ISTANBUL TECHNICAL UNIVERSITY

Istanbul Technical University
Faculty of Computer and Informatics

Exploring Supervised and Self-Supervised Methods on Cassava Leaf Detection Challenge

Deep Learning Interim Project Report

Authors
Batuhan Faik Derinbay - 150180705 - derinbay18@itu.edu.tr
Ufuk Demir - 150170710 - demiruf17@itu.edu.tr

July 24, 2021

Contents

Introduction

1.1 Abstract

Cassava is a significant source of dietary energy for more than 500 million people and is the most important tropical root crop. Nevertheless, like many others, cassava plants are vulnerable to diseases. In this report, state of the art deep learning techniques are explored and evaluated on the problem of classifying cassava leaf diseases in order to get an understanding of performance metrics of the latest widely-used models.

1.2 Background

Cassava, Manihot esculenta Crantz in Latin, ranks fourth as a food crop in the developing countries according to the United Nations Food and Agriculture Organization because of its importance in the food chain. In addition, cassava is the highest producer of carbohydrates among staple crops; its leaves are protein-rich, and they can be stored in the ground for several seasons. Moreover, with the upcoming improvements to the technology, it is being used in the manufacturing, industrial processes, and as an animal feed. With all these great qualities, cassava has a critical role for humankind and has to be produced in ample amounts.

People mass-producing cassava often face a variety of challenges. The most important of which and the main problem that this project aims to tackle is the widely known types of leaf diseases. The four types of defects within the interest of this report are Cassava Bacterial Blight (CBB), Cassava Brown Streak Disease (CBSD), Cassava Green Mottle (CGM), and Cassava Mosaic Disease (CMD).

These defects can be recognized with a trained human eye but require many scientists and knowledgeable farmers to detect on the crops in a field. As technology improves, new methods are being tested to detect diseases of cassava leaves. From unmanned air vehicles (UAV) to simple smartphones, farmers take pictures of crops within the field, share them with experts, and look for defects, deterioration on the leaves. As might be expected, this process takes incredibly long and requires many human resources. In order to shorten this time and lessen the workload, Marker University Artificial Intelligence Laboratory has hosted a classification challenge on Kaggle.

In this currently active challenge (kaggle.com/c/cassava-leaf-disease-classification), competitors are asked to distinguish cassava leaves affected with four types -as mentioned earlier- of diseases and healthy ones. Since the challenge is intriguing, provides a dataset

to work with, and is currently being tackled with heavy augmentations to the data with little to no improvements to the novel models, the team composed of Batuhan Faik Derinbay and Ufuk Demir has decided to use the provided dataset in their interim project and implement state of the art innovations in order to increase the accuracies achieved by existing models.

Methodology

2.1 Data Preparation

Dataset used within the scope of the project is provided by the Kaggle challenge and can be found on the website. It features 21397 labeled images with a total size of 2.6 gigabytes and a resolution of 800×600 pixels. The number of images per class and their relative percentages to the dataset are given in Table 2.1. By examining the table, it is clear that there is a data imbalance between classes. To overcome this problem, the team tried to implement "Self-supervised Label Augmentation via Input Transformations" paper [1] in section 2.3.

Moreover, since the given dataset consists only of training samples of the challenge, the team decided to do first a train-test split with $90 - 10$ percent respectively, second, a train-validation 5-fold split on the resulting training data. In the end, the team ended up with 2140 never seen test data, 3852 validation data per fold, and 15404 training data per fold. These numbers represent 10%, 18%, and 72% of the total dataset, respectively. Due to the 5-fold split, each implemented method and combinations of methods resulted in five different models that are trained on the folds and never seen the 10% test data.

	CBB	CBSD	CGM	CMD	Healthy	Total
# of Images	1087	2189	2386	13158	2577	21397
Percentage	5.08	10.23	11.15	61.49	12.04	100

Table 2.1: Distribution of the Data

2.2 Data Augmentation, Model Selection and Hyperparameter Tuning

In order to set a baseline for the project, the team fed all the models with transformed images. Considering the computational limitations (E.g., number of available GPUs, VRAM sizes), they decided to rescale the images to 300×225 pixels, preserving the aspect ratio, then cropping the center 224×244 pixels region. After obtaining baseline

results from various models, they further investigated with different data augmentation methods explained in section 2.3 and also increased the input size to 800×600 pixels.

When it comes to model selection, the team decided to use EfficientNet-B0 because it sits at the perfect spot within image classification models where high accuracy meets a small model size. High accuracy and a minimal number of parameters were critical when selecting the model because due to the limited time of the project submission and the large number of models that need to be trained, a suitable model has to be able to train fast and achieve relatively high accuracies. To illustrate why EfficientNet-B0 is the right choice, the team also obtained baseline results using a four-fold larger state of the art deep learning model, ResNet34, with the same hyperparameters and got 4.6% lower validation accuracy on average. Hence, EfficientNet-B0 is chosen to be the only model that novelties of the project should be implemented.

Training runs of all the models used randomly initialized weights with the same seed and the stochastic gradient descent optimizer (momentum $\upsilon = 0.9$) with cross-entropy loss. The initial runs with the learning rate $\alpha = 0.00256$ (halved every 20 epoch) caused the model to converge very poorly and often get stuck within a local minimum. After such attempts with meager accuracy results, the team decided to start with a learning rate of $\alpha = 0.1$ and reduce it by a factor of 10 for every 5 epoch that did not decrease the loss. Satisfied with the results, hyperparameters are then kept constant between models in order to get more comparable outputs from the models. With the constant hyperparameters, loss curves given in Figure A.1 are obtained. Note that only loss curves of the models that are given in Table 3.1 are present in Figure A.1.

