AFRICAN INSTITUTE FOR MATHEMATICAL SCIENCES
AFRICAN MASTER OF MACHINE INTELLIGENCE

# Cassava Disease Classification

*Mawada Omeralfaroug*, mahmed@aimsammi.org
*Randriatahina Andry Heritiana*, arandriatahina@aimsammi.org
*Vangelis Michael Oden*, omichael@aimsammi.org

Date: *May 10, 2020*

## 1. Presentation of the project

As the 2nd largest provider of carbohydrates in Africa, cassava is a key food security crop grown by small-holder farmers because it can withstand harsh conditions. At least 80% of small-holder farmer households in Sub-Saharan Africa grow cassava and viral diseases are major sources of poor yields. The goal is to learn a model to classify a given image into these 4 disease categories or a 5th category indicating a healthy leaf, using the images in the training data. This competition is part of the fine-grained visual-categorization workshop (FGVC6 workshop) at CVPR 2019.

## 2. Data Inference

### 2.1. Description of the data

In this competition, we introduce a data-set of 5 fine-grained cassava leaf disease categories with 9,436 labeled images collected during a regular survey in Uganda, mostly crowd-sourced from farmers taking images of their gardens, and annotated by experts at the National Crops Resources Research Institute (NaCRRI) in collaboration with the AI lab in Makarere University, Kampala. The data-set consists of leaf images of the cassava plant, with 9,436 annotated images and 12,595 unlabeled images of cassava leaves. Participants can choose to use the unlabeled images as additional training data. Here, we have 5656 labeled images that can be used for training and 3774 images for testing. As we can see from Figure 1, the data for training is imbalanced.

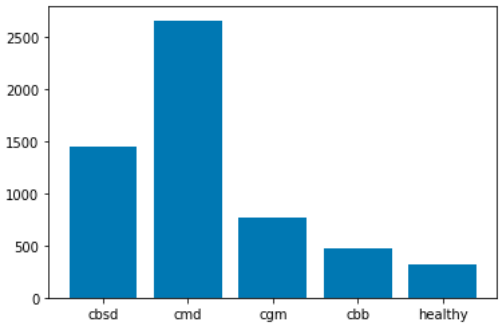


Figure 1: Distribution of the train set

### 2.2. Pre-processing the data

For the data pre-processing, we experimented using both *Global* centering for calculating and subtracting the mean pixel values across color channels, as well as *Local* centering for calculating and subtracting the mean pixel values per color channel. Both methods were tested per mini-batch and per training data. We eventually settled for per training data local centering because we noticed significant model performance when using the VGG-19 pre-trained model. For augmentation, we created a function for some custom data augmentation on the fly using PyTorch's augmentation. These included:

- RandomResizedCrop that takes an image and randomly resizes/crops it.
- RandomHorizontalFlip that takes an image and randomly flips it horizontally.
- RandomVerticalFlip that takes an image and randomly flips it vertically.
- RandomColorJitter that takes an image and makes brightness, contrast and saturation changes to it.
- Normalize that takes an image and normalizes it using the local mean and standard deviation gotten from the entire training data.

## 3. Model Inference

### 3.1. Description of the models used

In approaching model development for the task, we explored building our Conv-Net models for classifying, however, we had quite a lot of inaccuracies because we had limited computing resources to carry out large scale learning on the task, this then led us to testing transfer learning approaches. The first model tried was the VGG-19 pre-trained model, with two added fully connected layers trained with TanH activation functions. We arrived at quite impressive results as shown in Table [1], but noticed the model did not improve with additional layers and despite increasing the number of epochs and changing the batch size. Thus we decided to try the Res-Net family of models starting from the Res-Net34 model and finally the Res-Net152 model, each trained with 100 epochs. The performance of these models is shown in Table [1] with Res-Net50 as our champion model.

### 3.2. Training process and drawbacks

With each of our selected models, we used Stochastic Gradient Descent as our optimization algorithm and trained for 100 epochs. However, we noticed, just using a static learning rate (0.1, 0.01) did not yield good results for any of the models, we thus decided to continue the learning using a learning rate scheduler with decay properties. This helped with improving the performance of the models, helping us achieve our main goal which was to reach >90% accuracy in the public leader-board. We started our learning at 0.01 learning rate and decayed based on a factor of the current epoch using a batch size of 8, train split of 80%, and validation 20%. The training was done using Google Cloud Computing Virtual Machine instances equipped with Nvidia Tesla K80 GPU cards and 13 Gigabytes of RAM. This machine configuration allowed us to train 100 epochs for about 12 ($\pm$ 2) hours. During training on the various models, we had one major drawback which was *over-fitting*. We discovered this using our validation data and the result gotten from uploading the results to *kaggle*.

*Mawada Omeralfaroug,* mahmed@aimsammi.org
*Randriatahina Andry Heritiana,* arandriatahina@aimsammi.org
*Vangelis Michael Oden,* omichael@aimsammi.org

## 4. Results and Discussion

| Models | Acc@20 epochs | Acc@50 epochs | Acc@100 epochs | Test Acc.@100 epochs |
|---|---|---|---|---|
| Custom | 58.02 | 71.43 | 75.61 | 72.32 |
| VGG-19 | 75.66 | 82.51 | 85.71 | 83.21 |
| Res-Net34 | 87.14 | 92.51 | 94.98 | 87.40 |
| Res-Net50 | 87.32 | 94.62 | 98.57 | 92.19 |
| Res-Net101 | 88.49 | 94.33 | 97.02 | 91.11 |
| Res-Net152 | 88.21 | 95.65 | 96.84 | 90.27 |

Table 1: Model Accuracy

The results above show various pre-trained model performances at different epochs. Due to fluctuating internet availability, we check-pointed the training and resumed learning where there was a break in connection. This technique allowed us to save on time while training. We believe the success of the Res-Net50 model in this task is primarily due to the *Residual-block* and *Bottle-Neck* properties this model possesses. A residual block is a pattern of two convolutional layers with ReLU activation, where the output of the block is combined with the input to the block through shortcut connection and the residual modules use a bottleneck design with 1 x 1 filters to reduce the number of feature maps for computational efficiency reasons.

## 5. Conclusion

In conclusion, this project allowed us to put together concepts that were taught during the Computer Vision course and laboratories, as well as our own independent research to achieve the results we obtained. There are still a lot of improvements that can be done with respect to the data, the models and the training, some of which include;

- Semi-supervised learning on the extra-images to classify them into the existing train class labels and training yet another classifier with the additional data.
- Train a suitable classifier and predict on the extra-images using a high prediction threshold. Load the new images together with the old and train another classifier to predict on the test data.
- Train for longer (150 - 200) epochs and start learning rate from a higher value, while varying the batch size.

However, this is how far we can go and are quite glad we achieved what we set out to.

## References

[1] *iCassava 2019Fine-Grained Visual Categorization Challenge*. Mwebaze, Ernest and Gebru, Timnit and Frome, Andrea and Nsumba, Solomon and Tusubira, Jeremy. arXiv preprint arXiv:1908.02900, 2019

*Mawada Omeralfaroug,* mahmed@aimsammi.org
*Randriatahina Andry Heritiana,* arandriatahina@aimsammi.org
*Vangelis Michael Oden,* omichael@aimsammi.org