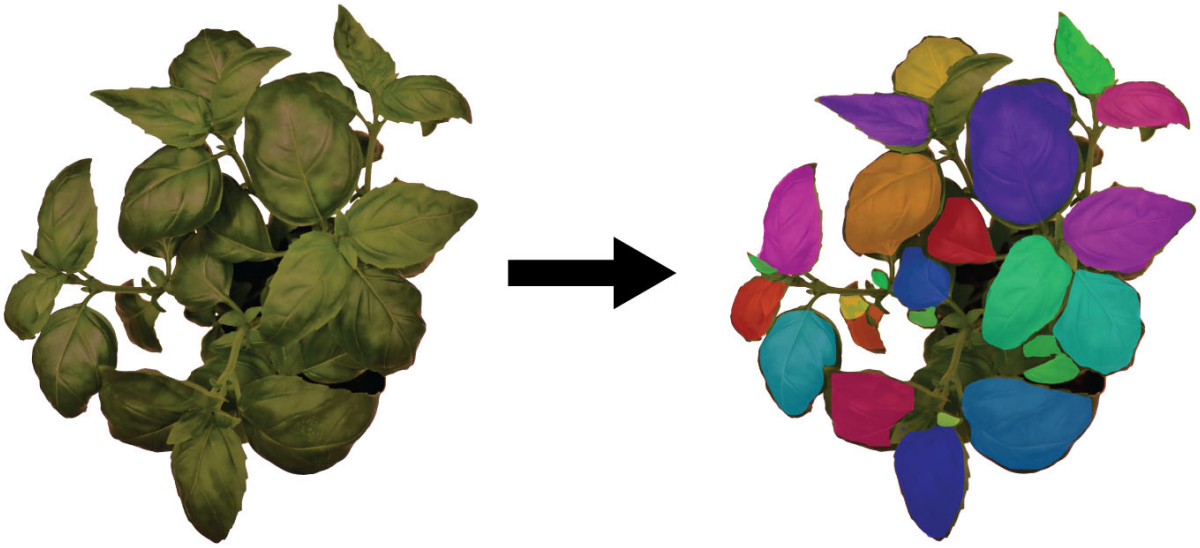
CHALMERS
UNIVERSITY OF TECHNOLOGY



Use of image processing to determine greenhouse crop control parameters

A thesis within image analysis and neural networks

Master's thesis in Biomedical Engineering

HELENA PETTERSSON

Department of Electrical Engineering

Master's thesis 2018:EENX30

Use of image processing to determine greenhouse crop control parameters

A thesis within image analysis and neural networks

HELENA PETTERSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Research and Development, Heliospectra AB
Chalmers University of Technology
Gothenburg, Sweden 2019

Use of image processing to determine greenhouse crop control parameters
A thesis within image analysis and neural networks
HELENA PETTERSSON

Supervisor: Daniel Bånkestad, Heliospectra AB
Supervisor: Torsten Wik, Automatic Control, Chalmers
Examiner: Fredrik Kahl, Computer Vision, Chalmers

Master's Thesis 2018:EENX30
Department of Electrical Engineering
Research and Development, Heliospectra AB
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Instance segmentation of a basil plant performed using neural networks.

Typeset in LaTeX
Gothenburg, Sweden 2019

Abstract

This study is conducted in collaboration with Heliospectra AB, a developer of highly controllable LED-lamps. Both the intensity and light spectrum can be controlled in the LED-lamps, giving uprise to intelligent lighting strategies which in the end will lead to large energy savings. Taking the development further and exploring the possibilities of optimizing the control system using image analysis and thereby making it feasible to have a feedback signal from the crops.

To investigate what potentially useful information that can be derived from the crops by image analysis, with particular focus on image segmentation, the aim of this project is to develop a small platform for image analysis in a plant lab. The platform shall include a database consisting of datasets with annotated images of basil plants and a program which shall be able to segment leaves in an aerial view of the plants.

By using image segmentation, transfer learning, COCO dataset, and the neural network framework Mask R-CNN, the program was able to perform leaf instance segmentation on aerial images of basil plants with an IoU_{mask} accuracy of up to 85%. Future work could be focused on developing new features for the platform, such as a monitoring program that aids the grower to keep track of the growth process of the plants. Another feature may be an alert system that detects disease and vermin in the plants. By focusing more on the development of the database, future work may instead be directed towards implementing more species of plants into the platform, making the platform broader in another sense.

Keywords: Image analysis, object detection, instance segmentation, image dataset, transfer learning, Mask R-CNN, annotating, neural networks.

Sammanfattning

Detta arbete är genomfört i samarbete med Heliospectra AB, som utvecklar kontrollerbara LED-lampor för växter. Både intensitet och spektrum kan kontrolleras på en hög nivå, vilket ger upphov till användandet av intelligenta ljus strategier som i slutändan kan leda till stora energibesparingar. För att ta utvecklingen vidare och optimera kontrollsystemet mer, kan man introducera bildanalys för att på så sätt få en återkopplande signal från plantorna.

För att undersöka vilken information man kan få ut av plantorna med hjälp av bildanalys, speciellt med fokus på instanssegmentering, är målet med det här projektet att ta fram en liten plattform för bildanalys i ett växtlabb. Plattformen ska innehålla en bilddatabas som består av ett dataset med uppmärkta och kategoriserade bilder av basilikaplantor, samt ett program som ska kunna segmentera blad i bilder tagna ovanifrån av plantorna.