2.3 Self-supervised Label Augmentation

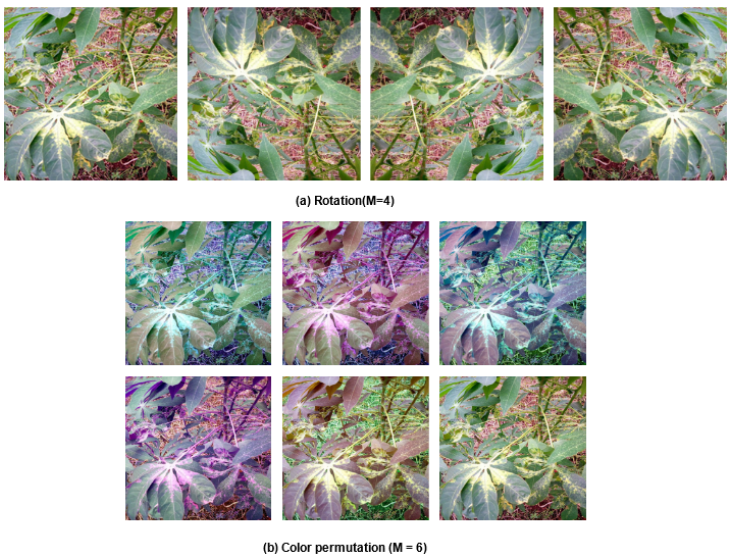


Figure 2.1: Rotation-based augmentation in (a) Color-permutation-based augmentation in (b)

Imbalance in the dataset will adversely affect model performance. To prevent that, self-supervised label augmentation methods, which are rotation and color permutation-based, were applied during the training. As stated in the paper, the critical point is learning

a single unified task concerning the original and self-supervised labels' joint distribution. In order to achieve the learning of a unified task, the original labels are augmented via self-supervision of input transformation.

In the prior works [2], [3], generally, two classifiers are trained separately for the original and self-supervised task, then their objectives are optimized simultaneously. However, recent studies show that the multi-task learning approach does not work well with fully-labeled datasets and does not increase the model accuracy.

In the paper, the authors realized that the multi-tasking learning approach forced the model for the primary classification task to be invariant for transformations of a self-supervised task. Instead of learning two separate tasks, learning a single unified task concerning the joint distribution of the augmented labels is used to handle this.

In this project, rotation-based (E.g. Figure 2.1) and color permutation-based augmentation (E.g. Figure 2.1) are used. When training on the model with the self-supervision on rotation (4 labels), the model learns the joint probability distribution on all possible combinations of 20 labels. When training with the color permutation-based augmentation, the model learns the joint probability distribution on all possible combinations of 30 labels.

2.4 Sharpness-Aware Minimization

Sharpness-aware minimization (SAM) [4] is a state of the art procedure that minimizes the loss value and the loss sharpness simultaneously. It has recently been proposed by the Google research team and is under the review of the International Conference on Learning Representations (ICLR 2021). In order to turn the cost into a min-max optimization problem for gradient-descent to carry out effectively, SAM searches for parameters that lie in areas having a uniformly low loss. By optimizing the way loss calculations are carried out, SAM achieves state of the art results on various datasets. Further details and mathematical derivations are given in the original SAM paper "Sharpness-Aware Minimization for Efficiently Improving Generalization", please refer if necessary.

After examining loss curves of the models (E.g., Figure 2.2), the team decided to implement SAM within their training procedure in order to deal with abnormalities seen with the loss. Models that use sharpness-aware minimization with stochastic gradient descent had error rates about 5% less than the baseline models, which are explained in more details in chapter 3.

The increase in the accuracy of the models is due to the fact that SAM helps generalize the data better by minimizing both the loss value and the sharpness. However, it comes with a cost. SAM requires gradient calculations of the loss twice. Calculating gradients of the loss two times increases the training time and the need for higher computational power. Moreover, SAM comes with a single hyperparameter ρ, which is tuned via grid search and is found to be $\rho = 0.05$ as an optimal value in the original paper. In this project, the team did not tune this hyperparameter because of the satisfactory results obtained when using the optimal value and the project's time limitations.

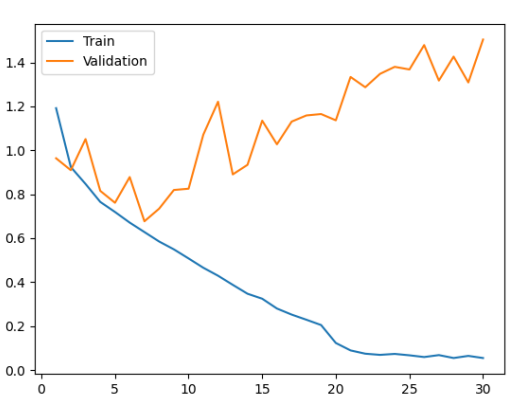
Within this research project's scope, implementation of the SAM is adopted from a GitHub repository [5], which in turn is also adopted from the GitHub repository of the

paper [6]. The code below is from the project sources and depicts the two step loss gradient calculation in order to calculate sharpness-aware minimization presented in the paper.

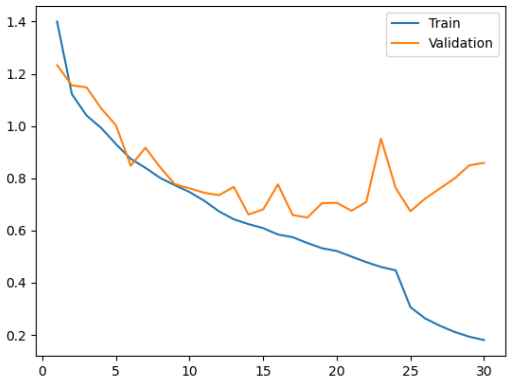
```
# First pass
loss_value = loss(model(img), img_class)
loss_value.backward()
optimizer.first_step(zero_grad=True)

# Second pass
loss(model(img), img_class).backward()
optimizer.second_step(zero_grad=True)
```

Two Step Loss Gradient Calculation



(a) EfficientNet-B0 with SGD Optimizer



(b) EfficientNet-B0 with SGD Optimizer and SAM

Figure 2.2: Loss Curves of Identical Models With and Without SAM
(Axis y : Loss, x : # of epochs)

2.5 SpinalNet Fully-connected Layers

SpinalNet [7] is a revolutionary deep neural network architecture that mimics the human somatosensory system's anatomy and achieves state of the art accuracy on various datasets with less computational resources. It offers gradual inputs, local outputs with probabilities of global influence, and weights that reconfigure during training. Moreover, the authors of the paper address the effectiveness of the Spinal Convolutional Layers as well as Spinal Fully-connected Layers and get baseline results on Wide ResNet101 and VGG19 models.

