Plant Disease Classification: Classification Based on Images of Leaves

A. Abstract

Automated plant disease classification systems using deep learning algorithms have the potential to revolutionize the way we manage plant diseases and increase crop productivity, leading to a more sustainable and food-secure future. However, there are several challenges associated with analyzing plant images, including the high variability in appearance of healthy and diseased leaves, and the impact of environmental factors. This study addresses these challenges by utilizing various techniques, resulting in high accuracy rates for plant disease classification using deep learning algorithms. Specifically, the MobileNet, ShuffleNet, and ResNet models achieved accuracy rates ranging from 92.64% to 99.19% on various datasets of diseased and healthy plant leaves. With the availability of lightweight models and hardware accelerators, these systems can be trained and deployed efficiently on various devices, ultimately improving the efficiency and cost-effectiveness of plant disease diagnosis.

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) [11] 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. [5]

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. [6]

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. [7]

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.

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 preprocessing, respectively.

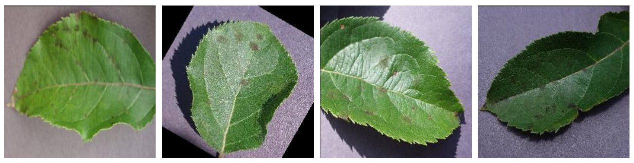


Figure 1. Before pre-processing

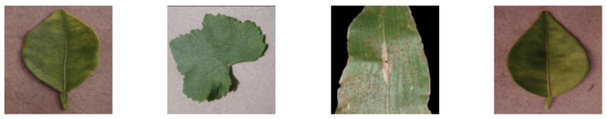


Figure 2. After pre-processing

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 [10] 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.

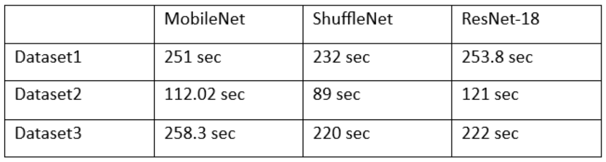
C.2.2 MobileNet

Another compact CNN design that is suitable for mobile devices is MobileNet_v2 [8] . 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 [9] 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 3. Time taken per Epoch during training

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.

C.3. Optimization Algorithm

To validate and optimize the models, the dataset was split into training, validation, and testing sets using an 75-10-15 split. The models were trained on the training set, and the hyperparameters were tuned using the validation set. The model with the best performance on the validation set was selected as the final model, and its performance was evaluated on the testing set.

The optimization algorithm used was Adam, which is an adaptive learning rate optimization algorithm. It combines the benefits of two other optimization algorithms, namely, AdaGrad and RMSProp. Adam uses the first and second moments of the gradients to compute adaptive learning rates for each parameter, which results in faster convergence and better performance.

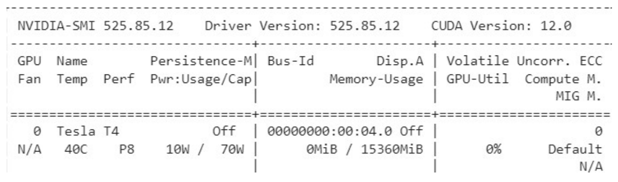
For hyper-parameter optimization the Shufflenet model trained from scratch on the second dataset was chosen, as it was slightly underperforming.

The performance of the models was evaluated using precision, recall, F-score, and confusion matrix. Precision is the ratio of true positives to the total number of positive predictions, while recall is the ratio of true positives to the total number of actual positives. F-score is the harmonic mean of precision and recall, and it provides a balance between the two metrics. The confusion matrix provides a summary of the predictions made by the models, and it shows the number of true positives, false positives, true negatives, and false negatives.

