DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2021

Using AI to Estimate Height of Plants through Surveillance Cameras at an Industrial Scale

CNNs on Basil Plants with Robel Poles

FILIP VON REIS MARLEVI

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Author

Filip von Reis Marlevi, marlevi@kth.se
School of Electrical Engineering and Computer Science
Degree Project at the Department of Intelligent Systems
Degree Project in Computer Science and Engineering, specializing in Systems, Control and Robotics
KTH Royal Institute of Technology

Place for Project

Stockholm, Sweden
At Svegro and Codon Consulting

Examiner

Mårten Björkman
Associate Professor
Stockholm, Sweden
KTH Royal Institute of Technology

Supervisor

Josephine Sullivan
Associate Professor
Stockholm, Sweden
KTH Royal Institute of Technology

Abstract

This report presents the results of investigations into whether, and how well, Artificially Intelligent (AI) algorithms can be used to estimate the height of plants by using images from regular surveillance cameras, setup over one of Svegros basil farms. The project is of great economical importance as too tall basil plants will not fit the shelves at stores and too small plants will disappoint customers. This is a part of a bigger movement at Svegro to automate the monitoring and caring for the growing plants, aiming at lowering energy consumption and minimizing waste. To measure the heights, rulers (Robel poles) were placed behind the plants that moved on conveyor belts under cameras so the plants' heights could manually be established from the number of visible lines on the Robel pole, not covered by the plant. The research problem was to engineer an AI based solution to predict how many lines were visible above the plant. After two months of gathering images and manually annotating them, three Convolutional Neural Network (CNN) models of varying complexity were trained on the images of individual Robel poles from the basil field. Results obtained with Grad-CAM showed that the networks do not learn to count the lines but to correlate the leafs size and shape to the height. The best score was a Mean Absolute Error of 0.74 and a Mean Squared Error of 0.83, where a MAE of 2.53 and MSE of 11.11 corresponded to just predicting the data sets median. This was achieved with EfficientNet0B. The results were compared with a human being's performance which showed that the human still performed better but due to the noisy data, the results are considered impressive and the score exceeded the expectations of the team at Svegro so the final model is now used there today. It was also shown that reasonably good results could be obtained even without the Robel pole in the training images, meaning the Svegro team could stop setting out the Robel poles but with a slight loss in precision.

Suggestions for improvements, like changing the design of the Robel poles, are presented to aid future research to fully automate the process with higher accuracy.

Index terms - CNN, Plants, Phenotyping, Robel Pole, Automation, Computer Vision.

Sammanfattning

I denna rapport presenteras resultaten från undersökningen av huruvida en Artificiellt Intelligent (AI) algoritm kunde användas för att estimera höjden på plantor från bilder tagna med övervakningskameror som satts över en av Svegros basilikaodlingar. Projektet är av stor ekonomisk vikt eftersom basilikan inte får vara för lång för att inte passa i hyllorna i butiker eller för korta för att göra konsumenterna missnöjda. Detta är en del av ett större projekt som innebär övergång till automation av övervakandet och odlandet hos Svegro med förhoppningen om att kunna minska energiförbrukningen och svinnet. För att mäta höjden placerades linjaler (Robel-pinnar) bakom plantorna som rörde sig längs ett stort rullband under kameror så att plantornas höjd manuellt kunde bestämmas från antalet sträck på linjalen som täcktes av plantan. Forskningsuppdraget blev därmed att ta fram en AI som kunde uppskatta hur många linjer som syntes. Efter två månaders samlande av data samt manuellt annoterande av dem testades tre CNNs (Convolutional Neural Network) med olika komplexitet genom att tränas på bilderna av individuella Robel-pinnar från basilikafältet. Resultat som erhölls med Grad-CAM visade att nätverken inte lär sig räkna linjerna utan istället korrelerar basilikabladen form och storlek till höjden. Det bästa resultatet som erhölls var ett MAE (Mean Absolute Error) på 0.74 samt MSE (Mean Square Error) på 0.83, där ett MAE på 2.53 och ett MSE på 11.11 hade motsvarat gissande på datasettets median. Detta resultat erhölls med EfficientNet0B. Resultatet gämfördes med en människas prestation vilket visade att människan presterade bättre, men på grund av osäkerhet i datan ansågs resultaten vara imponerande och överträffade förväntningarna från teamet på Svegro som idag använder modellen. Det visades även att tillfredsställande resultat kunde erhållas med bilder som inte innehöll Robel-pinnen vilket innebär att teamet på Svegro skulle kunna sluta sätta ut Robel-pinnarna i krukorna men då med en liten förlust i precision.

Förslag på förbättringar, som att förbättra desingnen på Robel-pinnarna, tas också upp för att hjälpa framtida forskning att snabbare komma till resultat som kan leda till en fullständigt automatiserad process med bättre noggrannhet.

Indexerings termer - CNN, Plantor, Phenotyping, Robel-pinne, Automatisera, Bildbehandling.

Acknowledgements

First and foremost I would like to thank Mikael Huss, Ph.D., who was my supervisor while working at Codon Consulting and for generously sharing his expertise. I am also grateful to Erik Fredlund, Ph.D and CEO for Codon, for giving me the opportunity to work on this project. Also Alejandro Fernandez, Ph.D., deserves great appreciation for helping out with all the technical challenges and setup of the Codon server.

I would also like to extend my appreciation for my supervisor at KTH, Josephine Sullivan, Assoc. Prof., for accepting me on a short notice and guiding me through this report. The same goes for my examiner, Mårten Björkman, Assoc. Prof.

Lastly, a big acknowledgment goes to Aleksander Marlevi, Ph.D., for helping out with the tedious annotations.

Contents

Chapter 1

Introduction

The world's population is constantly growing and so grows the demand for food. Many reports envisage that more grown and vegetarian food will be necessary. In 2017, it was predicted that by 2050, the world would need to increase the food production by approximately 25% - 70% [16].

Adding to the fact that we need more food, we are faced with the problems caused by global warming. Changes in the climate can lead to lower yields from farms that cannot adjust their production or change seeds to others that can cope better with the new weather and climate.

To deal with this, a good course of action is to find techniques that can determine which plants grow well in different conditions and techniques to map how different surroundings affect a plant's attributes [20].

Another word for determining a plants attributes is phenotyping. Among one of the most important phenotypes for determining how much a plant has grown is its height, which also indicates how much produce the plant will be able to yield. The height of a plant can be measured in several different ways: the depth of the root system, the surface area of individual leafs, the lowest root point to highest plant point and the plants height from the soil level to the top of the plant. Also, traits that happen on the cellular level like protein and cell wall synthesis can be measured and indicate how much the plant is growing [20]. Since the root system is not easily measurable due to it being under ground, and the extraction of proteins and cell wall analysis requires a lab, the easiest way to measure a plant's height is to measure its height above ground [15], [20].

To manually measure anything is often tedious, time consuming and repetitive. Such tasks are generally not appreciated by human beings that rather strive to make more of their abilities. From a company's standpoint, just find-

ing employees for these trivial, but yet necessary tasks can be hard and then the salaries to these employees are expenses that the company would rather cut down on.

To stop having to perform tedious and expensive labor, the task either needs to be deemed obsolete or automated. As the task of measuring the height of plants cannot be deemed obsolete, it needs to be automated. The ways the task could be automated are many but, as discussed in Section 2, some are better than others and in the end they all have to make economic sense.

Due to the recent strong development in Artificial Intelligence (AI) research, an increasingly common way of solving automation tasks is to gather as much data as possible regarding the task, feed it to a Neural Network (NN) of some sort, and hope it learns the patterns in the data correlated to the task. First and foremost, in order to make a classical NN learn anything, the data needs to be annotated (have labels of what they represent), referred to as supervised learning. This annotation is in itself a tedious task but does only need to be done once, after which the NN can learn enough from the annotated data.

The group of NNs that are commonly used for image data are called Convolutional NNs (CNNs) and are what were used in this project.

1.1 Reason for this Project

The company Svegro wants to change the way they grow plants by implementing AI technology in their greenhouses. Their objective is to increase yields while at the same time reducing the manpower and energy needed to produce these increased yields. They have therefore applied for VINNOVA funding to support their research into the feasibility of AI to meet their objectives [32].

1.2 Overview of the Setup at Svegro

Svegro grows several different plants but this project focused on the basil plants (Italian Genovese Basil - *Ocimum basilicum*), their most popular plant with 5.5 million sold per year [17]. They are grown in long rows that move on a conveyor belt for around 20 days until they have grown to size. In Figure 1.1 the growing area for the basil plants can be seen.

The task of automating the basil production has been divided into three different sub tasks, they are:

1. **Estimate the height of plants from the cameras**

Figure 1.1 – A view from one of the corners in one of the greenhouses at Svegros complex where basil can be seen everywhere.

2. Detect anomalies in the plants from the images
3. Predict the harvest date depending on several variables from different sensors

This report will focus on the estimation of the height of plants from the cameras as this was of the greatest importance for the Svegro team and believed to be the most generalizable result that can be of interest for future implementations on other plants than basil, or other research.

One prerequisite in this project was that regular surveillance cameras were to be used due to their low cost. Their positions had already been chosen and they had been mounted. The cameras were set up 1.5 meter above the plants on a wall to the side of the conveyor belt. The cameras could not be set further down as it then would interfere with the mechanics of the conveyor belt. The cameras were aimed with a slight downwards angle to only capture the parts of the conveyor belt with the plants, as seen in Figure 1.3. The cameras were set up at a distance from each other to ensure that their field of view just overlap, to be able to keep track of the whole length of the conveyor belt with as few cameras as possible. Four cameras were required for the conveyor belt used in this project. An image from each camera is shown in Figure 1.2. One of the camera's position and mounting can be seen in Figure 1.3.

Also, it had been decided that white sticks with horizontal black lines were going to be placed behind the plants in some of the pots that were traveling under the cameras so that the camera could take images of the sticks, as seen in

Figure 1.2 – On the same day at the same time, this is the images that the four cameras respectively took. To the left, the first camera is looking at the beginning of they conveyor belt, then moving towards the right are the cameras that come later along the conveyor belt.

Figure 1.3 – The two images show how and where the cameras are mounted.

Figure 1.3. From the sticks, the height can manually be estimated as the total number of lines that are on the stick minus the number of lines that can be seen above the plant. On top of each stick is a QR Code that contains information of when the plant was planted, so that the current height can be correlated to data from other sensors from when it was planted. Svegro had made the sticks of white thin plastic with black lines at increments of one centimeter, effectively making it a ruler (Robel pole, see Section 3.1). A placeholder for a QR Code was set above the Robel pole to ensure data about the plant, such as when the plant was sown, could be easily stored and retrieved. A close up of the Robel pole can be seen in Figure 1.4. They were placed in about every third row at about the same position in each row so that it would pass under the camera in about the same way every time. The position was chosen so that it was easy to read it when placed in the back of a pot at an slight angle, as seen in Figure 1.2.

Svegro started collecting images from the cameras on the 29th of May 2020 at a rate of one image per hour from each camera. The cameras had a resolution of 2560×1920 pixels.

To make it easier to look at individual samples of each plant with a Robel pole, they used Detectron2 [34] to crop out the area of interest. Detectron2 is a powerful image recognition algorithm that in this case only required the Svegro crew to draw about 100 bounding boxes to specify where the Robel pole was in some of the large images from the cameras and then it could crop them out with good accuracy from any future image. This made it easier for the crew to inspect each sample fast and was what they thought would serve well as the data set for implementations of AI algorithms.

Regarding the setup at Svegro, it is worth mentioning that the roof of the growing house is of see through glass so clouds and the beams in the roof cast shadows over the plants. Also, during the night, if the plants require more light, armatures with High-Pressure Sodium (HPS) lamps are used. These lights are known to emit yellow light, as can be seen in Figure 1.5 compared to Figure 1.4.