Even though the authors of the paper obtain worse results from Wide ResNet101 due to the decrease in gradients at the additional layers, since a decrease in the error and computational resources is in favor of the research team, they also decide to replace fully-connected (FC) layers of their models with SpinalNet FCs. Knowing that EfficientNet-B0 is shallower than Wide ResNet101 (Gradual gradient decrease will not affect as much)

and the added Spinal FCs decrease the number of parameters, the team expected and achieved better results than the baseline model.

The structure of spinal hidden layers versus traditional hidden layers is given in Figure 2.3 (Obtained directly from the paper). By configuring four spinal hidden layers and replacing the output of the base model with spinal layers, the team achieved a reduction in error of around 2.5%, which is explained with more details in chapter 3. Though some might argue that increasing the number of layers in the fully-connected output of the model increases the performance, it should be noted that the number of parameters is less in the model with a Spinal FC, which drastically improves training time. To be precise, the baseline model has 5.2 million parameters, whereas the model with a Spinal FC output layer has only 4 million parameters. The difference between models is a massive 23%, considering how shallow the EfficientNet-B0 is. It should also be noted that the number of Spinal fully-connected layers and their width are hyperparameters that can be tuned. In all of the experiments, both parameters are fixed to 5 fully-connected layers and width of 20 neurons to get more comparable results between models.

Furthermore, the original implementation of "SpinalNet: Deep Neural Network with Gradual Input" can be found in the paper's GitHub repository [8]. The Spinal FCs used within this project are adopted from the original repository, implemented and benchmarked on the EfficientNet architecture for the first time as far as the team knows.

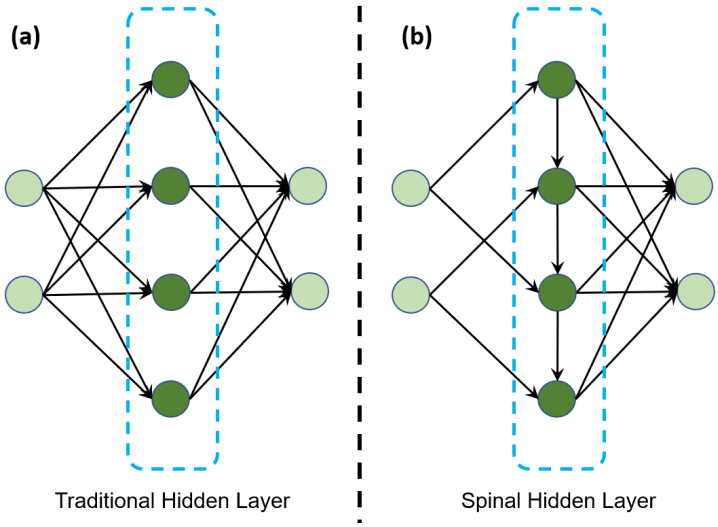


Figure 2.3: The Traditional Hidden Layer in (a) is Converted to a Spinal Hidden Layer in (b)

Evaluation and Results

The team tried to find a solution to the classification task by applying the methods described in part 2 on the Cassava dataset. The results were obtained by using the aforementioned methods both separately and combined with each other. The dataset used in the project is taken from an ongoing Kaggle competition. In this competition, since there is one image in the public test set, train, validation, and test set were obtained by using the methods described in section 2.1. In all experiments, the augmentations described in section 2.2 were applied. Accuracy is used as the evaluation metric for this project. Non pretrained EfficientNet-B0 used as base model. Each model trained up to 100 epochs.

When the FC layer of the base model was replaced with the Spinal FC layer and the self-supervised label augmentation was not applied, the performance on the validation set slightly increased, while the success on the test set increased significantly. When the spinal layer method is used together with rotation-based and color permutation-based label augmentation, there is a serious decrease in model performance. It is observed that self-supervised label augmentations reduce the performance as they make optimization difficult. When Spinal FC was used with SAM, the performance was slightly improved compared to Spinal only. Also, it has been observed that the loss is slightly minimized.

From Table 3.1, it can be concluded that there is a significant increase in accuracy when the baseline model is used with SAM. Among the suggested methods, the highest accuracy was achieved using only SAM. As suggested in the paper, minimizing loss value and sharpness has enabled the data to be better generalized. Although SAM required high computational power, it made an important contribution to the classification task. When SAM is used with rotation-based and color permutation-based label augmentation, there is a significant decrease in model performance. It is observed that self-supervised label augmentations reduce the performance as they make optimization difficult. Moreover, rotation and color permutation-based label augmentations are not suitable for this dataset. The distribution of the color intensity on images in the dataset are not various. Hence offered color-based label augmentation did not work properly.