Genom att använda instanssegmentering, överföringsinlärning, MS COCO dataset, och det neutrala nätverks-ramverket Mask R-CNN, möjliggjorde detta att programmet kunde utföra instans segmentering, av basilikablad på bilder tagna ovan plantorna, med en IoU_{mask} noggrannhet på 85%. Fortsatt utveckling av arbetet kan vara att implementera ett övervakningsprogram som kan hjälpa odlarna med att hålla koll på plantornas växtmönster. En annan funktion skulle istället kunna vara att utveckla ett varningssystem som kan upptäcka sjukdomar och skador på plantorna. Vill man utveckla plattformen i en annan riktning, kan man istället fokusera på att få in andra typer av växter än basilika i databasen, så att plattformen får ett bredare utbud åt ett annat håll.

Nyckelord: Bildanalys, objektdetektering, instanssegmentering, bilddatabas, överföringsinlärning, Mask R-CNN, annotering, neurala nätverk.

Acknowledgements

In order to accomplish results in this thesis a great help have been provided by the people at Heliospectra AB and an extra thanks to Daniel Bånkestad, Ida Fällström, Grazyna Bochenek and Johan Lindqvist.

Helena Pettersson, Gothenburg, June 2019

Nomenclature

CNN Convolutional neural network

FPN Feature pyramid network

IoU Intersection over union

$MaskR-CNN$ Mask region based-convolutional neural network

$ResNet$ Residual neural network

RNN Recurrent neural network

RoI Region of interest

RPN Region proposal network

Contents

List of Figures **xv**

List of Tables **xix**

1 Introduction **1**
1.1 Background . . . 1
1.2 Purpose and aim . . . 1
1.3 Objectives . . . 1

2 Theory and related work **3**
2.1 Plant CV and other related research . . . 4
2.2 Datasets . . . 4
2.3 Neural networks . . . 6
2.3.1 Training a neural network . . . 8
2.3.2 Transfer learning . . . 11
2.3.3 The Mask R-CNN framework . . . 13
2.3.4 Evaluation methods . . . 16

3 Methods **21**
3.1 Data acquisition setup . . . 21
3.1.1 Plants . . . 21
3.1.2 Camera setup . . . 22
3.2 Creating the datasets . . . 23
3.2.1 Augmenting images . . . 24
3.3 Training the neural network . . . 25
3.4 How the result shall be evaluated . . . 27

4 Results **29**
4.1 Data acquisition setup . . . 29
4.1.1 Plants . . . 29
4.1.2 Setup . . . 29
4.2 Annotating images and creating the datasets . . . 30
4.3 Network evaluation . . . 32
4.3.1 Training of the neural network . . . 32
4.3.2 Evaluating the network performance . . . 34

5 Discussion **39**

5.1 Data acquisition setup 39
5.1.1 Annotating images and creating the dataset 39
5.1.2 Augmenting images 41
5.2 The neural network 41
5.2.1 Training the neural network 42
5.2.2 Evaluating the network performance 42
5.3 Relation to medical field 42
5.4 Future work 42

6 Conclusion **45**

Bibliography **46**

A Measuring data during growth process **I**

B Datasets **V**

List of Figures

2.1 Illustrating how four different types of computer vision tasks have been solved on the same image. Image credit: (Lin et al. [2015]). . . . 3
2.2 A sample of images from the MNIST dataset with corresponding labels. Image credit: (*MNIST dataset introduction* [2017]). 4
2.3 A sample of two root-to-leaf branches of ImageNet, where 9 random images are presented for each synset. The top sample showing the mammal subtree and the bottom showing the vehicle subtree. Image credit: (Deng et al. [2009]). 5
2.4 Samples of annotated images in the MS COCO dataset. Image credit: Lin et al. [2015]. 6
2.5 Schematic of the human brain cell compared to an artificial neural network. 7
2.6 An example of how a (simplified) deep layer neural network for image classification might look like. The input image is feed into the trained neural network, and the neural network then predicts an output label, which in this case is "Face" i.e. the network predicted that the image contains a face. 7
2.7 Showing the RGB color profile. The intensity of each color channel is at its maximum to create white. Image credit: (Sullivan [2016]). . . 8
2.8 Showing the different planes that construct a color image and a grayscale image. In this case the grayscale image is created by taking the mean value of the RGB color image planes. Image credit: (Peyre [2010]). 8
2.9 **R.F.** Sigmoid function and the derivative of the sigmoid function. **L.F.** ReLU function and the derivative of the ReLU function. 10
2.10 **R.F.** Softmax function, with $K = 10$. **L.F.** Softmax function, with $K = 20$. 10
2.11 The dotted sigmoid function has been moved with the help of a bias. 11
2.12 A simplified deep neural network with hidden layers. How features can look like, lower level features in the earlier hidden layers, higher level features in the later hidden layers. The lower level features consist of rough shapes such as edges and corners, whilst higher level features consist of more real life objects such as rough face shapes. Image credit: (Aashay Sachdeva [2017]). 12
2.13 An illustration of the structure of the Mask R-CNN framework. Image credit: (Ren [2017]). 13

