# COMP9417 Project Report

## Cassava Leaf Disease Classification

Group No.19th

Bingyu Yang z5328465

Dingheng Li z5291137

Jianyi Huang z5230370

Jie Mei z5173405

# 1. Introduction

With the rapid development of artificial intelligence, image recognition has become a popular technology, and thousands of companies and millions of consumers are using this technology every day. It has many application scenarios, including e-commerce, games, automobiles, manufacturing, and agriculture. In agriculture, it is reflected in the low yields caused by viral diseases of crops. With the help of image recognition, common diseases can be identified so that they can be treated.

Now find the Cassava Leaf Disease Classification (Makerere University AI Lab, 2021) project on Kaggle, which allows us to apply image recognition algorithms. The purpose of this project is to identify diseased cassava through the automation of image classification, which can effectively reduce the impact of labor-intensive, low-supply and expensive. On the other hand, there is a greater challenge that African farmers can use low-bandwidth mobile cameras to achieve the same recognition effect. The data set is a data set of 21,367 labeled images collected during regular surveys in Uganda provided by Kaggle and was carried out by experts from the National Crop Resources Research Institute (NaCRRI) in cooperation with the Artificial Intelligence Laboratory at Makerere University in Kampala Comment.

In this project we aimed to classify each cassava image into four disease categories or a fifth category representing healthy leaves: Cassava Bacterial Blight (CBB), Cassava Brown Streak Disease (CBSD), Cassava Green Mottle (CGM), Cassava Mosaic Disease (CMD), Healthy. Cassava Mosaic Disease (CMD), Healthy. worthy of our attention is to diversify the score of the 5 labels, we will try to get the optimal model from different perspectives. Our project will be evaluated according to its classification accuracy in line with the Kaggle evaluation criteria.

# 2. Implementation

In this section we examine the distinct factors that need to be considered for the final model, which include sampling of the dataset, image pre-processing, selection of an appropriate network model, selection of different optimizers and learning rate adjustment strategies for the existing network model. Finally, data calculation and analysis based on model evaluation criteria.

## 2.1 Data set sampling

With the entire training dataset provided by Kaggle, we found several times as many labels 3 categories as other categories (as shown in Table 1). A straightforward way to address the imbalanced dataset is to smooth them out, oversampling a few categories, or under sampling the main categories.

This allows us to create a balanced dataset that theoretically keeps the classifier from favoring one of the classes. However, we believe that there may be drawbacks, and that oversampling a

few classes can lead to overfitting the model, as it introduces duplicate instances drawn from an already small pool of instances. Similarly, under sampling major categories may eventually lead to missing important instances that reflect significant differences between the two classes. Therefore, we trained the model with the full dataset and the balanced dataset and finally compared the results. Our balanced samples were randomly selected from 2000 label 3 samples, and the remaining samples were thrown away to obtain a new balanced dataset.

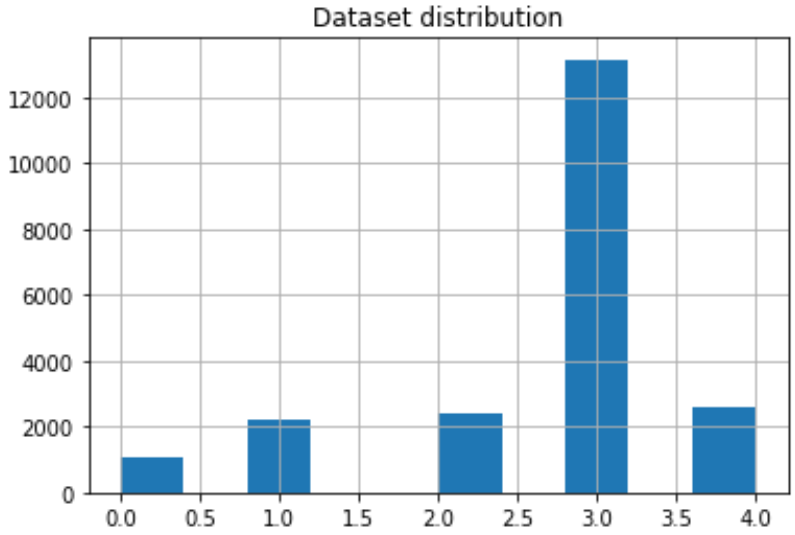



Table1 Unbalanced data sets

## 2.2 Image pre-processing

The quality of images in image classification directly affects the design of recognition algorithms and the accuracy of the results, so pre-processing is required before image classification (feature extraction, segmentation, matching and recognition, etc.). The main purpose of image preprocessing is to eliminate irrelevant information from images, recover useful and true information, enhance detectability of relevant information, maximize simplification of data, and thus improve the reliability of feature extraction, image segmentation, matching and recognition.

In this section we have used three main methods to achieve this using pytorch.transform.

• geometric transformation: RandomVerticalFlip implements a random vertical flip of a given image with a given probability.

• Image Enhancement: GaussianBlur implements blurring the image using a randomly selected Gaussian blur.

• Affine transform: Affine transform is a 2-dimensional linear transform consisting of 5 basic operations, namely, rotation, translation, scaling, misalignment, and flip.

## 2.3 Review of suitable network models

In this section we choose to use the two known popular network models Resnet and ResNeXT.

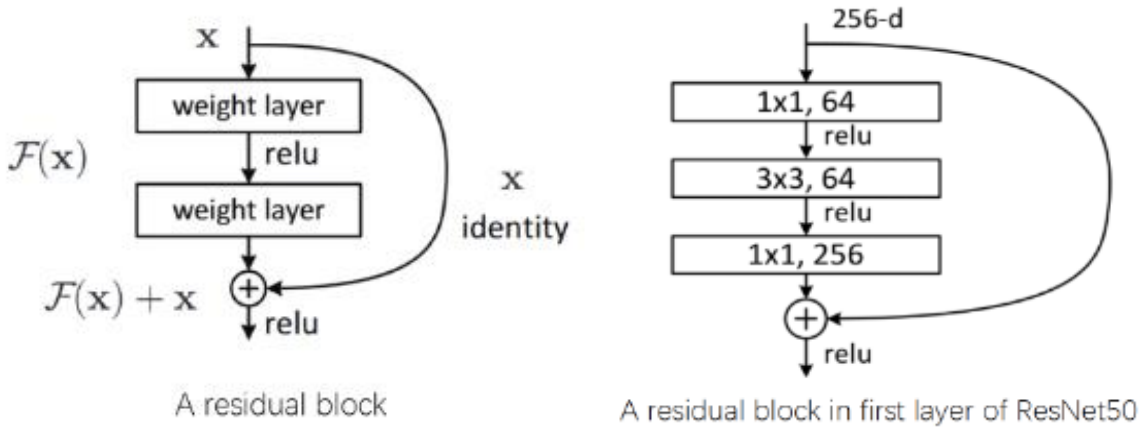
### ResNet (He, 2015)

Intuitively, adding layers to the shallow network will improve the performance of the model on the training set and test set, because the model is more complex and can fit the mapping relationship better. However, in fact, after adding more layers to the network, the performance drops rapidly. ResNet can effectively alleviate this problem.

ResNet makes the model easier to optimize by adjusting the model structure. A stack of server layers is called a residual block. A skip connection which bypasses the layers is added. And their output will add the skip connection. This method can map to activations earlier in the network to preserve the gradient.

For a residual block, suppose the residual is F(x)=H(x)-x, H(x) is the expected network layer output H(x), and x is the input of the layer. ResNet does not learn H(x) but learns the residual F(x). The learning goal of ResNet is to keep the F(x) close to 0.

In the project, we used a typical ResNet model ResNet50, which has 1 MaxPool, 48 Convolution layers and 1 Average Pool layer. Compared with ResNet18 and ResNet34, the skip function of ResNet50 skips 3 layers instead of 2.




