models to process each epoch. Additionally, MobileNet and ResNet-18 are more complex models compared to ShuffleNet, which may also contribute to the longer training time.

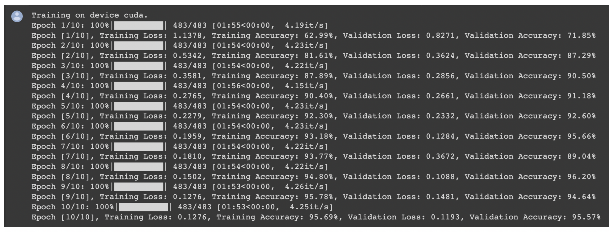
For Dataset2, which has a larger number of images, it takes less time per epoch for all models compared to Dataset1. However, we can observe that MobileNet takes longer than ShuffleNet and ResNet-18, which may be due to its larger number of parameters and deeper architecture.

For Dataset3, it takes less time per epoch for all models. Additionally, we can observe that ShuffleNet takes the least time for training, followed by ResNet-18 and then MobileNet. This may be due to ShuffleNet's efficient architecture that reduces the number of parameters and computations required.

D. Optimization Algorithm

Optim.Adam() optimizer has been used for the optimization of the models. Shufflenet with the second dataset was chosen for optimization the hyper parameters as it was slightly under performing. Different tests with 32, 64, 128 and 256 batch sizes has been conducted and the results are shown below.

It is clearly visible that the both training and validation accuracy increase with the epoch with slight variations with



```
Training on device cuda.
Epoch 1/10: 100%|██████████| 483/483 [01:55<00:00,  4.19it/s]
Epoch [1/10], Training Loss: 1.1378, Training Accuracy: 62.99%, Validation Loss: 0.8271, Validation Accuracy: 71.85%
Epoch 2/10: 100%|██████████| 483/483 [01:54<00:00,  4.23it/s]
Epoch [2/10], Training Loss: 0.5342, Training Accuracy: 81.61%, Validation Loss: 0.3624, Validation Accuracy: 87.29%
Epoch 3/10: 100%|██████████| 483/483 [01:54<00:00,  4.22it/s]
Epoch [3/10], Training Loss: 0.3581, Training Accuracy: 87.89%, Validation Loss: 0.2856, Validation Accuracy: 90.50%
Epoch 4/10: 100%|██████████| 483/483 [01:56<00:00,  4.15it/s]
Epoch [4/10], Training Loss: 0.2765, Training Accuracy: 90.40%, Validation Loss: 0.2661, Validation Accuracy: 91.18%
Epoch 5/10: 100%|██████████| 483/483 [01:54<00:00,  4.23it/s]
Epoch [5/10], Training Loss: 0.2279, Training Accuracy: 92.30%, Validation Loss: 0.2332, Validation Accuracy: 92.60%
Epoch 6/10: 100%|██████████| 483/483 [01:54<00:00,  4.23it/s]
Epoch [6/10], Training Loss: 0.1959, Training Accuracy: 93.18%, Validation Loss: 0.1284, Validation Accuracy: 95.66%
Epoch 7/10: 100%|██████████| 483/483 [01:54<00:00,  4.22it/s]
Epoch [7/10], Training Loss: 0.1810, Training Accuracy: 93.77%, Validation Loss: 0.3672, Validation Accuracy: 89.04%
Epoch 8/10: 100%|██████████| 483/483 [01:54<00:00,  4.22it/s]
Epoch [8/10], Training Loss: 0.1502, Training Accuracy: 94.80%, Validation Loss: 0.1088, Validation Accuracy: 96.20%
Epoch 9/10: 100%|██████████| 483/483 [01:53<00:00,  4.26it/s]
Epoch [9/10], Training Loss: 0.1276, Training Accuracy: 95.78%, Validation Loss: 0.1481, Validation Accuracy: 94.64%
Epoch 10/10: 100%|██████████| 483/483 [01:53<00:00,  4.25it/s]
Epoch [10/10], Training Loss: 0.1276, Training Accuracy: 95.69%, Validation Loss: 0.1193, Validation Accuracy: 95.57%
```

