Camila Santos Celes

Machine Learning for Diagnosis of Foliar Diseases in Apple Trees

FINAL PROJECT

DEPARTAMENT OF INFORMATICS

Undergraduate program in Computer Engineering

Rio de Janeiro
June 2022

Pontifícia Universidade Católica do Rio de Janeiro

Camila Santos Celes

Machine Learning for Diagnosis of Foliar Diseases in Apple Trees

Final Project

Final Project, presented to the Computer Engineering program at PUC-Rio as partial requisite to obtain the title of Computer Engineer.

Advisor :	Prof. Marcelo Gattass
Co-advisor:	Felipe Jordão Pinheiro de Andrade

Rio de Janeiro
June 2022

Abstract

Celes, Camila S.; Gattass, Marcelo; Jordão, Felipe. **Machine Learning for Diagnosis of Foliar Diseases in Apple Trees**. Rio de Janeiro, 2022. 28p. Final Project – Departmet of Informatics, Pontifical Catholic University of Rio de Janeiro.

Foliar diseases are a threat to the apple orchard industry, and misdiagnosis or delay in treatment can cause great economic loss. In this project, we aim to build a machine learning model capable or identifying a range of pathogens in apple trees based on pictures of their leaves. We explore and discuss different approaches, from simple CNN models to more complex architectures, using different loss functions and ways to handle the multi-label problem.

Keywords

plant pathology, machine learning, deep learning, image classification, supervised contrastive learning, multi-label, class imbalance

Resumo

Celes, Camila S.; Gattass, Marcelo; Jordão, Felipe. **Aprendizado de Máquina para Diagnóstico de Doenças Foliares em Macieiras**. Rio de Janeiro, 2022. 28p. Projeto final – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Doenças foliares são ameaças à indústria de pomares de maçã, e diagnósticos falsos ou atrasos no tratamento podem gerar grandes perdas econômicas. Nesse projeto, buscamos desenvolver um modelo de aprendizado de máquina capaz de identificar uma série de patógenos em árvores macieiras, baseado em fotos de suas folhas. Exploramos e discutimos diferentes abordagens, desde redes convolucionais simples até arquiteturas mais complexas, usando diferentes funções de perdas e modos de lidar com o problema *multi-label*.

Palavras-chave

patologia de plantas, aprendizado de máquina, aprendizado profundo, classificação de imagens, aprendizado por contraste supervisionado, multi-label, desequilíbrio de classes

Table of contents

1 Introduction

Apple orchards are a multi-billion dollar industry per year, and many of those millions are threatened by foliar diseases. Several different pathogens can cause them, and sometimes the same tree can be infected by multiple pathogens at the same time. Today, diagnosis for foliar diseases is made via manual scouting by specialists yielding a costly and time-consuming task. The lack of qualified people for this job can lead to significant economic loss since a misdiagnosis can compromise the tree. The delay in identifying the disease allows the pathogens to spread throughout the orchard.

The recent developments in Machine Learning and Image Analysis have shown incredible results in a range of application fields, including the diagnosis of both human and plant illnesses. However, identifying foliar diseases in apple trees poses a challenge to the usual approach to Deep Learning since the same pathogen can look very different on trees of apples of different kinds. The shape, color, age of the leaves, different angles, and lighting conditions at the moment of capturing the picture, among other factors, also make it harder for a Machine Learning model to classify different diseases correctly.

Therefore, this project aims to develop a Machine Learning model capable of identifying foliar diseases in apple trees based on pictures of their leaves. Members of the Botanical Society of America have captured and labeled thousands of images of several apple trees that suffer from different pathogens. Some of them are shown in Figure 1.1.

This project will use the available data to study different approaches to image classification with all the challenges intrinsic to the problem.

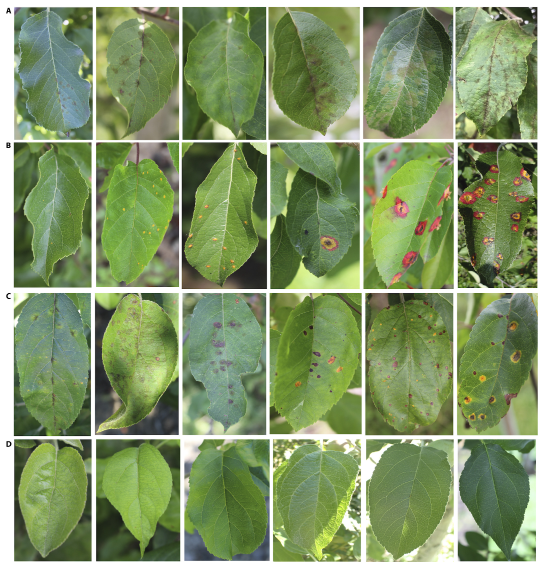

Figure 1.1: Sample images from the data set showing symptoms of cedar apple rust (A), apple scab (B), multiple diseases on a single leaf (C), and healthy leaves (D) (Thapa 2020).

2
Getting to know the data

In order to design an effective Neural Network for the stated problem, it is essential to know the data. In this chapter, we will analyze some aspects of the dataset – the images and their given labels –so that we can choose how to best handle the data for designing and training the Machine Learning model.

2.1
The images

There are 18.632 images in the dataset, all RGB (3 color channels) of varying sizes. Most images have 2672 pixels in width and 4000 in height, but some are as small as 1728×2592 or as big as 3456×5184 pixels. To train our Neural Network (NN) and classify new images, we must standardize these dimensions to a common one.

2.2
The labels

The images have labels that can contain more than one active class, i.e., the classes are not mutually exclusive. Thus the problem at hand is multi-class (there are multiple diseases) and multi-label (multiple diseases can occur in a single sample). The unique class values include four diseases – scab, rust, powdery mildew, frog eye leaf spot – and healthy, which means the lack thereof. If an unhealthy leaf has too many diseases to classify visually, it is classified as complex and can additionally be labeled with a subset of the identified diseases.