Note these are figure 2 and figure 5 from Deep Residual Learning for Image Recognition

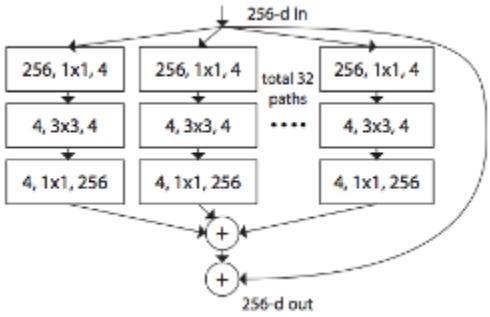
### ResNeXt (Xie, 2017)

ResNeXt is a variant of ResNet. In general, the model can deepen or widen the network to improve accuracy, however, the number of hyperparameters (such as the number of channels, filter size, etc.) will also increase, and the difficulty of network design and computational cost will also increase. ResNeXt uses a similar strategy to Inception's split-transform-merge and VGG/ResNets' strategy of repeating layers.

Unlike Inception, each transformation in the block has the same topology. By using repeated topologies, the design of the model is simplified because the hyperparameters used in each topology is the same. The representation power of each set of transformation is still preserved because of aggregation. This allows this structure to improve accuracy without increasing the complexity of the parameters, while also reducing the number of hyperparameters.

Like ResNet, the residual function of ResNeXt can be present as. C is cardinality, indicating the size of the set of transformations. In ResNeXt, cardinality is a crucial factor in addition to the depth and width dimensions. By adding cardinality, the accuracy can be improved. In the article by Saining Xie et al [2]., in some cases, increasing cardinality is more effective than increasing depth and width.

.




A block of ResNeXt with cardinality = 32.

Note figure from Aggregated Residual Transformations for Deep Neural Networks [2]

## 2.4 Appropriate optimizers and learning rate strategies

### Warm up learning rate

Warmup is a method of learning rate pre-warming mentioned in the ResNet paper. It uses a smaller learning rate at the beginning of the training, trains some epochs, and then uses a preset learning rate for the rest of the training.

At the early stage of training, each data is new to the model, and the weights of the model are initialized randomly. If the data at this stage are highly correlated or have irrelevant strong features, a large learning rate may cause the model to be skewed to these features, resulting in overfitting. Multiple rounds of training are needed to correct the model. Warm-up method uses a smaller learning rate at the beginning, so that the model can gradually stabilize.

Suddenly increasing the learning rate will cause the performance of model fluctuating. One of the solutions is to dynamically change the learning rate during the warm-up phase, so that the learning rate changes smoothly to the preset learning rate. The torch library provides multiple

learning rate adjustment strategies such as *LambdaLR*, *MultiplicativeLR*, *StepLR* and *MultiStepLR*. In our project, MultiStepLR is used.

*MultiStepLR* (optimizer, milestones, gamma, last_epoch)

$$lr_{\text{epoch}} = \begin{cases} Gamma * lr_{\text{epoch - 1}}, & \text{if epoch in [milestones]} \\ lr_{\text{epoch - 1}}, & \text{otherwise} \end{cases}$$

Once the number of epochs reaches one of the milestones, the learning rate is decayed by gamma. Set learning rate to the preset learning rate when epoch reaches last epoch.

## Gradient descent optimization selection

• Stochastic gradient descent (SGD)

SGD is a variant of the gradient descent method. The general gradient algorithm needs to calculate all samples of the data set for calculation. This is an overhead when the training data set is huge. SGD uses a random subset of the sample to replace all samples to adjust the parameter vector θ.

• Adaptive Moment Estimation with decoupled weight decay (AdamW)

In SGD, we have a common learning rate for all parameters, so we can update all parameters at once. However, when the change of the gradient of each parameter is quite different, the direction of each update of the gradient may not be globally optimal. Even if the optimal solution can be obtained in the end, it is costly. RMSProp is a method that use exponentially decaying average of past squared gradients to calculate different learning rate for each parameter θ.

Adam is an update to the RMSProp optimize. Adam not only uses exponentially decaying average of the past squared gradients but also use exponentially decaying average of past gradients.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

AdamW is an improved algorithm based on Adam+L2 regularization, which fixes weight decay in Adam. Adam+L2 regularization will add regularization term when calculating the gradient. Because moving averages of the gradient and its square (m and v) are calculate with gradient, they will be affected by regularization term. AdamW fixed this problem by using regularization term when updating the parameters θ. See blow:

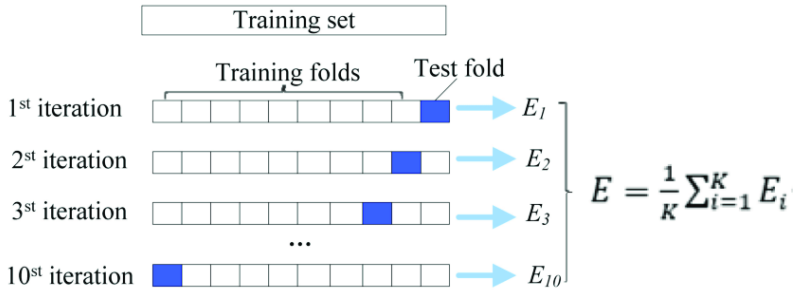
**Algorithm 2** Adam with $L_2$ regularization and Adam with decoupled weight decay (AdamW)

**given** $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}, \lambda \in \mathbb{R}$
**initialize** time step $t \leftarrow 0$, parameter vector $\boldsymbol{\theta}_{t=0} \in \mathbb{R}^n$, first moment vector $\boldsymbol{m}_{t=0} \leftarrow \boldsymbol{0}$, second moment vector $\boldsymbol{v}_{t=0} \leftarrow \boldsymbol{0}$, schedule multiplier $\eta_{t=0} \in \mathbb{R}$
**repeat**
  $t \leftarrow t + 1$
  $\nabla f_t(\boldsymbol{\theta}_{t-1}) \leftarrow \text{SelectBatch}(\boldsymbol{\theta}_{t-1})$ ▷ select batch and return the corresponding gradient
  $\boldsymbol{g}_t \leftarrow \nabla f_t(\boldsymbol{\theta}_{t-1}) + \lambda \boldsymbol{\theta}_{t-1}$
  $\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + (1 - \beta_1)\boldsymbol{g}_t$ ▷ here and below all operations are element-wise
  $\boldsymbol{v}_t \leftarrow \beta_2 \boldsymbol{v}_{t-1} + (1 - \beta_2)\boldsymbol{g}_t^2$
  $\hat{\boldsymbol{m}}_t \leftarrow \boldsymbol{m}_t / (1 - \beta_1^t)$ ▷ $\beta_1$ is taken to the power of $t$
  $\hat{\boldsymbol{v}}_t \leftarrow \boldsymbol{v}_t / (1 - \beta_2^t)$ ▷ $\beta_2$ is taken to the power of $t$
  $\eta_t \leftarrow \text{SetScheduleMultiplier}(t)$ ▷ can be fixed, decay, or also be used for warm restarts
  $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \eta_t \left( \alpha \hat{\boldsymbol{m}}_t / (\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon) + \lambda \boldsymbol{\theta}_{t-1} \right)$
**until** *stopping criterion is met*
**return** optimized parameters $\boldsymbol{\theta}_t$

Note Figure from DECOUPLED WEIGHT DECAY REGULARIZATION [5]

## 2.5 10-Fold Cross Validation

Cross-validation is to repeatedly use the data, slice and dice the obtained sample data, combine them into different training and testing sets, use the training set to train the model, and use the testing set to evaluate how well the model predicts. On this basis, we can get several different training sets and test sets, and a sample in a training set may become a sample in a test set in the next time. We use 10-fold cross-validation to divide the sample data into 10 randomly and choose 9 randomly each time as the training set and the remaining 1 as the test set. When the round is completed, 9 copies are randomly selected again to train the data.




