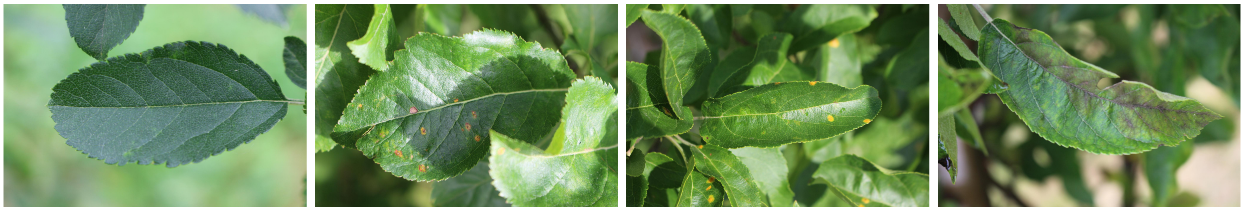
Plant Pathology Diagnosis

Youssef Allouah, Paul Calot

Summary



(a) Healthy (b) Multiple diseases (c) Rust (d) Scab

Figure 1: Examples from each class

1 Introduction

1.1 Context

This project is based on a scientific publication and a Kaggle competition that took place from March 9 to May 26, 2020 [1], [2].

1.2 Description of the problem

Misdiagnosis of diseases that spread in fields can lead to misuse ofchemicals resulting in:

- Greater resistance of pathogens ;
- Increase in production costs ;
- Decrease in economic profitability ;
- Destruction of the environment.

Human detection of these diseases consumes a lot of time and resources. In addition, a computer vision approach with basic models is not sufficient to take into account the large variance of leaves (age, genetics, symptoms etc.).

In the specific case that interests us, it is a question of developing a resilient solution to a problem of classification has four classes: healthy, multiple diseases, rust and scab. Figure 1 shows an example of each class.

Initially, 1821 high resolution images (2048×1365) are available for training. The distribution of these images between the different classes is given by figure 1.

1.3 General idea of the solution

The training dataset of 1821 images shows some variety within the same type of leaves: symptoms may vary from leaf to leaf, age also creates differences as well as exposure to the sun, the presence of blur or that of other leaves etc. However, the basic models of neural network cannot easily accommodate this diversity.

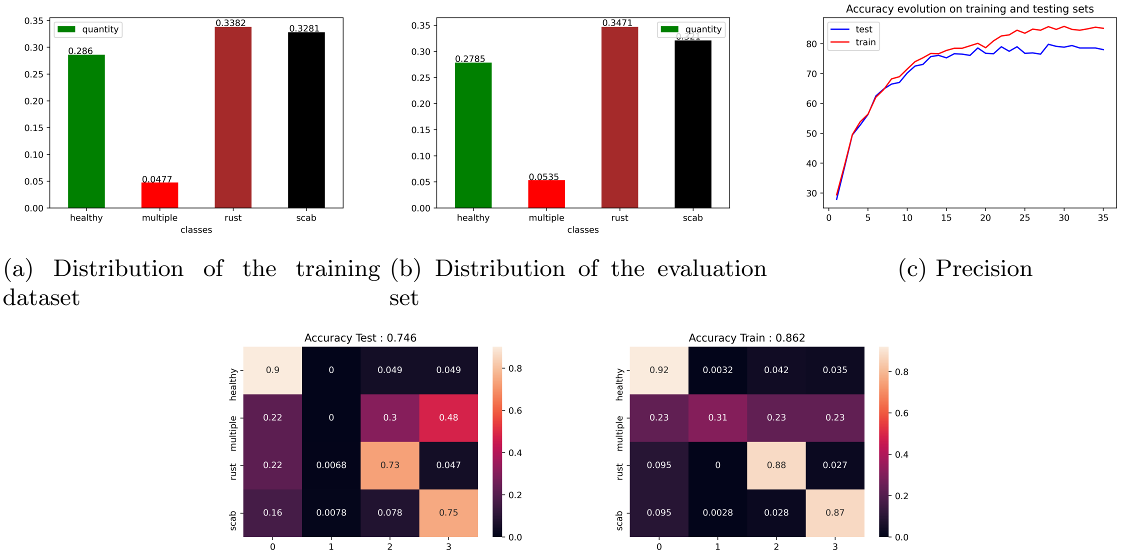
The precision of the details that must be recognized by the neural network requires taking an architecture rather deep. However, the small amount of data (1821 images) prevents us from training too many new weights, which undermines the discovery of features relevant for classification.

