Plant Disease Detection and Classification using Deep Learning

1. Abstract

This study assesses the effectiveness of three CNN models—GoogleNet, MobileNetV2, and EfficientNet—in detecting and classifying plant diseases using images from different plant species. The models were tested on three datasets of varying sizes and classes, employing both training-from-scratch and transfer learning methods. The analysis focuses on the influence of data quality, preprocessing techniques, and hyperparameter adjustments on model performance, using metrics such as Accuracy, Precision, Recall, and F1 Score. EfficientNet showed competitive results across all datasets, with accuracies of 64.96% on Cassava, 78.38% on Crop Disease, and 98.61% on Plant Village. GoogleNet achieved the highest accuracy on the Crop Disease dataset at 78.92%, while MobileNetV2 excelled on the Cassava and Plant Village datasets with accuracies of 66.55% and 99.46%, respectively. Transfer learning improved feature detection in images benefitting from pre-trained weights for learning complex embedded features. Additionally, tuning of hyperparameters (batch size and learning rate) enhanced EfficientNet's performance by 5% on the Cassava dataset.

2. Introduction & Problem Statement

2.1. Problem Statement and Challenges

Agriculture is a major sector that impacts society and the environment while defining the economy of a nation, this necessitates precise plant disease identification for timely intervention. Plant disease recognition poses significant challenges due to the diverse range of symptoms, complicating manual categorization by the human eye. Plant diseases can show a range of symptoms, including color changes, spots, wilting, and deformities. These symptoms are affected by factors like the type of pathogen and the environment. These diseases often escape manual identification as they may not exhibit symptoms in the early stages, necessitating the development of advanced techniques for early detection. Monitoring for disease outbreaks over large agricultural areas is hard with manual methods because there's just so much land to cover. Therefore, it is important to develop automated solutions that can aid in identifying crop diseases to timely manage disease infestation.

Convolutional Neural Networks have proven to be effective for computer vision tasks like classification, detection, recognition, etc. due to their ability to perform embedded feature extraction. However, implementing CNNs can be quite complex, as there are various challenges involved. Overfitting is a significant concern with Deep Convolutional Neural Networks (Deep CNNs), given their large number of parameters, it can hinder generalization and lead to poor performance. Additionally, selecting the appropriate architecture for a given problem, considering factors like image type, dataset size, and specific task requirements, is crucial for achieving optimal results. Moreover, image quality issues such as variations in lighting, growth stages, and environmental factors can introduce further obstacles, often leading to incorrect classification. Addressing these challenges requires a well-analyzed thorough approach to model design and training, emphasizing the importance of finding common ground to navigate through these complexities effectively.

In existing solutions, these problems have been addressed through various mechanisms. Data augmentation techniques like rotation, flipping, resizing, and scaling of images, along with background suppression, are commonly employed to diversify and expand the training dataset. These augmentations not only enhance model robustness but also contribute to improved generalization. Moreover, the transfer learning approach, which leverages pre-trained weights, has emerged as a valuable methodology for various computer vision tasks. By fine-tuning these models for specific plant disease detection tasks, transfer learning mitigates the need for extensively annotated datasets while enhancing overall model performance. Through these integrated approaches, existing solutions have enhanced model performance and minimized information leakage risks [13] [14]. However, pre-trained models may not always capture all relevant features for a specific domain such as plant disease detection. Both data augmentation and transfer learning techniques can require significant computational resources, especially when dealing with large datasets and complex models.

Our study involves training nine models from scratch and two via Transfer Learning on three datasets. Using Cross Entropy Loss and SGD Optimizer, models were trained for 20 epochs with a batch size of 32 and a learning rate of 0.001. Evaluation metrics include Accuracy, Precision, Recall, F1-Score, and continuous monitoring of these metrics enabled adjustments for optimal learning. Overall, the performance of the 3 architectures varies across datasets, with EfficientNet showing consistent competitive performance across all, making it a good choice for solving our problem. It is also observed that all the architectures achieve higher performance on the Plant Village dataset as images feature lower background noise. In terms of computational efficiency, MobileNetV2 tends to achieve a minimum time per epoch across all datasets.

2.2. Literature Review and Related Works

In recent years, numerous studies have examined the efficacy of different deep learning models in identifying plant diseases from leaf images. Nandi et al. conducted

a study [13] to develop a device-friendly diagnostic system for guava plant disease detection. They compared the performance of VGG-16, GoogleNet, ResNet-18, MobileNet-v2, and EfficientNet across a dataset comprising images of guava plants. For deployment, they employed Float16 and Dynamic Range Quantization to reduce model size. Their evaluation revealed that EfficientNet attained the highest overall accuracy at 99%. However, quantized GoogleNet, with its size of 0.143 MB and accuracy of 97%, emerged as the optimal candidate for deployment, particularly suitable for devices with limited storage capacity. Boulent et al. conducted a comprehensive survey [14] comprising 19 studies, aimed at elucidating the advantages and utilization of Convolutional Neural Networks (CNNs) in crop disease identification. Their findings underscored the necessity of a diverse and large image dataset for achieving high performance with CNNs. They emphasized the importance of employing transfer learning, fine-tuning, and careful architecture selection in the development of efficient models. Additionally, the survey highlighted the promotion of regularization techniques and data augmentation to bolster model generalization capabilities.

Shelar et. al. explain a sophisticated architecture for their image classification task, specifically employing the VGG-19 architecture, known for its depth and effectiveness in image classification tasks using multiple layers [12]. This depth allows it to capture a wide range of features from input images, which is crucial for plant disease identification. They also highlighted the importance of deep learning in agriculture, offering a scalable solution for both all sizes of operations. Also, in Sharma et al.'s study, the performance of Convolutional Neural Networks (CNNs) was investigated for identifying diseases in rice and potato crops [15]. They employed a model trained from scratch, comprising convolutional layers, batch normalization, max-pooling, dropout, and fully connected layers. Their model achieved high classification accuracies of 99.58% for rice leaf diseases and 97.66% for potato leaf diseases, with resizing being the only pre-processing step applied.

In our methodology, we drew inspiration from various works, including those mentioned above, and opted to utilize GoogleNet, EfficientNet, and MobileNetV2 architectures. To adapt these architectures to our specific problem, we modified the final fully connected layer. Additionally, we employed transfer learning, fine-tuning, and data augmentation techniques in our approach.

3. Methodology

Our study focuses on using three CNN architectures that were carefully chosen for their optimal balance between size and performance. We conducted a series of experiments, training nine models across various scenarios and employing transfer learning on two models to compare performance with training-from-scratch methods. Additionally, we focused on extensive hyper-parameter tuning for the EfficientNet architecture using the Cassava Dataset. Through adjustments in learning rate and batch size, we aimed to identify the optimal settings for enhanced efficiency and accuracy.