10-fold Cross-validation

Note figure from Predicting Mechanical Properties of High-Performance Fiber-Reinforced Cementitious Composites by Integrating Micromechanics and Machine Learning. Materials [6]

## 2.6 Model Evaluation Metrics

The results of this Kaggle competition are judged based on categorization accuracy. In the selection of the final model, we make a judgment by confusion matrix. The confusion matrix provides a good visualization of how well a model is classified on each category (positive and negative).

Each row of the confusion matrix M denotes the true class, each column denotes the predicted class. That is, M[i][j] denotes the number of samples predicted to be class j among all samples whose true class is I (Steven, 2019) [4]. We focus on the diagonal region of the confusion matrix, which indicates that the real class and the predicted class coincide, that is, the TP region.

FP of a category: the sum of all elements in that column minus the TP of that column

FN of a category: the sum of all elements of the row minus the TP of the row

TN of a category: the sum of the entire matrix minus the (TP+FP+FN) of the category

The accuracy rate indicates the percentage of samples (TP and TN) with correct predictions among all samples.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

When the data set is not balanced, the accuracy rate will not be a good representation of the model performance. There may be cases where the accuracy rate is high, and a few classes of samples are all scored wrong, the precision rate and recall rate should be selected as important factors for judging the model.

$$\text{Precision} = \frac{TP}{TP + FP} \qquad \text{Recall} = \frac{TP}{TP + FN}$$

The precision rate represents the proportion of samples in the true class that are positively predicted; the recall rate of the positive class represents the proportion of samples in the true class that are positively predicted by the model.

The recall rate is only related to the sample with true positive, and not to the sample with true negative, while the precision rate is affected by both classes of samples.

$$F1\text{-}score = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

The F1 score can be considered as a reconciled average of the accuracy and recall of the model. When there is a conflict between precision and recall, the F1 score is used to determine the merit of the model.

# 3. Results

## 3.1 Exploratory data analysis and splits

In this competition, there are 21397 images available in the training set including the image ID and ID code for disease. The ten-fold cross validation method was selected to assess the accuracy of the model. As mentioned above, based on the unbalanced data distribution, it is obvious that we need two thousand samples of label three to get a balanced dataset (Table 2 below).

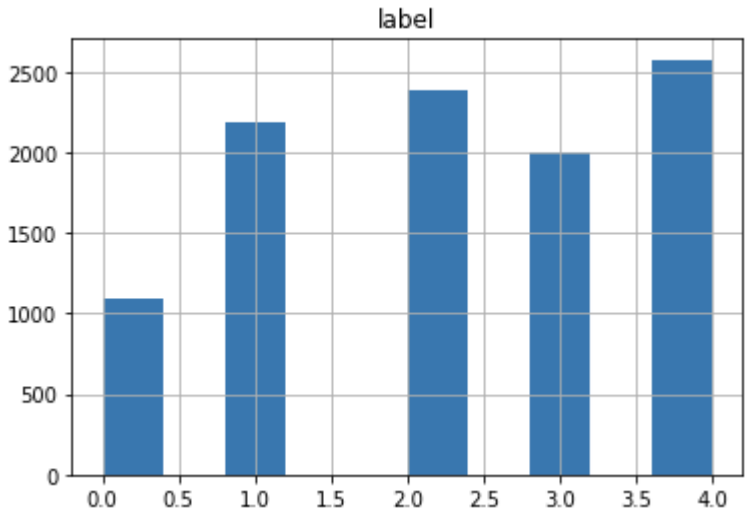



Table 2 Balanced data sets

Four out of five labels are around two thousand samples, which means the proportion of each label is similar. Only one of these labels is well below the average.

## 3.2 Comparison of partial and full datasets

Here two comparisons of partial and full datasets are performed as ResNet50 and ResNet50 with warm-up models are used. Optimizers all default to AdamW.

### 3.2.1 Partial dataset vs. Full dataset in ResNet50 model

Among the ten groups sampled in Resnet50 model for the partial, as shown in the Table 3, the range of accuracy is between 81 and 85. Most accuracy values are above 82.5 and the mean value is about 83. In full datasets with the same model, there is no need to sample. The ten-fold validation here is to divide all the data into ten groups which ten accuracies are displayed on Table 4. The overall accuracy values are higher than 88, and the first seven points are stable between 89 and 90.

### 3.2.2 Partial dataset vs. Full dataset in ResNet50 with warm-up model

In another Resnet50 with warm-up model, the accuracy values of partial dataset, in the line chart of Table 5, are from 77 to 82 and average value is about 80 while the range of accuracy in full dataset is between 85 and 90 and Table 6 provides a clear demonstration. Its minimum accuracy is higher than the maximum accuracy in partial dataset.

In the first case of both using Resnet50 model, the comparison is made by varying how much of the selected dataset. The accuracy values from the full dataset are higher overall than those from the partial dataset. The second comparison with Resnet 50 with warm-up model, also have the same result, that is the average value of accuracy in full dataset is higher than it in partial dataset.

#### 3.2.3 Accuracy in partial dataset vs. Accuracy in full dataset in ResNet50 model

Within the same model, the training accuracy values, and test accuracy values also reflect different trends. In the ResNet50 model, the range of training accuracy and test accuracy in partial dataset (shown in Table 7) is between 0.6 and 0.85, so the variation interval is about 0.25 while the variation interval in full dataset is about 0.12, the range from 0.78 to 0.9 in Table 8. Comparing the fits in Table 7 and Table 8, the two lines in the latter are closer together, as most training accuracies in blue are much higher than test accuracies in red provided in the former.

## 3.3 Comparison of Resnet50 with and without warm-up datasets

There are also two comparisons of Resnet50 with and without warm-up model and we ensure that the number of selected datasets is equal. The default optimizer is also AdamW.

#### 3.3.1 ResNet50 vs. ResNet50 with warm-up in partial dataset

In the same partial dataset, the average of Resnet50 model is already calculated in the above in Table 3 and Table 5, that is 83. After the warm-up method is added, the mean value of accuracy is 80, which is lower than the former.

#### 3.3.2 ResNet50 vs ResNet50 with warm-up in full dataset

Continue with the full datasets in different models for second comparison, the average in Resent50 mode is 90 while it in Resent50 with warm-up model is 87. The result as the first comparison, the average accuracy of the add warm-up model is smaller than that of the unadded.

#### 3.3.3 Accuracy in partial dataset vs. Accuracy in full dataset in ResNet50 with warm-up model

After adding the warm-up method, both of Table 9 and Table 10 show that the accuracy values of training and test fluctuate after 10 epochs caused by the influence of warm-up method. In the full dataset that is Table 10, two lines are closer, and the variation interval, 0.11 ranging from 0.76 to 0.87, is smaller than it, 0.16 ranging from 0.64 to 0.8 in Table 9 indicating the partial dataset.

## 3.4 Comparison of ResNet50 model and ResNeXt50 model

Only the full dataset was selected for the comparison here, as it can be seen from the results above that the more data, the higher the accuracy value.

The variable here is the model. Unlike the previous ResNet50, we use ResNeXt50 as the model and have chosen the same optimizer as AdamW. In Table 11, the range of accuracy values in

ReNeXt50 is from 86.5 to 89, which has a smaller change interval 2.5 than that 3.5 in ResNet50 model, but the overall accuracy is lower than ResNet50 model.

## 3.5 Comparison of optimizers: AdamW vs. SGD in ResNet50 model

After that, we changed optimizer to SGD, to compare these two models. Although the datasets are identical, all test accuracy values in ResNeXt50 with SGD are higher than that in the ResNeXt50 with AdamW. The minimum value is higher than 92 and the maximum value is 94.5 from Table 12. It may be the ideal model for us.

Table 13, also known as the training and test accuracy values in ResNeXt50 with AdamW, its two lines follow the same trend as the previous ResNet50 with warm-up model, that is a clear rise after the $10^{th}$ epoch. However, the change in the line of Table 14 shows a flat upward trend in ResNeXt50 with SGD optimizer.