1.3 Conditions that Affected the Project

Due to the current ongoing Corona Pandemic, a visit to the Svegro greenhouse was not suitable. The whole work had to be done remotely, which meant that I could not personally perform upgrades to the setup that could have made the progress much easier and faster. Thankfully it worked well enough to investigate the setup that they had. My thoughts and insights on how to make the

Figure 1.4 – A normal crop (from a larger image) only containing the Robel pole with some background. from the Detectron2 model.

Figure 1.5 – Also from Detectron2 as Figure 1.4 but this time during the influence of the HPS lamps

setup better for future implementations will be brought up as Future Work in Section 6.2.

1.4 Hypothesis and Research Question

As has already been mentioned, Robel poles will be used to indicate the height of the plants. The idea from Svegro is that these Robel poles can be detected in the pictures of the field of plants, cropped out and then annotated manually by counting the visible line for every Robel pole seen. An AI, a CNN for example, can then train on this data set and should then be able to detect how many lines are visible on new images of Robel poles. The CNNs estimation of the number of visible lines on the Robel pole can then be turned into a height estimation of the plant which should represent the whole row of plants.

The implementation should preferably be easily maintained and fast to train due to limited computational power at the company.

So the research question is: How accurately can an CNN give estimations of plants heights with basic surveillance cameras in an industrial greenhouse using Robel poles? With the goal of achieving a MAE of less than one.

1.5 Outline

The rest of the report starts with Related Works where it is concluded that the method of choice is among the most cost effective although more invasive than others. What is relevant for understanding what will be done is explained in Theory. The Methodology section describes how the procedure of operating on the data is done and then all the results are presented in Results and Discussion. While performing preliminary tests, a lot of new insights were made that led to more important tests that are discussed and presented in Results and Discussion. The final conclusion is presented in Conclusions and lastly ideas for Future Work are proposed.

Chapter 2

Related Work

As shown in Figure 2.1 from [20], the research in the field of tracking plants phenotyps, like height, is growing with over thousands of publications per year for the last years.

The research that has been done shows very good results and use different sensors and techniques to measure plant growth in different ways, some of which will be mentioned here and evaluated against how well they can be used in this project and regarding this projects boundary conditions.

2.1 3D Scanning of Plants

A well studied way of tracking the growth and height of a plant is to use 3D scanning techniques.

[10] used laser-scanning to capture a 3D model of plants from 12-16 pots

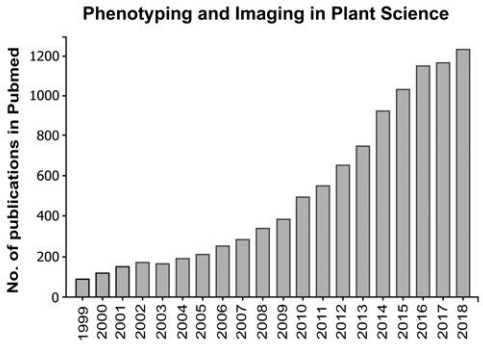


Figure 2.1 – This figure, taken from [20], shows the number of publications on the topic of plant phenotyping over the last years.

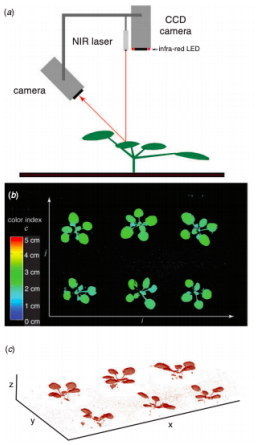


Figure 2.2 – A figure taken from [10] showing their setup and results from the 3D scans of the plants to give an understanding of how it worked.

with 6 plants in each pot simultaneously. The growth was then estimated based on the rosette size and the plants leafs hyponasty (upwards bend of leaf). The method was shown to be robust to different lightning conditions when tested on Arabidopsis plants and, as they concluded, their method is completely non-invasive to the plants and requires nothing but the laser scan itself, in a well known location. See Figure 2.2 to see how the laser was set up and the resulting graphs of the plants.

[1] used a light-field camera to obtain both 2D images and 3D information. Their software in combination with the light-field camera is presented as the product Phytotyping4D. It was shown to be able to keep track of 12 pots, with 4 Arabidopsis plants in each, at the time and to produce time series of the plants growing. By using segmentation techniques, they showed that they could keep track of each leaf of the plant and thereby monitor the growth, hyponasty and rosette development with very high accuracy. This method was non-invasive and only required a preliminary calibration of the camera. In Figure 2.3 some images and graphs from [1] are shown that give an understanding of how the 3D and segmenting was used to study the plants leafs.

The above mentioned strategies all require expensive hardware. [3] present good 3D modeling of up to nine plants with a photometric stereo technique that is much cheaper but requires a very controlled setting of lighting. Unfortunately the lightning is not easily controlled in this set up as the roof is see through and the HPS lamps causes inconsistencies.

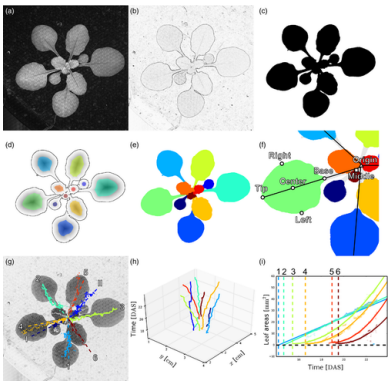


Figure 2.3 – A Figure taken from [1] showing their results from the 3D scans of the plants to give an understanding of how it worked.

2.2 Regular Cameras

A big research topic in computer vision is how to make height estimations from regular 2D images. Researchers showed in [6] how reliable estimations of peoples height can be done if enough dimensions in the surrounding image can be known, like a reference plane if the image is taken with an angle from above or a line on a wall that the persons highest point can be correlated to if the image is taken from the side directly towards the wall.

In [18], a way of measuring the height of rice plants in a big field is presented. A camera was placed in front of four rice plants that were of interest to measure, as seen in Figure 2.4. In Figure 2.4, it is also shown that behind the plants, a white mesh is placed (preferably a white board would have been placed behind the plants but due to wind and outside conditions a mesh had to do) and alongside the mesh a stick with markings that indicate incremental steps of 0.5 cm along its height was placed (Robel pole). The setup is shown as a graph in Figure 2.5. Using simple geometry, from Figure 2.5, the equation for a plants height is calculated according to equation 2.1. Images were taken every 3 hours, the camera used in this paper had a resolution of 2560×1920 pixels and the camera was placed 280 cm from the rice plants ($L - \delta L$). To get the height readings for each rice plant in each photo, the authors had to estimate horizontal lines from what could be seen as the top of each plant to the Robel pole and from there make the reading.

$$h_t = \frac{L - \delta L}{L} h_m + \frac{\delta L}{L} h_c \tag{2.1}$$

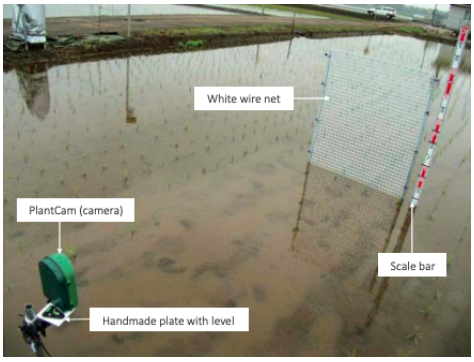


Figure 2.4 – Showing the setup and Robel pole for the paper measuring rise height [18].

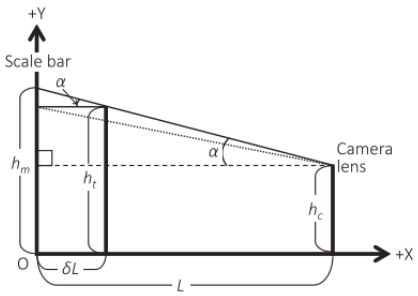


Figure 2.5 – Graph of the setup from Figure 2.4 which defines the variables in equation 2.1 [18].

2.3 Other Sensors

A interesting and extensive comparison between five different height measurement sensor was conducted by Wang, X., Singh, D., Marla, S. et al. [33]. The methods were also compared to their price. The five sensor that were used were:

1. ultrasonic sensor
2. LIDAR-Lite v2 sensor
3. Kinect v2 camera
4. array of four high-resolution cameras
5. a digital camera mounted on an unmanned aerial vehicle platform (drone)

All but the last sensor was mounted on a plate that was attached to a tractor that drove around a field, keeping the plate horizontal over the plants so the sensors could capture the height of the plants. Apart from the LIDAR techniques that has already been mentioned to work well here by other papers, this paper found that ultrasonic sensors could give measurements that correlated very well with the true value, also the Kinect camera worked well and they both were relatively cheap.

2.4 Evaluation of Related Works Relevance for this Project

Out of all the methods presented by other authors, the one from [18] is the most invasive to the plants as it requires a background and a stick to be placed in the field. However, it is by far the least expensive solution. The only thing that costs anything substantial is the camera which yet can be far cheaper than any 3D camera.

As mentioned in 1.2, other goals like anomaly detection will also be investigated at Svegro. That project will certainly need surveillance cameras as is why this technology got implemented, hence if good results can be drawn from data from these cameras, the company will not have to set up other sensors, saving money.

To extract height measurements from the images like the researchers did when attempting to measure human beings [6] would be of interest, but this

is deemed not to work as there is no true ground plane from the angle of the cameras (it can be seen in Figure 1.2 that the ground plane is made up of the basils which constantly change). If the Robel poles had been acutely constant there could have been a possibility but as also seen in Figure 1.2, they vary in placement.

As the Robel poles in this study will be placed directly behind the plant, looking at equation 2.1, δL can be assumed to be 0, resulting in that $h_t = h_m$ which will be assumed for this problem.

To this day, this is seemingly the first paper attempting to combine Robel poles for annotation with neural networks to learn to estimate the number of lines still visible, and especially to find digital techniques to monitor basil plants at such a large scale.

Chapter 3

Theory

This section presents and explains the main techniques used in the project.

3.1 Robel Pole for Measuring Height

A Robel poles is a tool that is frequently used by biologists when measuring the biomass above ground. They are widely used due to their simplicity and since they can be made very cheaply and reused many times. The form of a Robel pole can vary but it is made up of a long pole that can be stuck down into the soil and has a length that exceeds the surrounding plants. Along the pole, markings are made at regular intervals. Often, these markings or lines have number next to them starting from 0 at the bottom of the pole to make it easy to identify which number of line is observed.

Effectively it works as a ruler that gets stuck in the ground with its zero indication at ground level. The height of the surrounding plants can be estimated from either the lowest number that is visible on the pole or by the number of lines that the pole is known to have minus the number of lines that still are visible on the pole (abbreviated L_V for Lines Visible), as those lines that are not visible are covered by the plant. The height is then obtained by multiplying the number of lines covered by the known measurement of the interval between the lines [29], [18]. In this application for example, the pole used has 16 lines at intervals of one centimeter. The height of these plants can then be estimated as $(16 - L_V) * 1 => 16 - L_V$.

3.2 QR Codes

QR Codes (abreviation for Quick Response Codes) are a sort of barcode that can encode messages as a pattern of white and black squares that has a very distinct and recognizable form. This makes them perfect for detection in larger images and to carry information of the thing it is connected to [13]. In this instance, they will contain information about the date when the plant that it is mounted above was sown.

3.3 Machine Learning and AI

In the area of machine learning, their are two broad types of learning tasks. The first one is called supervised learning and means that every datapoint that will be used as input has a label and so the machine learning algorithm has to learn how to derive new labels to new inputs of the same sort. The other sort is unsupervised learning which means the datapoints used as input do not have any labels. Then the model has to learn to classify the input by itself. The problem in this paper consists of images that has labels and so it is a supervised learning problem.

