Plant Disease Detection using CNN and Transfer Learning

Abstract

Plant disease detection is crucial for maintaining healthy crops and ensuring good yields in agriculture and plantation sector. Convolutional Neural Networks (CNNs) have displayed significant potential in accurately detecting plant diseases through image analysis. In this study, the effectiveness of CNNs like ResNet18, MobileNetV2, and InceptionV3 was explored for plant disease detection by training a model on a dataset of diseased plant images. The model's performance was optimized by experimenting with various hyperparameters, specifically the learning rate. Moreover, the impact of data augmentation techniques such as flipping, rotation, and normalization on the model's accuracy was evaluated. The effectiveness of models trained with and without weights in transfer learning was also compared.

A. Introduction

Farmers and plantation businesses worldwide face significant losses in crop yields due to plant diseases. Traditional manual techniques for detecting plant diseases can be time-consuming and often ineffective due to a lack of expertise [16]. However, deep learning-based techniques, particularly Convolutional Neural Networks (CNNs), provide a possibility to automate the process and achieve better accuracy in disease detection. CNNs have the potential to revolutionize plant disease detection by providing quick and accurate identification of diseases, enabling farmers to take necessary actions to prevent disease spread and increase crop yields [13] [18].

Despite the potential of CNNs for plant disease detection, researchers face several challenges. One of the significant challenges is the limited availability of large datasets, making it difficult to train models effectively [12]. Additionally, plants' leaves exhibit a wide range of appearances, making it challenging to generalize the models' performance. Lastly, training CNNs can be computationally expensive, posing a challenge for small farmers or regions with limited resources. Novel techniques and strategies are necessary to develop effective models to detect plant diseases.

This study aimed to employ CNNs to detect plant diseases using three distinct datasets sourced from Kaggle. The main objective was to evaluate and compare the performance of three distinct CNN architectures: Resnet18, MobileNetv2, and InceptionV3. All three models were trained under the same environment and hyperparameters to ensure a fair comparison. The models' performance was evaluated using various metrics such as accuracy, precision, recall, and F1-score.

To address some of the challenges, the three datasets utilized in this study differed in terms of class, the number of samples, and image quality. All images were pre-processed by resizing them to a uniform size to reduce computational complexity. Additionally, hyperparameter tuning was conducted with various parameters to assess whether any performance improvement could be achieved. Transfer learning using fine-tuning and deep-tuning was also performed to compare the results with pre-trained weights. Stratified k-fold was implemented for one of the models to assess the model's ability to generalize.

The study's findings provide insight into the performance of three distinct CNN architectures for plant disease detection using different datasets. The results showed that all three models achieved reasonably good accuracy, precision, recall, and F1-score. Moreover, hyperparameter tuning and transfer learning were successful in enhancing the models' performance.

A.1. Related Works

In "Plant Disease Detection and Classification by Deep Learning", the authors used a custom CNN model as well as two pre-trained models, ResNet-50 and Inception-v3, to classify 38 different plant diseases on the Plant Village dataset [20]. The dataset consisted of 54,306 images of leaves belonging to different plant species. The custom CNN model was trained from scratch, while the pre-trained models were fine-tuned on the dataset. The authors achieved an accuracy of 95.71% with the custom CNN model, while the pre-trained models achieved much higher accuracies of 99.08% and 98.21% for ResNet-50 and Inception-v3, respectively. The results showed that using pre-trained models can significantly improve the accuracy of plant disease classification. The study also highlighted the potential of deep learning approaches in the early detection of plant diseases, which can help prevent the spread of diseases and improve crop yield.

In the study "Plant Disease Identification using Transfer Learning of Inception-v3 and Resnet-v18 Model" by N. K. Yadav et al. (2020) [12], the authors aimed to develop a system for identifying plant diseases using deep learning techniques. They utilised the Plant Village dataset, which consists of 54,309 images of 14 different crop species and 26 different diseases, to train their models. The authors used transfer learning with the pre-trained Inception-v3 and ResNet-18 models and achieved high accuracy rates of 98.95%and 98.33%, respectively. They also compared the performance of their models to other state-of-the-art models and found that their approach outperformed them. The results show the potential of transfer learning for plant

disease identification using deep learning techniques.

