

000

054

001

055

002

056

## 1. Abstract

003

057

004

058

005

059

006

060

007

061

008

062

009

063

010

064

011

065

012

066

013

067

014

068

015

069

016

070

017

071

018

072

019

073

020

074

021

075

022

076

023

077

024

078

025

079

026

080

027

081

028

082

029

083

030

084

031

085

032

086

033

087

034

088

035

089

036

090

037

091

038

092

039

093

040

094

041

095

042

096

043

097

044

098

045

099

046

100

047

101

048

102

049

103

050

104

051

105

052

106

053

107

# Plant Disease Detection and Classification using Deep Learning

ture 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

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 archi-

108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
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. Bouleit 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 com-

pare 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].

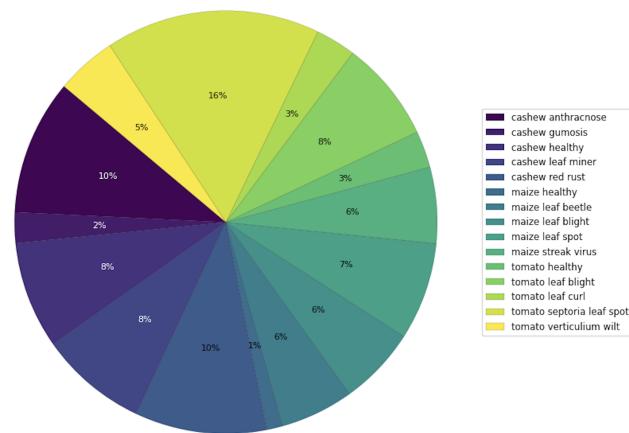


Figure 1. Inter-Class variance of Crop Disease Dataset

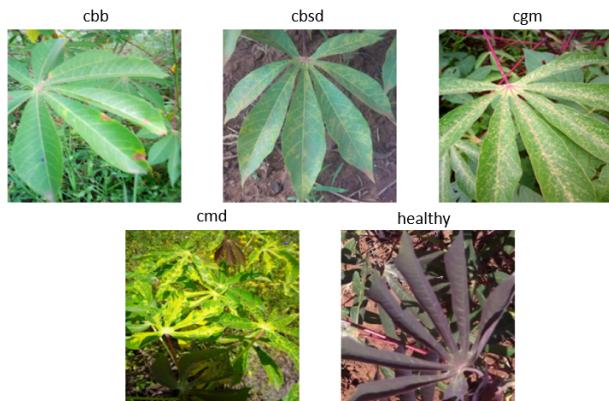


Figure 2. Samples of Cassava Dataset

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

162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215

216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
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 224x224x3, 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
<b>Images</b>	5,656	24,881	50,091
<b>Size</b>	Var	400x400x3	256x256x3
<b>Color Space</b>	RGB	RGB	RGB
<b>Format</b>	JPG	JPG	JPG
<b>Classes</b>	5	22	37
<b>Trimmed Datasets</b>			
<b>Images</b>	5,656	16,704	35,320
<b>Classes</b>	5	15	30

247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
Table 1. Agricultural Datasets Summary

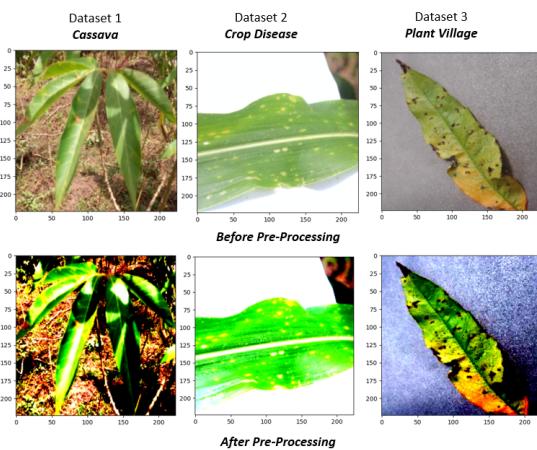


Figure 3. Image samples before and after data augmentation

These pre-processing steps, as seen in Figure 3, including resizing, applying geometric and photo-metric transformations, and normalization according to ImageNet stan-

dards, 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 ( $\times 2$ ) 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

