Artificial Neural Network and Deep Learning Homework 1

Paolo Piccirilli, Marco Riva, Federico Sarrocco, Alessandro Vacca

November 18, 2023

Figure 1: Example of both image classes

1 Introduction

The challenge was to build a binary classification model to distinguish between healthy and unhealthy plants using a dataset of 5200 images. The dataset was slightly imbalanced, leaning towards the healthy class, and contained outliers like memes and unrelated images that needed identification and removal. The strategy of our team developed methodically, starting with inspecting and cleaning the data, then implementing an initial model. We progressed to adopting augmentation strategies, gradually increasing the complexity of the model, followed by fine-tuning hyperparameters and employing model ensembling.

2 Development

2.1 Initial Work

Before defining the model to test, it was necessary to decide how to deal with the problems related to the dataset, such as the presence of outlier elements, imbalance and the low quantity of the images. As a first approach towards these issues we decided to remove those outliers manually (this being a relatively simple task, as these were only of two types), to calculate the weight of each class and to implement data augmentation to try to mitigate these problems. After that, the simplest attempt to begin with seemed to be implementing one of the networks presented during the laboratories using transfer learning: we first started with **MobileNetV2**, because of its lightweightness, flexibility and the fact that it comes as a pre-trained network on ImageNet, just to start.

The initial network performed quite well, but we were still left with a lot of room for improvement.

In order to do so, we started working locally trying first to remove the data augmentation, noticing a slight improvement. After some attempts we realized that, we were unsing too many augmentation layers, hence introducing some changes that ruined distinctive information of the classes, in particular those related to colours and contrast.

During the first phase, we became aware that we had used too small validation and test sets, dictated by concerns about removing too many images from a already limited sample, but this were resulting in unreliable evaluations of the network, which was confirmed with submissions that performed significantly worse than what we were obtaining locally.

2.2 Towards the Final Model

We continued to improve the **MobileNetV2**-based model with Transfer Learning, as it reduces the need for extensive data resources (a key aspect of this challenge), making it easier to mitigate overfitting. But we kept getting significant discrepancies between our test set and the competition's one. We identified some critical issues that might limit its performance, in fact after consulting the Keras documentation, we realized that the chosen **MobileNetV2** may not have been the best choice considering the given dataset, as for its limitations for relatively small datasets like ours. We therefore decided to experiment with different network architectures, such as n manually written **CNN**, **ResNet**, **Xception**, **EfficientNet**, **ConvNext**, and **DenseNet**. Out of these, **DenseNet** and **ConvNext**, after several adjustments with fine-tuning, proved to be the best performing. Our final reasoning is that the **ConvNext** architecture is effective because of its use of a blended CNN with Transformers, one of the most modern approaches available and the **DenseNet** architecture is effective because it creates a highly interconnected network, leading to efficient feature reuse and reduced number of parameters, thus highly mitigating overfitting.

Although the accuracy obtained in the various models increased gradually, the challenge prove quite comptetitive: we therefore decided on combining multiple models together to increase accuracy, resulting in an ensemble model, which turned out to be our final solution.

3 Final model

3.1 Data Pre-Processing & Augmentation

A key part of the final model is the processing of the training dataset. We begun by inspecting the data, drawing the images in a plot, considering the **PCA** dimensionality reduction on the features derived from a pretrained **MobileNetV2** backbone. We removed outiliers in the dataset tresholding with the Mahalanobis distance from the mean feature space representation, quickly identifying and removing outliers, as shown in Figure 2. The second step was to check for **duplicates** and removing them. The dataset was reduced from 5200 images to just 4850, thus we chose to implement some techniques of **data augmentation**. The following augmentation pipeline (`imgaug` [2]) was chosen, based on its effectivness on the validation set and its ability to handle class imbalance through augmentation itself and class weights: we combined random affine transformations, including rotations, scaling, and translations, with a probability of 40%. We also applied random cropping, padding, and flipping. Additionally, we selectively used augmentations such as noise addition, dropout, and color adjustments, each applied randomly with a 30% chance, to improve robustness and effectiveness. The preprocessing pipeline is applied directly to the Keras training dataset. Rather than augmenting the images offline, we chose to map the function to the dataset to save RAM. All images undergo scaling from the 0 to 255 interval to fit within the 0 to 1 range.