3.1. Datasets

For solving the problem discussed, we have selected the *Cassava Dataset*, *Crop Disease dataset*, and *Plant Village Dataset* as shown in Table 1. The Cassava Dataset contains leaf images of the cassava plant crowd sourced from farmers and analysed by experts at the National Crops Resources Research Institute (NaCRRI) and AI lab in Makarere University, Kampala [1]. The Crop Diseases dataset contains leaf images sourced from farms in Ghana with multiple classes of different diseases [2]. The PlantVillage dataset contains healthy and unhealthy leaf images divided into various classes across multiple species and diseases [3].

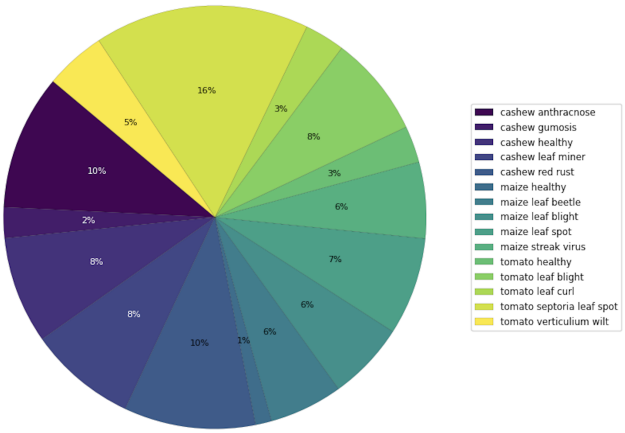


Figure 1. Inter-Class variance of Crop Disease Dataset

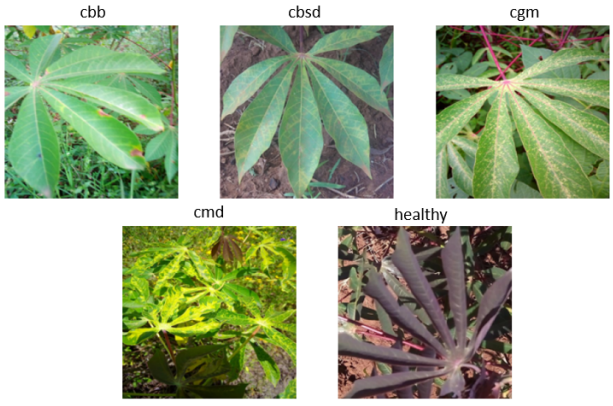


Figure 2. Samples of Cassava Dataset

The inter-class variance of Crop Disease dataset is illus-

trated in Figure 1 and we see that Plant Village dataset (Figure 7) also has a similar distribution of images among the classes. In contrast, the Cassava dataset (Figure 8) has an uneven distribution with Cassava Mosaic Disease images contributing to almost half of the dataset. The samples of the different images of the classes are illustrated in Figures 2, 10 and 11.

In our project, we have allocated the data as follows: 80% to training, 10% to validation, and 10% to testing, across all datasets. The solution pipeline begins with data pre-processing, given that the datasets exhibit varying geometric and photo-metric properties. Initially, we resized all images to a uniform resolution of 224×224×3, ensuring compatibility with the chosen architectures. Additionally, we applied geometric transformations, such as horizontal and vertical flips, to 10% of the images. Photo-metric transformations were also performed by adjusting the brightness and contrast of the images by a factor of 0.2. Finally, we normalized the images using the ImageNet mean and standard deviation values [5].

	Cassava	**Crop Disease**	**PlantVillage**
Images	5,656	24,881	50,091
Size	Var	400x400x3	256x256x3
Color Space	RGB	RGB	RGB
Format	JPG	JPG	JPG
Classes	5	22	37
Trimmed Datasets			
Images	5,656	16,704	35,320
Classes	5	15	30

Table 1. Agricultural Datasets Summary

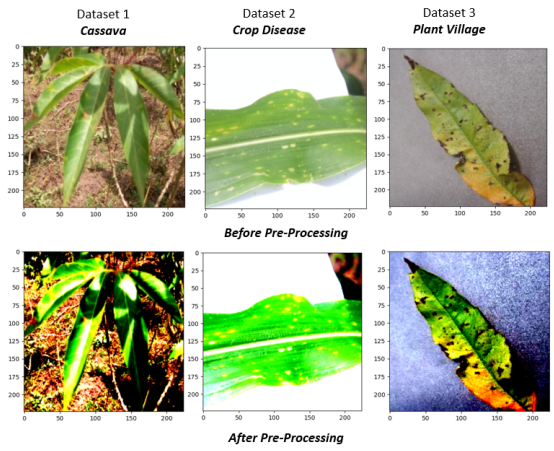


Figure 3. Image samples before and after data augmentation

These pre-processing steps, as seen in Figure 3, including resizing, applying geometric and photo-metric transformations, and normalization according to ImageNet standards, have significantly enhanced model compatibility and robustness, thereby improving generalization across diverse datasets.

3.2. CNN Models

To address the challenges described above, we have used GoogleNet, MobileNetV2, and EfficientNet B0. Consistent architectural modifications were applied across all CNN models, specifically focusing on their final output layers. This standardization ensures that the output dimensions from the models' last layers correspond to the number of classes present in the dataset. To achieve this, three linear layers were introduced to map the output dimensions sequentially to 512, then 256, and finally to the number of classes. Both BatchNorm and Relu layers were incorporated before and after each linear layer, respectively, to enhance model performance and convergence.

MobilenetV2 has depthwise separable convolution layers due to which the complexity cost and size of the network are reduced, making it useful for devices with low computation power. This model utilizes inverted residuals with linear bottlenecks to maintain representational power and achieves state-of-the-art performance for semantic segmentation along with object detection. We chose MobileNetV2 as it excels in computational efficiency, which is advantageous for resource-constrained environments. MobileNetV2 consists of 53 layers and 3.01 million parameters [22]. It has 0.61 billion FLOPS and took an average of 92s per epoch on all datasets. MobileNetV2 surpasses both MobileNetV1 and ShuffleNet (1.5) in performance, despite having similar model size and computational requirements. When equipped with a width multiplier of 1.4, MobileNetV2 (1.4) outperforms ShuffleNet (×2) and NASNet while achieving quicker inference times [17].

GoogleNet is a 22-layer (without pooling layers) deep CNN architecture having the inception module as its core building block incorporating multiple filter sizes (1x1, 3x3, 5x5) within a single module. This enables the network to learn features of different scales in the images, improving its ability to capture complex patterns [23]. GoogleNet heavily utilizes 1x1 convolutions for dimensionality reduction, a global average pooling layer for spatial reduction, and auxiliary classifiers to address the vanishing gradient problem to improve overall training stability which makes it suitable for our application. GoogleNet consists of 6.26 million parameters, has 3.01 billion FLOPS, and took an average of 104s per epoch on all datasets. As compared to other models, it allows for parallel processing of information at different spatial scales which helps capture both local and global features effectively and strikes a balance between accuracy and computational efficiency.

