Sesame Plant Disease Classification

Ahmed Omer Abdelaziz Musa[1[7015711]], Ibrahim Siddig Ibrahim Eltayeb[1[7015925]], and Sujatha Senthurnathan[1[7010492]]

University of Saarland
High Level Computer Vision Course
Team 21

Abstract. In this project we explore the plant disease classification problem on a dataset of sesame leaves of 1453 images, containing three classes, including healthy plants. Resnet50 architecture pre-trained on ImageNet was used, in combination with other methods and techniques to increase the test accuracy and overcome overfitting and the challenge of small dataset size. Experiments using data augmentation, transfer learning and self-supervised pre-training were carried out. Empirical results showed that data augmentation and multi-step transfer learning with a dataset that shares similarity (both are leaves in this case) showed the best performance (pre-trained weights on ImageNet, followed by pre-training on the Cassava Leaf Challenge dataset), reaching an accuracy of 96.71%. Adding a self-supervised angle classification pre-training phase decreased the performance, which can be explained by the learned features being less useful than the ones from transfer learning. Experiments on freezing the model and training only the last layer were also performed and analyzed, achieving a result of 90.9%. The significantly high success rate makes the model a very useful advisory or early warning tool, and an approach that could be further expanded to support an integrated plant disease identification system to operate in real cultivation conditions.

Keywords: Data-Augmentation · Mix-Up · Self-supervised · ResNet50 · Transfer-learning · Classification · Plant disease · RandAugment · ImageNet Policy

1 Introduction

Plant disease is an abnormal state of a plant that prevents it from growing normally. Plant diseases come in a variety of forms, each of which can result in significant crop losses. Early detection of diseases may aid in taking action to prevent losses and generate a high-quality harvest of excellent grain.

Weed growth, insect pests, and disease are all problems that sesame seed producers face. To protect the quantity and quality of crops, it is vital to detect plant diseases in their early stages with great accuracy. In Sudan(where the data was collected), two primary diseases damage the sesame crop: bacterial leaf spot and gall[2]. It is critical to detect infections early on so that they do not spread and leave the entire harvest unusable. Classification is done with the naked sight,

which leads to a lot of human error and a lot of labor, which results in a large increase in costs.

Deep-learning-based techniques, particularly CNNs, are the most promising approaches for the classification problem. Deep learning suffers from some drawbacks; If the dataset does not have enough images the neural network may not perform well. [9] We make use of several techniques to handle the issue of data shortage. Transfer learning is one of the techniques used to improve performance of the neural network without needing a large amount of data to train the network[15]. Another technique is data augmentation which is used to generate more data by applying a set of transforms such as horizontal flip[14]. Self-supervised learning can also be used to learn some pattern from the data by applying a self-supervised task on the model with the dataset before using the model on the main task[12].

1.1 Goals and Objectives

- The baseline paper reports overfitting, to combat this and the problem of small dataset size we use three methods: Transfer Learning, Data Augmentation, and Self-Supervised Pre-training.
- Get access to, filter and prepare the datasets: Sesame Plant Disease dataset, and Cassava Leaf Disease Challenge dataset
- Align the Sesame Plant Disease dataset for the purpose of rotation (angle) classification self-supervised learning.
- Use ResNet50 architecture with pre-trained weights on ImageNet (transfer learning).
- Apply Data augmentation techniques: RandAugment, ImageNetPolicy and MixUp.
- Apply a second transfer learning pre-training phase, by training the model to learn plant disease patterns on the Cassava Leaf Challenge dataset
- Apply a self-supervised learning pre-training phase, using rotation and angle detection task.
- Fine-tune the model weights to fit by training on the Sesame Plant Disease dataset.
- Tune the hyper-parameters
- Analyze and compare the experimental results.
- Suggest further improvements and different approaches to increase the performance.

2 Related work

Our baseline work, Bashier et al. [2] collect a dataset for Sesame plants with three classes, healthy, Gall midge diseased plants, and baterial infected plants. They try different architectures and report a test accuracy of **61.65% for ResNet50**, and their best results from **VGG16 and VGG19 are 91.15%** (see Table 1).

Liu and Wang [9] offer a review of the methods and challenges of plant disease detection methods based on deep learning. They highlight the challenge of small

dataset size, suggesting data augmentation, transfer learning and using better suited network architectures. They also address the influence of illumination and occlusion on the detection process. Finally, they concluded their review with the importance of balance between the accuracy and the model speed.

Barbedo [1] give a review of the main challenges facing plant disease identification, such as background complexity, variable capture conditions, wide symptoms range of plant diseases, shared symptoms across diseases and the possibility of several diseases being present simultaneously.

Deep residual networks were introduced in He et al. [8] and have since shown great success. In the context of plant disease classification, Fang et al. [6] make use of a ResNet50-based CNN approach to identify 10 types of disease images for 8 crops and achieve a result of 96.61%.

Data augmentation have been shown to counter over-fitting [14]. Examples of data augmentation methods include RandAugment, which essentially picks data augmentation transforms randomly from a set of transforms[5], and MixUp, which mixes two images when training with a random chosen ratio and take the same ratio to define the loss [16].

