Leaf Health Classification - 1^{st} Homework

Francesco Caserta, Nuno Costa, Shodai Fujimoto, Rio Ishibashi
Artificial Neural Networks and Deep Learning, 2023
Politecnico di Milano, Italy

1 INTRODUCTION

This report highlights our approach for classifying leaf health in the AN2DL course's 1^{st} homework assignment. It involves analyzing 96x96 leaf images to distinguish between healthy and unhealthy instances.

The report will include concise data analysis, preprocessing, augmentation methods employed, descriptions of model architectures, and other techniques used.

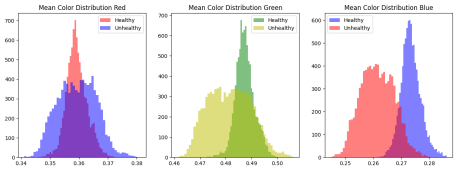
2 DATA ANALYSIS

Before the classifying task, we decided to better grasp the data. The dataset was initially composed of 5200 RGB images. We found 196 fabricated images (Shrek and TROLOLO images) upon inspection.

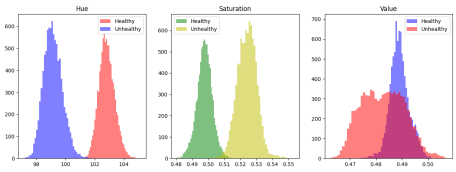
We also searched for duplicates. To do so, we first checked and found 154 true duplicates – images that were precisely the same pixel-wise. After removing those, we ended up with a dataset of 4850 images. Since we'll do data augmentation, we also checked for similar duplicates corresponding to transformations in the data itself. We leveraged the *imagededup* package to check that (Figure 1 shows the result). However, we did not find any relevant duplicates. We only noticed images of very similar plants (or the same plant even) but of different parts. As such, we decided to keep them.

Figure 1: Images with more than 95% of similarity

After this cleanup, we could adequately analyze the properties of the images in the dataset. We analyzed the average image of both classes, obtaining no clear distinction. The channel-wise distribution did not yield significant differences, as seen in Figure 2. However, the HSV conversion showed that, on average, hue and saturation were distinct between healthy and unhealthy classes. This led us to believe random augmentation involving these properties might lead to label noise.



(a) RGB Decomposition



(b) HSV Decomposition

Figure 2: Channel-wise mean image composition

Figure 3: Contrast-limited Adaptive Histogram Equalization

After this, we evaluated the visual effectiveness of image-processing techniques such as erosion, dilation, and contrast-limited Adaptive Histogram Equalization (CLAHE). The latter yielded visibly improved images, as seen in Figure 3.

Regarding dataset properties, we noticed a clear class imbalance, with 63% of the images belonging to the *healthy* class. Aside from this, we also did outlier detection through the Isolation Forest algorithm and noticed that no clear outliers existed in the dataset - the ones

the algorithm returned were simply darker images but still relevant to the task.

We leveraged all this relevant information in the preprocessing stage.

3 DATA PREPROCESSING AND AUGMENTATION

We addressed class imbalance by considering sampling strategies or model-related solutions. Although we attempted to oversample the minority class, our efforts using a GAN architecture failed. We resorted to undersampling the majority class with no additional data sources available.

We began by removing random samples from the majority class to balance it, seeing better results when we specifically eliminated closely related images using cosine similarity. However, cosine similarity isn't the most accurate way to measure image similarity because it relies on pixel-wise comparisons. Instead, Structural Integrity (SSI) offered a more effective method by assessing image similarity using a better algorithm. Our tests with SSI yielded similar results to cosine similarity and were carried out using the *imagededup* package.

Regarding augmentation, we explored various combinations, using standard transformations like Random Flip, Translation, Rotation, and Zoom from the *keras* package and additional techniques like Random Sharpness, Cutout, and Shear from *keras_cv*. Despite trying CutMix, no significant improvement was observed.

4 MODEL ARCHITECTURES

Below, we summarise the validation and test set accuracies for the different architectures we tested. Due to the computational expense of cross-validation, we opted for random validation and test sets, which might vary in size across models.

Our experiments involved various configurations. Initially, in transfer learning, we explored different setups involving the number of neurons, additional dense layers, and regularization techniques like L1L2 regularizer and dropout to prevent overfitting. We conducted trials using Adam and AdamW optimizers at a learning rate of 10^{-3}.

In the fine-tuning phase, we experimented by unfreezing different network blocks and adjusting the learning rate to 10^{-5}. Batch sizes of 32 and 64 were tested as well.

Table 1: Model Performance

Model	Validation Accuracy	Test Accuracy
Basic CNN [No Augmentation]	0.8675	0.7900
Basic CNN	0.8775	0.8150
MobileNetV2	0.7975	0.6260
EfficientNetB6	0.9220	0.8760
EfficientNetB7	0.8500	0.8310
ConvNeXtBase	0.9140	0.9140
ConvNeXtLarge	0.9180	0.9260

4.1 Basic CNN Models

Initially, we experimented with a basic model with 4 Convolutional layers using ReLU activation and Max Pooling, followed by a Global Average Pooling layer. The final layer comprised a dense layer with 2 neurons, employing categorical cross-entropy as the loss function. Despite achieving a test accuracy of 79%, this outcome, while not optimal, validated the reliability of our data preprocessing and splitting methods. We also extended the exploration by incorporating a data augmentation layer immediately after the input layer, increasing the accuracy to 81%.

4.2 Transfer Learning + Fine Tuning

To achieve better results, transfer learning and fine-tuning was the primary approach. The base models used are explained below.