## 3.6 Comparison of confusion matrices

It is easy to visualize where there are errors from the confusion matrix, as they are presented outside the diagonal. The deeper the diagonal the better the classification. We will analyze the advantages and disadvantages of each model by discussing a diagram of the confusion matrix for each of the four models mentioned above.

In Appendix Table 15-18. It is shown that the colors of the diagonal lines all present different degrees of depth. The true positive value is higher in ResNet50 model than in ResNet50 with warm-up and with the same optimizer AdamW, this value performs better in the full dataset, which means the diagonal of the darkest color in ResNet50 full dataset model.

After excluding the warm-up method, we used ResNeXt50 with different optimizers. The same effect as that presented by test accuracy. ResNeXt50 with AdamW optimizer did not show a deeper diagonal in Table 19 and the darkest diagonal is from ResNeXt50 with SGD optimizer.

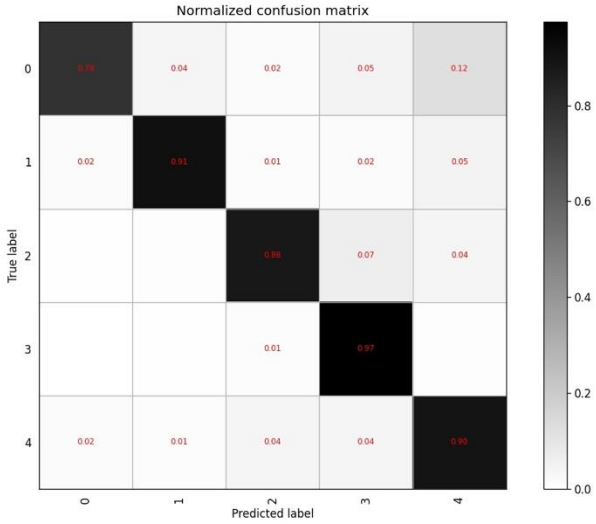



Table 20 Confusion Matrix ResNeXt50 (SGD)

We generated a classification report to compare more directly with figures. The type and data size of the model are the same, only the optimizer has changed. The macro average values of precision, recall and F1-score are 0.84, 0.78, and 0.8, respectively, and weighted average value is 0.89 in ResNeXt50 with full dataset (AdamW) in Table 19. However, all those values in ResNeXt50 with full dataset (SGD) are higher than Table 19 in Table 20.

| Calssification Report in ResNeXt50 (AdamW) | | | | |
|---|---|---|---|---|
| Label | Precision | Recall | F1-score | Support |
| 0 | 0.83 | 0.59 | 0.69 | 100 |
| 1 | 0.87 | 0.75 | 0.81 | 219 |
| 2 | 0.88 | 0.75 | 0.81 | 233 |
| 3 | 0.94 | 0.97 | 0.95 | 1346 |
| 4 | 0.67 | 0.82 | 0.74 | 241 |
| Accuracy | | | 0.89 | 2139 |
| Macro Avg | 0.84 | 0.78 | 0.8 | 2139 |
| Weighted Avg | 0.89 | 0.89 | 0.89 | 2139 |

Table 19 Classification report in ResNeXt50 with Full Dataset (AdamW)

| Calssification Report in ResNeXt50 (SGD) | | | | |
|---|---|---|---|---|
| Label | Precision | Recall | F1-score | Support |
| 0 | 0.88 | 0.73 | 0.8 | 115 |
| 1 | 0.92 | 0.83 | 0.87 | 211 |
| 2 | 0.87 | 0.86 | 0.86 | 242 |
| 3 | 0.96 | 0.97 | 0.97 | 1297 |
| 4 | 0.76 | 0.85 | 0.81 | 274 |
| Accuracy | | | 0.92 | 2139 |
| Macro Avg | 0.88 | 0.85 | 0.86 | 2139 |
| Weighted Avg | 0.92 | 0.92 | 0.92 | 2139 |

Table 20. Classification report in ResNeXt50 with Full Dataset (SGD)

### 3.7 Model evaluation results

Classifying leaves, we set four variables to make comparisons. The first two variables are dataset and learning methods. ResNet50 model are tested separately for the full and partial datasets and then adding warm-up method to this model. Continue with full and partial dataset testing, we changed the type of model as the third variable and add different optimizers as the fourth variable.

In comparison of datasets, each accuracy values of the full datasets containing both of training and test values are higher than those of the partial datasets, and in the comparison of the models with and without the learning method, the Resnet50 model without warm-up has higher accuracy values. Before the model became ResNeXt50, all the optimizers were AdamW. When ResNeXt50 did not yield the desired accuracy values, we add SGD optimizer to the ResNeXt50, and it produced the best result that we can produced.

Although the application of ResNeXt50 with AdamW results in the generated values lower than them in Resnet50 model with the same optimizer, the application of SGD optimizer is more powerful and improves the overall values, making it the best model for tree leaf classification. The diagonal line of its confusion matrix appears visually extremely dark black, especially in the label three, one of the disease leaves, the true positive value is 0.97 almost close to one.

## 4. Conclusion

In this project, we first compare the accuracy values by the amount of the dataset. The model with full dataset always has higher accuracy values. By adding learning method, accuracy values are reduced instead. Therefore, we dropped not only the partial sampling but also the warm-up learning rate. To get an ideal model, ResNeXt50 models are used but its accuracy values are lower than ResNet50. We noticed the default optimizer and change it to SGD optimizer. The values of each term are the highest in this model. We can finally conclude that ResNeXt50 model with SGD optimizer is the most suitable model for this classification task.

## 5. Further work

Since the competition has a running time limit for training models and the local machine has insufficient graphics memory, we can only select models that can be completed within the specified time or graphics memory. We hope to try more complex networks to test on this dataset, for example, we found that Efficient Nets, one of the emerging networks in recent years, has achieved reliable results in the field of image classification, and EfficientNet-B7 has obtained top-1 accuracy of 84.4% and top-5 accuracy of 97.1% on ImageNet dataset. We hope to have the opportunity to use some scaled-up network models to obtain better training results through the hardware resources of the university in the future.

On the other hand, since this competition is also limited to African farmers may only use low bandwidth mobile quality cameras for image recognition, so that if we choose a very large network may lead to a difficult application for African farmers. Therefore, we hope to learn how to select the right depth of model for training according to different size of dataset in the future, and to get the best model by adjusting different parameters in the image pre-processing session to better extract the features of the image.

## 6. Reference

[1] He, K., Zhang, X., Ren, S., & Sun, J. (2015, December 10). *Deep Residual Learning for Image Recognition*. Cornell University. https://arxiv.org/pdf/1512.03385v1.pdf.

[2] Xie, S., Girshick, R.B., Dollár, P., Tu, Z., & He, K. (2017). Aggregated Residual Transformations for Deep Neural Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 5987-5995.

[3] Bieniecki, W., Grabowski, S., & Rozenberg, W. (2007). Image Preprocessing for Improving OCR Accuracy. 2007 International Conference on Perspective Technologies and Methods in MEMS Design. https://doi.org/10.1109/MEMSTECH.2007.4283429

[4] Simske, S. (2019). *Meta-analytic design patterns*. ScienceDirect. https://www.sciencedirect.com/topics/computer-science/confusion-matrix.

[5] Loshchilov, I., & Hutter, F. (2019, January 4). *Decoupled weight decay regularization*. arXiv.org. https://arxiv.org/abs/1711.05101v3.

