Using Machine Learning to Detect Grapevine Disease in Wine Production

Ivar Vänglund

Master's thesis
2021:E13

Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

CENTRUM SCIENTIARUM MATHEMATICARUM

Abstract

When growing grapevines there are common risks for disease outbreaks when the pathogen has remained from earlier growing cycles and the weather and soil conditions work in the diseases' favour. The diseases may spread a lot before being highly noticeable. This means that the harvests of grapes for wine production are in jeopardy. This thesis concerns a branch of a, from the Swedish Board of Agriculture funded, project "*IoT and AI to increase the competiveness in Swedish fruit and wine production*" with the special focus for the thesis to use machine learning tools such as convolutional neural networks to detect the most common grapevine diseases downy mildew and powdery mildew. By using an image inventory from the project staff the purpose is to give decision support regarding the mentioned diseases of an input image using neural networks and image analysis. As being able to notice different types and stages of the grapevine diseases is desired, the network was sought to have 5 class outputs but also testing to have fewer (3 and 2). The resulting model, after training to reach a good validation accuracy, reached an accuracy of $\sim 65\%$ when predicting to 5 labels, but with lower recall on the infected classes with fewer image samples. Training for predicting 3 and 2 different outputs gave higher results of $\sim 70\%$ and $\sim 75\%$ respectively and better tendencies of finding non-healthy leaves. It is believed that a larger amount of training images could improve the performance of the network.

Key words

Grapevine disease, Downy mildew, Powdery mildew, Image analysis, Deep learning, Convolutional neural network, Data augmentation

Acknowledgements

I would first and foremost like to thank my supervisor Kalle Åström and co-supervisors Martin Ahrnbom and Mikael Nilsson. Åström has provided great guidance since the thesis's start by discussing the framework with me and the project staff, suggesting initial tasks and coding tools for the machine learning part and later giving good support by having regular meetings with me throughout the winter. Ahrnbom has given useful advice for the image analysis and machine learning problems and answered many general questions from me that helped improving the results. Nilsson gave good guidance in the start of the project by suggesting initial machine learning tools and examples that helped with a few tasks on the way. In general, all supervisors and especially Åström have provided with helpful advice from start to finish.

I want to thank Lotta Nordmark of the Swedish University of Agricultural Sciences and Mikael Gilbertsson of RISE who helped handling the image data in the start of the project by assigning labels to the images and being of good guidance for the biological aspects of the thesis.

I would also like to thank the project staff for *Freja-AI*, Christina Skjöldebrand, Olle Hydbom and Christer Johansson, for giving a great introduction to the project and discussing the framework of the project with me so that it was possible to make a thesis out of it. Skjöldebrand has been of continuous support by keeping in touch throughout the winter, asking about the progress and making sure that advice could be provided. Hydbom and Johansson were also helpful in arranging an excursion to Glemmingebro and Flyinge where I could make some grapevine inspections and discussing tools for the thesis.

I would also like to thank *Teknologkåren vid LTH* for letting me use the office space in the reception of Kårhuset for my programming and writing, as well as being a nearby source of coffee. Furthermore, I want to give special thanks to Maria Sörensson, student counselor of Engineering Physics at LTH, and Jan Kjellström of the *Student Chaplaincy at Lund University* for being of great comfort and guidance when times have felt very difficult for me.

At last I want to thank my friends for making the study time more fun and meaningful and my family for always being supportive of me. Finally, I give the greatest gratitude to my friend Sofia Leonardsson for supporting me and having my back when I needed it the most in the past year and a half.

Contents

1 Introduction

1.1 Background

Wine has an important role among food culture for many people all over the world with lots of resources being put into preserving production quality. Within agriculture and wine production there are many parameters to consider to make a harvest of crops successful. Each planting is a biological process that takes weather conditions and caution of pests and diseases into consideration in getting the desired result. As the society and technology develops it's a natural step to implement more technical resources to make wine production as profitable as possible. That is what the *Freja-AI* project and this master thesis concerns, [1].

The European Union made in 2012 an investment, EIP-Agri, with the purpose to "foster competitive and sustainable farming and forestry that 'achieves more and better from less'", [2]. Between 2014 and 2020 the Swedish Board of Agriculture (Jordbruksverket) launched several innovation projects within EIP-Agri and one of those is *IoT and AI to increase the competiveness in Swedish fruit and wine production*, also called *Freja-AI*. IoT stands for Internet of Things which are systems to integrate and transfer data between physical components e.g. domestic supplies and machines. AI is a more known abbreviation for Artificial Intelligence.

Freja-AI's mission is to implement a decision-support system so that the planter can determine when it's time to take the right measures in the plantation process of grapes and apples, instead of only doing manual inspections of the crops. It should also prevent variations of quality in the harvest with respect to physiological damage, fruit size and colour. A decision tool that's been developed by the project is a mobile application which will provide the user with weather data and analyze photographs of leaves, with more developing steps to be made with the software. Sensor measurements of conditions as air and soil temperature, sun hours, wind and precipitation are made at apple and viticultures outside of Lund and through AI and historical data the time of harvest should be predicted more accurately, [1].

The next step of the Freja-AI project, that this thesis takes its starting point in, is to make the decision-support system able to identify diseases appearing on vine leaves that could disrupt yields and quality if not treated early. It will assist in being more economical in the amount of pesticides being required.

1.2 Goal

The goal for this thesis is to try and implement an artificial neural network that can identify grapevine diseases from images of vine leaves. It will be done using deep learning structures and machine learning tools in MatLab. More precisely, the network will use annotated images of vine leaves as inputs and possibly associated sensor data for the surrounding conditions, to try and identify different stages of downy mildew and powdery mildew infections on leaves.

The performance of the program should be compared from having many output classes in the program (different diseases and severity levels) to fewer (different diseases without stage separation or healthy) to just two (healthy or non-healthy). One boundary for the project is to not handle leaves with multiple diseases present as it might be a too deviated focus. It will also focus

on leaves instead of grapes or stalks as leaf images are more abundant of the available images.

1.3 Resources

Resources for the project will be a computer with MatLab and the Deep Learning Toolbox installed. An inventory of images taken at vineyards by the Freja-AI staff will be used. There is also the possibility of using the back-end of the Freja-AI application to acquire measured data of the weather conditions from the sensors at the vineyard.

1.4 Outline

The outline for this thesis report is initially a theory section with subsections on descriptions of the, for this thesis, relevant grapevine diseases (section 2.1). The complexity of the descriptions are kept to a lower level than a thorough biology report/thesis as this thesis has machine learning as its primal focus. It is followed by a larger section on the machine learning theory (section 2.2) about the components of a neural network and the training and evaluation process.

The methodology is divided in two parts with the gathering and preparation of image data in section 3.1 and the implementation of machine learning in section 3.2.

The results are shown in section 4 divided into subsections for the different number of outputs and how the accuracy will differ. The results are then interpreted in the section for Discussion (section 5) along with overall comments of the thesis and possible future development.

Finally there will be a section of summary and conclusions to wrap things up.

2 Theory

2.1 Diseases concerning wine production

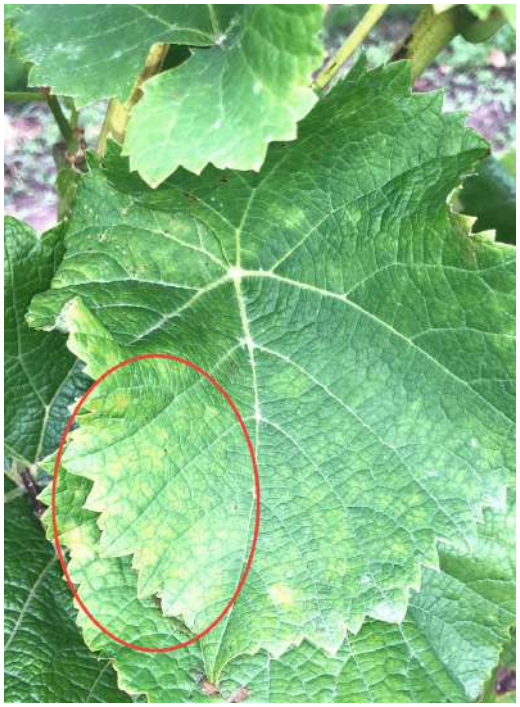
Plantations of grapevines tend to face issues from fungal and/or bacterial grapevine diseases during a growing year that ends in harvest in September or October. Two of the most common are the fungal diseases downy mildew and powdery mildew.

This section concerns shortened descriptions of the symptoms, disease cycles and conditions where the diseases prosper, and not in-depth origins or treatment.

2.1.1 Downy mildew

Downy mildew, caused by the fungi known as *Plasmopara viticola* when attacking grapevines, is for warm and humid climates considered the most troublesome grapevine disease. Oospores (sexual spores) may exist on the vineyard floor or ground in dead leaves over the winter and may survive up to 3-10 years. In late summer, the oospores produce zoospores under favorable conditions called 10-10-24 i.e. when there are at least 10 mm of rainfall and temperature above 10°C in 24 hours. The zoospores may spread through rain and wind to lower parts of the vineyard, as many vineyards are landscaped in slopes, and are the ones infecting the vines, [4].

Yellow oil spots appear on the vine leaves, as seen in figure 1a, which if untreated can cover the entire leaf or the shoots. After warm and humid nights, white downy fungal growth develops that can also cover the grapes as seen in figure 1b.



(a) Early stage of downy mildew infection with yellow spots partly marked in a red ellipse.

(b) Further infection with white down on the grapes. Image from [5].

Figure 1: Symptoms of downy mildew infection on a leaf and on grapes.

If the early stages of infection on the leaves are treated early, much of the yield of the harvest may

be saved. If the disease reaches further the yield may be destroyed but it's still important to locate the infected areas as the fungi may live on and infect and disrupt future yields, [4].

2.1.2 Powdery mildew

Powdery mildew, caused by the fungi known as *Uncinula necator* when attacking grapevines, is the most widespread disease of grapevines. It is a polycyclic disease and the pathogen may overwinter in either the stalks and buds or in spherical fruiting bodies on stems, fruits and leaves lying on the ground. Spores are developed during the spring and are spread to all green plant tissues (leaves, twigs and berries) in the late summer. The spores then produce (germinate) gray-white powdery mycelium, first on leaves and later on berries. The powder sucks nutrients of the leaves and grapes leading to poor quality wine in the long run, or that grapes become destroyed entirely, [7].

(a) Early stage of powdery mildew infection with gray-white powdery.

(b) Later stage with grapes covered in ash grey, floury spores. Image from [8].

Figure 2: Symptoms of powdery mildew infection on a leaf and on grapes.

Powdery mildew is also abundant for warm (15-30°C) and humid (40-100 %) climates but not as dependant on rainfall as downy mildew, [6]. Like downy mildew it's important to identify what stage the disease has reached, so that measures can be made for the upcoming planting cycles.

