Artificial Neural Networks and Deep Learning

Image Classification - Homework 1 - Report 2023

Paolo Pertino, Alberto Sandri, Enrico Simionato

Problem Description In an era where agriculture and plant well-being are critical for sustaining ecosystems and supporting global food production, accurate and efficient plant health classification methodologies become indispensable. The following report delves into the realm of plant health classification, where the primary objective is to categorize plants into two distinct groups based on their state of health. This task is framed as a binary classification problem, requiring the prediction of the correct class label from the set {*healthy*: 0, *unhealthy*: 1}. This report outlines the methods, techniques, and results of the analysis conducted to achieve the specified goal, shedding light on the application of deep neural networks in addressing crucial issues in the domain of agriculture and environmental science.

Data inspection During data inspection we noticed the presence of two different outliers repeated many times. They are not part of the problem since they are not plants, therefore all their appearances have been removed. Moreover, we found out that duplicates are present thus we decided to remove those instances.
After the removal of these images, the dataset is composed of 4850 images, the images of class *healthy* are 3060 (63.09% of the samples), and the images falling into the class *unhealthy* are 1790 (36.91% of the samples) [IMG1]. Thus, the dataset is a little unbalanced towards the *healthy* class. The images to classify are of size 96x96 in RGB format.

Preprocessing The implemented approach starts from the hold-out split of the input dataset. We adopted 500 samples for the validation set and 500 samples for the test set, split in a stratified fashion.

Best model - ConvNeXt The most successful approach involved fine-tuning a pre-trained feature extractor network with an added classification head. Prior to fine-tuning, a transfer learning step focused on training a binary classifier. In the latter stages of the initial phase of the challenge, there was a shift to FENs based on the ConvNeXt model family. This transition was made as an alternative to the initially used EfficientNet, which had demonstrated limitations in improving overall performance. Our exploration encompassed experiments with small, base, and large variants of the ConvNeXt models. Ultimately, the base model emerged as the optimal compromise, considering both performance and computational requirements.
We observed that the training of ConvNeXt models was stable and resilient to overfitting, enabling robust generalization. This characteristic allowed the establishment of the local test set as a reliable proxy for evaluating the model's performance on CodaLab. The most successful model was a ConvNeXt base model trained by applying data augmentation techniques on the images, whose preprocessing pipeline is elaborated in the subsequent section.
For transfer learning, as classifier, we employed a dense network comprising 64 neurons, a subsequent batch normalization layer, the ReLU activation function, and dropout preceding the final layer with two output neurons. To address class imbalance, we incorporated sample

weights during training, assigning a higher weight to *unhealthy* samples in contrast to *healthy* ones according to their relative frequency.
Fine-tuning was executed by unlocking five blocks, obtaining a model that achieved an accuracy of 91% on the CodaLab test set in the first phase. Encouraged by this success, we constructed an analogous network with identical hyperparameters leveraging the entirety of available data. Notably, this subsequent model, trained without the use of callbacks, achieved an accuracy of 94% on CodaLab development phase. The results underscore the efficacy of the ConvNeXt architecture and the strategic application of data augmentation and weight balancing strategies in optimizing the performance of the implemented network. Still, there was some level of overfitting since in the second phase the test accuracy dropped by some points.

Experiments

Augmentation techniques Various augmentation methods have been investigated. Our networks' preprocessing layer leverages both geometric transformations—such as random horizontal and vertical flips and random rotations (factor equal to 0.2)—and a photometric transformation, specifically random adjustments to brightness (with a factor of 0.1) [IMG2]. Given that color variation is a key distinguishing factor between a *healthy* and an *unhealthy* plant, we opted to implement fewer and lighter transformations focusing on color modifications.
In conclusion, certain experiments were conducted utilizing "modern" augmentation techniques such as MixUp and CutMix [IMG3]. Consequently, new versions of the training dataset were generated and employed. While there wasn't a remarkable improvement in performance, a noticeable regularization effect was observed during the training phase of the networks.

Custom models We attempted to construct and train a custom model from the ground up, specifically designed for the task. However, early on, we encountered challenges in identifying the optimal set of layers and hyperparameters, resulting in networks that either easily underfit or overfit the data. The training phase proved to be highly unstable, with the network struggling to learn effectively. Recognizing these difficulties, we swiftly transitioned to transfer learning, which demonstrated greater stability. Additionally, it became apparent that our available data was insufficient to train a comprehensive model entirely from scratch.

In addition to the models we are going to mention, we also experimented with DenseNet and MobileNet. However, we discontinued their use early on as they consistently yielded inferior results with respect to other pre-trained feature extraction networks (FENs).

