AFRICAN MASTERS IN MACHINE INTELLIGENCE

(AMMI RWANDA, KIGALI)

Group Name: Covid-19 Date: May 10, 2020

Members:

Aimable Manzi ISHIMWE

Lloyd Acquaye THOMPSON

Michel MURWANASHYAKA

Report: Kaggle Cassava Disease Classification Competition.

1. **Overview**

   The Cassava Disease Classification Competition ran from 16th April, 2020 to 11th May, 2020. This competition is a partial fulfilment to the Computer Vision Course and as a pragmatic project to aid understanding. We competed with a 3-member group named "COVID-19", and achieved a final Kaggle Categorization Accuracy score of 0.90596 with 8 entries. Cassava is the most common provider of carbohydrates in Subsaharan Africa, which is grown by 80% of small scale farmers. Viral diseases that attack the leaves of cassava hinders food security and can be a cause for starvation and increase in poverty. The dataset is made up of images of cassava leaves with 9,436 annotated images, and 12,595 unlabelled images. The labelled images is futher grouped into 5 classes, namely; **CBB, CBSD, CGM, CMD**, and **HEALTHY**. The main aim of this competition is to train a model that will be able to classify the images into their respective labels by learning a mapping between "x-(image)" and "y-(labels)".

2. **Dataset Preprocessing and Data Augmentation:**

   We preprocessed the images in the dataset using transforms to resize, crop, flip and normalize. Then we used Scikit Learn to split dataset into train, validation and test sets. The shuffle is also set to "True" to ensure the model does not memorize the order of the data. Initially, upon visualization, we reliazed that the dataset is imbalanced with the CMD class having 47% of entire volume, so we tried the SubsetRandomSampler to data augment and generate a faily balanced dataset.

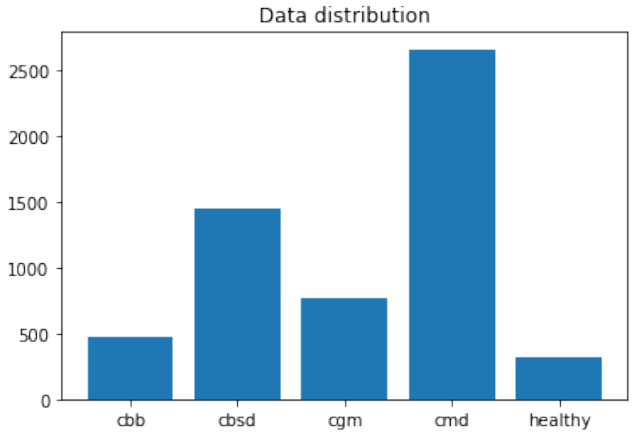
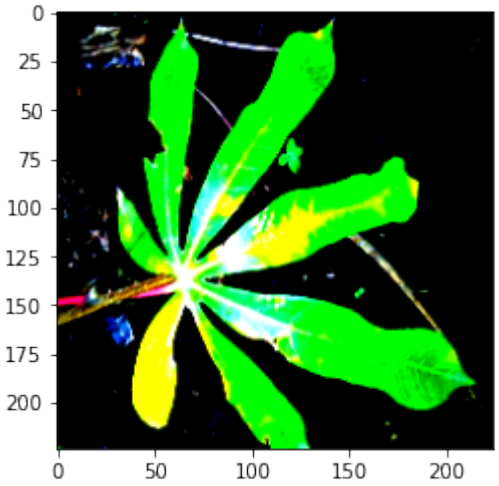


Figure 1: Bar chat for distribution of classes.

(a) Unnormalized Image



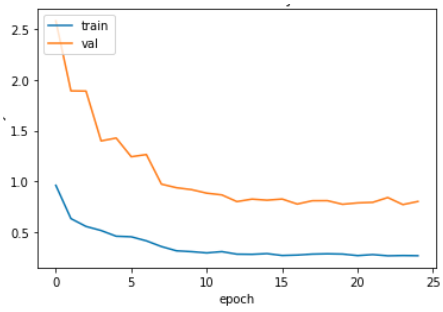
(b) Normalized Image.

3. **Model Architecture**

We started with a very simple VGG model architecture, but unfortunately it failed to generalized and we noticed undefitting since it did not perform well on both train and test sets. ResNet50, ResNet101 were the subsequent models that we used. Finally, we had to concatinate a number of models to get a better accuracy. Below is the documentation of the various models and their performance;

(a) Model Architecture.

N^o	Model Name	Batch Size	Score
1	VGG	64	0.59933
2	densenet121	64	0.57218
3	densenet161	64	0.72185
4	resnext101-32x8d(lock previous layer)	64	0.79139
5	resnet50	64	0.80662
6	resnet50(lock previous layer)	64	0.85165
7	resnet50	64	0.89072
8	resnet50 + model 6 + model 4	64	0.90596



(b) Loss curve.

4. **Conclusion**

In summary, we can say it is worth starting with a simple architecture and making it more complex by adding features and tuning parameters to achieve desireable results. We still feel there is more room for improvement to make the model perferm better, however, the overall accuracy of 0.90596 is a good way to start.

References

[1] https://arxiv.org/pdf/1908.02900.pdf

[2] https://www.kaggle.com/c/ammi-2020-convnets/leaderboard