2.2 Machine learning and image analysis

2.2.1 Structure and layers of neural networks

An artificial neural network (ANN) is a computer structure where data information, such as an image, is given as input to a layer of nodes to produce an output value at the output layer. The output can be an image such as the input one with marked segments, or fewer nodes representing classes that the input could belong to. Between the input and output are one or several hidden layers with functions that process the information before the output layer. An exampled of ANN structure is shown in figure 3.

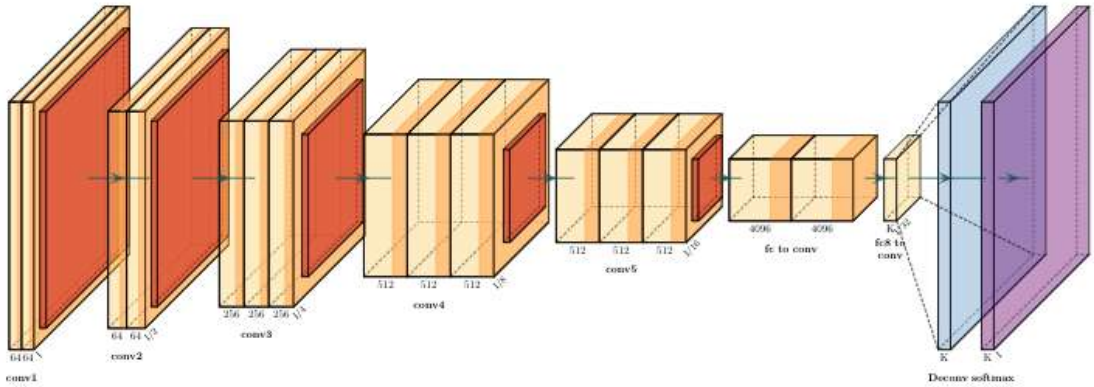


Figure 3: An example of a neural network structure. An image is sent as input and processed through different hidden layers that finds features and alters the size to give an output in the end. Image from [9].

The hidden layers are in general convolutional layers, pooling layers, fully connected layers and activation functions. Towards the end of the network there is also a loss function that evaluates the network performance so that the parameter are updated for the following iterations. This is described in section 2.2.3. An ANN with a large number of hidden layers forms, if visualized, a more tube-like structure of the layers which is interpreted as a Deep Learning network, or just having more than one hidden layer could be considered Deep Learning, [10].

Convolutional layer

An ANN that uses one or many convolutional layer in the network is called a Convolutional neural network (CNN). A convolutional layer has one, or often many, filters that are convolved over the image. The 2D-convolution consists of a filter in shape of a $m \times n$-matrix that moves along the width and height of the image matrix. For each pixel region matching the filter size a new pixel value is calculated through a dot product of the pixel region and the filter flipped along its antidiagonal. As the filter is larger than 1×1 the resulting image will be $(m-1)$ smaller in row width and $(n-1)$ smaller in column height. To retain the image size the output image can use padding to pad out the edges with zeros. One can also define the hyper parameter stride to select how long steps the filter should move along the sides, which also results in a smaller new image. An example of a small dimensional convolution is shown in figure 4.

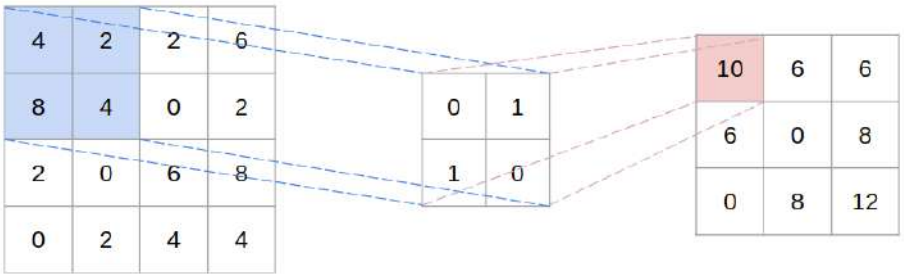


Figure 4: An example of a convolutional layer. A 2×2-filter convolves a 4×4-matrix (image) resulting in a 3×3-matrix with its elements being features convolved from 2×2-regions of the previous matrix. This particular filter is symmetrical along its (non-zero) antidiagonal meaning the addition is simple along this diagonal.

Pooling layer

A pooling layer uses a filter to extract or calculate one new value for each region it overlays, but the filter itself does not have matrix elements as parameters. Instead the type of pooling layer defines the calculation that is made over the filter-sized region, such as extracting the largest value (max pooling) or calculating an average value (average pooling). There is only one filter for each pooling layer, producing one output matrix per input instead of several as in the convolving case. An example of a maxpooling layer is shown in figure 5. A pooling layer also has filter size, stride and padding as hyper parameters that defines the layer. The stride determines how large pixel steps the filter takes between each calculation. For instance, a 2×2-sized filter with a stride of 2×2 only uses each element once, and halves the dimension size to the next layer.

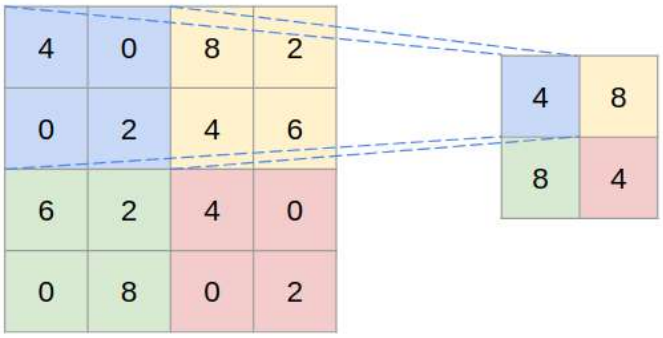


Figure 5: An example of a maxpooling filter. The filter is of size 2×2 with a stride of 2×2 and extracts the largest element from each 2×2-region and taking two pixel steps at the time.

Batch normalization

The network processes what is known as a *batch* (or *mini-batch*) of data for one iteration and then updates the parameters of the layers after each iteration. The time it takes for the network to process the entire training set is called an *epoch* meaning that a batch is one fraction of an epoch.

Deep learning networks produces a vast number of parameters and output elements at each layer for each input image to handle. To make the computing process go faster and avoiding unstable gradients for optimization, batch normalization (batch norm) layers is used to transform each batch of data. The data is transformed to get a zero-mean, $\mu_B = 0$ and unit standard deviation, $\sigma_B = 1$ which speeds up computation and rate of convergence which in turn reduces the number of training epochs that's needed to produce a high performing network, [11].

Activation function - ReLU

A layer of activation functions is a layer of nodes that then defines an output from an input through a function. A non-linear activation function can create complex mappings from input to output making the network more powerful in approximating functions. The function most used for deep neural network is the Rectified Linear Unit (ReLU), defined as $g(x) = \max(x,0)$ and shown in figure 6, which is the only one used for this thesis. It is piecewise linear, and therefore easy to optimize through gradient-based methods (such as stochastic gradient descent, SGD) but still produces nonlinear outputs from linear transformation.

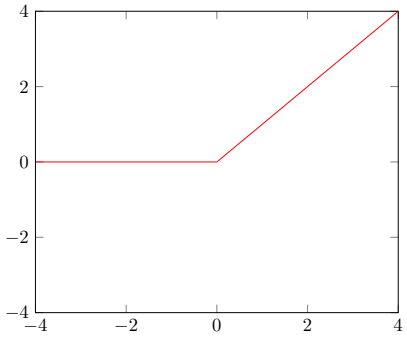


Figure 6: The Rectified Linear Unit (ReLU) activation function which is $g(x) = 0$ for $x \leq 0$ and $g(x) = 0$ otherwise.

Softmax layer

In a neural network the final hidden layer before classification is one that prepares the data to being treated as probabilities where the node with the highest probability assigns the label of the input image. A probability must be in the interval of [0,1] with all individual probabilities adding up to 1. A softmax function σ turns an input vector $\vec{z} = z_{\{i \cdots K\}}$ to a probability distribution using the exponential function and normalizing according to equation

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}. \tag{1}$$

2.2.2 Learning and training process

Supervised learning

For a problem where an input X belongs to a output y and should be classified as such through machine learning, the quest for the mapping $y = f(X)$ is called *Supervised learning*. Each sample used for training the network has a known correct output and for each classification a result of whether the output is correct or not is supervised. In *Unsupervised learning* algorithms may be used to find structures and similar features between the samples such by clustering data, but classification is better set for supervised learning. To have supervised learning, the training data (images) must be labelled so the performance of the network may be sequentially evaluated by comparing the predicted outputs to the ground truth i.e. the correct labels.

Training, validation and testing

After selecting a structure for the neural network all nodes and filters of the layers must have their parameters trained. In other words, the network must learn to classify images correctly. The process of training is done by having a set of data or images called the training set and making the network process the set and predict the output for each data sample. The network then tries to improve its score for each iteration as the parameters of the layers and weights between nodes are updated.

The training set is sought to be large in numbers and diverse with respect to what the outputs are sought to be. It must have representatives for each sought output label or it won't be able to train the network's parameters in finding that label's defining traits and features. The

training set is sent as inputs to the network as smaller (mini)-batches to get parameter updates along the way, and not after the entire set has been processed. An entire pass of the data set is called an *epoch*. The network will after more and more training epochs tend to perform better as the parameters and weights are updated, but the goal for a classifying neural network is rarely to perform flawlessly in classifying the training set, *overfit* to the training data, and then perform less than good on new data. After training the network should be sufficiently generalized to perform well for new and unknown data as well.

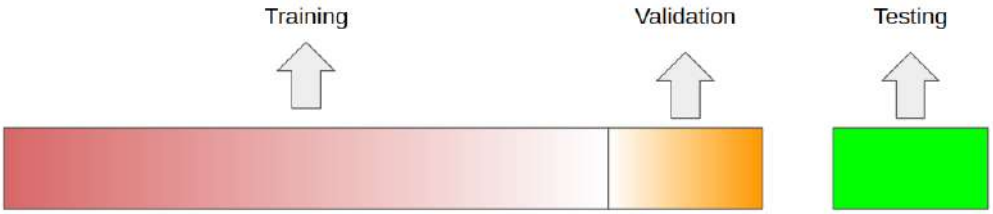


Figure 7: An illustration of having different sets of data for evaluating a machine learning process. The validation data may be interchangeable with the training data while the test set should be independent.

One approach is to evaluate the network with another (often) smaller data set, a validation set that is independent to the training set, throughout the training process. The performance for the validation set gives indications in how to tune the hyperparameters and architecture of the structure. The validation set is passed every epoch, or at another rate if preferred, and its performance is noted. The goal is then to have the validation accuracy increase as it means that the model performs better for new data. Doing too many attempts to reach an acceptable validation accuracy could however also lead to a risk of overfitting to the validation data.