EfficientNet handles the trade-off between accuracy, computational efficiency, and model size very well with

the help of compound scaling. It is a concept that focuses on scaling and balancing three dimensions: width, depth, and resolution. It consists of MBConv blocks, inspired by MobileNetV2, which utilizes the squeeze-to-excitation technique to focus on informative features that also make it suitable for our use. The baseline network model, B0, comprises around 18 layers making it relatively lightweight compared to larger variants in the series [21]. EfficientNet consists of 4.8 million parameters, has 0.79 billion FLOPS, and takes an average of 99s per epoch on all datasets. EfficientNet models exhibit either competitive or superior performance on diverse benchmark datasets and computer vision tasks, making them excellent pre-trained models for transfer learning, particularly when trained on large-scale datasets such as ImageNet [19] [20].

GoogleNet and MobileNetV2 were chosen for transfer learning as these models support fine-tuning which enabled us to tweak and optimize the models' parameters according to our target task. By leveraging the pre-trained weights, we could adapt these models to the unique features and characteristics of our dataset resulting in an enhanced performance. Overall, the models which are popular and effective in image classification tasks were chosen while making sure to address plant disease detection considering computational efficiency and performance. More information on time/epoch for all models is given in Table 2.

3.3. Optimization Algorithm

In our study, the choice of optimization algorithm was critical to effectively train the CNN models, considering the challenges posed by the complex image data and the need for efficient computational performance. We have primarily used Stochastic Gradient Descent (SGD) as our optimization algorithm across all experiments which has helped maintain a good training and validation accuracy over multiple epochs.

We have chosen SGD for its robustness and effectiveness in handling large datasets and its ability to converge even with noisy gradients, which is seen in real-world image data like those in our datasets. While batch gradient descent, which computes the gradient using the entire dataset, SGD updates the parameters more frequently with gradients calculated from randomly selected subsets of the data or mini-batches [24]. We have set the Learning Rate to 0.001, which is a standard value allowing for gradual but steady learning. While performing hyper-parameter tuning, we have also sampled learning rates uniformly from the range 0.0001 to 0.001 and chosen the best configuration. The momentum coefficient of 0.9 was also used to help accelerate the SGD in the relevant direction, thereby avoiding local minima and faster convergence. A batch size of 32 was used across the datasets while a study on different batch sizes of 8, 16, 32, and 64 was tested for hyper-parameter tuning. Furthermore, we assessed the model's performance using metrics such as F1-score, Precision, Recall, and Accuracy. By leveraging the optimization algorithm, we achieved better accuracy rates, showcasing the significance of optimization algorithms in enhancing model performance. We can see the Accuracy and Learning Loss plots of the different architectures across datasets in Figures 12-33.

4. Results

4.1. Experimental Setup

The experiments were run on the local with Python 3.9 and PyTorch 2.2. We leveraged both an NVIDIA RTX 3080 Ti GPU and Apple M2 Silicon for acceleration. Three CNN architectures, GoogLeNet, EfficientNet B0, and MobileNetV2, were used for experimentation. The datasets were split in the ratio of 80:10:10 for training, validation, and testing, respectively.

In the main study, we have trained 9 models from scratch using these architectures over 3 datasets. Each model has been run on 20 Epochs with a batch size of 32, a learning rate of 0.001, and a momentum of 0.9. We utilized Stochastic Gradient Descent (SGD) to find optimal weights and evaluated our models using Cross Entropy as the Loss Function. Additionally, we have also trained GoogLeNet and MobileNetV2 architecture using the Transfer Learning approach on the Crop Diseases dataset with the same experimental setting as earlier.

In our ablation study, we investigated the impact of data augmentation techniques, and hyperparameter tuning on the performance of the EfficientNet architecture on the Cassava dataset. We chose the Cassava dataset because its images contain background noise, which adds complexity to the classification task. Additionally, the EfficientNet architecture was selected due to its consistent performance in our main study. We experimented with batch sizes of 8, 16, 32, and 64, while the learning rate was uniformly sampled from a range of 0.0001 to 0.001. We utilized Ray 2.10 to track our experiments with 10 randomly sampled batch size and learning rate configurations.

The performance of models was also evaluated using accuracy, precision, recall, and F1 Score. Since the datasets were imbalanced, we employed macro-averaging techniques while computing precision, recall, and F1 Score. This approach gives equal weight to each class. Additionally, we also generated Confusion Matrices to gain a deeper insight into how often a particular class (ground truth) gets confused with other classes.

4.2. Main Results

Table 2 summarizes the performance of the chosen architectures, trained without pre-trained weights, on un-

seen images from the selected datasets. We observe that the performance of the three architectures varies across datasets. However, EfficientNet shows consistent competitive performance and generalizes better across all datasets in terms of accuracy, precision, and recall. This is likely due to its architectural design, which utilizes the squeeze-and-excitation technique in MBConv blocks and compound scaling. These features enable EfficientNet to capture complex features effectively.

Dataset	Metrics	GoogleNet	EfficientNet	MobileNetV2
Cassava	Accuracy (%)	59.65	64.96	66.55
	Precision (%)	51.39	51.09	58.14
	Recall (%)	45.20	47.84	44.13
	F1 Score	0.46	0.49	0.42
	Time/Epoch (sec)	45	41	42
Crop Disease	Accuracy (%)	78.92	78.38	74.07
	Precision (%)	77.81	76.46	73.43
	Recall (%)	79.63	76.98	70.33
	F1 Score	0.78	0.76	0.71
	Time/Epoch (sec)	101	97	83
Plant Village	Accuracy (%)	98.13	98.61	99.46
	Precision (%)	98.17	98.80	99.48
	Recall (%)	97.85	98.61	99.20
	F1 Score	0.98	0.98	0.99
	Time/Epoch (sec)	167	161	153