## GROUP Q: FINAL REPORT

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

339 GoogleNet and MobileNetV2 were chosen for transfer  
 340 learning as these models support fine-tuning which enabled  
 341 us to tweak and optimize the models' parameters according  
 342 to our target task. By leveraging the pre-trained weights, we  
 343 could adapt these models to the unique features and char-  
 344 acteristics of our dataset resulting in an enhanced perfor-  
 345 mance. Overall, the models which are popular and effec-  
 346 tive in image classification tasks were chosen while making  
 347 sure to address plant disease detection considering compu-  
 348 tational efficiency and performance. More information on  
 349 time/epoch for all models is given in Table 2.

### 351 3.3. Optimization Algorithm

352 In our study, the choice of optimization algorithm was  
 353 critical to effectively train the CNN models, considering  
 354 the challenges posed by the complex image data and the  
 355 need for efficient computational performance. We have pri-  
 356 marily used Stochastic Gradient Descent (SGD) as our opti-  
 357 mization algorithm across all experiments which has helped  
 358 maintain a good training and validation accuracy over mul-  
 359 tiple epochs.

360 We have chosen SGD for its robustness and effective-  
 361 ness in handling large datasets and its ability to converge  
 362 even with noisy gradients, which is seen in real-world im-  
 363 age data like those in our datasets. While batch gradient de-  
 364 scent, which computes the gradient using the entire dataset,  
 365 SGD updates the parameters more frequently with gradi-  
 366 ents calculated from randomly selected subsets of the data  
 367 or mini-batches [24]. We have set the Learning Rate to  
 368 0.001, which is a standard value allowing for gradual but  
 369 steady learning. While performing hyper-parameter tun-  
 370 ing, we have also sampled learning rates uniformly from  
 371 the range 0.0001 to 0.001 and chosen the best configura-  
 372 tion. The momentum coefficient of 0.9 was also used to  
 373 help accelerate the SGD in the relevant direction, thereby  
 374 avoiding local minima and faster convergence. A batch size  
 375 of 32 was used across the datasets while a study on differ-  
 376 ent batch sizes of 8, 16, 32, and 64 was tested for hyper-

377 parameter tuning. Furthermore, we assessed the model's  
 378 performance using metrics such as F1-score, Precision, Re-  
 379 call, and Accuracy. By leveraging the optimization algo-  
 380 rithm, we achieved better accuracy rates, showcasing the  
 381 significance of optimization algorithms in enhancing model  
 382 performance. We can see the Accuracy and Learning Loss  
 383 plots of the different architectures across datasets in Figures  
 384 12-33.

## 385 4. Results

### 386 4.1. Experimental Setup

387 The experiments were run on the local with Python 3.9  
 388 and PyTorch 2.2. We leveraged both an NVIDIA RTX  
 389 3080 Ti GPU and Apple M2 Silicon for acceleration. Three  
 390 CNN architectures, GoogLeNet, EfficientNet B0, and Mo-  
 391 bileNetV2, were used for experimentation. The datasets  
 392 were split in the ratio of 80:10:10 for training, validation,  
 393 and testing, respectively.

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

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

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

### 424 4.2. Main Results

425 Table 2 summarizes the performance of the chosen ar-  
 426 chitectures, trained without pre-trained weights, on un-

## GROUP Q: FINAL REPORT

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

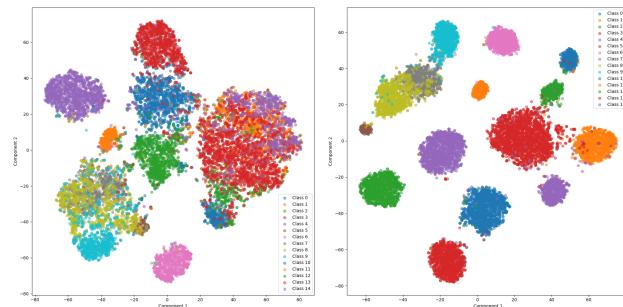
When analyzing the model performance across various datasets, we observe that on the Plant Village dataset, the three architectures achieve high accuracy, precision, and recall, with MobileNetV2 performing slightly better than both GoogLeNet and EfficientNet. This superior performance of all models can 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