2.14 Different feature maps. Image (d) shows how the different features can transcend between the different layers increasing accuracy compared to (a)-(c). Image credit: (Lin et al. [2017]). 14
2.15 A RPN illustrating how the sliding-window generates anchor boxes. Image credit: (Ren et al. [2016]). 15
2.16 RPN anchor box proposals (more commonly called Region of Interests (RoIs)) showing the individual intersection over union (IoU) score at each box as well. Image credit: (Ren et al. [2016]). 15
2.17 Showing the ground truth bounding box and the predicted bounding box. Image credit: (Rosebrock [2016]). 18
2.18 **R.F**. The yellow field in the image represents the ground truth mask. **L.F.** The yellow field represents the predicted neural network output. Image credit: (Jordan [2018]). 19
2.19 **R.F.** The yellow field represents the intersection between the ground truth and the predicted output. **L.F.** The yellow field represents the union between the ground truth and the predicted output. Image credit: (Jordan [2018]). 19

3.1 The plant setup shown from the G3 camera. Each plant was numbered in order to follow its growth pattern. The setup was kept the same throughout the whole growth process. 21
3.2 The plant setup shown from the G3 Dome camera. 22
3.3 Illustrating the setup for obtaining the raw data. 22
3.4 Using the VIA image annotating tool to create a polygon train around each basil leaf. The polygon region shape in the VIA tool is used to create the polygon trains. 23
3.5 Leaves physically smaller than 2 cm are not considered to be leaves and therefore are not annotated. Too small leaves are denoted by the red circle. 23
3.6 Using the VIA tool to add object names to each instance in order to confine them to a category, the category being "basilleaf" in this case. Instances not confined to a category is automatically categorized as "BG" (background). 24
3.7 Illustrating how the augmentation was done. The left image is the input image with corresponding instances visualized by colored polygon trains. The right image is a rotated version of the input image, the instances have been rotated in the same manner as the image. . . 25

4.1 Images of the same basil plants, cropped and scaled down into the same sizes to show what the difference in camera placement does, as well as what role camera resolution have. The left and middle image have the same camera resolution, however the left camera is placed ~40cm above the middle camera. The right image have a camera resolution of roughly 11 times more than the other two cameras. . . 30

4.2 Showing the images in dataset #1, both the training set as well as the validation set. Images starting with a number was taken with the G3 Dome camera, other images were taken with the Canon 750D camera. Note that the annotations are not visible in the images. . . . 31
4.3 Validation loss and training loss for the network setups with 10 epochs. 33
4.4 Validation loss and training loss for the network setups with 30 epochs. 33
4.5 The input image, Basil-13, in the test dataset. 35
4.6 The input image, Basil-13, with different neural network setups trained on 10 epochs. 35
4.7 The input image, Basil-13, with different neural network setups trained on 30 epochs. 35
4.8 The input image, IMG-4000, in the test dataset. 36
4.9 The input image, IMG-4000, with different neural network setups trained on 10 epochs. 36
4.10 The input image, IMG-4000, with different neural network setups trained on 30 epochs. 36
4.11 The mask intersection over union (IoU_{mask}) results for the different network setups. The IoU_{mask} result was calculated over the whole test dataset. An average for each network setup was calculated, represented as the red line in the graph. 37
4.12 Leaf count for the different network setups. The leaf count result was calculated over the whole test dataset. An average for each network setup was calculated, represented as the red line in the graph. 37
4.13 Each mask is denoted by a different color to emphasize the different segmentations. The neural network segments two leaves with one mask (denoted by the red circle) in (a) or segments one leaf into several masks as in (b). 38
4.14 A neural network result performed on input image IMG-4000 and with network setup: 30e 160im. Each mask is denoted by a different color to emphasize the different segmentations. The neural network have segmented a leaf that is too small to be counted as a leaf (denoted by the red circle). This in turn effects the number of leaves counted. . 38

5.1 A stem from another leaf (denoted by the red circle) overlapping a leaf. The stem is included in the underlying leaf, in order for the underlying leaf to not be divided into two instances. 40
5.2 A large stem covering the underlying leaf (denoted by the red circle). A small portion of the stem is included into the underlying leaf in order for the leaf not to be divided into two instances. 40
5.3 A leaf overlapping another leaf. The underlying leaf, (denoted in red) had to be divided into two separate instances. 40
5.4 Hard for the eye to distinguish between two separate leaves (denoted by red circle). The underlying leaf is also very blurry, making it even harder to distinguish, thus the annotations are done approximately. . 41
5.5 Hard for the eye to distinguish between two separate leaves (denoted by red circle), thus the annotations are done approximately. 41

B.1 List of images in the different training datasets. The x marks if an image belongs to a dataset. VI
B.2 List of images in the different validation datasets at the top. List of images in the different testing datasets at the bottom. The x marks if an image belongs to a dataset. VII
B.3 Showing the images in dataset #2, both the training set as well as the validation set. Images starting with a number was taken with the G3 Dome camera, other images were taken with the Canon 750D camera. VIII
B.4 Showing some of the images in the training set of dataset #3. Images starting with a number was taken with the G3 Dome camera, other images were taken with the Canon 750D camera. IX
B.5 Showing the rest of the images in the training set of dataset #3. All images were taken with the Canon 750D camera. X
B.6 Showing the images in the validation set of dataset #3. Images starting with a number was taken with the G3 Dome camera, other images were taken with the Canon 750D camera. XI
B.7 Showing the test dataset, which is the same for all four datasets. All images were taken with the Canon 750D camera. XII

List of Tables

2.1 The different symbols used in the equations. Table credit: (Weng [2017]). 17

4.1 Camera resolutions for the different cameras. 29
4.2 The setup of the four different training datasets as well as for the test dataset, visualization of the datasets can be seen in appendix B. . . . 31
4.3 The different network setups during the training of the network. . . . 32