Model Name	Validation Results	Test Result
Baseline EfficientNet-B0	**75.51 ± 1.05**	63.33
Baseline ResNet34	72.05 ± 2.19	**76.03**
EffNet-B0 w/ Spinal FC	77.35 ± 1.81	78.82
EffNet-B0 w/ SAM	**79.32 ± 1.08**	**80.29**
EffNet-B0 w/ SAM + Spinal FC	77.95 ± 1.74	79.36
EffNet-B0 w/ Label Aug.	68.02 ± 1.10	71.26
EffNet-B0 w/ Label Aug. + Spinal FC	66.82 ± 0.41	68.56
EffNet-B0 w/ Label Aug. + SAM	67.51 ± 0.98	70.77
EffNet-B0 w/ Color Perm.	64.69 ± 1.02	68.99
EffNet-B0 w/ Color Perm. + Spinal FC	61.72 ± 0.42	60.56
EffNet-B0 w/ Color Perm. + SAM	64.87 ± 2.33	63.20
Baseline EfficientNet-B0 (800x600)	82.28 ± 0.64	**68.26**
EffNet-B0 w/ Spinal FC (800x600)	**82.35 ± 0.88**	65.26

Table 3.1: Experiment Results

Results represented in Table 3.1 are calculated using accuracy data given in Table 3.2. In order to achieve the same results, model checkpoints can be requested from the team. Folds and the test set are provided within the code submission and the dataset can be found on the Kaggle challenge. Furthermore, in order to visualize results of the models, t stochastic neighbor embedding of the outputs are obtained and represented in Figures A.2 through A.8.

Model Name	Validation Accuracies (Top 1)				
	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Baseline EfficientNet-B0	**75.13**	**75.72**	**76.01**	**76.74**	73.95
Baseline ResNet34	70.85	71.64	71.71	70.26	**75.83**
EffNet-B0 w/ Spinal FC	78.65	74.17	78.05	78.13	77.79
EffNet-B0 w/ SAM	**78.80**	**77.79**	**79.34**	80.23	**80.44**
EffNet-B0 w/ SAM + Spinal FC	78.20	76.54	78.54	**80.44**	76.07
EffNet-B0 w/ Label Aug.	67.33	67.39	67.64	69.97	67.81
EffNet-B0 w/ Label Aug. + Spinal FC	66.48	66.93	66.97	67.38	66.37
EffNet-B0 w/ Label Aug. + SAM	66.64	66.86	69.14	67.51	67.4
EffNet-B0 w/ Color Perm.	64.71	65.97	64.37	65.21	63.21
EffNet-B0 w/ Color Perm. + Spinal FC	61.79	61.12	61.66	62.29	61.77
EffNet-B0 w/ Color Perm. + SAM	61.01	64.81	66.00	67.18	65.37
Baseline EfficientNet-B0 (800x600)	81.96	**81.89**	**83.32**	82.48	**81.76**
EffNet-B0 w/ Spinal FC (800x600)	**81.99**	81.62	82.90	**83.63**	81.63

Table 3.2: Accuracies of the Models on All Folds

Related Work

Knowledge distillation technique can be used for the classification task. This technique works via transferring knowledge of a pre-trained larger network to a smaller network. In some papers, the networks call as a teacher and student networks. In [9] [10] they said that the knowledge distillation technique improves the student network's accuracy.

In [11], Berrada, Zisserman, and Kumar said that, smoothed cross-entropy loss is more robust to noise and overfitting than cross-entropy. Also, with the smoothed cross-entropy, a more robust t-SNE can be obtained.

Conclusion

Throughout this interim project, the team implemented four state of the art novelties described in sections 2.3, 2.4, 2.5 in order to boost the accuracy, decrease the size, and overall help the generalization of the models. The performance of the methods and their combinations used in the project are clearly evaluated, described, and compared. In light of the results obtained (see Table 3.1), the team has achieved higher accuracies than those of the base models'. With the novelties added, it can be observed that error rates decrease when training configurations using SAM and Spinal FC layers are applied. Moreover, the team realized that self-supervised label augmentation and color permutation methods are not suitable for use on this dataset.

To carry the project further, works mentioned in chapter 5, as well as model ensembling and hyper-parameter search can be applied.

Bibliography

[1] Hankook Lee, Sung Ju Hwang, and Jinwoo Shin. *Self-supervised Label Augmentation via Input Transformations*. 2019. eprint: arXiv:1910.05872 (page 2).

[2] Lucas Beyer, Xiaohua Zhai, Avital Oliver, and Alexander Kolesnikov. *S4L: Self-Supervised Semi-Supervised Learning*. 2019 (page 4).

[3] David Berthelot, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. *ReMixMatch: Semi-Supervised Learning with Distribution Alignment and Augmentation Anchoring*. 2019. arXiv: 1911.09785 [cs.LG] (page 4).

[4] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. *Sharpness-Aware Minimization for Efficiently Improving Generalization*. 2020. eprint: arXiv:2010.01412 (page 4).

[5] David Samuel. *SAM Optimizer*. https://github.com/davda54/sam. 2020 (page 4).

[6] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. *SAM: Sharpness-Aware Minimization for Efficiently Improving Generalization*. https://github.com/google-research/sam. 2020 (page 5).

[7] H M Dipu Kabir, Moloud Abdar, Seyed Mohammad Jafar Jalali, Abbas Khosravi, Amir F Atiya, Saeid Nahavandi, and Dipti Srinivasan. *SpinalNet: Deep Neural Network with Gradual Input*. 2020. eprint: arXiv:2007.03347 (page 5).

[8] H M Dipu Kabir, Moloud Abdar, Seyed Mohammad Jafar Jalali, Abbas Khosravi, Amir F Atiya, Saeid Nahavandi, and Dipti Srinivasan. *SpinalNet: Deep Neural Network with Gradual Input*. https://github.com/dipuk0506/SpinalNet. 2020 (page 6).

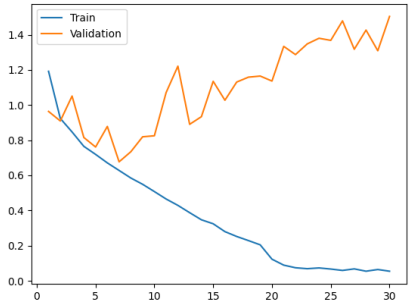
[9] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. *Distilling the Knowledge in a Neural Network*. 2015. arXiv: 1503.02531 [stat.ML] (page 9).

[10] Xu Lan, Xiatian Zhu, and Shaogang Gong. *Knowledge Distillation by On-the-Fly Native Ensemble*. 2018. arXiv: 1806.04606 [cs.CV] (page 9).