The final subset to evaluating the model is a testing set containing more data independent to the training and validation set. While the training and validation set may interchange data between attempts, the testing set should be set aside from the start of the data acquiring, as seen in figure 7. The validation data is not part of the training phase and affecting parameter updates while the testing set is a more final representation of 'new data' that the finished network model should be evaluated to handle. The testing set is just passed through the network once in the end of the completed training.

2.2.3 Loss functions

A loss function in an ANN is what is used to optimise parameters and make the network learn to correctly classify the data. Different loss functions define different prices to pay for making errors during the training process. This affects how quick the parameters adapt to more high performing iterations. An ANN will strive to minimize the loss given by the function, which decreases as more correct predictions are made by the network.

Categorical cross-entropy loss

Categorical cross-entropy loss is a loss function used for multi-label classification, where there are more than two output labels. For one data or image sample, output probabilities $\hat{y}$ have been calculated through the softmax function. The loss function of comparing the probabilities to its true label y can be expressed as

$$L(y,\hat{y}) = -\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik}\log(\hat{y}_{ik}). \tag{2}$$

The loss function iterates over all N data values and K possible labels. If the sample i truly belongs to label k then $y_{ik} = 1$, otherwise it's $y_{ik} = 0$. The predicted probability of belonging to label k is $\hat{y}_{ik} \in (0,1]$ meaning the logarithm will return a non-positive value which the first minus sign changes to a positive value of the loss. The contribution to the loss is larger if it's less likely to be correctly predicted i.e. when the probability value is low.

Weighted cross-entropy loss

A further application of the categorical cross-entropy loss is to adjust for when the data set has a skew distribution of the ground truth labels. To force the network to put emphasis in predicting the more rare labels, equation (2) can be supplemented with an array of weights, w_k corresponding to their inverted prominence in the data set. The new loss function is shown as

$$L(y,\hat{y}) = -\sum_{i=1}^{N}\sum_{k=1}^{K} w_k y_{ik}\log(\hat{y}_{ik}). \tag{3}$$

If a data set has the label distribution of $\mathbf{c} = \begin{bmatrix} c_1 & c_2 & c_3 \end{bmatrix} = \begin{bmatrix} 0.8 & 0.15 & 0.05 \end{bmatrix}$ then their weights for the weighted cross-entropy loss are the inverted concentration i.e. $\mathbf{w} = \mathbf{c}^{-1} = \begin{bmatrix} \frac{5}{4} & \frac{20}{3} & 20 \end{bmatrix} \propto \begin{bmatrix} 1 & \frac{16}{3} & 16 \end{bmatrix}$. Applying this to equation (3) means that misclassifying the third label is 16 times more costly than misclassifying the first label, giving the network more incentive to classify the rarer labels correct. The factor w_k in the formula may be altered to be e.g. squared to increase the penalty further, or to use $\log(w_k)$ to dampen the penalty. For this thesis, the linear approach to w_k of equation (3) is used.

2.2.4 Performance analysis

The evaluation for a classification problem is efficiently done by plotting a confusion matrix where each case of label prediction is noted along with its true label. One example of a confusion matrix for classifying fruits is shown in figure 8.

A few measures for evaluating the model are *Accuracy*, *Precision* and *Recall*, defined as

$$\begin{cases} \text{Accuracy} &= \dfrac{TP+TN}{TP+TN+FP+FN}, & (4) \\ \text{Precision} &= \dfrac{TP}{TP+FP}, & (5) \\ \text{Recall} &= \dfrac{TP}{TP+FN}. & (6) \end{cases}$$

TP refer to true positives, correct classifications to the first label defined as positive, TN refer to true negatives, correct classification to the other (negative) labels, FP refer to false positive, incorrect classifications to the positive label and FN refer to false negatives, incorrect classifications of the other labels. For multilabel classification, which label is positive or negative is interchangeable for which label should be in focus. In figure 8 for the label 'Orange' the precision is 33.3 % since there are 2 true positives and 3+1 false positives (other fruits predicted as oranges).

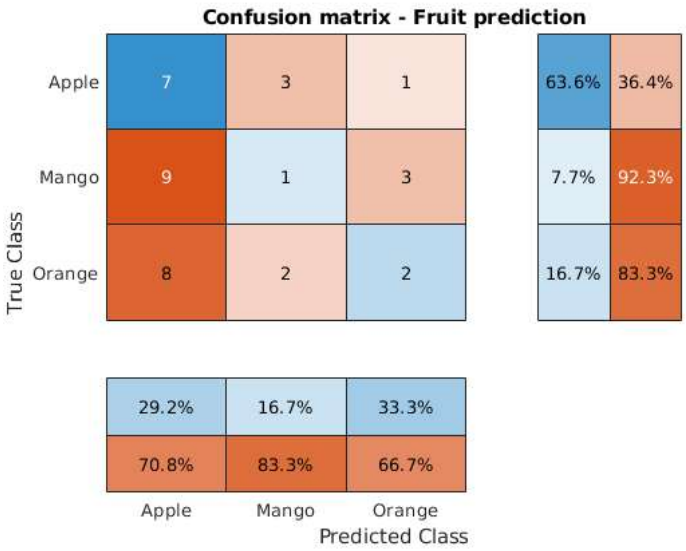


Figure 8: An example of a confusion chart for a prediction of 36 fruits to three labels. The row declares what the true label of each sample while the column declares what it was predicted as. The statistics in the left represents recall for each label while the statistics in the bottom declares precision. The accuracy of the model is calculated as the sum of the elements in the blue diagonal divided by the sum of all matrix elements.

The precision defines how well a model finds correct positives out of predicted positives while recall defines how well it finds positives and may be interpreted as a relative accuracy for each label. Other definitions are that recall for positives are called *Sensitivity* and for negatives are called *Specificity*. A further measure that may be used for evaluation of classes with a skewed distribution is *Balanced accuracy* defined as

$$\text{Balanced accuracy} = \frac{1}{n}\sum_{i}^{n} R_i. \tag{7}$$

where R_i is the recall for each of the n classes. An optimal prediction model has an accuracy, precision and recall close to 1.

2.3 Previous work in similar fields

One similar project to scouting for grapevine diseases using technology is *VineSens*, a decision support system and application developed by Pérez-Expósito et. al. from Universidade da Coruña, [13]. It differs from this project in that it doesn't use machine learning or neural networks but gave inspiration in how data from installed sensors could be used for downy mildew detection.

A previous work in image analysis and convolutional neural networks are the *U-Net* by Ronneberger et. al., [14]. The project concerned effective biomedical segmentations using few training images and repeated convolutions, ReLU and maxpooling without fully connected layers.

Another project of using image analysis and machine learning for improving agriculture was a project of detecting movements for cows around automatic milking stations by Guzhva et. al., [15].

3 Methods

3.1 Preparation of image data

The first task of the project was to go through the inventory of images already taken by Lotta Nordmark of Swedish University of Agricultural Sciences and Mikael Gilbertsson of RISE, who are two participants of the Freja AI-project. In total there were 1295 images taken between 2019-08-15 and 2020-09-29 with wine leaves, apple leaves, grapes, stalks or shrubbery as motives. The only documented information about each image were the date each photograph was taken along with geographical coordinates. The purpose of going through the inventory was to find the ones with wine leaves (healthy and non-healthy) as a clear focus.

3.1.1 Image motives and ground truth labelling

By going through the images sorted by date it was possible to discard 463 images that only contained leaves from apple trees, which wasn't of interest for this branch of the Freja AI-project. The images taken between October 2019 and May 2020 (in total around 170 images) were also for the most part discarded, as the image motive were branches and stalks without leaves. All remaining images were also inspected to remove the images with grapes (healthy or non-healthy) or bushes and shrubbery as motives.

In total 520 original images remained for possible use. An initial labelling of the images was made by the author to four different labels; Healthy and powdery mildew (stage 1, 2 and 3). The lay labelling was made by looking for the known symptoms of powdery mildew and evaluating the date as leaf infections mostly occur in late summer.

Map plots of all images using their coordinates was made to check which locations most images originated from and also if the initial lay labelling seemed accurate. The latter was done by looking for if there were clusters of images with infections. The map plots are shown in figure 9 and 10. The colour of each spot corresponds to the label (stage of disease outbreak) made by the author defined by the legend (in Swedish) in the lower right of the images: 0 (no outbreak), 1 (small outbreak), 2 (medium outbreak) and 3 (large outbreak).

Figure 9: Map plot showing the geographical position of each image around the vineyard in Glemmingebro. A few spots are far off from the planting site meaning that the GPS data may be inaccurate to an extent.

Figure 10: Map plot showing the geographical position of each image around the vineyard in Flyinge.

The map plots showed that most images were taken in Glemmingebro, where no weather and soil sensors had been installed on the vineyard, in contrast to Flyinge. Therefore the possibility of using weather and soil data from the Freja AI application backend as additional model parameters was too limited. The focus was instead kept solely on image analysis for this thesis project. There were also a few (18) images that were taken in Linköping and Mikael Gilbertsson recommended that those were discarded as they could be too deviant from the other images that were taken in Scania.

In consultation with the more biology proficient Nordmark and Gilbertsson a more thorough labeling of the remaining images was made. It was initally sought to have seven different labels: Healthy, downy mildew (stage 1, 2 and 3) and powdery mildew (stage 1, 2 and 3). There were however only three and nine images labeled as stage 3 downy mildew and stage 3 powdery mildew respectively. A decision was made to merge the labels of stage 2 and 3 for each disease into two new stage 2 labels. 22 images showing leaves with signs of malnutritions were also discarded as it would be too difficult to take into account those along with images of downy and powdery mildew. The number of images in each new label is shown in table 1.

Table 1: Number of images assigned to each label. In total there are five labels combining the leftmost column and uppermost row, except for the healthy ones.

	Stage 0	Stage 1	Stage 2
Healthy	336		
Downy mildew		37	26
Powdery mildew		23	26

(a) Healthy leaf

(b) Downy mildew, stage 1

(c) Downy mildew, stage 2

(d) Powdery mildew, stage 1

(e) Powdery mildew, stage 2

Figure 11: One sample image from each label is shown above. As seen by comparing figure 11a and 11d it can be very difficult to distinguish infected leaves from healthy. In this case the leaf with stage 1 downy mildew have a few yellow spots on the right side of the larger leaf. The light from the sun may also make it difficult in finding the infected regions.

Furthermore it was decided to also run network training and classification with fewer labels by merging label categories to see if the performance would improve. Different tests were made for three different labels, by merging stages 1 and 2 for each disease, and two different labels, by merging all labels with disease stages. The label spread for three and two different labels is shown in table 2 and 3 respectively.

Table 2: Number of images assigned to each label if merging stage 1 and 2 of the infected labels, leading to three usable labels.