Transfer learning have also been shown to be effective [15], and Chen et al. [3] experiment with VGGNet pre-trained on ImageNet and Inception for plant disease classification, demonstrating the effectiveness of transfer-learning for our task.

For self-supervised learning, Newell and Deng [12] compare the usefulness of different self-supervised methods for pre-training for vision tasks. Performance of tasks differs for different settings. There are many self-supervised learning methods, for example predicting image rotations (angle) [7], predicting (generating) cropped parts of images known as Inpainting [13], or image colorization by reverting colored images to gray scale and predicting pixel colors[17].

Table 1. Baseline Accuracies

Model	Training Accuracy	Validation Accuracy	Test Accuracy
VGG16	98.15%	89.71%	91.15%
VGG19	98.89%	90.81%	91.15%
ResNet50	83.21%	58.82%	61.65%
ResNet101	90.31%	68.01%	65.5%
ResNet152	92.90%	63.97%	63.72%
Developed CNN	90.77%	87.87%	88.5%

3 Methodology

This work aims to solve the sesame disease classification problem by using one of the CNN Architecture models. ResNet50 architecture (see Fig. 4) was chosen to perform the classification, using pre-trained weights from ImageNet provided

by Pytorch public library. Combinations of different techniques (Data Augmentation, Transfer Learning, and Self-Supervised Learning) were applied to reduce over-fitting and increase the validation and test accuracy of the model. The model is trained with stochastic gradient descent using SDG optimizer, and a cosine annealing scheduler (without warm restarts) is used to adjust the learning rate [10].

For Evaluation, cross-entropy loss and the accuracy is used (number of correctly predicted examples divided by the number of examples). The data was split into 6 equal parts (folds), this is done as as a pre-processing step hence the split is the same for all experiments. 1 part is left for doing final tests. Training with k-fold cross-validation is done. Since over-fitting is a major concern we want a more accurate measure of our accuracy. The model is trained independently from the start using 4 parts and 1 part acts as a validation data set. The validation accuracies were used in the process of comparison between different methods to improve the accuracy and tuning of our hyper-parameters, and the test data set was not be touched until the very end so as not to affect our choice of hyper-parameters. For the test, we save the model with the best validation accuracy from each fold and apply it to the test data, then averaging the results. Splitting the data into folds allows us to get a more reasonable estimate for the accuracy to tune our hyper-parameters.

The results were collected on different stages. First, the model was trained using data augmentation only. On the second stage the self-supervised was added to data augmentation. The third and fourth stages applied data augmentation and transfer learning from Cassava dataset techniques with 20 and 50 epochs respectively. The fifth stage performed the three methods stated earlier to see the effect of the techniques combined together (see Fig 1). When switching between pre-training tasks and/or the main classification task, the last Softmax layer is replaced with a different one that outputs the desired number of classes.

3.1 Dataset

The sesame dataset used to train the model was collected manually from the fields using a mobile phone camera [2], with 3 classes: Healthy, Bacterial Leaf disease, Gall disease (see Fig. 2). The dataset has a total of 1453 images. Table 2 shows the number of images per class in the dataset. The sesame dataset required some filtering, some images were duplicated, and we filtered through the dataset to prepare it for the training.

Table 2. Sesame disease dataset classes

Class Name	Number Of Images Per Class
Healthy Class	77
Bacterial_Leaf_Spots Class	860
Gall_Midge Class	516