To understand the relationship between these classes, we calculate the Pearson correlation coefficient (r) for each pair (Freedman 2007). The computation of this value is done through the equation 2-1, where x_i and y_i are the i-th value of samples in the classes x and y, respectively, and $\bar{x}$ and $\bar{y}$ their mean values.

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2(y_i - \bar{y})^2}} \tag{2-1}$$

Figure 2.1 shows the calculated r coefficients for the dataset classes. We can see they are not correlated – i.e., having one disease is not closely related to having another. The more considerable coefficients shown in the matrix are between the classes "healthy" and other diseases. Those values are negative, as expected since being healthy implies the lack of pathogens.

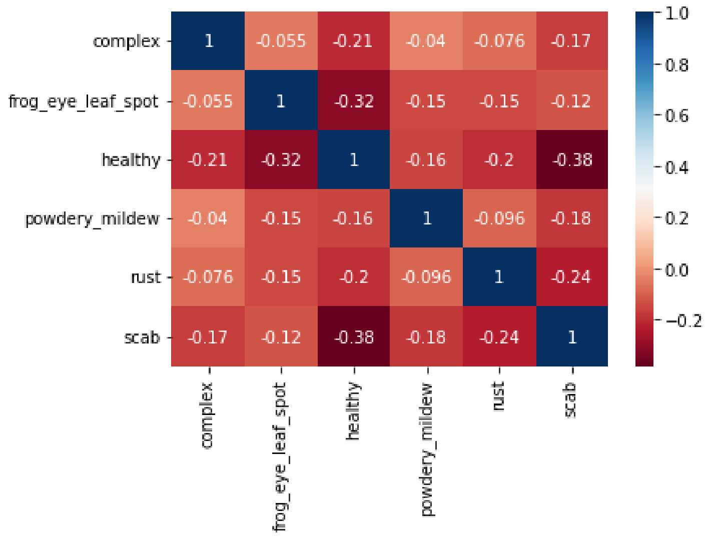

Figure 2.1: Correlation matrix between dataset classes

The distribution of labels in the described classes is illustrated in Figure 2.2. The histogram shows that the dataset is not balanced – some pathogens have very few positive samples. This imbalance can lead to a drop in the accuracy of the designed Neural Network.

2.3 The problem with imbalanced data

Most machine learning algorithms rely on maximizing the accuracy to fit the model to the training data. Therefore, if the dataset is imbalanced, i.e., the number of samples in each class varies greatly, the model can become biased to more frequently choose classes that occur more often (Provost 2000).

Many algorithm-level approaches have been used to deal with this problem, such as introducing misclassification costs (Fernández 2018) and using different performance metrics (**?**). Another approach is to apply re-sampling strategies to obtain a better-distributed dataset (Branco 2015) by upsampling the minority or downsampling the majority ones.

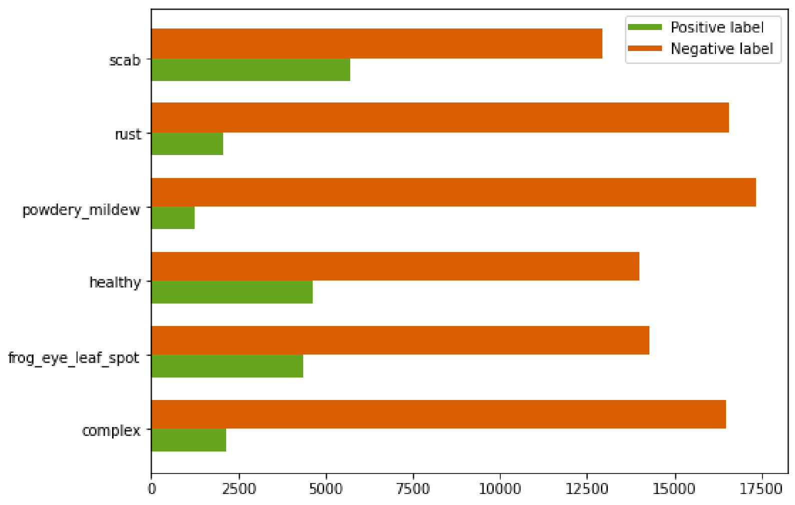

Figure 2.2: Distribution of labels for each class

3
Initial ML approach

In order to develop a Neural Network model able to identify the pathogens in our dataset images accurately, we first develop a standard CNN approach to Image Classification and apply it to our problem.

The ImageNet dataset (Deng 2009) is a collection of millions of annotated images, labeled according to the presence (or lack thereof) of more than 22 thousand classes. Since 2010, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), an annual computer vision competition, has benchmarked state-of-the-art algorithms for the image classification problem. In 2015, ResNet (He 2015) achieved the best result to date in the competition by introducing a deep residual learning framework, implemented via shortcut connections. This introduction addressed the *degradation* problem previously identified in deep convolutional networks (as depth increases, the accuracy gets saturated).

We use ResNet's 50-layer model, pre-trained on ImageNet, to measure its off-the-shelf performance when trained on our dataset.

3.1
Loss and metrics

Take x_i, $i \in \{1, ..., N\}$ as the input images, where N is the batch size. We encode the associated labels into n-hot vectors $y_{ij} \in \{0, 1\}$, $j \in \{1, ..., L\}$, where L is the number of classes. This means that, for an image i and a label j, if $y_{ij} = 1$, the label j is active in i. Now consider the classification model outputs s, where each $s_{ij} \in [0, 1]$ is the predicted probability of label j being active in i.

