Cassava Leaf Disease Classification

Nicolas DUFOUR
ENS Paris Saclay
4 Avenue des Sciences, 91190 Gif-sur-Yvette
nicolas.dufourn@gmail.com

Abstract

The objective of this project is to be able to discriminate from 4 of the most common leaf disease that infect cassava crops. To do this, we will use the last fine grained visual classification techniques. Being able to discriminate disease with a picture taken from a cheap phone could help a lot farmers improve yields. We will introduce a novel approach to train models with noisy labels.

1. Introduction/ Motivation

Cassava is the main staple food for millions of peoples in Africa. Indeed, 80% of Sub-Saharan household farm grow this starchy root. Communities of farmers depend on this root to feed their families. Therefore, it's very important for them to identify diseases when they appear on crops so that it can be eradicated from the field. Using computer vision to detect the diseases could help improve the yield and therefore help with the food security of the household. Using a mobile phone, if we manage to make a model that can leverage the information given by a picture of the leaves, this can be an easy way for farmers to diagnostic their crops. Once the farmer have identified the crop, he can proceed to give the plantation the adequate treatment. A misclassification could lead to the wrong pesticide being used and cause financial loss for the farmer [10]. The farmer also risks the spread of the disease to all the field and loose his harvest

2. Problem Definition

We are going to base this study on the dataset provided by the Makerere University AI Lab for the Cassava Leaf Diseases competition[2]. This dataset present a few challenges. First as most of fine grained datasets, each disease has only a few thousand samples. We are going to train our architecture having in mind a data efficiency strategy. To do this we are going to use transfer learning over pretrained state of the art models. We will also develop a data augmentation pipeline to manage to get the most out of our data avoiding model overfitting

The dataset also suffer from noisy labels. We are going to have to tackle this problem to train a model under a noisy labels paradigm. Indeed, The dataset has been labeled by one field expert. However, even for expert, diseases aren't easy to distinguish from a picture The dataset have been collected by the farmer themselves using their smartphone. Some picture suffer from blurriness or bad quality which can avoid in label noise. However, it is key to deal with this bad quality data since the applications in the field are going to have the same bad quality challenges.

For this project, we are going to try and exploit the last advanced in image recognition. We are going to use transformer based models. Transformers have been a revolution for a lot of fields. Indeed, it has seen the best progress in the NLP field replacing almost any model. Recently, Transformers have progress in new fields such as Computer Vision. Image transformers have emerge in the past year and have beaten the state of the art on the Imagenet dataset.

3. Related Work

3.1. Vision Transformer (ViT)

The vision transformer [5] (ViT) is the first model to remove Convolutional layers from the architecture of the model. From the beginning, the model use a Transformer. A transformer is a model that uses an attention layer to help understand sequences. That way, our neural network can learn complex relationships from the sequences. Therefore, a sequence is needed as an entry to the model. However, images are 2D objects which doesn't have a natural order. We could consider the row-wise sequence of pixels as an entry. However this would lead to millions of elements sequences and at the moment, transformers aren't quite good to handle long sequences. Indeed, the complexity of the transformer is quadratic regarding the sequence length. Also, we would loose a lot of local information (nearby pixel in 2D aren't near in 1D).

To adapt this, the model transform an image into a sequence of 16x16 patches. To try and keep positional infor-