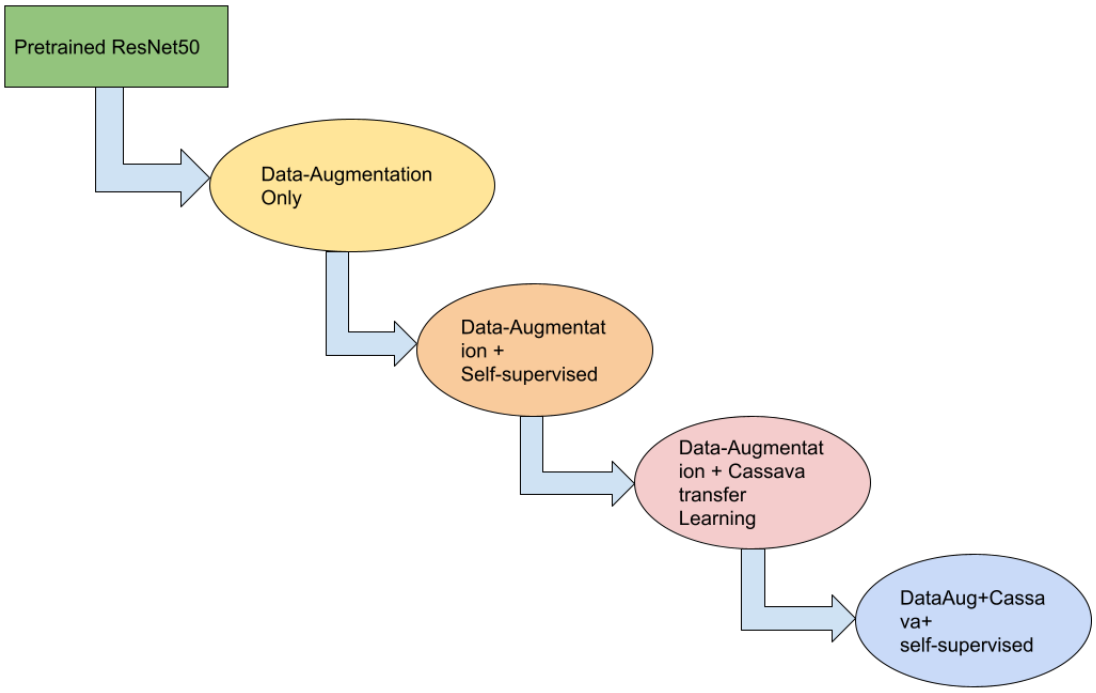

Fig. 1. Methodology Flow chart

The Cassava Leaf Challenge dataset used for transfer learning is a mostly crowd sourced dataset collected in a survey Uganda from farmers taking images of their gardens, with a total of 5 categories (see Fig. 3) and 9,436 labeled images, and annotated by a collaboration of experts at the National Crops Resources Research Institute (NaCRRI) and the AI lab in Makarere University [11].

Finally, for the self-supervised task, the Sesame Plant Disease dataset was aligned to be facing upwards as 0 degree.

3.2 Techniques used to improve the baseline performance

Transfer-Learning Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task. It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given the vast compute and time resources required to develop neural network models on these problems and from the huge jumps in skill that they provide

Fig. 2. Healthy (Right), Bacterial Leaf Spot (Center), Gall (Left)

Fig. 3. Cassava Dataset classes

on related problems[15]. In this paper the ResNet50 model (See Fig. 4) weights used were pre-trained on ImageNet as the initial point.

An extra pre-training (transfer learning) phase is added at the beginning, where we pretrain our model using the Cassava Leaf Disease Challenge dataset. We hypothesize that by pre-training the model on a dataset that shares some similarity with our dataset (both are leaves datasets) we will obtain better models, countering over-fitting and the small dataset size.

Data Augmentation Data augmentation is a strategy that enables practitioners to significantly increase the diversity of data available for training models, without actually collecting new data. There are several transforms used to augment the data [14]. In this work random horizontal, ImageNet policy, RandAugment and Mix-up are the techniques used for the data augmentation process.

RandAugment is a class which creates a pool of policy combinations from a set of defined transforms with a random strength from 1 to 10. The RandAugment object then selects randomly two of the polices . The two selected transforms were applied sequentially on the images to generate more data (see Fig 5 and Fig 6). [5]

ImageNetPolicy function selects and applies randomly one of the best transformations that presented the best performance when applied on ImageNet dataset. [5]

Mixup mixes every training image with a random image as a weighted sum with proportions chosen randomly from a beta distribution. The loss is then calculated as a weighted sum between the loss for the two labels. This typically

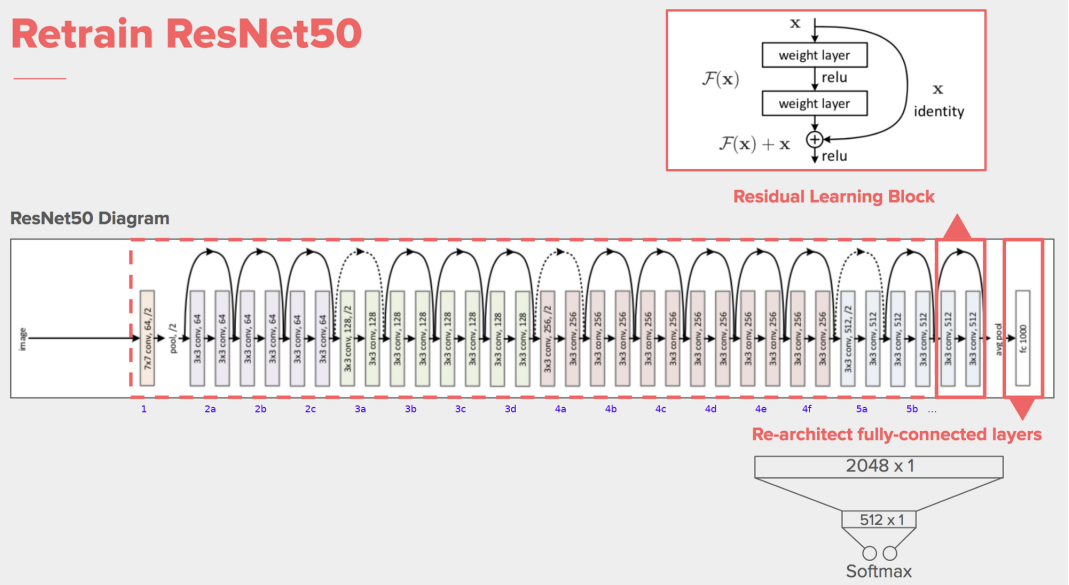

Fig. 4. ResNet50 Structure[4]

needs the predictions to be in one-hot encoding, but as we are using Cross-Entropy loss, we can use it as a weighted sum of the Cross-Entropy loss of the two labels. [16]

```
transforms = [
'Identity', 'AutoContrast', 'Equalize',
'Rotate', 'Solarize', 'Color', 'Posterize',
'Contrast', 'Brightness', 'Sharpness',
'ShearX', 'ShearY', 'TranslateX', 'TranslateY']

def randaugment(N, M):
"""Generate a set of distortions.

  Args:
    N: Number of augmentation transformations to
        apply sequentially.
    M: Magnitude for all the transformations.
"""

  sampled_ops = np.random.choice(transforms, N)
  return [(op, M) for op in sampled_ops]
```