A.1 Result of the growth pattern for each basil. The first column represents the pot number for each basil plant, in order to follow each basil's growth pattern. The number in each column represents the number of leaves physically larger or equal to 2 cm. The yellow colored rows represents the pots containing only 3 plants per pot, every other pot contained 8 plants per pot. On harvest the height of the plant was measured as well, seen in the last column. II
A.2 Cont. Result of the growth pattern for each basil. The first column represents the pot number for each basil plant, in order to follow each basil's growth pattern. The number in each column represents the number of leaves physically larger or equal to 2 cm. The yellow colored rows represents the pots containing only 3 plants per pot, every other pot contained 8 plants per pot. On harvest the height of the plant was measured as well, seen in the last column. III

1
Introduction

This chapter presents the introduction to the master thesis work as well as stating the research questions which shall be answered throughout the thesis.

1.1 Background

Heliospecrta AB is a relatively young company working at the forefront developing highly controllable LED-lamps for both greenhouses and research purposes. Both the intensity and light spectrum can be controlled in the LED-lamps, giving uprise to intelligent lighting strategies which in the end will lead to large energy savings. In order to further investigate the possibilities of light control to be able to optimize the control system, more information about the crop is needed. A recent price drop for high performance cameras over the years has made it feasible to equip the LED-lamps with such a camera, creating an opportunity to have a feedback signal from the crops using image analysis.
The development in open source communities and technologies have made it possible to separate objects in an image using neural networks and image analysis frameworks such as the Mask R-CNN, (He et al. [2017]). Another one of these open source platforms is solely focusing on plants, namely the PlantCV, which can determine crop parameters with the help of a camera, (Gehan et al. [2017]). This framework however is not applicable for the over all horticulture community, since the user have to input specifications such as the number of pots, how the pots are placed and the framework is not able to find overlapping leaves in an image.

1.2 Purpose and aim

The purpose of this thesis is to investigate what potentially useful information can be derived by image analysis, in particular with focus on image segmentation.
The aim of this project is to develop a small platform for image analysis in a plant lab. The main program shall be able to segment leaves in an aerial view of the plants.

1.3 Objectives

The objective of this thesis is to investigate methods on how to incorporate image analysis into the horticulture environment in order to streamline the growth process.

The major question to keep in mind during the project is **"Can image analysis be used to determine growth parameters?"** and if so, **"What can we determine?"** with follow-up-questions such as **"Can it aid the grower for a more sufficient way of determining such parameters?"** and **"Is it sufficient enough in order to replace the conventional way of counting leaves by hand?"**.

2

Theory and related work

This chapter deals with theory used to solve the task in hand along with related work conducted around the world.

Programs using computer vision tasks can be divided into four main groups as following; **image classification**, **object localization**, **semantic segmentation**, and **instance segmentation**. If a program is using image classification it is able to tell that the image in figure 2.1a contains a person, a sheep and a dog but can not tell where in the image these objects are placed. To determine where in the image an object is located, the program have to use object localization instead. With this, the program can represent the location of an object as a bounding-box around that object, as shown in figure 2.1b. In the case of semantic segmentation, shown in figure 2.1c, the program is also able to, for each pixel in the image, determine if that pixel either belongs to the background, the person, the dog, or any of the five sheep. The program can not account for the fact that there are five sheep in the image and sees them as the same object. If the program should be able to separate the sheep and actually see them as five different objects, instance segmentation has to be applied. The separation of each sheep is represented by the different colored sheep in figure 2.1d.

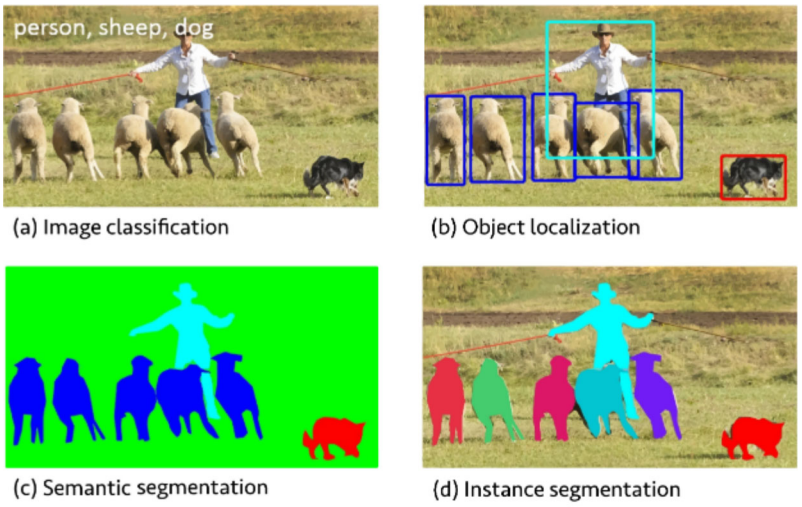


Figure 2.1: Illustrating how four different types of computer vision tasks have been solved on the same image. Image credit: (Lin et al. [2015]).