The study conducted "Plant Disease Classification using Convolutional Neural Networks: A Comparative Study" by S. Ghosh et al. (2021) [21], aimed to compare the performance of different CNN models, including ResNet-50 and MobileNet-v2, for plant disease classification on the Plant Disease 224 dataset. The dataset consists of 38 classes of plant diseases, and the models were trained on a total of 54,306 images. The authors reported an accuracy of 99.27% using ResNet-50 and 98.80% using MobileNet-v2 for plant disease classification. In addition to accuracy, the study also compared other evaluation metrics such as precision, recall, and F1-score for each model. The findings suggest that ResNet-50 performed better than MobileNet-v2 for this task, which may be due to ResNet-50's deeper architecture and larger number of trainable parameters.

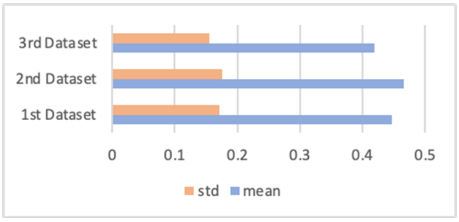
The study "Plant Disease Detection and Identification using Convolutional Neural Network (CNN)" by P. Prajapati et al. (2019) [17] is an important contribution to the field of plant disease detection. In this study, the authors employed state-of-the-art deep learning models, ResNet-50 and InceptionV3, for the detection and identification of plant diseases. The dataset used in this study was collected from tomato and potato plants, and consisted of more than 20,000 images of healthy and diseased plants. The authors trained their models on this dataset and achieved an impressive accuracy of 99.41%To achieve such high accuracy, the authors utilised transfer learning and fine-tuned the pre-trained models for their specific task. They also applied data augmentation techniques to increase the size of their dataset, which helped to improve the robustness and generalisation of their models. The authors' experimental results demonstrate that deep learning models, particularly CNNs, can effectively classify plant diseases with high accuracy, which could have important implications for precision agriculture and food security.

B. Methodology

The plant leaf images contain significant features of plant diseases, which can assist in detecting the diseases through computerised extraction of these features. In this study, an approach was developed to detect plant diseases among various crops using plant disease images. Convolutional Neural Network (CNN) architectures were utilized in this study, as they excel in classification tasks related to computer vision, making it possible for them to evaluate plant images. The following sections discuss various CNN architectures and their methods for identifying plant diseases in different plants.

B.1. Datasets

The study utilised three distinct datasets sourced from Kaggle, each varying in terms of the number of images per class and the types of plant diseases included. Statistical comparison of the three datasets is shown in Figure 1, which highlights that the first two datasets are similar, with only a small difference of 10-12% in mean and standard deviation compared to the third dataset.



(a) Unhealthy Class

Figure 1. Summary statistics of datasets

Dataset 1: potato-tomato-dataset

(a) Unhealthy Class (b) Healthy Class

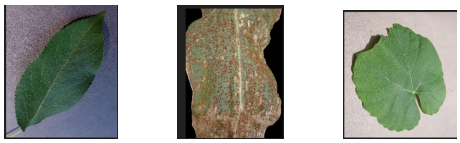
Figure 2. Sample datapoints in first dataset

The first dataset used in this study contained 2283 images of healthy and unhealthy tomato and potato leaves, divided into two classes. Pre-processing techniques, including transformations and resizing, were applied to ensure suitability for the model. The dataset was already split into training, validation, and test sets, and modelling was performed accordingly. Figure 2 displays a few images from each class to provide a basic understanding of the dataset's images.

The standard deviation between the healthy and unhealthy classes differed by 0.025 points, with no difference observed in the mean. Images for the statistics are presented in additional document.

Dataset 2: Plantvillage-dataset

PlantVillage dataset, containing 54,303 leaf images, was utilized, focusing on 10 categories with 16,810 healthy and unhealthy images. Figure 3 presents two healthy and one unhealthy sample image from dataset 2. The PlantVillage dataset provides a diverse range of images for machine learning model development and testing, highlighting its significance in advancing the field. A maximum difference of 0.15 pixels was observed between the minimum and maximum points in mean, and a maximum difference of 0.1 pixels was observed between the minimum and maximum points in standard deviation, with corresponding images presented in an additional document.



(a) Apple Healthy (b) Corn Unhealthy (c) Grape Healthy

Figure 3. Sample datapoints in second dataset

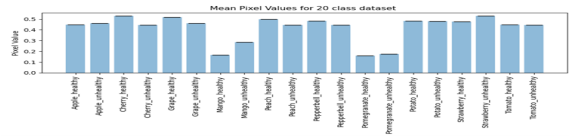
Dataset 3: Plant disease [50 classes]

20 categories of the Plant disease dataset were used, including Apple, Grape, Cherry, Mango, Peach, Pepper Ball, Potato, Pomegranate, Strawberry, and Tomato, which were selected based on their relevance, containing 19,719 healthy and unhealthy images; Figure 4 shows sample images from some of the classes.

