Artificial Neural Networks and Deep Learning

First Homework – Binary Classification

This first challenge consisted in a binary classification problem: classify images of plants as ‘Healthy’ or ‘Unhealthy’. We were provided with a single .zip file, containing 5200 96x96 RGB images and their associated labels.

Preprocessing

The first step to build the classifier is data inspection [1]. We proceeded to visually inspect the whole dataset. It was immediately noticeable that there were a few “sneaky” outliers within the dataset, so we applied manual inspection to detect all those images, supported by “Anti-Twin” tool usage to have a guaranteed clean dataset to work on. In the end the result was stored in a new file called “dataset_wo_duplicates.npz” which was then used as a dataset for the whole project.

Splitting and Setups

Then, we decided to settle for a 80/10/10 splitting of the images available, with the introduction of augmentation techniques and noticing inconsistencies between scores obtained in local and post submissions, we tried some different partitions, eventually agreeing on 60/20/20 for the final submitted model. Randomness for each experiment was fixed with a seed (also for other aspects rather than splitting, as follows in this report). Labels were one-hot encoded.

Image Augmentation

Being the number of images limited we almost immediately noticed that this would be a determinant factor, so we applied random translation (later dropped), rotation, flip and later contrast and brightness thus creating more data to train the model on and improve generalization but trying to keep enough information to distinguish the classes.
We can now proceed to think about model architecture. We started from the basics: a CNN based on the laboratory lessons, then proceeded with Transfer Learning and subsequently Fine Tuning.

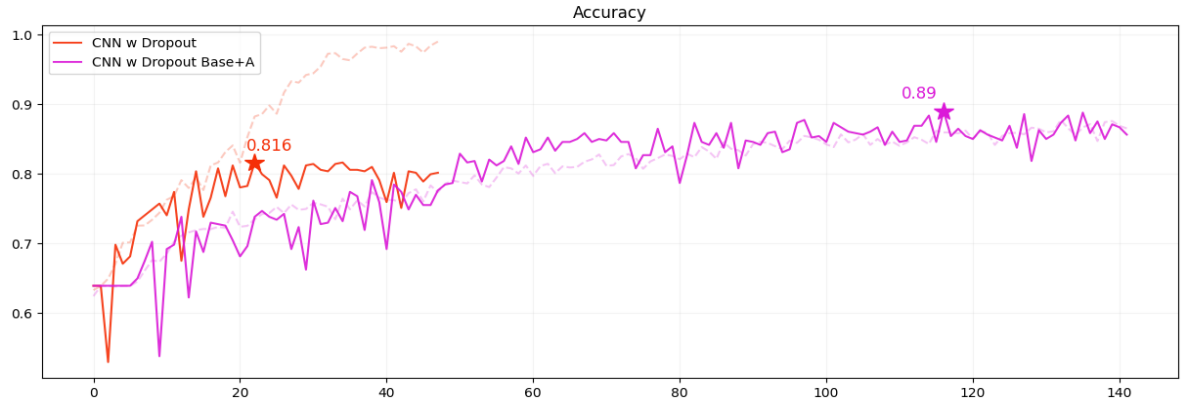
Custom CNN

The very first attempt to tackle the challenge was referring to the laboratory lectures: a model made by 5 convolutional layers, starting from 32 filters and doubling at each block, composed by the convolutional layer itself, ReLU activation, MaxPooling2D and a Dropout layer. Dropout rate was set to a low one (= 1/8) after stating that higher values prevented the model from learning effectively. Following these 5 blocks, a GAP layer densely connected the output with 2 units and softmax activation, this was chosen because we are dealing with binary classification. Additionally, to prevent overfitting, during every phase of the work plan

we employed EarlyStopping, with patience adjusted accordingly on the converging speed and training time per epoch of the tested model, this approach resulted in more generalized models.

Other techniques:

Together with dropout layers, we tried to include regularization using both L1 and L2 singularly and L1L2, trying different values for lambda: decent results were obtained with a value of 0.0075 using L1L2 and reducing the learning rate of Adam optimizer function, although overall this couldn't better the score of 0.68 obtained with the previous model. Adding Image Augmentation had a significant impact on local score but the difference, at least in this Custom CNN approach, was barely enough to get 0.69/0.70 on the hidden test set, convincing the whole team to take the next step, exploiting existing pretrained nets.



Transfer Learning and Fine Tuning

Remembering that the dataset was unbalanced, we had 3028 "Healthy" and 1711 "Unhealthy", so we added to the model.fit the class weights calculated according to class distribution, thus giving more weight to the less represented class.
In order to proficiently use TL/FT techniques we referenced Keras documentation [2].
Our approach was to use SOTA pre-trained models for automatic feature extraction (AFE) followed by a custom MLP for classification. The overall pipeline: input->augmentation->AFE->classification->output. During TL the whole AFE section was set to not be learnable, hence training only the classification part. During FT some layers in the AFE were unfrozen and consequently trained.