Fig. 5. Python code for RandAugment based on numpy

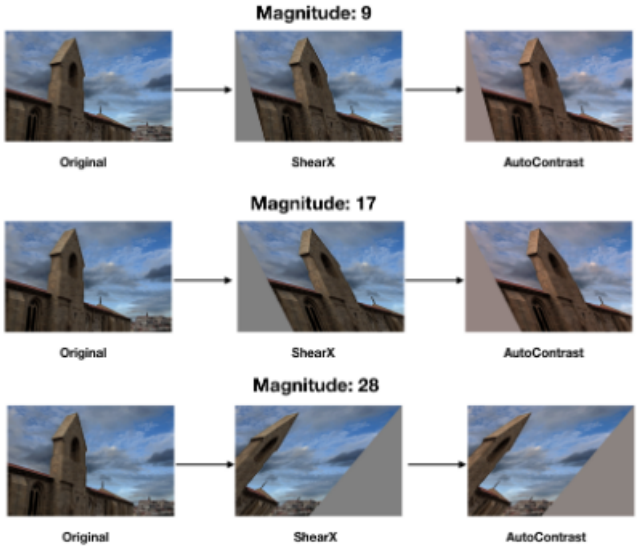

Fig. 6. Example images augmented by RandAugment

Self-Supervised Learning Self supervised learning creates an artificial task and trains the model on self generated example/label pairs[12]. The self-supervised pre-training phase is added after the transfer learning phase. We hypothesize that by pre-training the model on an artificial self-supervised task the model can learn useful features about the structure of the images and hence increase the accuracy.

We specifically use the rotation (angle) classification task[7]. Images are rotated randomly and the model is trained to predict the orientation of the rotated image. The task implement a function which rotate the images with a random angle chosen from four different angles [0,90,180,270] and generate image/label pairs from the aligned dataset. (see Fig. 7)

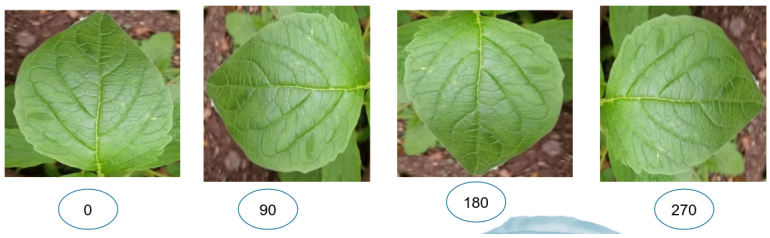

Fig. 7. self-supervised rotation task

4 Experimental results and Analysis

We test our model by training the full ResNet50 model for 100 epochs, or freezing the model and training only the last layer for 20 epochs. Different combinations for the learning rate and exponential (with step size=7, gamma=0.1) or cosine annealing scheduler (with $T_{max} = 20$) implemented in Pytorch were trie; the best results were achieved using cosine annealing scheduler (without warm restarts) and learning rate = 0.01 (see Table 3 and Fig. 8).

The model is trained with SDG optimizer with momentum=0.9. Table 4 gives a summary of the average results (see Fig. 9 and 10). Figures 11 through 15 show the progress of the accuracy through time for different folds. We can see the accuracy fluctuates, this is because we use a cosine scheduler.

Table 3. Different Learning Rates for Data Augmentation Only

Model	cos[1]lr[2]= 1	cos lr = 0.01 [3]	cos lr = 0.001	exp[4] scheduler lr=0.001
Full Model val	41.62	96.03	94.43	94.62
Only Last Layer Trained val	81.61	79.57	85.68	83.67

[1] Cosine scheduler
[2] Learning rate
[3] Note that here the last layer was trained only for 20 epochs due to time constraints, while for other learning rates it was trained for 100 epoch.
[4] Exponential scheduler

Table 4. Average Results

Model	DA [1]only	DA + SS [2]	DA + CA20 [3]	DA + CA50 [4]	DA + CA20 + SS
Full Model val	96.03	95.35	96.93	96.15	94.56
Full Model test	96.59	95.67	96.33	96.71	95.67
Only Last Layer Trained val	79.57	75.6	85.68	83.0	74.74
Only Last Layer Trained test	87.13	82.07	90.90	88.65	84.88

[1] Data Augmentation
[2] Pre-training (Self-supervised learning) with Rotate
[3] Pre-training (Transfer Learning) with Cassava Leaf Disease Challenge dataset for 20 epoch
[4] Pre-training (Transfer Learning) with Cassava Leaf Disease Challenge dataset for 50 epoch

We can see that when training the full model, just by using pre-trained weights from ImageNet, adding data augmentation and selecting good hyper-parameters we get a result of **96.59%**, vastly outperforming the best reported