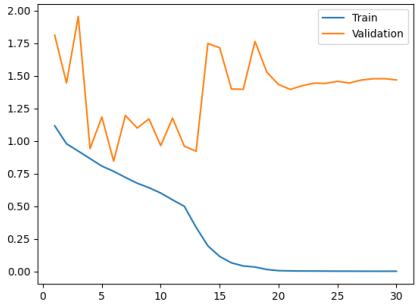
[11] Leonard Berrada, Andrew Zisserman, and M. Pawan Kumar. *Smooth Loss Functions for Deep Top-k Classification*. 2018. arXiv: 1802.07595 [cs.LG] (page 9).

Appendix A

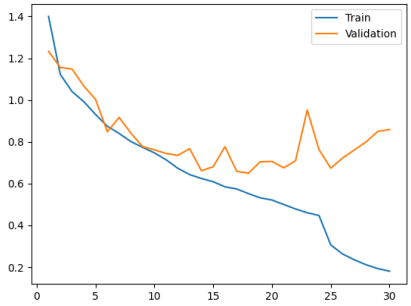
Figures



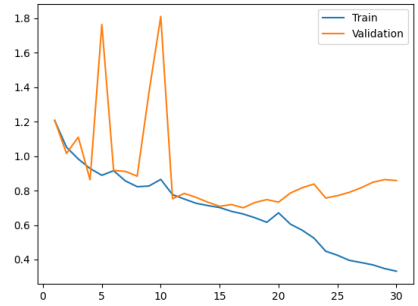
(a) EfficientNet-B0 Baseline



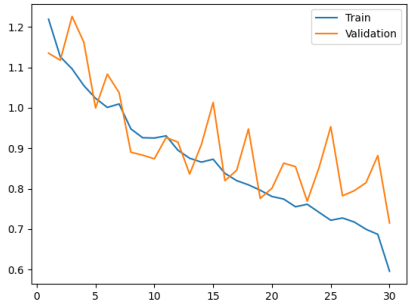
(b) ResNet34 Baseline



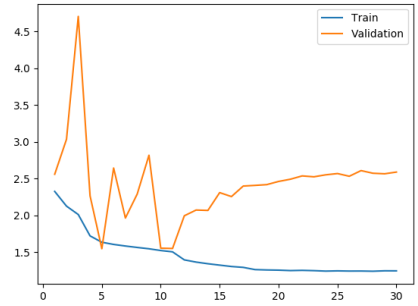
(c) EfficientNet-B0 with SAM



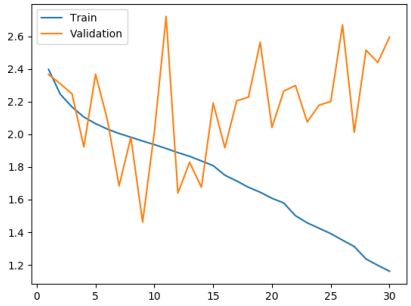
(d) EfficientNet-B0 with Spinal FC



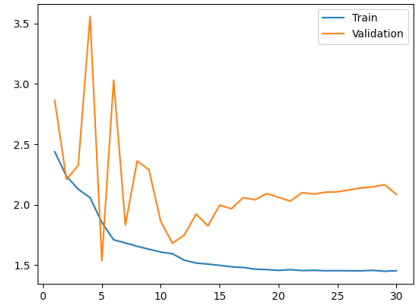
(e) EfficientNet-B0 with SAM and Spinal FC



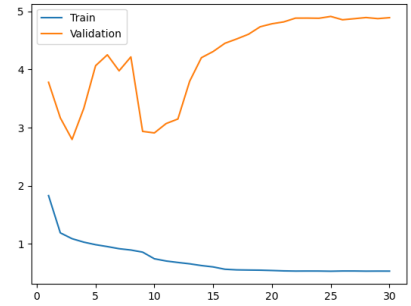
(f) EfficientNet-B0 with Label Augmentation



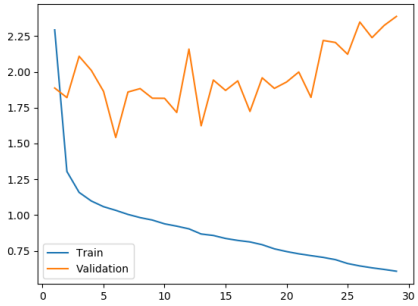
(g) EfficientNet-B0 with Label Aug. and SAM



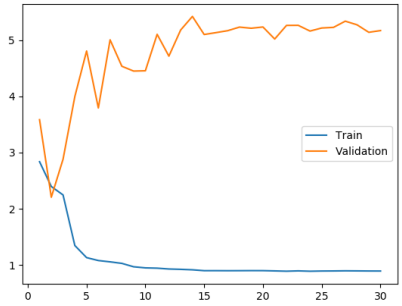
(h) EfficientNet-B0 with Label Aug. and Spinal FC



(i) EfficientNet-B0 with Color Permutation



(j) EfficientNet-B0 with Color Perm. and SAM



(k) EfficientNet-B0 with Color Perm. and Spinal FC

Figure A.1: Loss Curves of Best Models up-to 30$^{\text{th}}$ Epoch (Axis y : Loss, x : # of epochs)