4.2.1 MobileNetV2. We experimented with MobileNetV2, following class notebooks. However, the model's accuracy fell below expectations despite employing class weight balancing. Notably, it showed a strong bias toward predicting the majority class, "healthy," indicating extreme sensitivity to class imbalance.

4.2.2 EfficientNet. We assessed the performance of both B6 and B7 architectures, and unexpectedly, B6 outperformed B7 on the test set. Notably, undersampling the data might have favored a smaller network like B6 during this phase due to the reduced dataset. However, it's essential to highlight that B7 excelled with test-time augmentation, significantly enhancing accuracy from 83.0 to 88.0 on the test set.

4.2.3 ConvNeXt. Our experiments with ConvNeXtBase and ConvNeXtLarge showed differing outcomes. When using the distributed dataset, ConvNeXtLarge resulted in higher accuracy on both validation and test sets. However, during the 1st phase submission, ConvNeXtBase yielded an impressive accuracy of 90.0, marking the highest achievement in our models during that phase.

4.3 Explainability: CAM

We utilized CAM (Class Activation Mapping) as a visualization method to analyze the model's focus on image features. In Figure 4, the CAM outputs of EfficientNetB7 and ConvNeXt are displayed. Our observation of EfficientNetB7 revealed its tendency to classify healthy images by emphasizing corners and unhealthy images by highlighting the middle strip. Notably, an example on the right-most image, despite 99.9% confidence, misclassifies an unhealthy image as healthy by focusing on the corners. To address this issue, we proposed employing RandomCutout as an augmentation technique. Conversely, ConvNeXt's CAM output exhibited a more robust model focusing on relevant classification features. The same misclassified image on EfficientNetB7 was correctly identified as unhealthy with 100% confidence by ConvNeXt.

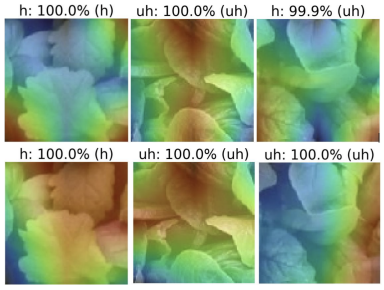


Figure 4: CAM outputs on EfficientNetB7(top) and ConvNeXt (bottom) with confidence score and true label in parentheses. (h: healthy, uh: unhealthy.)

5 OTHER TECHNIQUES

5.1 Class weights

Undersampling is a viable approach when there's an abundance of data, but that wasn't the case in our scenario. Therefore, an alternative method utilized was leveraging class weights. This technique involves assigning different weights to samples based on their labels, enabling *keras* classifiers to be trained accordingly. Interestingly, this approach produced results comparable to undersampling concerning the *AUC* metric.

5.2 Test-time augmentation

To enhance model performance, we implemented test-time augmentation. Our strategy involved generating six diverse augmentations of the input image: identity, rotation, translation, zoom, and horizontal and vertical flips. Subsequently, the outputs from these augmentations were averaged to produce the final classifier result. While this approach decreased performance in specific models, presumably due to their limited generalization ability, our top-performing models showcased a notable performance boost in the range of 3-5%.

5.3 Ensemble Approach

To further refine our model, we merged them to obtain an ensemble model. Like the TTA approach, we would obtain the prediction of several models and average the result for a final prediction – also known as a voting ensemble. This result improved our performance on the competition's test set by 1.5%.

6 BEST SUBMISSION

The best-performing model was an ensemble comprising three top-performing models from Phase 1 of the competition. The first model involved fine-tuning the ConvNeXt Large from layer 100 onward, utilizing all augmentations mentioned in Section 3. The second model utilized ConvNeXt Base from layer 280 onward, employing primary augmentations from Keras. The third model was a modified version of the first, incorporating outlier removal and eliminating similar images from the training set. All three models were trained using the Adam Optimizer with a 10^{-3} learning rate. The third model underwent further training with unfrozen layers from 250 onward, using a learning rate of 10^{-5}. A batch size of 64 was consistent across all models. The ensemble approach used the three models to make predictions through an equal-weight voting system on inferred inputs. Test-time augmentation, detailed in Subsection 5.2, was applied to all three models before voting. It achieved 82.7% accuracy on Codalab.

7 CONTRIBUTIONS

All team members were part of the process equally and involved in all project parts. In terms of particular contributions:

- Francesco focused more on the model training and test-time augmentation
- Nuno focused more on the data analysis and preprocessing, as well as the Ensemble approach
- Shodai focused more on model training, trying with various architectures and hyperparameters
- Rio focused more on the model explainability, as well as other approaches to training such as Contrastive Learning

REFERENCES

[1] G. Bradski. 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).

[2] François Chollet et al. 2015. Keras. https://keras.io.

[3] Nathan Hubens. 2019. Test time augmentation (TTA) and how to perform it with keras. https://towardsdatascience.com/test-time-augmentation-tta-and-how-to-perform-it-with-keras-4ac19b67fb4d

[4] Tanuj Jain, Christopher Lennan, Zubin John, and Dat Tran. 2019. Imagededup. https://github.com/idealo/imagededup.

[5] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. 2022. A ConvNet for the 2020s. arXiv:2201.03545 [cs.CV]

[6] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research* 12, Oct (2011), 2825–2830.

[7] Joseph Rocca. 2021. Ensemble methods: Bagging, boosting and stacking. https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205

[8] Mingxing Tan and Quoc V. Le. 2020. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. arXiv:1905.11946 [cs.LG]