Artificial Neural Networks and Deep Learning – Challenge 1

Giada Silvestrini 10659711
Ludovica Tassini 10663137
Giulia Venturini 10680310

Group: AI Queens

THE PROBLEM: *Binary classification*

The assigned challenge is a binary classification problem. Given a training dataset of 5200 labeled images of leaves, where the labels are “healthy” and “unhealthy”, we had to implement a neural network to correctly classify new unlabeled images.

FIRST PHASE*: Development*

We faced the first challenge using different approaches, starting from the more naive and simple ones, to understand the basics, and ending with more efficient and complex methods.

During the first day of the challenge, we tried to understand and visualize the dataset. We developed a preprocessing code in which:

- We loaded the data on Google Drive
- We extracted the data from the .npz file
- We plotted the data to visualize them

From this first analysis, we immediately noticed that there were some unexpected images, such as Shrek and Trololo. We updated our preprocessing code to filter out these outliers from the dataset and created a new clean dataset in the .npz format.
After that, we started implementing a simple model based on the notebook shown in the laboratory session regarding “LeNet”. We tried to upload it in order to check the submission procedure on Codalab platform: this first attempt resulted in an accuracy of 53%.

Then, starting from this code, we modified some parameters related to the training phase (number of epochs and early stopping patience) to “play with the network” and have a better understanding of its behavior, depending on the different hyperparameters. We submitted this new model with an increased value of patience and number of epochs, yielding an accuracy of 62%.
Based on these results, we gathered that the first LeNet attempt was learning too slowly and, by stopping the training phase before time, the model was underfitting the training set.

The next day, we tried to improve the LeNet network by adding some layers. However, since we obtained a poor local result for the validation accuracy, we opted for the exploration of new paths.
The next approach we experimented was a VGG-like architecture to which we added batch normalization and shortcut links, taking inspiration from the material provided. Unfortunately, it did not significantly improve our performance.

The three of us implemented different versions of the same VGG-like architecture, changing some parameters. We also added some first attempts of image augmentation, by including many transformations: rotation, translation, crop, flip, zoom, brightness and contrast.

Despite our efforts, the improvement we obtained was not as significant as we expected: the best of these models reached an accuracy of 64% on Codalab, even though locally it performed considerably better. Due to the fact that we used a low learning rate, the training needed a high value of the early stopping patience in order to let the network learn. For this reason the model was probably overfitting, since locally the validation accuracy was around 85%, while the training one was about 97%, with a difference between training and validation loss of almost one order of magnitude.
For these reasons, we started to think about more advanced and complex models, using the keras.applications API to exploit the most famous convolutional neural networks trained on Imagenet. This way, we could take advantage of pre-trained and well-performing architectures.

Furthermore, we thought that another way to improve the model might be to use Transfer Learning. The keras networks that we explored were, in order:

- MobileNetV2: we chose this network as seen in the laboratory sessions, since we wanted to have a solid baseline model for transfer learning experiments. However, we did not obtain satisfying results.
- ResNet50: once applied transfer learning to this network, we obtained very poor results with high and inconsistent validation errors.
- EfficientNetB2: with this network we experienced encouraging results, since this was our first submission that got more than 80% on validation accuracy. Afterwards, we wanted to experiment with slight variations to this architecture, by including an additional dense layer as a last step before the output layer. Such modification allowed us to reach a result of 84%, both locally with our validation set and on Codalab: the model was not overfitting, nor underfitting.

Based on these results, we decided to implement fine tuning on top of the aforementioned EfficientNetB2 model. The improvement we got was not as meaningful as expected, thus we decided to apply this technique to a larger network. ConvNeXtBase was the most promising architecture that we experienced and, for this reason, we focused on maximizing its results.

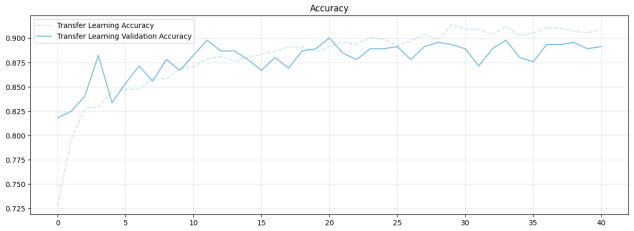
Specifically, we first obtained a validation accuracy of 85% using:

- Data augmentation: horizontal, vertical flip and rotation (with a factor of 0.15). Other transformations such as brightness, contrast and zoom resulted in worse results on our training dataset. This makes sense since rotating and flipping the images doesn't affect the healthiness of the leaves, it's just like taking a picture from a different angle, while contrast and brightness might mess with the colors and nuances of each leaf.
  We also thought that adding many transformations could be counterproductive, thus we focused only on some of them.
- Early stopping callback function with patience 10.
- Number of freezed layers equal to 180.

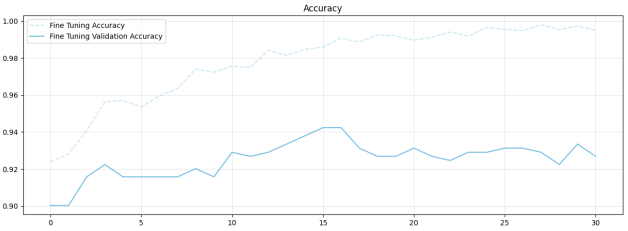
The main problem with this attempt was overfitting: the model trained with fine tuning reached a peak of 95%, with even higher training accuracy. In particular, one thing that we noticed was that we needed to further lower the early stopping patience, in order for the model not to overfit. This problem resulted in a very high difference of performance, once the model was submitted on Codalab: validation accuracy dropped to 85%.

