KAGGLE COMPETITION REPORT AMMI2022
CASSAVA LEAVES DISEASE

By Albert Agisha Ntwali and Catherine Monoue Konga
AMMI-2022(CATAL team)

I. Introduction [1]

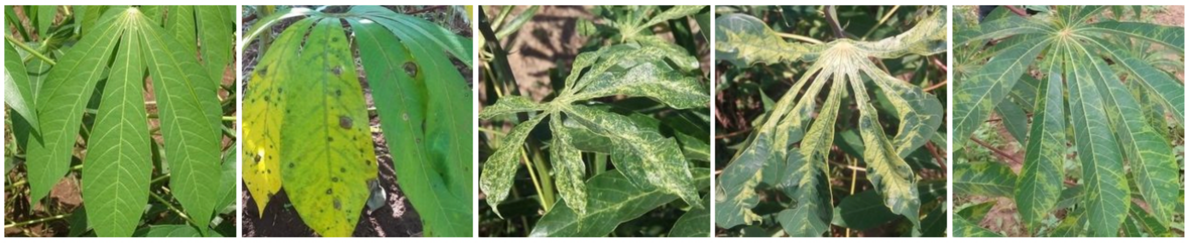
As the 2nd largest provider of carbohydrates in Africa, cassava is a key food security crop grown by small-holder farmers because it can withstand harsh conditions. At least 80% of small-holder farmer households in Sub-Saharan Africa grow cassava and viral diseases are major sources of poor yields.

In this competition, we introduce a dataset of 5 fine-grained cassava leaf disease categories with 9,436 labeled images collected during a regular survey in Uganda, mostly crowdsourced 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.

I.1. The Cassava disease leaf Data (https://arxiv.org/pdf/1908.02900.pdf)

The dataset consists of 9, 436 labeled and 12, 595 unlabeled images of cassava plant leaves. The annotations consist of 5 classes; healthy plant leaves (316/211 train/test examples) and diseased plant leaves representing the 4 diseases; CMD (2658/1773 train/test images), CBSD (1443/963 train/test images), CBB (466/311 train/test images), and CGM (773/516 train/test images).

I.2. Visualization of the classes:


(a) Healthy (b) CBB (c) CGM (d) CMD (e) CBSD

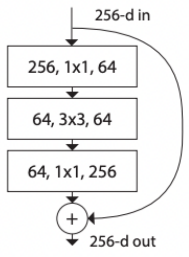
We have the class of healthy plants leaves, the diseased ones correspond on the 4 remaining classes represent disease (CMD, CBSD, CBB, CGM).

II. Model Used and Hyperparameters:

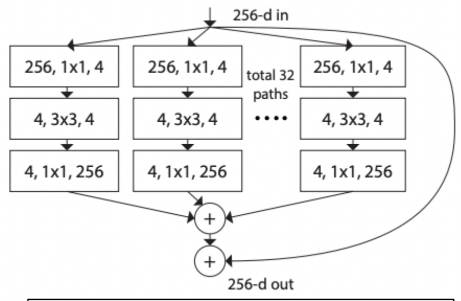
For our final result that led us to the position we hold in the competition, we used a pretrained model namely the *SE-ResNeXt101-32×4d.*

The SE-ResNeXt101-32×4d is a ResNeXt101-32×4d model with added Squeeze and Excitation model.

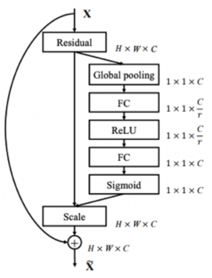
1. Architecture:



ResNet



ResNext with cardinality=32



SE-ResNet Module

2. Hyperparameters
   i. Default configuration[2]

The model uses SGD with momentum optimizer. Hyperparameters are Momentum (0.875), learning rate (0.256 for 256 batch size), label smoothing (0.1). The training was made for 90 epochs on ImageNet data. The model uses the following data augmentation:

[1] https://www.kaggle.com/competitions/ammi-2022-convnets/overview
[2] https://catalog.ngc.nvidia.com/orgs/nvidia/resources/se_resnext_for_pytorch

- For training: Normalization, Random resized crop to 224x224, Scale from 8% to 100%, Aspect ratio from 3/4 to 4/3, Random horizontal flip.
- For inference: Normalization, Scale to 256x256, Center crop to 224x224

ii. Used for the competition

For the competition, our best score (**public score**) was obtained by using the *AdamHD* optimizer (lr=1e-4, hypergrad_lr=1e-9), as criterion the *Cross-Entropy Loss*, we used *Classification model wrapping* for Test-Time Data Augmentation. The batch size=8, Number of epochs=20.
We used data augmentation for the model:

- For training: Random Rotation (30), Resize (550), Random Crop (500), Random Horizontal Flip (0.3), Random Vertical Flip (0.3), Random Erasing(0.1), Normalization
- For inference: Resize (550), Center Crop (500), Normalization

Actually, the best private score, we got was from the same model but with 10 epochs. We recall that early stopping is one of the solutions to overfitting.

III. Evolution on the scores and tried models:

Time	Models	Public score	Private Score	Description, Epochs, optimizer, batch-size
1st	from scratch	0.55827	0.57862	Convolution layers blocks, 10 epochs, Adam(lr=0.001),32
2nd	Resnet34	0.73973	0.76766	Pretrained,10 epochs, Adam(lr=0.001),32
3rd	Resnet34	0.84569	0.85777	Pretrained,10 epochs, Adam(lr=0.001),32
4th	ResNext101	0.91059	**0.91916**	Pretrained,10 epochs, AdamHD(lr=0.0001),8
5th	ResNext101	0.91324	0.91651	Pretrained,20 epochs, AdamHD(lr=0.0001),8

IV. Conclusion and Observations

The five presented models are the ones submitted on the platform for the competition, by the way we tried several models and we noticed that, to get better accuracy, one should take into account different technics to fight the overfitting but also the underfitting(start-notebook). Fixing hyperparameters at “good” scales, working on the data processing (splitting and transforming data: data augmentation), choosing good optimizer with its optimum parameters.

Using pretrained models is one of the key ideas from what we observed, one can build a model from scratch, as we did in the first case. Also, one can use pretrained model and make sure to optimize it for best solution. The challenging part, when deciding to work with a pretrained model was to get the sweetest model and to find interesting parameters for it.

Finally, one should make sure that the model is optimized, from the training and testing. As one can see in the above table, a model can perform well on a small data set (40% in our case), but wouldn’t do on the large. Training longer can work but not in all the cases. So, focusing on data processing and optimal hyperparameters is a good option to emphasize for the actual problem. Our best private score (**0.91916**) was gotten by using ResNext101 with a training time of 10 epochs, batch size=8, learning rate=1e-4, AdamHD optimizer.