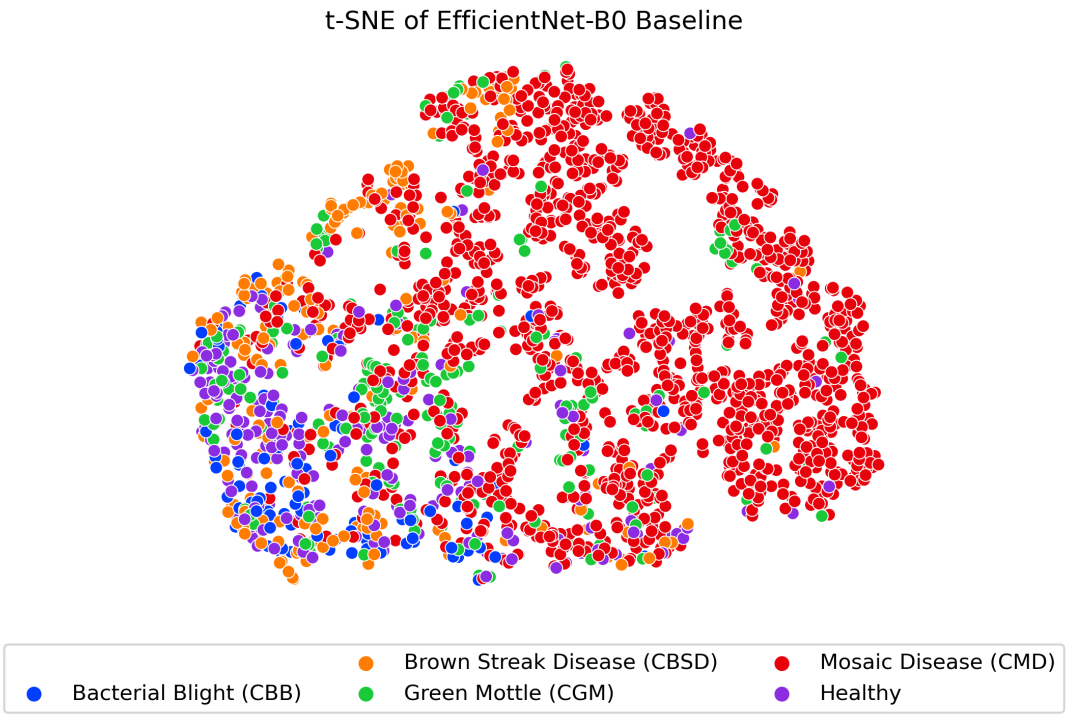


Figure A.2: t-SNE of EfficientNet-B0 Baseline

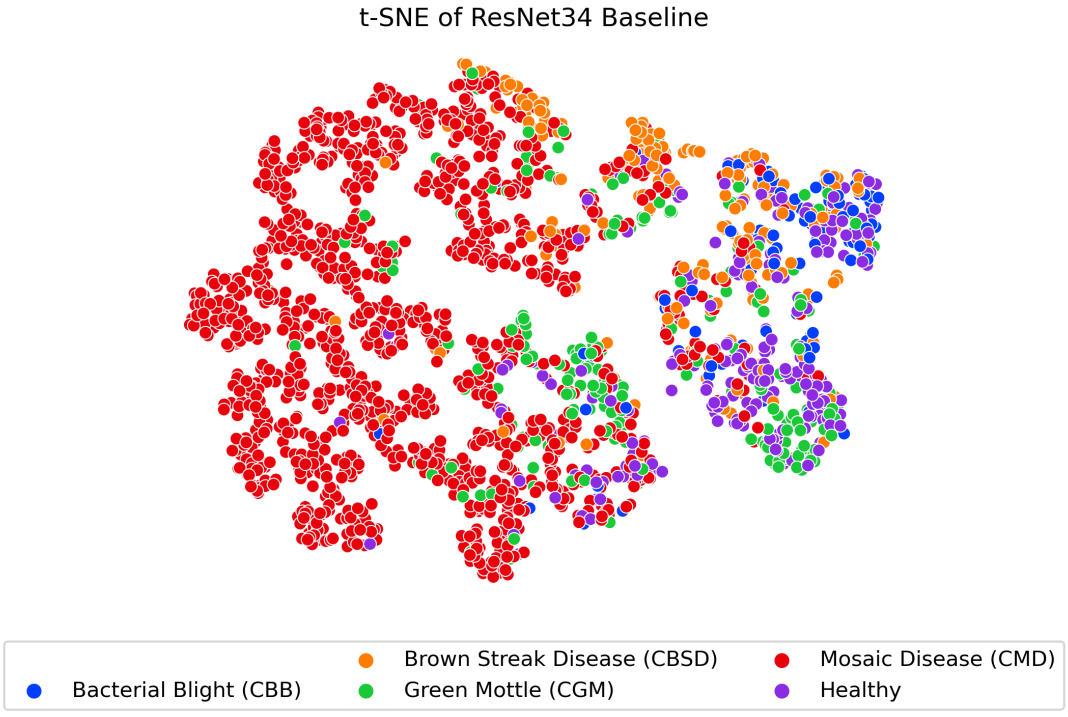


Figure A.3: t-SNE of ResNet34 Baseline

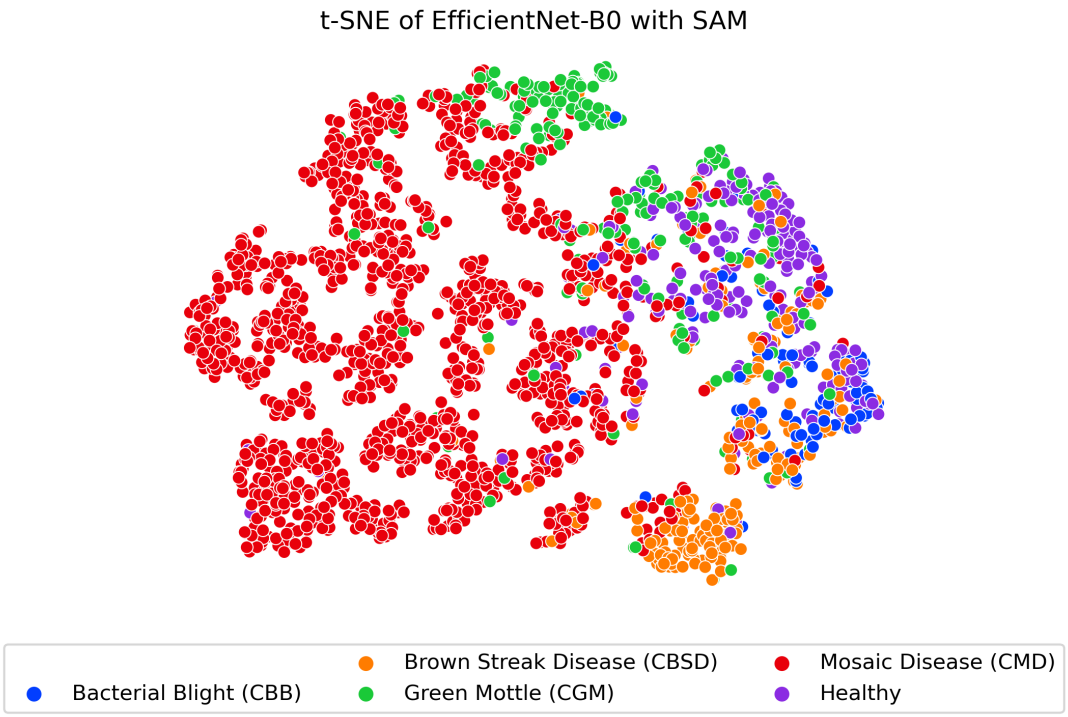