[6] Guo, Pengwei & Meng, Weina & Xu, Mingfeng & Li, Victor & Bao, Yi. (2021). Predicting Mechanical Properties of High-Performance Fiber-Reinforced Cementitious Composites by Integrating Micromechanics and Machine Learning. Materials. 14. 3143. 10.3390/ma14123143

## 7. Appendix

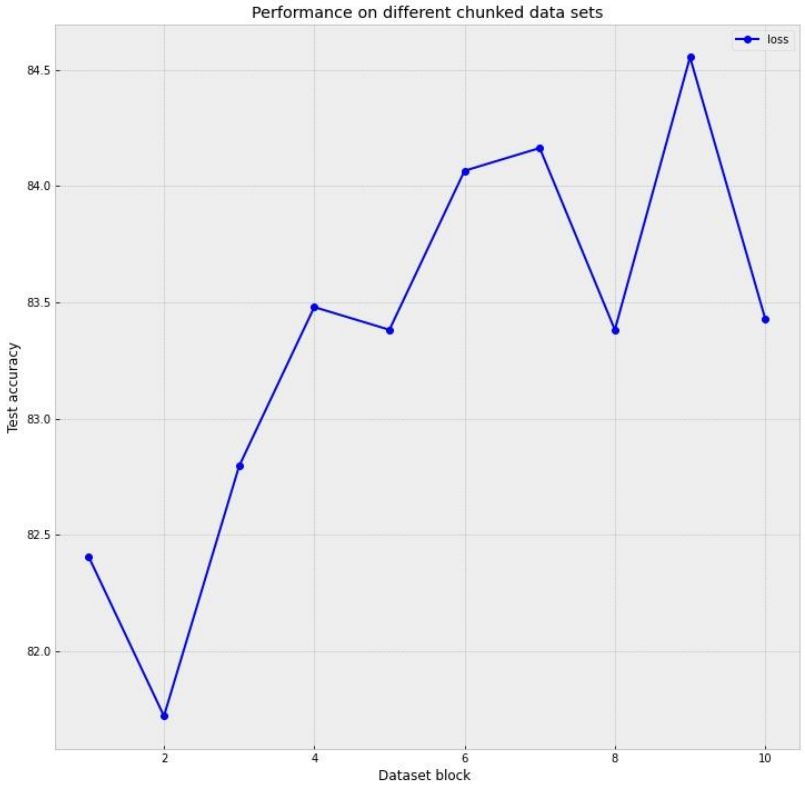



Table 3. Test accuracy values of partial dataset in ResNet50

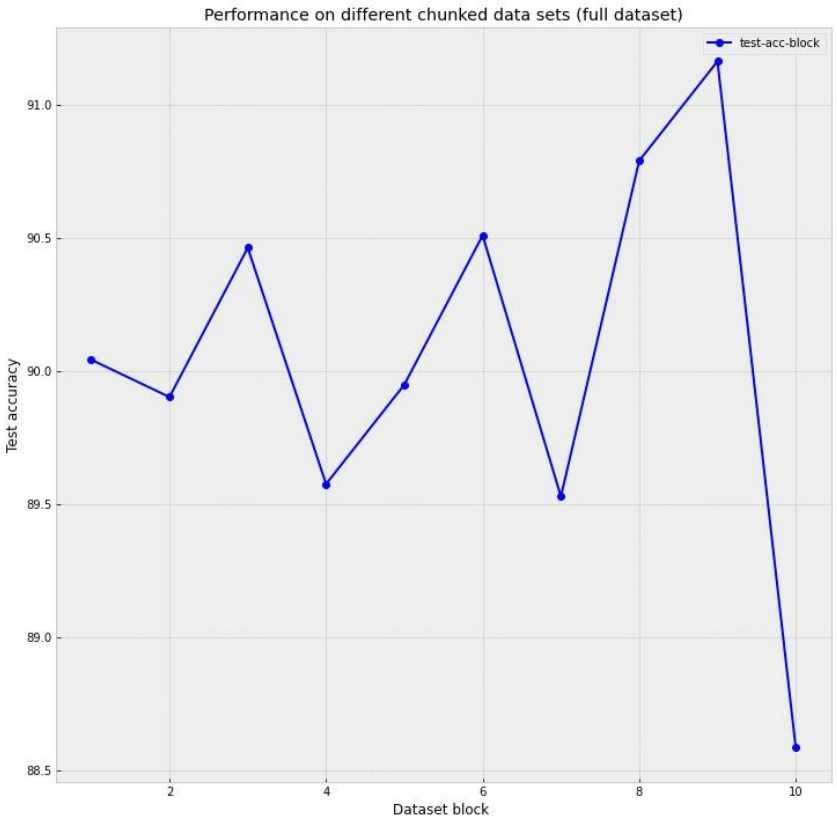



Table 4. Test accuracy values of full dataset in Resnet50

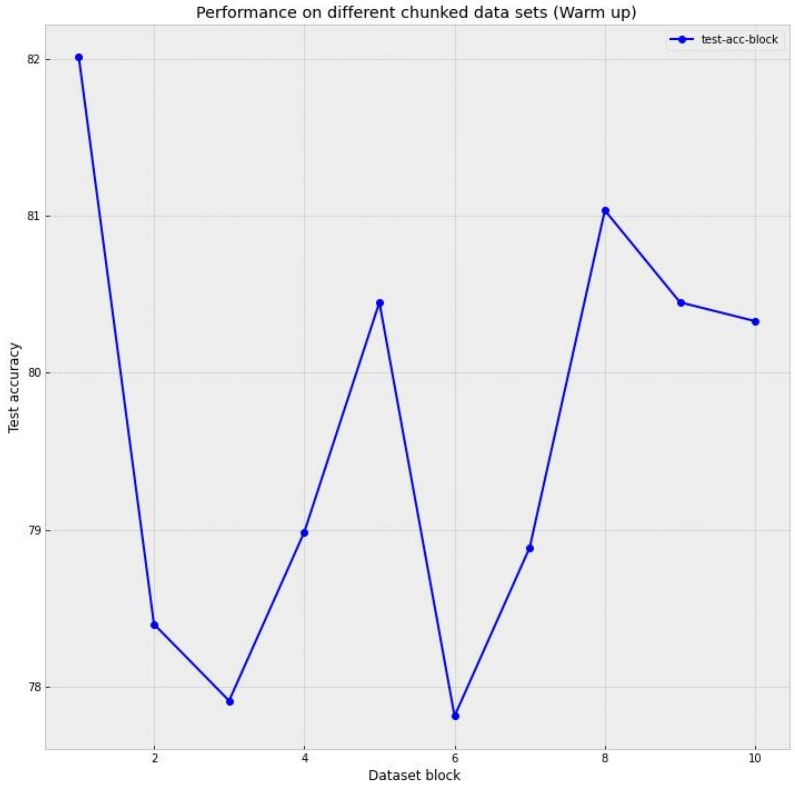