Our solution therefore mainly presents the following blocks::

- Augmenting the dataset ;
- Reuse of pre-trained architectures and comparison.

Healthy	Multiple diseases	Rust	Scab
517	92	622	590

Table 1: Initial distribution of classes



(a) Distribution of the training dataset (b) Distribution of the evaluation set (c) Precision

(d) Confusion matrix - Evaluation set (e) Confusion matrix - Training set

Figure 2: Results on ResNet-50 with all layers frozen and for 35 epochs. For the matrix of confusion, the real classes are noted on the ordinate, the sum on a line of probabilities is 1.

2 Description of the implementation

2.1 Augmentation of the dataset

2.1.1 Justification and result of the data augmenting

In order to justify the need to increase the data, let us take an interest in the pre-trained and to its performance on the initial data set. When all the layers are frozen and that we only add a dense layer allowing to go from 1000 classes to only 4, we get the results of the figure 2. Note that 60% of the data was used for training.

We thus see a limitation of precision as well as the beginning of over-learning. Forto counter this, an increase in the data set was carried out with the creation of about fifteen newdatasets of which the best are presented in table 2. Each of the images of these games is at theresolution $224 * 224$.

Note that the low percentage of data for the class "multiple diseases" led to the creation of additional data sets restoring a certain balance. The confusion matrix in figure 2 shows a much lower percentage of true positive for class 1 than in the evaluation dataset. This difference is less visible in the training game probably because of over-learning.

These datasets were created as follows:

1. Separation between the training game and the evaluation game;
2. Augmented training set.

The use of these datasets, as well as their strengths and weaknesses, will be discussed in the next part.

Dataset	random	image ops	noise	pixels ops	% separation	$\alpha_{1,3,4}$	α_2	total
0					60	1	1	1092
7	x	x	x	x	60	1	1	1092
5	x	x	x	x	77	1	6	1740
6	x	x	x	x	60	2	3	2238
4	x	x	x	x	77	2	10	3344
8	x	x	x	x	77	4	7	5804
10	x	x	x	x	77	4	12	6144
9	x	x	x	x	77	6	10	8672
11	x	x	x	x	77	6	16	9080
3classes	x	x	x	x	77	3	0	3994

Table 2: Augmentations performed. Random qualifies the randomness of these changes on the game of data. The following three columns qualify in the order of the modification of the image (cropping, rotations), the addition of Gaussian noise and finally the modification of the pixels (brightness and contrast). The probabilities of augmentation by any of these means are high in order to allow maximum variation of one augmentation over another. The size proportion of the initial data set used is then given for training, the augmentation factor for the two class groups and finally the amount of final data available for training.

An example of an augmentation on a leaf picture of the multiple disease class is given in figure 3.

2.1.2 Difficulties encountered

The difficulties we encountered in establishing these data sets are multiple:

1. We initially augmented the whole dataset before separating between training data and evaluation. In this case, two images from the same original image can be found in the training game for the first and the evaluation game for the second. This biases the precision during the evaluation and can lead to believe in a good model when there is a massive over-fitting;

2. Among the first datasets to be augmented was one that was increased by a factor of 14. This caused a bias, since the variance of the data set decreased. Indeed, our increase algorithms are unable to give, from a single image, fourteen sufficiently different images. We will see that this can be true for lower values of the augmentation factor (depending on the architecture used);

3. Finally, class imbalance remains a problem and will be discussed in more detail later.

2.2 Choice of architecture

To choose our architecture, we gave ourselves a certain number of pre-trained architectureson the ImageNet dataset: VGG[3], ResNet[4], GoogLeNet[5], EfficientNet[6].

We followed the following process for the choice of architecture:

- Test the architecture with a freezing level f;
- Produce the confusion matrix and visualize the gradient flow;
- Retest with a freezing level f + 1.