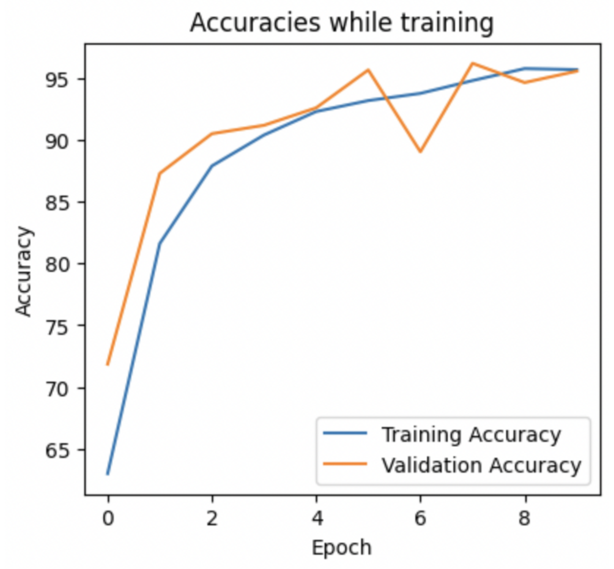
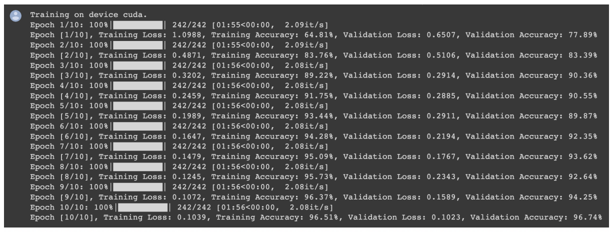


Figure 5. Accuracy results with batch size 32



```
Training on device cuda.
Epoch 1/10: 100%|██████████| 242/242 [01:55<00:00,  2.09it/s]
Epoch [1/10], Training Loss: 1.0988, Training Accuracy: 64.81%, Validation Loss: 0.6507, Validation Accuracy: 77.89%
Epoch 2/10: 100%|██████████| 242/242 [01:55<00:00,  2.09it/s]
Epoch [2/10], Training Loss: 0.4871, Training Accuracy: 83.76%, Validation Loss: 0.5106, Validation Accuracy: 83.39%
Epoch 3/10: 100%|██████████| 242/242 [01:56<00:00,  2.08it/s]
Epoch [3/10], Training Loss: 0.3202, Training Accuracy: 89.22%, Validation Loss: 0.2914, Validation Accuracy: 90.36%
Epoch 4/10: 100%|██████████| 242/242 [01:56<00:00,  2.08it/s]
Epoch [4/10], Training Loss: 0.2459, Training Accuracy: 91.75%, Validation Loss: 0.2885, Validation Accuracy: 90.55%
Epoch 5/10: 100%|██████████| 242/242 [01:56<00:00,  2.08it/s]
Epoch [5/10], Training Loss: 0.1989, Training Accuracy: 93.44%, Validation Loss: 0.2911, Validation Accuracy: 89.87%
Epoch 6/10: 100%|██████████| 242/242 [01:56<00:00,  2.08it/s]
Epoch [6/10], Training Loss: 0.1647, Training Accuracy: 94.28%, Validation Loss: 0.2194, Validation Accuracy: 92.35%
Epoch 7/10: 100%|██████████| 242/242 [01:56<00:00,  2.08it/s]
Epoch [7/10], Training Loss: 0.1479, Training Accuracy: 95.09%, Validation Loss: 0.1767, Validation Accuracy: 93.62%
Epoch 8/10: 100%|██████████| 242/242 [01:56<00:00,  2.08it/s]
Epoch [8/10], Training Loss: 0.1245, Training Accuracy: 95.73%, Validation Loss: 0.2343, Validation Accuracy: 92.64%
Epoch 9/10: 100%|██████████| 242/242 [01:56<00:00,  2.08it/s]
Epoch [9/10], Training Loss: 0.1072, Training Accuracy: 96.37%, Validation Loss: 0.1589, Validation Accuracy: 94.25%
Epoch 10/10: 100%|██████████| 242/242 [01:56<00:00,  2.08it/s]
Epoch [10/10], Training Loss: 0.1039, Training Accuracy: 96.51%, Validation Loss: 0.1023, Validation Accuracy: 96.74%
```