EfficientNet models Since the first usage of the EfficientNet models as FEN, it was noticed that they had many advantages. Their performance was very high and the training was faster than many other models, also of smaller size.
In particular ‘EfficientNetB0’, ‘EfficientNetB4’, and ‘EfficientNetB7’ were tried and achieved promising results. The problem we encountered with these models was the fact that they were not too stable. In fact, even if with the holdout test set we were reaching good results, around 85%-90% accuracy, once submitted we frequently witnessed a drop of 5-6% of the same metric. This happened even though the models did not seem to have overfitted as the accuracy values on train, validation and test data were very similar.

ROC Analysis In cases of dataset imbalance, adjusting the classification threshold away from 0.5 can be beneficial. This adjustment aims to demand more evidence before classifying a plant as *healthy*, prioritizing the *unhealthy* class (the actual positive class). The threshold, which was fine-tuned for maximizing the Area Under the Curve (AUC) on the validation set, was found to be around 0.65 [IMG4]. In other words, the model classifies a plant as healthy (belonging to the negative class) only if its confidence in that classification is greater than 0.65. This adjustment resulted in a slightly higher recall, as expected, favoring the model's ability to predict the positive class.

Test-time augmentation During training with augmentations, the model may not achieve perfect invariance to transformations. To address this, test-time augmentation is employed, involving rotations, flips, and brightness changes, i.e. the same applied during training. Each test image undergoes the same modifications multiple times, e.g. 10, resulting in many predictions per image, which are then averaged. This approach is particularly effective for uncertain test images, as averaging predictions on randomly modified images helps mitigate errors, allowing the correct answer to stand out. In our case, we noticed improvements of around 2-3% in all the evaluation metrics both locally and on CodaLab test sets.

Ensemble Trying to enhance the performance, we implemented some ensemble methods.

We firstly employed bootstrapping to train a collection of similar models. We created the ensembles either combining custom models or combining EfficientNet models. The results were comparable to the ones of the individual models, offering no significant improvement.

The second ensemble configuration consisted of a FEN combining latent representations from various pre-trained networks and a simple dense neural network as the classification head. Despite the very fast training achieved through transfer learning, overall results were satisfactory but did not justify the increased model size.

In a final attempt, we constructed an ensemble using three ConvNeXt large models and two ConvNeXt base models. Techniques such as majority voting and soft voting were employed to compute the predictions. The collective performance did not surpass that of the best individual model. We expected this behavior since ConvNeXt seemed quite stable.

One hypothesis for the limited improvement with ensembles is the complexity of neural networks, which differs from the simplicity of traditional learners employed in ensembles. Additionally, the similarity in training data among models, despite architectural differences, may have hindered the potential benefits of ensemble methods.

Cross-validation Cross-validation has been tried on some hyperparameters. Still, due to the high number of hyperparameters and the time complexity of the training, it ends up being unfeasible, even for relatively small networks.

Explainability To comprehend our model, we employed the LIME library, which treats the model as a black box. LIME generates predictions by providing input data to the model and examines how predictions change with variations in the input data. It creates a new dataset with permuted samples and corresponding predictions, then trains an interpretable model on this dataset. In analyzing unhealthy plant images, we discovered that the network often identifies features like slightly yellow or purple spots on leaves as crucial for classification.[IMG5]

Contributions

In the pursuit of optimizing our CNN models for a binary classification task, our team adopted a collaborative yet diversified approach. We proceeded to work step by step in parallel, starting from building a baseline model, moving then to transfer learning by exploring the available pre-trained feature extraction networks, down to fine-tuning. This means that every one of us tried on his own to do each of the aforementioned steps reporting to the others remarkable results. This collaborative approach enabled everyone to actively participate in every significant phase, maximizing our collective learning experience. Moreover, each one of us explored more deeply, based on its interests, some specific parts of the pipeline: Pertino, for instance, dedicated some effort to exploring, implementing, and validating some data augmentation techniques like MixUp and CutMix; Sandri undertook the task of cross-validation and explainability; Simionato, on the other hand, tried different forms of ensembles, experimenting with different techniques to combine model predictions and enhance overall performance.

Image reference

[IMG1] Dataset distribution - Notebook Section 1.2
[IMG2] Classical augmentations used - Notebook Section 2.2.1
[IMG3] MixUp and CutMix dataset creation - Notebook Section 1.6 & 1.7
[IMG4] ROC Analysis - Notebook Section 2.2.3
[IMG5] Explainability with LIME - Notebook Section 3