Healthy	336
Downy mildew	63
Powdery mildew	49

Table 3: Number of images assigned to each label if merging all of the infected labels, leading to two usable labels.

Healthy	336
Non-healthy	112

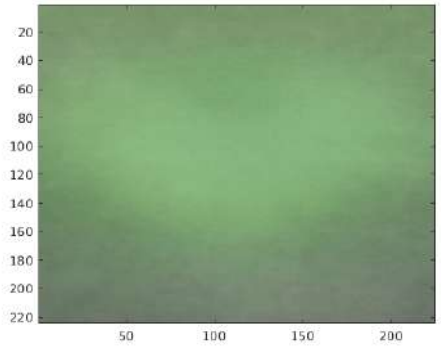
3.1.2 Image size and augmentation

The original resolution for the majority of the available images were 2360×4032 pixels with 3 colour channels (It's always assumed to have 3 colour channels from now on). As the network would be enormous and take very long time to execute with the original image size all images were scaled down to a more manageable size. By starting off small with an image size of 28×28 a few images were ocularly inspected by the author. As it was close to impossible to distinguish any infectious regions from images with known infections it was concluded that it would be too difficult for the deep learning network as well, with the given resolution. With an image size of 56×56 the infected regions of some stage 2 images were visible but most were not. By raising the resolution by another factor of 2 for each side to 112×112 most features of infected images were visible so it would be more feasible for the network.

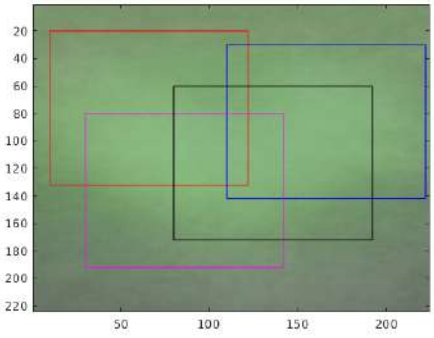
20 % (89 images) out of the 448 original images were removed and saved as a testing set, and the remaining 359 were used for training and validation. It was made sure that all individual classes were present in the testing set. To get more data for the training and validation of the network a few augmentation techniques were performed. By getting an expanded data set the network would be more robust by learning more generalised characteristics of the images instead of memorizing details.

The first goal was to generate images that were zoomed in at different regions of the original images. The original images were reshaped to 224×224, then regions at a fourth of the size i.e. 112×112-regions were cropped from the larger ones. Firstly, all 448 original images of size 224×224 were added together and then divided by 448 to get an average image out of the ones focused on a wine leaf. The calculated average image is shown in figure 12a.

The average image had a vague resemblance to a leaf meaning that it can be assumed that most images have the leaf in central focus of the image. This average image was used to choose four coordinates where 112×112-regions could be cropped that has different corners of the leaf zoomed in focus. In figure 12b four 112×112-boxes were marked and the same coordinates were used to generate four extra images for each original. Assumptions were made that a corner of a leaf will be featured in each new image.



(a) Average image of all 448 leaf images of size 224×224 summed together

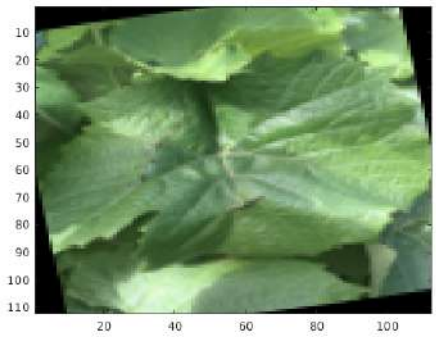


(b) Smaller 112×112 image croppings at different corners of the lighter region.

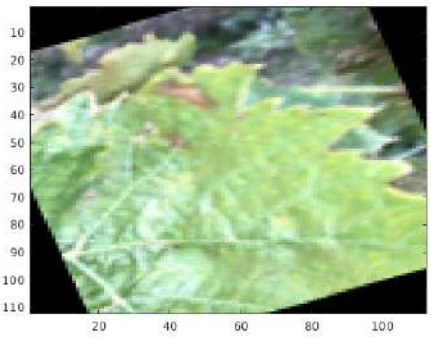
Figure 12: Average image of all 448 leaf images, with and without 112×112 image croppings. The lighter green vaguely resembles a leaf in the left image, so croppings according to the right image might be effective in getting a zoomed in region of a leaf for each original image.

Using this procudure an additional 1436 images of size 112×112 were created out of the 359 original images for training and validation. To simplify the labelling, each new image were given the same label as the image it was cropped from, even though an infected pixel region only may be clearly visible in another part of the leaf, that's not part of the cropping.

The second goal was to augment the image inventory by rotating the 1795 images by small angles to make the network more applicable for input images of different camera angles. Each image was rotated 10 and 20 degrees to the left and right with the command `imrotate` in MatLab along with the option to crop the rotated image to maintain it's size of 112×112. Two rotated images are shown in figure 13a and 13b.



(a) An original image rotated 10 degrees to the left.



(b) A zoomed image rotated 20 degrees to the left.

Figure 13: Two sample images rotated 10 and 20 degrees respectively. The `imrotate` method extrapolates the image with black pixels and crops away the corner areas to maintain the wanted size.

The rotating procedure gave an additional 7180 images to the inventory for training and validation. The spread of the labels is shown in table 4 for five output labels, and in table 5 and 6 for three and two labels respectively.

Table 4: Number of images assigned to each label with the augmented image inventory.

	Stage 0	Stage 1	Stage 2
Healthy	6725		
Downy mildew		750	525
Powdery mildew		450	525

Table 5: Label spread with three output labels.

Healthy	6725
Downy mildew	1275
Powdery mildew	975

Table 6: Label spread with two output labels.

Healthy	6725
Non-healthy	2250

3.2 Design of deep learning network

The first attempt at a deep learning network structure was taken with inspiration of an assignment in the Machine Learning course (FMAN45) at LTH in Spring 2019. The network concerned classifying digits of size 28×28 from the MNIST data base which was used as a starting point. The layer array as printed from MatLab is shown below in table 7, including a few adjustments made at the start:

- The image input (layer 1) was changed to have 3 colour channels
- The number of convolutions in layer 13 was changed to match the number of output classes.

Table 7: Overview of the initial deep learning network which was later expanded.

# Layer	Type of layer	Specification
1	Image Input	$28 \times 28 \times 3$ images with 'zerocenter' normalization
2	Convolution	5 3×3 convolutions with stride [1 1] and padding [0 0 0 0]
3	Batch Normalization	Batch normalization
4	ReLU	Rectified linear unit
5	Max Pooling	2×2 max pooling with stride [2 2] and padding [0 0 0 0]
6	Convolution	10 4×4 convolutions with stride [1 1] and padding [0 0 0 0]
7	Batch Normalization	Batch normalization
8	ReLU	Rectified linear unit
9	Max Pooling	2×2 max pooling with stride [2 2] and padding [0 0 0 0]
10	Convolution	20 3×3 convolutions with stride [1 1] and padding [0 0 0 0]
11	Batch Normalization	Batch normalization
12	ReLU	Rectified linear unit
13	Convolution	5 3×3 convolutions with stride [1 1] and padding [0 0 0 0]
14	Softmax	Softmax
15	Classification Output	Cross entropy

For each quadrupling of the input resolution i.e. increasing the 2D dimensions by a factor of 2 for each side, additional layers had to be added to lower the dimensions for the final layers. After the first convolution layer another batch normalization, ReLU and maxpooling layer was added when raising the input size to $56 \times 56 \times 3$, with the max pooling layer (with stride 2×2) efficiently halving the dimensions further in the network.

The usable image inventory was very skewed with nearly 75 % of the images being labelled as healthy and the spare 25 % was divided into four infected labels. Early network testing faced the problem that the network "took a shortcut" in getting a high performance rate by converging to classifying all images as healthy. By doing so the network nearly always got an accuracy of around 75 %. This was fixed by replacing the final classification layer from regular Cross entropy to Weighted cross entropy and using an array of inverted occurrence of all labels. The weights of occurrence, w is

$$w = \frac{1}{N} \begin{bmatrix} N_H \\ N_{PM1} \\ N_{PM2} \\ N_{DM1} \\ N_{DM2} \end{bmatrix} \approx \begin{bmatrix} 0.75 \\ 0.08 \\ 0.06 \\ 0.05 \\ 0.06 \end{bmatrix} \tag{8}$$

and the inverse weights w^{-1} was used as parameters for the classification layer. A similar calculation was made for the cases with three or two output classes.

The final structure of the deep learning network was chosen as below with an input size of 112×112 and layer 19 being adjustable to how many output classes that were desired.

Table 8: Overview of the final deep learning network.

# Layer	Type of layer	Specification
1	Image Input	$112 \times 112 \times 3$ images with 'zerocenter' normalization
2	Convolution	5 3×3 convolutions with stride [1 1] and padding [0 0 0 0]
3	Batch Normalization	Batch normalization
4	ReLU	ReLU
5	Max Pooling	2×2 max pooling with stride [2 2] and padding [0 0 0 0]
6	Batch Normalization	Batch normalization
7	ReLU	ReLU
8	Max Pooling	2×2 max pooling with stride [2 2] and padding [0 0 0 0]
9	Batch Normalization	Batch normalization
10	ReLU	ReLU
11	Max Pooling	2×2 max pooling with stride [2 2] and padding [0 0 0 0]
12	Convolution	10 4×4 convolutions with stride [1 1] and padding [0 0 0 0]
13	Batch Normalization	Batch normalization
14	ReLU	ReLU
15	Max Pooling	2×2 max pooling with stride [2 2] and padding [0 0 0 0]
16	Convolution	20 3×3 convolutions with stride [1 1] and padding [0 0 0 0]
17	Batch Normalization	Batch normalization
18	ReLU	ReLU
19	Convolution	5 3×3 convolutions with stride [1 1] and padding [0 0 0 0]
20	Softmax	Softmax
21	Classification Output	Weighted cross entropy

For each attempt, the 8975 images were shuffled and 25 %, 2244 images, were held out as the validation set, and the remaining 6731 images became the training set. The mini batch size for each training period were set to 336 giving 20 iterations per epoch. The learning rate was kept constant as 0.1.

To evaluate how well the network performed with respect to being able to distinguish all classes, and not just the healthy ones, Balanced accuracy was used as the mean of the recall for all classes.

For example in a case where all healthy images are classified correctly but none of the others would result in an accuracy of 75 % for the total set. It would be a recall of 100 % for the healthy images and 0 % for each of the infected classes. This would result in a balanced accuracy of $\frac{1}{5}(100 + 0 + 0 + 0 + 0)\% = 20\%$ which is low. While trying to improve the overall performance by tuning the network and image augmentations it become useful to take balanced accuracy into ongoing consideration.

