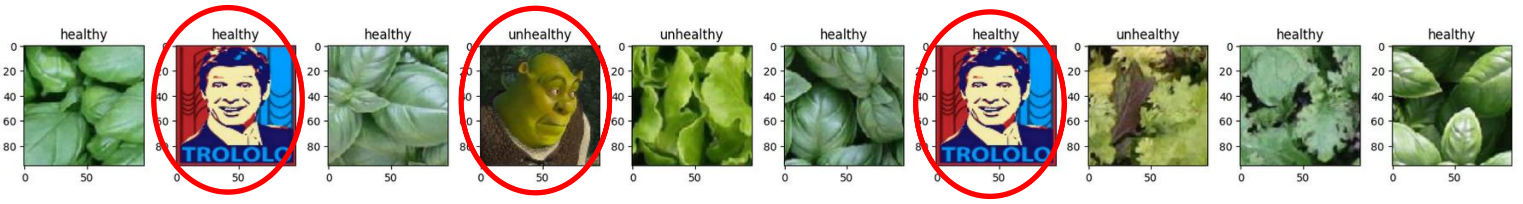
DATASET INSPECTION

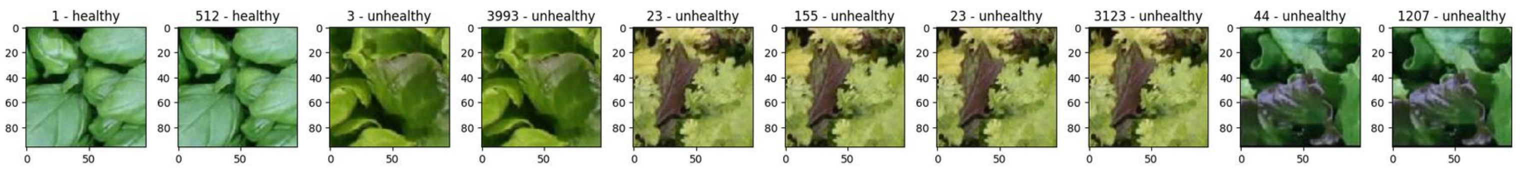
The provided dataset comprises **5200** RGB images, each with a resolution of 96x96 pixels, representing various leaves. Each leaf is accompanied by a corresponding label that classifies it as either healthy or unhealthy, necessitating a binary classification task. Notably, the dataset exhibits a deliberate class imbalance, with a higher representation of healthy leaves, amounting to **3199** samples, constituting approximately **61.52%** of the dataset, while the remaining **2001** samples represent unhealthy leaves, accounting for approximately **38.48%** of the dataset.

OUTLIERS

One of the initial tasks when working with a dataset is to generate plots of sample data to gain an understanding of the dataset's characteristics. By visualizing a subset of the dataset through sample plots, we can promptly identify any outliers or anomalies:



Another fact we noticed by inspecting the dataset is the presence of duplicates:



Next, we proceed to cleanse the dataset by eliminating outliers and duplicate entries: considering the above-mentioned operations, the "new" dataset consists of a total of **4850** samples, comprising **3060** healthy leaves and **1790** diseased leaves, which represent **63.09%** and **36.9%**, respectively.

Another analysis conducted as part of our preprocessing involves the identification of potentially mislabeled samples, those that significantly deviate from other samples within the same class. To achieve this, we employed K-Means with **k = 2** on embeddings extracted using ConvNeXtXLarge. We ensured that there were no points deviating excessively from their respective clusters. This analysis revealed that the data is accurately labeled, and there are no significantly dissimilar data points within the dataset.