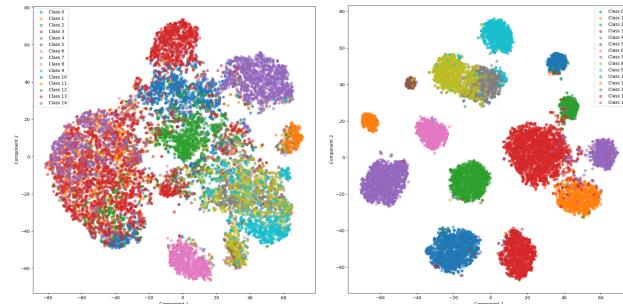
4, MobileNetV2 demonstrates a performance comparable to GoogLeNet, suggesting it as a viable alternative when used with pre-trained weights, especially for deployment on edge devices.

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.

## GROUP Q: FINAL REPORT

540

**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.

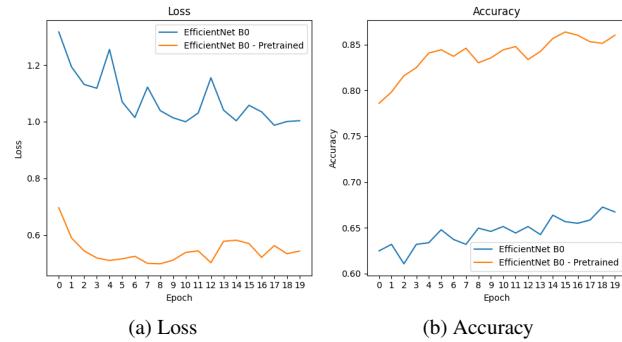


Figure 5. EfficientNet with and without pre-trained weights on Cassava dataset

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-

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

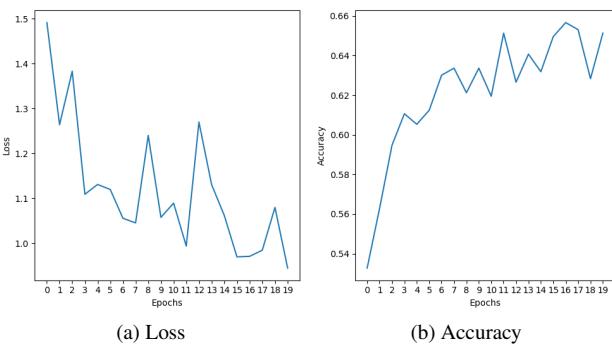


Figure 6. EfficientNet on Cassava dataset with a batch size of 64 and a learning rate of 0.002621

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.

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., 0.00028 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.

594

595

596

597

598

599

600

601

602

603

604

605

606

607

608

609

610

611

612

613

614

615

616

617

618

619

620

621

622

623

624

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

## GROUP Q: FINAL REPORT

648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701

## 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>. 2 702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755
- [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](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. 3
- [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] Boulet 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