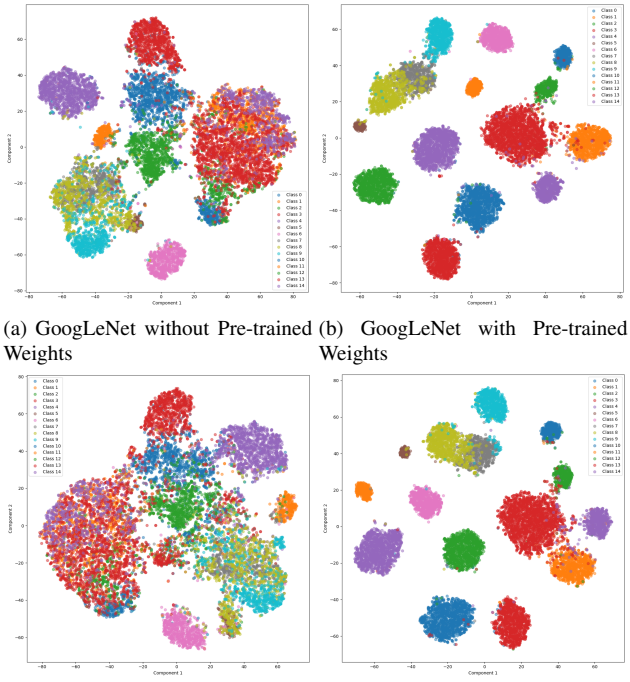
Table 2. Performance of 3 Architectures without Pre-trained weights on different Datasets

When analyzing the model performance across various datasets, we observe that on the Plant Village dataset, the three architectures achieve high accuracy, precision, and recall, with MobileNetV2 performing slightly better than both GoogLeNet and EfficientNet. This superior performance of all models can likely be attributed to the large training set and the simpler backgrounds in the input images compared to those in other datasets. The less complex backgrounds allow the models to focus more effectively on the key features of the plants, leading to improved performance. Apart from the dataset's reduced background noise, the performance of the MobileNetV2 architecture is also attributed to its use of inverted residuals with linear bottlenecks, which help capture fine-grained features with fewer parameters.

As the complexity of input images increases in the Crop Disease dataset, which features images with varying degrees of background noise across different classes, GoogLeNet tends to outperform both EfficientNet and MobileNetV2 in terms of precision and recall. This superior performance is likely attributable to the inception module within GoogLeNet, which enables the network to capture and integrate multi-scale features more effectively from the same input. This leads to a nuanced understanding of the images, resulting in improved precision and recall metrics. Table 3 illustrates the comparative performance of GoogLeNet and MobileNetV2 when utilizing pre-trained weights on this dataset. Furthermore, as depicted in Figure 4, MobileNetV2 demonstrates a performance comparable to GoogLeNet, suggesting it as a viable alternative when used with pre-trained weights, especially for deployment on edge devices.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1 Score
GoogleNet	87.90	87.23	88.21	0.88
MobileNet V2	88.68	87.79	88.59	0.88

Table 3. Transfer Learning: Performance of GoogleNet & MobileNet V2 Architectures on Crop Disease dataset



(a) GoogLeNet without Pre-trained Weights (b) GoogLeNet with Pre-trained Weights

(c) MobileNetV2 without Pre-trained Weights (d) MobileNetV2 with Pre-trained Weights

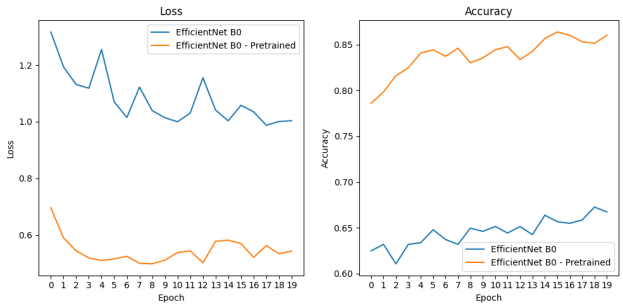
Figure 4. t-SNE for four models on train set of Crop Disease dataset

With input images featuring the highest degree of background noise, as seen in the Cassava dataset, MobileNetV2 outperforms other network architectures, achieving an accuracy of 66.55%, a precision of 58.14%, and a recall of 44.13%. While the high accuracy could be attributed to the utilization of inverted residuals with linear bottlenecks which helps capture fine-grained embedded features in the input, we observe a significant trade-off in precision and recall compared to other architectures. Figure 34 shows the model's confidence in identifying a subset of classes (class 1 and 3) precisely but struggling with others, leading to lower recall. Despite having better performance on the Crop Disease dataset, the relative complexity of GoogLeNet hinders its performance on this smaller dataset, making it challenging to learn nuanced patterns in the images.

4.3. Ablative Study

In our ablation study, we investigate the impact of employing additional data augmentation techniques, varying the number of images per class, applying transfer learning, and tuning hyperparameters such as batch size and learning rate on the Cassava dataset using the EfficientNet architecture. Among the three datasets examined, we found the Plant Village dataset to be the easiest to classify due to the minimal background noise in the inputs, followed by the Crop Diseases dataset. The Cassava dataset presents the most significant classification challenge, as evidenced by the results presented in Table 2. This difficulty arises from the high level of background noise in the images, which impedes the model's ability to capture essential features. To mitigate the effects of high background noise, we employed center cropping, as seen in Figure 9. Although this transformation reduces background noise by reducing the dimensions of an image while focusing on the central region, which contains the subject of interest, it is not as effective as segmentation and leads to no improvement in EfficientNet's performance.

As we observed an imbalance in the dataset impacting the model's performance, we reduced the number of images for the oversampled classes. Specifically, we sampled only 15% of the images for the "CMD" class and 50% for the "CBSD" class. Upon retraining the EfficientNet for 20 epochs on this modified dataset with a batch size of 32 and a learning rate of 0.001, we noted that the model was unable to generalize well due to the decreased overall dataset size. This necessitated an increase in the number of epochs and tweaking of hyper-parameters leading to a slower convergence to optimal solution.



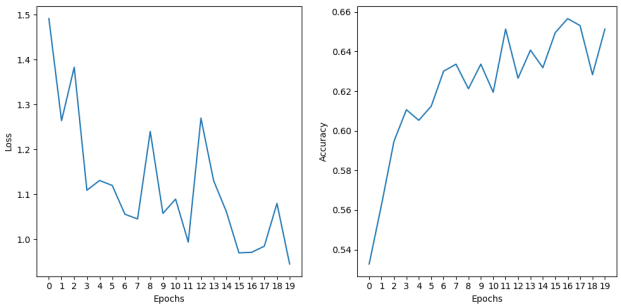
(a) Loss (b) Accuracy

Figure 5. EfficientNet with and without pre-trained weights on Cassava dataset

We also investigated the impact of training EfficientNet with pre-trained weights on the original Cassava dataset through a transfer learning approach. When equipped with pre-trained weights, EfficientNet achieved an accuracy of 84.78%, a precision of 80.91%, and a recall of 78.31%. As depicted in Figure 5, the model benefits from the use of pre-trained weights, which act as powerful feature extractors, resulting in approximately 20% improvement in accuracy over the non-pretrained model. This leads to faster convergence and improved performance across the dataset.

