Cassava Disease Classification

Haingoharijao Faniriniaina Ramandiamanana
Jama Hussein Mohamud
Nkosinathi Hlophe
(Group Name: No_fastAI)

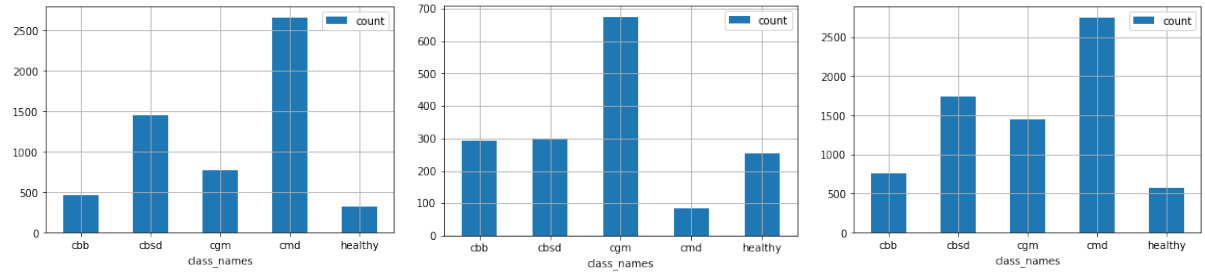
May 9, 2020

1 Introduction

Cassava still plays a crucial role in food security in Africa as it is a rigid plant that can handle hard climate conditions. However as most of vegetation, their leaves are victims of diseases that are mostly difficult to identify. The goal of this project is to identify a cassava leaf disease among 4 categories (or healthy) which are *cmd*, *cgm*, *cbsd*, *cbb*. For that we have $9,436$ labeled images collected during a regular survey in Uganda and $12,595$ unlabeled images. The data has been made available to us thanks to [1]. Of the $9,436$ labeled images only 5666 was made available to us. The section 2 of the report reviews the various preprocessing we used; section 3, utilized models are reviewed and finally, in section 4, the experiments and results are discussed.

2 Data preprocessing

The number of training examples in the trained data was not balanced as seen in figure 1 (a). The *cmd* class had more data points. To solve this problem a weighted sampler was used. Since there were no predefined training and validation dataset, 80% 20% train validation split was used. There were a number of transformation performed on the training data. Some of them are: random crop of 500×500, horizontal flipping, vertical flipping, random affine, rotation by 30 degrees and normalization. For the validation dataset only centre cropping of 500 and normalization were done on the images. We have also used the best performing model to do pseudo labeling on the unlabeled extra images. The extra images were added to the training data.



(a) *original* (b) *extra* (c) *extra + original*

Figure 1: *The distribution of the training images*

3 Methodology

We trained various models with transfer learning technique which were pretrained on *Imagenet* dataset. Some of the pretrained models we utilized include; *resnext50*, *resnext101* [2] and *densenet161* [3]. We used *CrossEntropy* as our loss function. Since training deep learning with different hyper-parameters is computationally expensive, throughout our experiments, we have used the same hyper-parameters (i.e. *batch_size* $= 16$, *lr*$= 2 \times 10^{-3}$, *optimizer* $=$ *ADAM*, *momentum* $= 0.9$, *weight_decay* $= 1 \times 10^{-3}$). We validated our models with Accuracy and Confusion matrix and our work was done in pure *Pytorch*.

4 Experiments and Results

Training deep learning models is challenging and requires a lot of time and experiments. We have done a lot of experiments with different models and obtained different performances. We realized the performance margin is not big among models though *resnext101* was better than others. However, averaging the predictions (*ensemble*) of the three pretrained models we used gave us the best performance on the public leadership board which is 0.88874. In figure 3, all the models took around 10 epochs to reach above 80% accuracy. The model performed exceptionally well on predicting class *cmd* and poorly - as compared to other classes - on *cbb* class (see figure 2).

Below are some of the techniques we tried but didn't lead improving performance with a big margin.

- We computed the mean (μ) and the standard deviation (std) of the original data.
- We tried pseudo labeling. The three models were used to do prediction on the extra images provided, then we averaged predictions. An image was given a label if the average prediction probability was above 0.95.
- We tried to add one more linear layer.
- We tried test time augmentation using four transformations.

Layer freezing and unfreezing was the best operation that improved our performance with a big margin across all models. For instance, we unfreeze the last layer in our best performing model (i.e. *resnext101*).

Figure 2: *confusion matrix on validation data*

(a) *densenet161* (b) *resnext101* (c) *resnext50*

Figure 3: *The accuracy of the models trained on 50 epochs*

5 Conclusion

In conclusion, our models performed fairly well, the performance we achieved is acceptable in such a noisy and imbalanced data. Possible things to try can be further data augmentation, test time augmentation (TTA), k-fold cross validation, careful hyper-parameter tuning. However, these operation come at their cost, and due to limited computation power it is hard to try every possible option. Other than that the project was fascinating in the sense that small techniques like we did can lead to a better performance. We hope, in the future, better techniques to be developed in training deep neural networks with less computational budget.

References

[1] Ernest Mwebaze, Timnit Gebru, Andrea Frome, Solomon Nsumba, and Jeremy Tusubira. icassava 2019fine-grained visual categorization challenge. *arXiv preprint arXiv:1908.02900*, 2019.

[2] Saining Xie, Ross Girshick, Piotr DollÃ¡r, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *arXiv preprint arXiv:1611.05431*, 2016.

[3] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.