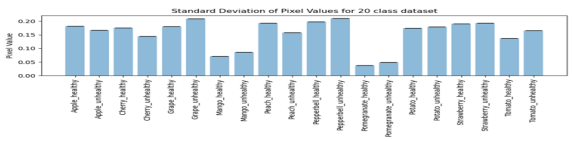
(a) Cherry Healthy (b) Mango Unhealthy (c) Peach Healthy

Figure 4. Sample datapoints in third dataset

The plots presented in Figures 5 illustrate the mean and standard deviation of the 3rd dataset. This dataset showed a significant variance in the distribution of data compared to the other two datasets, and the Mango and Pomegranate healthy and unhealthy classes were the primary contributors. The average gap between the means of these four classes and the dataset's mean was 0.3 pixels, while the average gap between the standard deviation of these four classes and the dataset's average was 0.1 pixels. The large image sizes in these classes led to bottlenecks and required significant computation power, resulting in a considerable size gap. To overcome this issue, the images were resized during the cleaning phase, and any remaining problems were addressed during runtime [24].



(a) Mean



(b) Standard Deviation

Figure 5. 20 Class Dataset statistics

The datasets were split into train/validation/test sets with 70/10/20 ratios and the second and third datasets were transformed with image transformers while the first dataset was used without modification to ensure robustness and generalizability of the models.

B.2. CNN Models

Restnet18, a CNN architecture with 18 layers, uses residual blocks with skip connections to learn residual functions and overcome the degradation problem. It has 11.4 million learnable parameters and four residual blocks, and the final output is fed into fully connected layers for classification [15].

MobileNetV2, a lightweight CNN designed for mobile and embedded devices with 3.5 million parameters that use inverted residual blocks and linear bottlenecks, reducing computational cost and memory footprint while maintaining high performance. Inverted residual blocks include a pointwise convolution, depthwise convolution, and another pointwise convolution, reducing the number of channels, performing a depthwise convolution, and expanding the number of channels [22].

InceptionV3, a 22-layer deep convolutional neural network with 9 inception modules that extract high-level features from input images using parallel convolutions and pooling. Batch normalization and auxiliary classifiers are employed to speed up training and enhance network regularization, and the architecture has a total of 24 million learnable parameters [25].

The FLOPs, which measure the computational requirements of deep learning models, were calculated for ResNet18, MobileNetV2, and InceptionV3 using a Mac M1 hardware CPU. ResNet18, MobileNetV2, and InceptionV3 had FLOPs of 37.16, 74.85, and 168.7, respectively, with InceptionV3 requiring the most FLOPs due to its greater number of convolutional layers. Additionally, longer epoch durations were observed for larger datasets, with ResNet18 and MobileNetV2 taking an average of 145-150 seconds per epoch and InceptionV3 taking around 350 seconds per epoch, while dataset 1 only took an average of 12-13 seconds per epoch.

B.3. Optimization Algorithm

In order to classify images of plant diseases, this study employed convolutional neural network (CNN) models. The CNN model was chosen because it can learn features directly from the images without requiring manual feature engineering. To optimize and validate the model, the stochastic gradient descent (SGD) optimization algorithm with a learning rate of 0.001 and momentum of 0.9 was used. The cross-entropy loss function was chosen as the evaluation metric, which is commonly used in multi-class classification tasks [19].

Additionally, a learning rate scheduler called Cosine Annealing LR was used to gradually reduce the learning rate over time, which helped the model converge to a better minimum [23]. The batch size for the model was set at 64. However, during the training process, it was observed that the model was not converging with a learning rate of 0.0002. To address this, the learning rate was increased to 0.001, which led to a noticeable improvement in convergence speed and resulted in better overall results. Through experimentation with various learning rates, the Learning Rate Finder determined that 0.001 was an optimal value for the model. Additionally, after testing different epoch numbers, it was discovered that training the model for 100 epochs was unnecessary.

After tuning the hyperparameters, the model achieved high accuracy on the test set with 50 epochs, a batch size of 64, a learning rate of 0.001, and a learning rate scheduler. The performance was evaluated using Accuracy, Precision, Recall, F1, and Confusion matrix metrics for 11 models. The results showed that the model was effective in classifying plant disease images.