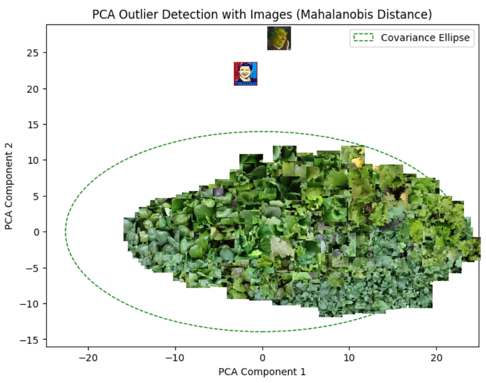

Figure 2: Outlier Detection

3.2 Hyperparameter Search - Black-Box Optimization

The exploration of optimal hyperparameters represents a time-intensive task, prompting the implementation of an automated hyperparameter search algorithm. Our chosen method relies on black box function optimization, employing the `Optuna` framework [1]. This implementation has led to significant enhancements in both trial sampling performance and the overall efficacy of our models.

The primary objective of this search is to maximize model validation accuracy. Optuna dynamically proposes diverse hyperparameter configurations, not only refining numerical values but also shaping the structural architecture of our neural network (e.g., determining the number of layers, integrating dropout layers, employing pooling layers...).

This process also allows us to estimate the importance of hyperparameters, as seen in Figure 4.

The primary hyperparameters we primarily focus on for transfer learning are:

- Learning rate
- Configuration of layers within the head:
  - Number of layers
  - Number of neurons

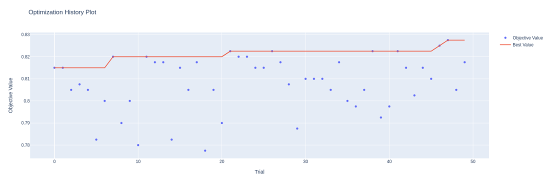

Figure 3: Optimization History of a ResNet model during an hyperparameters optimization

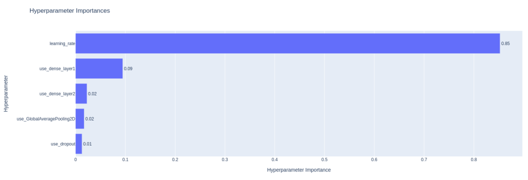

Figure 4: Hyperparameters Importance from optuna framework

- Choice between:
  - Pooling layer
  - Flattening layer

3.3 Model

Our final model is an ensemble of multiple models, combining various architectures to maximize performance by leveraging their strengths and adding diversity through ensembling. Each model underwent a hyperparameter search described in 3.2.

The ensemble model employs a weighted average strategy, utilizing individual leaderboard performances of distinct models:

$$prediction = \sum_{i=0}^{N_{models}} prediction[i] \times weight[i]$$

We adopted two distinct approaches within this ensemble:

1. Development of a convolutional neural network from scratch.
2. Employing transfer learning and fine-tuning larger pre-trained models.

3.3.1 Transfer learning and fine-tuning models

We incorporated a range of pre-trained architectures into our ensemble, including **Xception**, **EfficientNetV2S**, **ResNet50V2**, **ConvNeXtBase**, and **DenseNet121**. These were chosen to leverage their individual strengths in feature extraction and classification. To accommodate our machine's memory limitations, we primarily utilized the smaller versions of these architectures. Hyperparameters including the number of fully connected layers in the head, the number of neurons, the use of pooling or flattening layers, and learning rates varied for each network, optimized using the approach described in 3.2.

All the model are initially initialized with weights pre-trained on ImageNet. We fine-tuned them by freezing most layers during the transfer learning phase. Subsequently, fine-tuning involved selectively unfreezing and training specific deeper layers, enabling the model to adapt to our domain-specific dataset.

The models are trained using the ADAM optimizer with a carefully chosen learning rate. This was optimized using a systematic hyperparameter search methodology detailed in 3.2.

3.3.2 Final training

A final training of all networks utilized the full dataset, fixing the number of epochs and considering the same number of improving epochs used in the training with the train-validation-test dataset split.

4 Conclusion

The ensemble allowed us to obtain a satisfying result, far better than any single net model. The accuracy returned is around 80% while both precision and recall asseses around 74% wich means that the number of false positives is very similar to that of false negatives, therefore the net has an equal capibility of recognizing each of the two classes. An interesting point to be underlined is the gap between the performance of our models on our local data and the performance on the (hidden) test data, which was always consistently worse, this is probably due to some differences between the images that were provided versus the ones that we were testing our model against.

5 Contributions

As for the distribution of the work, we decided to start by working together, to best define the starting point. As the foundations of the code were laid, we distributed the tests of the various networks individually, to parallelize the work. We have therefore not defined specific roles and activities for each of us, although some tasks have mostly been carried out as indicated in the following list:

- Research in new approaches: Federico, Alesandro
- Training and testing: Paolo, Alessandro
- Coding and development: Federico, Alessandro, Marco
- Report: Paolo, Marco

References

[1] Takuya Akiba et al. "Optuna: A Next-generation Hyperparameter Optimization Framework". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2019.

[2] Alexander B. Jung. *imgaug*. `https://github.com/aleju/imgaug`. [Online; accessed 30-Oct-2018]. 2018.