## GROUP Q: FINAL REPORT

- 756 [21] Tan, Mingxing, and Quoc V. Le. "EfficientNet: Re- 810  
757 thinking Model Scaling for Convolutional Neural Net- 811  
758 works." International Conference on Machine Learn- 812  
759 ing, 2019. 4 813  
760
- 761 [22] Sandler, Mark, et al. "Mobilenetv2: Inverted residuals 814  
762 and linear bottlenecks." Proceedings of the IEEE con- 815  
763 ference on computer vision and pattern recognition. 816  
764 2018. 3 817  
765
- 766 [23] Szegedy, Christian, et al. "Going deeper with convo- 818  
767 lutions." Proceedings of the IEEE conference on com- 819  
768 puter vision and pattern recognition. 2015. 3 820  
769
- 770 [24] Ruder, Sebastian. "An overview of gradient de- 821  
771 scent optimization algorithms." arXiv preprint 822  
772 arXiv:1609.04747 (2016). 4 823  
773
- 774
- 775
- 776
- 777
- 778
- 779
- 780
- 781
- 782
- 783
- 784
- 785
- 786
- 787
- 788
- 789
- 790
- 791
- 792
- 793
- 794
- 795
- 796
- 797
- 798
- 799
- 800
- 801
- 802
- 803
- 804
- 805
- 806
- 807
- 808
- 809