Batch Size	Learning Rate	Loss	Accuracy (%)
8	0.0013	1.53	62.48
32	0.0095	4.90	53.99
16	0.0001	1.07	61.95
32	0.0039	1.04	66.02
8	0.0086	1.16	59.23
64	0.0026	0.94	65.13
8	0.0096	3.66	46.19
64	0.0095	1.38	61.59
16	0.0028	1.77	53.63
8	0.0021	16.54	52.74

Table 4. Summary of Hyper-parameter Tuning Trials



(a) Loss (b) Accuracy

Figure 6. EfficientNet on Cassava dataset with a batch size of 64 and a learning rate of 0.002621

Additionally, we optimized the batch size and learning rate for EfficientNet using the Random Search algorithm. We randomly sampled 10 configurations from the specified batch sizes: 8, 16, 32, 64, with learning rates ranging from 0.0001 to 0.001. The performance of the model under these configurations, as illustrated in Table 4, underscores the critical role of hyperparameter tuning in achieving a balance between training stability and model performance. Our observations indicate that larger batch sizes, particularly 64, when paired with a moderate learning rate of 0.0026, tend to offer an optimal balance by providing training stability and a model loss of less than 1. Conversely, smaller batch sizes combined with moderate to high learning rates (e.g., 0.0028 or 0.0096) appear to induce training instability, resulting in a loss greater than 1. With an optimal configuration of a batch size of 64 and a learning rate of 0.0026, EfficientNet achieved an accuracy of 70.09%, precision of 59.33%, and recall of 46.34% over unseen images of the Cassava dataset.

References

[1] Mwebaze, Ernest, et al. *"iCassava 2019 Fine-Grained Visual Categorization Challenge Dataset."* Google Research, Artificial Intelligence Lab Makerere University, National Crops Resources Research Institute. Accessed [Access Date], https://tensorflow.google.cn/datasets/catalog/cassava. 2

[2] Mensah Kwabena, Patrick, et al. *"Dataset for Crop Pest and Disease Detection."* Mendeley Data, vol. 1, 2023, doi:10.17632/bwh3zbpkpv.1. https://data.mendeley.com/datasets/bwh3zbpkpv/1 2

[3] Hughes, D. P., and M. Salathé. *"An open access repository of images on plant health to enable the development of mobile disease diagnostics through machine learning and crowdsourcing."* CoRR, vol. abs/1511.08060, 2015, http://arxiv.org/abs/1511.08060. https://www.tensorflow.org/datasets/catalog/plant_village 2

[4] Mohanty, Sharada P., David P. Hughes, and Marcel Salathé. *"Using deep learning for image-based plant disease detection."* Frontiers in plant science 7 (2016): 1419.

[5] J. Deng, W. Dong, R. Socher, L. -J. Li, Kai Li and Li Fei-Fei, *"ImageNet: A large-scale hierarchical image database,"* 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 2009, pp. 248-255. 3

[6] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... and Rabinovich, A. (2015). *"Going deeper with convolutions"*. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

[7] Tan, Mingxing, and Quoc Le. *"Efficientnet: Rethinking model scaling for convolutional neural networks."* International conference on machine learning. PMLR, 2019.

[8] Sandler, Mark, et al. *"Mobilenetv2: Inverted residuals and linear bottlenecks"*. Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

[9] Paszke, Adam, et al. *"Pytorch: An imperative style, high-performance deep learning library"*. Advances in neural information processing systems 32 (2019).

[10] Punam Bedi, Pushkar Gole, Plant Disease Detection Using Hybrid Model Based on Convolutional Autoencoder and Convolutional Neural Network, *Artificial Intelligence in Agriculture*, Volume 5, 2021, Pages 90-101, ISSN 2589-7217.

[11] Hassan, S.M.; Maji, A.K.; Jasiński, M.; Leonowicz, Z.; Jasińska, E. Identification of Plant-Leaf Diseases Using CNN and Transfer-Learning Approach. *Electronics* 2021, *10*, 1388.

[12] Shelar, Nishant; Shinde, Suraj; Sawant, Shubham; Dhumal, Shreyash; Fakir, Kausar. Plant Disease Detection Using CNN. *ITM Web Conf.* 2022, 44, 03049. 2

[13] Nandi, Rabindra Palash, Aminul Siddique, Nazmul Zilani, Mohammed. (2022). Device-friendly Guava fruit and leaf disease detection using deep learning. 1, 2

[14] Boulent J, Foucher S, Théau J, St-Charles PL. Convolutional Neural Networks for the Automatic Identification of Plant Diseases. Front Plant Sci. 2019 Jul 23;10:941. doi: 10.3389/fpls.2019.00941. PMID: 31396250; PMCID: PMC6664047. 1, 2

[15] Sharma, Rahul Singh, Amar ., Kavita Jhanjhi, Noor Masud, Mehedi Jaha, Emad Verma, Prof. (2021). Plant Disease Diagnosis and Image Classification Using Deep Learning. Computers, Materials and Continua. 71. 2125-2140. 10.32604/cmc.2022.020017. 2

[16] Khyeh0719.*"Pytorch EfficientNet Baseline Train AMP Aug"*. Kaggle, https://www.kaggle.com/code/khyeh0719/pytorch-efficientnet-baseline-train-amp-aug. Accessed 10 April 2024.

[17] Sik-Ho Tsang. "Review: MobileNetV2 — Light Weight Model for Image Classification." Towards Data Science. https://towardsdatascience.com/review-mobilenetv2-light-weight-model-image-classification-8febb490e61c. Accessed 10 April 2024. 3

[18] Sandler, Mark, et al. "Mobilenetv2: Inverted residuals and linear bottlenecks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018. 3

[19] Petru Potrimba. "What is EfficientNet?" Roboflow Blog. https://blog.roboflow.com/what-is-efficientnet/. Accessed 10 April 2024. 4

[20] DhanushKumar. "EfficientNet: Scaling Depth, Width, Resolution." Medium. https://medium.com/@danushidk507/efficientnet-scaling-depth-width-resolution-11e2d4311357. Accessed 10 April 2024. 4

[21] Tan, Mingxing, and Quoc V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks." International Conference on Machine Learning, 2019 4

[22] Sandler, Mark, et al. "Mobilenetv2: Inverted residuals and linear bottlenecks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018. 3

[23] Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015. 3

[24] Ruder, Sebastian. "An overview of gradient descent optimization algorithms." arXiv preprint arXiv:1609.04747 (2016). 4