C. Results

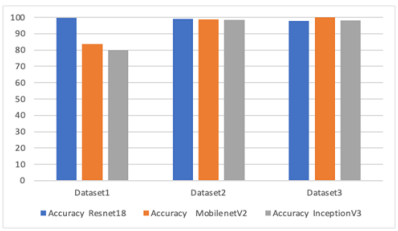
C.1. Experiment Setup

As mentioned in the earlier sections, data from three sources in Kaggle were collected and pre-processed by cleaning, resizing, reducing and merging labels, and applying image processing techniques. Data augmentation techniques such as normalization were used to improve data quality. Three models - ResNet18, MobileNetV2 and Inception V3 - were selected and trained with SGD as optimizer and ReLU activation function. Hyperparameters such as learning rate, batch size and epochs were optimized, and the Cosine Annealing LR method was implemented. The dataset was split into a 70:10:20 ratio for training, validation, and testing. The performance of the model was evaluated using accuracy, loss, and confusion matrix. The model was deployed and evaluated on both validation and testing data to draw conclusions about the accuracy and effectiveness of pre-processing and training techniques.

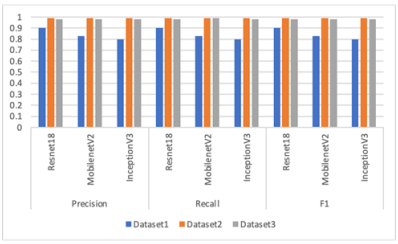
C.2. Main Results

The below plots depicts the comparison of performance metrics i.e accuracy, precision, recall, F1 score of the 3 different CNN models on the 3 datasets.

Overview of CNN models performance on datasets:



(a) Accuracy



(b) Precision, Recall, F1 Score

Figure 6. Comparison of Performance Metrics

Figure 6 displays the performance of Resnet, Mobilenet, and Inception on three datasets. In the first dataset, Resnet achieved the highest accuracy of 90.33%, while Mobilenet and Inception scored lower accuracy rates. Despite variations in accuracy, precision and recall scores were consistent across all models except for Mobilenet, which had a slightly lower recall score. The F1 score remained constant across all models. Dataset size and nature played a significant role in the models' performance, with biases in the first dataset affecting wider models. In the second dataset, all models demonstrated high accuracy rates, with consistent precision, recall, and F1 scores. In the third dataset, Mobilenet achieved the highest accuracy rate, followed by Resnet and Inception. All models scored a precision of 0.98, while recall scores varied slightly between models. The F1 score remained consistent across all models.

More details about results will be presented in the following subsection.

Comparison of 3 datasets on one model

Inception was chosen for this comparison because it exhibits varying levels of performance across the three datasets.

(a) Dataset 1 (b) Dataset 2 (c) Dataset 3

Figure 7. Accuracy of Inception with three datasets

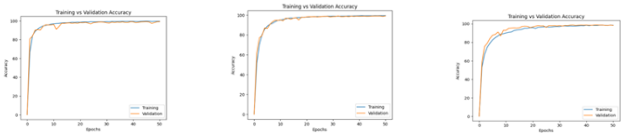
(a) Dataset 1 (b) Dataset 2 (c) Dataset 3

Figure 8. Loss of Inception with three datasets

Figures 7 and 8 display Inceptionv3's accuracy and loss across all datasets from left to right. Dataset 1 started with validation accuracies in the 70s, reaching a maximum validation accuracy of 83% at the end of 50 epochs, while dataset 2 and 3 started in the 50s and 30s, respectively, but reached 90% validation accuracy within 10 to 20 epochs, unlike dataset 1. For dataset 1, even though validation losses were high, they gradually fell below 0.6 and ended up at 0.4 loss. In contrast, validation losses for dataset 2 were inconsistent and fluctuating. For dataset 3, losses consistently decreased from 2.5 to 0.4 within 20 epochs. As previously mentioned, the test accuracies for datasets 1, 2, and 3 were 79.8%, near 99%, and 98%, respectively. Biases in data and the limited size of dataset 1 likely affected the results.

Comparison of one dataset on 3 model

Dataset 2 has been selected as the basis for comparing the 3 models due to the consistent performance of this dataset resulting from its high quality. The following plots have been constructed from 1 to N.