4 Results

In the sections below are prediction scores for when declaring 5, 3 or 2 different output classes of the data set. In section 4.4 are tables that compare the different models' abilities to classify infected leaves.

4.1 5 output classes

Training and validation

In figure 14 graphs are shown of how the performance (accuracy) and loss progresses when the model is trained for 30 epochs i.e., 600 iterations with a validation once per epoch. Towards the end of the training period the increase and decrease for the validation performance and loss function respectively stagnates as they won't keep up with the training plots. The graphs appeared to have converged and if training further there would be risk of overfitting the model to the training data.

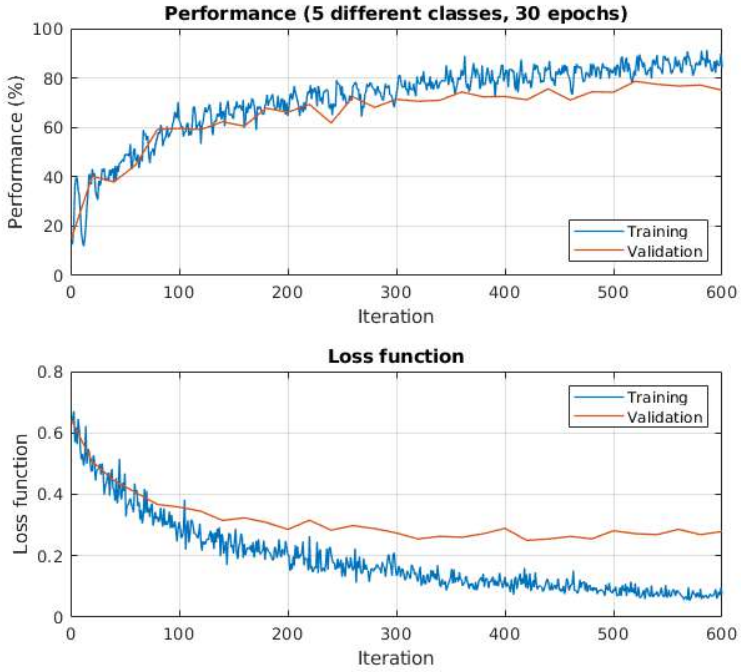


Figure 14: The progression for performance and loss when training the 5-class network in 30 epochs and validating once per epoch. The upper graph shows the accuracy for each iteration when the classification layer compares to the ground truth. The lower graph shows the total loss for each iteration (the classification part) which tends to decrease as the performance increases.

Table 9 summarizes the scores of the model when classifying the entire training and validation sets after completed training with the training set naturally having a higher accuracy as the parameters of the network layers are better adjusted to its images. The higher balanced accuracy for the training means that the network got higher accuracy on the infected classes with fewer images, as can be seen in the right margin of the confusion matrix in figure 15.

Table 9: Summary of the classification scores for training and validation of the 5-class problem after training in 30 epochs.

	# Correct images	Accuracy	Balanced acc.
Training set	5740 / 6731	0.8528	0.9535
Validation set	1688 / 2244	0.7522	0.7448

Figure 15 and 16 show the confusion matrices for the classification of the training and validation set respectively. The matrices summarizes which individual class each image has been classified to rather than just declaring right or wrong. For the training matrix, the recall was very high for each infected class and slightly lower for the more abundant healthy class. Conversely the precision was superb for healthy and lower for the infected ones. Especially 'Downy_1' had a low precision and it occurred often that healthy images were classified as 'Downy_1'.

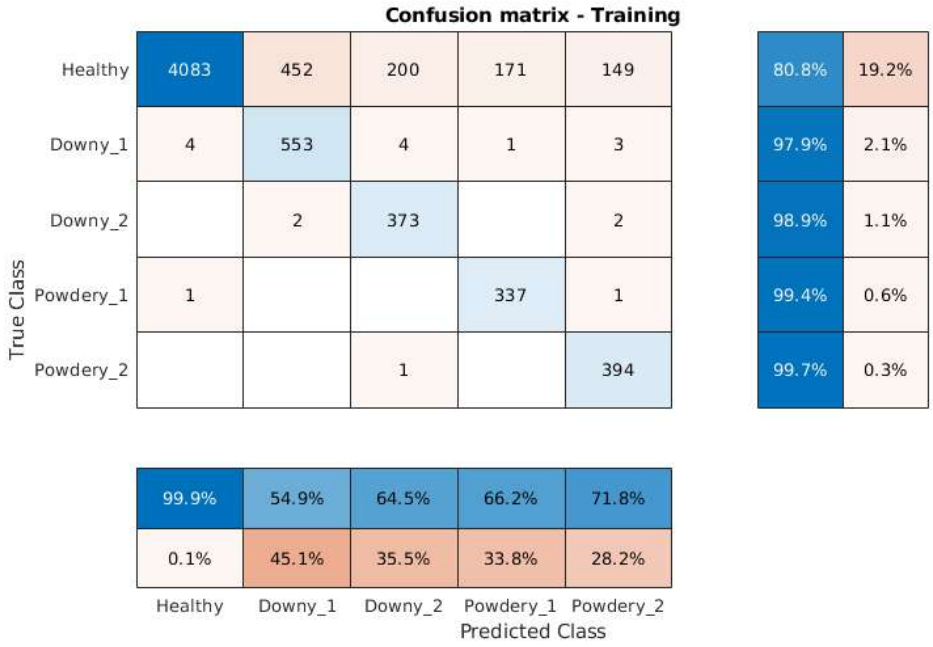


Figure 15: Confusion matrix of the predictions for the training set after completed 5-class training.

The validation matrix showed that the recall was weaker for most infected classes but still high for 'Powdery_2'. The precision stats were similar in pattern to the training ones and still weakest for 'Downy_1' and also relatively worse for 'Powdery_1' compared to training. More healthy leaves were classified as 'Downy_1' compared to the true label itself.

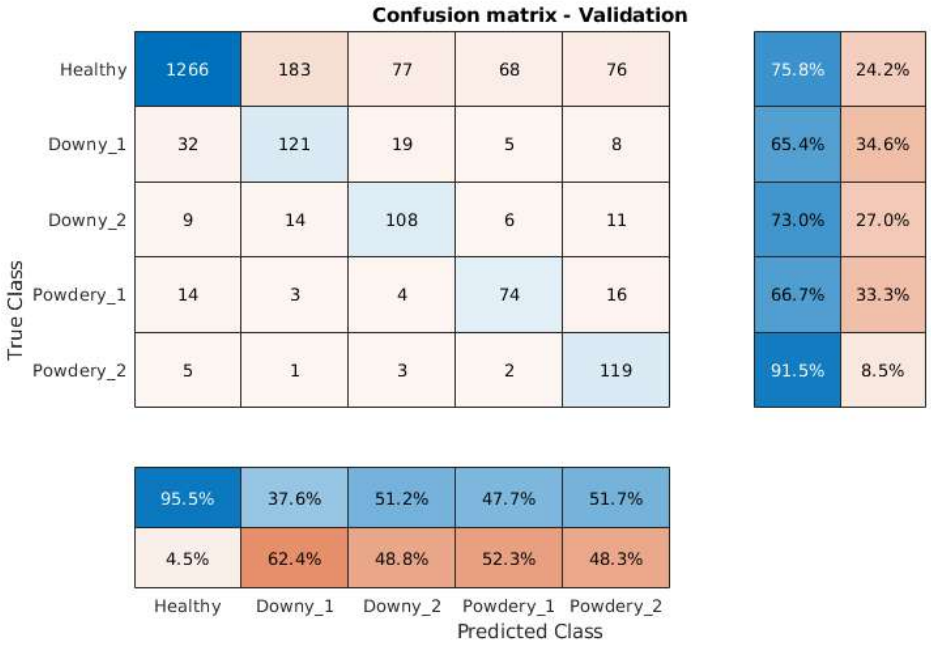


Figure 16: Confusion matrix of the predictions for the validation set after completed 5-class training.

Testing

The testing set was originally set aside as 20 % of the 448 original images ≈ 89 images. As it would be a very small data set the same images were augmented by first adding 336 images of zoomed quarters and also 1780 images of when the 445 images had been rotated ±10° and ±20°.

The classification scores of the testing sets are listed in table 10 with the smallest set reaching the highest accuracy of 66 %. All sets however showed a fallback in the balanced accuracy by only reaching around 40 % meaning that the healthy leaves carried the largest weight of the high accuracy at the expense of the infected leaves.

Table 10: Summary of the classification score for the different sized testing sets of the 5-class problem after training in 30 epochs.

	# Correct images	Accuracy	Balanced acc.
Original testing set	59 / 89	0.6629	0.3754
Incl. zoomed images	288 / 445	0.6472	0.4082
Zoomed and rotated images	1330 / 2225	0.5978	0.3985

In figure 17 the confusion matrix for the medium sized testing set is shown. Recall and precision were still good for healthy leaves and a solid recall for 'Powdery_2', but very low performing for all others leading to the low balanced accuracy. The precision are lowest for the stage 1 diseases with it being more that healthy leaves are classified as infected.

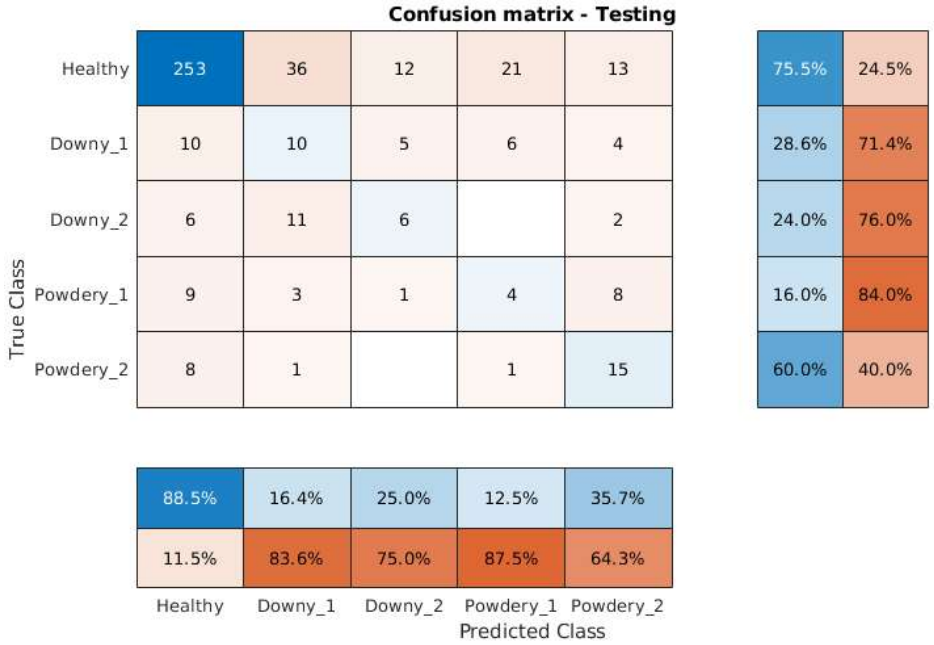


Figure 17: Confusion matrix of the predictions to the 5-class problem for the testing set with the original images and zoomed quarters (445 images in total).