2.1 Plant CV and other related research

The research within the field of image analysis directed to the horticulture community is limited and therefore the availability of these applications are as well. There are however some research and software available for usage, most of them are focused on height determination of the plants and decease detection in plants (Lin et al. [2012]), (Radha [2017]), (Scharstein and Szeliski [2001]), (Grenzdörffer [2014]). One publicly available framework includes both of the aforementioned methods as well as a method for counting and segmenting leaves, namely the PlantCV framework, (Gehan et al. [2017]). This framework is on the cutting edge of image processing within the horticulture environment. Despite that, there are some restraints when using the leaf count functionality, it cannot distinguish overlapping leaves and the user have to specify how many plants are visible in the image as well as how the plants are organized (e.g. 3 plants in a row and 4 rows in total). The requirement of user input of how the plants are placed does not make the PlantCV very applicable for the overall user, since the plants may be placed differently compared to a perfect matrix as well as the problem with overlapping leaves.

2.2 Datasets

In order for a neural network to work it requires data, more specifically a dataset. The dataset connected to a neural network is task specific and determines the application. For visual tasks the dataset contains images, along with an associated file containing some (or all) of the following attributes; image classifications, object labels and annotations for each image in the dataset.

A dataset consists of 3 subgroups; training dataset, validation dataset and test dataset. As a rule of thumb when dividing the data into the different subgroups, a somewhat altered Pareto Principle is applied, i.e. the training dataset should contain ~80% of the total amount of data and the validation dataset should contain ~20%, (Box and Meyer [1986]). However since we have a third dataset an altercation is needed and the distributions is instead as following; ~60% training dataset, ~20% validation dataset and ~20% test dataset, (Andrew Ng [2019]).

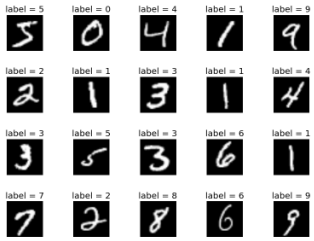


Figure 2.2: A sample of images from the MNIST dataset with corresponding labels. Image credit: (*MNIST dataset introduction* [2017]).

Obtaining a large dataset is the basis of a good neural network performance. However obtaining a large scale dataset is very time consuming since each image needs to be hand labeled, meaning a person needs to decide which classes are present in an image and going deeper into segmentation tasks, each pixel in an image needs to be associated with a class. In order to not reinvent the wheel for each visual task and to aid in research, there are some large scale datasets made publicly available for

download. Some common datasets are MNIST, MS COCO, ImageNet and PASCAL VOC.

The Mixed National Institute of Standards and Technology[1] (MNIST) dataset is to machine learning what "Hello World" is to programming i.e. it is the first dataset one encounters when starting to learn neural networks. The MNIST dataset consists of handwritten digits (see figure 2.2) and contains 60,000 training images and 10,000 test images and became publicly available in 1998, (LeCun Yann [1998]).

The Pattern Analysis Statistical modelling and Computational Learning Visual Object Classes (PASCAL VOC) dataset is a dataset with focus on object detection in natural images.
The dataset roughly contains 11,000 images with 20 object categories (classes), 27,000 object bounding-boxes and 7000 labeled instances (segmentations), (Everingham et al. [2010]). The dataset was discontinued in 2012 and has since been surpassed by the ImageNet and MS COCO datasets.

The ImageNet dataset focuses on the ability to capture a large amount of object categories. This is done in a WordNet manner, i.e. a hierarchical structure based on synonym sets (synsets) whom connects the root-to-leaf branches in a subtree, see figure 2.3.
When released in 2009 ImageNet had 3,2 million images with 5247 object categories sorted into 12 subtrees such as mammal and vehicle, (Deng et al. [2009]). As of now the dataset provides bounding boxes but no instances.

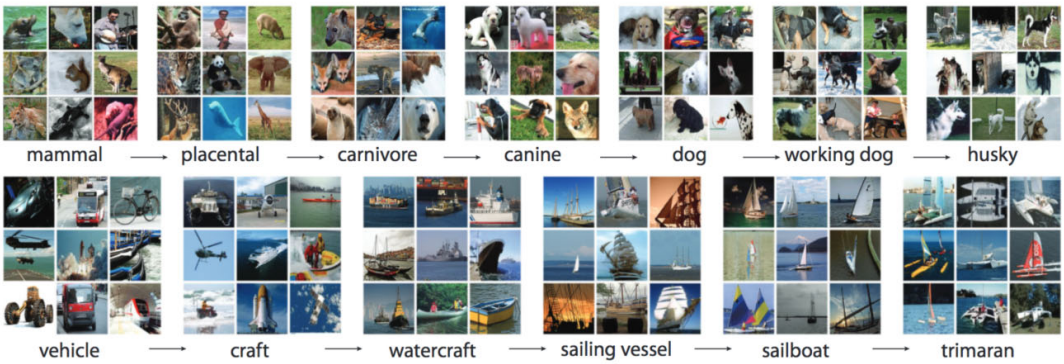


Figure 2.3: A sample of two root-to-leaf branches of ImageNet, where 9 random images are presented for each synset. The top sample showing the mammal subtree and the bottom showing the vehicle subtree. Image credit: (Deng et al. [2009]).

Microsoft Common Objects in COntext (MS COCO) is a dataset based on common items people encounters on a daily basis and would be easily recognizable by a 4 year old. MS COCO is designed for detection and segmentation of these objects occurring in their natural context, see figure 2.4.

[1]The National Institute of Standards and Technology (NIST) is a physical sciences laboratory, and a non-regulatory agency of the United States Department of Commerce.

When released in 2015 the MS COCO dataset contained 328,000 images with 91 object categories and 2,8 million labeled instances, (Lin et al. [2015]).