D. Results

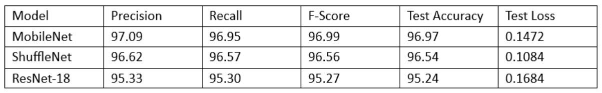
D.1. Experimental setup

The models were trained on a GPU available in Google Colab. Figure 4 shows the specifications of the GPU used. The system had 12 GB of RAM and 80 GB of disk space. The dataset was imported directly to the Colab machine from Kaggle using the Kaggle API.



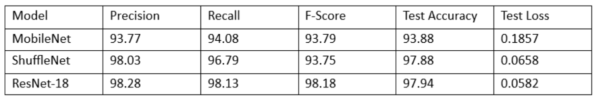
```
NVIDIA-SMI 525.85.12    Driver Version: 525.85.12    CUDA Version: 12.0
-------------------------------+----------------------+----------------------
GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC
Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M.
                              |                      |               MIG M.
==============================+======================+======================
  0  Tesla T4            Off  | 00000000:00:04.0 Off |                    0
N/A   40C    P8    10W /  70W |      0MiB / 15360MiB |      0%      Default
                              |                      |                  N/A
```

Figure 4. GPU configuration of collab



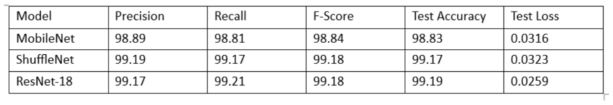
Model	Precision	Recall	F-Score	Test Accuracy	Test Loss
MobileNet	97.09	96.95	96.99	96.97	0.1472
ShuffleNet	96.62	96.57	96.56	96.54	0.1084
ResNet-18	95.33	95.30	95.27	95.24	0.1684

Figure 5. Comparison of Performance Metrics on Dataset 1



Model	Precision	Recall	F-Score	Test Accuracy	Test Loss
MobileNet	93.77	94.08	93.79	93.88	0.1857
ShuffleNet	98.03	96.79	93.75	97.88	0.0658
ResNet-18	98.28	98.13	98.18	97.94	0.0582

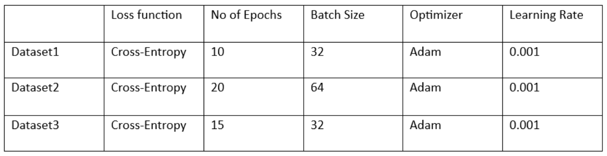
Figure 6. Comparison of Performance Metrics on Dataset 2



Model	Precision	Recall	F-Score	Test Accuracy	Test Loss
MobileNet	98.89	98.81	98.84	98.83	0.0316
ShuffleNet	99.19	99.17	99.18	99.17	0.0323
ResNet-18	99.17	99.21	99.18	99.19	0.0259

Figure 7. Comparison of Performance Metrics on Dataset 3

Figures 5, 6, and 7 display the values of performance metrics for ResNet-18, MobileNet, and ShuffleNet on all three datasets. Overall, the results indicate that ResNet-18 performed better than MobileNet and EfficientNet. This can be attributed to ResNet-18's deeper architecture that can capture more complex features and patterns in the data. Moreover, ResNet-18's residual connections help prevent the problem of vanishing gradients during training, leading to better optimization and more accurate predictions.



	Loss function	No of Epochs	Batch Size	Optimizer	Learning Rate
Dataset1	Cross-Entropy	10	32	Adam	0.001
Dataset2	Cross-Entropy	20	64	Adam	0.001
Dataset3	Cross-Entropy	15	32	Adam	0.001

Figure 8. fig: Value of Hyperparameters

Figure 8 depicts the values of hyperparameters used for all our three datasets. The selection of hyper-parameters was based on empirical testing and observing the performance of the models on validation sets. The choice of optimizer and learning rate was made based on the ability to converge quickly and efficiently. The batch size was selected based on the memory constraints of the GPU. The number of epochs was chosen based on observing the convergence of the loss and accuracy on a validation set.