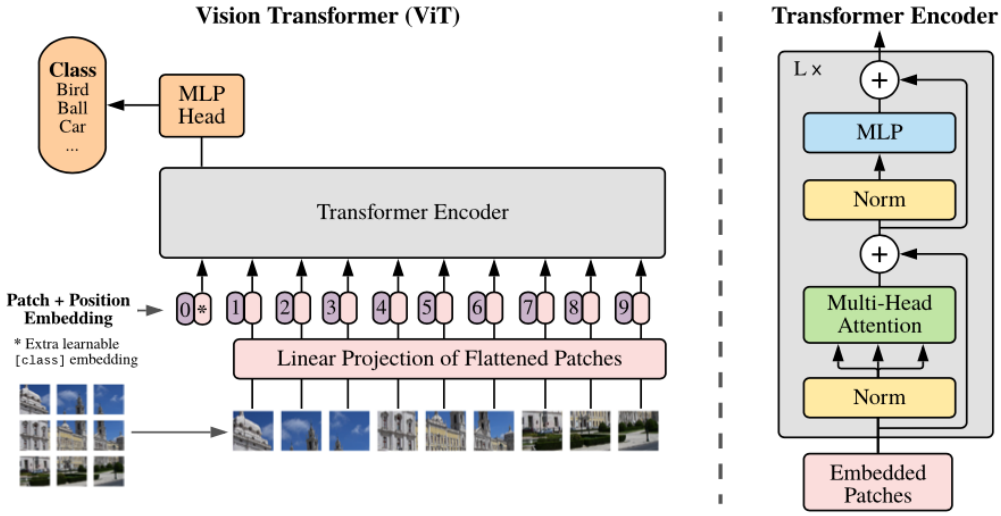

Figure 1. The Vision Transformer Architecture. Figure taken from [5]

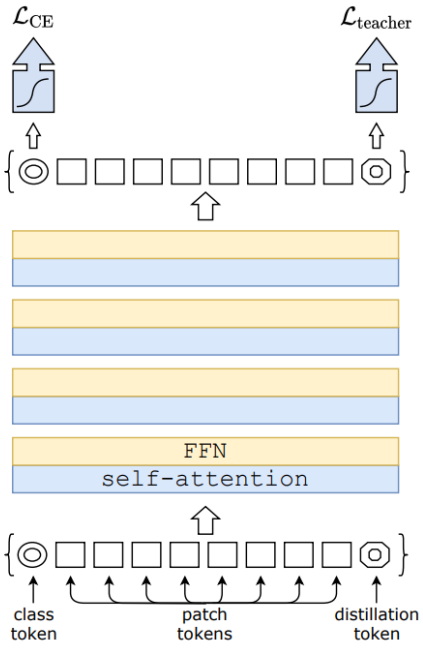

Figure 2. The Data Efficient Image Transformer. Figure taken from [17]

mation, each element of the sequence is associated a positional embedding. See Figure 1 for more details.

This model has the characteristic to have been trained on a huge private dataset owned by Google, the JFT dataset [15] with 303 millions images and 18k different classes. We however can use the pre-trained weights that have been made available by the Timm[18] library.

3.2. The Data Efficient Image Transformer Model

Building upon the works of the Vision Transformer, the authors of the Data Efficient Image Transformer[17] (DeiT) propose an image transformer model that is 100 time more data efficient than ViT. Indeed, they succeed to train the model by "only" training it on ImageNet [4] and not using some special massive dataset. They succeed to to this using knowledge distillation. See Figure 2

They train the network in a teacher student fashion using a Convolutional Network as a teacher. The idea is to teach a student network based on the teacher output. If we call ϕ_T the teacher output and ϕ_S the students output, we denote y the true label and $y_T = argmax\phi_T)$, we have the following loss

$$loss = \frac{\mathcal{L}_{CE}(\phi_S, y) + \mathcal{L}_{CE}(\phi_S, y_T)}{2} \quad (1)$$

With $\mathcal{L}_{CE}$ the cross-entropy loss. The data efficiency of the model make it a good candidate for transfer learning tasks. The authors have pretty good performances on multiples fine grained dataset such as Stanford Cars [9] or iNaturalist[8]

3.3. Learning With Noisy Labels

To tackle the noisy labels problem, we base on the work of [3]. They prove some theorical results when learning with noisy labels. They prove the accuracy metric is robust to noisy labels and that evaluating a model on a noisy validation set yields good results.