By changing a few values in the parameters and by re-applying both transfer learning and fine tuning, we reached a local test accuracy of 92% (that is similar to the one obtained previously): when submitting the model on Codalab, differently than before, it resulted in a 91% of accuracy. This time the model was not overfitting. This is the most promising result we obtained in the first phase of the challenge.

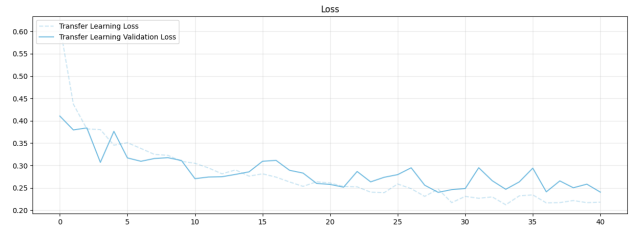
We also tried to improve our ConvNeXtBase model: we used the same parameters for the image augmentation, the epochs and the early stopping patience, but changed the transfer learning model with ConvNeXtLarge. We also added a Dense layer and a Dropout before the output layer. Fearing the model was overfitting, we added a weight decay of 1^{-3} to the Adam optimizer used in the transfer learning. Given the encouraging results obtained just from transfer learning, we decided to also do fine tuning, freezing the first 260 layers and adding a weight decay of 1^{-4} to the Adam optimizer to prevent overfitting. The local validation accuracy resulted in an encouraging 93%, but dropped to 90% when submitted to Codalab. Probably, despite our attempts, the model slightly overfitted the training dataset, even though the loss and validation loss were not too far from each other.



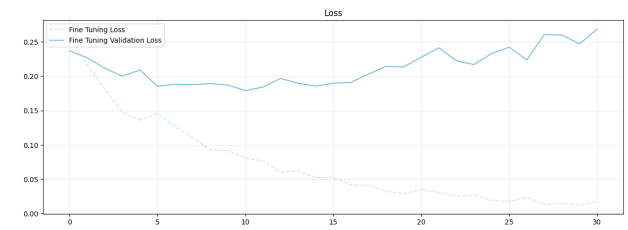
Accuracy of the ConvNeXtLarge TL model



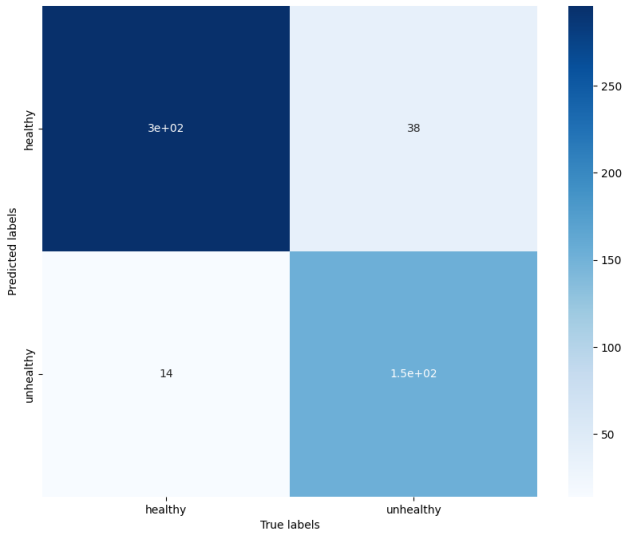
Accuracy of the ConvNeXtLarge FT model



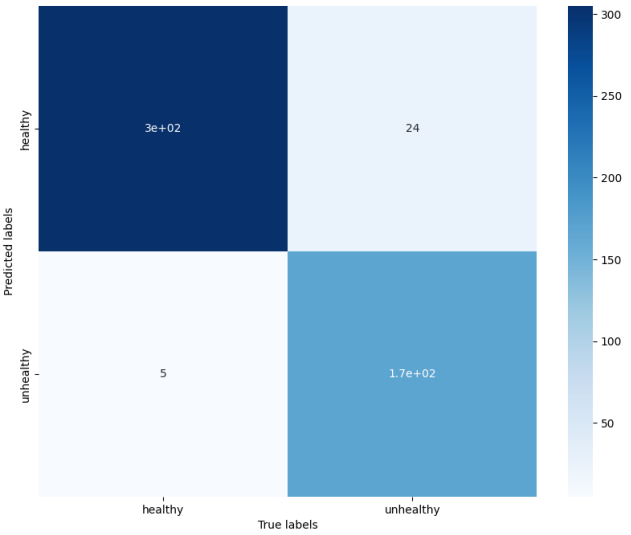
Loss of the ConvNeXtLarge TL model



Loss of the ConvNeXtLarge FT model



Confusion matrix of the ConvNeXtLarge TL model



Confusion matrix of the ConvNeXtLarge FT model

SECOND PHASE*: Final*

The best three models that we submitted in the second phase are:

- ConvNeXtLarge with fine tuning (development accuracy of 90%): it resulted in 85.80% of accuracy.
- ConvNeXtBase with fine tuning (which was the most promising one in the first phase, with a development accuracy of 91%): it resulted in 82.10% of accuracy.
- ConvNeXtBase with fine tuning and a different number of freezed layers (development accuracy of 90%): it resulted in 83.50% of accuracy.

CONTRIBUTIONS

During the first days of the development phase, the three of us worked together on the same tasks, trying to figure out the best way to start the project. In particular, it was necessary to start from a common cleaned dataset to further experiment the best techniques to build our models, so we joined our efforts on it.

After these first days, we worked both together and alone, ensuring that tasks were distributed equitably: this is proved by the fact that all of us submitted nearly the same number of models.

Aware of the fact that we needed to make several attempts and train numerous models in order to succeed in the competition, we continued to work in parallel over different architectures.

In the end, our group exhibited a very good level of dedication and communication, by maximizing individual strengths for the benefit of the entire team.

LINKS

Link to the preprocessed dataset: clean_public_data.npz