4.2 3 output classes

Training and validation

In figure 18 the progression of the training and validation accuracy and loss is shown when the 3-class network is trained for 20 epochs. The network was trained for a shorter period compared to the 5-class network, as the validation performance appeared to have converged earlier, but still reached a higher performance for both training and validation.

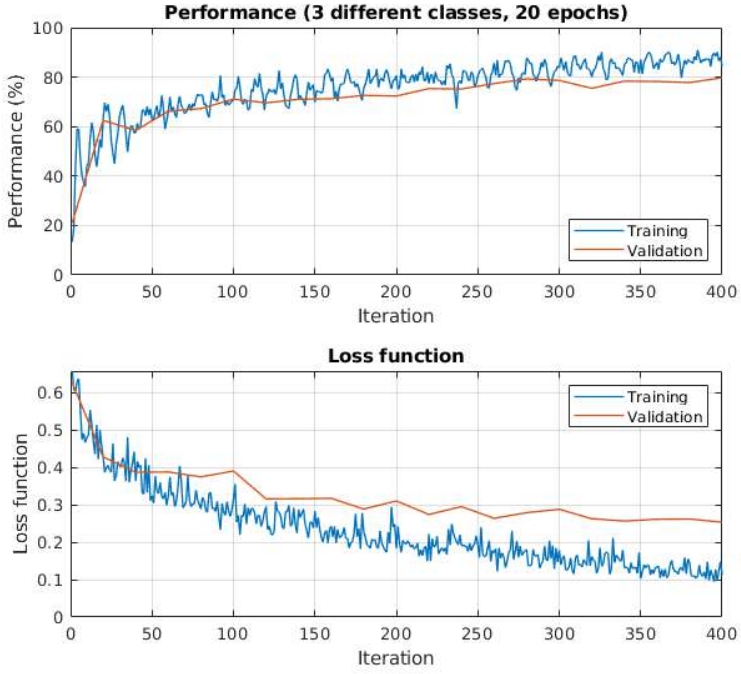


Figure 18: The progression of training the network for 20 epochs (400 iterations), with 3 sought output classes, and validating once per epoch.

Table 11 summarized the final training and validation scores of the 3-class model with the accuracies being higher than for 5 classes (except that the BA for training is slightly lower). The accuracy and balanced accuracy are closer to each other for the training set, compared to the 5-class problem where they differed 10 percentage points.

Table 11: Summary of the classification scores for training and validation of the 3-class problem after training in 20 epochs.

	# Correct images	Accuracy	Balanced acc.
Training set	5873 / 6731	0.8770	0.9260
Validation set	1790 / 2244	0.7977	0.8054

Figure 19 shows the confusion matrix for the classification of the training set. The recall for the healthy leaves was higher here compared to the 5 class problem (with the healthy class being nearly unchanged in size) meaning fewer healthy leaves had been classified as infected. Otherwise the recall was similar to the stats in 15. The precision was higher for the 'Downy' images compared to the separate stages in the 5-class problem but slightly lower for the 'Powdery' images.

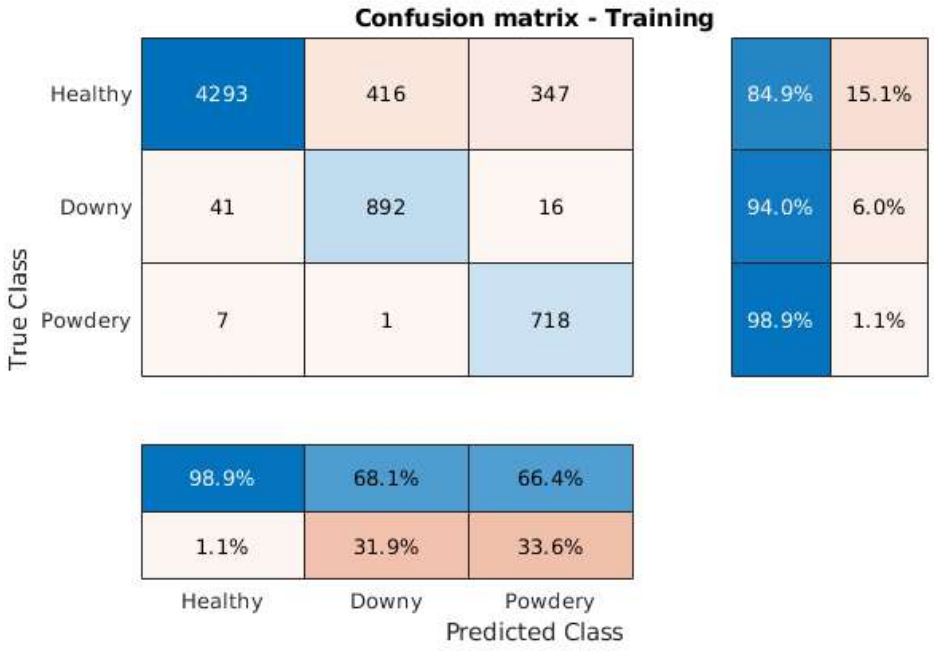


Figure 19: Confusion matrix of the predictions for the training set after completed 3-class training.

Figure 20 shows the confusion matrix for the classification of the validation set. The recall was similar but higher than the stats in 15 when combining the separate stages. The precision was also higher for both diseases, instead of slightly decreased for 'Powdery' as in figure 19 compared to figure 15.

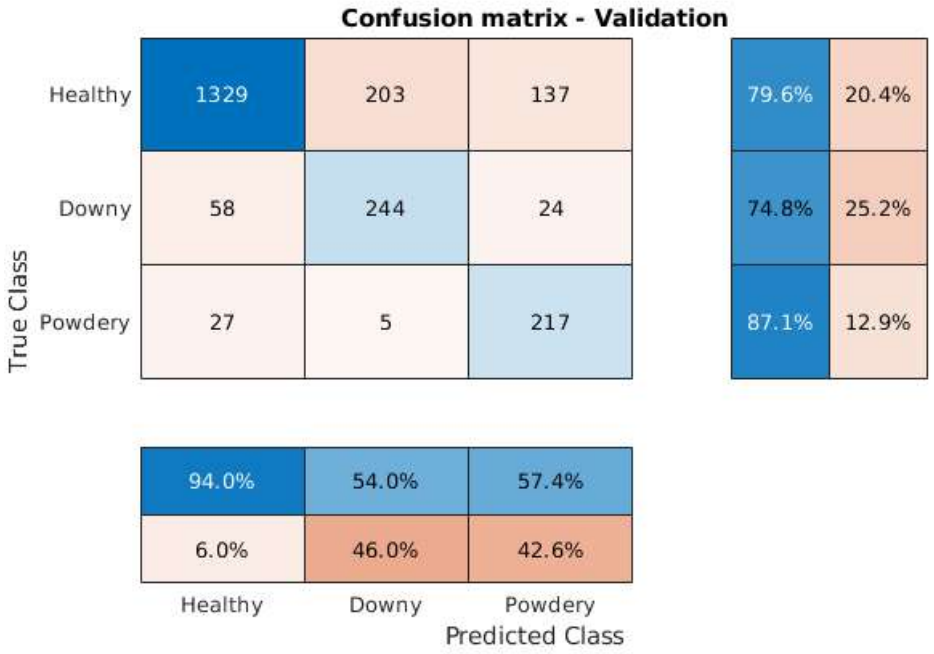


Figure 20: Confusion matrix of the predictions for the validation set after completed 3-class training.

Testing

The classification scores for the different sized test sets are shown in table 12. The accuracy was similar but 4-8 percentage points higher than in table 10. The balanced accuracy was however much better and in this case better for the smallest set compared to the 5-class testing.

Table 12: Summary of the classification score for the testing set (with and without augmentations) of the 3-class problem after training in 20 epochs.

	# Correct images	Accuracy	Balanced acc.
Original testing set	63 / 89	0.7079	0.6716
Incl. zoomed images	310 / 445	0.6966	0.6336
Zoomed and rotated images	1504 / 2225	0.6760	0.6253

The confusion matrix for the medium sized testing set is shown in figure 21. The results were more positive than the 5 class testing, by having a recall above 50 % for each label and the precision were also higher for the infected labels. Only the recall for 'Powdery_2' in figure 17 out of the stats of the infected labels was comparable to these results.

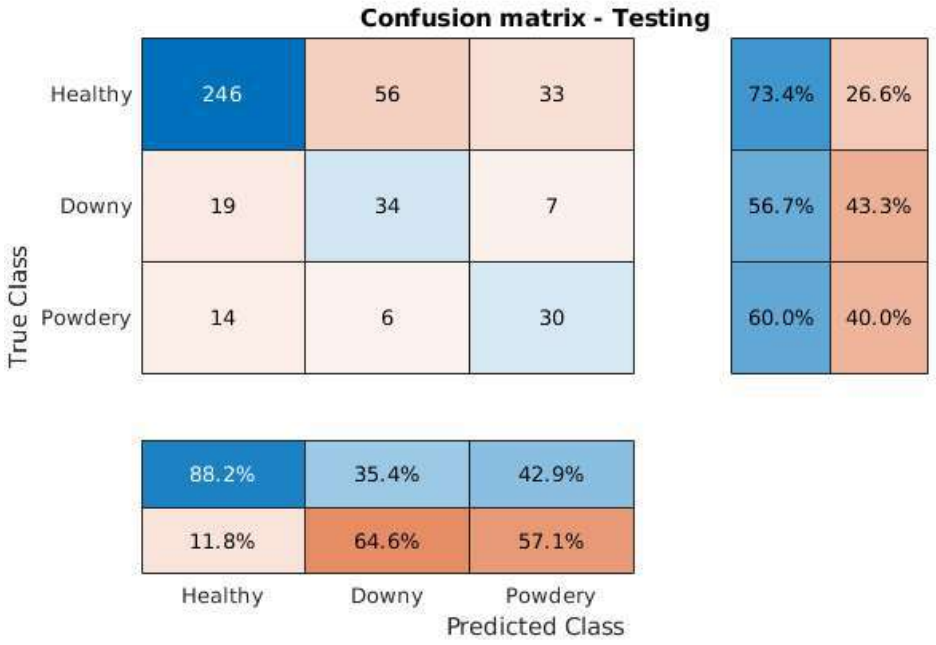


Figure 21: Confusion matrix of the predictions to the 3-class problem for the testing set with the original images and zoomed quarters.