Table 5. Test accuracy values of partial dataset in Resnet50 with warm-up

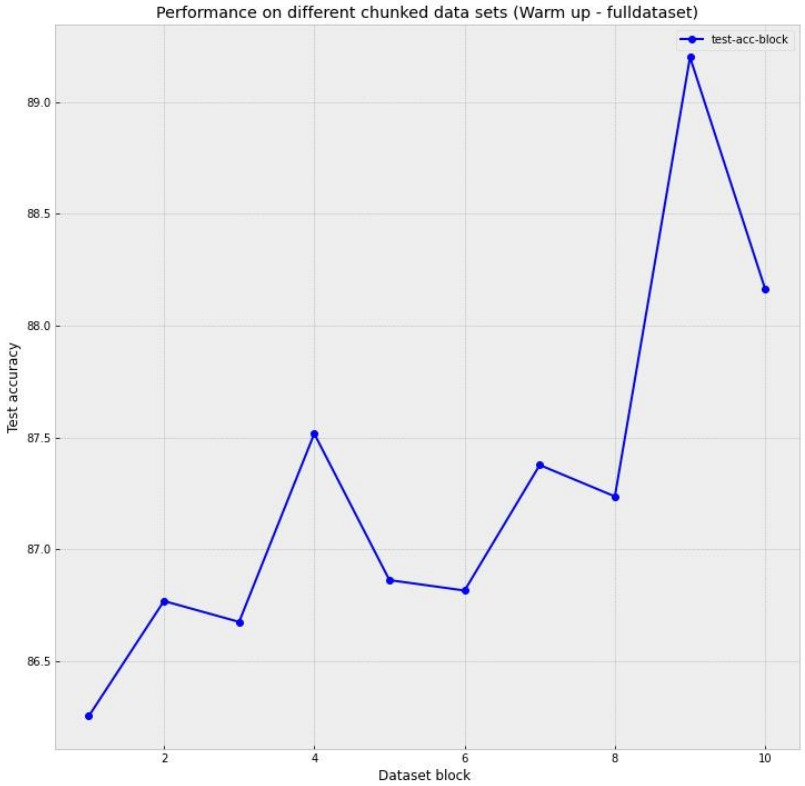



Table 6. Test accuracy values of full dataset in Resenet50 with warm-up

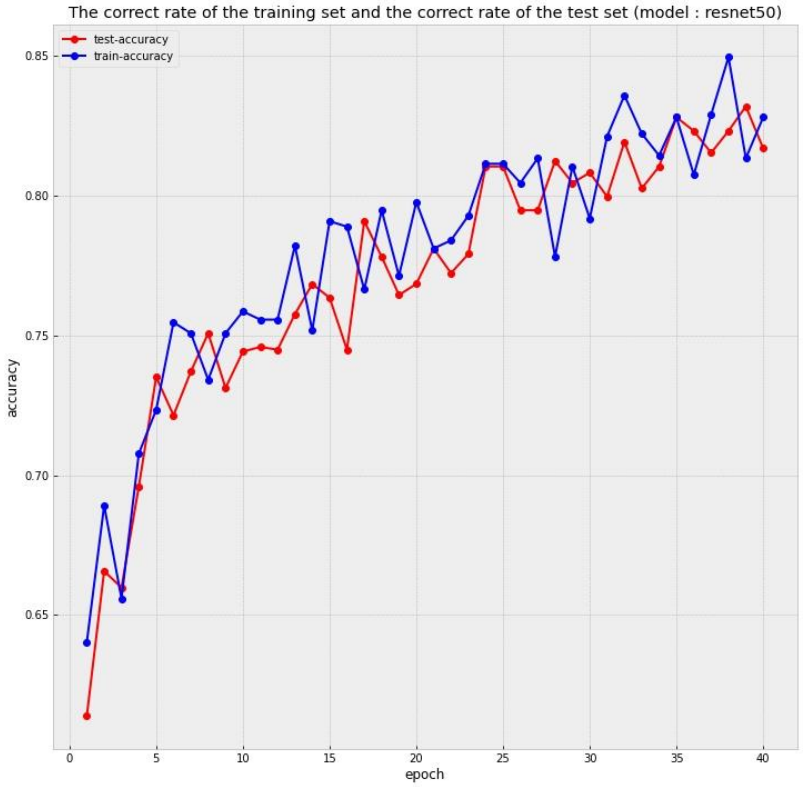



Table 7. Train and test accuracy values in Resnet50 partial dataset

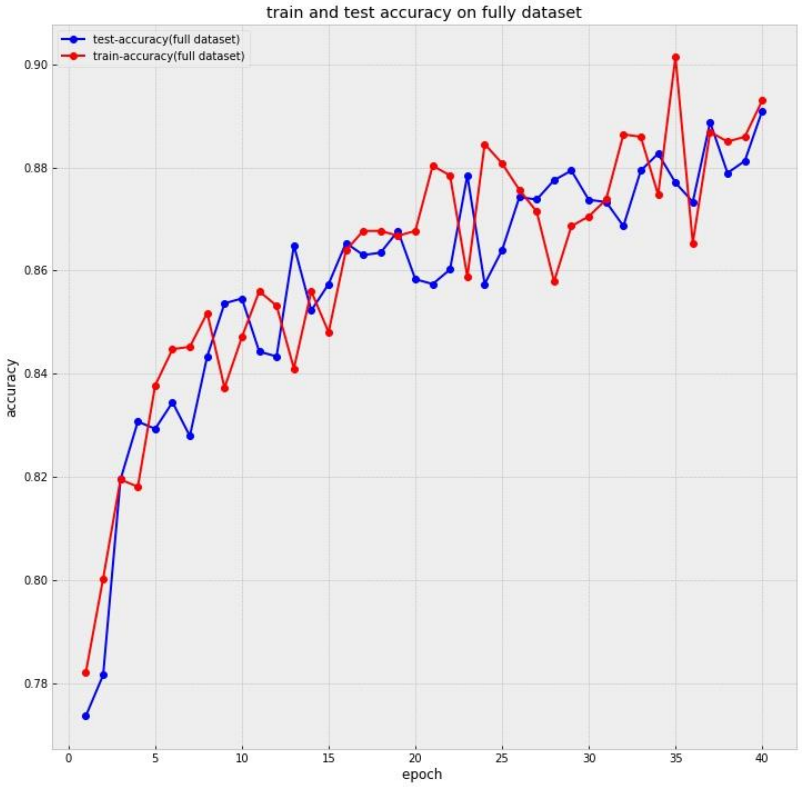



Table 8. Train and test accuracy values in Resnet50 full dataset

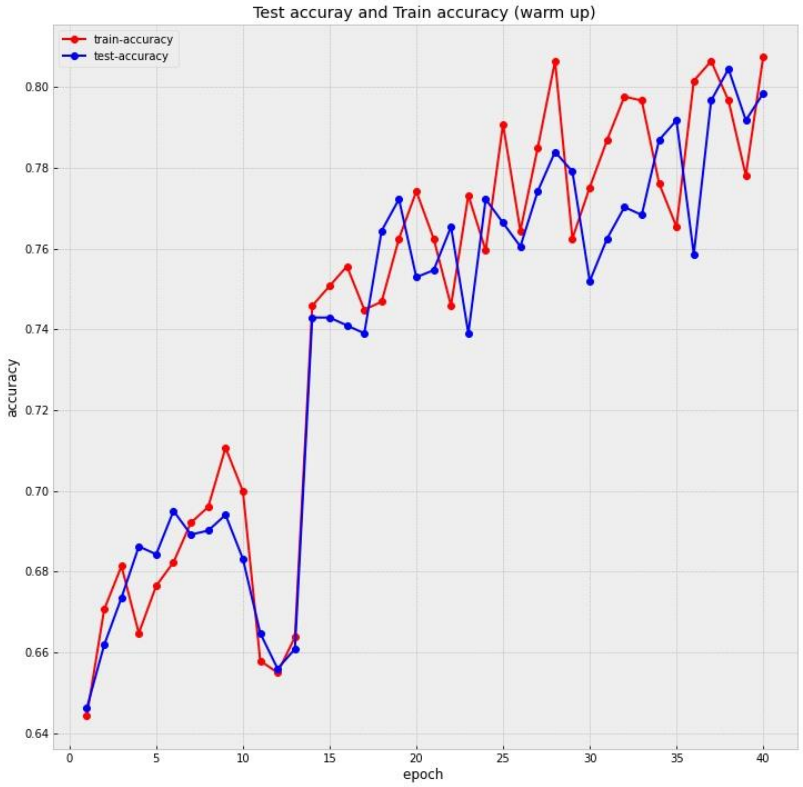