The freezing levels tested are: unfreezing of the last layer, unfreezing of dense layers, unfreezing of a third layer, freezing of the first two layers only, total freeze. We gradually freeze in one direction towards the inputs. The freezing levels are given below:

- Freezing 1: Unfreeze the last layer alone

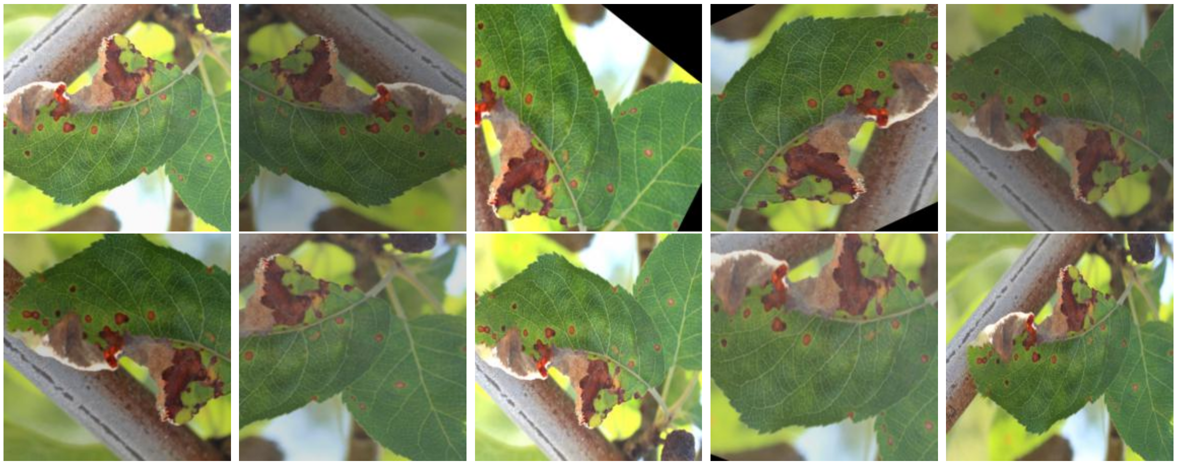


Figure 3: Example of an increase performed on an image of the multiple diseases class. The last imagecorresponds to the original only converted to the resolution $224 * 224$.

Architecture	Freezing 1	Freezing 2	Freezing 3	Freezing 4	Freezing 5	#Parameters (millions)
VGG-16	55	62	32	X	34	138
GoogLeNet	75	53	69	83	79	6
ResNet-50	80	74	73	28	82	25
EfficientNet-b3	66	67	51	X	32	12

Table 3: Tests with different gel levels. This table gives, for each gel level, the precision obtained on the test data set. The number of parameters is also given.

- Freezing 2: Unfreeze all the dense layers
- Freezing 3: Unfreeze one third of the layers
- Freezing 4: Freeze the first two layers alone
- Freezing 5: Total unfreeze

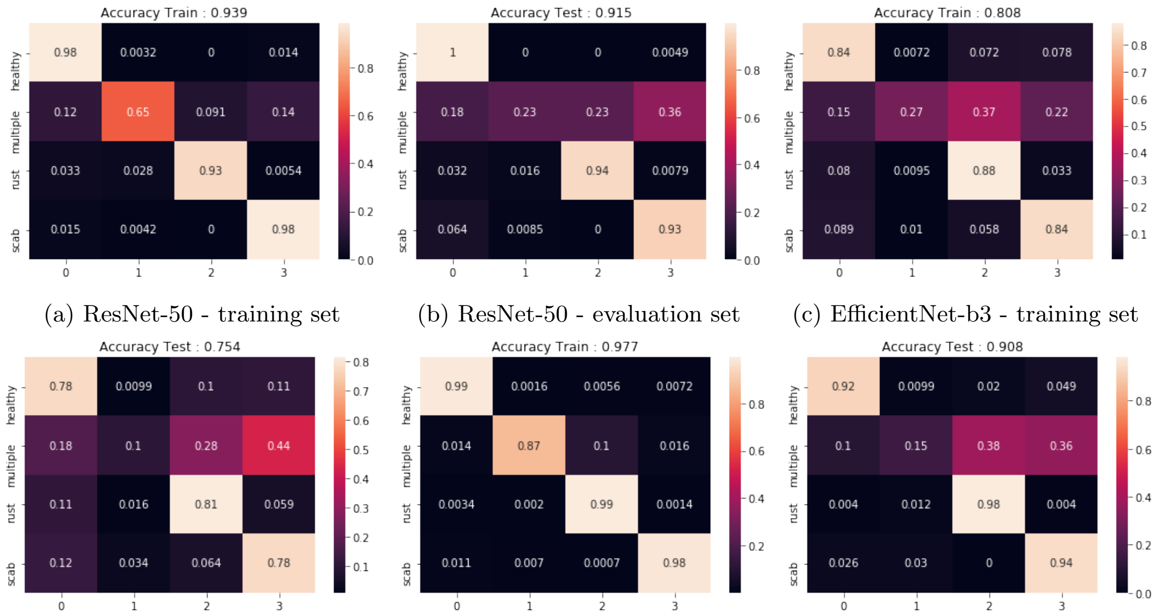
We give in table 3 a comparison of the models at the end of these tests on set 6 (which is medium size and unbalanced, see table 2).