Apart from other methods like support vector machines, the neural networks that will be presented need no assistance in selecting what features to focus on as they can take raw input and learn what is of important from it. Today, Feedforward Neural Networks (FNNs) have gained a lot of popularity and is widely used in the area of AI, where it, when used with images as input, is most common as Convolutional Neural Networks (CNNs).

3.3.1 Feedforward Neural Networks (FNNs)

A FNNs objective is to calculate the optimal parameters Θ such that the equation $y = f(x; \Theta)$ becomes as close to $y = f^*(x)$ as possible, where f^* maps the input x to the output y [14].

FNNs are commonly based around perceptrons, see Figure 3.1.(a). A perceptron takes input x, multiplies it with a weight w and sums all its connections to a sum. This sum then passes through an activation function which gives an output from the perceptron. Each perception can hence be described by the mathematical notation $y = f(\sum x_i w_i)$ where f refers to the activation function that for example can be of the sort Linear, ReLu or Softmax [14]. It should be noted that here the w is what the network needs to optimize and hence what Θ referred to earlier.

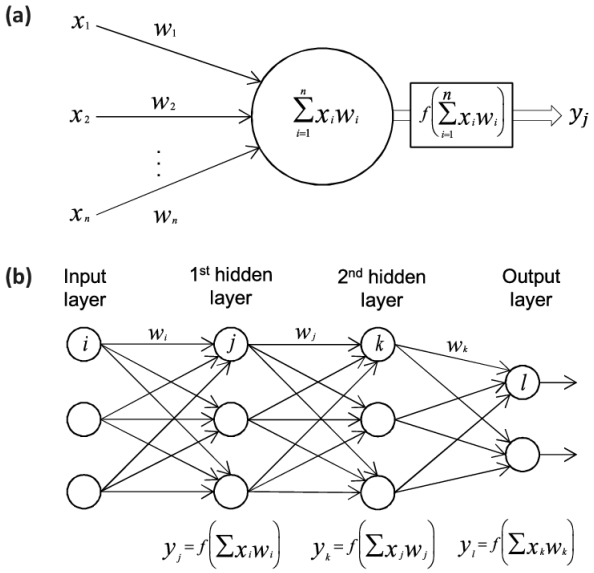


Figure 3.1 – (a) depicts a perceptron and how it works. (b) depicts several perceptrons, fully connected, to create a NN [31].

By using several individual perceptrons (also called neurons in this context) in parallel, different layers are created that are fully connected together in series to create different depths of the networks. The first layer is always referred to as the input layer, as this is the layer that takes the input. The last layer is called the output layer and the layers in between are called hidden layers since they have no direct connection to the outside world, see Figure 3.1.(b) [14], [31].

As each layer is fully connected to the other in front of it, the information from the input layer is sequentially moved to the next layer in the network until finally reaching the output layer, hence the name "feedforward".

At each layer the network learns more and more complex features of the input. The structure of the model is often made to converge towards the last layers to fit the label. For this often "Flattening" and "Dense" layers are used. When the data is divided into different classes, the output can be made to encode for a class as an one-hot encoded vector, the network is then referred to as classifying. If the labels do not consist of individual classes, but instead of numbers that even if not predicted spot on, still can be deemed good if numerically close, the problem is called regressive. The output of a FNN on a regression problem should then be a single continuous number value instead of an one-hot encoded vector.

The exact amount of hidden layers that gives the optimal result cannot be

calculated and so cannot the number of neurons in each layer. Hence these design parameters have to be made by the engineer along with other parameters that have to be manually decided in advance, these parameters are referred to as hyperparameters. Finding algorithms that give optimal hyperparameters is a big area of research [2].

3.3.2 Training FNNs

Once the FNN is designed so that it can produce an output for each image that corresponds to each image's label, the CNN can start training. Training means going through all the images of the data set a specified number of times (epochs) by taking a specified number of images at the time (batch size) and using a cost function to give a numerical score, indicating how well the predictions from the CNN was made compared to the labels. There are several different cost functions that suit different outputs and tasks, for example Ordinal loss and Cross Entropy loss. In the task of regression, a cost that can be used is Mean Squared Error (MSE) for which the average of the squared difference between the label y and the predicted output $\hat{y}$ is calculated, describe by the equation: $C = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$, where $i = 1, ..., n$ and n is the number of samples [14].

While training the input images are divided into batches corresponding to the hyperparameter *batch size*. During the training procedure, the cost function is used for each batch (where the cost functinos score is referred to as the loss) to give a score on how well the network performed, this score is then used with a backpropagation algorithm too make changes to the weights of the network. The changes to the weights should not be to drastic but made in small increments. The parameter for this is called $learning\ rate = \eta$ and is often much less than 1. Algorithms such as ADAM incorporate the technique of adaptive learning rates which is highly beneficial for achieving better performance [22].

The batch size is of great importance for making the CNN learn well. The batches should be large enough to be able to represent the distribution of different labels in the data set but not too large to prevent the network from taking gradual steps towards its optimum, minimum, value. If the loss of the training set gets unstable, the batch size should be altered [4].

3.3.3 Overfitting and Actions to Prevent it

Overfitting is a problem that can occur and needs to be avoided. It means that the CNN learns to recognize just the specific samples of images that it is training on, memorizing them in a sense. To recognize if this happens, after each epoch, the CNN gets to evaluate a set of images that it has not seen before during training. If the loss for this validation data set is about the same as for the training set, the model has not yet overfitted. To then finally test if the CNN works, it is advised to use data that has been collected at a specific separate time so that it cannot be exactly the same as the data it trained on. If the loss is about the same for new data, the model can be said to have learned to recognize the features that are general for the task [12]. The training set is suggested to be about 2/3 of the total size of the complete data set with the remaining 1/3 divided into validation and test [9].

To prevent overfitting, augmentations to the input images can be used. This means that the training data gets slightly altered before every time it is trained upon, for example rotating the image slightly, changing the brightness or shifting the image makes it so that the network cannot just remember specific images[27].

Apart from the main convolutional and fully connected layers, other layers like "Dropout" and "Maxpooling" are frequently used with the purpose of making the model more robust [23]. Dropout freezes certain parts of the model. Then, while training, those parts do not count during the forward pass or during the backpropagation, hence hindering the model from overfitting specific parts of itself to the training data [31]. Maxpooling works by dividing the feature space in a more coarse space and then taking the max value from each of the input space's positions that falls within each of the coarser segments [8].

3.3.4 Convolutional Neural Networks (CNNs)

In the field of AI, Convolutional Neural Networks (CNNs) are a very common group of FNNs that were inspired by the visual cortex to be used when images need to be analyzed. What sets CNNs apart from other FNNs is that they contain convolutional layers. These layers operate by dividing the images into smaller parts, starting from the top left corner, going to the right before going down and repeating until at the right most bottom corner, as depicted in Figure 3.2. These smaller parts are then convolved with filter masks, essentially matrices made of weights that are changed during training so that they together with the input give certain outputs for certain features in the images [7], [8]. Adding many convolutional layers can then be described to have the effect of

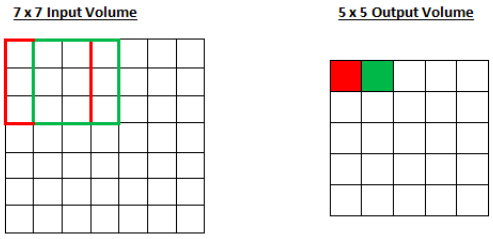


Figure 3.2 – A visual description of how a convolutional layer works by taking image segments correspond to red and green lines that then becomes output as a pixel to the next layer [8].

learning different sorts of features at each level. The first convolutional layers can learn to react to textures and fine grain features while the later ones can learn to react to how parts of the images are set up such as broad structural compositions [7], [31]. It is therefore tempting to have many convolutional layers, making a very deep model, but depending on the data, to deep architectures can cause overfitting [23], [24] and the vanishing gradient problem makes it hard to train too deep networks.

After each convolution it is convention to introduce non-linearities, this is achieved with an activation function which for example can be of the sort ”ReLu” or "Softmax", but also "Linear". More information of these activation functions can be found in [21].

When the convolutional layers are combined with the fully connected layers that often make up a FNN, the CNN architectures require a fixed input size. In many real life situations, images vary in size as for example different cameras can have been used, or the images are cropped out of large image to only include the region of interest. If this is the case, measures have to be taken. One approach is to resize the images to the smallest size of the whole data set, this means information will be lost as the number of pixels gets reduced [35]. Another is to add black pixels as a border round each image (zero-padding) to make the biggest dimension of the data set, thereby preserving all of the information in the image [28].

3.3.5 Modern CNNs and Implementing them

To make CNNs have become very easy. In Python the library TensorFlow can be used to create models with just a few lines of code. Therefore, it is easy to find models that have been created for specific data sets online. Researchers within this field make frequent publications of CNNs that exceed previous state

of the art models. Among the well known models today are VGG, ImageNet and ResNet. Recently, the EfficientNet architectures were shown to perform well while requiring relatively few parameters, meaning a powerful model that is fast to train [30]. EfficientNet takes advantage of the Mobile inverted Bottleneck Convolution (MBConv) layer, first seen in MobileNetV2 presented by Google, which was specifically designed for the purpose of bringing Deep Neural Networks to mobile phones, where memory and computational power is extra limited [25].

3.3.6 Grad-CAM

To understand what the CNN bases its predictions on, explanation techniques can be employed. Gradient-weighted Class Activation Mapping (Grad-CAM) can be used for this. Grad-CAM works for a wide range of classifying networks and produces a map of the input image that highlights the regions that were of the greatest importance for the the output it produced [26].

The Grad-CAM algorithm works by gathering importance weights (α) by calculating the gradient of the predicted class with respect to each convolutional layer's feature maps, that then are global-average-pooled and, they are then combined. Lastly, a ReLu is applied to obtain the output heat map [5]. An illustration of the algorithm can be seen in Figure 3.3.

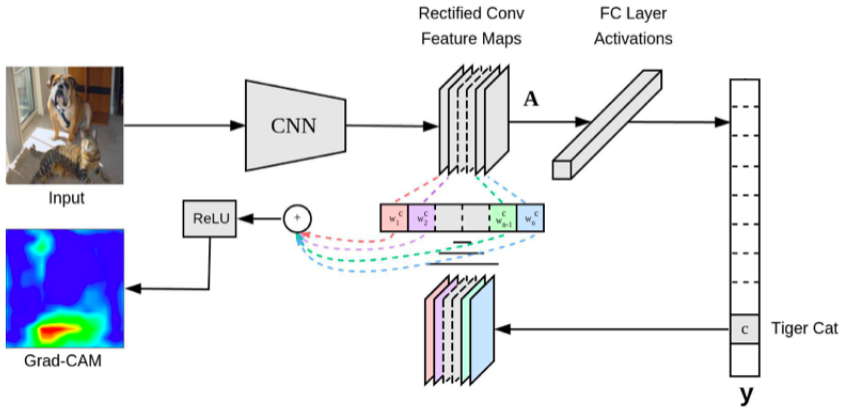


Figure 3.3 – Illustration of how Grad-CAM works step by step. Taken from [5] but modified.

Chapter 4

Methodology

Here we present how the number of visible lines above the Robel pole are estimated using machine learning.

4.1 Data Set

4.1.1 Image Preparation

To attain a data set suitable for neural networks, the images collected via the Detectron2 algorithm had to be modified. First, from looking at the data set, it was apparent that the Robel poles and QR Codes in some images would not be able to be read due to bad lighting conditions or bad camera angles, see Figure 4.1 for examples. The Python library **pyzbar** was used to detect and read the QR Codes in the images. Only images from Detectron2 that could be detected by **pyzbar** were used for further analysis. This decision was made as the data on the QR Codes were necessary for the future steps of the amortization of Svegros production, see the enumerated list in Section 1.2.