Compiling the vast amount of annotated images is, as stated earlier, very time consuming and both MS COCO and ImageNet have obtained this data by utilizing Amazon's Mechanical Turk (AMT)[2]. In total the MS COCO took ~70,000 working hours to complete.

Figure 2.4: Samples of annotated images in the MS COCO dataset. Image credit: Lin et al. [2015].

Some other programs that are more specifically designed to aid in creating datasets for image visualization tasks are LabelMe, Labelbox, RectLabel and VGG Image Annotator (VIA) tool. Both LabelMe and Labelbox are free of charge in exchange for letting them keep the data created with their programs. RectLabel is free of charge and does not ask to keep the data created, however it is specific to Mac OS and cannot be used on Windows. The VIA tool is free of charge and runs on a web-platform so does not require a specific operating system and the program does not keep the data created.

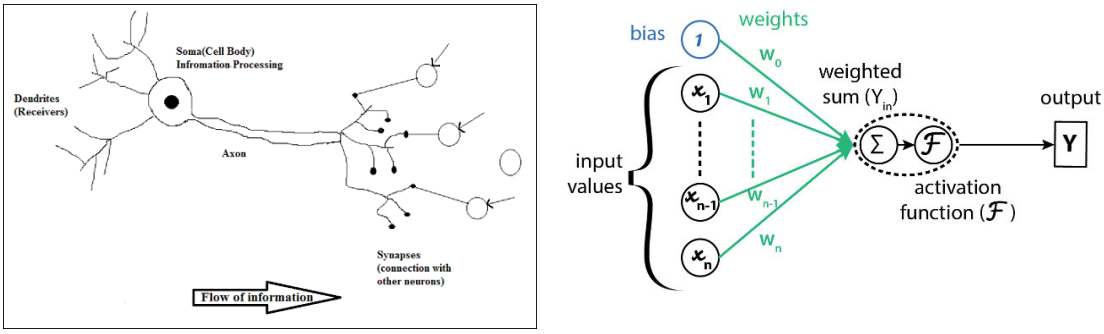
2.3 Neural networks

The basic structure of an artificial neural network (called neural network for simplicity) mimics, as good as it can, the human brain cells and their connections. The equivalent to a brain cell and its connections in the neural network is called a neuron. In many cases it goes by another name, perceptron, and acts as a binary linear classifier. When using multiple layer of perceptrons a perceptron can perform either classification or regression, depending upon its activation function, (SHARMA [2017]), (*Multilayer perceptron — Wikipedia, The Free Encyclopedia* [2019]).

The direction of information flow in a human brain can be seen in figure 2.5a. The dendrites connected to the brain cell receives an input signal, this signal is passed via the axons who in turn tells the synapses to send out signals to another cells' dendrites. A neural network needs to operate in a similar manner, and the different parts in the human brain can be translated to the neural network. The dendrites translates to the input in the neural network, the synapses are the weights and the axon translates to the output, see figure 2.5 for clarifications.

To translate the interconnections that happens when synapses interact with (activates) the dendrites in the human brain we also need a type of activation function in the neural network. Common activation functions used are the sigmoid function, the softmax function and the rectified linear unit (ReLU) function (all of which are described in detail in section, 2.3.1) (Garbade [2018]), (*Artificial Neural Network - Basic Concepts* [2019]), (He et al. [2017]), (Hinton Geoffrey E [2013]).

[2]Amazon Mechanical Turk is a online platform that enables individuals and businesses to outsource their processes and jobs for users to complete and get paid for.



(a) A schematic of a human brain cell and its connections. Image credit: (*Artificial Neural Network - Basic Concepts* [2019]).

(b) A schematic of a perceptron for a artificial neural network. The dotted circle represents a hidden neuron.

Figure 2.5: Schematic of the human brain cell compared to an artificial neural network.

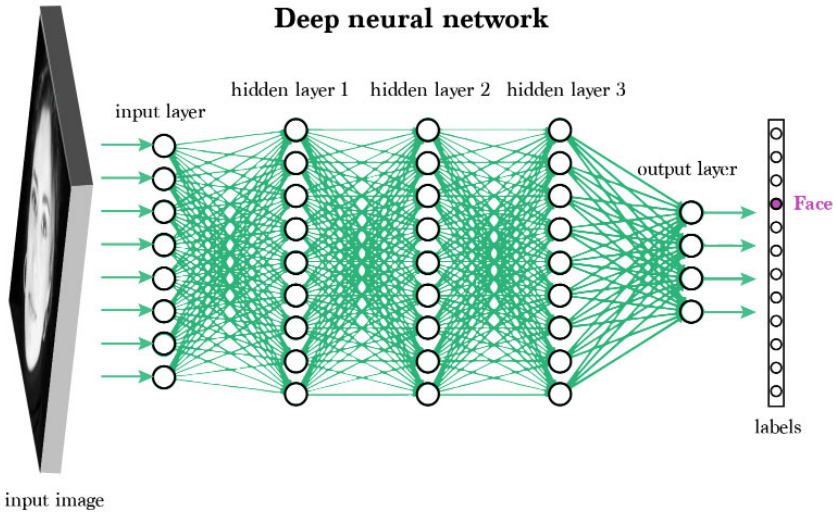


Figure 2.6: An example of how a (simplified) deep layer neural network for image classification might look like. The input image is feed into the trained neural network, and the neural network then predicts an output label, which in this case is "Face" i.e. the network predicted that the image contains a face.