Figure A.4: t-SNE of EfficientNet-B0 with SAM

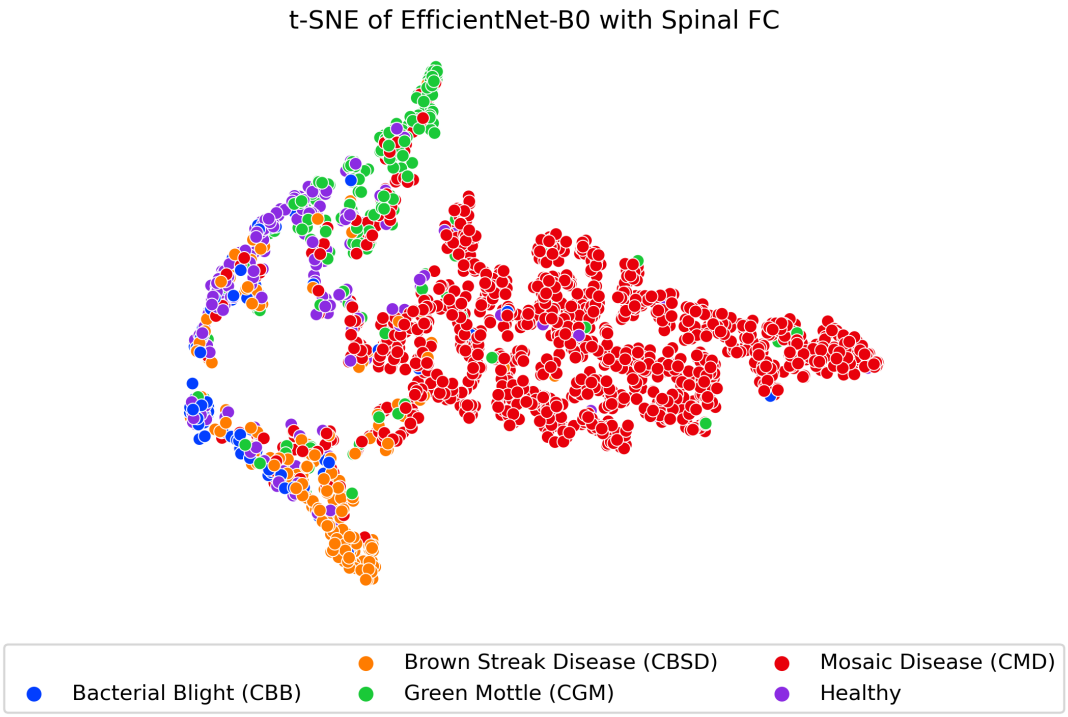


Figure A.5: t-SNE of EfficientNet-B0 with Spinal FC

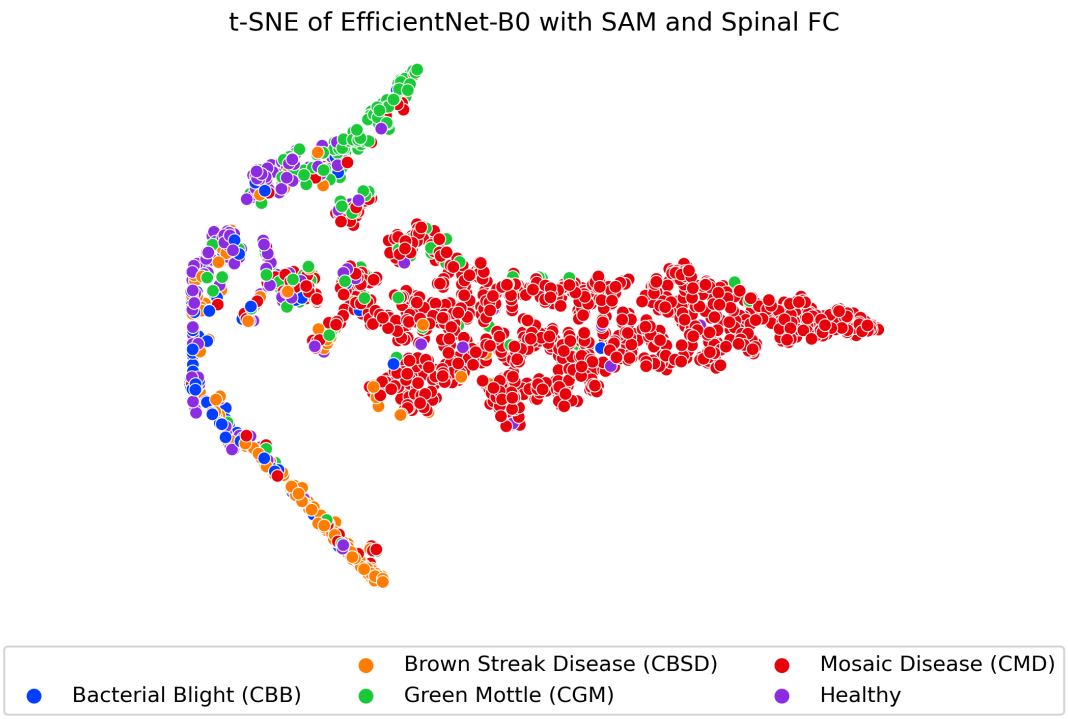


Figure A.6: t-SNE of EfficientNet-B0 with SAM and Spinal FC

