Healthy Plant Detection - Challenge 1 Report

Anne-Lys LANORE, Jakub JASTRZEBSKI, Maxime LAFONT, Ahmed HARRABI - Team Name: kuba

Introduction

This document presents the outcomes of the first challenge conducted in the Artificial Neural Networks and Deep Learning course for the academic year 2023-2024 by the group of Anne-lys Lanore, Maxime Lafont, Ahmed Harrabi and Jakub Jastrzebski (Team Name: kuba). Task of homework was a binary classification of images using a deep learning approach presented during the course. Dataset contained pictures of plants which were labeled with two tags: “healthy” and “unhealthy” (Fig. 1.).

Fig. 1. Example of a Healthy and a Unhealthy Pictures

The goal of the challenge was to predict the correct class label for a picture from the prepared test set.
Our group considered using a transfer learning approach with an application of fine tuning methods to obtain the best performing model. The final deep learning network that we used was the EfficientNet B0 network. Detailed description of the approach is presented in the following sections of this report: Data Preprocessing, Neural Network Architecture and Performance where we describe methods that we used for dataset preprocessing (outliers and duplicates removal and data augmentation), fine tuning of used network and results of the experiments on the CodaLab platform.
We based our work on the lectures given by Professor Boracchi and Professor Matteucci [1] [5] and Eugenio Lomurno laboratories [3]

Data Preprocessing

The provided dataset is composed of 5200 pictures of plants and 5200 labels. We began by normalizing the data by dividing it by a factor of 255 which also made it easier to display the pictures. Then we moved on to the inspection of the dataset in order to remove outliers and duplicates.

A. Removing outliers.

After inspecting the dataset, we very quickly noticed (thanks to the hint provided in the Logbook [4]) that it was infected with outliers, mainly two that are shown in Fig. 2. To detect them we used the MobileNet network, the dataset has been put as input of the MobileNet network. The two outliers were predicted by MobileNet as ’rock-beauty’ and ’comic-book’. Thanks to these predictions, we then removed all the pictures predicted as such from our dataset.

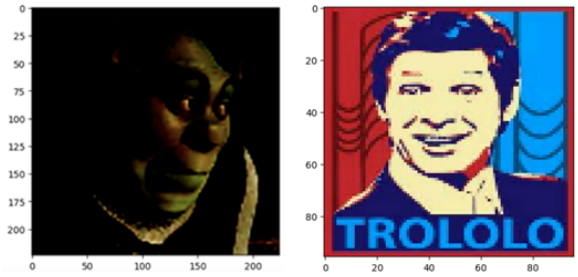

Fig. 2. Example of outliers

At this point, we wondered if some data were wrongly labeled. Indeed while reviewing the dataset, we noticed that some pictures labeled as healthy were showing black leaves and not a very healthy aspect. Nevertheless, some plant species have such characteristics so we decide to trust the labels and keep them as such.

B. Removing duplicates.

During preprocessing we noticed that the dataset was containing duplicates of the some pictures. We made sure that the labels were all the same for each duplicate so as removing them do not impact the outcome. Since they were all the same, we did not have to choose which duplicate had the “right” label and simply had to keep one random occurrence of each duplicate in our dataset.
This step was only taken towards the end of the development phase. Indeed, it took us a while to notice the duplicates which were hindering our accuracy. See section 4 for more details about performance with and without duplicates.

C. Split of the data.

After the removal of the duplicates and the outliers our original data set was now composed of 4841 images.
We decided to split the data into 2 sets: the training set and the validation set, since the test set is online on Codalab platform. The validation set was always the same for all the models tried and represented 33% of the original data set, so 1614 pictures. We chose 33% because we decided to apply data augmentation directly on the training set after,

so the training set would have been too big compared to the validation set had we chosen 25%.
At this stage, the training set is composed of 3227 pictures and 1614 pictures for the validation set.

D. Data Augmentation.

A natural step after removing the outliers and duplicates was to perform data augmentation. The cleaned dataset was now rather small (3227 pictures) and not sufficient to allow for training an efficient model.

Two approaches of data augmentation have been implemented, one directly included in the model construction and one applied on the training dataset.
The one included in the model construction is composed of random flipping (vertical one), random zooming (with parameter 0.25), random translation (with parameters 0.2 and 0.2) and random rotation (with parameter 0.15).
For the last type of data augmentation, which could not be added in the model, we tried several data augmentation methods directly on the training set: color jittering, mixup (with a lambda value randomly chosen between 0 and 1) and cutout. Examples are provided on the Fig. 3. We then kept the original picture and added the new augmented images.