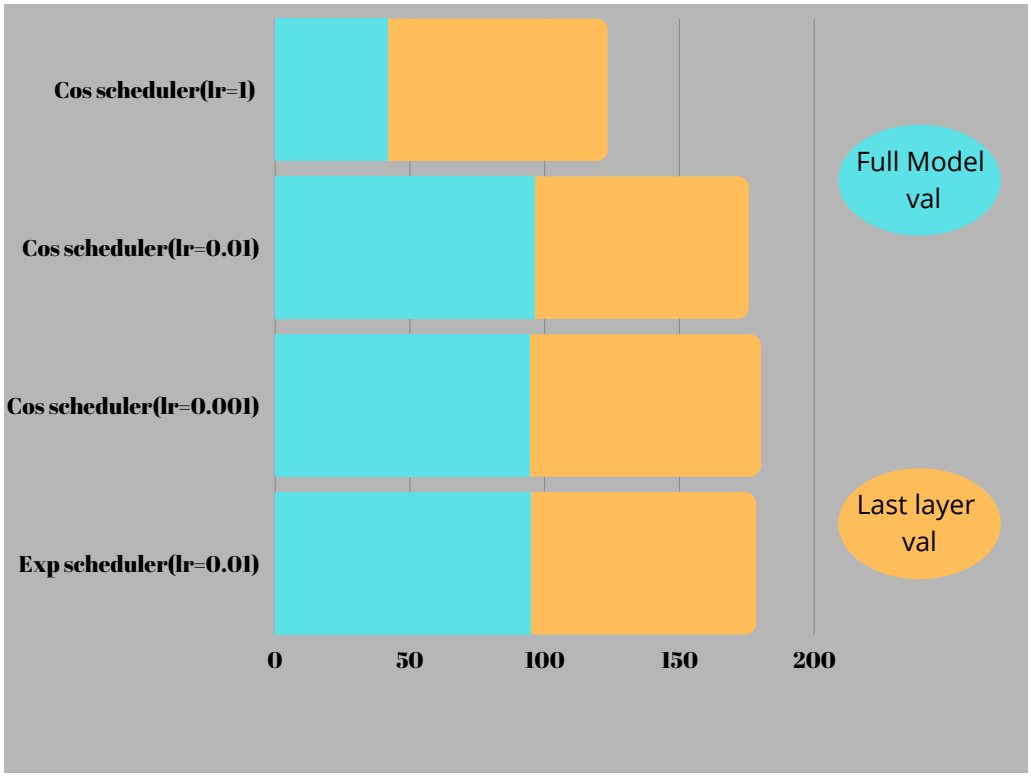

Fig. 8. Different Learning Rates for Data Augmentation Only (table 3)

accuracy of our baseline of 91.15% using VGG16 and VGG19, and their reported accuracy when using ResNet50 of 61.65%. This can be explained by the fact that they do not use pre-trained weights, and as the size of the dataset is small; the model does not generalize well to unseen data.

Our Data Augmentation combination of RandAugment, ImageNet Policy, and Mixup have shown a strong effect in countering over-fitting and increasing the accuracy of our model. This is expected as these methods have shown significant success individually, but further experiments may be needed to study the interaction and individual influence of added components.

The best test accuracy achieved was 96.71% by using pre-trained weights from ImageNet and pre-training on the Cassava Leaf Disease Challenge dataset for 50 epochs. This demonstrates that when faced with small dataset size, it is helpful to use a multi-step transfer learning from a huge general dataset (ImageNet) followed by pre-training on a dataset that shares structure similarity. It also shows it reduces overfitting as it generalizes better, illustrated by better performance on the test dataset.

We note however that when pre-training for the Cassava Leaf Disease Challenge dataset for 20 epochs (instead of 50), the validation accuracy increased