We then compared the architectures[1] frozen at level 5 based on their performance on a larger and more balanced dataset (Dataset 8 visible in table 2) :

ResNet-50 and GoogLeNet show an accuracy of 0.915 and 0.908 respectively. EfficientNet-b3 has an accuracy of 0.754. On the other hand, the precisions are now so high that they can no longer be used as the sole metric. We have therefore produced confusion matrices visible in figure 4. We see clearly in Figure 4 that the models have poorly learned the characteristic features of class 1 (multiple diseases). We therefore decided, after seeing that rebalancing the classes by increase did not succeed, to balance the contributions to the learning error in an inversely proportional way to the size of each class. Thus, an error on class 1 will be much more penalizing than an error in another class.

After doing this work, ResNet-50 displays 0.842 precision and EfficientNet-b3 displays 0.918 precision. Here again are the confusion matrices in figure 5. Details in learning phase are generally better before the adaptation of the error coefficients. However, despite its high overall accuracy, EfficientNet shows poor performance on the prediction of class 1 with respect to ResNet which happens, almost once in two, to detect correctly the images of this class. The best architecture seems to be ResNet-50.

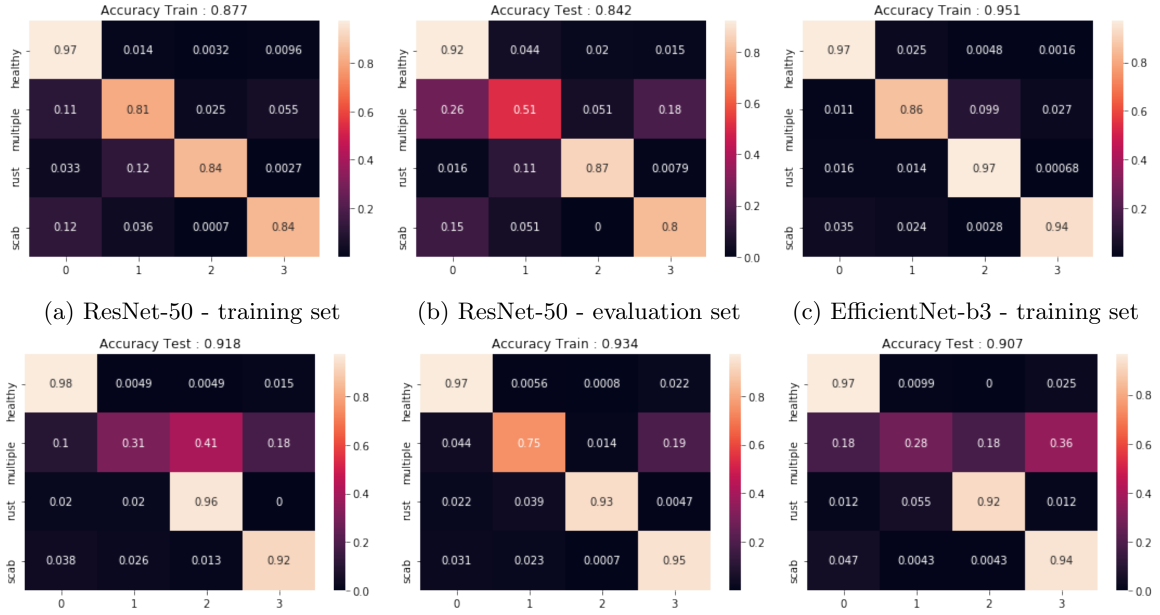
[1] We do not show the results for the VGG-16 architecture as they are unsatisfying due to the too low precision relatively to