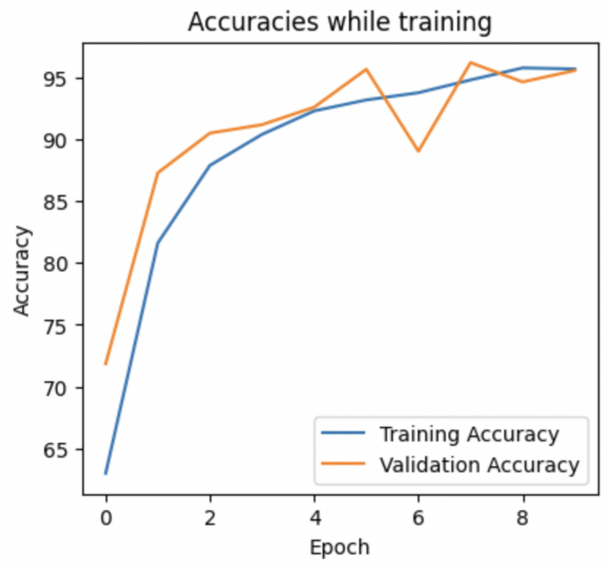


Figure 6. Accuracy results with batch size 64

	Epochs	Batch Size	Learning rate	number of classes
Dataset 1	10	32	0.001	3
Dataset 2	20	64	0.001	15
Dataset 3	15	32	0.001	20

each batch sizes. For this model best training and validation accuracy was obtained through 64 batch size. Hence it is identified that the best suiting batch size as 64.

D.1. Results

Experiment Setup. you need to describe how you setup your experiments, optimized and validated your models, the performance of your models using appropriate metrics (precision, recall, F1-measure, . . .). Explain the ranges of hyper-parameters and rational behind selecting as such in relation to your data and models.

b) Main Results. Demonstrate the main results in figure/table formatting and analyze the performance of your trained models, as well as comparison with other available results. c) Ablative Study. Demonstrate the ablation results from tweaking different hyper-parameters such as number of classes for training, number of images per class training, different range of learning rates, different range of batch-size, etc, and explain your observations. All graphics should be centered. Please ensure that any point you wish to make is resolvable in a printed copy of the response. Resize fonts in figures to match the font in the body text, and choose line widths which render effectively in print. Readers (and reviewers), even of an electronic copy, may choose to print your response in order to read it. You cannot insist that they do otherwise, and therefore must not assume that they can zoom in to see tiny details on a graphic.

When placing figures in LaTeX, it is almost always best to use `\includegraphics`, and to specify the figure width as a multiple of the line width as in the example below

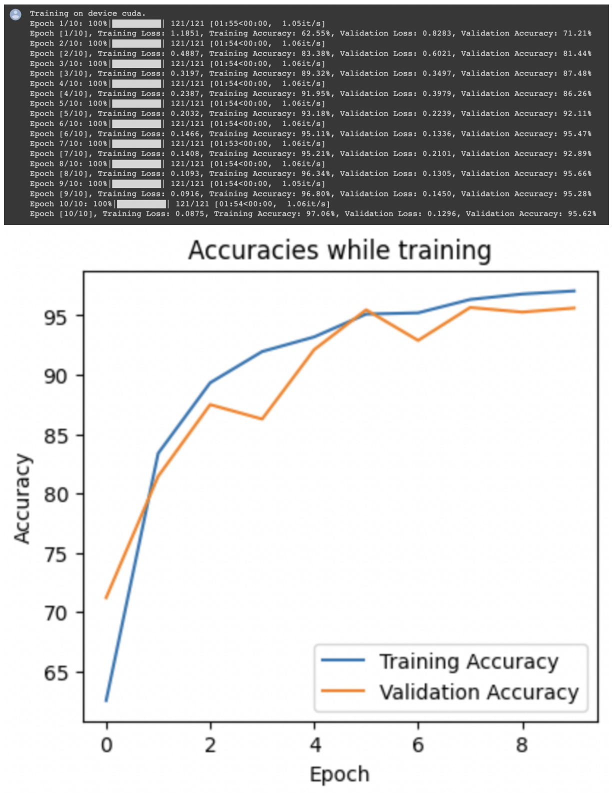
```
\usepackage{graphicx} ...
\includegraphics[width=0.8\linewidth]
                {myfile.pdf}
```

