1. Introduction

For this first homework, we were asked to classify plants into two categories defined by their state of health: healthy and unhealthy. To solve the binary classification problem, we were provided with a dataset containing 5,200 RGB images of size 96x96, together with their associated labels.

1. Data preparation
   1. Cleaning of the data

Among the 5,200 images supplied, we first noticed the repeated presence of intruding images, hardly fitting in with plants, but rather with Mr Trololo and Shreck. We used the Fastdup tool [1] to find and delete the duplicate images in the dataset, as well as outliers. The dataset resulting from cleaning contains 4,532 images, in which 2,900 are labeled as healthy and 1,632 as unhealthy.

* 1. Balancing data

Based on the results of our initial tests, we believe that this imbalance in label distribution has led our model to overestimate the number of healthy plants. Thus we tried different method to fix it.

* + 1. Class weights approach

Firstly we tried to fight this problem by modifying class weights. Based on an unbalanced set we calculated a weight for each class to try to compensate for the imbalance. This method gave good results on our dataset but wasn’t working on the testing set because the label balance was different. Thefore we decide to try another method.

* + 1. Data augmentation

We decided to augment our data in order to rebalance the proportion of healthy and unhealthy images in the dataset. We used augmentation methods such as random flip, random translation and random rotation. We chose these augmentation methods so the data remains realistic. Our objective was to avoid introducing irrelevant data that could lead the model to form unintended correlations. Our fear was that by modifying propreties as luminosity, the unhealthy images could look more healthy since it is relate to the color of the plant.

1. Building the model
   1. Transfer learning

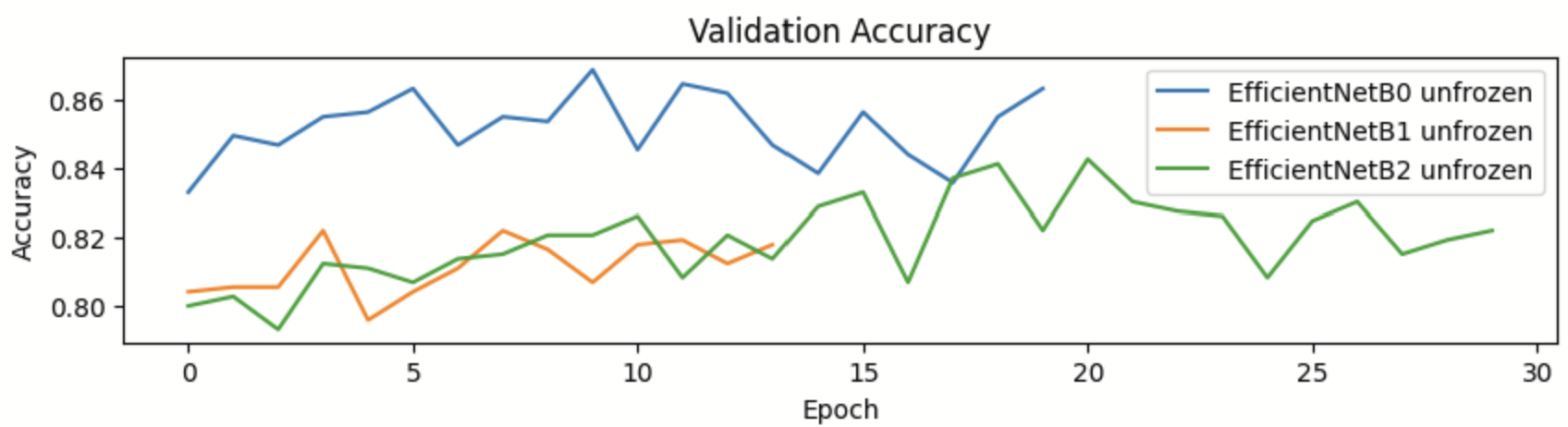
At first, simple CNN models were tested. However, it quickly became apparent that models built using transfer learning were much more effective and faster to train. Therefore, the decision was made to focus on this method.

* 1. Transfer learning models

We started by testing multiple transfer learning models from Keras Applications [2] and compared the results to find which model is the best fit for our problem. As it is visible in the table and the chart, the EfficientNet model has provided the best results.