(a) ResNet-50 - training set (b) ResNet-50 - evaluation set (c) EfficientNet-b3 - training set

(d) EfficientNet-b3 - evaluation set (e) GoogLeNet - training set (f) GoogLeNet - evaluation set

Figure 4: Confusion matrices before adaptation of the error coefficients. We note that the class 1 is very poorly learned by the two models. The choice is almost random. It is also useful to note that ResNet and GoogLeNet over-train on class 1 but do not generalize correctly on the evaluation game.



(a) ResNet-50 - training set (b) ResNet-50 - evaluation set (c) EfficientNet-b3 - training set

(d) EfficientNet-b3 - evaluation set (e) GoogLeNet - training set (f) GoogLeNet - evaluation set

Figure 5: Confusion matrices after adaptation of the error coefficients. Details in learning phase are much better before the adaptation of the error coefficients. However, despite its high overall accuracy, EfficientNet shows poor performance on prediction of class 1 relative to ResNet which manages, almost half of the time, to correctly detect the images of this class.

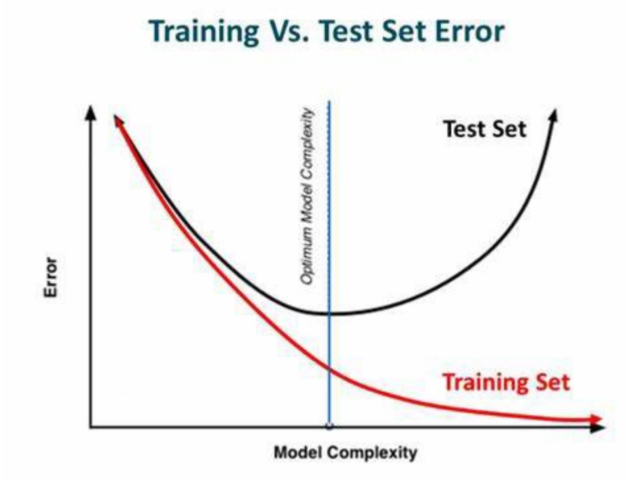


Figure 6: Figure showing the Early Stopping technique on learning curves. Credits: `https://medium.com/ai%C2%B3-theory-practice-business/dropout-early-stopping-188a23beb2f9`

2.2.1 3-classes model

The leaves of the Multiple diseases class group together both the characteristics of the Rust class and Scab. Thus, if such a leaf is given as input to a model, the neurons activating for Rust(resp. Scab) will also be activated for this leaf. Class images are expected multiple diseases unfairly classified are in the last two classes and not in the first.

Based on this observation, we created an architecture, based on ResNet-50, this time classifying between three classes, the multiple disease class having been temporarily excluded. Once the training of this finished network, we added a new dense layer allowing to go from 3 classes to 4 classes and frozen the previous one. In addition, different weights have been assigned to each class: 1 for the Healthy classes,Rust and Scab and 5 for Multiple diseases. Adam optimization was used.

Then, during the training, we gradually unfrozen all the layers of the classifier in leading by the latter. The results are presented in section 3.3.

2.3 Choice of hyperparameters and analysis tools

We have optimized the following hyperparameters for our architectures: the number of iterations (epochs) and the learning rate. We also used two tools to assess the correction of our models: learning and validation curves, and gradient flow.

2.3.1 Optimization of the number of epochs

e used the Early Stopping technique. This technique consists of keeping a backup of the model with minimal validation loss on training epochs. In addition, this technique causes the training process to stop, with a patience parameter, as soon as the validation loss no longer evolves. Thus, we stop learning at a level beyond which we would overfitting. The technique is illustrated in figure 6.