Finally, the recommend using a teacher student network to try to learn under noisy distributions of labels. The teacher student paradigm was introduced by [7]. The idea is that the teacher network is going to filter the label distri-

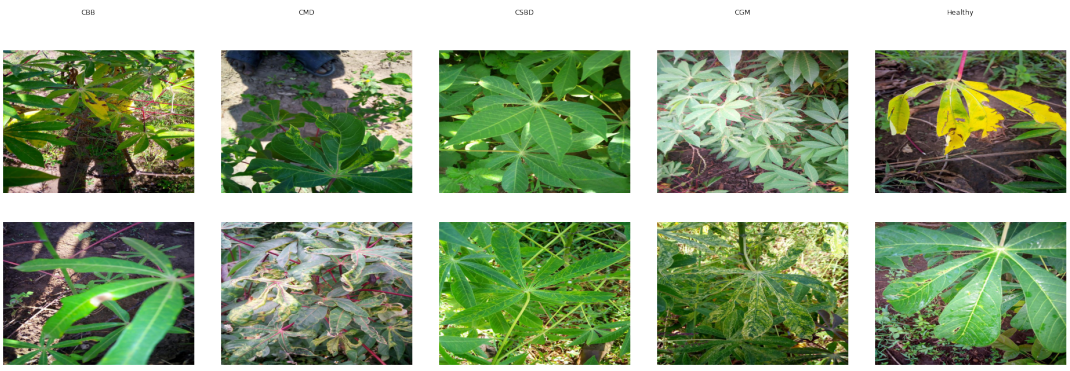

Figure 3. Samples for each class over the dataset

bution for the student and allow the student to learn a more reliable representation of the data.

4. Methodology

4.1. The Dataset

The dataset comes from the kaggle competition *Cassava leaf disease classification*. It is composed of 21397 cassava leaf pictures. The pictures have been taken by Ugandan farmers using a low budget smartphone. The data has then been then annotated by a cassava disease expert. This pictures are 600x800 and have a lot of artifacts such as noise or blur. We use this low quality pictures because the goal is that farmers can take a picture of a crop that they think is sick and want to know what crop it is. The farmers aren't professional photographers and do not own high budget cameras. Therefore, it's important to use this noisy data and not a dataset that would have been gathered with a professional camera.

The labels represent the four more common cassava leaf diseases. We have Cassava Bacterial Blight (CBB) with 1087 samples, Cassava Brown Streak Disease with 2189 samples, Cassava Green Mottle (CGM) 2386 samples, Cassava Mosaic Disease (CMD) with 13158 and finally the healthy class with 2577. We can see that CMD is over represented over the dataset. Indeed, more than 60% of the data has the label CMD.

To tackle the data unbalance that is present we are going to use a data augmentation technique that we are going to specify. Data augmentation is going to allow to have as many different sample that we need for each class.

4.2. Data Augmentation

The principle of data augmentation is to augment the data that we have to try to mimic new data. Indeed, we have a few thousands samples for each class and if we only base our training the data that we have we could quickly overfit our models.

We need to run some data augmentation. For each picture, we run a succession of random functions. We then get an unique sample. The function we use are chosen from the following:Random rotation, Axial Symmetry, Shift, Scaling, Random Brightness, Random Contrast, Random Saturation, Random Crop.

We also add Coarse Dropout that consist in removing random squares from the image. This force the model to not overfit certain regions in images since it can be occluded on certain iterations.

We can see at Figure 4 that the data augmentation generate different samples. However we took care to avoid distribution drift by transforming to much the input (we don't alter the coloration for example, we don't expect to encounter blue cassava leaves in the test set). Data augmentation have to be crafted having in mind what does the data could look like.

4.3. Handling the Dataset Noise

I has appear that the dataset contains quite a bit of noisy labels. Indeed this dataset is quite challenging to label and even an expert makes mistakes. Some of the disease are quite similar and this induce some label noise.

Ideally, the best thing to do when encountering dataset noise would be to try and detect the noisy labels and relabel them. One technique could be to have other labelers and resolve the conflicting labels. However in this, case it's not possible. The kaggle dataset is fixed and the people able to label such dataset is pretty scarce.

Even if we could relabel the dataset, that wouldn't be beneficial for us since the test dataset could also be noisy. We need to be able to deal with the label noise without relabeling it. Indeed [3] show that training on a noisy dataset can still show good performances. They also prove that looking at a noisy validation set to evaluate a model still shows good results.