It is also apparent that the images that comes from Detectron2 are of varying size. Examples of this can be seen in Figure 4.1 where each image has an individual widths and heights. Since CNNs require a specified input size of the images, they all need to be made to have the same dimensions. Before training, every Detectron2 image with a readable QR Code was resized to the smallest width and smallest height in the data set.

The collection and annotation of images from the four cameras started the 29th of May and went on until the 31th of July. The data set was divided such that the test set consisted of the images only from the last week (the 24th of July to the 31st of July) as it would be interesting to see how the model performs on

Figure 4.1 – Examples of image crops that comes directly from Detectron2 where the QR Code cannot be read.

data that came after what it was trained on, such as it will be used in real life after the end of this project. The validation set consisted of 15% of randomly selected images from the training dates.

4.1.2 Manual Annotation Rules

The annotation of the images was not completely straight forward. A lot of uncertainty on how to count the lines occurred very early in the process, resulting in the creation of a set of guidelines for the annotators. More specifically, what was uncertain was mostly how to count the top line and when to stop counting at the bottom. In Figure 4.2 some examples of what could cause dilemmas when annotating the images are shown.

The first thing that was identified as very unclear was the top line, from where to start counting. The Svegro team said that the top most line should not be counted as it only served as a marking for where the QR Code paper should be put on the stick. In Figure 4.2a, 4.2c and 4.2f, the top markings are clearly seen as the line that is a bit longer than the others, right under the paper where the QR Code paper is. For the other examples in Figure 4.2, the top marking is either just barely visible or not at all due to the QR Code paper just covering it. To the human eye and to someone who knows how it should look, it is quite easy to know where it is and to start counting from below it.

The second rule that was set was that the highest leaf that is in front of the ruler should count as its highest point and only the lines that are above it should be counted when accounting for the horizontal angle the camera had to the Robel pole. In other words, one should imagine that the lines are extended over the whole Robel poles width (much like in [18]). Examples of when this rule was employed are shown in Figure 4.2d, 4.2e and 4.2f.

The third and final rule that had to be established was how a line should be counted if the top leaf just about touched or partially covered a line in the vertical direction. For example, in Figure 4.2b and 4.2f the highest point of a leaf that covers the Robel pole touches a line. It was decided that a line that is touched should not be counted according to the Svegro team that made them.

The Robel pole was known to have 16 lines. Since the scenario of all lines being covered means a label of 0, in total: 17 different labels will be used (0 to 16). Notice that the images will be labeled with the number of lines seen on the Robel pole and not its height. The height is however easily determined from the number of lines minus the number of visible once (16 - prediction from model).

(a) Quite clear example of 14 lines.

(b) 13 lines but the top marking is unclear and the bottom line is barely visible.

(c) Clear top line but unclear bottom line. Gets annotated as 13.

(d) The top line is barely visible and at the bottom, one line can be seen partially although it is below a leaf that is higher to the left of the lines. gets annotated as 11

(e) At the bottom, there lines can be seen although they are below a leaf that stretches higher on the left side of the ruler. Gets annotated as 10.

(f) Pretty clear but to the far left of the ruler, a leaf stretches higher than on the right side, making one line not count. Gets annotated as 6

Figure 4.2 – Uncertainties of which lines to include in the count.

4.2 Implementing AI models (CNNs)

As stated in Section 1.4 preferably the network used should have a small memory footprint and computationally efficient. Therefore we initially test simple and small networks and then increased their size and depth until satisfactory results were achieved.

Since the problem fundamentally is to make predictions on images, the models that will be used are CNNs. As the problem is believed to be to get the model to react on the number of black lines visible in an image with a white background (although surrounded with green leafs), a candidate that comes to mind is a network that works well for the MNIST data set (a well known data set which consists of white images with black numbers written on them).

Models used for MNIST are usually set to classify as every digit should be just that digit and just because a prediction is close numerically, does not mean that it is better than any other guess. In this problem however, it is better to be close to the target label than to be far away. For example, if the label for an image is 9 (9 lines are visible), then a prediction from the model of 8 is much better than a prediction of 6 since 8 at least is close and would be of better information to the Svegro team, trying to estimate when the plants will be fully grown and ready to be harvested. Therefore a model used for MNIST would have to be converted to a regression model. Other models that will be used will also be made to work as regression types. This is easily done by adding a final densly-connected layer with liner activation that outputs one digit and exchanging the last softmax activation function to a relu. A relatively simple architecture that was reported to work well for MNIST is shown in Figure 4.3 where it has been changed to work for this input and output. This model will be referred to as the **SMALL model** as it has the least deep architecture compared to the others used in this report.

Compared to the SMALL model, a slightly deeper CNN known to perform well on the CIFAR10 data set will be used. The CIFAR model was also changed to fit this problem, it will be referred to as the **MEDIUM model** and its architecture is shown in Figure 4.4.

Lastly a more advanced model called EfficientNet0B will be used to see if the model complexity can increase performance. The EfficientNet0B has a structure that, due to its complexity, has to be depicted as in Figure 4.5.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 155, 132, 32)	896
last_conv (Conv2D)	(None, 153, 130, 64)	18496
max_pooling2d (MaxPooling2D)	(None, 76, 65, 64)	0
dropout (Dropout)	(None, 76, 65, 64)	0
flatten (Flatten)	(None, 316160)	0
dense (Dense)	(None, 128)	40468608
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 17)	2193
dense_2 (Dense)	(None, 1)	18

Figure 4.3 – The TensorFlow summery for the model refereed to as SMALL.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 157, 134, 16)	448
leaky_re_lu (LeakyReLU)	(None, 157, 134, 16)	0
conv2d_1 (Conv2D)	(None, 157, 134, 32)	4640
leaky_re_lu_1 (LeakyReLU)	(None, 157, 134, 32)	0
max_pooling2d (MaxPooling2D)	(None, 78, 67, 32)	0
dropout (Dropout)	(None, 78, 67, 32)	0
conv2d_2 (Conv2D)	(None, 78, 67, 32)	9248
leaky_re_lu_2 (LeakyReLU)	(None, 78, 67, 32)	0
conv2d_3 (Conv2D)	(None, 78, 67, 64)	18496
leaky_re_lu_3 (LeakyReLU)	(None, 78, 67, 64)	0
max_pooling2d_1 (MaxPooling2	(None, 39, 33, 64)	0
dropout_1 (Dropout)	(None, 39, 33, 64)	0
flatten (Flatten)	(None, 82368)	0
dense (Dense)	(None, 256)	21086464
leaky_re_lu_4 (LeakyReLU)	(None, 256)	0
dropout_2 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 17)	4369
activation (Activation)	(None, 17)	0
dense_2 (Dense)	(None, 1)	18

Figure 4.4 – The TensorFlow summery for the model refereed to as MEDIUM.

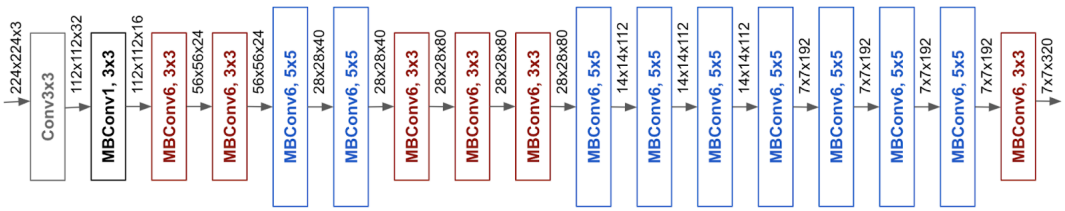


Figure 4.5 – A visualization of the model called EfficientNet07 [19]. Notice that only the blocks are correct and not the sizes, printed as numbers next to the intermediate arrows.

All of the implementations were done using the coding language Python and the implementation of the CNN was done using TensorFlow.

To hinder the models from overfitting to the training data, augmentations were used. The augmentations that were considered suitable for this data set were:

- Rotation between -10 and +10 degrees
- Shift horizontally from -12 to +12 pixels
- Shift vertically from -5 to +5 pixels
- Brightness change from 60% to 140%
- Shear horizontally from -12 to +12 pixels

The augmentations were all employed at the same time with parameters taken from within the boundaries randomly. The corners of the images that were exposed after the augmentations were filled with black pixels.

Lastly before the images were fed to the model, they were normalized with the factor $1/255$ to be have values between 0 and 1 instead of 0 to 255.

The regression models were trained with the loss as MSE using Adam optimizer with a learning rate of 0.001 (and other parameters set to default from the TensorFlow implementation). The networks were set to train as long as the validation MSE for each epoch improves with at least 0.0001 within the 10 latest epochs, then the training was stopped and the version that worked the best for the validation MSE was saved and further analyzed for performance (known as early stop callback with patience 10). The batch size was set to 50.

Every model was trained 3 times per implementation to make sure that the randomly initialized weights and shuffling of images during training not caused the model to perform badly just once. If the 3 runs ended with uniform results, the training procedure was assumed to be robust to the initialization. The best results from the 3 runs for each model were used for further comparison.

The training was done on a Nvidia GeForce GTX 1080, 8GB GPU.

4.2.1 Evaluating Models Performance

Since regression models were used, the most suitable way of measuring accuracy was with MAE and MSE, where the MAE was used to evaluate performance and the MSE gave an indication on how many, very, bad predictions were made (as it weighs each error with its square which gets significantly

larger for large errors) [11]. To get clear indications on how the models predict, confusion matrices were used by rounding the output of the model to the closest integer and bounding them by 0 and 16.

The team at Svegro said that a MAE of less than 1 would be considered a success. A MAE of 2 would still be within the margins of what they would be able to draw good conclusions from since the plants on a row do vary in height among themselves with around 2 cm.

It is of high interest to know what the model bases its predictions on, what features in the image it reacts to, the Grad-CAM algorithm was used for this. Grad-Cam only works for classifying networks which means we cannot use it directly with our regression models. Therefore, just when using Grad-Cam, the models were re-modified to classification mode.

Chapter 5

Results and Discussion

The testing began with a preliminary test with just a small amount of data to give insight into how the models behaved on the data, leading to the creation of three versions of the data set. These data sets were then used for training the proposed models. The results will be presented and compared to each other and against a human being's. All the relevant discoveries included in the final system are described here.

5.1 Preliminary Tests with CNNs

Since the gathering of data took significant time, preliminary tests were conducted along the way to gain insight in how CNNs would perform on the data set and to see whether CNNs would give satisfactory results from just a small amount of data.

5.1.1 Preliminary Tests of CNNs

Preliminary tests were performed after having two weeks of annotated data. The small CNN referred to as the SMALL model was implemented as classifying mode. All of the annotated data set up to this point was used.

This first small CNN achieved a classification accuracy of 75% on the test set after 20 epochs. As the data set consists of 17 discrete labels, a total guess by the network would have resulted in an score of $(1/17 = 5.88\%)$. Since 5.88% is much less than 75%, this gave high hope that a CNN of quite low complexity could solve the problem of height estimation well.

However, when the model was evaluated on completely new data from the following week, the accuracy decreased from 75% to around 10%. This large

Figure 5.1 – Grad-CAM result overlaid on some sample images from the data set, showing which areas of the image played the biggest part to the CNNs output. Brighter colors like yellow indicate stronger importance than darker colors, like blue.

decrease in performance means the model did not learn to recognize features in the images general for other days, rendering it useless. As mentioned in Section 1.4, the hypothesis was that the CNN should learn to identify the number of lines on the Robel pole but due to these results, this had to be investigated.

To investigate what the model made its predictions on, Grad-CAM was used. Some results of the Grad-CAM, overlaid on the input image, are shown in Figure 5.1.

As can be seen in Figure 5.1, the model clearly focuses the most on the QR Code. Barely any highlighting is shown over the Robel pole, apart from the side edges of it in some cases. In conclusion, the model learned to "read" the QR Code and associate it with the height of each plant in this small data set. This is not acceptable so a solution that stops the model from being able to analyze the QR Code is required, as described in the next sub-section.