D.2. Main Results

A total of 9 models were trained using 3 different CNN architectures and 3 distinct datasets with random initial weights. The accuracy and loss plots for each model on both the training and validation sets are presented in Figures 9, 10, and 11. The plots demonstrate that the models effectively learned from the training data, as evidenced by the increasing accuracy and decreasing loss over the epochs. The validation data exhibited similar trends, indicating that the models possess good generalization capability.

Transfer learning was applied to the MobileNet and ShuffleNet models, which were initially trained on dataset2 with randomly initialized weights, by using pre-trained

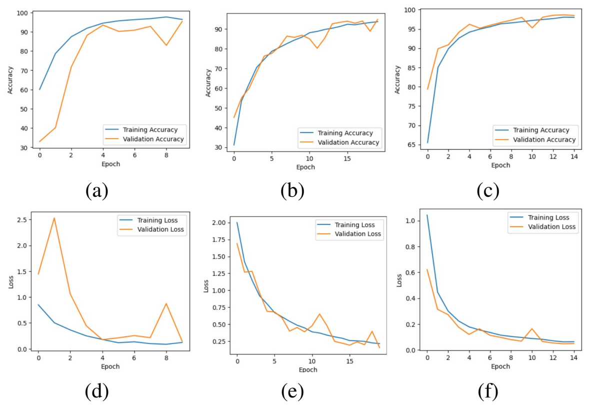


Figure 9. (a), (b), and (c) represent the accuracy of MobileNet CNN on Dataset 1,2,3 respectively while (e),(f),(g) show loss plots for MobileNet on training data Dataset 1,2,3 respectively

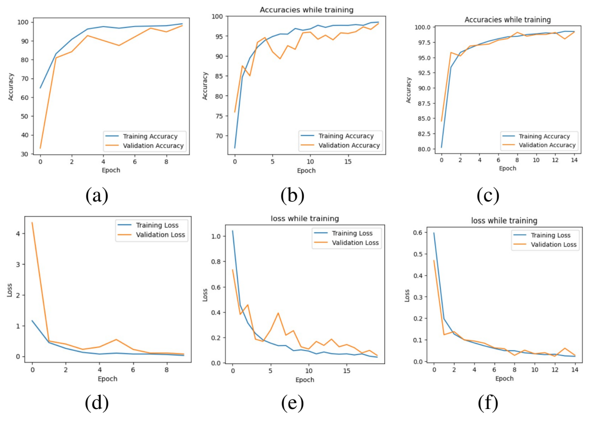


Figure 10. (a), (b), and (c) represent the accuracy of ShuffleNet CNN on Dataset 1,2,3 respectively while (e),(f),(g) show loss plots for ShuffleNet on Dataset 1,2,3 respectively

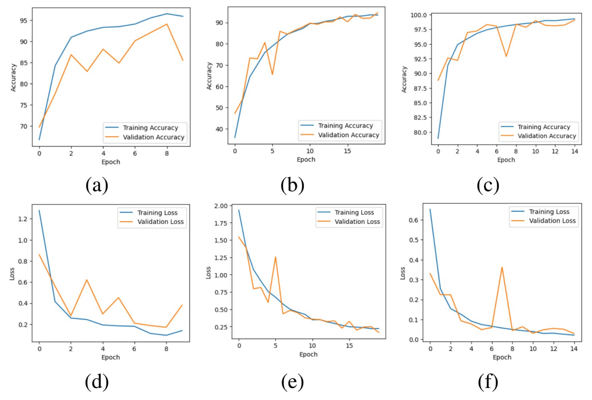


Figure 11. (a), (b), and (c) represent the accuracy of ResNet18 CNN on Dataset 1,2,3 respectively while (e),(f),(g) show loss plots for ResNet18 on Dataset 1,2,3 respectively

weights from the IMAGENET dataset. Despite the good performance of our models with random initial weights, it was noticed that transfer learning allowed the models to quickly adapt to the data and converge faster. This proves that transfer learning has a lower computational footprint and produces equally good results.

