Convolutional Neural Network Challenge
Plants Binary Classification
Artificial Neural Networks and Deep Learning
Geoinformatics Engineering
Politecnico di Milano

Lorenzo Carlassara, Ellen Poli, Matteo Gobbi Frattini, Zhongyou Liang
Student IDs : 10601118, 10728490, 10581031, 10905937
Nicknames: lorenzo.c, EllenPoli, MatteoG25, zyl
Team: Geo Palle

1. Data Exploration

The provided dataset contains 5200 images of plants divided in two classes: healthy and unhealthy. With a preliminary analysis we find out that the class distribution is slighty unbalanced, divided in 2001 images labeled unhealthy and 3199 healthy. We decide not to implement any class weighting.

2. Preprocessing

The first preprocessing procedure is the normalization from the RGB [0;255] to values ranging in [0;1] if required by the Network implemented. For example this procedure is already included in some keras applications networks (e.g. EfficientNetV2B3) other networks, instead, is required the ad-hoc preprocess input.

2.1. Outliers

We notice the presence of some pictures of ”Trololo” and ”Shrek”, these very strange plants images are considered as outliers. Taken two different samples, we identify all the indices of the images with a structural similarity index within a threshold of 85% and 35%. All the images related to the identified indices have been afterwards removed using a more efficient algorithm, in order of time computing, which is based on the hash table of each image (Figure 1). Moreover, we notice also the presence of equal multiple images that we decide to remove to avoid the overfitting on the training set and also reduce train-validation overlap. This, later, make us think that maybe can be overcame by augmentation techniques.

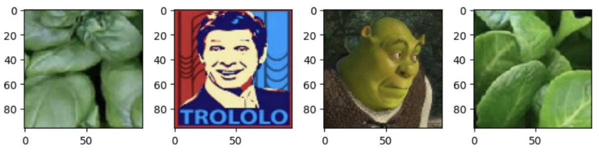


Figure 1: *Examples of two outliers (2nd and 3rd picture) with respect to dataset images (1st and 4th picture)*

2.2. Augmentation

The data-set can be characterized as moderately-sized and comparing it to the complexity of the possible architectures the data augmentation can be introduced to increase the dimension of the data-set. This approach aims to highlight different features within the images, ultimately contributing to a more generalized learning process and improved model performance in fact usually data augmentation is commonly used to reduce overfitting and improve the classification performance. [6]
Since inside the data-set there is also the presence of duplicates that we removed, in order to balance the dataset decreasing size, the augmentation technique becomes even more relevant. We decide to try out

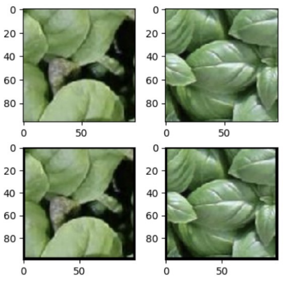
different combinations of augmentation to verify which one fits better on the data-set. From this we infer that some augmentation improves the accuracy on the validation and testing set. In particular we tried two different combination2.2:

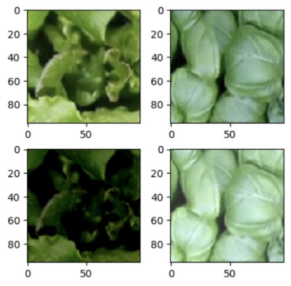
- **Combination 1:** RandomFlip + Crop + ZeroPadding
- **Combination 2:** RandomFlip + Crop + ZeroPadding + Brightness + Contrast

Combination 1	**95%**
No Augmentation	90%
Combination 2	89%

Against the expectation the Combination 2 do not increased the validation performance with respect to the performance without augmentation or with the Combination 1. Therefore we tried the same augmentation also on the following tested models.

Two of the tried augmentation are show in the images below.