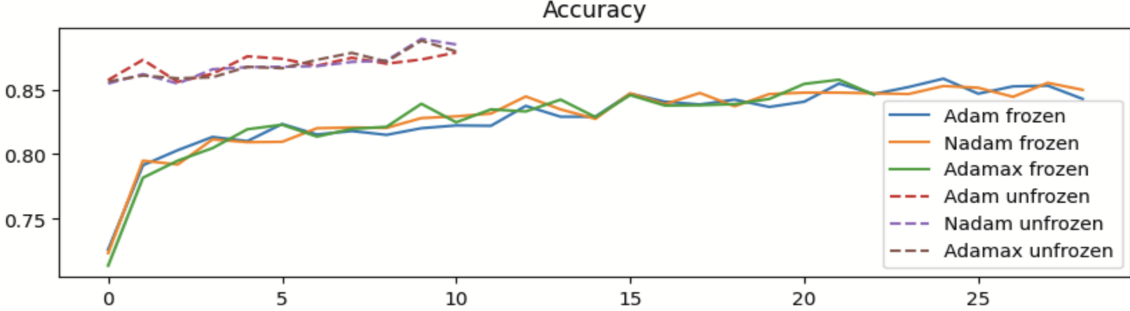
A graph of a line graph

Description automatically generated with medium confidence

Then we tried different EfficientNetModels and kept the best one wich was EfficientNetB0.

* 1. Parameters
     1. Optimizers

Models using transfer learning undergo two training phases. The first phase involves freezing the layers of the pretrained model, while the second phase unfreezes all layers. As a result, it is possible to use two different optimizers for a single model. Preliminary tests identified Adam, Nadam and Adamax as good potential candidates. More in-depth tests were then conducted on these optimizers to find the two bests.



The difference between the optimizers is small but the tests seemed to show that the best option is to use Nadam for the two phases.

* + 1. Dense layers

After testing various dense layer configurations, as seen in our *DenseLayerChoice* notebook, the best combination was using *GlobalAveragePooling, Dropout(0.3), Dense Layer(10, relu)* and *Batch Normalization.*

* + 1. Unfreezing layers

The layers of the transfer learning models can also be unfrozen to be optimized on a different dataset than the one it was pre-trained on. We have tried unfreezing 20, 25, 30, 35 and 40 layers, and the results have shown that the best accuracy is obtained when 30 layers are unfrozen. The testing is shown in notebook *UnFreezeLayerChoice.*

* + 1. Callbacks

To improve the training of the models, as well as to prevent overfitting, we used two callback functions. The callbacks we used are *EarlyStopping* [3] and *ReduceLROnPlateau* [4]. Both callbacks are monitoring the validation set accuracy, with *EarlyStopping* having the patience value equal to 5 and the *ReduceLROnPlateau* 10.

1. Final implemented solution
   1. Model and parameters

Finally, combining all the tested parameters we tested 5 different transfer learning models, EfficientNetB0, MobileNetV2, ResNet50V2, DenseNet169 and InceptionResNetV2. The best results were given by the EfficientNetB0.

The results from the previous chapter show that the best model should be build as follow :

|  |  |  |  |
| --- | --- | --- | --- |
| Optimizers | Transferlearning | Dense layers | Unfrozen depth |
| Nadam-Nadam | EfficientNetB0 | *GlobalAveragePooling, Dropout(0.3), Dense Layer(10, relu)* and *Batch Normalization* | 30 |

However since the outcome of some tests where pretty close and the final test dataset will be different from our dataset, we decided to test some variations that can be found in the final notebook.