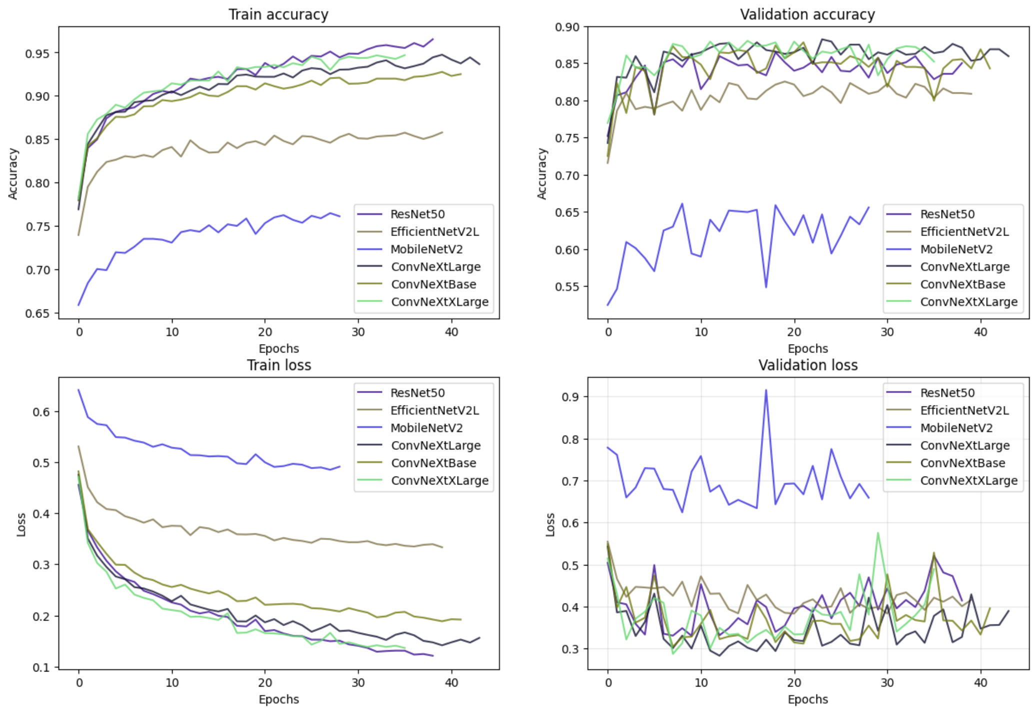
DEALING WITH DATA IMBALANCE

An important aspect of the preprocessing phase was addressing data imbalance. As previously mentioned in the section above, the dataset is inherently imbalanced, favoring healthy leaves. Primarily, it seemed appropriate to extract a balanced validation set (20% of the training set) to ensure unbiased evaluation of the accuracy metric. Furthermore, balancing the validation set did not lead to a drastic change in the distribution of labels in the training set, in fact, after balancing, we observed **~66.35%** healthy leaves and **~33.65%** diseased leaves.

We also tackled the problem of balancing the training set. For this purpose, we tested various approaches, including data augmentation with classic transformations (rotation, flip, etc.) and using the Synthetic Minority Over-sampling Technique (S.M.O.T.E) (N. V. Chawla, 2011). However, we ultimately chose to keep the training set imbalanced at the sample level and adjusted the weights assigned to each class, in the loss function. As a result of this weighting, the model gives more “importance” to errors in the classification of diseased leaves rather than healthy ones. The weights are calculated based on the classes distribution and are: **~0.7535** for the healthy class, and **~1.4862** for the unhealthy class. This latter technique yielded better performance.

FEM SELECTION

As the initial step in the model selection phase, we chose the optimal (pretrained) network for feature extraction. This selection involved comparing several pretrained models, including ResNet50 (Kaiming He, 2015), MobileNetV2 (Mark Sandler, 2018) (Mingxing Tan, 2021), EfficientNetV2L (Mingxing Tan, 2021), ConvNeXtBase, ConvNeXtLarge, and ConvNeXtXLarge (Zhuang Liu, 2022).



Upon reviewing the results, we chose **ConvNeXtXLarge** as it yielded the best performance.

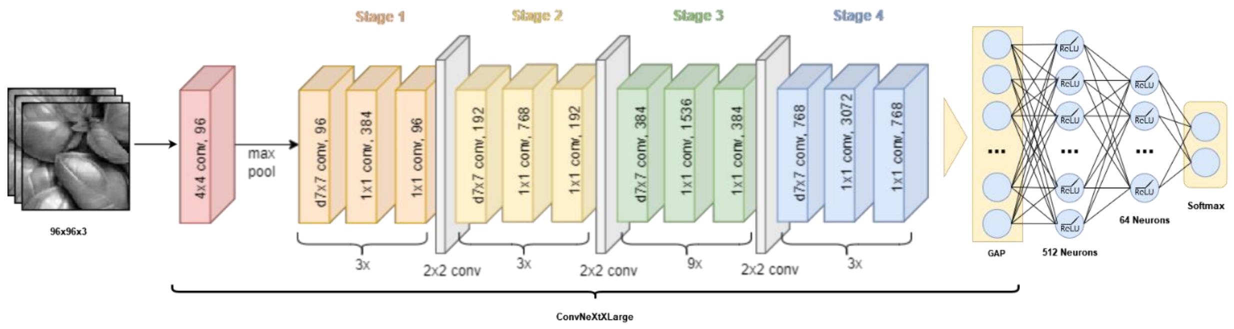
MODEL TUNING – TRANSFER LEARNING

Once the backbone network was selected, we proceeded with hyperparameter tuning of the model, specifically focusing on the FFNN attached to ConvNextXLarge. To accomplish this, we froze the weights of the FEM (**we performed transfer learning**) and employed algorithms for optimal parameter search.