## GROUP Q: FINAL REPORT

864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917

## 5. Supplementary Material

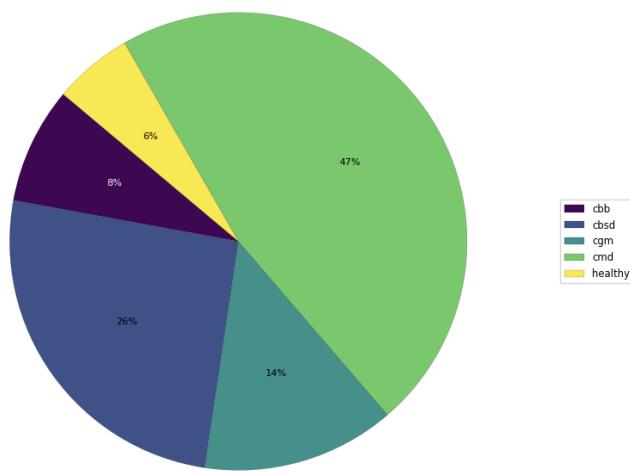


Figure 7. Inter-Class variance of Cassava Dataset

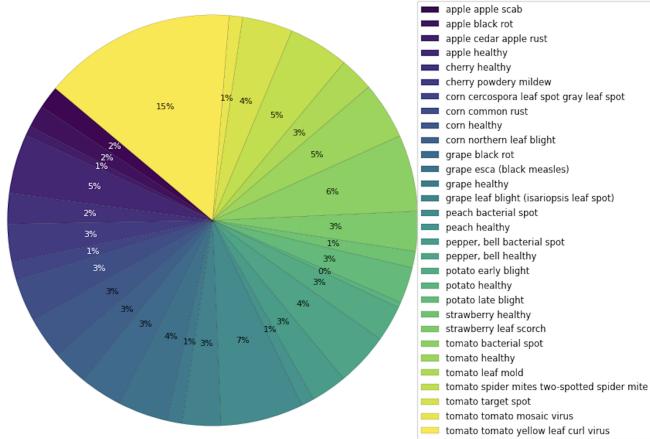


Figure 8. Inter-Class variance of Plant Village Dataset



Figure 9. Example of center cropping on Cassava dataset



Figure 10. Samples of Plant Village Dataset



Figure 11. Samples of Crop Disease Dataset

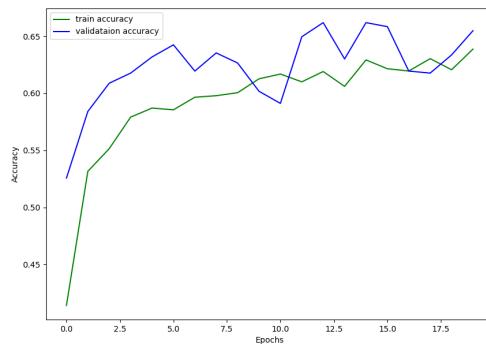


Figure 12. Accuracy of GoogleNet on Cassava Dataset

918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971

## GROUP Q: FINAL REPORT

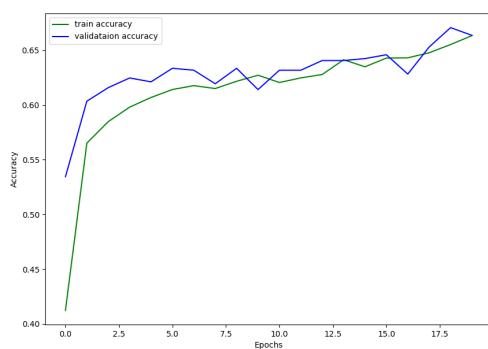


Figure 13. Accuracy of EfficientNet B0 on Cassava Dataset

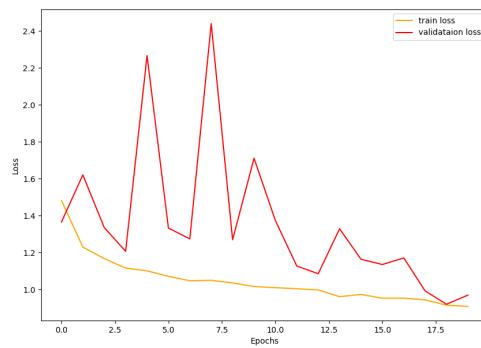