To compute the loss of our model during training, we use the binary cross-entropy (BCE) loss, which is a common approach to multi-label problems:

$$\mathcal{L}_{BCE} = \sum_{j=1}^{L} y_{ij} \log s_{ij} + (1 - y_{ij}) \log (1 - s_{ij}) \tag{3-1}$$

To evaluate the performance of the trained models, we use the F1-score (equation 3-2) (Fernández 2018), where TP is the number of true positives,

FP of false positives and FN of false negatives. This metric is computed for each class, and we use their mean to evaluate global performance.

$$F1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{3-2}$$

3.2 Training the ResNet

We reserve 30% of the training dataset to use for validation. With that, we are able to measure how the model performs on unseen data samples throughout epochs. As mentioned, we initialize the ResNet-50 with the weights pre-trained on the ImageNet dataset. Then, we fine-tune those weights, training the network until convergence, performing early stopping when the loss on the validation set stops improving. We use Adam (Kingma 2014) for our optimizer, with a learning rate of 10^{-3} and a batch size of 32. The evolution of loss and F1 values during training is shown in Figure 3.1.

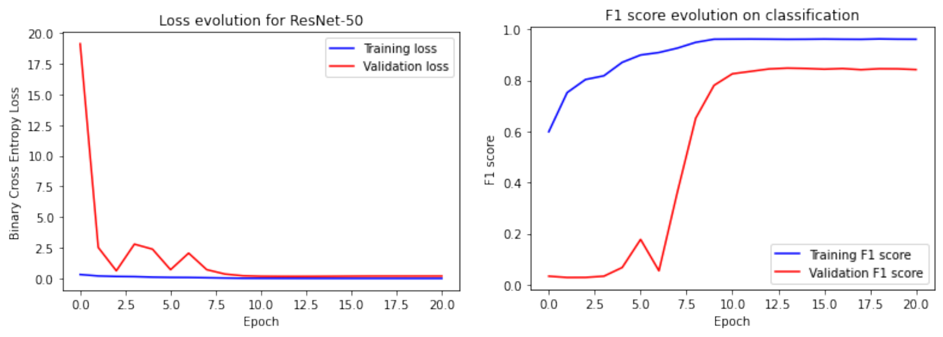

Figure 3.1: Evolution of loss and F1 score for ResNet-50 model during training

4
Supervised Contrastive Loss

Cross-entropy has been the most widely used loss in image classification for years. However, recent works have achieved new state-of-the-art results by introducing Supervised Contrastive Loss (Khosla 2020). It is based on contrastive self-supervised, which is extensively used for non-labeled data by learning similarity and distinctiveness between images in the dataset.

In an augmented batch, $i \in I = \{1, ..., 2N\}$, N size of the batch, we define an anchor z_i as the latent representation of an image x_i, and $z_{j(i)}$ its augmented version (which we call the positive). We further define $A(i) \equiv I \backslash \{i\}$, and so z_k for $k \in A(i) \backslash \{(i)\}$ are the negatives. Self-supervised contrastive learning uses the loss in 4-1 (Chen 2020), where $\tau \in \mathcal{R}^+$ is a scalar temperature parameter. Note how this formulation pulls the positive close to the anchor, and pushes the negatives away from it.

$$\mathcal{L}^{\text{self}} = \sum_{i \in I} \mathcal{L}_i^{\text{self}} = -\sum_{i \in I} \log \frac{\exp\left(z_i \cdot z_{j(i)}/\tau\right)}{\sum_{a \in A(i)} \exp\left(z_i \cdot z_a/\tau\right)} \tag{4-1}$$

In order to extend this idea to a supervised setting, in which for each image x_i we know its label y_i, we now define $P(i) \equiv \{p \in A(i) \mid y_p = y_i\}$, i.e., the positives are now all embeddings which have the same label as the anchor. Thus, the formulation for the Supervised Contrastive Loss in 4-2 pulls together samples of the same class and pushes apart those in different ones.

$$\mathcal{L}^{\text{sup}} = \sum_{i \in I} \frac{-1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp\left(z_i \cdot z_p/\tau\right)}{\sum_{z_a \in A(i)} \exp\left(z_i \cdot z_a/\tau\right)} \tag{4-2}$$

However, note how this formulation only works for single-label problems since the positives are defined by comparing embedding labels directly ($y_p = y_i$). Therefore, to make Supervised Contrastive Learning fitting for the multi-label setting, we must either transform the problem or adapt the algorithm. In the following sections, we present a method for each approach.

4.1 Label Powerset

Many approaches have been used to handle multi-label problems, either adapting the algorithm or the problem (transforming it into a single-label setting). We can, for example, train a binary single-class predictor for each one of the possible classes (Read 2011). Alternatively, the image can be segmented and re-labeled so that each segment is associated with one class (Campbell 1996). However, these strategies are costly and time-inefficient.

A simpler approach is to consider each combination of classes as a single label, called the label-powerset method (Boutell 2004). This means that given a leaf with two pathogens, e.g., "scab" and "rust", instead of attributing both labels to the input, we would consider creating a new label, "scab+rust", and only attribute this single one to the input. So, by using this method, we transform our multi-label problem into a single-label one and then use Supervised Contrastive Learning off-the-self.

For each image in our dataset x_i, and its associated "powerlabel" $\tilde{y}_i$, we first compute a representation vector $r_i = \text{Enc}(x_i)$, then feed it through a projection network to compute the embeddings $z_i = \text{Proj}(r_i)$. We also feed the representation vectors through a dense layer to produce the classification predictions, $s_i = \text{Dense}(x_i)$, $s_{ij} \in \{(0,1)\}$, $j \in \{1, ..., L\}$. The architecture of the network is illustrated in 4.1.

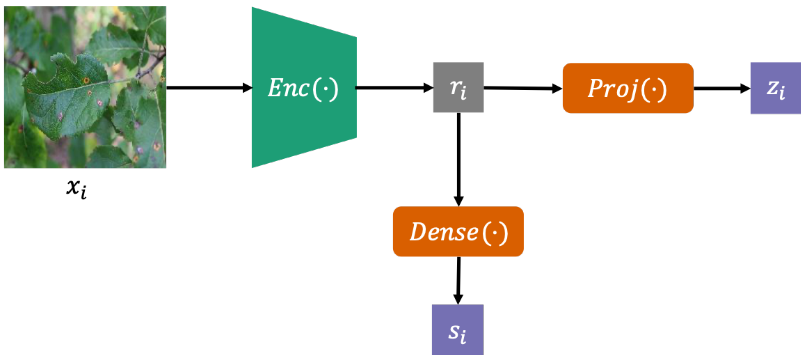

Figure 4.1: Architecture for classifier used in label powerset setting

4.2 Training the network

The training of this network is done in two steps. First, we fit the encoder and projection head to our dataset using Supervised Contrastive Loss in the projection space, which is calculated as in equation 4-2, now with $P(i)$ being defined in the label powerset setting, $P(i) \equiv \{p \in A(i) \mid \tilde{y}_p = \tilde{y}_i\}$.