(a) ZeroPadding


(b) Brightness

2.3. Splitting

We strategically determine the sizes of the training, validation, and test sets in alignment with the chosen batch size. This ensures a uniform distribution of data across batches, guaranteeing a consistent amount of data in each batch, including the last one. During the data splitting, in order to keep an homogeneous distribution between training, validation and testing set we use a Stratified Sampling technique provided by scikit-learn.

3. Neural Network Architectures

We decide to use as loss function the Categorical Crossentropy feeding the two label classes in a one-hot representation.[2], and to evaluate the goodness of the model we monitor as metrics the accuracy.

For the activation function selection we try both Softmax and Sigmoid functions, but we conclude that there are not evident reasons to choose one with respect the other. We keep Softmax.

The head/top of pretrained neural networks is built in two different ways in the model used for the development phase and the model used in the final phase with fine tuning. In both phases we use transfer learning.

3.1. Models Comparison

The first approach to the task starts from simple and popular architectures presented during the course (e.g. quasiVGG9) till ending up with the ones listed below. The initial aim, while getting familiar with the convolutional network blocks, was to prevent GPU memory exhaustion. This lead us to explore the utilization of lighter models, allowing for resource conservation. However, this approach come with its own set of trade-offs, as lighter models often sacrificed complexity and expressiveness in favor of computational efficiency. In order to train and try larger networks we decide to implement the TPU connection strategy which allows the same performance of the GPU in colab.

Once noticed underwhelming values of accuracy, we decide to move on to different techniques such as using pre-trained nets [3] with transfer learning and fine tuning [7]. Using this bigger architectures we face some issues of OOM (out-of-memory). To avoid this we introduce some techniques to reduce the use of the memory and to run faster. To speed up the training we introduce the mixed precision with this purpose. [5]

The architectures that we try are the following:

- ResNet
- VGG16 and 19 and other custom VGG-like architectures.
- EfficientNetB1, B2 and B3
- EfficientNetV2B0, V2B3 and V2S
- ConvNeXtXSmall, ConvNeXtLarge and ConvNeXtXLarge

From the validation test accuracy we find out that the VGG-like were not good as the EfficientNet architectures. Thus, we decided to compare different EfficientNet and, among them, the best results were obtained from the V2B0 and V2B3. The good performance of the EfficintNetV2B0 is shown also through the confusion matrix in *Figure 2* with an accuracy of 91% in the validation set. In particular as shown in *Figure 3* the most performant is the EfficinetNetV2B3 [1] that reached a value of 96% in validation accuracy. The other nets can't generalize enough and they don't perform as well as this one on the test set and they are not able to reach a higher peak of performance in both loss and metric. Thus we decided to keep the EfficinetNetV2B3.

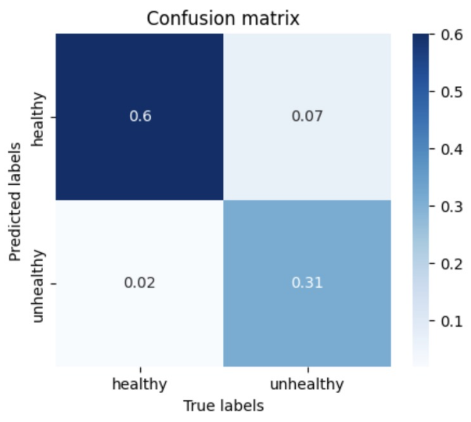


Figure 2: *Confusion Matrix of EfficientNetV2B0*

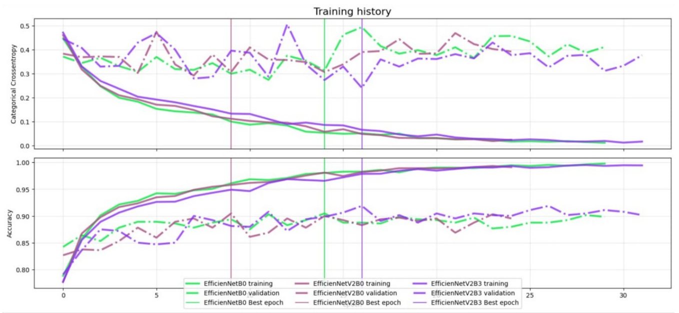


Figure 3: *Comparison between Crossentropy and Accuracy of different EfficientNet architectures*