2.3.2 Optimization of the learning rate

We used the pytorch-lr-finder module which implements the following paper "Cyclical Learning Rates for-Training Neural Networks "by Leslie Smith, [7]. The technique used consists of the path (exponential or

the number of parameters.

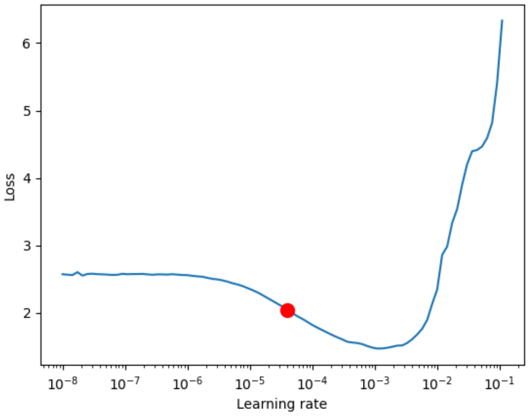


Figure 7: Figure showing the graph of the loss as a function of the learning rate. The best rateslearning are found in the region around the red dot. Credits: `https://pytorch-lightning.readthedocs.io/en/latest/lr_finder.html`

linear) of different learning rates, from the smallest to the largest, and to follow the evolution of the loss function over a few iterations (generally around 5 epochs). We then have,thanks to the modulus mentioned above, a curve similar to that of figure 7. On this curve, we choose as much as possible a point preceding the overall minimum and having the greatest possible slope. With this learning rate, we are sure to quickly converge towards the minimum of the loss.

2.3.3 Analysis tools

As mentioned previously, we used two tools to monitor and evaluate the correctness of our models.The first tool is the monitoring of learning curves and loss functions (see figure 8) which we used to assess the degree of overfitting or the quality of early stopping. The second is the visualization of the gradient flow (see figure 9) . The latter was particularly useful for checking whether there had been an explosion / vanishing of gradients.

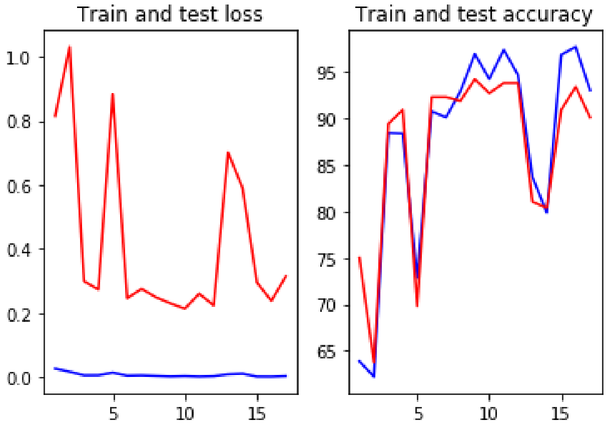


Figure 8: Figure showing the loss (left) and precision (right) curves during learning depending on the number of iterations (epochs). In red the validation curves, in blue the training curves.

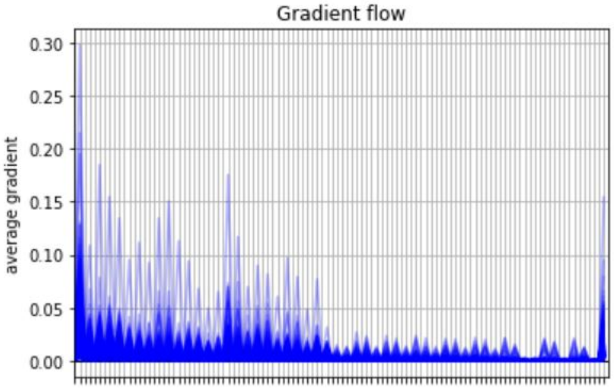


Figure 9: Figure showing the gradient flow through the different layers of the architecture. The greater the gradient, the faster the affected layer learns.

