POLITECNICO
MILANO 1863

ARTIFICIAL NEURAL NETWORKS AND DEEP LEARNING

Report competition

AI Army

Thomas Pezda, Tobias Löwenthal and Emile Vandenbussche

Year 2023-2024

Problem

In this assignment, the objective is to implement a binary classification on plant images. This involves categorizing the images into 'healthy' or 'unhealthy'.

Start

In the project's early stages, we start with establishing a collaborative workspace. Utilizing a basic CNN model, we familiarized ourselves with the dataset, the submission process, and explored the images. Overcoming various submission challenges, we were ready to work on our model.

Data Augmentation and Labels balance

The next step was to work on our data. Data augmentation emerged as an important step in this process. Initial experimentation involved applying simple transformations to the images. Surprisingly, the initial results exhibited a decline in accuracy. Certain augmentations, like adjustments to brightness, adversely affected the model's performance: it obscured disease-related features. In response, we only retained flip, rotation, and translation—transformations deemed most meaningful for our plant images. Post-refinement, the impact of data augmentation on our results was remarkable. While accuracy saw a modest improvement, the more significant contribution played in mitigating overfitting. The histories showed a join of training and validation curves.

We also tried to balance the ratio in our data set, with random oversampling[1] and then distance-based undersampling. Unfortunately, none of the techniques showed great results on the competition dataset.

Data cleaning

To make sure there were no duplicates in the data, the numpy.unique() function was used on the data so that only distinct pictures remained. While inspecting random images, non-leaf pictures, such as Trololo, unexpectedly surfaced. An image retrieval model based on MobileNetV2 was constructed to identify and remove outliers, avoiding manual exploration of the extensive dataset. The outliers, Trololo and Shrek, were identified by the model and deleted from the data.

Subsequently, some more research was done on the duplicates in the data. The numpy.unique() function exclusively considers images for duplicate removal and does not address label disparities. To address this, a function systematically checked for duplicates with differing labels using a double for-loop. However, no such duplicates were found, confirming the suitability of using numpy.unique() for duplicate removal.

Transfer Learning

It was quickly determined that using transfer learning would give us the best results for this competition. Despite our enthusiasm as aspiring deep learning students, the architectures available online are crafted by researchers possessing significantly more expertise in the field than we currently do.

Additionally, our dataset is relatively small. Making use of transfer learning will thus greatly help our score.

The first transfer model used is the MobileNetV2 from the lab. This already gave us better results than before with an accuracy score of 0.78. Afterwards, more models were explored on Keras Applications. All EfficientNet, EfficientNetV2, and ConvNeXt models were trained on our dataset. The model selection process involved evaluating local test scores, and ultimately, the ConvNeXt models emerged as the top performers [2]. Notably, ConvNeXtBase, ConvNeXtLarge, and ConvNeXtXLarge yielded comparable results. ConvNeXtBase stood out as the preferred choice, thanks to its smaller size and thus superior efficiency in training speed without compromising on performance.

Once we had found the pre-trained model most appropriate for our task, namely ConvNeXtBase, it was time to decide the structure of our network. So far we had arbitrarily decided to only add one classification layer after the frozen pre-trained model, with two neurons and a softmax activation. However, ConvNeXtBase has been trained to label all kinds of images on what they contain, and not specifically to predict the health level of plants. In other words, while it would have been very good at recognizing our images as plants, it was not certain that it could have categorized them as healthy or unhealthy. Therefore, to make our network more task-specific, it was necessary to add some more dense layers in between, as is often the case in transfer learning. The question was how many and with how many neurons. To do that we performed a grid-search on those parameters, with an additional dropout parameter to avoid overfitting, and discovered that the best trade-off was one layer with 256 neurons and a drop-out layer. As the Gaussian error linear unit (GELU) activation function is used in ConvNeXtBase, we use this function for the hidden dense layer. The AdamW optimizer is used to train the model for the same reason.

Upon finalizing the model structure, we initiated the fine-tuning phase. Fine-tuning happens in two stages. Initially, we unfreeze the last half, corresponding to layers 149-295, and retrain the network with these layers unfrozen, employing a notably low learning rate. Subsequently, all layers are unfrozen, and the network undergoes full retraining. This improves the local test score to 0.93.