The model they propose is a teacher student model where the teacher predict the labels that the student has to learn. The idea behind this is that the teacher model acts like a

Figure 4. Data augmented samples

denoiser on the labels and we hope to give better labels.

We are going to modify the model such as the teacher model learn at the same time than the student model. That way, the teacher model gets better at predicting the label but also at teaching the student as we can see in Figure 5. The loss of the student impact both the student and the teacher. This approach were the teacher model improve with the student as recently been proved to be quite efficient [12] to train in an semi-supervised setup. Instead of training 2 separate models, we are going to train 2 different classifications heads. That way, we avoid blowing up our number of parameters and still be able to train our model on a single GPU. The first head is going to be the teacher head which will receive the true labels. The second head will be the student head which will learn the teacher labels.

This could mean that the student network would yield the best performances. This would make the architecture quite resilient to noise and encourage using a distillation process training a a neural network to be more resilient to label noise.

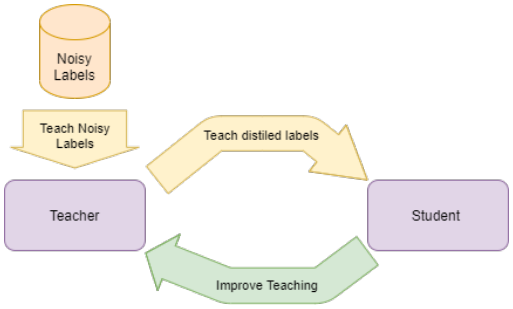

Figure 5. Data augmented samples

If we call ϕ_T the teacher output and ϕ_S the students output, we denote y the true label and $y_T = argmax(\phi_T)$, we have the now the following loss call the hard loss

$$loss_{hard} = \frac{\mathcal{L}_{CE}(\phi_T, y) + \mathcal{L}_{CE}(\phi_S, y_T)}{2} \tag{2}$$

We can also consider the soft version where we consider the teacher output (and not the predicted output) as the supervised signal. This could have the advantage to deal with model overconfidence. Indeed, Neural networks have been proved to be overconfident [11] on misclassification and an approach proposed by [11] called label smoothing consist in add noise in the label ground truth. Here, we use the natural uncertainty given by the teacher as such noise. We will then consider the following loss:

$$loss_{soft} = \frac{\mathcal{L}_{CE}(\phi_T, y) + \mathcal{L}_{CE}(\phi_S, \phi_T)}{2} \tag{3}$$

4.4. Ensemble Methods

To diminish the variance of our predictions, we are going to do an ensemble model. An ensemble model combine the prediction of multiples models to have a better prediction. The idea is that each model can learn something different than the other models and we want to be able to exploit this. For each model, we are going to do a cross validation. We split the dataset in K even chunks and we are going to learn K models. For each model, we select K-1 chunks on which we train the model. We then evaluate on the remaining chunk. We then keep our K models and use them as an ensemble model. For a data point x, we get for $k \in [|1, K|]$, $\phi_k(x)$ the output of model k which is the classification probability for each class. We then have:

$$\phi_{CV}(x) = \frac{1}{K}\sum_{k=1}^{K}\phi_k(x) \tag{4}$$

And the final prediction is given by

$$y_{CV} = argmax(\phi_{CV}) \tag{5}$$

We can also do more advanced ensemble techniques such as fitting a logistic regression classifier over the concatenation of all the output vectors, but we have observed in practice that it could lead to overfit the validation set.

Finally, if we dispose of multiple good models, we could do ensemble methods over the different methods. For example, we could try to ensemble a transformer model with a ConvNet model. Because of the architecture differences, this could lead to a However, we did not have the time to explore this option and has been left for future work

5. Experiments

5.1. Experimental Setup

We sample a test set from the training data. Indeed, the test set given by Kaggle doesn't have labels so it doesn't allow us to evaluate our methods.

We find that we have the best results using the AdamW optimizer and an learning rate scheduler called the One Cycle scheduler described in [13]. The idea is to have a warm-up phase were we train with a smaller learning rate that we increase gradually. We want to do this to avoid bias that can be induce from the first samples. Once we reach a maximal learning rate we slowly decrease the learning rate to allow for faster convergence.