NETWORK ARCHITECTURE SEARCH

As a primary step towards enhancing our model, various experiments were conducted to determine the optimal number of layers and neurons per layer. In pursuing this, two primary approaches were considered: one involved maintaining a consistent number of neurons across layers as we approached the output layer, while the other entailed adopting a funnel-shaped (or pyramidal) structure.

Through a GridSearch-based exploration, the outcome of this investigation revealed that the most suitable configuration for our network comprises two hidden layers consisting of 512 and 64 neurons respectively (activated by ReLu), followed by a Softmax activated output layer.



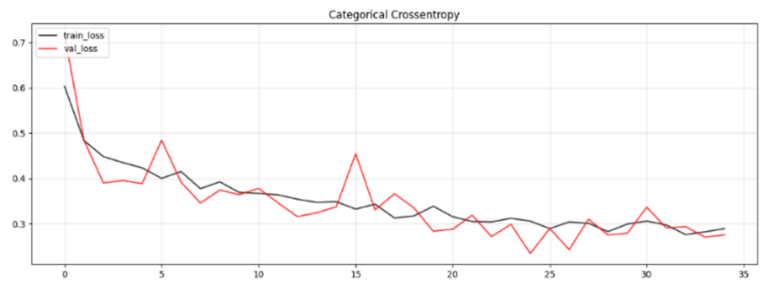
HYPER-PARAMETERS TUNING

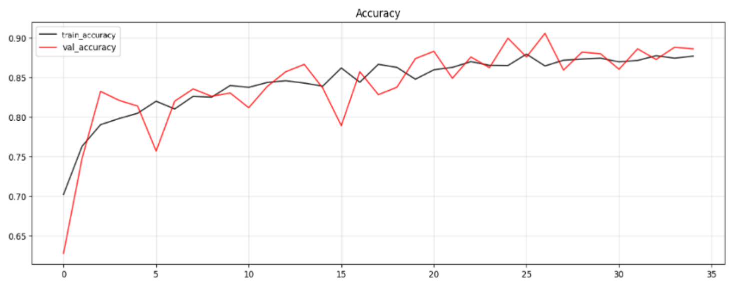
During the parameter tuning step, we consistently employed the grid search approach on the previously mentioned network configuration. The parameters under focus were the learning rate, batch size, and the λ parameter for L_2 regularization. This process led us to identify an optimal parameter configuration: learning rate (Adam optimizer) = 0.001, batch size = 16, and λ = 0.

The tuning process resulted in a model with good performances, notably achieving: Accuracy: 0.89 Recall: 0.90 Precision: 0.89 F1-score: 0.90. However, the model exhibited signs of overfitting. To address this, we deliberated on applying data augmentation techniques to mitigate the model's variance.

DATA AUGMENTATION

As introduced in the preceding section, to counter the overfitting observed in the model, in addition to the trials shown in the "Dealing with data imbalance" section, we opted to employ data augmentation techniques. To implement this, we introduced a preprocessing layer that incorporated various geometric transformations such as RandomFlip (vertical and horizontal), RandomRotation, and RandomTranslation. Through the application of data augmentation, we successfully removed any signs of overfitting while maintaining consistent performance, as evidenced by the graphs depicted below.





FINE TUNING THE MODEL

As the final phase in the model design pipeline, we endeavored to further enhance the already high performance using the fine-tuning technique. Consequently, **starting from** the transfer learning-trained model, we proceeded to unfreeze a portion of the backbone. As a starting point, we have decided to use 51 as the unfreeze threshold. Additionally, we adjusted the learning rate to $2e^{-5}$, a common practice in such scenarios. While these adjustments led to improved performances, they also reintroduced the phenomenon of overfitting. To deal with this phenomenon we tuned the geometric augmentation to find the optimal parameters through **Bayesian Optimization**, resulting in RandomRotation = 0.2, RandomTranslation = (0.2, 0.2) and a freezing threshold of 51, effectively enhancing the model's generalization capabilities. We also reconsidered regularization at this step, and so and we applied lambda $\lambda = 2e^{-3}$.