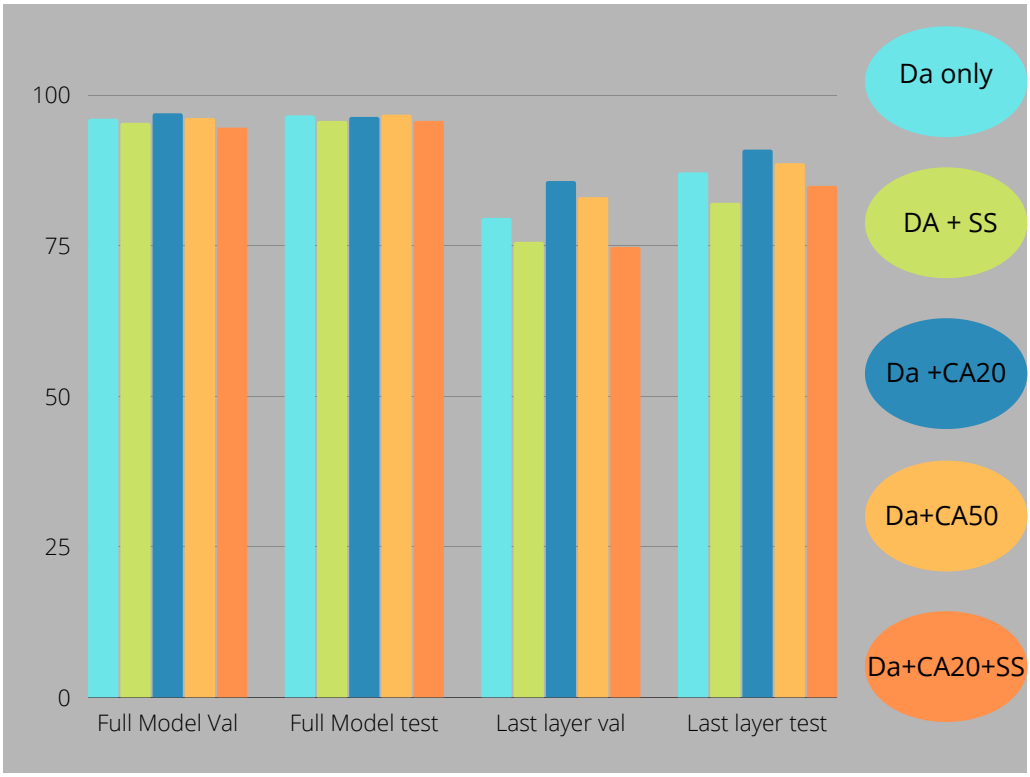

Fig. 9. Bar chart visualization for the average results (table 4)

but the test accuracy decreased compared to transfer learning from ImageNet only (Data Augmentation only). This may be attributed to overfitting, or losing the weights/features learned from ImageNet at first and not learning enough features from the Cassava dataset, but as we train on the Cassava further the model learns better features and increases accuracy.

Further, we can see that by training only the last layer, we were able to achieve an accuracy of 87.13% by using data augmentation and pre-trained weights from ImageNet. The highest accuracy achieved was **90.9%** when pre-training on the Cassava Leaf Disease Challenge dataset for 20 epochs, followed by 88.65% when pre-training for 20 epochs. This decrease in accuracy when pre-training for more epochs is in contrast to training the full model, and may be explained that the model loses some generality (fine tuned to the Cassava Leaf Disease Challenge dataset) as we pre-train for more epochs, requiring more layers to be trained to adapt to the specifics of the Sesame Plant Disease dataset.

The results from training the last layer only have shown that a reasonable performance can be obtained by transfer learning and freezing the network and training a single layer, especially when pre-training the network with a similar dataset (Cassava Leaf Challenge). This is significant because the training time

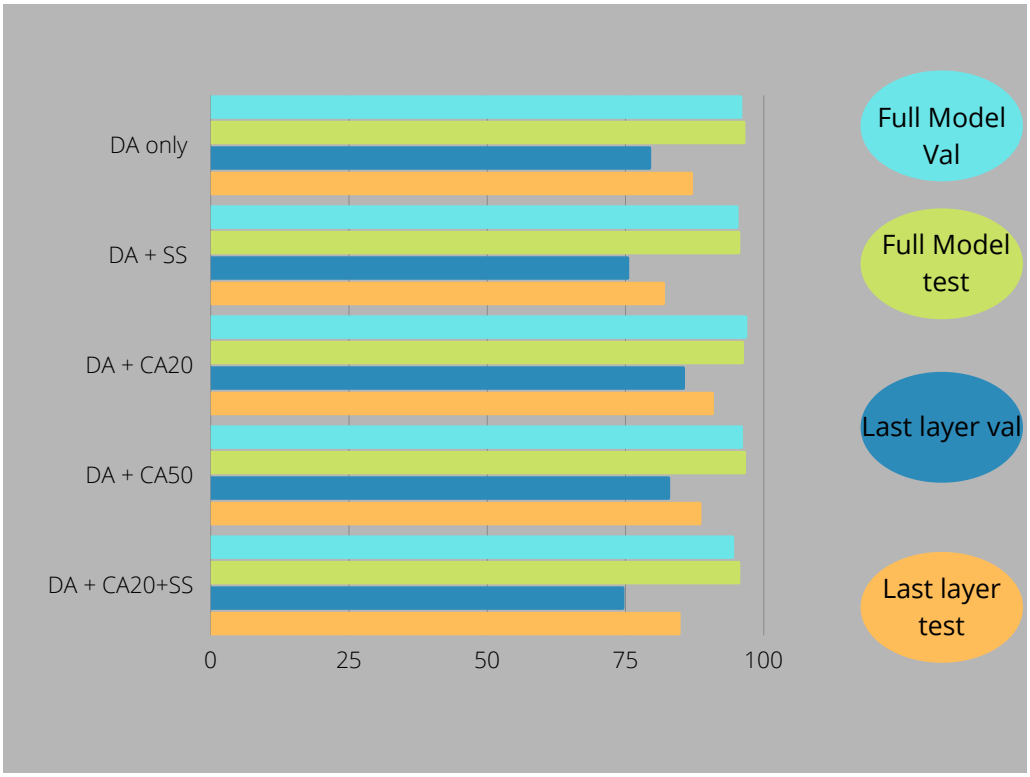

Fig. 10. Bar chart visualization for the average results (table 4)