5. Supplementary Material

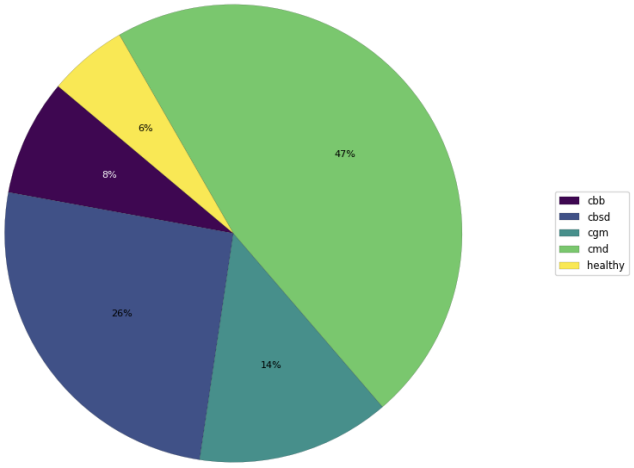


Figure 7. Inter-Class variance of Cassava Dataset

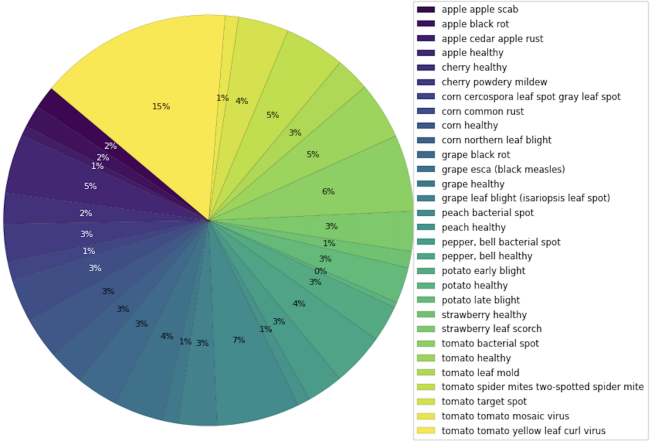


Figure 8. Inter-Class variance of Plant Village Dataset

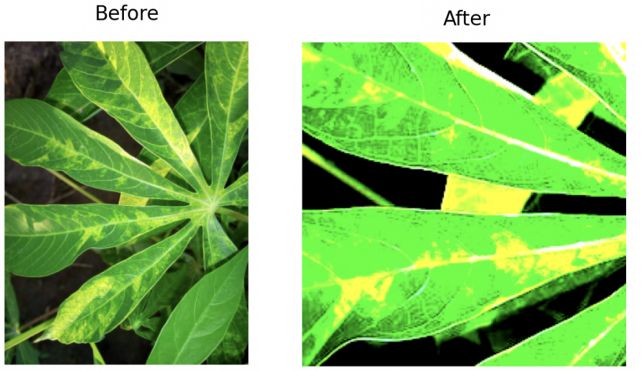


Figure 9. Example of center cropping on Cassava dataset

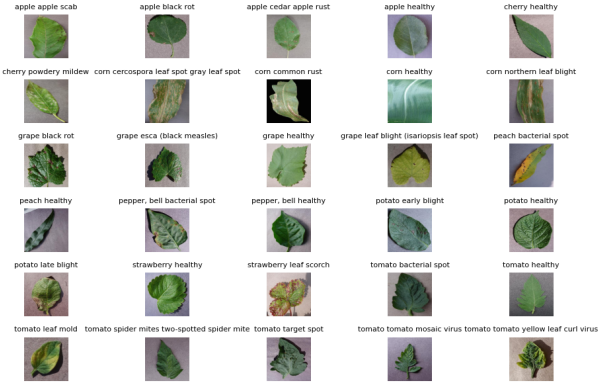


Figure 10. Samples of Plant Village Dataset

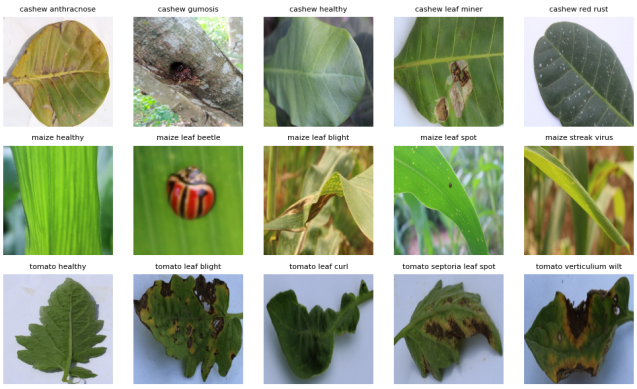


Figure 11. Samples of Crop Disease Dataset

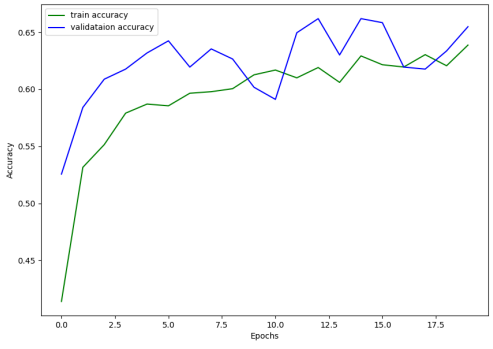


Figure 12. Accuracy of GoogleNet on Cassava Dataset

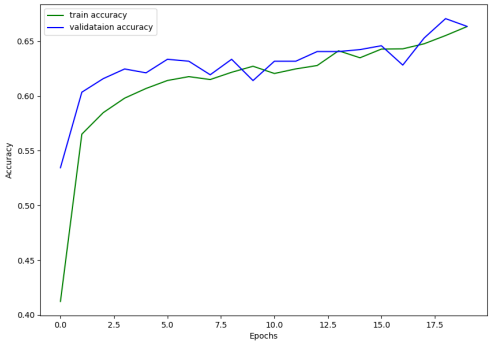


Figure 13. Accuracy of EfficientNet B0 on Cassava Dataset

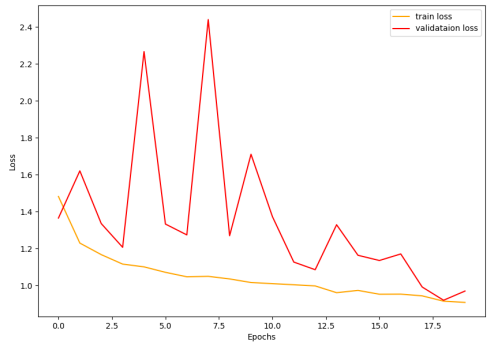


Figure 16. Loss of EfficientNet B0 on Cassava Dataset

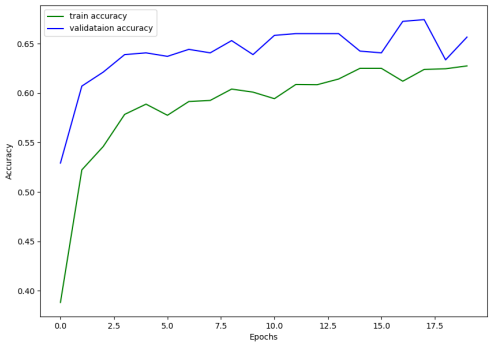


Figure 14. Accuracy of MobileNetV2 on Cassava Dataset