(a) Dataset 1 (b) Dataset 2 (c) Dataset3

Figure 9. Accuracy of second dataset on 3 model

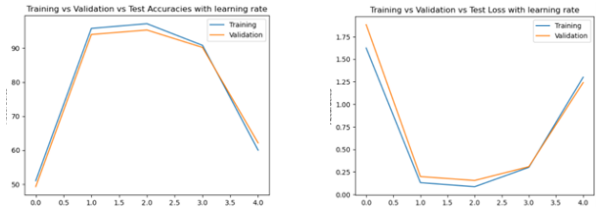
(a) Dataset 1 (b) Dataset 2 (c) Dataset3

Figure 10. Loss of second dataset on 3 model

The validation accuracy and loss for all models are consistent, with initial high losses and accuracies. However, within 10 epochs, all models except InceptionV3 reach good accuracy and low losses. InceptionV3 shows consistent accuracy, but its losses fluctuate every other epoch between a maximum of 7 and a minimum below 0.5, possibly due to learning rate, batch size, and model architecture.

Additional information regarding all the comparisons can be found in the accompanying document.

HyperParameter Search



(a) Accuracy (b) Loss

Figure 11. Hyperparameter Tunning Summary plots

Hyperparameter tuning is a crucial step in improving CNN models' performance, and in this study, it was achieved by adjusting the learning rate of the MobileNetV2 model on dataset 3. Various learning rates were experimented with to identify the optimal value [14]. Although tuning on batch size and epochs was not performed due to a lack of computation resources, the accuracy and loss plots showed that a learning rate of 0.001 is the optimal value for training the model, resulting in better convergence towards a local minima. Thorough tuning of hyperparameters, including batch size and epochs, can further improve model performance.

In the initial iterations, when the learning rate is too high, the model might be skipping the local minima, resulting in low accuracy. Similarly, in the later iterations, when the learning rate is too low, the model might not be able to converge to the local minima, leading to minimal or no learning both the cases. On the other hand, with the optimal learning rate, which is indicated as the centre of the parabola, the loss is very low, and accuracy is high.

Transfer Learning

(a) Without pretrained weights (b) Imagenet weights (c) 3rd dataset weights

Figure 12. Transfer Learning on first dataset with Inception

Two techniques were utilized for transfer learning: fine-tuning, which utilized pre-trained weights from ImageNet, and deep tuning, which employed weights from a saved model on the Inception 2 class dataset that had the lowest accuracy [11] [3]. Without pre-trained weights, validation accuracy started in the 70s and reached a maximum of 80s. However, with pre-trained weights, validation accuracy improved significantly to nearly 95%. Similarly, when weights

were transferred from dataset 2 to dataset 1, promising results were observed, with validation accuracy starting in the 90s and reaching 98%. The plots for these results are shown in Figure 12. Test results confirmed this pattern, with accuracy of 79.8% without pre-trained weights, 85.33% with ImageNet pre-trained weights, and 92% when transferring weights from dataset 2 to dataset 1.

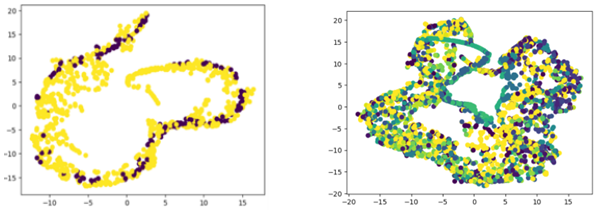
(a) without pretrained weights (b) with imagenet weights (c) 1st dataset weights

Figure 13. Transfer Learning on third dataset with MobileNet

To further explore the concept of transfer learning, the same approach was applied to MobileNet using a third dataset as shown in Figure. As shown in Figure 13 fine-tuning was performed with ImageNet pre-trained weights, while deep tuning was carried out using weights from the lower class dataset 1 to the 20-class dataset. In contrast to previous observations, ImageNet showed better performance, with maximum validation and test results both standing at 99.8%. On the other hand, training without pre-trained weights and using weights from the lower dataset showed similar patterns in validation and test results, with both standing just above 98%. The results may have been influenced by the small size of the dataset and the limited number of data points on which the saved models were trained, which may have contributed to the lack of similar promising results observed in the previous transfer learning case.

Features understanding using t-SNE:

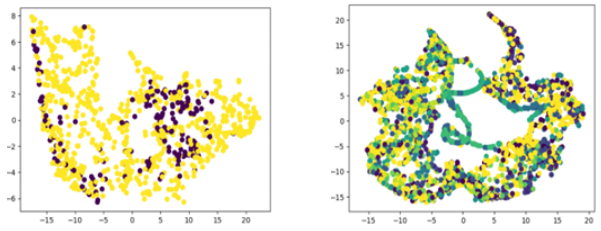
T-SNE is a machine learning and data visualization technique that reduces high-dimensional data to a lower-dimensional space while preserving the local structure of the data, revealing patterns and relationships that may not be visible in the original data [26].