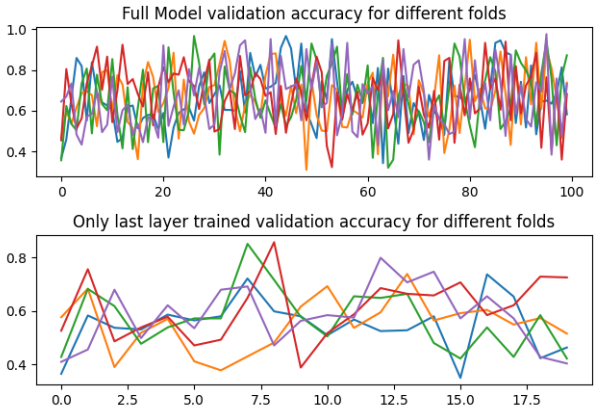

Fig. 11. Data Augmentation

249 and resources required for training only the last layer are much less compared 249

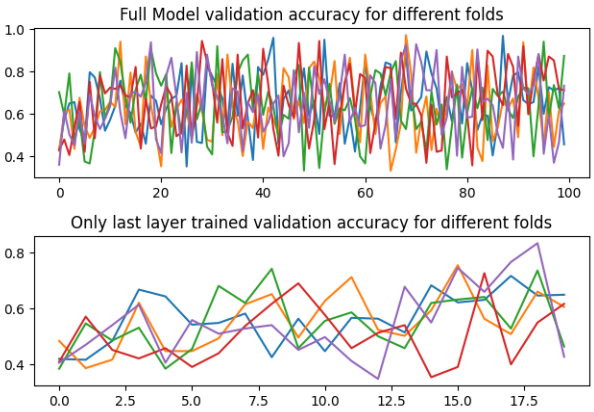

Fig. 12. Data Augmentation + Self-supervised

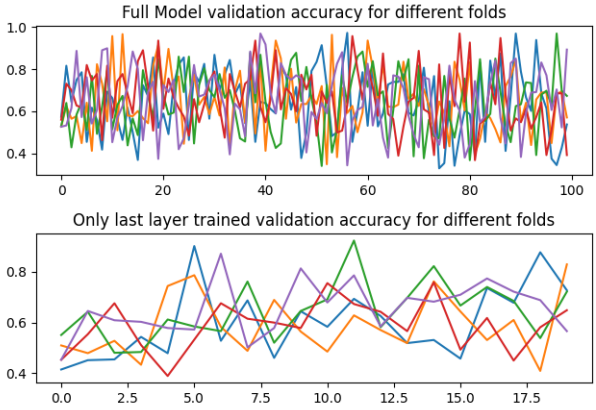

Fig. 13. Data Augmentation + Cassava transfer learning (20 epochs)

to training the whole model. This suggests that the model can be robust if pre-trained on a similar dataset, and only the last layer(s) may need to be trained for a reasonable performance.

Pre-training with self-supervised learning with rotation (angle) classification task has shown a consistent decrease in performance, whether adding it alone or after pre-training on the Cassava Leaf Challenge dataset. This may be explained by several possible factors: (1) recognizing the angle of the image may not be a good self-supervised pre-training task for our dataset, and another task such as Inpainting and colorization may be more relevant, (2) inappropriate choice of hyper-parameters or number of epochs, (3) As the weights are adjusted for the artificial task of self-supervised learning (rotation prediction task); **the model may be losing the advantage of transfer learning from ImageNet (the**

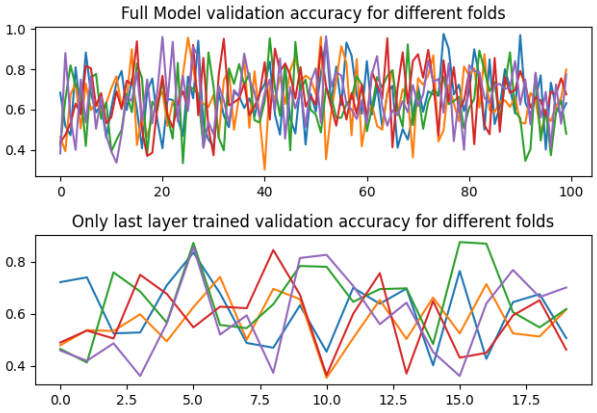

Fig. 14. Data Augmentation + Cassava transfer learning (50 epochs)

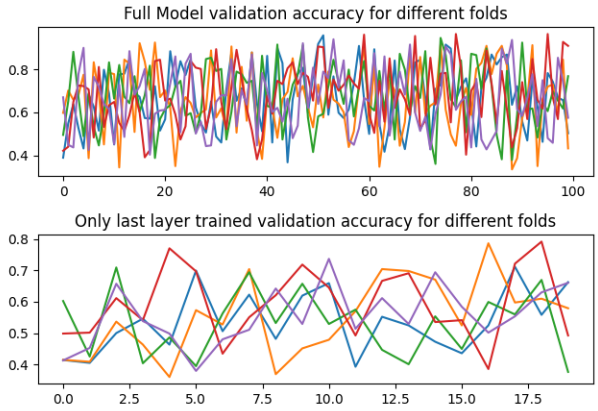

Fig. 15. Data Augmentation + Cassava transfer learning (20 epochs) + Self-supervised

learned features are changed, and the new features are not as useful); hence the benefit brought about by self-supervised learning may be less than the lost advantage, causing the overall accuracy to drop.