To choose the learning rate we run a learning rate finder base on this paper [14] We find out that the optimal learning rate is 1.5e-5. We will tune our model with this learning rate in the future.

We train our models on Google Colab with a V100 GPU. To fit our model, we need a small batch size of 16.

For either the ViT model and the DeiT model, we use pretrained models. We choose to use the models that take as an entry a 384x384 entry since [16] shows that using a bigger resolution when fine-tuning allows to have better results.

We must disclose that we didn't do the experiments of the improvement of data augmentation and cross validation over DeiT since the weights of the model weren't available at that moment. However, since the architectures are quite similar, we conjecture that that results still hold.

5.2. Data Augmentation

We evaluate our data augmentation strategy over the baseline. We evaluate it on the Vit Model. The baseline only convert the image to a 384x384x3 tensor. The data augmentation strategy is described in Section 4.2.

Model	Test Accuracy
Baseline	0.878
Data augmentation	**0.888**

Table 1. Impact of data augmentation over a ViT Model

We see in Table 1 that we have an increase of 1 point using data augmentation. This is a great increase. Also, we must disclose that the baseline began to over-fit after 2 epochs and had to use early stopping to get the best possible results without any augmentation.

In the future we will always use this data augmentation pipeline.

5.3. Cross Validation and Ensemble Models

Now we want to evaluate the impact of cross validation and model ensembling on the performances of our model. We use a 5 Fold Cross validation.We then aggregate the prediction computing the mean distribution.

Model	Test Accuracy
Single Model	0.888
5 fold ensemble	**0.897**

Table 2. Impact of ensembling over a ViT Model

We have seen in Table 2 that we gain 0.9 accuracy points. This shows that using ensembling actually helps a lot improving the perforances. We will use ensembling for the rest of the models we build.

5.4. Model Selection

Now, we want to compare our 2 Image transformers models. We will compare the performances of both model using data augmentation and 5-fold cross validation. For DeiT we don't use the distillation signal to have a fair comparison between the two pretrained models.

Model	Test Accuracy
ViT	0.897
DeiT	**0.9035**

Table 3. Comparison of the 2 pretrained image transformer models

We see that DeiT have better performance than ViT. That can be explain by the fact that DeiT was built with data efficiency in mind that transfer into better fine tuning

5.5. Effect of Distillation

We are going to evaluate the impact of using knowledge distillation using the method proposed in 4.3 . We are going to compare teacher student supervision with hard and soft teacher labels. We will run this experiments on the DeiT model.

Model	Test Accuracy
No Distilation	0.9035
Hard Distil Student	0.9053
Hard Distil Teacher	0.9035
Hard Distil Both	**0.9053**
Soft Distil Student	0.8997
Soft Distil Teacher	0.9016
Soft Distil Both	0.9007

Table 4. Performance of Teacher Student training

We perform both a hard teacher and a soft teacher. We can see in Figure 4 that the hard teacher has better performances than the soft teacher. The student net have the best performances. We will keep the student network as our best model.

6. Conclusion

In this work, we have leveraged the advanced in computer vision to achieve good performances on classifying cassava leaf diseases. This results could help farmers have a cheap an easy way to quickly diagnostic diseases. This could lead to an increase on yields and decreasing the use of pesticides. Indeed, if the farmer is confident on which treatments to apply, he's not going to apply multiple treatments.

However, this work was done in the context of a Kaggle competition. The goal here is to maximise the accuracy and we don't dive in depth on the performance of the model over each disease. Further work could investigate the per class performances of our model. Also, we use a big model that could have trouble scaling for an on device use. Indeed, farmers from remote locations might not be able to upload pictures to a powerful server. A big improvement could be to consider the problem on a low ressources context. Methods such as pruning could be used to try to compress our model.