(a) Dataset 1 (b) Dataset3

Figure 14. t-SNE plots of ResNet18

From the t-SNE visualizations in Figure 14, it is clearly evident that the majority of the data points in both datasets are highly clustered together in terms of their properties. Interestingly, ResNet exhibits nearly identical points, which could be the reason for its superior performance compared to other models.



(a) Dataset 1 (b) Dataset 3

Figure 15. t-SNE plots of MobileNetv2

Similar to ResNet, the data points in dataset 1 are close but not too close to each other. However, unlike ResNet, most of the nearest neighbouring points are little distributed and spread, which could explain the MobileNet's 1st dataset result of 88.%. On the other hand, in dataset 3, some points are much closer to each other, unlike in dataset 1.

C.3. Ablative Study

Learning Rate Finder: The Learning Rate Finder tool was used to determine the optimal learning rate for the models by gradually increasing the learning rate during training and plotting the loss or accuracy against the learning rate. The optimal learning rate was found to be 0.001, resulting in faster convergence and better model performance.

Stratified K-Fold: The Stratified K-Fold technique was implemented on the Inception V3 model dataset to split the training data into K-folds, ensuring a similar distribution of classes in each fold. The results showed a significant improvement in model performance with a more consistent accuracy and lower loss compared to the model trained without it. A range of test accuracy between 82% to 86% was achieved using the same dataset and hyperparameters.

More information about these are presented in additional document.

References

[1] [3] how to calculate the mean and standard deviation of your image dataset (pytorch). 7

[2] Cosineannealinglr — pytorch 2.0 documentation. 7

[3] Finetuning torchvision models — pytorch tutorials 1.2.0 documentation. 5

[4] matplotlib.pyplot.bar — matplotlib 3.4.3 documentation. 7

[5] Plant disease [50 classes]. www.kaggle.com/datasets/fabinahian/plant-disease-50-classes. 7

[6] Plantvillage-dataset. www.github.com/spMohanty/PlantVillage-Dataset/tree/master/raw. 7

[7] potato-tomato-dataset. www.kaggle.com/datasets/alyeko/potato-tomato-dataset. 7

[8] torch.optim — pytorch 2.0 documentation. 7

[9] Torchvision object detection finetuning tutorial — pytorch tutorials 1.5.0 documentation. 7

[10] torchvision.transforms — pytorch master documentation. 7

[11] Machine learning mastery, 2016. 5

[12] Muhammad Sufyan Arshad, Usman Abdur Rehman, and Muhammad Moazam Fraz. Plant disease identification using transfer learning. *2021 International Conference on Digital Futures and Transformative Technologies (ICoDT2)*, 05 2021. 1

[13] Konstantinos P. Ferentinos. Deep learning models for plant disease detection and diagnosis. *Computers and Electronics in Agriculture*, 145:311–318, 02 2018. 1

[14] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The Mit Press, 2016. 5

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 06 2016. 3

[16] Lukas Wiku Kuswidiyanto, Hyun-Ho Noh, and Xiongzhe Han. Plant disease diagnosis using deep learning based on aerial hyperspectral images: A review. *Remote Sensing*, 14:6031, 11 2022. 1

[17] Jun Liu and Xuewei Wang. Plant diseases and pests detection based on deep learning: a review. *Plant Methods*, 17, 02 2021. 2

[18] Sharada P. Mohanty, David P. Hughes, and Marcel Salathé. Using deep learning for image-based plant disease detection. *Frontiers in Plant Science*, 7, 09 2016. 1

[19] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2016. 3

[20] Muhammad Hammad Saleem, Johan Potgieter, and Khalid Mahmood Arif. Plant disease detection and classification by deep learning. *Plants*, 8:468, 10 2019. 1

[21] Muhammad Hammad Saleem, Johan Potgieter, and Khalid Mahmood Arif. Plant disease classification: A comparative evaluation of convolutional neural networks and deep learning optimizers. *Plants*, 9:1319, 10 2020. 2

[22] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 06 2018. 3

[23] Leslie N. Smith. A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay. *arXiv:1803.09820 [cs, stat]*, 04 2018. 4

[24] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S. Emer. *Efficient Processing of Deep Neural Networks*. Springer International Publishing, 2020. 3

[25] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 06 2016. 3

[26] Martin Wattenberg, Fernanda Viégas, and Ian Johnson. How to use t-sne effectively. *Distill*, 1, 10 2016. 6