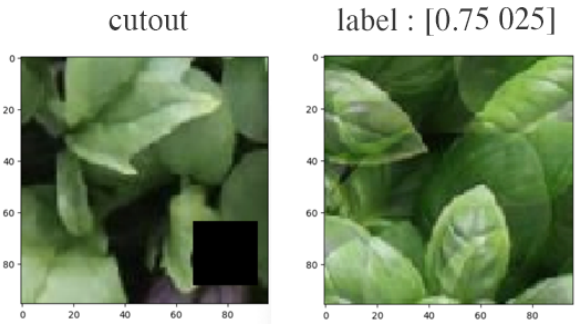

Fig. 3. Example of a cutout augmentation(left) and mixup(right)

The implementation of these methods made the training much richer and helped the model be more robust to changes in the input images. After this augmentation, we were left with 4627 pictures in the training.

Neural Network Architecture

We opted first for a transfer learning approach and then applied a fine tuning method to build our model. This approach was chosen by us considering the fact that the dataset was small and that building a specific Convolutional Neural Network and training it would take a lot of time.

The first model built was designed using the MobileNet network, that we used during laboratories with the approach of transfer learning. We also used and compared several Feature Extractor Networks : ResNet, ResNetv2, Xception, EfficientNet.

In the end, the best model for the training phase was one built using the EfficientNet B0 network (seen in [1]), the smallest of all EfficientNet networks, to which we added a Dropout layer in order to avoid overfitting. This is understandable since the EfficientNet family of networks are much smaller (in numbers of parameters) than the ResNet or MobileNet, and much easier to optimize for a small dataset such as the one we have. On the Fig. 4 we present an outline of the model:

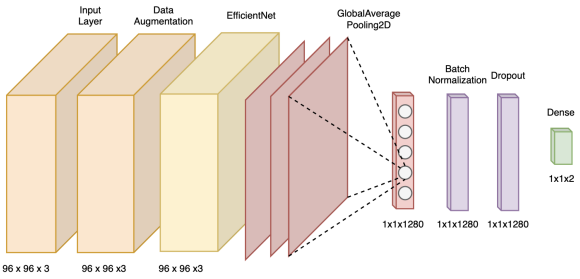

Fig. 4. Rough Outline of the Model

We decided to choose a rather small batch size since smaller batch sizes lead to better generalization and introduce a bit of noise in the gradient computation to avoid getting stuck in a local minima. We used Early stoping to limit overfitting with a patience of 30 at the begining. The number of epochs is set high since we use Early stoping. The optimizer was AdamW which is more accurate than Adam.

Optimization parameters
Max number of Epochs : 250
Batch size : 70
Patience : 30

Table 1. Optimization parameters table

For the fine tuning we unfroze 2 blocks by 2 blocks with a learning rate reducing as we got closer to the entrance. We stopped at 4 blocks because after that the results were not improving. We also increased the patience a bit (by increasing it from 30 to 45) because since the learning rate is smaller, it might take more time to converge. As recommended in the Keras documentation [2] we decided to keep the Batch normalization as non trainable during the Fine tuning.

Performance

The models were evaluated on the accuracy so all results are computed according to this metric.

E. During training.

After training of models with only transfer learning, we can see that the accuracy of the training set is lower than the one of the validation. This can be explained by the fact that the training set is composed of images augmented with cutout, mixup and color jittering so the EfficientNet network struggles with these images.
However when we use training with 4 blocks unfrozen, we

