Artificial Neural Network and Deep Learning Challenge 1 Report

Rishabh Tiwari [Icon1c], 10987397, rishabh.tiwari@mail.polimi.it
Erik Gustav Malmsten [gmalmsten], 10992793, erikgustav.malmsten@mail.polimi.it
Nicolas Filip Johansson [nicolasj], 11005901, nicolasfilip.johansson@polimi.it
Sophie-Claire Antoun [sophieantoun], 10989152, sophieclaire.antoun@mail.polimi.it

M.Sc. Computer Science and Engineering
Team Name: The Avengers

1 Introduction

In this project, we tackle the binary classification of plant health using a dataset comprising RGB images stored as a NumPy array with dimensions 5200x96x96x3. By applying advanced neural network methodologies, we aim to accurately distinguish between healthy and unhealthy plants, a binary categorisation that has significant implications for agricultural management and plant disease diagnosis.

Our initial objective during the project was to construct a rudimentary Convolutional Neural Network (CNN) specifically designed for the purpose of image classification. This CNN was composed of numerous layers, beginning with convolutional blocks that extracted image features using progressively increasing filters. Subsequent to this, MaxPooling and Global Average Pooling layers were applied to reduce dimensionality and condense the feature maps, respectively. Despite its efficient design, which incorporated Batch Normalization and LeakyReLU to facilitate learning, the fundamental CNN model encountered obstacles in attaining the intended levels of precision and generalization when applied to our dataset.

In light of these constraints, our group shifted its focus to investigating more advanced models. We extensively explored sophisticated architectures such as DenseNet, ResNet, and later ConvNeXt; models renowned for their feature extraction and streamlined processing of intricate image data.

Figure 1: Example of plants categorized into healthy and unhealthy states.

2 Data Pipeline

As the original dataset contained 5200 RGB images, consisting of 5000 images of plants classified as either healthy(3101 images) or unhealthy(1899 images) and 200 images of Shrek and the Trololo man, measures were needed to clean the dataset as well as tackle the problem of the class imbalance in the dataset.

2.1 Data Cleaning

The dataset was cleaned using two different methods, one based on removing outliers in a down-dimensionalised feature space(2), and the other consisting simply of manual removal of apparent outliers.

The features of each picture were retrieved by passing it through a pretrained large convolutional model stripped of the classifying dense layers. The resulting high-dimensional feature vectors were then down-dimensionalised to two dimensions using PCA analysis followed by t-SNE. In this feature space, clusters were identified using both the DBSCAN algorithm and manual removal of small clusters. Both these approaches were unsuccessful by either removing non-apparent outlier images or failing to remove apparent outlier images and was thus discarded.

However, the small size of the dataset made manual cleaning of the dataset a realistic option, which was quickly done and used throughout the remainder of the challenge. It is noteworthy that in the case of mislabelled images being present in the dataset, this manual cleaning would not be sufficient. One should then instead tune the clustering algorithm better, or explore alternatives to obtain a reliable result. We thus **assume that all plant images are labelled correctly.**

2.2 Training

During the training phase, two methods related to the weaknesses of the dataset were used to avoid the model overfitting; data augmentation and class weighting(1).

2.2.1 Data Augmentation

While the size of the dataset proved advantageous for manual data cleaning, it poses challenges for the training of the large amount of parameters in the large models. To address this issue, real-time data augmentation was employed throughout the training process using the ImageDataGenerator class from Keras. This approach exposes the model to randomly augmented images during training, effectively simulating new, unseen, images. The hyperparameters for the data augmentation were optimized using a grid-like random search, as detailed later in this discussion.

2.2.2 Class weighting

There exists a variety of methods for dealing with imbalances in class distributions. One approach is amplifying the relative weighing of the losses of the smaller classes in the loss function, and thus mitigating the impact of class imbalances during model training. This method is easily implemented using a lookup table with weights obtained from the sklearn.utils.class_weight class in combination with the hyperparameter class_weight availible in the model.fit function and was thus used in the training of the final model (3). Other approaches such as equalising the class distribution in the dataset before training and oversampling the minority class were also tested without success.

3 Transfer Learning and Fine Tuning

It was eventually realised that creating a customized CNN is a rather difficult task and not feasible during such a short time frame. The focus was therefore shifted towards transfer learning, which provides the opportunity to reuse scientifically proven, pre-trained, models. Transfer learning allows for faster training, better performance (in most cases), and training with data scarcity. We experimented with several different architectures, such as DenseNet121, ResNet50 and ConvNeXt.

3.1 Feature Extraction Network

The final model utilizes the FEN of the ConvNeXtLarge model, pretrained on imagenet. The specific architecture was choosen because its high performance on the aforementioned dataset. We also considered using the larger version (ConvNeXtXLarge), but the performance was largely equivalent, despite the significant increased computational intensity.