Ensemble learning

To prevent overfitting, we formed an ensemble of three models. Each model trained on a different split of the dataset with all previous techniques, introducing diversity to their learning processes.

Additionally, a bagging approach, involving the use of bootstrap samples, was considered to increase diversity among the models. However, due to the substantial time required for training a single model, we limited to three models. Unfortunately, this approach yielded a suboptimal test score, rendering it insufficient for effective bagging. Therefore bagging was not employed in the composition of the final ensemble. Although K-fold validation is a viable option to achieve more distinct splits too, regrettably, we lacked the time necessary to train models using this method.

Another idea, concerning a two-way ensemble of our two best pre-trained models so far (namely ConvNeXt and EfficientNet), was also introduced and explored. Two pre-trained and fine-tuned model instances of transfer learning from those were used, and the output of the two was injected in a small neural network, with a 4-neurons weighting trainable layer and an output layer. The idea was to train the network in taking the best out of the two models. However, the end accuracy was only a bit better than ConvNeXt on its own, so we decided to use Occam's razor principle and give up that idea to make our final model architecture simpler.

Post-training, a distinctive ensemble approach was employed during predictions. The three trained models collectively evaluated each image, and the final prediction was determined through a voting system. Interestingly, about 7% of the pictures had different outputs between the models. This shows that using multiple models together can help them understand different things and stop them from focusing too much on tiny details. This mix of ideas makes our plant classification system stronger.

Conclusion

In the end, after a lot of research, reasoning and a bit of trial and error, we came up with the following final architecture:

Model: "final_model"

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 96, 96, 3)]	0
preprocessing (Sequential)	(None, 96, 96, 3)	0
convnext_base (Functional)	(None, 1024)	87566464
dropout (Dropout)	(None, 1024)	0
dense1 (Dense)	(None, 256)	262400
dense1_dropout (Dropout)	(None, 256)	0
output (Dense)	(None, 2)	514

Total params: 87829378 (335.04 MB)
Trainable params: 87829378 (335.04 MB)
Non-trainable params: 0 (0.00 Byte)

Figure 1: Model architecture

This, as was said earlier, is the architecture of a single ConvNeXt-based fine-tuned model instance, and we use three in our ensemble technique with a voting system.

With a final accuracy of 0.88 on Codalab, we believe our model is rather effective. We wanted to understand how it made its decision, so we drew their class activation maps:

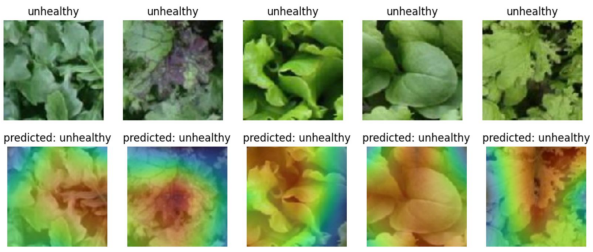

Figure 2: Class Activation Maps

As you can see, the model focus seems to make sense. In the second picture, for example, the focus is on the purple part of the plant, which probably tells a lot about the plant's health. We noticed that the model was often focusing on parts of the plants that were not green while avoiding the parts where there were no leaves at all. However, it sometimes directed its focus on the edges of the pictures, which was hard to understand.

Overall, we believe that we were successful in developing a neural network that made sense both in its architecture and in its "reasoning", and that was pretty effective with the task at hand.

Contributions

The data cleaning was done by Emile. He also did the research on which model we should use for Transfer Learning, performed the fine-tuning of the final model and worked on the ensemble. Thomas worked on data augmentation, balance of the dataset and the ensemble method. Tobias worked on the grid search used to select the best network architecture, on class activation maps and on one ensemble technique. He also researched self-supervised and contrastive learning, but that was not used in the end.

References

[1] J. Brownlee. (2021) Undersampling algorithms for imbalanced classification. Machine Learning Mastery, January 27, 2021. [Online]. Available: https://machinelearningmastery.com/undersampling-algorithms-for-imbalanced-classification/

[2] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," 2022.