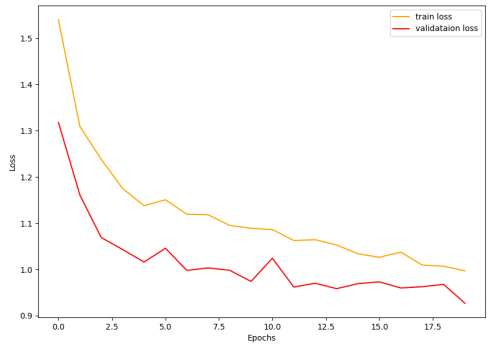


Figure 17. Loss of MobileNetV2 on Cassava Dataset

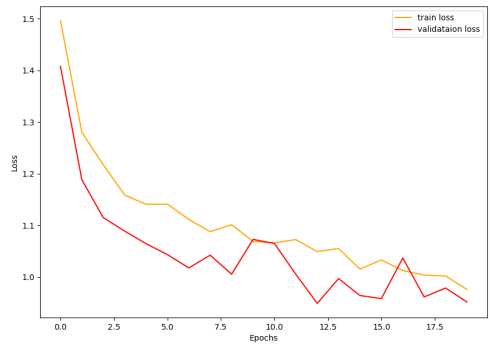


Figure 15. Loss of GoogleNet on Cassava Dataset

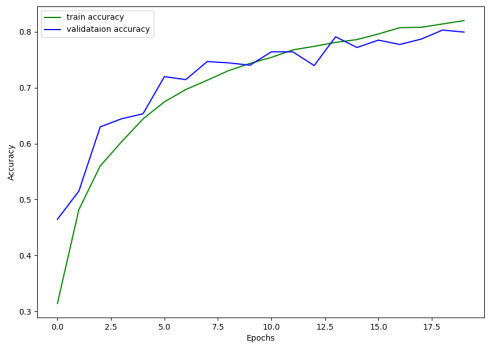


Figure 18. Accuracy of GoogleNet on Crop Disease Dataset

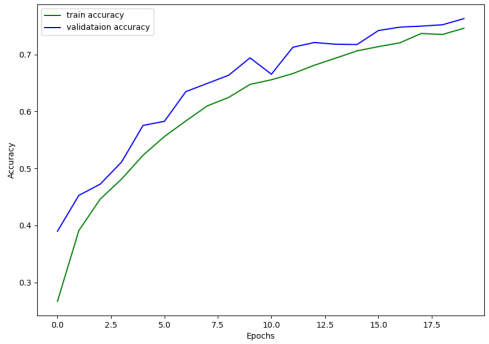


Figure 19. Accuracy of EfficientNet B0 on Crop Disease Dataset

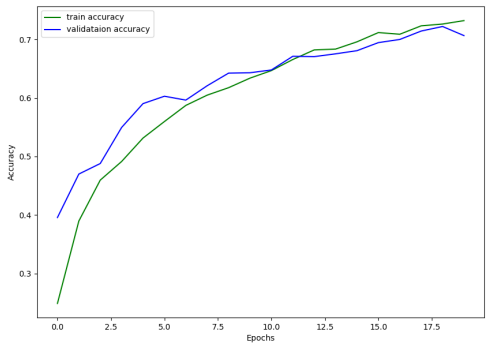


Figure 20. Accuracy of MobileNetV2 on Crop Disease Dataset

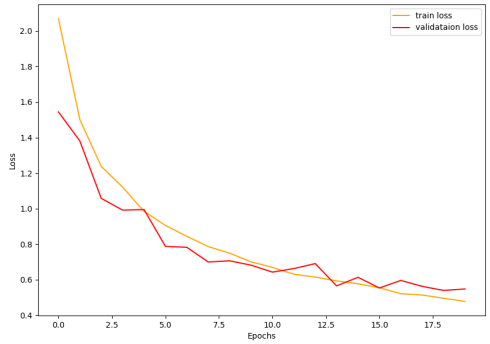


Figure 21. Loss of GoogleNet on Crop Disease Dataset

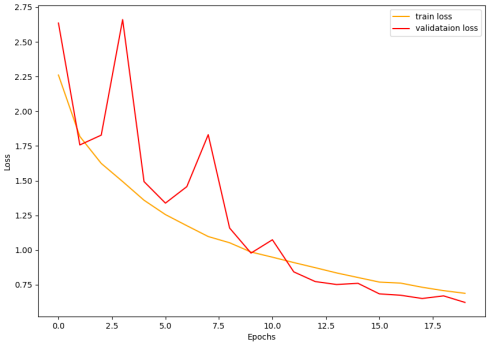


Figure 22. Loss of EfficientNet B0 on Crop Disease Dataset

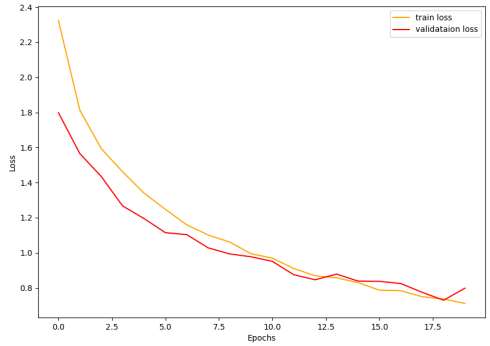


Figure 23. Loss of MobileNetV2 on Crop Disease Dataset

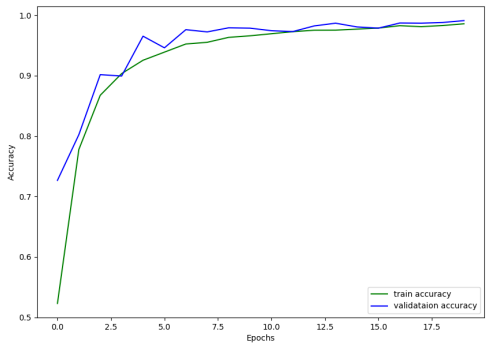


Figure 24. Accuracy of GoogleNet on Plant Village Dataset

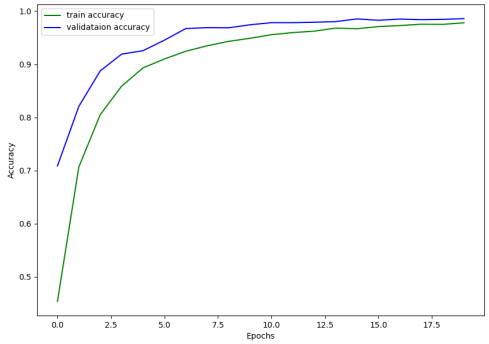