Table 9. Train and test accuracy values in Resent50 with warm-up partial dataset

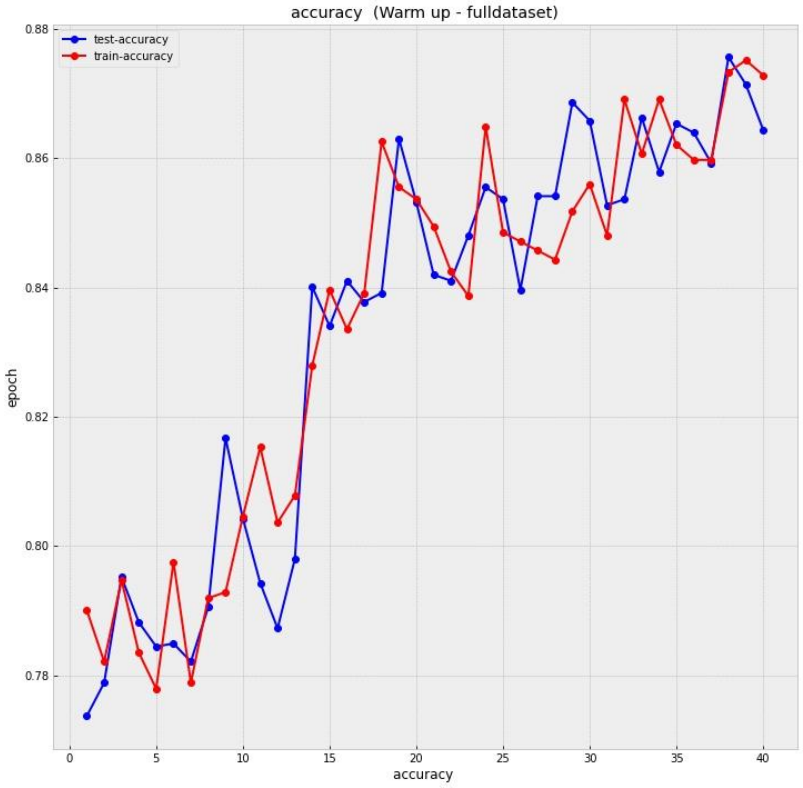



Table 10. Train and test accuracy values in Resent50 with warm-up full dataset

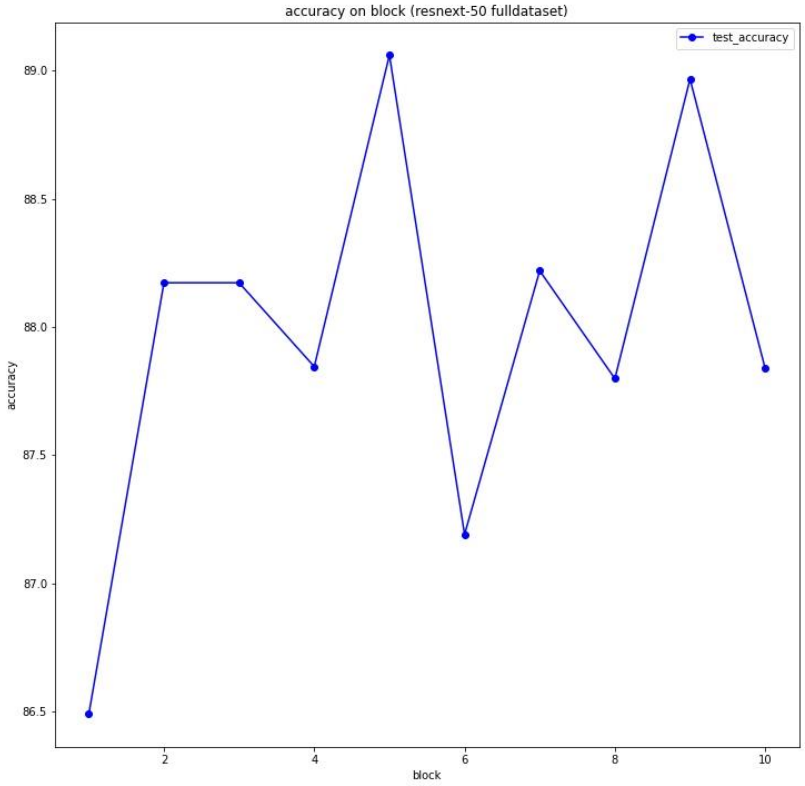



Table11. Test accuracy values in ResNeXt50 with full dataset (AdamW)

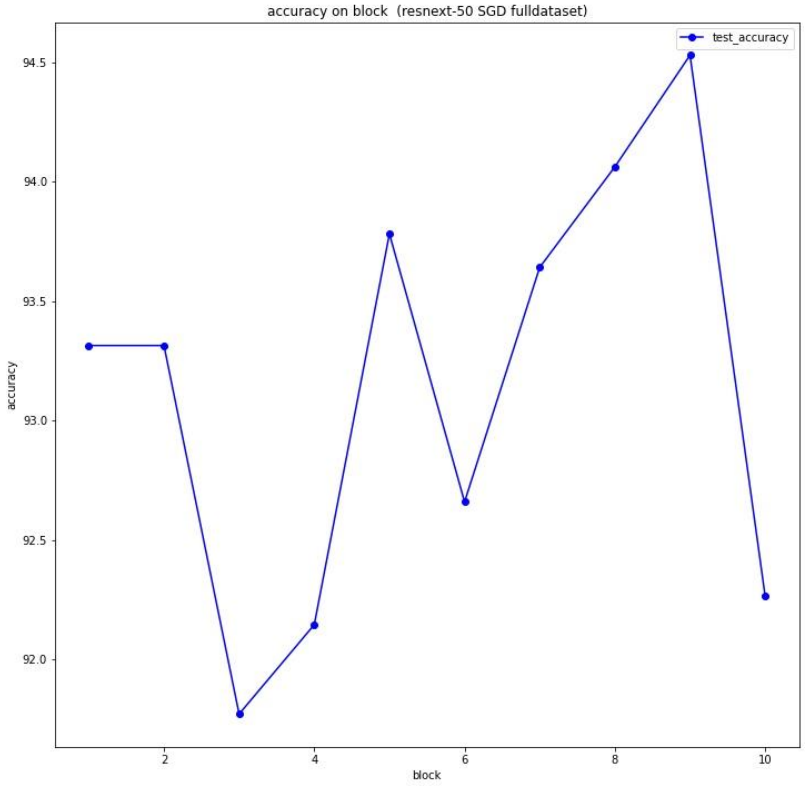



Table 12. Test accuracy values in ResNeXt50 with full dataset (SGD)

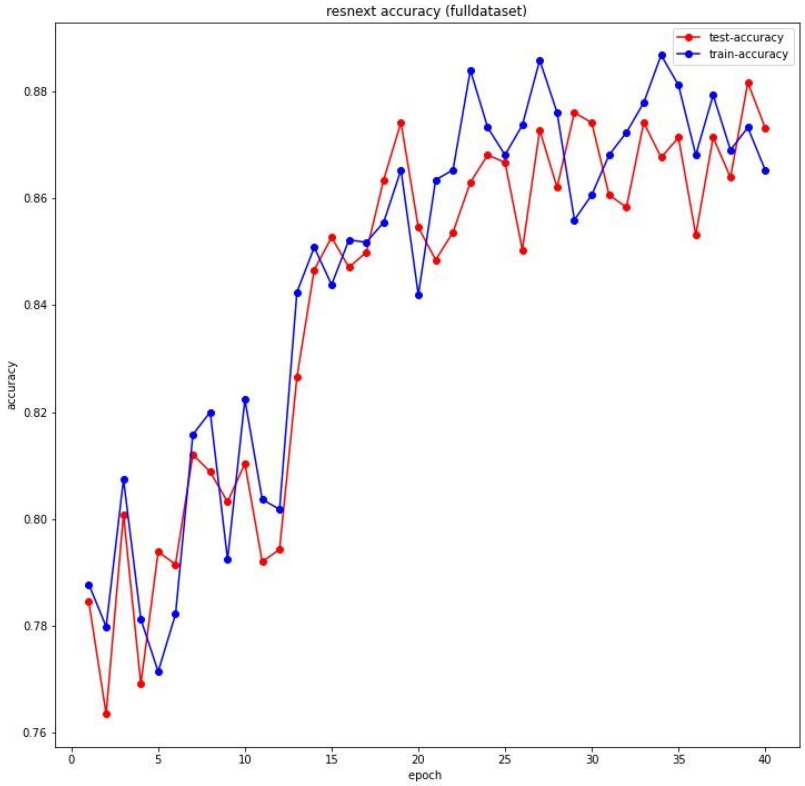



Table 13. Train and test accuracy values in ResNeXt50 with full dataset (AdamW)