5.1.2 Unable CNNs to Read QR Codes

By using the Python QR Code reader package **pyzbar**, the exact coordinates for each QR Codes corners could be found in each image of the data set. By using the edges of these coordinates, a rectangle could digitally be placed over the QR Codes, thereby obscuring them. The color of the rectangle could be chosen arbitrarily but some thought went into it. By making the rectangle of a color that is not present in the rest of the image, hopefully the CNNs would be able to easily correlate just the color (and not even the shape) to an area that should have no impact on the output. Judging by looking through a lot of the

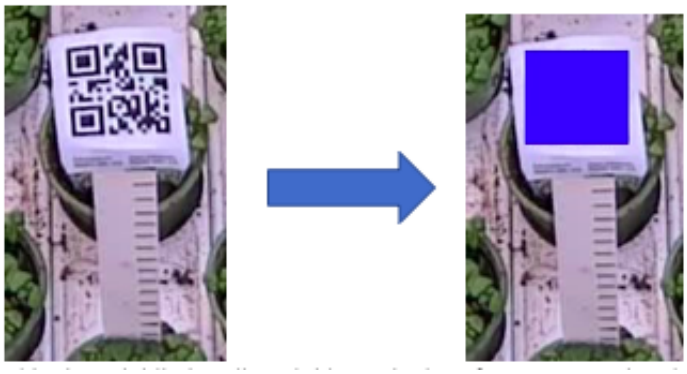


Figure 5.2 – To the left, a image of a Robel pole with a QR Code in one of the pots is shown. To the right, the same image but with a blue rectangle placed over the QR Code is shown.

images, a color that is not present is blue. Hence the rectangle was made blue. An example of how the images was changed can be seen in Figure 5.2.

With this change to the images of the data set, the CNNs will have to find other features of the images than the QR Code to correlate to the number of lines visible on the Robel pole, preferably the lines in it themselves.

5.1.3 Preliminary Tests of CNNs on Images with Masked QR Codes

With the new and improved data set with masked QR Codes, the same model as in the previous section was trained and evaluated using Grad-CAM. The new results can be seen in Figure 5.3 where the output from Grad-CAM has been overlaid on the input image. This image should be compared to Figure 5.1, it is hard to see if a lot of focus is put on the blue rectangle or if it is just due to color distortion when overlaying the two images into one. One thing that can be seen is that more emphasis has been put on the contours of the plants and leafs than before. To get a clearer observation of this, in Figure 5.4, the input image is not present so that just the results from the Grad-CAM output is seen. It is clear from all those images that the CNN now do not get any information from the QR Code, as it is not observable. It is noticeable that not much emphasis seems to be put on the Robel pole, although, in two of the six images of Figure 5.4, the contours of the Robel pole has been highlighted, but for the rest it was not highlighted at all. Instead, most of the highlighting is clearly put on the plants themselves in such a way that the individual leaves is what seem to be important for the output. Studying the images leads to the

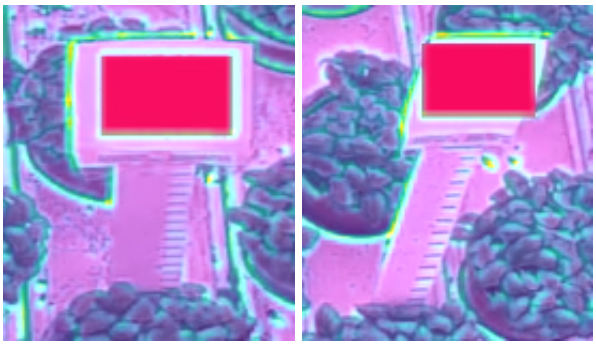


Figure 5.3 – Grad-CAM result overlaid on some sample images from the data set, showing which areas of the image played the biggest part to the CNNs output. Brighter colors like yellow indicate stronger importance than darker colors, like blue.

conclusion that it possibly is the contours of the leaf that can be correlated to the labels. This is understandable since small plants have smaller leafs and taller plants have larger leafs that also, due to getting physically closer to the camera appears bigger in the images.

The discovery that it is the leaf size and structure in the image that most easily correlates with the heights of the plants, and that the Robel pole is obsolete, suggests that the presence of the Robel pole possibly only has a negative impact on the CNNs performance as it hides more leafs behind it that could further give information to determining the height. Therefore the question of how good the performance could be without the Robel pole comes up. This is further developed in the Section 5.2.1.

5.2 Creating the Full Data Set

After the two full months of gathering data, the crops from the Detectron2 script of individual Robel poles turned out to have 3373 images with a readable QR Code and the possibility to count the lines on the Robel pole. The number of images from this set that was captured during the last week, that will be used for testing, was 533 images. This means that the final data set for training has 2840 images. The test set therefore accounts for 533/3373 = 15.8 %.

In Figure 5.5a, 5.5b and 5.5c the distribution of the labels in the training, validation and test data set respectively are shown as histograms. The histograms show the label distribution is not well balanced with especially the smaller values being under represented.

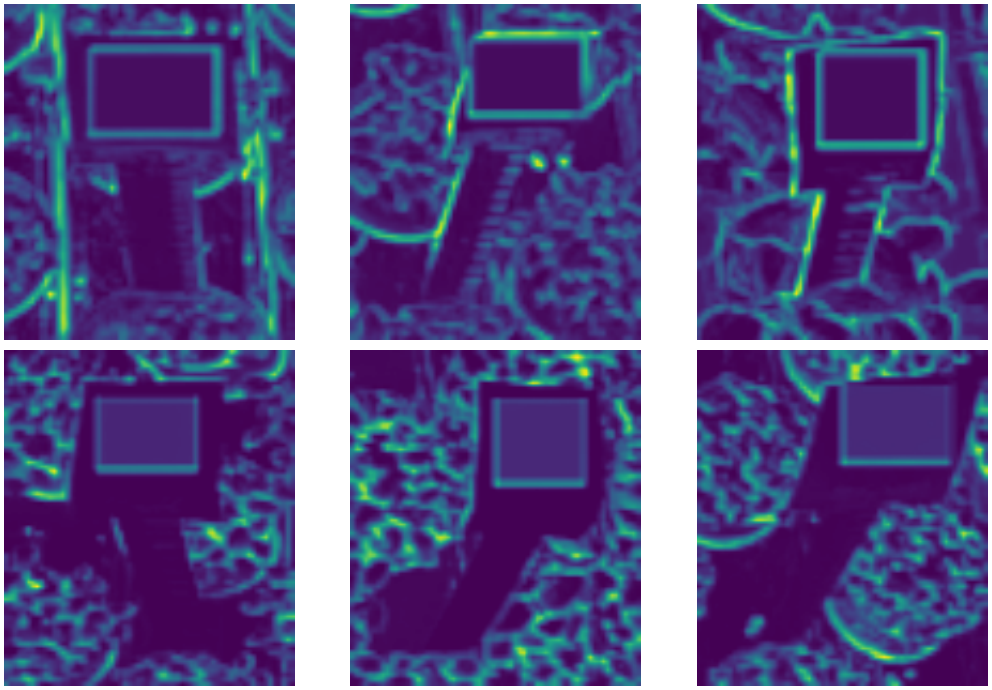
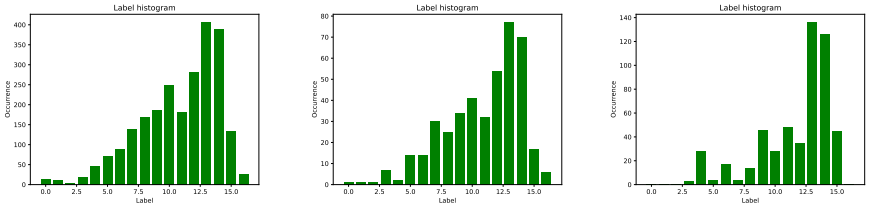


Figure 5.4 – Grad-CAM result showing which areas of the image played the biggest part to the CNNs output. Brighter colors like yellow and green indicate stronger importance than darker colors.



(a) Training data set. (b) Validation data set. (c) Test data set.

Figure 5.5 – Histogram of amount (y-axis) of labels (x-axis) that makes up the respective data sets.

The smallest width in the data set was 134 pixels and the smallest height was 157 while the largest width and height was 253 and 343 respectively. As described in the method section, Section 4.1.1, the standard approach of resizing all the images to the smallest size in the data set will therefore results in a data set with the dimensions of 157 as height and 134 as width. This data set will from here on be referred to as **resized**, examples are shown in Figure 5.7a.

Since the preliminary tests showed that the model emphasized the leaves' structure, it might have been the case that changing the aspect ratio of the images and distorting the leaves' shape may have had an adverse effect. To preserve as much information about the leaves' sizes in the images as possible, a second data set was created from the same images but without resizing. Instead all images got padded with a black border to have size 253×343. This data set is referred to as **padded**, examples are shown in Figure 5.7b.

5.2.1 Extracting Images Under the Robel Pole

As was concluded in Section 5.1.3 it would be of great interest to investigate how well a CNN performs without the Robel pole in each image. To not have to redo all the annotations that had been made on plants without a Robel pole, a data set was created based on the annotations that already existed. The new data set was obtained by going back to the large images from the surveillance cameras and cropping out a specified sized area immediately under the Robel pole crop that had previously been taken out. The area to crop out was set to have the dimensions 200×300. An example of these new images and the context of the original Robel pole image above it can be seen in Figure 5.6. More examples of this data set can be seen in Figure 5.7c.

This new data set is referred to as **under** (as the images of this data set comes from "under" the original ones), where the labels were taken to be the same as the respective image containing the Robel pole. This is not an optimal assumption as upon later inspection of the greenhouse, it was apparent that the height of plants in the same row, even next to each other, always varied from one to two centimeters. It is not clear how much impact this assumption has on the CNNs performance from this under data set but since each image contains multiple plants, the network probably averages its prediction across these plants, presumably making for a good and more robust indication of the height. Also, as the manually made labels have an error of ± 1cm it is of similar magnitude to the difference in heights between neighboring plants, we felt the labels transferred to the under data set were sufficiently accurate to be useful.

Figure 5.6 – The top image is a image that belongs to the regular data set of Robel poles that have been cropped from the surveillance camera images. The bottom image is a crop from the same original image from the surveillance camera but taken from below the top image, hence not containing a Robel pole.

(a) Examples from the resized data set.

(b) Examples from the padded data set.

(c) Examples from the under data set.

Figure 5.7 – Examples from the three different data sets.

5.2.2 Evaluating the Annotation Rules

The annotation was tedious and time consuming. Labeling images with many lines took much longer than those with fewer lines. For the whole data set it took on average 16 seconds per image to count all the visible lines, write it in the excel file and change to the next image.

The annotation rules were clear but it was in many cases not easy to decide if the leafs touched the line when the labeler needed to extrapolate the line over the ruler's width. This undoubtedly will have led to inconsistencies in how the images were labeled with $\pm$ 1 unit.

5.2.3 Evaluating the Data Collection Technique

Two main problems with the data set are:

1. the label distribution within the data set is unbalanced
2. there are labels that are somewhat noisy.

Looking at the distribution plots in Figure 5.5 it is noticed that much more images of high labels have been gathered than of low labels. Since the labels correspond to the reading of the Robel pole, this means that many images of short plants have been taken compared to tall ones. This came as a surprise as it was expected to be uniform since every Robel pole passes under the cameras and do not disappear until the end. The main explanation for this is that at the first day, only the basils at the beginning of the conveyor belt got a QR Code. For this reason, the short plants have been observed for longer, and since the plants move at a constant rate, this partially explains the linear slope that the distribution seems to have. It was also reported that the QR Codes that Svegro had made sometimes curled up on themselves at the end of the conveyor belt, due to being of such thin paper that it was affected by the surrounding moister. This would have caused less of the tall plants (low labels) to be found by the **pyzbar** reader and hence omitted to a greater extent as the curling presumably increased towards the end of the conveyor belt. Once this was noticed by Svegro, they switched to thicker paper to prevent this.