For the second step, we go back to the multi-label scenario. We freeze the encoder and train the dense layers using binary cross-entropy (equation 3-1), as we did previously.

Note the first training step. The goal is to learn similarities and distinctions between the inputs so that, in the latent space, the embeddings r_i are positioned in a way such that vectors belonging to the same class are close together and those in different classes, apart from each other. Once these embeddings are learned, the second step of training is responsible for producing classification predictions s_i based on the obtained r_i.

We use the ResNet-50 (again pre-trained on ImageNet) as the encoder and a dense layer as the projection head. The described network is trained until convergence using a batch size of 32 and Adam optimizer with a learning rate of 2×10^{-4}, similar to what we did in Chapter 3. Also, the temperature value used for SupCon loss is $\tau = 0.2$. The measured values for the Supervised Contrastive loss during the first training step are shown in Figure 4.2. The values for BCE loss and F1 score for the second training step are illustrated in Figure 4.3.

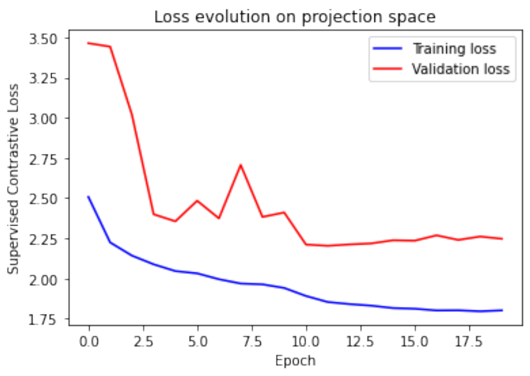

Figure 4.2: Evolution of Supervised Contrastive loss on step 1 of training

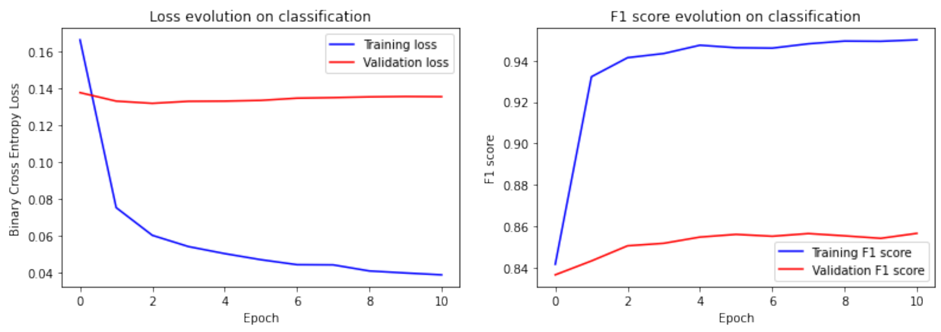

Figure 4.3: Evolution of loss and F1 score on step 2 of training

Notice how step 1 of training (using the SupCon loss) is more significant to convergence, creating bigger loss decrease across more epochs. This, in fact, is the main step in training, since it fits the encoder to the dataset, creating the representation vectors r_i. In step 2 we simply train a few dense layers to, from r_i, produce the classification predictions s_i.

However, note that, by using the label powerset method, the number of possible class combinations can reach 2^L. This makes number of samples assigned to each "powerlabel" small, specially since some combinations are rare in the dataset, heightening the identified data imbalance issue. Other than that, take two images that have a subset of pathogens in common, however are assigned different "powerlabels" due to additional ones also being present. In the given setting, Supervised Contrastive Learning will force distinctiveness between these two samples, when truly we wish to learn the similarity between the labels in common, and distinctiveness between the remainders.

5
MulCon

In order to use Contrastive Learning in a multi-label setting without appealing to the label powerset method, the framework MulCon (Dao 2021) proposes an interesting adaptation of the usual approach to Supervised Contrastive Learning. Instead of transforming the loss formulation to take into account multiple labels, it proposes a network that learns "multiple representations of an image under the context of different labels". That is, for each image, we learn L latent representations, each then associated to a single label. With that, we are able to perform Supervised Contrastive Learning normally (as originally proposed in (Khosla 2020)) in this latent space.

The architecture for this framework is presented in Figure 5.1, which will be explained in detail further in this chapter.

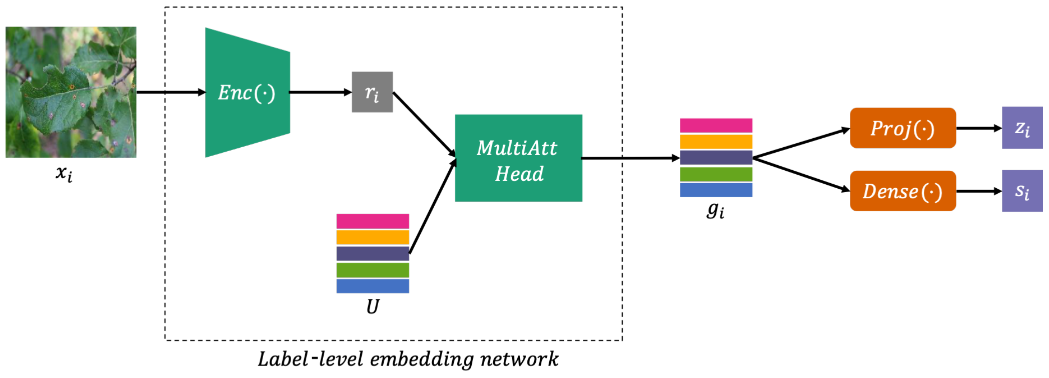

Figure 5.1: Architecture for MulCon

The network is constituted by a label-level embedding network, which learns the mentioned latent representations, a projection head and a classifier (dense layers).