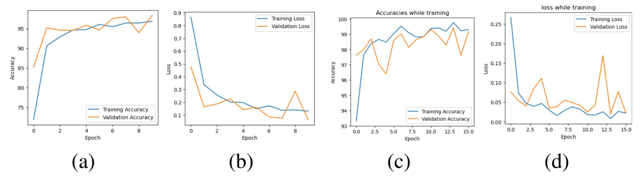


Figure 12. (a) and (b) depict the accuracy and loss of the MobileNet, while (c) and (d) show accuracy and loss plots of ShuffleNet using pre-trained weights respectively.

With regards to applying optimization to one model trained from scratch, batch size was chosen as a hyperparameter for tuning the ShuffleNet model trained on Dataset2. Multiple tests were conducted with different batch sizes (32, 64, 128, and 256). Results showed that the best training accuracy was obtained with batch sizes 128 and 256, but the best validation accuracy was obtained with batch size 64. Validation accuracy fluctuated as batch size increased. Hence, batch size 64 was identified as the best fit for this model. The fluctuations in the validation accuracy with increasing batch sizes may be due to the fact that larger batch sizes lead to less noisy gradients and a more accurate estimate of the true gradient. However, larger batch sizes also require more memory and computational resources, which may lead to slower convergence or overfitting. In this case, it appears that a batch size of 64 achieved the best balance between accuracy and efficiency for the given model and dataset.

Batch size	32		64		128		256	
Epoch	TA %	VA %	TA %	VA %	TA %	VA %	TA %	VA %
1	62.99	71.85	64.81	77.89	62.55	71.21	58.38	73.06
2	81.61	87.29	83.76	83.39	83.38	81.44	82.13	80.13
3	87.89	90.50	89.22	90.36	89.32	87.48	89.06	85.19
4	90.40	91.18	91.75	90.55	91.95	86.26	91.76	90.50
5	92.30	92.60	93.44	89.87	93.18	92.11	93.44	91.52
6	93.18	95.66	94.28	92.35	95.11	95.47	94.98	94.64
7	93.77	89.04	95.09	93.62	95.21	92.89	95.98	93.42
8	94.80	96.20	95.73	92.64	96.34	95.66	96.87	93.33
9	95.78	94.64	96.37	94.25	96.80	95.28	96.81	93.38
10	95.69	95.57	96.51	96.74	97.06	95.62	97.06	94.11

Figure 13. Impact of Batch Size on Train and Validation accuracy.

In addition to utilizing deep learning algorithms for automated plant disease classification,t-SNE (t-Distributed

Stochastic Neighbor Embedding) was incorporated as a visualization tool to gain insight into the distribution of feature representations learned by the models. By reducing high-dimensional feature representations into a two-dimensional space, t-SNE allows for the visualization of clustering patterns and the identification of similar features. Through the use of t-SNE, we were able to observe that the feature representations learned by the deep learning models effectively separated the different classes of diseased and healthy plant leaves, indicating the robustness of our models. Figure 14 showcases the TSNE plots for dataset1 and dataset2.

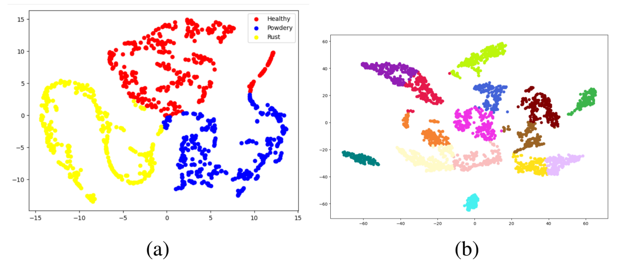


Figure 14. (a) TSNE plot for dataset1, (b) TSNE plot for dataset2

Figure 15 provides a simultaneous comparison of all three models on each dataset. Based on the average validation and training accuracy and loss results, ResNet18 emerges to be the most suitable model for the task of Plant Disease Classification using Leaf Images. However, ShuffleNet and MobileNet are not far behind and can also be utilized, especially in cases where lightweight machine-learning models are necessary.