4.3 2 output classes

Training and validation

In figure 22 the progression of the training and validation accuracy and loss is shown when the 2-class network is trained for 15 epochs. The performance reaches high level (above 80 %) even faster and the loss shows peculiar traits in its few spikes in training loss that are larger than the validation loss.

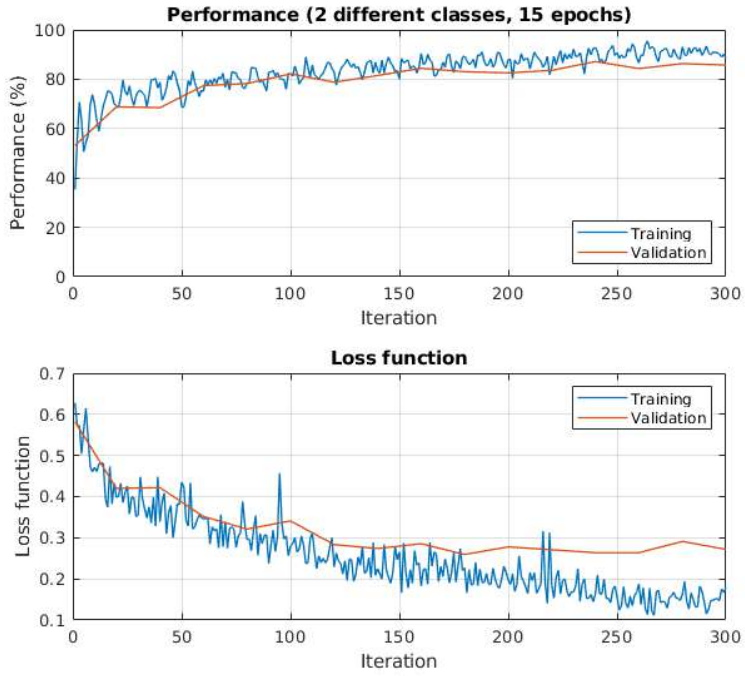


Figure 22: The progression of training the network for 15 epochs (300 iterations), with 2 sought output classes, and validating once per epoch.

Table 13 shows the final classification scores for the training and validation sets after 15 epochs of training to classify binary as healthy or not. All accuracy measures are higher compared to the 3-class and 5-class problem (except for the BA of the 5-class training set).

Table 13: Summary of the classification scores for training and validation of the 2-class problem after training in 15 epochs.

	# Correct images	Accuracy	Balanced acc.
Training set	6108 / 6731	0.9074	0.9301
Validation set	1996 / 2244	0.8574	0.8614

Figure 23 shows the confusion matrix for the classification of the training set. Following the trend when decreasing the number of outputs, the precision are higher for this matrix compared to the matrix in figure 19. The recall for the non-healthy is similar to the average of 'Downy' and 'Powdery' in the aforementioned matrix.

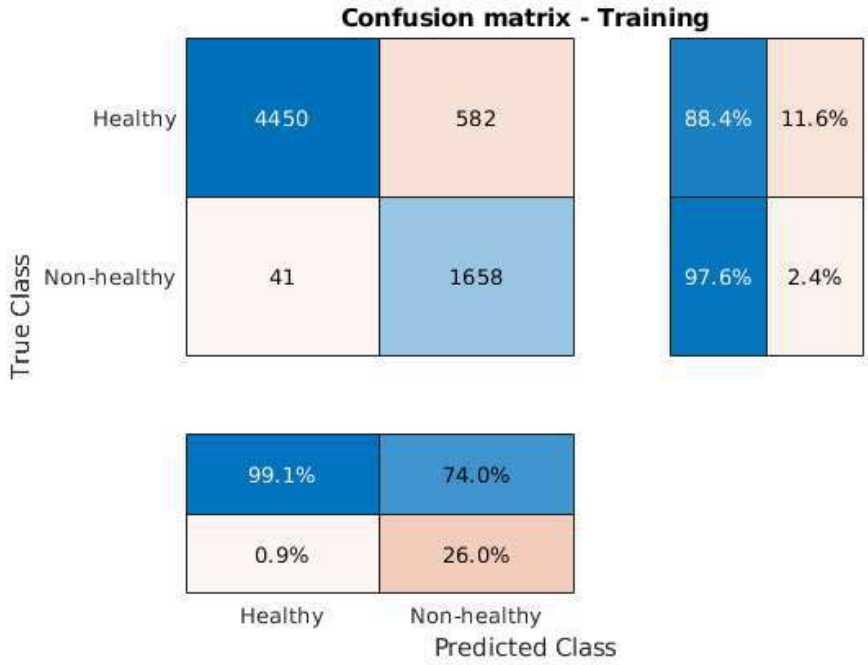


Figure 23: Confusion matrix of the predictions for the training set after completed 2-class training.

Figure 24 shows the confusion matrix for the classification of the validation set. The recall and precision are higher for this table compared to the other validation scores, especially the recall for non-healthy ones being solid above 60 %.

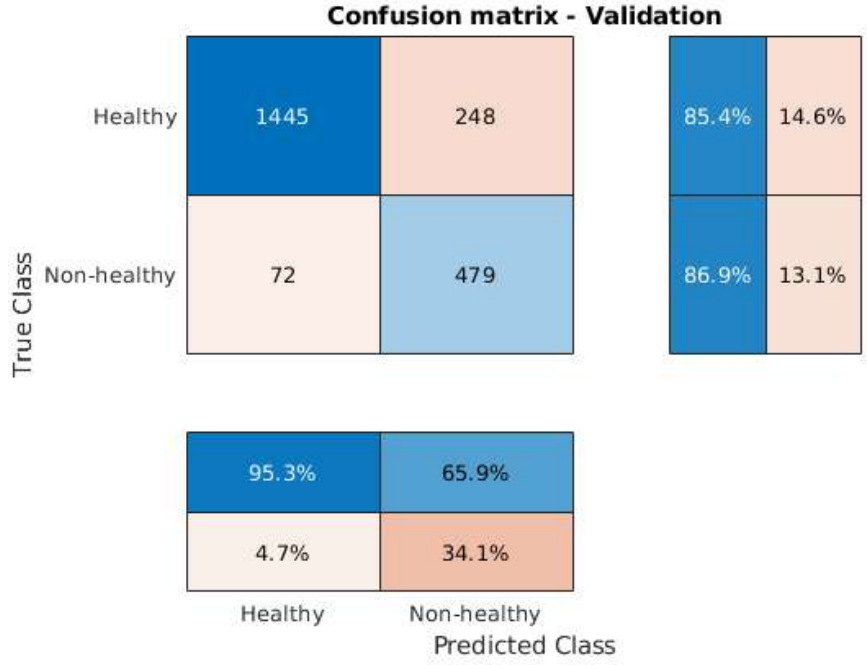


Figure 24: Confusion matrix of the predictions for the validation set after completed 2-class training.

Testing

The classification scores for the different sized test sets are shown in table 14. The accuracy reached around 75 % for each testing try which were higher than for 5 and 3 classes, while the balanced accuracy also reached the best score out of the models.

Table 14: Summary of the classification score for the testing set (with and without augmentations) of the 2-class problem after training in 15 epochs.

	# Correct images	Accuracy	Balanced acc.
Original testing set	69 / 89	0.7753	0.7286
Incl. zoomed images	339 / 445	0.7618	0.7502
Zoomed and rotated images	1666 / 2225	0.7488	0.7422

The confusion matrix for the medium sized testing set is shown in figure 25. Both classes reached a recall of over 70 % and the non-healthy leaves had a precision of over 50 % for the first time of the testing scores for the different models.

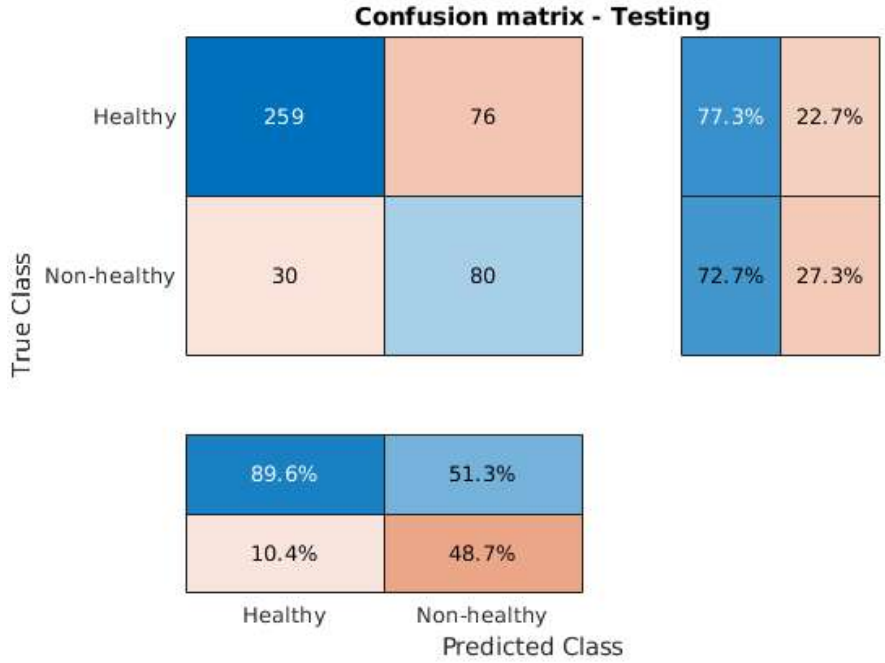


Figure 25: Confusion matrix of the predictions to the 2-class problem for the testing set with the original images and zoomed quarters.

4.4 Comparison between output predictions

Predicting separate grapevine diseases

Table 15 lists the scores and recall of the 5-class model predicting each disease (e.g. adding the scores of Downy_1 and Downy_2 to a collected label Downy) and compares it to the 3-class model. The * means also including the wrongly predicted stage of the same disease, for instance a 'Downy_1'-image classified as 'Downy_2' is included in Downy*, as a way of displaying a lower performance threshold for the model with more classes and making it more comparable with the model that has a wider scope.

Table 15: A summary for adding up the specific disease stage predictions of the 5-class problem (figure 16) to a collected amount for each disease and compared against the 3-class results (figure 20). The testing scores correspond to the matrices in figure 17 and 21.

	Val. predictions	**Val. recall**	**Test. predictions**	**Test. recall**
5-class, Downy	229/333	0.6877	16/60	0.2667
5-class, Downy*	262/333	0.7868	33/60	0.5500
3-class, Downy	244/326	0.7485	34/60	0.5667
5-class, Powdery	193/241	0.8008	19/50	0.3800
5-class, Powdery*	211/241	0.8755	27/50	0.5400
3-class, Dowdery	217/249	0.8715	30/50	0.6000

The table shows that the 3-class model's two infected labels got better recall for the testing set but slightly lower than the labels with * for the validation recall.

