Report Cassava challenge
Group name: Team_JOC

Charlesquin Kemajou Mbakam
Soh Ornella Lucresse
Joram Bakekolo

May 11, 2020

0.1 Introduction

Classification algorithms have been developed in symbolic learning, statistics, and neural networks, usually using different data modelling and representation techniques. Classification is here defined to be the problem of correctly predicting the probability that an example has a predefined class from a set of attributes describing the example. There exist many different algorithms, but their relative merits and practical usefulness are unclear. Thus, the need arises to evaluate their relative performances, in particular on large-scale industrial problems.

0.2 Problem Description

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.
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 Artificial Intelligence(AI) lab in Makarere University, Kampala.

The dataset consists of leaf images of the cassava plant, with 9,436 annotated images and 12,595 unlabeled images of cassava leaves. The unlabeld images will be used as an addition training data.

The goal of this work 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.

0.3 Data Description

The data consists of two folders, a training folder that contains 5 subfolders that contain the respective images for the different 5 classes and a test folder containing test images. The dataset contains the image in train folder name exactly matching the image name in the test folder and the corresponding class prediction with labels corresponding to the disease categories, cmd, healthy, cgm, cbsd, cbb.

The visualization of data leaded us to the following figure decribing the distribution over classes.

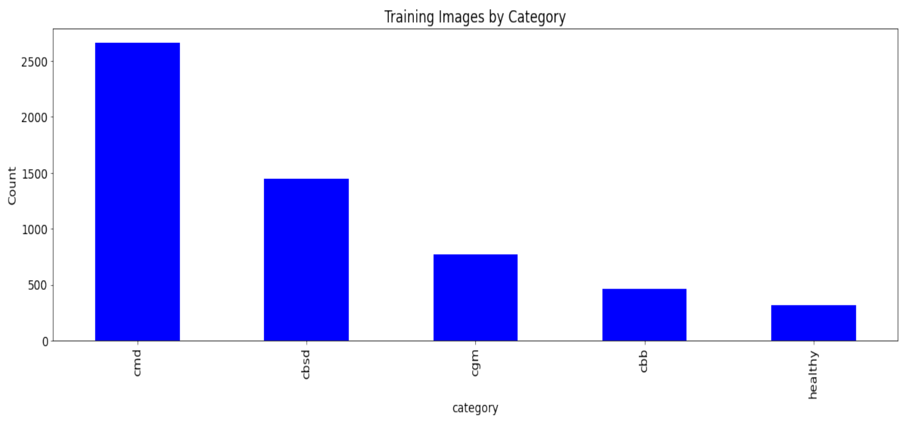

Figure 1: Data distribution

0.4 Method

Differents methods have been used to solved the problem which give differents significant results. In this section we will take turns explaining them.

0.4.1 Pytorch

1. **Imbalance data**
   After visualised data we observed the problem of imbalance data which, we addressed by upsampling the dataset.

2. **Data augmentation**
   We performed the preprocessing, we perform data augmentation.

3. **Pre-trained model**
   Then we imported a pre-trained model **ResNext50** and tuneed it according to the task we want to sole.

4. **Training loop**
We also defined the loss function that is the CrossEntropyLoss, the SGD optimizer. We train the model with learning rate 0.01, batch size of 64 and 40 epochs which end with 82% accuracy. then, Our prediction on kaggle gave an accuracy of 80%.

In order to improve our prediction, we decided to explore Tensor to build a machine for a good classification.

0.4.2 Tensorflow

1. **data preprocessing**
We first, split our dataset in two subsets: 80% for the training and 20% for validation. Secondly, we generated more data by doing data augmentation for the training and validation set.

2. **Pretrained Model**
The pretrained model used for this task was **ResNet50**.

3. **Training loop**
In the training loop, we used CrossEntropyLoss to track the loss and SGD optimizer to update parameters. The model was trained with a learning rate of 0.01, batch size of 20 and 10 epochs which end with 95% accuracy. then, Our prediction on kaggle gave an accuracy of 85%.

Unsatisfied with the above results, we decided to make used of FastAI to improve our result.

0.4.3 FastAI

In this framework, we start by setting the random seed. Then we defined the transformer for the model. We get the data with the transformer we just defined. We load the pre-trained model **ResNet50** and **ResNet101**. We then try to find the appropriate learning rate by plotting the curve. After we trained and saved the model. The above process is repeated for differents learning rate. At the end, we load the best model, use it to make the prediction and submittion.

In this section, we experimented with image classification using the high library deep learning Fastai which was built on top of Pytorch. This library simplifies training, by using modern practices. We make use transfert learning technique in order to build our model and this with the pretrained model ResNet50. To train our model, we first define a transformer as follow: do_flip= True, flip_vert= True, max_rotate= 180, max_zoom=1.1, max_lighting = 0.2 , max_warp = 0.2, p_affine = 0.75, p_lighting = 0.7, xtra_tfms = zoom_crop(scale=(0.5, 1.5), do_rand=True) [Fastai documentation].

The model was trained by applying the resizing where we started with 224*224 image size and batch size of 64. After, we resized images again at 256*256 with batch size of 32. Finally, we resized at 512*512 with batch size of 16. Knowing the impact of the learning rate in the training model, we were using the this important function *model.lr_find() and model.recorder.plot() to find the best learning rate and the interval where we can slice it. For the training epochs:

- First resizing 224*224: we start with 03 epochs then unfreeze the model and finally train on 04 epochs.
- Second resizing 256*256: We start by freezing the model, train on 10 epochs, unfreeze the model and finally train on 10 epochs.
- Third resizing 512*512: We start by freezing the model, train on 10 epochs, unfreeze the model and finally train on 10 epochs.

However, in order to make use of the extraimages, we did a prediction with extraimages and label all predictions with probability greater than 97% to augment our training dataset. Finally, train again the model by following the above training loop.

0.5 Conclusion

In conclusion, we have been able through this challenge to build many classifiers for cassava diseases while using Torch, Tensorflow and Fastai library. After training these models, we got in our private leaderboad 82%, 90% and .... for Torch, Tensorflow and fastai respectively. While in the public leaderboard we got 80%, 85% and ... for torch, Tensorflow and fastai respectively. It turns out that based on this specific problem, the model built with Fastai outperforms those built with Torch and Tensorfolw.

However the limitation of this study is that we were not able to make use of the unlabeled data while building models with Torch and Tensorflow. Our future perspective is to find out a good way to label unlabel while minimizing the error.