5.1 Label-level Embedding Network

The main component proposed by MulCon is the label-level embedding network, which learns the single-label representations of an image g_i, which we refer to as label-level embeddings. It consists of an encoder followed by a Multi Attention Head.

5.1.1 Encoder: Learning the visual representations

The first layer in MulCon is an encoder, which, as in the usual approach to Supervised Contrastive Learning, is responsible for learning the visual representation vector r_i of an image x_i. This will capture embeddings of locations in the image. Again, we choose to use the ResNet-50 pretrained on ImageNet as our encoder.

$$r_i = \mathrm{Enc}(x_i) \in \mathcal{R}^{W \times H \times C} \tag{5-1}$$

5.1.2 MultiAtt Head: Learning the label-level embeddings

The output of the encoder is then fed through a Multi Attention Head, which outputs the label-level embeddings g_i.

Attention mechanisms have been used in a variety of applications in Machine Learning, from text translation (Minh-Thang 2015) to image captioning (Xu 2015). It works by highlighting important parts of an input, i.e., giving attention to them, while fading out the rest. Given queries Q, keys K and values V, the attention output is computed as in equation 5-2.

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(QK^T\right) V \tag{5-2}$$

The inner product QK^T computes the similarity between the queries in Q and the keys in K. Then, we multiply V with these values, thus highlighting the data points where Q and K are similar, and fading out the rest.

More recently, a Multi Attention Head mechanism was proposed in order to efficiently "allow the model to jointly attend to information from different representation subspaces at different positions" (Vaswani 2017). It works similar to the regular attention mechanism, except it pays attention to multiple latent spaces instead of a single one. Its computation is expressed in equation 5-3, where the W matrices are projections.

$$\begin{aligned} \mathrm{MultiHead}(Q, K, V) &= \mathrm{Concat}\left(\mathrm{head}_1, \ldots, \mathrm{head}_{\mathrm{h}}\right) W^O \\ \text{where head} &= \mathrm{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right) \end{aligned} \tag{5-3}$$

Multi Head Attention is very useful in our problem. As previously discussed, the representation vectors r_i capture embeddings of locations in the image. By giving attention to only the locations that show a specific pathogen in the leaf, we are able to compute new latent vectors g_i which contains embeddings that are now associated with a single label. The label-level embeddings are obtained as:

$$g_i = \text{MultiHead}(U, r_i, r_i) \tag{5-4}$$

Here, U_j is a class-specific embedding – that is, a "classical" representation of the label j – which we use to compute similarity with the locations in r_i, and use these attention scores to highlight the locations in r_i associated with that specific class. U is a parameter, and is learned during training.

This all happens in latent spaces, which we cannot understandably visualize, however, for illustration purposes, Figure 5.2 shows how attention works in our network. Suppose we have two pathogens, whose classical representations are shown in U_1 and U_2. The attention mechanism compares these class-specific embeddings U to the locations in image x_i, and enhances the ones with high similarity. This produces outputs g_{ij}, which are embeddings of the image x_i under the context of the label j.

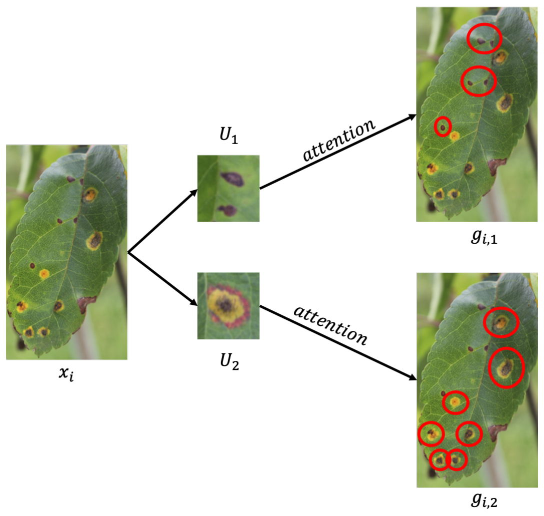

Figure 5.2: Function of attention mechanism in our network

5.2 Projector and Classifier

Now that we have the label-level embeddings, we can perform Supervised Contrastive Learning as we did in Chapter 4, since the latent vectors g_{ij} are now associated with a single label. Therefore, we follow the usual approach and project these onto a space where Supervised Contrastive Loss will be computed, such as we did in the previous chapter. The projection head is made up of linear (dense) layers, thus,

$$z_i = \mathrm{Proj}(g_i) \iff z_{ij} = \mathrm{Proj}(g_{ij}) \tag{5-5}$$

We also feed these embeddings through a classifier (dense layers) which will produce the class predictions, again following what we did previously.

$$s_i = \mathrm{Dense}(g_i) \iff s_{ij} = \mathrm{Dense}(g_{ij}) \tag{5-6}$$

6
Training the MulCon

Training the MulCon framework is, such as in the usual approach for Supervised Contrastive Learning, done in two steps – one in which we train the classifier, another in which we use contrastive loss in the projection space.

6.1
Step 1: training the classifier

Opposite to what we did in Chapter 4, we will now first train the classifier using binary cross entropy loss. This step will produce initial values for the label-level embeddings, g_i, such that, for an input x_i, the component g_{ij} corresponds to label j. For this step, we again use Adam optimizer, with a learning rate of 2×10^{-5}, but, due to increased complexity in the architecture and limited access to GPU memory, we now decrease the batch size to 25. The values for BCE loss and F1 for this step are shown in Figure 6.1.