3.2 Classifier

Our classifier is applied directly after the FEN and consists of a Gloabal Average Pooling (GAP) layer followed by a single fully connected layer of two output neurons. The GAP averages spatial information to focus on classification-relevant features. This results in efficient feature summarisation, fewer model parameters, and reduction in overfitting risks. For the output layer, we use softmax activation in order to get outputs corresponding to the propability to each class.

3.3 Training process

The training process is split in two parts. Both parts are guided by the EarlyStopping callback, in order to prevent overfitting and save the best-performing model with respect to the validation loss. Since the model has two outputs, CategoricalCrossentropy is used as loss function. The choosen optimizer is Adam, due to its general good performance, adaptive learning rate and fast convergence.

First, transfer learning is performed by having all layers of the FEN frozen, only improving the accuracy of the classification layer. Then, all layers are unfrozen, in order to fine tune the FEN to better extract the features of the plant dataset. As mentioned, the training process was employed with an ImageDataGenerator, which feeds randomly augmented data into the model, together with the original training data. The original result of the training (non-optimised hyperparameters) is demonstrated in figures 2 and 3.

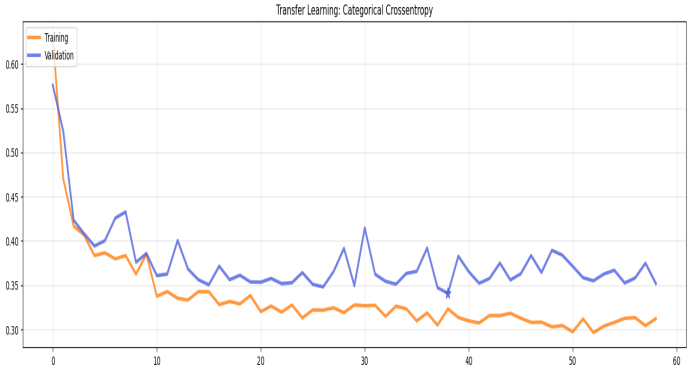

Figure 2: Transfer learning validation loss over time.

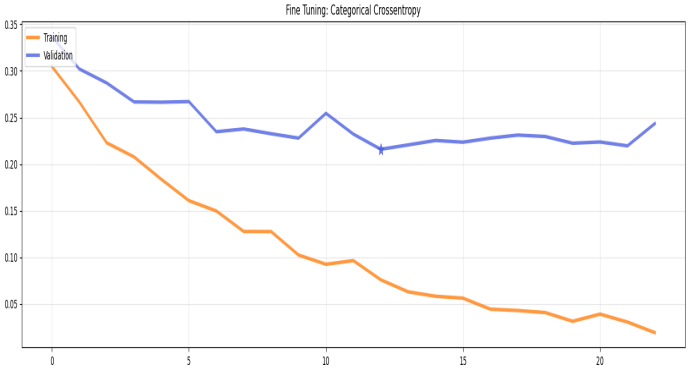

Figure 3: Fine tuning validation loss over time.

4 Hyperparameter optimisation

The last part of the development consisted of choosing the best parameters for the model. This step was mostly focused on the data augmentation parameters, but also covered the batch size.

The first idea was to use a more sophisticated method like k-fold cross validation or grid search. However, this was not feasible due to time consuming training and exhausted VRAM errors. Instead, the hyperparameter tuning consisted of simply, repeatedly, randomizing the parameters, training, and saving the statistics.

In order to evaluate the performance of each obtained setting, the (validation) cross-categorical entropy loss was monitored. The reason for preffering loss over accuracy, is simply because this metric tells us how close the network is to the correct output (rather than possibly having good accuracy due to lucky decision boundaries).

Contributions

- Rishabh Tiwari - Creating and running the initial tests on ResNet, EfficientNet, DenseNet from scratch.
- Nicolas - Built the MarkV (ConvNeXtLarge transfer learning) model and performed hyperparameter optimization.
- Sophie - Developed Basic CNN Model
- Gustav Malmsten - Worked on the data pipeline

References

[1] TensorFlow Authors. Handle imbalanced data with tensorflow. https://www.tensorflow.org/tutorials/structured_data/imbalanced_data, Accessed 2023-11-14.

[2] Iorio Paolino. Anomaly detection and clustering: A pca and t-sne approach, Aug 2018. URL: https://medium.com/@ioriopaolino62/anomaly-detection-and-clustering-a-pca-and-t-sne-approach-dbd4f57771cd.

[3] Scikit-learn. Scikit-learn - compute_class_weight. https://scikit-learn.org/stable/modules/generated/sklearn.utils.class_weight.compute_class_weight.html, Accessed 2023-11-12.