4. Development Phase: EfficinetNetV2B3

EfficientNetV2 is a new family of convolutional networks that have faster training speed and better parameter efficiency than previous models. [4] During the development phase, we employ EfficientNetV2B3 which is an image classification model with 1000 classes to which we append a single dense layer. We then make EfficientNetV2B3 trainable as simple transfer learning by fixing the weights of the classifier would be too unflexible regarding feature extraction. The pre-training of EfficientNetV2B3 ensures that its weights are only lightly modified with the steepest gradients being in the dense layer. This justifies the relatively low learning rate and the exponential scheduling.

5. Final Phase: Fine Tuning Deployment

In order to reach a further improvement in the final Networks we consider the Transfer Learning technique followed by Fine Tuning with large keras applications CNNs. During the development phase, against our expectations, the best validation accuracy reached within this framework was not as high as the validation accuracy provided by the EfficientNetV2B3. With a more accurate tuning procedure and improving the top of the model, we suppose it could increase the accuracy provided by the Fine Tuning technique. When try out the largest family architecture, i.e. ConvNeXt, the local test accuracy take off up to 0.98.

6. Conclusion

While training larger models we face some GPU Ram memory exceeding, that we manage to solve only reducing the batch size of the dataset. Or selecting a smaller Network from the same family.

Once reached high accuracy values, one of the biggest issues is the large gap between the local test accuracy on colab and the one obtained after the codalab submission. This does not guarantee us that a good performing model was perfmorming well in the codalab test set.

We face some memory issue while

During the final phase, one attempt is to add some gaussian noise on the dataset in order to let the model generalize well and prevent overfitting.

Contribution:

1) **lorenzo.c** implement and adapt different networks e.g. ConvNeXtLarge with Fine Tuning (final phase), EfficientNetV2B3 (development phase) as well as other networks we did not implement VGG-like architectures and EfficientNetLiteB0. Report writing.
2) **EllenPoli** dataset preprocessing, implement and adapt different networks e.g. EfficientNetB0, VGG19, CNN. Try and tune Augmentation techniques. Report writing.
3) **MatteoG25** dataset preprocessing, improving outliers detection as well as duplicates removals with image hash. Try but decide not to implement hyperparameters tuning via genetic algorithms.
4) **zyl** exploring new features: sigmoid implementation vs Softmax. Adding Gaussian Noise on the data before training. Trying different hyperparameters setup in order to better tune the networks.

References

[1] *EfficientNetV2 B0 to B3 and S, M, L.* https://keras.io/api/applications/efficientnet_v2/efficientnetv2b3-function. Accessed: 2023-11-17.

[2] *Keras Application.* https://www.tensorflow.org/api_docs/python/tf/keras/losses/CategoricalCrossentropy. Accessed: 2023-11-17.

[3] *Keras Application.* https://keras.io/api/applications/. Accessed: 2023-11-17.

[4] Quoc V. Le Mingxing Tan. *EfficientNetV2: Smaller Models and Faster Training.* https://arxiv.org/abs/2104.00298. 2021.

[5] *Mixed Precision.* https://keras.io/api/mixed_precision/. Accessed: 2023-11-17.

[6] Matheus Gutoski Leandro Takeshi Hattori Nelson Marcelo Romero Aquino1 and Heitor Silverio Lopes. *The Effect of Data Augmentation on the Performance of Convolutional Neural Networks.* https://www.researchgate.net/publication/321081886_The_Effect_of_Data_Augmentation_on_the_Performance_of_Convolutional_Neural_Networks. October 2017.

[7] *Transfer learning fine-tuning.* https://keras.io/guides/transfer_learning/. Accessed: 2023-11-17.