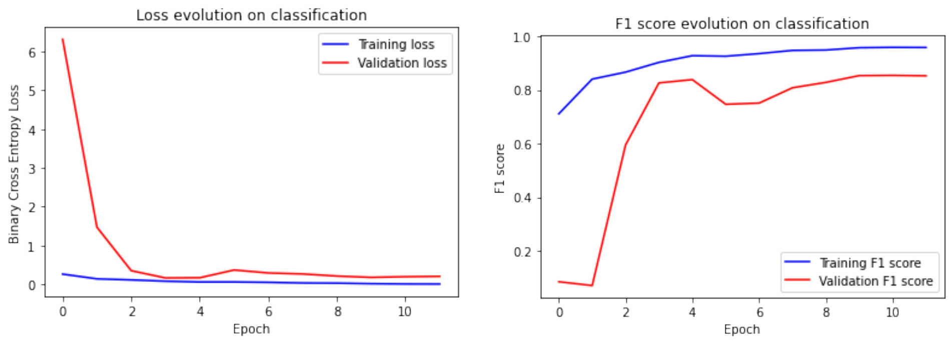

Figure 6.1: Evolution of loss and F1 score for step 1 of training the MulCon

6.2
Step 2: contrastive learning

We use Supervised Contrastive Learning in the second step of training the MulCon. The goal of this step is to fine-tune the weights of the label-level embedding network, in order to produce better label-level embeddings by learning distinctiveness and similarity between them.

We modify the formulation of Supervised Contrastive Loss in equation 4-2 to reflect the changes in variables for the MulCon as suggested in the original paper (Dao 2021). Now, z_{ij} is the anchor, $I = \{z_{ij} \mid y_{ij} = 1\}$ and $A(i,j) = I \backslash z_{ij}$. Additionally, we define $P(i,j) = \{z_{kj} \in A(i,j) \mid y_{kj} = y_{ij} = 1\}$. Thus, the contrastive loss for the anchor $z_i j$ is

$$\mathcal{L}_{ij}^{\text{sup}} = \frac{-1}{|P(i,j)|} \sum_{z_p \in P(i,j)} \log \frac{\exp\left(z_{ij} \cdot z_p/\tau\right)}{\sum_{z_a \in A(i,j)} \exp\left(z_{ij} \cdot z_a/\tau\right)\big)} \tag{6-1}$$

And, so, for the whole minibatch, we define the contrastive loss as the summation over all anchors,

$$\mathcal{L}_{con} = \sum_{z_{ij} \in I} L_{ij}^{sup} \tag{6-2}$$

As discussed previously, the contrastive loss will be computed on the projection space. However, we find experimentally that strongly enforcing distinctiveness in our problem is not most appropriate. And, so, it is beneficial to also optimize over the BCE loss on the classifier output in this second step, and so, the overall loss used in this step is

$$\mathcal{L} = \mathcal{L}_{BCE} + \gamma \mathcal{L}_{con} \tag{6-3}$$

We empirically set $\gamma = 0.1$, which was the value that yielded best results. Now, we use SGD as our optimizer, with momentum 0.9 and learning rate 10^{-3}. We also set $\tau = 0.2$ as we did previously. As for batch size, we are again limited to 25 due to limited GPU memory, yet notice that Supervised Contrastive Learning benefits greatly from bigger batch sizes, since it learns similarity among samples within the same batch only, so we could expect better performance if we were to increase this value. With that, we train the network until convergence, performing early stopping, and the evolution of losses values across epochs is shown in Figure 6.5.

6.3
Treating class imbalance

In order to treat the class imbalance problem identified in Chapter 2, we also train the MulCon imposing misclassification costs (Fernández 2018). This means that we assign weights to each class and label, in such a way that, for example, if a certain pathogen has 500 times more negative samples than positive ones, the cost of predicting a negative label is 500 times bigger, thus

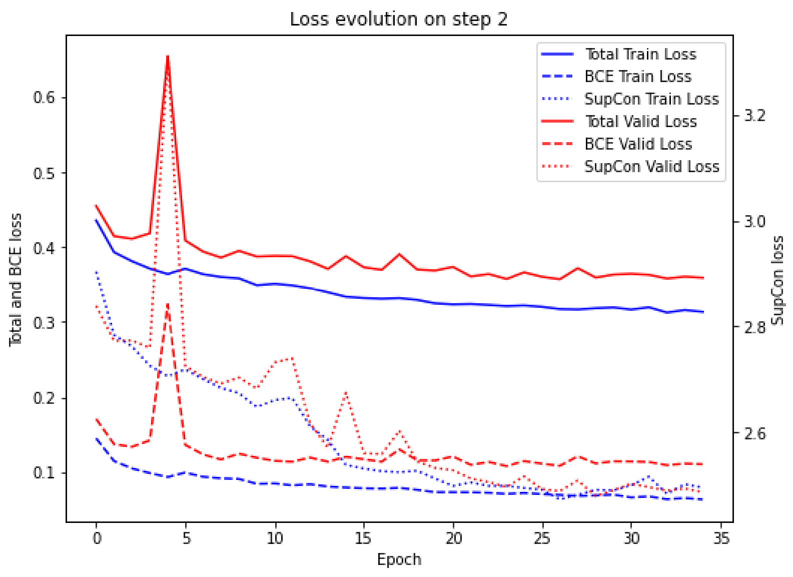

Figure 6.2: Evolution of loss for step 2 of training the MulCon

artificially creating balance in the dataset.

For each class j (pathogen), we compute how many data samples have a positive label for it (n_j) or a negative label for it ($\bar{n}_j$).

$$\begin{cases} n_j = \sum_{i=1}^{N} y_{i,j} \\ \bar{n}_j = N - n_j \end{cases} \tag{6-4}$$

Then, we define the class weights for each class j as such, where ω_j represents the confidence in the positive label for class j, and $\bar{\omega}_j$, the confidence in the false label,

$$\begin{cases} \omega_j = N/(2 * n_j) \\ \bar{\omega}_j = N/(2 * \bar{n}_j) \end{cases} \tag{6-5}$$

Finally, we weigh the Binary Cross Entropy with these values of confidence (Ibrahim 2020),

$$\mathcal{L}_{WBCE} = \sum_{j=1}^{L} \omega_j \Big[y_{ij} \log s_{ij} \Big] + \bar{\omega}_j \Big[(1 - y_{ij}) \log (1 - s_{ij}) \Big] \tag{6-6}$$

We then retrain the MulCon network swapping the BCE loss for its weighted version, WBCE, but keeping every other hyperparameter the same as used previously. The training of this network is shown in Figures 6.3, 6.4 and 6.5.