* 1. HeatMaps

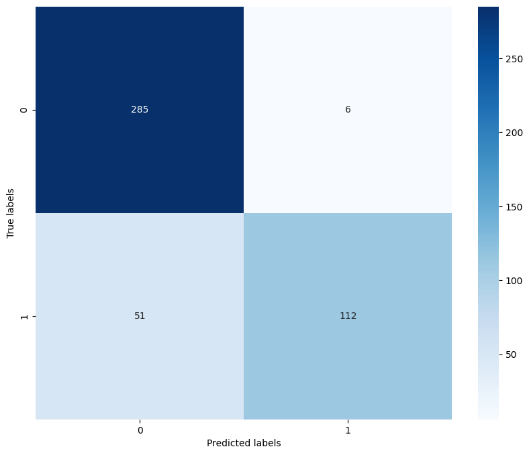
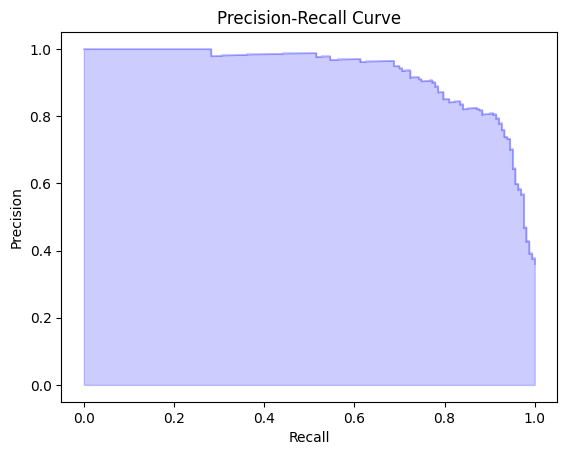
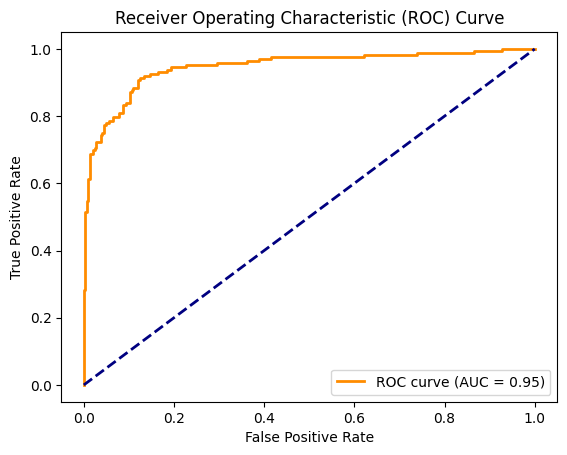
From top convolutional layer we also created heat map from which we were able to conclude that centre regions of an input image contributed the most to the prediction of a particular class.

A green and blue color scheme

Description automatically generated with medium confidence

* 1. Results
     1. ROC AUC, confusion matrix, precision-recall curve

Other than accuracy, the model was judged by the ROC AUC curve, in which it has the result of 0.95. This results shows that the model has a great ability to distinguish between different classes in a given dataset. Also, the Precision-recall curve shows the model’s effectiveness. The confusion matrix is representing the results of predicting on the test set, out of 454 images, there were 285 true positives (healthy plants), 112 true negatives (unhealthy plants), 51 false positives and 6 false negatives.



* + 1. Accuracy, precision, recall, F1 score

The main results by which the model was chosen as the best are the accuracy, precision, recall and F1 score. Accuracy shows how many images were correctly classified, and that score was 0.8744. Precision is a characteristic which describes how many of the predicted positive instances were actually correct. For this model, precision = 0.9492. Recall measures how many of the actual positive instances were correctly predicted by the model, and this model has a score recall = 0.6871. Lastly, F1 score describes the balance between precision and recall, and in this case the value is F1 = 0.7972.

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 |
| 0.8744 | 0.9492 | 0.6871 | 0.7972 |

1. Conclusion

The obtained model can classify images with an accuracy of 0.8744 (on test set). In the competition rankings, some groups achieved significantly higher scores, so it is evidently possible to do better. We believe that the most promising avenues for improvement are : using superresolution models for image resizing, using grid-fine tuning for hyperparameters instead of ou “greedy” approach, using K-fold or other cross validation technics, using multiple model and a vote system like a random forest (ensemble model).

1. **Contributions**
   * Filip Fabris - models testing, dense layers, transfer learning, unfreeze depth, heat map, report, research
   * Maxime Pellichero – models testing, optimizer choice, model comparaison,
   * Jana Penic – testing models, cleaning dataset, report, heatmap
   * Mai Lan Pho - testing models, cleaning dataset, report
2. **References**

[1] <https://visual-layer.github.io/fastdup/>

[2] <https://keras.io/api/applications/>

[3] <https://keras.io/api/callbacks/early_stopping/>

[4] <https://keras.io/api/callbacks/reduce_lr_on_plateau/>