Figure 16. Loss of EfficientNet B0 on Cassava Dataset

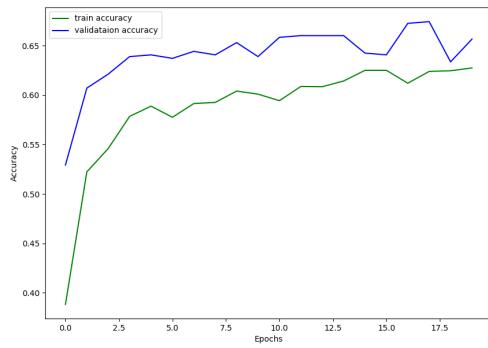


Figure 14. Accuracy of MobileNetV2 on Cassava Dataset

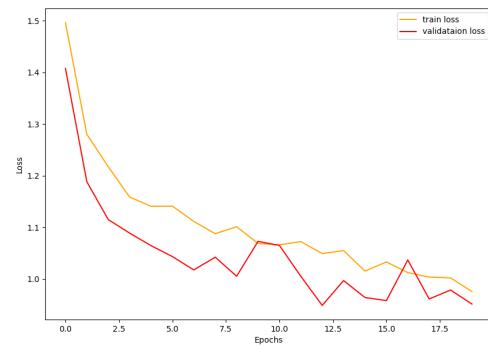


Figure 15. Loss of GoogleNet on Cassava Dataset

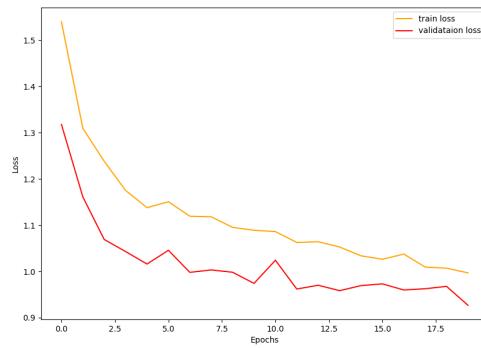


Figure 17. Loss of MobileNetV2 on Cassava Dataset

## GROUP Q: FINAL REPORT

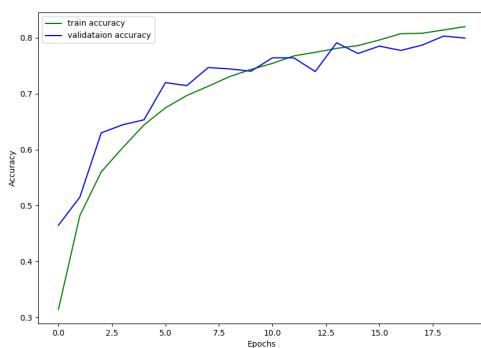


Figure 18. Accuracy of GoogleNet on Crop Disease Dataset

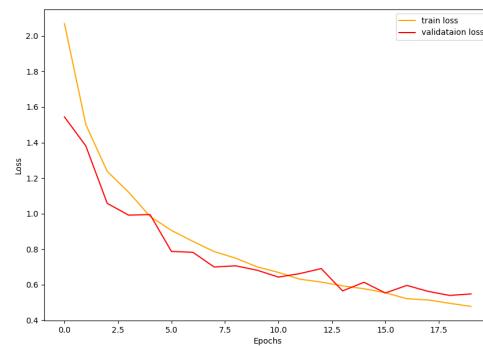


Figure 21. Loss of GoogleNet on Crop Disease Dataset

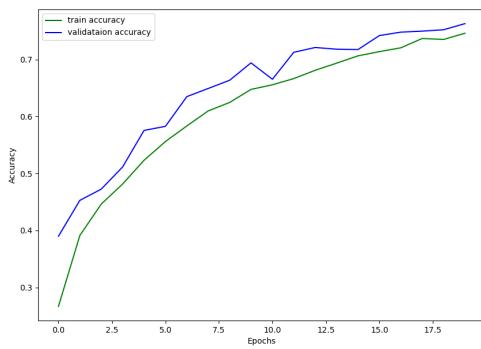


Figure 19. Accuracy of EfficientNet B0 on Crop Disease Dataset

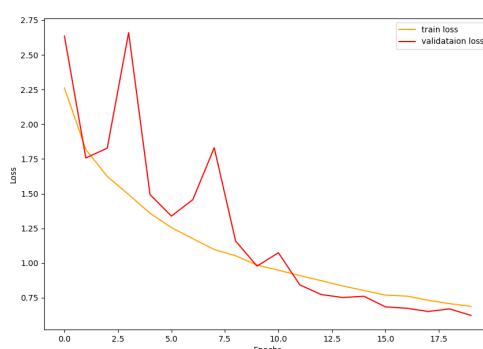


Figure 22. Loss of EfficientNet B0 on Crop Disease Dataset

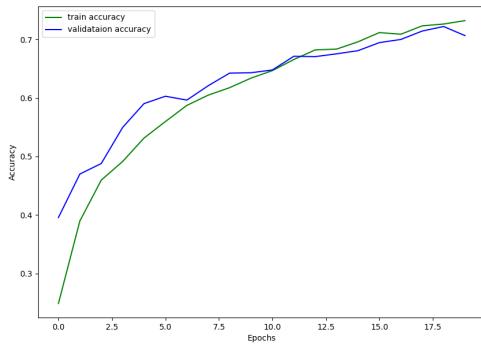