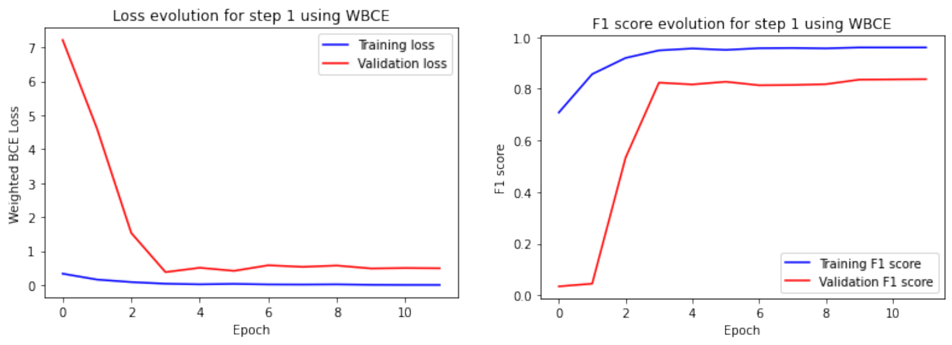

Figure 6.3: Evolution of loss and F1 score for step 1 of training the MulCon with WBCE

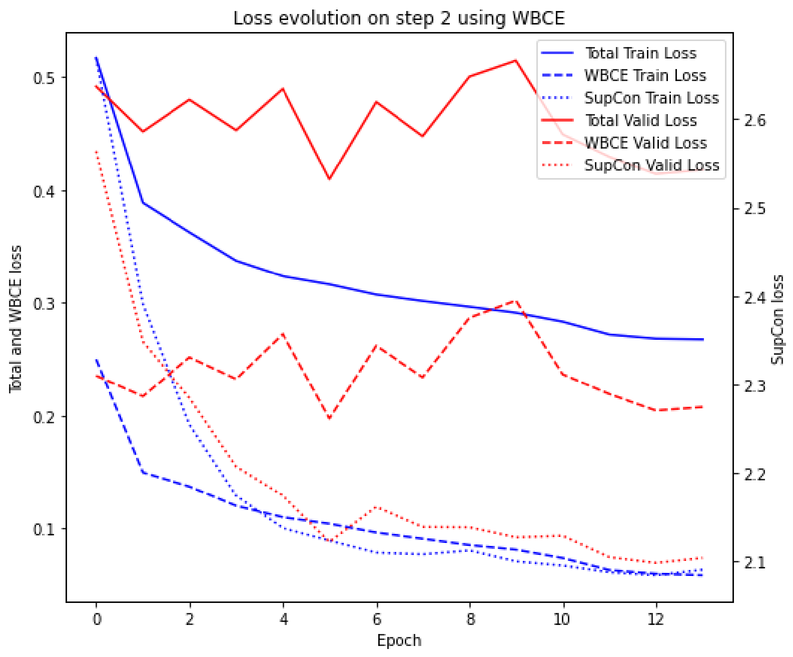

Figure 6.4: Evolution of loss for step 2 of training the MulCon with WBCE

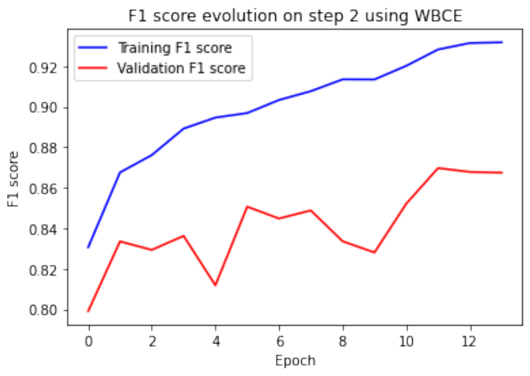

Figure 6.5: Evolution of F1 score for step 2 of training the MulCon with WBCE

7 Results

For the three Machine Learning models presented along this project, we measure their performance on a test subset composed of 3726 images, which corresponds to 20% of the total dataset.

Model	**F1 score**
ResNet-50	0.80
SupCon + Label Powerset	0.84
MulCon	0.85
MulCon + Weighted Loss	0.85

We can already see a big increase of 0.04 when moving to the simple ResNet to the SupCon with powerlabels, and again an increase of 0.01 on top of that when using MulCon.

When using Binary Cross Entropy, each label is classified independently, and, as a result, the model learns whether to predict certain disease or not. It does not, on the other hand, learn the distinctiveness between pathogens, i.e., how different they look. Supervised Contrastive Loss adds this extra layer of learning, which we illustrate in Figure 7.1, in which we can see how label level embeddings are placed before and after enforcing distinctiveness. Each color in the graphs corresponds to a class, and we can see how, after applying SupCon, samples having the same label form better and more compact clusters. Note how the teal colored class is specially affected by this – in the right, the samples located very sparsely and mixed with the orange colored ones, while, in the left, we can see it forms a cluster of its own. This reflects in the big performance improvement when using it with the sub-optimal label powerset method. When we remove this transformation and apply Supervised Contrastive Learning in the truly multi-label scenario with MulCon, we are able to improve the F1 score even more by 0.01.

Finally, we experimented with adding a weighted classification loss to MulCon to treat the class imbalance problem identified in our dataset. However, it did not yield any improvements in performance, and had a F1 score on the dataset same as the one obtained with MulCon using regular BCE loss.