On the technical side, we have exploited the Image Transformers. We are at the beginning of the development of such models and the models are going to evolve quickly. The work of[17] show that it is possible to train transformer models having in mind a data-efficiency constraint. According to our results, pretraining trying to make the most of the data leads to better performances on downstream tasks. We have also showed that data augmentation is key for finetuning this big models. If we don't fine tune, we suffer from overfitting quickly. Data augmentation also leads to better performances.

Model assembling was key to get the best predictions in our study. This is great for Kaggle competitions. However, in a real world setup, this might be removed since it increase the inference time by the number of folds we use. It also increase the storage need since we need to store as much models as folds. A future work direction could consist on exploring ensembling over different models and particularly with Convnets models. Indeed, the architecture differences could bring new insights on the data. We could also have a wider cross validation (K=10) and only keep the best models to do the ensembling.

Finally, we improved the training of our model using a teacher student training setup. The teacher allow to smooth the noisy labels so the student can learn a cleaner distribution. The approach we introduce consist of considering a teacher/ student head setup where we train the teacher on its own performances but also on the model performances. This leads to a teacher that improve on training the student. The student is therefore better in this setup.

7. Code Details

We implemented our models using Pytorch Lightning [6]. This allow us to easily train our models

We use Weights and biases [1] to do experiment tracking which help us organize and keep track of our experiments.

The code can be found at `https://github.com/nicolas-dufour/cassava-leaf-disease`

References

[1] Lukas Biewald. *Experiment Tracking with Weights and Biases*. Software available from wandb.com. 2020. URL: `https://www.wandb.com/`.

[2] *Cassava leaf disease classification*. 2021. URL: `https://www.kaggle.com/c/cassava-leaf-disease-classification`.

[3] Pengfei Chen et al. *Robustness of Accuracy Metric and its Inspirations in Learning with Noisy Labels*. 2020. arXiv: `2012.04193 [cs.LG]`.

[4] J. Deng et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR09*. 2009.

[5] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2020. arXiv: `2010.11929 [cs.CV]`.

[6] WA Falcon. "PyTorch Lightning". In: *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning* 3 (2019).

[7] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. *Distilling the Knowledge in a Neural Network*. 2015. arXiv: `1503.02531 [stat.ML]`.

[8] Grant Van Horn et al. *The iNaturalist Species Classification and Detection Dataset*. 2018. arXiv: `1707.06642 [cs.CV]`.

[9] Jonathan Krause et al. "3D Object Representations for Fine-Grained Categorization". In: *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*. Sydney, Australia, 2013.

[10] James Legg and Elizabeth Alvarez. "Diseases affecting cassava". In: Aug. 2017, pp. 213–244. ISBN: 9781786760043. DOI: `10.19103/AS.2016.0014.10`.

[11] Gabriel Pereyra et al. *Regularizing Neural Networks by Penalizing Confident Output Distributions*. 2017. arXiv: `1701.06548 [cs.NE]`.

[12] Hieu Pham et al. *Meta Pseudo Labels*. 2021. arXiv: `2003.10580 [cs.LG]`.

[13] Leslie N. Smith. “A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay”. In: *CoRR* abs/1803.09820 (2018). arXiv: 1803.09820. URL: http://arxiv.org/abs/1803.09820.

[14] Leslie N. Smith. “No More Pesky Learning Rate Guessing Games”. In: *CoRR* abs/1506.01186 (2015). arXiv: 1506.01186. URL: http://arxiv.org/abs/1506.01186.

[15] Chen Sun et al. *Revisiting Unreasonable Effectiveness of Data in Deep Learning Era*. 2017. arXiv: 1707.02968 [cs.CV].

[16] Hugo Touvron et al. “Fixing the train-test resolution discrepancy”. In: *CoRR* abs/1906.06423 (2019). arXiv: 1906.06423. URL: http://arxiv.org/abs/1906.06423.

[17] Hugo Touvron et al. *Training data-efficient image transformers distillation through attention*. 2021. arXiv: 2012.12877 [cs.CV].

[18] Ross Wightman. *PyTorch Image Models*. https://github.com/rwightman/pytorch-image-models. 2019. DOI: 10.5281/zenodo.4414861.