Xception

Due to the incapability of the CNN to increase performances, we swapped to a TL technique, keeping augmentation, we could choose in a variety of base models, but according to some thesis, the Xception model could have the high performance in a binary classification problem [3][4]. With this approach we couldn't reach accuracy over 0.78. We tried increasing batch size, decreasing patience. For FT, we freezed 46 layers, and we arrived at 0.86, even though we barely reached 0.83 on the hidden test set.

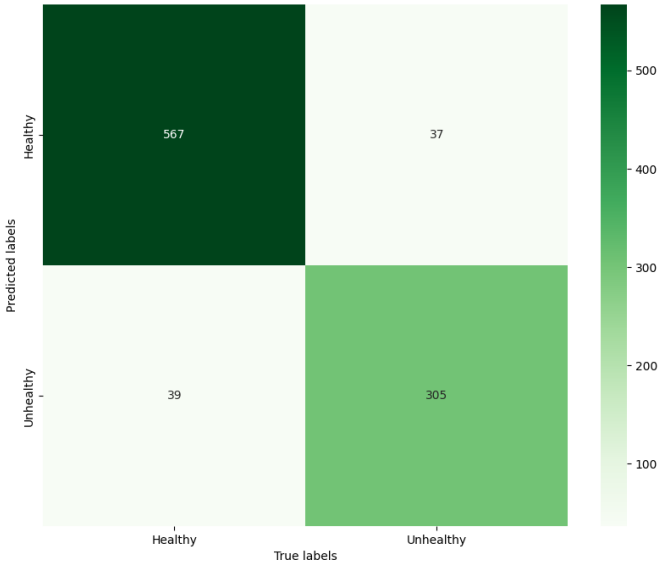
EfficientNet

In literature we found that for the task of binary image classification the best results are achieved through the family of EfficientNets [5].
We experimented with different architectures according to a good balance between number of parameters to learn and reported performance on the ImageNet dataset classification, gradually increasing model complexity.
In the context of FT EfficientNets we set layers beyond block 6 (included) to be learnable, since the first blocks are responsible for extracting low-level features (such as edge detection) there was no point in learning them again. Going instead deeper in the network, extracted features get more high-level and thus had to be learned again to fit the model on our specific task. Following, some results noticed while testing various EfficientNet applications:

- B4 had the best recall score among all models even on the hidden test set (with 0.86 accuracy) but low precision, indicating it was biased towards the “Unhealthy” class.
- B5 and B6 achieved the worst performances overall because they seemed to have learned the class distribution.
- V2S and V2M achieved the best performances among the family, respectively 0.85 and 0.88 on the hidden test set.

ConvNeXt

Following a similar approach with another family of networks: ConvNeXts. We tried with the Base model since it has size and performances similar to the V2M, then the Large model. During FT layers beyond block 18 (included) were set to be learnable for the same reasons stated before.
Interestingly the Base model outperformed the Large one achieving respectively 0.90 and 0.89 accuracy on the hidden test set, despite the latter having double the parameters of the former. Complexity doesn’t automatically translate to better performances.



Confusion matrix for the predictions on the test set of the ConvNeXt Base model after FT

Conclusion

In conclusion the model achieving the best results were achieved with the ConvNeXt Base model on both phases of the challenge.
Possible improvements implementing: K-fold cross validation, noisy student approach with relaxed labels, different architectures and hyperparameters, ensembling models together, using data generators to increase the size of the dataset (eventually other augmentations).

Contributions:

Overall we didn't decide to split tasks a priori, so everyone participated in every step of the task. In any case, these below are the aspects that each one of us feel like he focused on more during this first competition homework.

- *Luca Di Stefano*: Experiment Execution, Model Architecture Design, Xception TL/FT
- *Gabriele Romano*: Data Preprocessing, Experiment Execution, Xception TL/FT
- *Mattia Pezzano*: Data Inspection, Experiment Execution, Model Architecture Design, EfficientNet and ConvNeXt TL/FT, Report Tweaks
- *Riccardo Villa*: Initial Model Architecture (Custom CNN), General Hyperparameter Tuning

References:

1. https://karpathy.github.io/2019/04/25/recipe/ Tips for Preprocessing
2. https://keras.io/api/ Keras Documentation
3. https://arxiv.org/ftp/arxiv/papers/2002/2002.04189.pdf#:~:text=Given%20this%20particular%20two%2Dstage,a%20mean%2Dsquared%20loss%20function. Thesis in support of Xception
4. Fan Y-J, Tzeng I-S, Huang Y-S, Hsu Y-Y, Wei B-C, Hung S-T, Cheng Y-L. Machine Learning: Using Xception, a Deep Convolutional Neural Network Architecture, to Implement Pectus Excavatum Diagnostic Tool from Frontal-View Chest X-rays. *Biomedicines*. 2023; 11(3):760. https://doi.org/10.3390/biomedicines11030760
5. Marques G, Agarwal D, de la Torre Díez I. "Automated medical diagnosis of COVID-19 through EfficientNet convolutional neural network". Appl Soft Comput. 2020 Nov;96:106691. doi: 10.1016/j.asoc.2020.106691. Epub 2020 Aug 29. PMID: 33519327; PMCID: PMC7836808. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7836808/