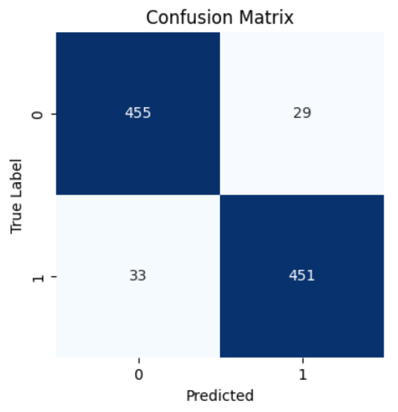
FURTHER IMPROVEMENTS

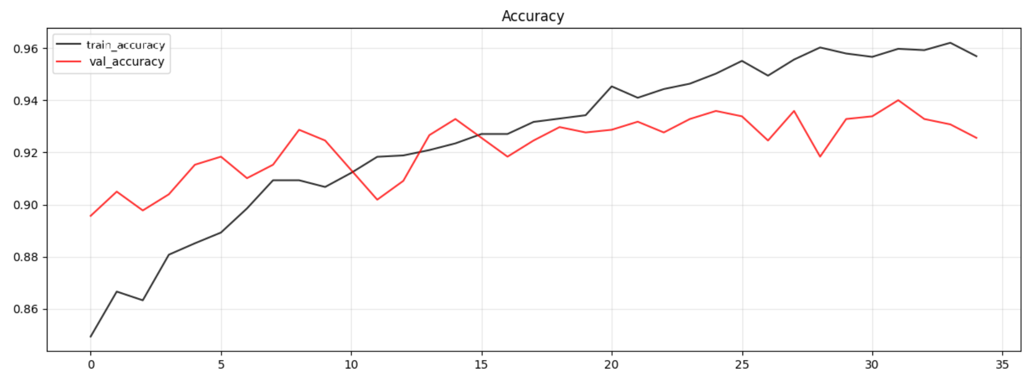
As we kick off the final phase, we've decided to further improve the model obtained in the first phase. To do this we've decided to conduct additional tests to make the loss and accuracy as consistent as possible between the training and validation sets. To achieve this, we experimented with amplifying the geometric transformations (RandomRotation = 0.3 and RandomTranslation = (0.4,0.6)) within the preprocessing layer and tested the dropout (dropout rate = 0.4) as a regularization method. This led us to further enhance the model and achieve the final performance outcomes. As shown in the notebook, we applied all these changes with TL and then we applied FT.

RESULTS

To summarize, our analyses led us to consider a model comprising a pretrained backbone (ConvNeXtXL) extended by a FFNN consisting of 2 hidden layers with 512 and 64 neurons respectively, and a softmax output layer. The final model was obtained via fine-tuning, keeping only the first 51 layers frozen. For regularization, Dropout was employed, and Data Augmentation was applied in a preprocessing layer considering only geometric transformations. The obtained performances on the final hidden test are obtained by training the model on all the data.

	ACCURACY	PRECISION	RECALL	F1-SCORE
VALIDATION	0.93	0.93	0.93	0.93
FINAL TEST	0.892	0.890	0.815	0.851





References

Chollet, F. keras. Retrieved from https://keras.io/

Kaiming He, X. Z. (2015). Deep Residual Learning for Image Recognition. Retrieved from https://arxiv.org/abs/1512.03385

Mark Sandler, A. H.-C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. Retrieved from https://arxiv.org/abs/1801.04381

Mingxing Tan, Q. V. (2021). EfficientNetV2: Smaller Models and Faster Training. Retrieved from https://arxiv.org/abs/2104.00298

N. V. Chawla, K. W. (2011). SMOTE: Synthetic Minority Over-sampling Technique. Retrieved from https://arxiv.org/abs/1106.1813

Zhuang Liu, H. M.-Y. (2022). A ConvNet for the 2020s. Retrieved from https://arxiv.org/abs/2201.03545

CONTRIBUTIONS

Andrea Bertogalli	***Niccolò Balestrieri***	***Nicolò Tombini***
Pre-processing: 60%	Pre-processing: 20%	Pre-processing: 20%
Model tuning: 20%	Model tuning: 20%	Model tuning: 60%
Fine tuning: 20%	Fine tuning: 60%	Fine tuning: 20%
Report: 33.3%	Report: 33.3%	Report: 33.3%
Notebook refactoring: 33.3%	Notebook refactoring: 33.3%	Notebook refactoring: 33.3%

**NOTE: all team members worked on all parts of the project.*