A neural network almost always consists of more than one perceptron which lay side by side with each other to construct a hidden layer, see figure 2.6. This figure is however slightly simplified, since each circle contains the weighted sum and the activation function. The bias is not illustrated at all but exists for every perceptron. The weights are in fact also constructed of matrices of float numbers between 0 and 1 and are therefore often refereed to as weight matrices. The output from each perceptron acts as the input to the next layer of hidden layers. The last layer decides the predicted output from the entire network and decides which label (class) the input most likely belongs to.

In figure 2.6 it is also illustrated that the input to the network is an image, which is feed into the network as a matrix containing all of the pixel-values for the image, e.g for a HD color image the matrix has the dimensions: 1024x1024x3. The first numbers are the height and width of the image, where each entry in the matrix corresponds to a pixel in the image. The number 3 comes from the fact that it is a color image and the most common color profile used in computer vision is RGB (R=red, B=blue and G=green, shown in figure 2.3), so in order for each color channel to be represented in the matrix, 3 planes is needed, see figure 2.8a for an example. A grayscale image consists of only one plane, that represents the intensity between black and white. A grayscale can be created from a color image by taking the mean value of the three planes for each individual pixel and thereby creating a new matrix, see figure 2.8b for an example.
The initial value of the pixels are normalized so that they have a value between 0 and 1. This yields for both the RGB image as well as the grayscale image. If the values of a color pixel (x,y) is (0,0,0) the pixel is black if it however is (1,1,1) it is white.

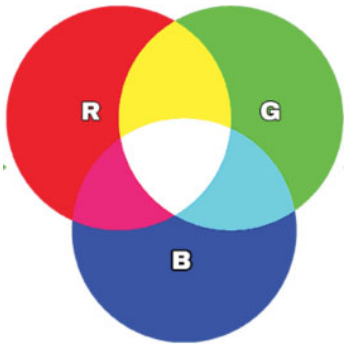
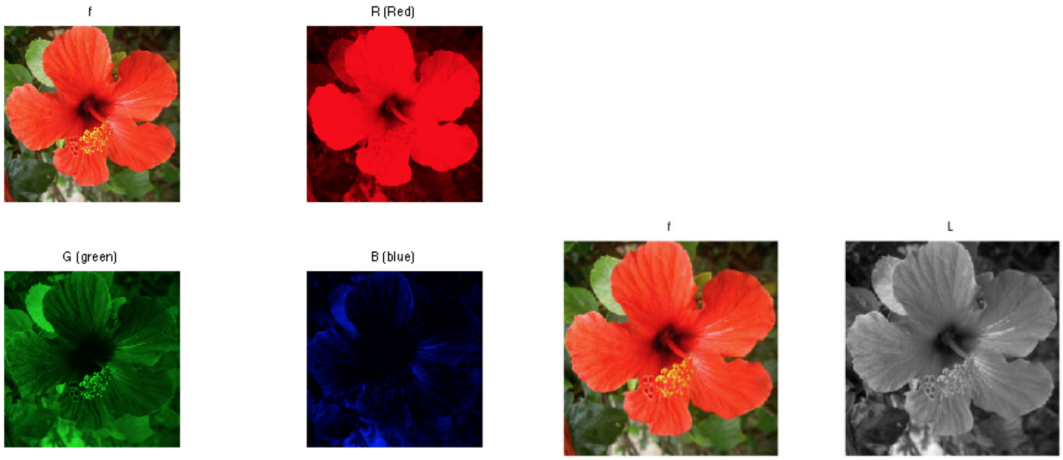


Figure 2.7: Showing the RGB color profile. The intensity of each color channel is at its maximum to create white. Image credit: (Sullivan [2016]).



(a) Showing the planes of a color image.

(b) Showing the plane of a grayscale image.

Figure 2.8: Showing the different planes that construct a color image and a grayscale image. In this case the grayscale image is created by taking the mean value of the RGB color image planes. Image credit: (Peyre [2010]).

2.3.1 Training a neural network

The goal of a neural network is to predict the output and come as close to the real output as possible. To obtain this, training is needed, and during the training period the weight matrices are changed (updated) for each loop of training.

As described in the previous section the output of one layer acts as the input to the next layer with the help of weight matrices and activation functions. So in order to calculate the predicted output of the neural network the output from each individual layer needs to be calculated.
Looking at figure 2.5b we see that the predicted output of the network is $Y = \mathcal{F}(y_{in})$. We also have the following input to the activation function (also known as the weighted sum):

$$y_{in} = \sum_{i=0}^{n} x_i w_i \ .$$

Keeping in mind that a neural network is looking more like in figure 2.6, it is clear that we have multiple of these equations and that the predicted output Y actually is the predicted output for just one perceptron.
To complete the calculations for one perceptron however, equations for the activation function need to be obtained.
The three activation functions of interest are, as previously mentioned, the sigmoid function, the ReLU function and the softmax function. The functions are described mathematically below and the characteristics of each function can be found in figures 2.9 and 2.10.
The sigmoid function and its corresponding derivative are

$$Y_{sigmoid} = \mathcal{F}(y_{in}) = \frac{1}{1 + e^{-y_{in}}} \ , \ Y'_{sigmoid} = \mathcal{F}'(y_{in}) = \mathcal{F}(y_{in})(1 - \mathcal{F}(y_{in})) \ .$$

The ReLU function and its corresponding derivative are

$$Y_{ReLU} = \mathcal{F}(y_{in}) = \begin{cases} 0 & \text{for } y_{in} < 0 \\ y_{in} & \text{for } y_{in} \geq 0 \end{cases} \ , \ Y'_{ReLU} = \mathcal{F}'(y_{in}) = \begin{cases} 0 & \text{for } y_{in} < 0 \\ 1 & \text{for } y_{in} \geq 0 \end{cases} \ .$$

The softmax function is

$$Y_{softmax} = \mathcal{F}(y_{in}) = \frac{e^{y_{in}}}{\sum_{k=1}^{K} e^{y_{in_k}}} \ ,$$

where K is the number of total output classes. A representation of the softmax function can be seen in figure 2.10.