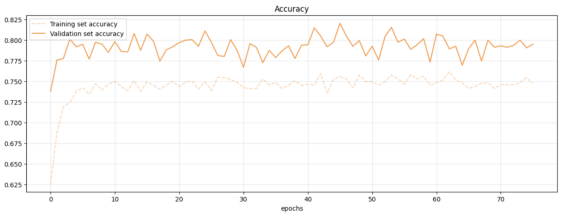

Fig. 5. Accuracy After Transfer Learning

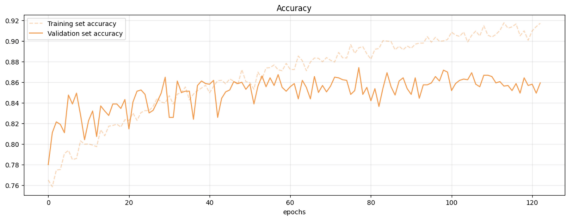

Fig. 6. Accuracy After Optimization of 4 blocks

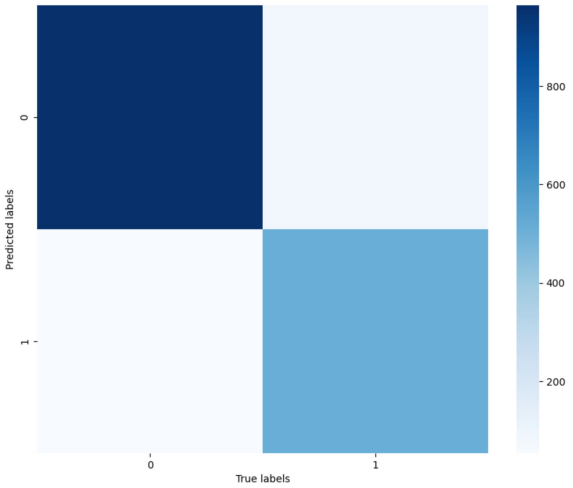

Fig. 7. Confusion Matrix of the Model

can see that the network adapts and the accuracy of the training set is now above the validation.
By the end, unsatisfied with our results upon uploading on Codalab, we decided to train all the blocks of the EfficientNet network. Keeping the same architecture, we reached an accuracy of 0.9256 for the validation set and 0.9670 for the training set. Sadly, when we uploaded it on Codalab we reached a lower than 81% accuracy. This can be explained by the fact that by unfreezing all the blocks at once, the model is more subject to overfitting, hence giving a lower accuracy with a test set very different from the one used for training and validation.

As mentioned in subsection B, we realized that there were duplicates in the given dataset. Table 2. present the comparision of accuraccy reached on the validation set before and after removing the duplicates.

Accuracy	With duplicates	Without duplicates
Validation set	0.8589	0.8917
Training set	0.8892	0.9251

Table 2. Comparison of accuracy of the model with and without duplicates

After these changes, we obtained the Confusion Matrix shown in Fig. 7.
We can observe that the model is detecting better the healthy plants (labeled as 0) than the unhealthy plants (labeled as 1). This can explained by the fact that the original data set has much more pictures labeled as healthy than unhealthy.

F. Final Performance.

The final performance during the final phase of the competition on Codalab is 0.785.

We achieved this result by using the model for which we had the best accuracy during the development phase. This accuracy is lower than the one we obtained with the same model during the previous phase. However, the margin is rather reasonable since we obtained 0.81 during this phase.

We can suppose that this is due to a introducing significant changes in the test set used during the final phase and overfitting in our model, hindering it from generalizing to a more different dataset.

Conclusions

In this report we presented the result of the first challenge conducted in the Artificial Neural Networks and Deep Learning course. Task of homework was a binary classification of the dataset containing plant images. Our approach was to use the EfficientNet B0 network with fine tuning and train this model on the preprocessed dataset.
The preprocessing part contained inspecting the dataset for outliers and duplicates with their removal. Next step was to augment the dataset using three different methods: color jittering, mixup and cutout (on the training set) and operations implemented in the model.
After that, we tried multiple networks but ended up using the EfficientNet B0 network which was the most fitting for our small dataset and added a Dropout layer to avoid overfitting.

In the end, we obtained a 0.81 Accuracy during development phase and 0.785 during the final phase.

One way to improve obtained results which we have not managed to finish due to the time constrain could be to study the ROC curve when we change the value of the threshold (which is set to 0.5 by default) and choose the curve that maximizes the accuracy. Another method could be to use cross validation or apply the same data augmentation that we did in the training to the test set. Also we should look for some novel techniques presented during lectures and laboratories of Artificial Neural Networks and Deep Learning course which we are going to try to include in the next challenge during this course.

References

[1] Giacomo Boracchi. *Famous CNN Architectures and CNN Visualization*. 2023. URL: https://boracchi.faculty.polimi.it.

[2] Yixing Fu. *Image classification via fine-tuning with EfficientNet*. 2020. URL: https://keras.io/examples/vision/image_classification_efficientnet_fine_tuning/.

[3] Eugenio Lomurno. *[2023-2024] AN2DL*. 2023. URL: https://drive.google.com/drive/folders/1x1yvXG9iBkUsSYrGL-OPhZpm5PzXMcb-.

[4] Eugenio Lomurno. *Logbook*. 2023. URL: https://docs.google.com/document/d/1sbLFtKa2ld99yTIiKcuVn7o4SPBUYQ2y_SmZkL6m340/edit.

[5] Matteo Matteucci. *Neural Networks Training and Overfitting*. 2023. URL: https://www.deib.polimi.it/eng/people/details/267262.

Contributions

Contribution	
Anne-lys	Model training, Data augmentation, Dropout, Report
Maxime	Final model, detection of outliers and duplicates, Data Augmentation, Report
Ahmed	Research on alternative models, Model training, Report
Jakub	Data preprocessing, Model training, Report

Table 3. Contribution table