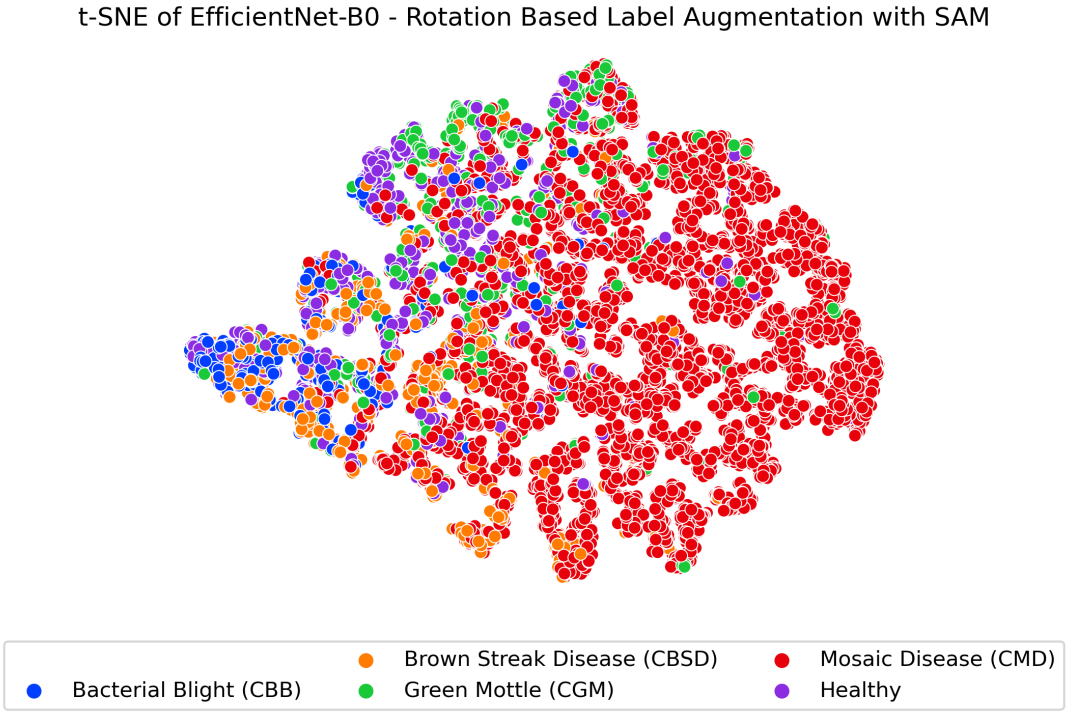


Figure A.7: t-SNE of EfficientNet-B0 - Rotation Based Label Augmentation with SAM

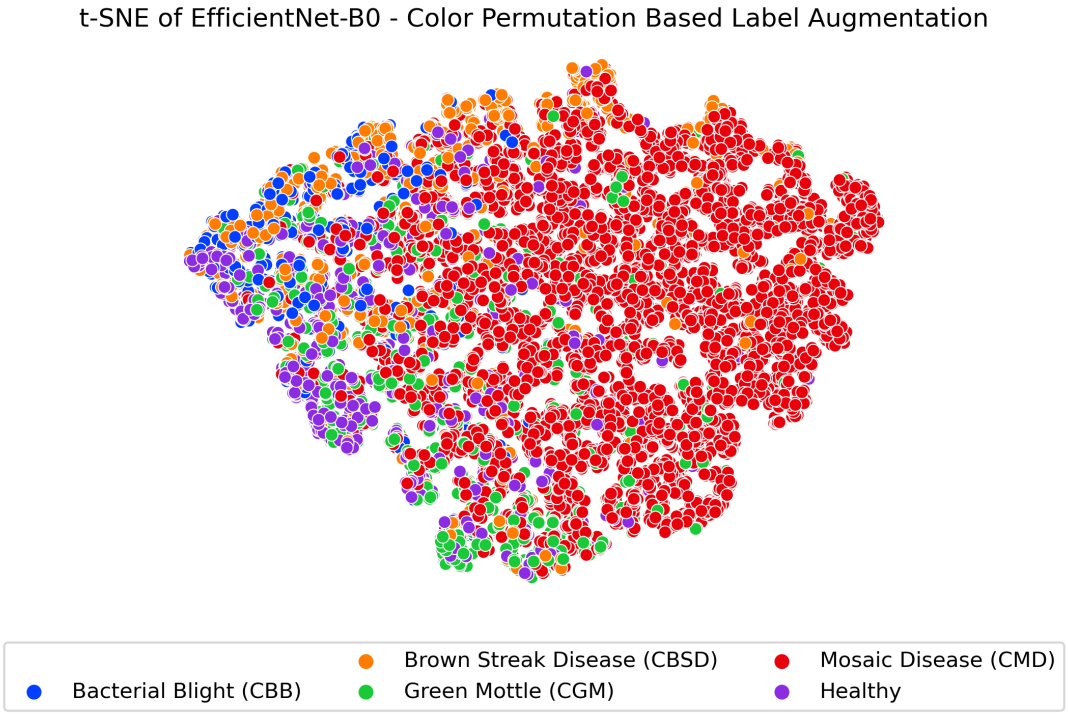


Figure A.8: t-SNE of EfficientNet-B0 - Color Permutation Based Label Augmentation