Architecture	Local precision	Precision on Kaggle
GoogLeNet (balanced error)	0.91	0.91
GoogLeNet	0.91	0.93
ResNet-50 (balanced error)	0.84	0.83
ResNet-50	0.92	0.93
EfficientNet-b3 (balanced error)	0.92	0.83
EfficientNet-b3	0.75	0.88

Table 4: Local and online precisions of our models

3 Results

3.1 Organization of results

Due to the small number of representatives of the ”multiple diseases” class, it is possible to obtain a good precision while ignoring this class. The choice as to the architecture is therefore made according to the objectives.Indeed, as the dataset suggests, leaves with multiple diseases are probably in low proportion in nature.

Thus, we can consider that if we already recognize precisely the classes Scab, Rust and Sound, then we treat the majority of cases and that is enough.

Conversely, we can consider that it is precisely the case of multiple diseases that is relevant here and we would then like to recognize it with precision. This objective will depend on the architecture. For these reasons,we offer two architectures: a balanced between the four classes and a second which seeks to maximize its precision. The results of the three-class ResNet-50 architecture are finally presented.

3.2 4-classes architectures

The results for architectures with 4 output classes are given in figure 10. On table 4, the details locally and on the validation game of the Kaggle competition are given. It is useful to notice that:

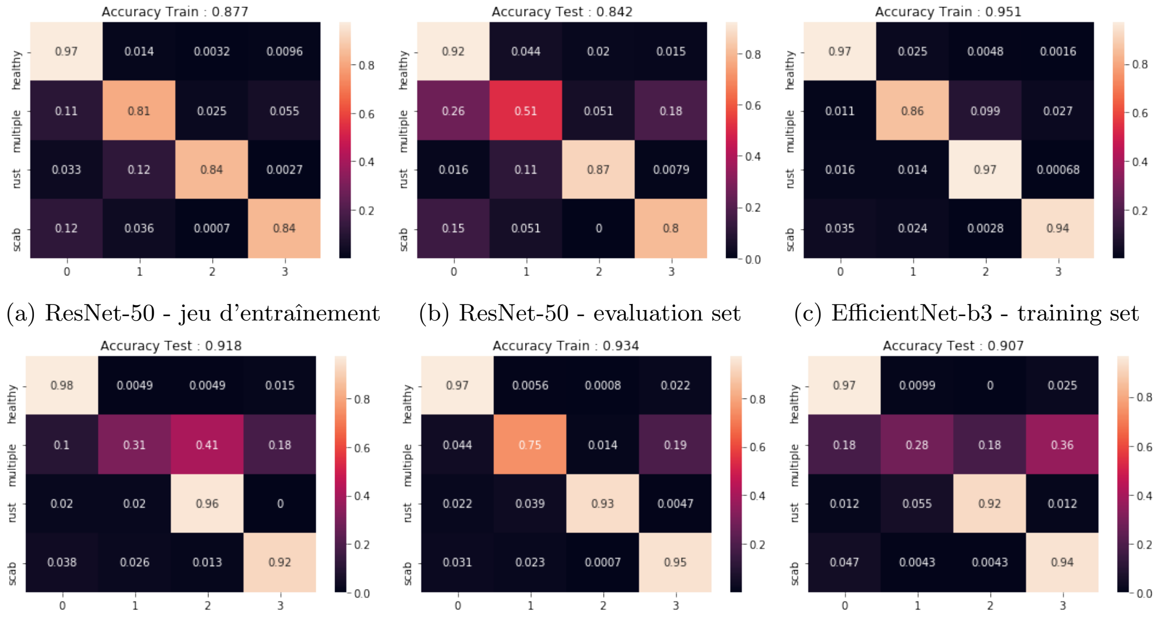
- Local and online precisions are very close, which suggests that our local datasets are of good quality and well representative
- We cannot know which of the two precisions is a better metric to evaluate the models.The fact remains that, locally, we have confusion matrices that tell us more than just Kaggle precision.
- We did not do any fine tuning because that was not the aim of the project. Precisions of the order of .93 are therefore very encouraging.

3.3 3-classes ResNet-50 architecture

For these results, we used the datasets called 3 classes and then dataset number 8. The results we obtained are given in figure 11.

They are not as conclusive as in the case of a classification on four classes. The precision of this model gives 82% on the Kaggle competition rating set.