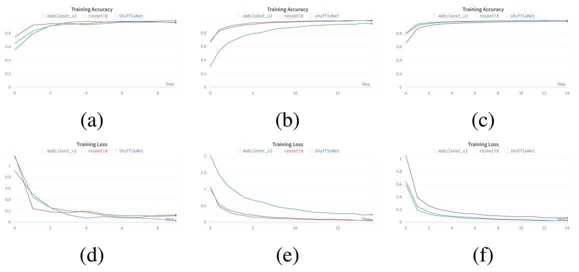
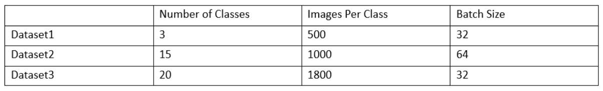


Figure 15. (a), (b), and (c) represent the accuracy of all three CNNs on Dataset 1,2,3 respectively while (e),(f),(g) show loss plots for MobileNet on training data Dataset 1,2,3 respectively

Various detailed comparisons are performed using weights and biases and are available at Wandb Dashboard.

D.3. Ablative Study

To assess the sensitivity plant disease classification model to different hyper-parameters, an ablation study was conducted by tweaking various parameters such as the number of classes for training, the number of images per class training, the learning rates, and batch sizes.



	Number of Classes	Images Per Class	Batch Size
Dataset1	3	500	32
Dataset2	15	1000	64
Dataset3	20	1800	32

Figure 16. Ablation Study

First, impact of varying the number of classes for training was evaluated on the performance of the Resnet-18 model. It was observed that increasing the number of classes from 15 to 20 resulted in a slight increase in accuracy from 97.94% to 99.19%. This improvement could be attributed to a better distribution of the data within the additional classes, leading to improved discrimination between the different disease categories. Overall, these findings highlight the importance of careful consideration of the number of classes used for training in order to achieve optimal model performance.

Next, the effect of varying the number of images per class was evaluated for training. We observed that increasing the number of images per class from 500 to 1000 improved the accuracy of our model from 95.24% to 97.94%. This could be attributed to the fact that a larger number of training images helps the model learn more robust features for classification.

We also experimented with different ranges of learning rates and batch sizes to optimize the training process. We found that a learning rate range of 0.001 and a batch size of 32 resulted in the highest accuracy rates of 99.19%. Lower learning rates led to slower convergence, while higher learning rates resulted in overfitting. Similarly, smaller batch sizes led to faster convergence but required more epochs for training, while larger batch sizes resulted in slower convergence and lower accuracy.

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

[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] Kaggle. ”Dataset-1 Plant diseases recognition Dataset”. Available: https://www.kaggle.com/datasets/rashikrahmanpritom/plant-disease-recognition-dataset [Accessed:Jan. 28, 2023]. 2

[6] Kaggle. ”Dataset-2 PlantVillage Dataset”. Available: https://www.kaggle.com/datasets/emmarex/plantdisease [Accessed:Jan. 27, 2023]. 2

[7] Kaggle. ”Dataset-3 New Plant Diseases Dataset”. Available: https://www.kaggle.com/datasets/vipoooool/new-plant-diseases-dataset [Accessed: Jan. 26, 2023]. 2

[8] PyTorch. ”MobileNet”. Available: https://pytorch.org/vision/main/models/mobilenetv2 [Accessed:Mar.12, 2023]. 3

[9] PyTorch. ”ResNet-18”. Available: https://pytorch.org/vision/main/models/generated/torchvision.models.resnet18.html [Accessed:Mar.15, 2023]. 3

[10] PyTorch. ”ShuffleNet”. Available: https://pytorch.org/vision/main/models/shufflenetv2.html [Accessed:Mar.14, 2023]. 3

[11] 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