Regarding the labels 15 and 16, the low amount of samples there is explained by that the basils are allowed to start growing a little bit in a "growing chamber" before placed in the beginning of the conveyor belt. Therefore, few of them were expected to be so short.

However, It has been speculated in that expecting a uniform distribution might not be plausible at all to beguine with. The exponential cell division

of plants might also be a cause for shorter plants to more prominent in the data set, but this was never thoroughly investigated as it was hard to do while working remotely.

It should be investigated if bigger QR Code could make the QR Codes more easy to be read via the cameras, generating more data for every label. This would mean that more data would be generated in less time, speeding up the data collection period which would be of interest for future implementations for other plants at Svegro. Then also data from over-represented labels could be discarded to creating more uniform data sets so that the models do not risk becoming bias.

The second problem mentioned in the list above could be solved with a new design of the Robel pole. The lines that indicate incremental steps must go all the way across the ruler to make it easier to understand where a plants highest point is relative to the lines. Also, an idea is to invert the colors on the Robel pole so that the main part of it is black and the lines are white, with the intention that this will not reflect as much light blinding the camera. It would also be beneficial with smaller increments between the lines to get better precision, like the researchers that observed rice plants where increments of 0.5 centimeter were used [18].

5.3 Evaluating CNNs trained on the Full Data Sets

Once the data collection period and annotation was completed and the three data sets obtained, tests with the different CNN architectures were performed.

The median of the training set is 12 and if one only guessed that, the test result would be a MAE of 2.53 and a MSE of 11.11. Hence, any results lower than these scores show the network is doing better than just learning the distributions basic traits.

5.3.1 SMALL Model

The best results from the SMALL model are gathered in Table 5.1.

To begin with, it is apparent that the SMALL model did not exceed the goal of MAE on test set less than 1 as no setup managed this in Table 5.1. From Table 5.1, we can see that the SMALL model did not reach the goal of a MAE less than 1 on the test set for any of the setups. However, for both the resized and padded data set, it got very close.

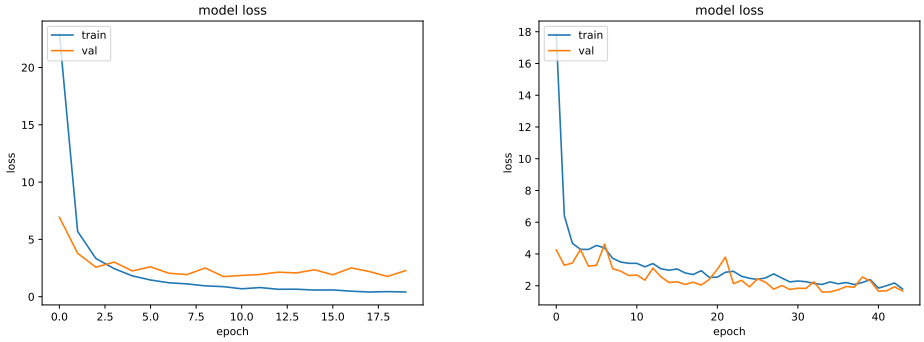
Table 5.1 – SMALL models best results for different data sets and with or without augmentations. The values are taken from the runs presented in Table A.1, see Appendix for more details.

Data set	Aug	# epochs	Sec per epoch	Train MSE	Val MSE	Test MSE	Test MAE
Resized	No	18	42	1.23	2.06	2.56	1.25
Resized	Yes	44	41	2.24	1.61	1.95	1.01
Padded	No	21	162	2.71	2.48	2.97	1.37
Padded	Yes	56	167	2.55	1.86	1.64	**1.00**
Under	No	25	112	1.04	2.15	4.08	1.58
Under	Yes	20	115	5.24	3.58	3.83	1.51

For every sort of image input, when it was trained without augmentations, it stopped training after much fewer epochs. To be exact, for resized it stopped on average 54-21 = 33 epochs earlier, for padded it stopped 44-25 = 19 epochs earlier and for under it stopped 29-24 = 5 epochs earlier. The stopping was due to the early stopping function that prevents the model from training further if the performance on the validation set is not improved within 10 epochs to prevent overfitting. The reason for no further improvements on the validation set when not using augmentations can be explained by that the model started overfitting to the trainingdata. This is apparent from the figures that show the training history for the model that trained without any augmentation, shown in Figure 5.8a where the training loss keeps going down but the validation loss does not follow.

With the augmentations implemented, much better performance was obtained. In Table 5.1 it is again shown that the models with augmentations to the input trained for more epochs (33, 19 and 5 for resized, padded and under respectively) and from Figure 5.8b it can be see that the validation loss follows the training loss very well (compared to in Figure 5.8a), it is apparent that the augmentations hinder the model from overfitting and allow the model to train for longer, learning the features of the image better and hence performing better on the test set as well. Therefore, for the rest of the testing, all networks were trained with augmentations employed.

When training on the resized data set, the training goes approximately 4 times faster than when training on the padded data set, and about 3 times fast than when training on the under data set. This is due to the images being bigger in the two last mentioned data sets and hence require more parameters to be



(a) No augmentations employed.

(b) Augmentations employed.

Figure 5.8 – The validation MSE per epoch when training the SMALL model on one of the resized data sets that gave the best performance, displayed in Table 5.1 .

updated.

Regarding the different data sets influence on the performance, Table 5.1 show that padded gave the best performance on the test set both on MAE and MSE. The difference is however not big from the results of train on the resized data set with augmentations, with only 0.01 in difference on the MAE, however since the MSE is bigger, some predictions have a larger error. The under data set resulted in worse results, still within the zone of MAE under 2 but here the MSEs were also significantly larger. The results on the under data set are promising given the potentially noisy training and test labels.

5.3.2 MEDIUM Model

The same testing procedure that was performed with both the SMALL model was performed with this MEDIUM model, but this time, as mentioned, augmentations were always used. The results are gathered in Table 5.2.

First and most importantly, as can be seen in Table 5.2, two out of the three different data sets have a MAE below 1, meaning that the goal was reached. The two data sets that produced those results were the resized and the padded data sets. This time resized gave better performance at a MAE of 0.79 after only 44 epochs where each epoch took 35 seconds, meaning about 25 minutes of training.

Table 5.2 – MEDIUM models best results for different data sets and with augmentations. The values are taken from the runs presented in Table A.2, see Appendix for more details.

Data set	Aug	# epochs	Sec per epoch	Train MSE	Val MSE	Test MSE	Test MAE
Resized	Yes	44	35	2.20	1.23	1.15	**0.79**
Padded	Yes	76	146	2.47	1.50	1.55	0.95
Under	Yes	50	102	3.10	2.14	3.22	1.39

5.3.3 EfficientNet0B Model

The results from the experiments in Section 5.3.1 and 5.3.2 show that the padded data set do not produce any significantly better results, but takes 4 times longer to train than the resized data set. We therefore then only applied the EfficientNet0B network to the resized data.

The best performance of the regular training progress for resized is shown in Table 5.3

Table 5.3 – EfficientNet models best results for different data sets and with augmentations. The values are taken from the runs presented in Table A.3, see Appendix for more details.

Data set	Aug	# epochs	Sec per epoch	Train MSE	Val MSE	Test MSE	Test MAE
Resized	Yes	138	68	1.27	0.89	1.11	**0.82**

The results are very close to the best one from the MEDIUM model. It should be noticed that it took this model 138 epochs before early stopping was employed which is significantly longer than the MEDIUM model which finished after 44 epochs.

When looking closely at the loss plots in Figure 5.9a (although hard to see with this scaling), it is noticed that throughout the training, the validation MSE is lower than training and that even after these 138 epochs, the validation MSE curve is still moving down. This is comparable to Figure 5.8b where the validation MSE curve looks almost completely flat after just about 25 epochs. Therefore, to see if even better results could be achieved, the same model with saved weights was further trained for 100 more epochs where the

epoch at which it produced the best validation MSE was saved for evaluation. The training history for this is shown in Figure 5.9b where it is clear that the slope is going down but slows down very much towards the end of in total 238 epochs. The final model gave the best result, seen in Table 5.4, with an MAE of 0.74, which in itself is marginally better than before, but the MSE is now also, for the first time ever, below 1 at 0.83, meaning that it predicts much fewer predictions with large errors, having better precision. The final test confusion matrix is shown in Figure 5.10, giving an understanding of how well the model classified certain labels.

As a last visualization of how the best model performed, in the top image of Figure 5.11 the predictions have been rounded to their closest integer (as is the procedure for the confusion matrix), subtracted with the label and then plotted as a histogram. To make comparisons with the other models, the same procedure was made for the best performing SMALL and MEDIUM model, seen lower down in Figure 5.11. From these plots it is apparent that the best EfficientNet model had less deviation in the accuracy (being more precise) than the others.

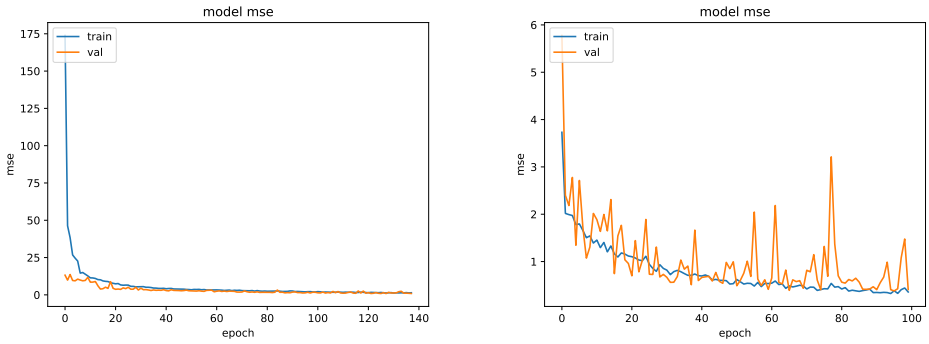
The extended training session was also applied to the under data set to see the top result from it, seen in Table 5.4. However, the results from this was much worse than ever before with a MSE of 6.79. This suggests that the extensive training is only harmful for this data set as it causes overfitting to the training datas' noisy labels.

Table 5.4 – EfficientNet models results with extended training sessions.

Data set	Aug	# epochs	Sec per epoch	Train MSE	Val MSE	Test MSE	Test MAE
Resized	Yes	238	69	0.36	0.40	0.83	**0.74**
Under	Yes	200	202	0.89	1.03	6.79	1.86

5.4 Human Being's Performance on Test Data Set

As mentioned, it was tedious to annotate the images manually and, although guidelines were introduce, in Section 4.1.2, for how to count the lines on the Robel pole to help produce consistent labeling, it was not always perfectly clear to the human eye how to judge some situations. For example when a leaf



(a) Training with the regular training procedure.

(b) Further training the model for 100 more epochs.

Figure 5.9 – The training loss (MSE) history for the best performing EfficientNet model on the resized data set.

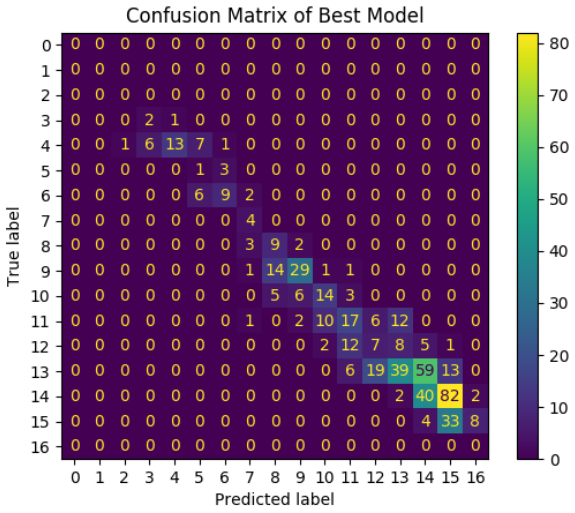


Figure 5.10 – The confusion matrix for the best performing model on the test set.