Figure 20. Accuracy of MobileNetV2 on Crop Disease Dataset

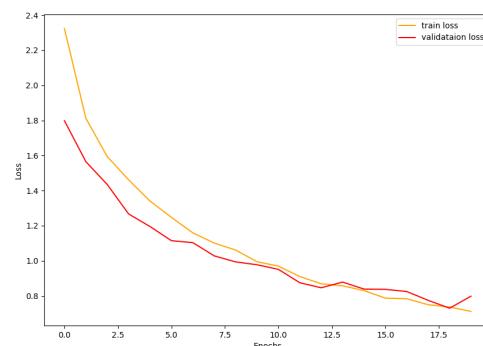


Figure 23. Loss of MobileNetV2 on Crop Disease Dataset

## GROUP Q: FINAL REPORT

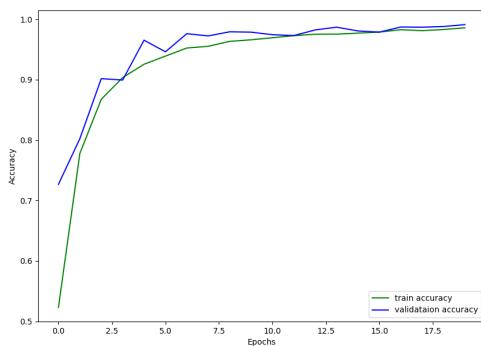


Figure 24. Accuracy of GoogleNet on Plant Village Dataset

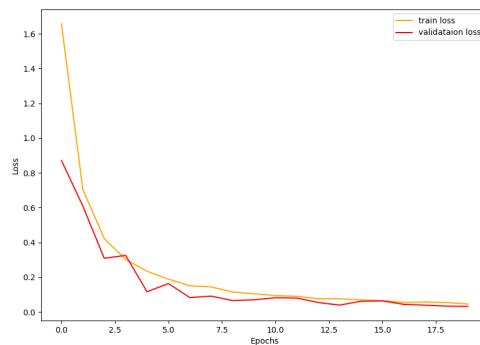


Figure 27. Loss of GoogleNet on Plant Village Dataset

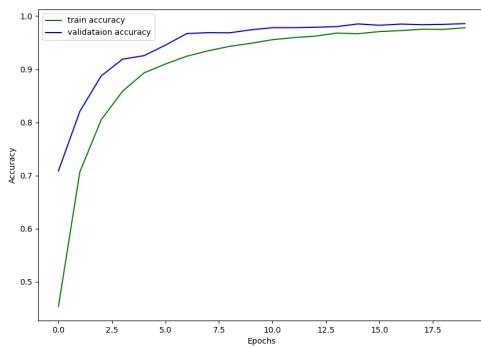


Figure 25. Accuracy of EfficientNet B0 on Plant Village Dataset

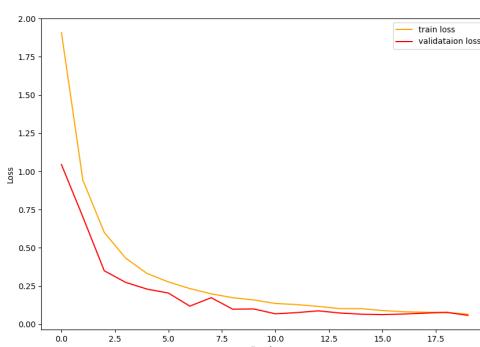


Figure 28. Loss of EfficientNet B0 on Plant Village Dataset

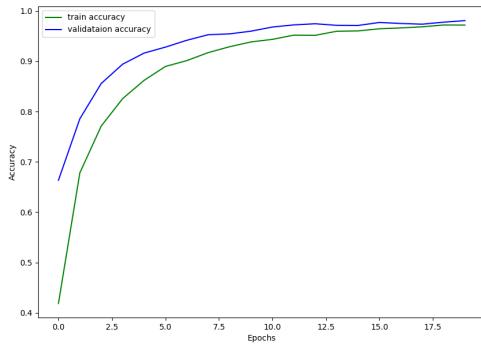


Figure 26. Accuracy of MobileNetV2 on Plant Village Dataset

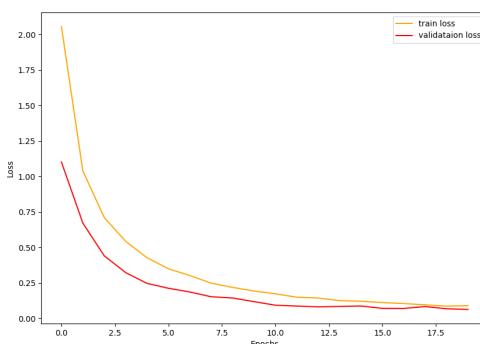
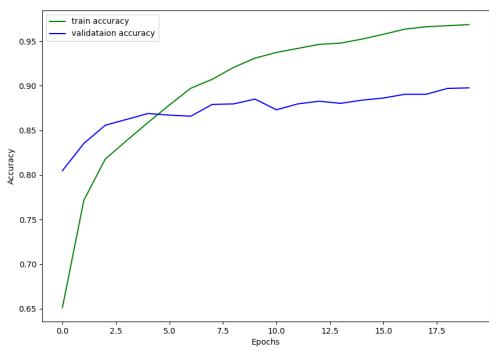