Predicting non-healthy leaves

Table 16 lists the scores and recalls of the 5-class and 3-class model predicting non-healthy (NH) images and compares to the 2-class model. The * means also including all images predicted as infected, and not necessarily the specific true label, to make it more comparable with the 2-class model.

Table 16: A summary for adding up the specific diseases and stage predictions of the 5-class (figure 16) and 3-class (figure 20) problems to a collected amount representing non-healthy (NH) and comparing against the 2-class results (figure 24). The testing scores correspond to the matrices in figure 17, 21 and 25.

	Val. predictions	**Val. recall**	**Test. predictions**	**Test. recall**
5-class, NH	422/574	0.7352	35/110	0.3182
5-class, NH*	524/574	0.9129	77/110	0.7000
3-class, NH	461/575	0.8017	64/110	0.5818
3-class, NH*	490/575	0.8522	77/110	0.7000
2-class, NH	479/551	0.8693	80/110	0.7273

The table shows that the 2-class model's two infected labels got better recall for the testing set and the validation set except for the 5-class model that includes all non-healthy predictions.

5 Discussion

5.1 Interpretation of results

In general the results of the 5-class problem were satisfying when looking at the scores of the validation set that gave an accuracy and balanced accuracy of around 75 % as shown in table 9. Another execution, not featured here, tried using 40 epochs of training but the results didn't improve. Instead the validation loss would increase meaning that 30 epochs for training feels sufficient. When trying the testing set the accuracy was still solid being between 60-66 % as shown in table 10 depending on including the augmentations or not.

The accuracy could however be interpreted as misleadingly large as for each run of the different sized testing sets the rate of healthy images among the correctly classified was around 87 %. The healthy images did constitute 75 % out of the data set so the results show that it is easier to predict the Healthy leaves. It is reflected in the column of balanced accuracy in table 10 which is around 40 % as figure 17 shown that the recall is very low for all infected labels except 'Powdery_2', as its symptoms could be considered more distinct with more white spores on the leaves.

Comparing the 5-class testing results to the table results for 3 and 2 classes it is apparent that the network faces issues in classifying many classes and finding the distinctions for the different diseases and infection stages. The confusion matrices for validation and testing showed that the precision was lowest throughout for the stage 1 diseases and there were more healthy images classified as 'Downy_1' than true 'Downy_1' images themselves. There were though about nine times more healthy images than 'Downy_1' and 11 % (one ninth) of the healthy ones classified as 'Downy_1' meaning that it might not be too extraordinary.

The images with powdery mildew performed collectively better recall-wise than the ones with downy mildew and 'Powdery_2' always performing best precision-wise of the infected labels. Through the author's eyes it could be derived that the leaves with downy mildew had less distinctive symptoms on many of the images of the inventory. As seen in figures 11, the powdery mildew images are more distinctive with their grey-white areas, than the more obscure yellow spots. 'Powdery_2' were the most distinctive and least prone to being classified as healthy, while 'Downy_1' were most frequently classified as healthy and healthy ones were had 'Downy_1' as the most common misclassification, as they could be very similar to a healthy leaf.

The tables for comparing different output predictions 15 and 16 showed that the testing recall were higher for when having a model for predicting fewer different labels than when having several labels but merging the results for the different stages and diseases.

Balanced accuracy was used to compensate for that the network is better at predicting healthy images than the non-healthy. What should be taken into consideration when comparing the BA for different number of outputs is the effect of the denominator in equation 7. In a skewed prediction result classifying all healthy ones as correct and none others gives a BA = 0.2 for 5 classes, 0.33 for 3 classes and 0.5 for 2 classes, meaning the columns showing balanced accuracy in the tables should have that in consideration.

Having different sizes of the test sets didn't affect the testing results remarkably. Instead, the sets including the images rotated $\pm 10^{\circ}$ and $\pm 20^{\circ}$ had the lowest accuracy.

5.2 Sources of errors

When evaluating the methods used in the thesis, possible errors exist primarly in the image inventory and effects of the chosen augmentation techniques.

Even when collaborating with biologists of the project and sifting out the images for usable ones with vine leaves in focus, some images have more than one leaf that may to an extent compete for the focus of the network predictions. The light from the sun could vary a lot on some images and either affect the lightness of the entire leaf and background of an image or produce small spots of lights on the leaf surface. These lighting effects could affect the possibility of finding the yellow spots of leaves with downy mildew. Data augmentation techniques of applying new lightness or color to the images could possibly improve the model's ability in aspect to this.

The choice of augmenting the data set by producing zoomed quarters at certain coordinates of each image might have had some fallbacks. A simplified approach was used in cropping the same coordinates for all images without inspecting all hundreds of them. The zooming might lead to a leaf not being a prime focus anymore and especially the infected (white or yellow) regions might not be included while the image is still labelled as infected. One zoomed image is figure 26 where more soil than leaf is in focus, and other images might have similar problems.

Figure 26: A zoomed quarter of a leaf image where a larger contiguous area of soil is in focus than the leaf.

5.3 Future work and suggestions

If all or more images could be associated to the weather data from the sensor stations then probably would another implementation be possible that is not limited to image analysis and looking for visible symptoms.

Another approach of augmentation for the image data could be to crop the entire leaf from each original image and then project to other backgrounds to train the network to distinguish leaves from varying settings. A large work load would be needed to manually crop each leaf from the image, or finding a usable tool to make it automated, that was too specific too put focus on for

this thesis.

There are probably an astronomical amount of different choices to be made regarding the network structure and hyperparameters for each layer that could improve the performance. A more simplified approach was used in layer selection for this thesis that found satisfaction in reaching high training and validation results, rather than trying to experiment further and further.

There are many known convolutional neural networks and detection tools that could be used as a method of transfer learning. The method would be to have pre-trained layers at the start and then manually implement the final layers that define the sought output labels.

The most general reflection obtained is that a larger inventory of images for training would largely help improving the performance and robustness of the network, rather than having just 448 original images, at least for the objective of having 5 output classes.

6 Summary and conclusions

In this thesis we study how machine learning methods improve wine production, by detecting grapevine diseases such as downy mildew and powdery that may spoil coveted harvests.

The results showed that a deep learning network, that has trained with nearly 7000 images, could reach acceptable (∼60 %) to high classification accuracies (∼80 %) depending on the sought specificity of the outputs. Having a narrow scope of 5 different output classes, with the purpose of detecting separate kinds and stages of grapevine diseases, performed worse when tested against entirely new data as the recall for each infected label was low. Having a wider scope by taking disease stage (3 classes) or disease type (2 classes) out of notice performed better so for this implementation and image data set too many specific labels was suboptimal.

Conclusions are that to try and classify objects, that are very homogeneous in visual traits, into a large number of outputs the data set has to be large in numbers so that the network can learn the not so distinctive traits as good as possible. More experimentation can be made regarding the network design and more carefulness could be taken into consideration regarding the image augmentation to make sure that each image shown something relevant.

There are further work to be done for Freja-AI to have a foolproof decision support system for working to preserve the quality of wine production, but this thesis have been a few steps down the road towards it and partially shown that technology such as machine learning has a developable place for agriculture in Sweden.

References

[1] Germundsson L, "Att mäta är att veta", https://www.landsbygdsnatverket.se/pagang/nyheter/nyhetsarkiv/attmataarattveta (2020)

[2] EIP-AGRI, "About EIP-AGRI", https://ec.europa.eu/eip/agriculture/en/about, Read 2021-03-10

[3] Grant B.L. "Treating Grapevine Problems: How To Take Care Of Grapevine Issues", https://www.gardeningknowhow.com/edible/fruits/grapes/treating-grapevine-problems.htm (2020)

[4] Burruano S. "The life-cycle of Plasmopara viticola, cause of downy mildew of vine". *Mycologist*, *14*(4), 179-182. (2000)

[5] Jones D.S. and McManus P. "Downy Mildew on 'Valiant' cultivar (Vitis vinifera x Vitis labrusca x Vitis riparia)", https://commons.wikimedia.org/wiki/File:Downy_Mildew_on_%27Valiant%27_cultivar_(Vitis_vinifera_x_Vitis_labrusca_x_Vitis_riparia).jpg (2015), CC BY-SA 4.0 via Wikimedia Commons

[6] Bacon R., Carisse O., Lasnier J. and McFadden-Smith W. "Identification Guide to the Major Diseases of Grapes", https://agr.gc.ca/eng/agriculture-and-the-environment/agricultural-practices/agricultural-pest-management/agricultural-pest-management-resources/identification-guide-to-the-major-diseases-of-grapes/?id=1544449361476 (2018)

[7] Halleen F. and Holz G. "An Overview of the Biology, Epidemiology and Control of Uncinula necator (Powdery Mildew) on Grapevine, with Reference to South Africa". *South African journal of Enology and Viticulture*, *22*(2), 111-121. (2001)

[8] Macdonald O. "Uncinula Necator On Grapes", https://commons.wikimedia.org/w/index.php?curid=971184 (2005), CC BY-SA 3.0 via Wikimedia Commons

[9] Iqbal H., "PlotNeuralNet", https://github.com/HarisIqbal88/PlotNeuralNet

[10] Brownlee J. "How to Configure The Number of Layers and Nodes in a Neural Network" ,https://machinelearningmastery.com/how-to-configure-the-number-of-layers-and-nodes-in-a-neural-network/ (2018), Read 2021-03-17

[11] Bora R. "Batch Normalization — an intuitive explanation", https://towardsdatascience.com/batch-normalization-an-intuitive-explanation-42e473fa753f (2020), Read 2021-03-17

[12] Hasan F. "Deep Learning With Weighted Cross Entropy Loss On Imbalanced Tabular Data Using Fast AI", https://towardsdatascience.com/deep-learning-with-weighted-cross-entropy-loss-on-imbalanced-tabular-data-using-fastai-fe1c009e184c (2020)

[13] Pérez-Expósito J.P., Fernández-Caramés T.M., Fraga-Lamas P. and Castedo L. (2017), "VineSens: An Eco-Smart Decision-Support Viticulture System", *Sensors*, *17*(3), 465, (2017)

[14] Ronneberger O., Fischer P. and Brox T. "U-Net: Convolutional Networks for Biomedical Image Segmentation", *International Conference on Medical image computing and computer-assisted intervention*, 234-241, Springer, Cham, (2015)

[15] Guzhva O., Ardö H., Herlin A., Nilsson M., Åström K. and Bergsten C. "Feasibility study for the implementation of an automatic system for the detection of social interactions in the waiting area of automatic milking stations by using a video surveillance system", *Computers and Electronics in Agriculture*, *127*, 506-509, (2016)

Master's Theses in Mathematical Sciences 2021:E13
ISSN 1404-6342

LUTFMA-3440-2021

Mathematics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/