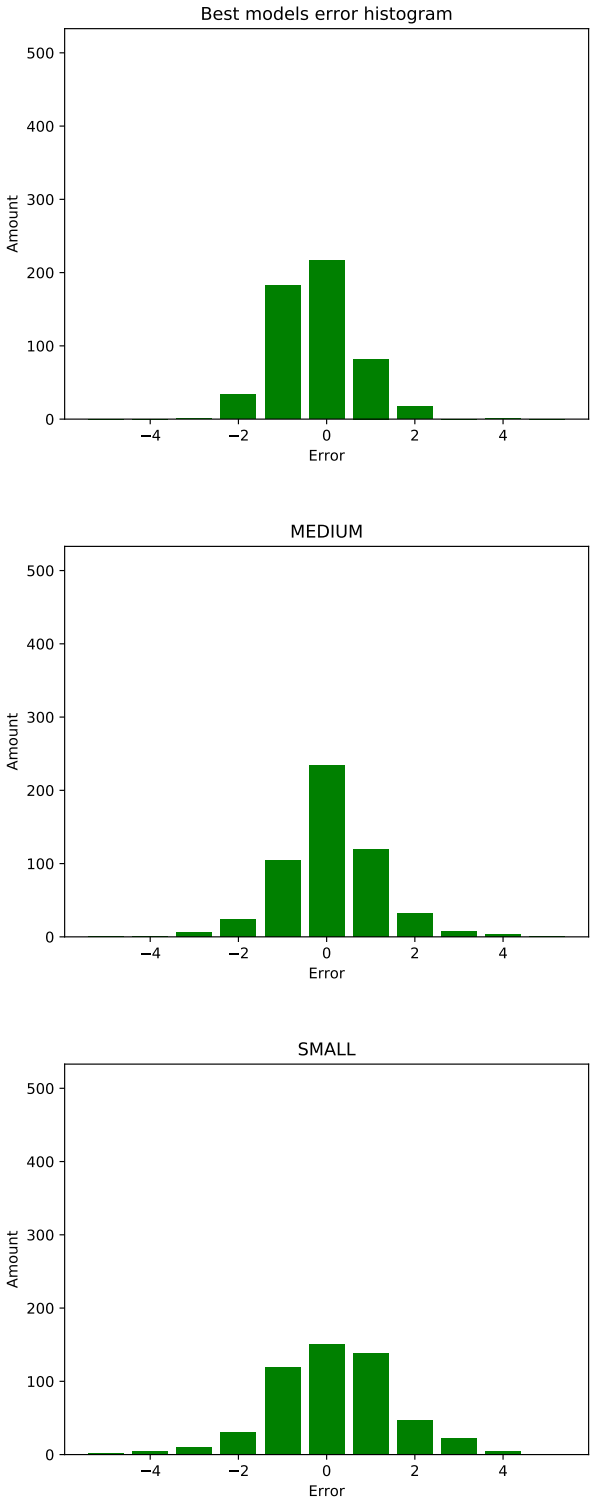


Figure 5.11 – Histograms of the magnitude of mispredictions. The three images show the best results that were achieved with EfficientNet, MEDIUM and SMALL models respectively from the top down. Note that the top image is visualizing the confusion matrix in Figure 5.10.

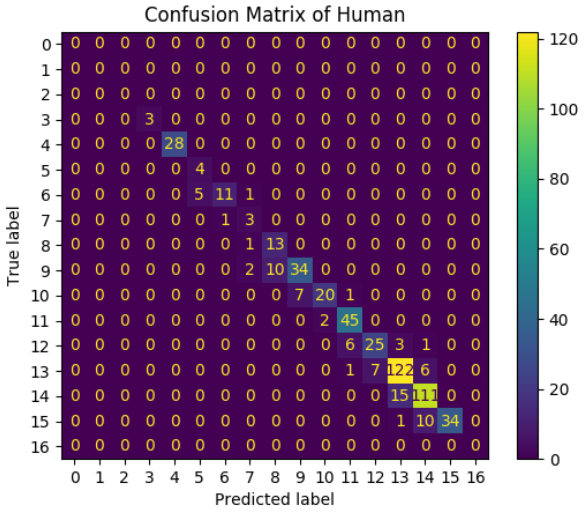


Figure 5.12 – The confusion matrix for the human's performance on the test set.

was very close to touching the line that had to be estimated to go across the ruler since the lines did not cover the whole width of the ruler.

To demonstrate just how difficult it was to interpret the current Robel poles, a human assistant was set on predicting the whole test set so it could be compared to the CNNs. The human got the rules clearly explained and examples on how to count the lines from the training set was shown. The exact outcome of the humans predictions, relative to the original labels, is shown in a confusion matrix shown in Figure 5.12. It can from this figure be observed that the human actually misclassified 80 of the 533 test images, resulting in a 15 % misclassification, of which all were misclassified with one unit but 5 of them which were misclassified with two units. This strengthens the justification of that a MAE of less than 1 is acceptable.

To compare the performance of the human being to the models, the MSE and MAE for the human's predictions were calculated. The MSE was 0.18 and the MAE was 0.16. A histogram of the difference between the human's predictions and labels are shown in Figure 5.13

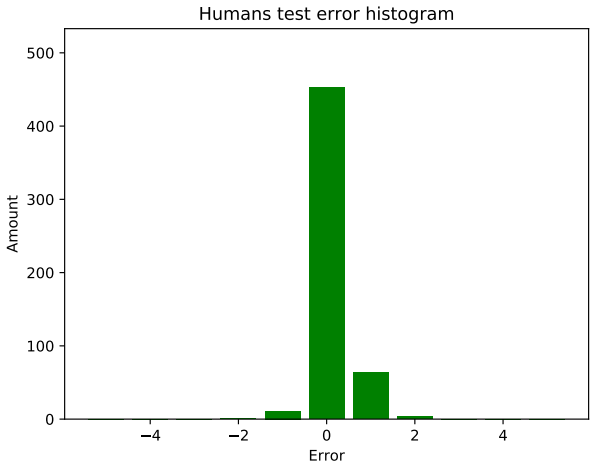


Figure 5.13 – Histogram of the magnitude of mispredictions for the human, visualizing Figure 5.12.

5.5 Comparing CNN's Performance to Human Performance

The human's result on the test set is unquestionably better than the best performing models. When looking at the MAE and MSE, the best model performs about 4.5 times worse. However, considering that the labels were so noisy that even a human missed 15% makes it impressive that learning took place to such a degree at all. We can focus on that the model only misclassified two labels in a worse way than the human, out of the 533 images. Since the team at Svegro stated that the height estimations still would be of good value to them even if they were within two centimeters of the true value, this means that the model only truly failed on 2/533 = 0.37%.

Chapter 6

Conclusions

6.1 Conclusions

The goal of this project was to investigate if and how well AI can be used to estimate the height of basil plants at Svegros plantation. After two months of image data collection from the surveillance cameras over the plants, 3373 separate images of individual Robel poles behind plants were obtained where the lines on the Robel pole above the highest point of the plant could be counted. It was early noticed that the Robel pole design, currently used, was hard to use to label the images properly and would therefore benefit from a redesign, proposed in Section 5.2.3.

From preliminary tests using Grad-CAM it was concluded that it was the basil plants leafs and structure that the CNNs correlated to the labels and not the visible lines on the Robel pole themselves.

The images were both resized to the smallest dimensions in the data set and also padded with a black border to the biggest dimensions in the data set. Three different CNN models were trained and results show that the padded data set did not give any significant gain in performance but took significantly longer time to train with. The tests show that the deeper networks performed the best if allowed to train for many epochs. The best performance was achieved with a model called EfficientNet0B which gave a MAE of 0.74 (where 2.53 would be just a median guess). With this, the goal of a MAE less than 1 was achieved and hence the conclusion is that AI can be used to measure height of basil's with the current set up, allowing for further automation of the production, paving the road for more energy efficient production and less wasted produce.

To get even better results, it has been suggested to change the design of the Robel pole and make bigger QR Codes with thicker paper so they can be more

easily read, thereby generating more usable data.

A test on image segments that did not include the Robel poles themselves was conducted by extracting image segments from under the Robel pole. The labels from the plants above these segments were taken, but since the height of plants can vary with about two centimeters although next to each other on the same row, the labeling now got considerably noisy. The results from this data was a MAE of 1.39 and MSE of 3.22, meaning it works relatively well but the high MSE indicates that it is not as precise as desired. It should however be investigated further, see Section 6.2.2.

6.2 Future Work

The dream scenario is that the only work required is to place the Robel poles behind the plants during a data collection period of about two months, and then the rest is fixed completely autonomously. By allowing a program to autonomously annotate and by improving the under data set, this can be achieved.

6.2.1 Eliminate Dependency of Robel Pole

Predicting plants heights without the Robel pole should be further researched as it would prove very useful for the team at Svegro since it would save them from the time consuming task of having to put out the Robel poles. This was tested with the under data set which gave encouraging results but did not exceed the expectations yet. Analyzing the results and the received information that plants can vary in height very much (although next to each other in the same row) gives the insight that taking 300x200 sized image segments from a bit under the Robel pole might not have been optimal. It is proposed for future research to test if better accuracy can be achieved by taking smaller crops from immediately under the Robel pole so that only the one plant that the Robel pole actually measures is what gets cropped out. This could potentially give as good results as a MAE of 0.2 as the labels then would suit the input perfectly.

6.2.2 Automate Annotation

Another proposal falls within the area of computer vision and should be conducted with the objective to automate the labeling of images of Robel poles. With a better Robel pole that is more suitable for image analysis, the visible lines on the Robel pole could automatically be counted. A small test for

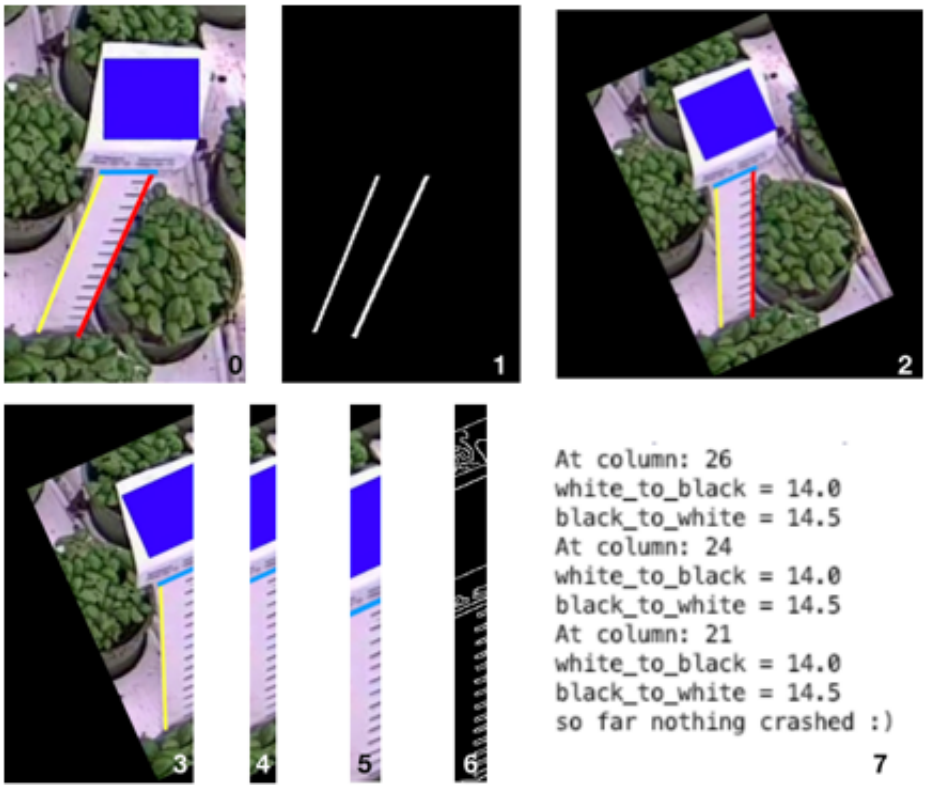


Figure 6.1 – Suggestion and demonstration of how to achieve automatic annotation for future implementations.

this has been conducted showing promising results. The suggestion is to add distinct boundaries to the Robel pole and to follow the recommended design suggested in Section 5.2.3. As the corona pandemic stopped any field tests, it had to be performed by editing the images digitally. The step by step process and results can be seen in Figure 6.1.