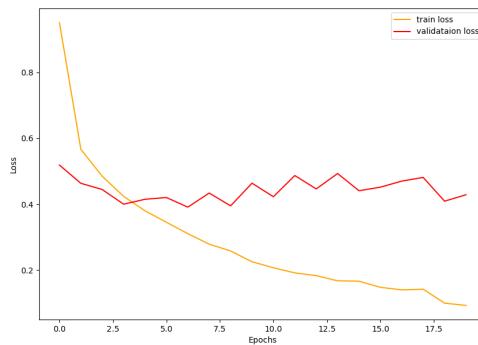
Figure 29. Loss of MobileNetV2 on Plant Village Dataset

1296   **5.1. Transfer Learning**

1350



1312   Figure 30. Accuracy-GoogleNet



1313   Figure 33. Loss-MobileNetV2

1351

1352

1353

1354

1355

1356

1357

1358

1359

1360

1361

1362

1363

1364

1365

1366

1367

1368

1369

1370

1371

1372

1373

1374

1375

1376

1377

1378

1379

1380

1381

1382

1383

1384

1385

1386

1387

1388

1389

1390

1391

1392

1393

1394

1395

1396

1397

1398

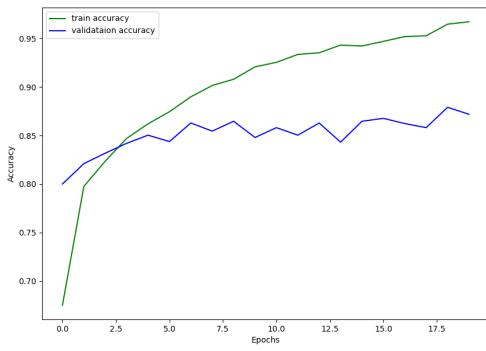
1399

1400

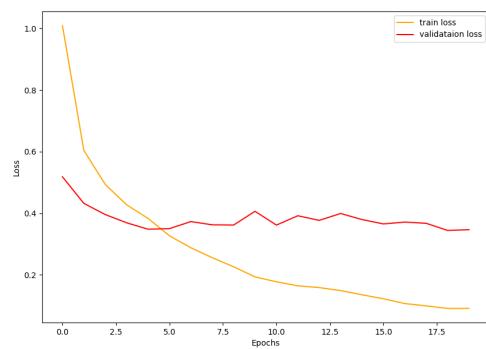
1401

1402

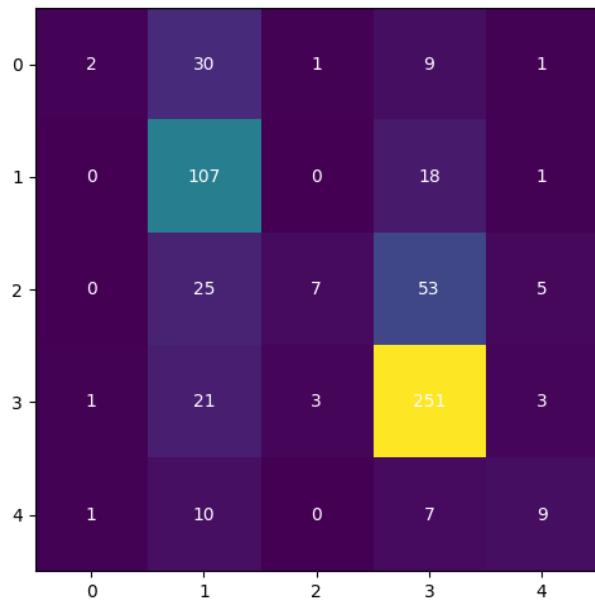
1403



1327   Figure 31. Accuracy-MobileNetV2



1343   Figure 32. Loss-GoogleNet



1344   Figure 34. Confusion Matrix for MobileNetV2 on Cassava dataset