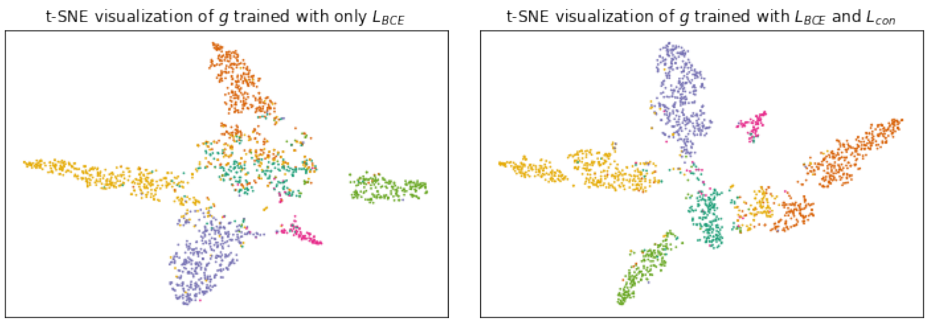

Figure 7.1: t-SNE visualization label-level embeddings after the first step of training, using only $\mathcal{L}_{BCE}$ (right), and after the second step, using $\mathcal{L}_{BCE}$ and $\mathcal{L}_{con}$ (left)

8
Conclusion

Through out this project, we experimented with different methods and neural network architectures for diagnosing diseases in apple trees. We trained and measured performance of a simple, usual approach to image classification, and found that it yielded unsatisfactory results. We then discussed the new state-of-the-art method of Supervised Contrastive Learning and how it is beneficial to learn similarities and distinctiveness among classes, but it is not directly applicable to the multi-label scenario. Therefore, we first tried to adapt our dataset to fit the expected single-label setting through the label powerset method, and used it in Supervised Contrastive Learning. This already created a major performance improvement in comparison to the first experiment. Nonetheless, we discussed how the label powerset method is sub-optimal, thus we tried to adapt our algorithm to be able to handle multi-label problems instead. For that, we implemented MulCon, which proposes learning label-level embeddings of each one of the input images, such that each one is only assigned one label. With that, we were able to improve performance even more, and arrived at a final F1 score of 0.85.

Bibliography

[Boutell 2004] BOUTELL, M. R.; LUO, J.; SHEN, X. ; BROWN, C. M.. **Learning multi-label scene classification**. Pattern Recognition, 37(9):1757–1771, 2004.

[Branco 2015] BRANCO, P.; TORGO, L. ; RIBEIRO, R. P.. **A survey of predictive modelling under imbalanced distributions**. CoRR, abs/1505.01658, 2015.

[Campbell 1996] CAMPBELL, N. W.; MACKEOWN, W. P.; THOMAS, B. T. ; TROSCIANKO, T.. **The automatic classification of outdoor images**. In: INTERNATIONAL CONFERENCE ON ENGINEERING APPLICATIONS OF NEURAL NETWORKS, p. 339–342. Citeseer, 1996.

[Chen 2020] CHEN, T.; KORNBLITH, S.; NOROUZI, M. ; HINTON, G. E.. **A simple framework for contrastive learning of visual representations**. CoRR, abs/2002.05709, 2020.

[Dao 2021] DAO, S. D.; ZHAO, E.; PHUNG, D. ; CAI, J.. **Multi-label image classification with contrastive learning**. CoRR, abs/2107.11626, 2021.

[Deng 2009] DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K. ; FEI-FEI, L.. **Imagenet: A large-scale hierarchical image database**. In: 2009 IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 248–255, 2009.

[Fernández 2018] ALBERTO FERNÁNDEZ, SALVADOR GARCÍA, M. G. R. C. P. B. K. F. H.. **Learning from Imbalanced Data Sets**. Springer, 2018.

[Freedman 2007] FREEDMAN, D.; PISANI, R. ; PURVES, R.. **Statistics**. 2007.

[He 2015] HE, K.; ZHANG, X.; REN, S. ; SUN, J.. **Deep residual learning for image recognition**. CoRR, abs/1512.03385, 2015.

[Ibrahim 2020] IBRAHIM, K. M.; EPURE, E. V.; PEETERS, G. ; RICHARD, G.. **Confidence-based weighted loss for multi-label classification with missing labels**. In: PROCEEDINGS OF THE 2020 INTERNATIONAL CONFERENCE ON MULTIMEDIA RETRIEVAL, p. 291–295, New York, NY, USA, 2020. Association for Computing Machinery.

[Khosla 2020] KHOSLA, P.; TETERWAK, P.; WANG, C.; SARNA, A.; TIAN, Y.; ISOLA, P.; MASCHINOT, A.; LIU, C. ; KRISHNAN, D.. **Supervised contrastive learning**. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, volumen 33, p. 18661–18673. Curran Associates, Inc., 2020.

[Kingma 2014] KINGMA, D. P.; BA, J.. **Adam: A method for stochastic optimization**. 2014. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[Minh-Thang 2015] LUONG, M.; PHAM, H. ; MANNING, C. D.. **Effective approaches to attention-based neural machine translation**. CoRR, abs/1508.04025, 2015.

[Provost 2000] PROVOST, F.. **Machine learning from imbalanced data sets 101**. Proceedings of the AAAI'2000 workshop on imbalanced data, 2000.

[Read 2011] READ, J.; PFAHRINGER, B.; HOLMES, G. ; FRANK, E.. **Classifier chains for multi-label classification**. Machine Learning, 85(3):333, 2011.

[Thapa 2020] THAPA, RANJITA; ZHANG, K. S. N. B. S. K. A.. **The plant pathology challenge 2020 data set to classify foliar disease of apples**. Applications in Plant Sciences, 8(9), September 2020.

[Vaswani 2017] VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, L. ; POLOSUKHIN, I.. **Attention is all you need**. CoRR, abs/1706.03762, 2017.

[Xu 2015] XU, K.; BA, J.; KIROS, R.; CHO, K.; COURVILLE, A. C.; SALAKHUTDINOV, R.; ZEMEL, R. S. ; BENGIO, Y.. **Show, attend and tell: Neural image caption generation with visual attention**. CoRR, abs/1502.03044, 2015.