5 Conclusions

5.1 Work Summary

In this project, we have applied transfer learning and data augmentation to successfully increase the test accuracy with **5.56% from 91.15 to 96.71%** compared to the best results reported in the baseline paper, and achieving vast improvement of about **35% over their use of ResNet50** with a reported test accuracy of **61.65%**.[2]

We demonstrate that a multi-step pipeline making use of transfer learning using pre-trained weights from ImageNet, followed by a second transfer learning phase where we pre-train the model on a dataset that shares some similarity is beneficial in overcoming small dataset size and over-fitting. We do this by pre-training on the Cassava Leaf Challenge dataset, increasing the accuracy by **0.12% from 96.59% to 96.71%**.

We show that the techniques applied can also benefit the model when frozen and only the last layer is trained, achieving an accuracy reaching up to **90.9%** when using transfer learning from ImageNet and pre-training on the Cassava Leaf Challenge dataset for 20 epochs. This shows that we can achieve good performance with limited resources. We were superised to find that the accuracy for training only the last layer decreased when pre-training on the Cassava Leaf Challenge dataset for more epochs, and explain this by suggesting that the model may have lost some generality (fine-tuned on the Cassava dataset) and may need more layers to be trained to adapt to the specifics of the Sesame Plant Disease dataset.

We experiment with adding a self-supervised learning phase after transfer learning by pre-training on an artificial angle classification task, but were surprised to find it decreases the accuracy. We explain this by the model losing the features we have from transfer learning as the weights are trained for the self-supervised task, and that the new features are not as useful as the ones from transfer learning.

5.2 Future work

Some avenues for future experimentation that we have identified, with the goal of improving the results of the methods explored in this work:

- Other Self-supervised tasks could be applied on the sesame dataset such as (Inpainting, colorization, etc).
- Apply Self-supervised task on cassava dataset for the model to learn more about the similar datasets.
- Testing the robustness of the model across different plant disease classification problems.
- Apply different data-augmentation techniques.
- Use GANs to Generate sesame disease images resembles the real ones.
- Use different architectures of the CNN models.
- Further experiments to tune the hyper-parameters as they presented a huge impact on the performance of the model.
- Make use of all data available, by training on all folds (i.e. train on 5 folds instead of leaving one out for validation) before testing on the test fold.

Appendix

6 Task Assignment

- **Ahmed Musa:**
  - Project proposal discussion and selection.
  - Interim report preparation.
  - Wrote functions to print the output figures and save them.
  - Coded the data separation to folds for the Sesame Plant dataset
  - Coded the Cassava Leaf Disease Challenge data preparation and pre-training (making use of Ibrahim's code).
  - Coded MixUp data augmentation.
  - Submitted jobs to the GPU.
  - Preparing result tables.
  - Responsible for Related Work, Experimental Results and Analysis and Work Summary, Contributed to Introduction, Methodology and Future Work.
- **Ibrahim Siddig:**
  - Project proposal discussion and selection
  - Interim report preparation
  - Coded the main ResNet50 base model code.
  - Wrote the RandAugment and ImageNetPolicy data augmentation code
  - Coded the accuracy and test model code.
  - Prepared the self-supervised dataset.
  - Coded the self-supervised pre-training task (data/label generation with random rotations, and training).
  - Responsible for Introduction, Methodology and Future Work, Contributed to Related Work, Experimental Results and Analysis and Work Summary.
- **Sujatha Senthurnathan:**
  - Project proposal discussion and selection.
  - Interim report preparation.
  - Prepared custom docker image.
  - Submitted jobs to the GPU.
  - Preparing result tables.
  - Baseline results table.

Bibliography

[1] Jayme Garcia Arnal Barbedo. A review on the main challenges in automatic plant disease identification based on visible range images. *Biosystems engineering*, 144:52–60, 2016.

[2] Israa Hassan Bashier, Mayada Mosa, and Sharief Fadul Babikir. Sesame seed disease detection using image classification. In *2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*, pages 1–5, 2021. https://doi.org/10.1109/ICCCEEE49695.2021.9429640.

[3] Junde Chen, Jinxiu Chen, Defu Zhang, Yuandong Sun, and Yaser Ahangari Nanehkaran. Using deep transfer learning for image-based plant disease identification. *Computers and Electronics in Agriculture*, 173:105393, 2020.

[4] Adria Ciurana. Resnet50 illustration. URL https://stackoverflow.com/questions/54207410/how-to-split-resnet50-model-from-top-as-well-as-from-bottom.

[5] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020.

[6] Tao Fang, Peng Chen, Jun Zhang, and Bing Wang. Crop leaf disease grade identification based on an improved convolutional neural network. *Journal of Electronic Imaging*, 29(1):013004, 2020.

[7] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[9] Jun Liu and Xuewei Wang. Plant diseases and pests detection based on deep learning: a review. *Plant Methods*, 17(1):1–18, 2021.

[10] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[11] Ernest Mwebaze, Timnit Gebru, Andrea Frome, Solomon Nsumba, and Jeremy Tusubira. icassava 2019 fine-grained visual categorization challenge. *arXiv preprint arXiv:1908.02900*, 2019.

[12] Alejandro Newell and Jia Deng. How useful is self-supervised pretraining for visual tasks? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7345–7354, 2020.

[13] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[14] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, 2019.

[15] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *International conference on artificial neural networks*, pages 270–279. Springer, 2018.

[16] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[17] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016.