References

[1] E.Oerke. ”Crop losses to pests”. The Journal of Agricultural Science, vol. 144, no. 1, pp. 31-43, 2006. 1

[2] De Cicco V. Zaccardelli G., Di Lernia. ”Diagnosis of tomato spotted wilt virus in pepper by visual inspection, ELISA, and RT-PCR”. Crop Protection, vol. 23, no. 6, pp. 497-501, 2004. 1

[3] Dipayan Biswas Ishrat Zahan Mukti. ”Transfer Learning Based Plant Diseases Detection Using ResNet50 ”. Available: https://ieeexplore.ieee.org/abstract/document/9068805 [Accessed:Mar. 29, 2023]. 2

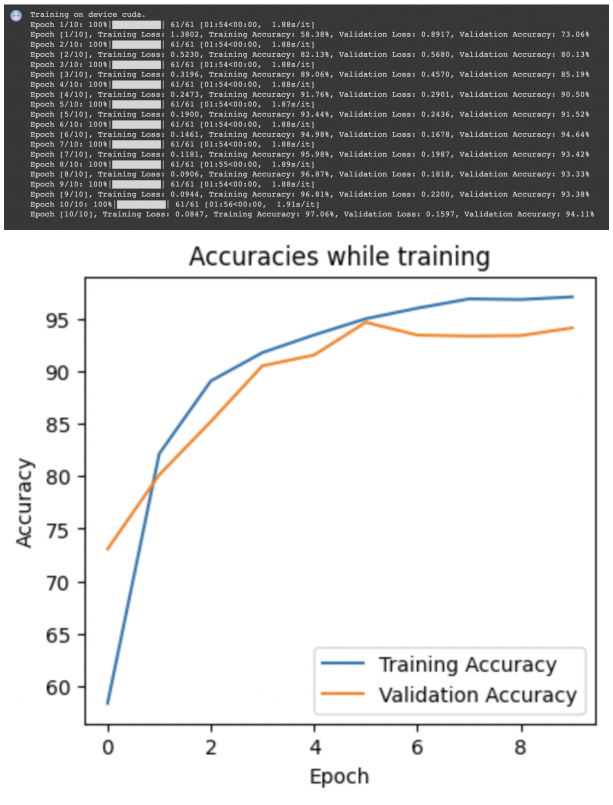


```
Training on device cuda.
Epoch 1/10: 100%|██████████| 121/121 [01:55<00:00,  1.05it/s]
Epoch [1/10], Training Loss: 1.1851, Training Accuracy: 62.55%, Validation Loss: 0.8283, Validation Accuracy: 71.21%
Epoch 2/10: 100%|██████████| 121/121 [01:54<00:00,  1.06it/s]
Epoch [2/10], Training Loss: 0.4887, Training Accuracy: 83.38%, Validation Loss: 0.6021, Validation Accuracy: 81.44%
Epoch 3/10: 100%|██████████| 121/121 [01:54<00:00,  1.06it/s]
Epoch [3/10], Training Loss: 0.3197, Training Accuracy: 89.32%, Validation Loss: 0.3497, Validation Accuracy: 87.48%
Epoch 4/10: 100%|██████████| 121/121 [01:54<00:00,  1.06it/s]
Epoch [4/10], Training Loss: 0.2387, Training Accuracy: 91.95%, Validation Loss: 0.3979, Validation Accuracy: 86.26%
Epoch 5/10: 100%|██████████| 121/121 [01:54<00:00,  1.06it/s]
Epoch [5/10], Training Loss: 0.2032, Training Accuracy: 93.18%, Validation Loss: 0.2239, Validation Accuracy: 92.11%
Epoch 6/10: 100%|██████████| 121/121 [01:54<00:00,  1.06it/s]
Epoch [6/10], Training Loss: 0.1466, Training Accuracy: 95.11%, Validation Loss: 0.1336, Validation Accuracy: 95.47%
Epoch 7/10: 100%|██████████| 121/121 [01:53<00:00,  1.06it/s]
Epoch [7/10], Training Loss: 0.1408, Training Accuracy: 95.21%, Validation Loss: 0.2101, Validation Accuracy: 92.89%
Epoch 8/10: 100%|██████████| 121/121 [01:54<00:00,  1.06it/s]
Epoch [8/10], Training Loss: 0.1093, Training Accuracy: 96.34%, Validation Loss: 0.1305, Validation Accuracy: 95.66%
Epoch 9/10: 100%|██████████| 121/121 [01:54<00:00,  1.05it/s]
Epoch [9/10], Training Loss: 0.0916, Training Accuracy: 96.80%, Validation Loss: 0.1450, Validation Accuracy: 95.28%
Epoch 10/10: 100%|██████████| 121/121 [01:54<00:00,  1.06it/s]
Epoch [10/10], Training Loss: 0.0875, Training Accuracy: 97.06%, Validation Loss: 0.1296, Validation Accuracy: 95.62%
```