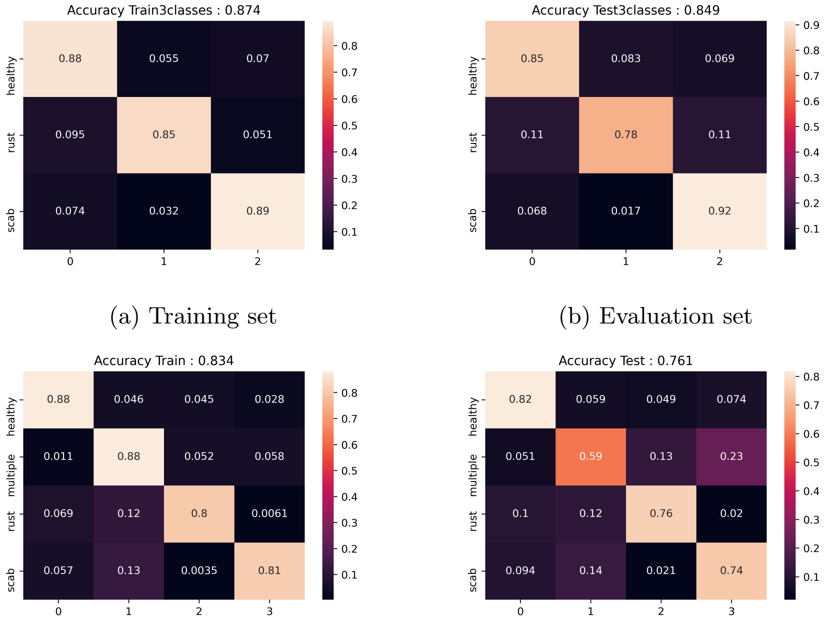
Overfitting on the multiple disease class is visible. We could probably improve the result using a dataset for which the increase on the multiple disease class is less consequent.



(a) ResNet-50 - jeu d'entraînement (b) ResNet-50 - evaluation set (c) EfficientNet-b3 - training set

(d) EfficientNet-b3 - evaluation set (e) GoogLeNet - training set (f) GoogLeNet - evaluation set

Figure 10: Results of 4-classes architecture models.



(a) Training set (b) Evaluation set

(c) Training set - 4 classes phase (d) Evaluation set - 4 classes phase

Figure 11: Results on ResNet-50 with all layers frozen and for 35 epochs. For the matrix of confusion, the real classes are noted on the ordinate, the sum on a line of probabilities is 1.

4 Conclusion and extension

One of the challenges of this project was the dataset and its imbalance between classes. So we have proceeded to increase the data.

Several architectures have been explored, freezing their layers to different degrees. Among these architectures, ResNet-50 and GoogLeNet give good results, both on local datasets and on the game competition data. With or without error adaptation, the results are different to appreciate:the details are better without adaptation of the errors, but the confusion on the under-represented classes are much less frequent, the model is thus more generalizable. During our research, we were led to implement classical methods of Machine Learning, in particular early stopping and adding weight in the loss function to compensate for the imbalance.

Improvements are possible. We have mainly carried out so far a benchmarking of architectures and datasets giving correct results for this problem. A second phase would consist in selecting the pair (dataset, architecture) which presents the most potential and in optimizing it as finely as possible. This could include the creation of a dataset tailored for the architecture, especially with problematic images removed. Indeed, a participant in the competition, for example, Kaggle noted the presence of the same image twice, classified in two distinct classes, in the dataset. We also tried to use pre-trained models on plant type recognition tasks, but certain technical difficulties prevented us from using them properly. In addition, ImageNet is sufficiently large and varied that models that are trained on its database are transferable, provided that it is thawed correctly.

References

[1] Ranjita Thapa, Noah Snavely, Serge Belongie, and Awais Khan. The plant pathology 2020 challenge dataset to classify foliar disease of apples, 2020.

[2] Kaggle - plant pathology 2020 - Link : https://www.kaggle.com/c/plant-pathology-2020-fgvc7/overview.

[3] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[5] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[6] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2019.

[7] Leslie N. Smith. Cyclical learning rates for training neural networks, 2015.