Figure 25. Accuracy of EfficientNet B0 on Plant Village Dataset

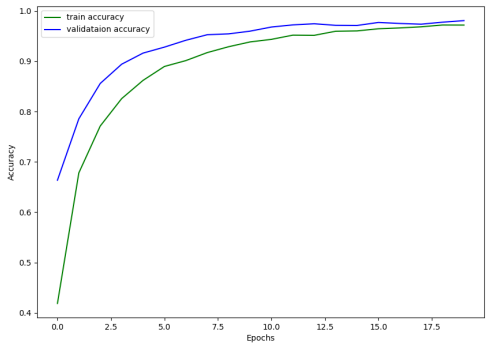


Figure 26. Accuracy of MobileNetV2 on Plant Village Dataset

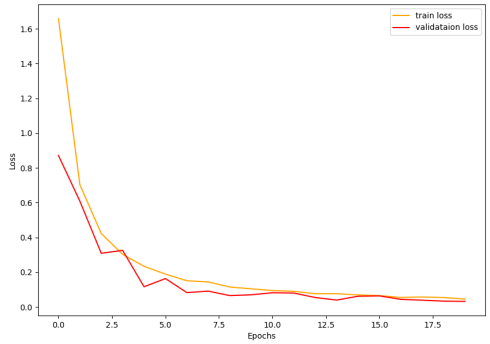


Figure 27. Loss of GoogleNet on Plant Village Dataset

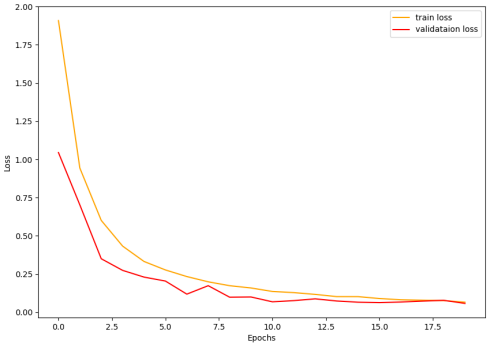


Figure 28. Loss of EfficientNet B0 on Plant Village Dataset

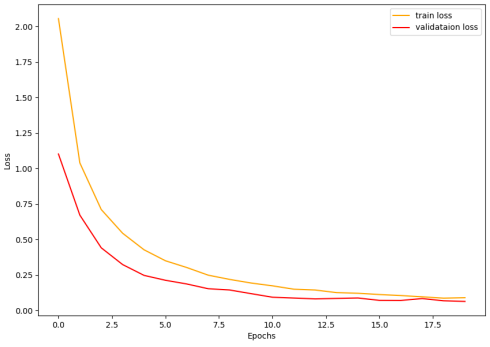


Figure 29. Loss of MobileNetV2 on Plant Village Dataset

5.1. Transfer Learning

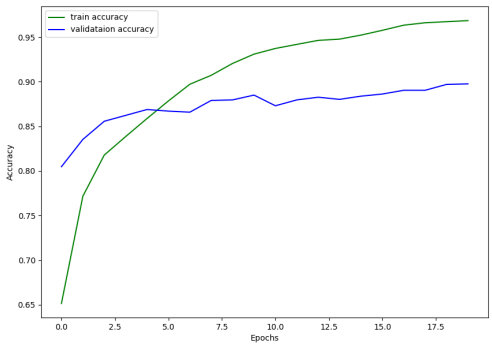


Figure 30. Accuracy-GoogleNet

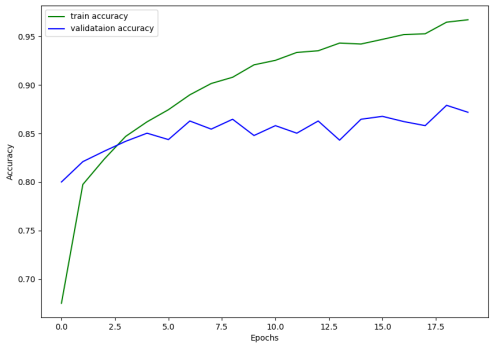


Figure 31. Accuracy-MobileNetV2

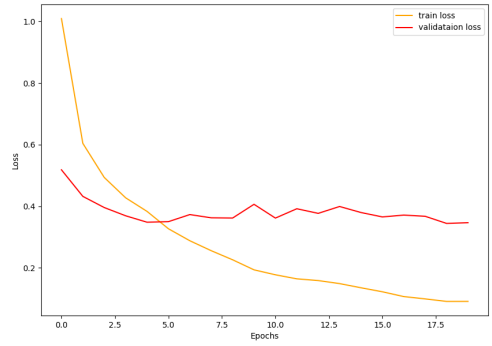


Figure 32. Loss-GoogleNet

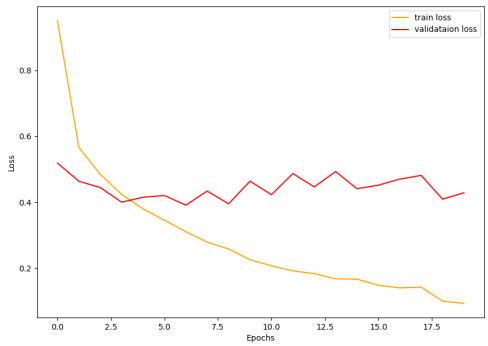


Figure 33. Loss-MobileNetV2

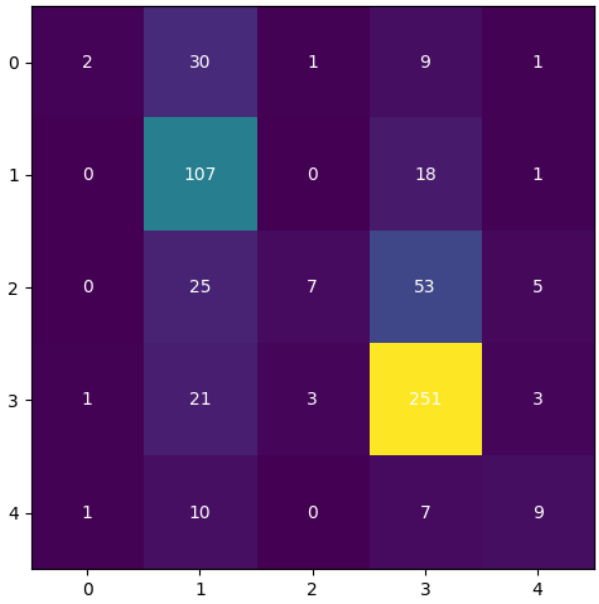


Figure 34. Confusion Matrix for MobileNetV2 on Cassava dataset