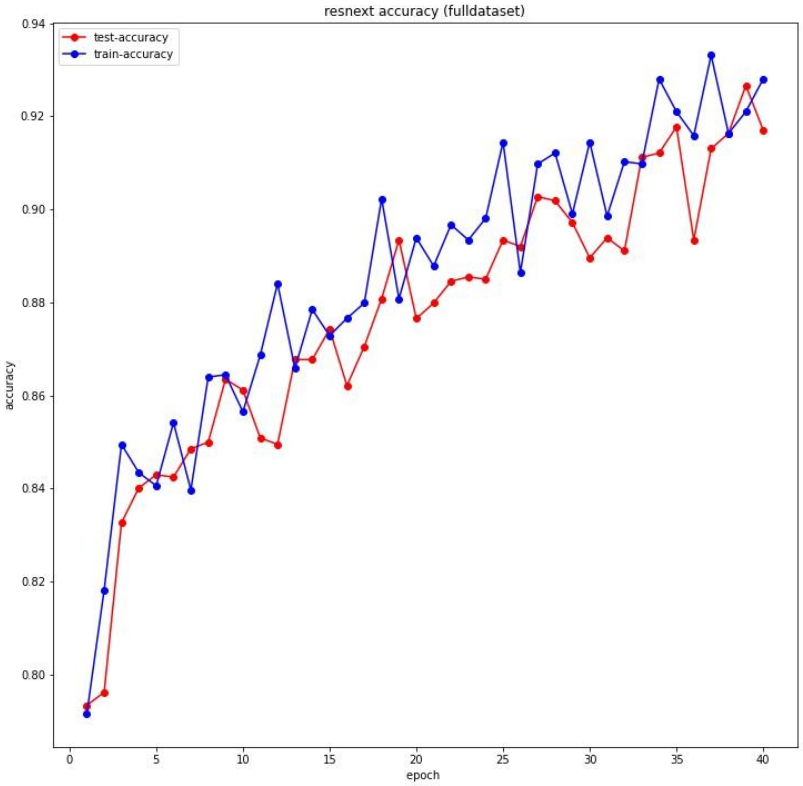



Table 14. Train and test accuracy values in ResNeXt50 with full dataset (SGD)

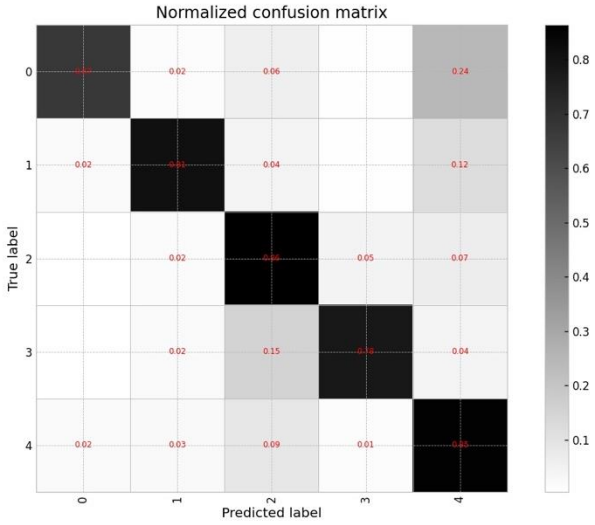



Table 15. Confusion matrix of ResNet50 partial dataset

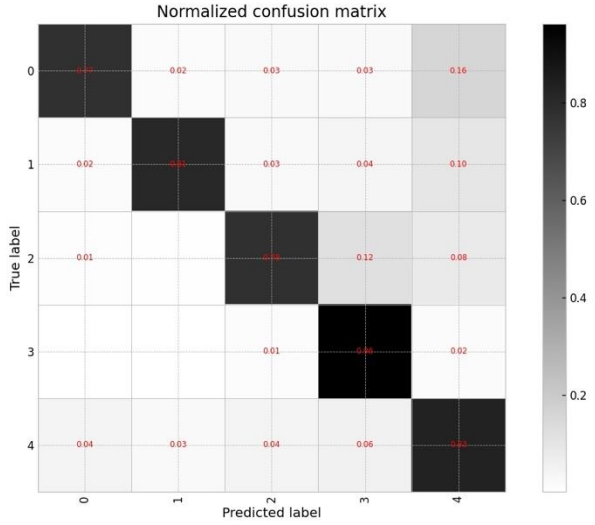



Table 16. Confusion matrix of ResNet50 full dataset

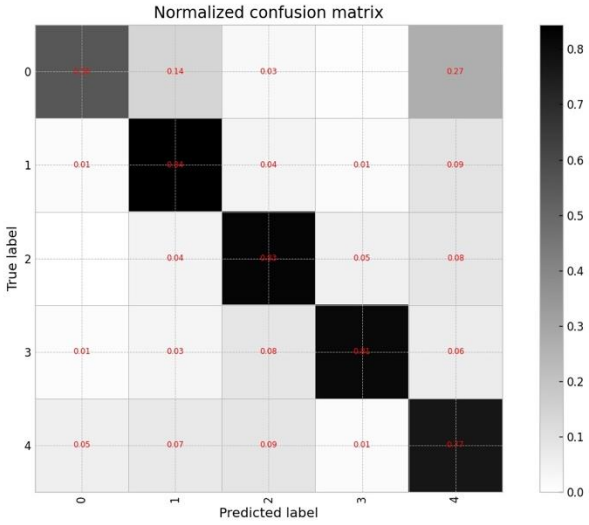



Table 17. Confusion matrix of ResNet50 with warm-up in partial dataset

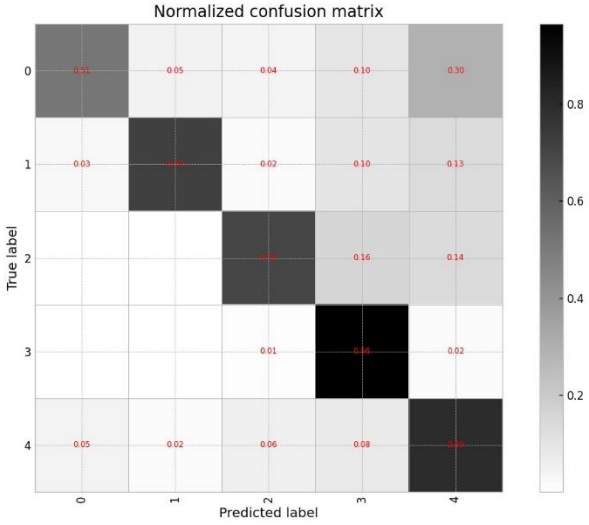



Table 18. Confusion matrix of ResNet50 with warm-up in full dataset

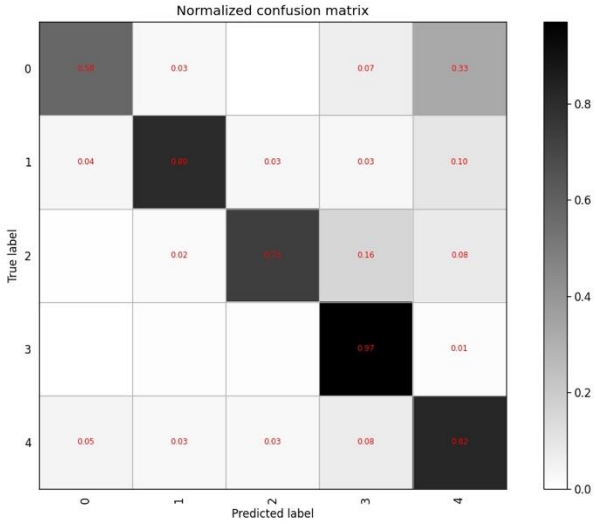



Table19. Confusion Matrix of ResNeXt50 (AdamW)