Figure 7. Accuracy results with batch size 128

[4] Erika Fujita; Yusuke Kawasaki; Hiroyuki Uga; Satoshi Kagiwada; Hitoshi Iyatomi. ”Basic Investigation on a Robust and Practical Plant Diagnostic System ”. Available: https://ieeexplore.ieee.org/abstract/document/7838282 [Accessed:Mar. 31, 2023]. 2

[5] SK Bodhe SB Patil. Basic Investigation on a Robust and Practical Plant Diagnostic System ”. Available: https://scholar.google.com/scholar_lookup?title=Leaf%20disease%20severity%20measurement%20using%20image%20processing&author=S.%20B.%20Patil&author=S.%20K.%20Bodhe&publication_year=2011 [Accessed:Mar. 27, 2023]. 2



```
Training on device cuda.
Epoch 1/10: 100%|██████████| 61/61 [01:54<00:00,  1.88s/it]
Epoch [1/10], Training Loss: 1.3802, Training Accuracy: 58.38%, Validation Loss: 0.8917, Validation Accuracy: 73.06%
Epoch 2/10: 100%|██████████| 61/61 [01:54<00:00,  1.88s/it]
Epoch [2/10], Training Loss: 0.5230, Training Accuracy: 82.13%, Validation Loss: 0.5680, Validation Accuracy: 80.13%
Epoch 3/10: 100%|██████████| 61/61 [01:54<00:00,  1.88s/it]
Epoch [3/10], Training Loss: 0.3196, Training Accuracy: 89.06%, Validation Loss: 0.4570, Validation Accuracy: 85.19%
Epoch 4/10: 100%|██████████| 61/61 [01:54<00:00,  1.88s/it]
Epoch [4/10], Training Loss: 0.2473, Training Accuracy: 91.76%, Validation Loss: 0.2901, Validation Accuracy: 90.50%
Epoch 5/10: 100%|██████████| 61/61 [01:54<00:00,  1.87s/it]
Epoch [5/10], Training Loss: 0.1900, Training Accuracy: 93.44%, Validation Loss: 0.2436, Validation Accuracy: 91.52%
Epoch 6/10: 100%|██████████| 61/61 [01:54<00:00,  1.88s/it]
Epoch [6/10], Training Loss: 0.1461, Training Accuracy: 94.98%, Validation Loss: 0.1678, Validation Accuracy: 94.64%
Epoch 7/10: 100%|██████████| 61/61 [01:54<00:00,  1.88s/it]
Epoch [7/10], Training Loss: 0.1181, Training Accuracy: 95.98%, Validation Loss: 0.1987, Validation Accuracy: 93.42%
Epoch 8/10: 100%|██████████| 61/61 [01:55<00:00,  1.89s/it]
Epoch [8/10], Training Loss: 0.0906, Training Accuracy: 96.87%, Validation Loss: 0.1818, Validation Accuracy: 93.33%
Epoch 9/10: 100%|██████████| 61/61 [01:54<00:00,  1.88s/it]
Epoch [9/10], Training Loss: 0.0944, Training Accuracy: 96.81%, Validation Loss: 0.2200, Validation Accuracy: 93.38%
Epoch 10/10: 100%|██████████| 61/61 [01:56<00:00,  1.91s/it]
Epoch [10/10], Training Loss: 0.0847, Training Accuracy: 97.06%, Validation Loss: 0.1597, Validation Accuracy: 94.11%
```

Figure 8. Accuracy results with batch size 256