References

[1] Federico Apelt et al. “Phytotyping4D: a light-field imaging system for non-invasive and accurate monitoring of spatio-temporal plant growth.” In: *The Plant Journal* 82.4 (Apr. 2015), pp. 693–706. DOI: 10.1111/tpj.12833. URL: https://doi.org/10.1111/tpj.12833.

[2] James Bergstra et al. “Algorithms for Hyper-Parameter Optimization.” In: *Advances in Neural Information Processing Systems*. Ed. by J. Shawe-Taylor et al. Vol. 24. Curran Associates, Inc., 2011, pp. 2546–2554. URL: https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf.

[3] Gytis Bernotas et al. “A photometric stereo-based 3D imaging system using computer vision and deep learning for tracking plant growth.” In: *GigaScience* 8.5 (May 2019). giz056. ISSN: 2047-217X. DOI: 10.1093/gigascience/giz056. eprint: https://academic.oup.com/gigascience/article-pdf/8/5/giz056/31952187/giz056.pdf. URL: https://doi.org/10.1093/gigascience/giz056.

[4] Daryl Chang. *Effect of Batch Size on Neural Net Training*. May 2020. URL: https://medium.com/deep-learning-experiments/effect-of-batch-size-on-neural-net-training-c5ae8516e57.

[5] Mohamed Chetoui. *Gradient-weighted Class Activation Mapping - Grad-CAM-*. Mar. 2019. URL: https://medium.com/@mohamedchetoui/grad-cam-gradient-weighted-class-activation-mapping-ffd72742243a.

[6] Antonio Criminisi, Luc Van Gool, and Simon Bramble. “A New Approach to Obtain Height Measurements from Video.” In: *Proc SPIE* 3576 (Jan. 2002). DOI: 10.1117/12.334540.

[7] Adit Deshpande. *A Beginner's Guide To Understanding Convolutional Neural Networks*. Apr. 2019. URL: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/.

[8] Adit Deshpande. *A Beginner's Guide To Understanding Convolutional Neural Networks Part 2*. Apr. 2019. URL: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/.

[9] Kevin K. Dobbin and Richard M. Simon. "Optimally splitting cases for training and testing high dimensional classifiers." In: *BMC Medical Genomics* 4.1 (Apr. 2011), p. 31. ISSN: 1755-8794. DOI: 10.1186/1755-8794-4-31. URL: https://doi.org/10.1186/1755-8794-4-31.

[10] Tino Dornbusch et al. "Measuring the diurnal pattern of leaf hyponasty and growth in Arabidopsis - a novel phenotyping approach using laser scanning." In: *Functional Plant Biology* 39.11 (2012), p. 860. DOI: 10.1071/fp12018. URL: https://doi.org/10.1071/fp12018.

[11] Michael J. Garbade. *Regression Versus Classification Machine Learning: What's the Difference?* Aug. 2018. URL: https://medium.com/quick-code/regression-versus-classification-machine-learning-whats-the-difference-345c56dd15f7.

[12] Google. *Splitting Your Data*. URL: https://developers.google.com/machine-learning/data-prep/construct/sampling-splitting/split.

[13] Erik Gregersen. *QR Code*. 2020. URL: https://www.britannica.com/technology/QR-Code.

[14] Tushar Gupta. *Deep Learning: Feedforward Neural Network*. Jan. 2017. URL: https://towardsdatascience.com/deep-learning-feedforward-neural-network-26a6705dbdc7.

[15] Roderick Hunt. *Basic Growth Analysis: Plant growth analysis for beginners*. 1990. URL: https://tinyurl.com/y9tm6eos.

[16] Mitchell C. Hunter et al. "Agriculture in 2050: Recalibrating Targets for Sustainable Intensification." In: *BioScience* 67.4 (Feb. 2017), pp. 386–391. ISSN: 0006-3568. DOI: 10.1093/biosci/bix010. eprint: https://academic.oup.com/bioscience/article-pdf/67/4/386/13719545/bix010.pdf. URL: https://doi.org/10.1093/biosci/bix010.

[17] Mikael Huss. *The journey toward precision farming*. Dec. 2020. URL: https://peltarion.com/customer-stories/svegro.

[18] Masayoshi Mano. "Precise and continuous measurement of plant heights in an agricultural field using a time-lapse camera." In: *Journal of Agricultural Meteorology* 73 (June 2017). DOI: 10.2480/agrmet.D-16-00021.

[19] V. Le Mingxing Tan; Quoc. *EfficientNet: Improving Accuracy and Efficiency through AutoML and Model Scaling*. May 2019. URL: https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html.

[20] Justyna Jadwiga Olas, Franziska Fichtner, and Federico Apelt. "All roads lead to growth: imaging-based and biochemical methods to measure plant growth." In: *Journal of Experimental Botany* 71.1 (Sept. 2019), pp. 11–21. ISSN: 0022-0957. DOI: 10.1093/jxb/erz406. eprint: https://academic.oup.com/jxb/article-pdf/71/1/11/31549965/erz406.pdf. URL: https://doi.org/10.1093/jxb/erz406.

[21] Abhishek Panigrahi, Abhishek Shetty, and Navin Goyal. "Effect of Activation Functions on the Training of Overparametrized Neural Nets." In: *CoRR* abs/1908.05660 (2019). arXiv: 1908.05660. URL: http://arxiv.org/abs/1908.05660.

[22] Sebastian Ruder. "An overview of gradient descent optimization algorithms." In: *CoRR* abs/1609.04747 (2016). arXiv: 1609.04747. URL: http://arxiv.org/abs/1609.04747.

[23] Rutger Ruizendaal. *Deep Learning #3: More on CNNs & Handling Overfitting*. May 2017. URL: https://towardsdatascience.com/deep-learning-3-more-on-cnns-handling-overfitting-2bd5d99abe5d.

[24] Shaeke Salman and Xiuwen Liu. "Overfitting Mechanism and Avoidance in Deep Neural Networks." In: *CoRR* abs/1901.06566 (2019). arXiv: 1901.06566. URL: http://arxiv.org/abs/1901.06566.

[25] Mark Sandler et al. "Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation." In: *CoRR* abs/1801.04381 (2018). arXiv: 1801.04381. URL: http://arxiv.org/abs/1801.04381.

[26] Ramprasaath R. Selvaraju et al. “Grad-CAM: Why did you say that? Visual Explanations from Deep Networks via Gradient-based Localization.” In: *CoRR* abs/1610.02391 (2016). arXiv: 1610.02391. URL: http://arxiv.org/abs/1610.02391.

[27] Connor Shorten and Taghi M. Khoshgoftaar. “A survey on Image Data Augmentation for Deep Learning.” In: *Journal of Big Data* 6.1 (July 2019), p. 60. ISSN: 2196-1115. DOI: 10.1186/s40537-019-0197-0. URL: https://doi.org/10.1186/s40537-019-0197-0.

[28] Connor Shorten and Taghi M. Khoshgoftaar. “A survey on Image Data Augmentation for Deep Learning.” In: *Journal of Big Data* 6.1 (July 2019), p. 60. ISSN: 2196-1115. DOI: 10.1186/s40537-019-0197-0. URL: https://doi.org/10.1186/s40537-019-0197-0.

[29] Michael A. Smith. “Robel pole technique and data interpretation.” In: (June 2008). URL: http://wildlifehabitat.tamu.edu/Lessons/vegetation-Measurement/Readings/Smith-2008.pdf.

[30] Mingxing Tan and Quoc V. Le. “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks.” In: *CoRR* abs/1905.11946 (2019). arXiv: 1905.11946. URL: http://arxiv.org/abs/1905.11946.

[31] Sandra Vieira, Walter Pinaya, and Andrea Mechelli. “Using deep learning to investigate the neuroimaging correlates of psychiatric and neurological disorders: Methods and applications.” In: *Neuroscience & Biobehavioral Reviews* 74 (Jan. 2017). DOI: 10.1016/j.neubiorev.2017.01.002.

[32] VINNOVA. *AI-baserad optimering av växthusodling*. Nov. 2020. URL: https://www.vinnova.se/p/ai-baserad-optimering-av-vaxthusodling/.

[33] Xu Wang et al. “Field-based high-throughput phenotyping of plant height in sorghum using different sensing technologies.” In: *Plant Methods* 14 (July 2018). DOI: 10.1186/s13007-018-0324-5.

[34] Yuxin Wu et al. *Detectron2*. https://github.com/facebookresearch/detectron2. 2019.

[35] Liang Zheng et al. “Good Practice in CNN Feature Transfer.” In: (Apr. 2016).

Appendix A

Models Results

A.1 SMALL Results

Table A.1 – SMALL model results for different separate runs.

Data set	Aug	# epochs	Sec per epoch	Train MSE	Val MSE	Test MSE	Test MAE
Resized	No	20	41	0.87	1.75	2.68	1.27
Resized	No	24	42	0.88	2.22	3.61	1.55
Resized	No	18	42	1.23	2.06	2.56	**1.25**
Resized	Yes	44	41	2.24	1.61	1.95	**1.01**
Resized	Yes	59	41	2.20	1.64	1.94	1.08
Resized	Yes	59	41	1.89	1.78	2.06	1.12
Padded	No	21	162	2.71	2.48	2.97	**1.37**
Padded	No	22	163	1.29	2.95	3.96	1.63
Padded	No	33	163	0.69	2.25	3.58	1.46
Padded	Yes	53	167	2.14	1.67	1.71	1.01
Padded	Yes	56	167	2.55	1.86	1.64	**1.00**
Padded	Yes	22	167	3.79	2.63	2.27	1.12
Under	No	25	112	1.04	2.15	4.08	**1.58**
Under	No	27	114	0.74	2.20	4.06	1.59
Under	No	21	114	1.35	2.69	4.51	1.65
Under	Yes	37	115	3.87	3.16	4.06	1.61
Under	Yes	20	115	5.24	3.58	3.83	**1.51**
Under	Yes	30	115	3.86	3.62	4.06	1.59

A.2 MEDIUM Results

Table A.2 – MEDIUM model results for different separate runs.

Data set	Aug	# epochs	Sec per epoch	Train MSE	Val MSE	Test MSE	Test MAE
Resized	Yes	44	35	2.20	1.23	1.15	**0.79**
Resized	Yes	43	35	2.14	1.04	1.41	0.92
Resized	Yes	45	35	2.17	1.13	1.52	0.93
Padded	Yes	76	146	2.42	1.56	1.74	1.00
Padded	Yes	76	146	2.47	1.50	1.55	**0.95**
Padded	Yes	57	146	3.25	2.14	1.80	1.06
Under	Yes	50	102	3.10	2.14	3.22	**1.39**
Under	Yes	56	102	4.40	3.34	3.57	1.40
Under	Yes	17	101	6.11	4.56	4.62	1.73

A.3 EfficientNet Results

Table A.3 – EfficientNet model results for different separate runs.

Data set	Aug	# epochs	Sec per epoch	Train MSE	Val MSE	Test MSE	Test MAE
Resized	Yes	58	71	3.79	1.97	2.09	1.15
Resized	Yes	138	68	1.27	0.89	1.11	**0.82**
Resized	Yes	105	69	1.80	1.17	1.65	1.01

TRITA-EECS-EX-2021:81