When training a neural network a common way is to feed the inputs through the network in a straight forward manner, called forward propagation, as well as doing so the weights are usually initialized randomly with float numbers between 0 and 1. The predicted output produced by the neural network, called Y in figure 2.5b, is compared to the true output, Y_{true}, (prelabeled by the user) as following: $error = Y_{true} - Y$. The $error$ determines how the weights of the hidden layer shall be updated, and the derivative of the activation function determines in which direction and how much the weight shall be updated. This method is called backpropagation, since it is done starting from the output and working its way through all the layers to the input.

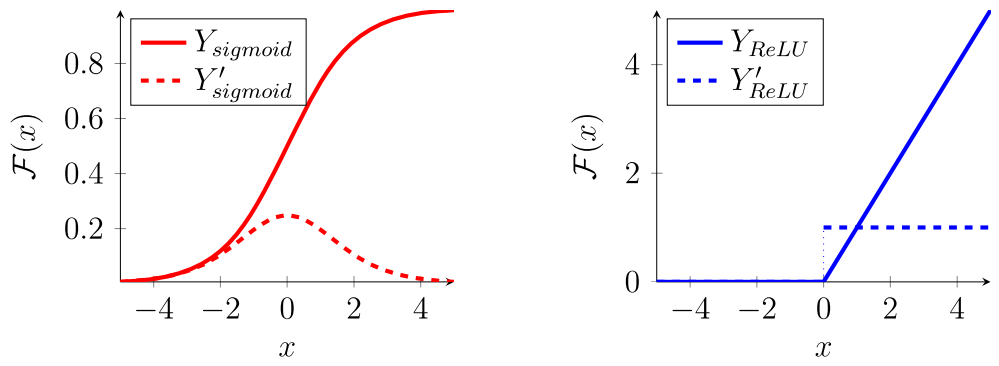


Figure 2.9: R.F. Sigmoid function and the derivative of the sigmoid function. **L.F.** ReLU function and the derivative of the ReLU function.

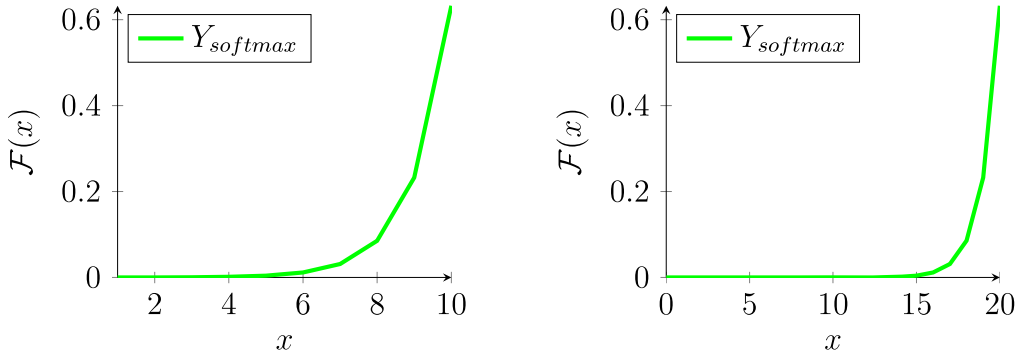


Figure 2.10: R.F. Softmax function, with $K = 10$. **L.F.** Softmax function, with $K = 20$.

In the case of backpropagation the derivative of the sigmoid activation function, seen in figure 2.9, will only amplify the error by a quarter for each hidden layer. This means that the hidden layers closest to the input may not be effected by larger errors generated from the output since a lot of information is lost with each step.

Looking at the derivative of the ReLU function in the same figure (2.9), one notice that the derivative is equal to 1 for all $x \geq 0$ which helps to avoid the effect of loosing information during backpropagation.
The sigmoid function is a binary function, which means it can only differentiate between either this or that, whilst the softmax function is a categorical probability distribution which means that it can determine the probability between multiple classes and tell you if any of them are true.
The ReLU function is easier to use compared to the sigmoid function and aids in faster training time for large networks, (Hinton Geoffrey E [2013]).
Depending on the application the activation functions can be used separately or in combination with each other. One of the most common way for a multiple classification problem is to have the ReLU activation function in the hidden layers and to use the softmax function in the output of the neural network.

The contribution of the bias at each neuron makes it possible to make the activation function more flexible in order for it to classify the data correctly, e.g. the activation function can be moved so that the function not always intersect origin, see figure 2.11.

There are several different structures of neural networks, some of the most common ones are Recurrent Neural Network (RNN), Convolutional Neural Networks (CNN), and Region Proposal Network (RPN). They all are very similar, built up of different layers which are connected with weight matrices. However there are some differences, which aids in different applications: a RNN is more applicable when doing computational tasks regarding speech and writing, wheres a CNN and a RPN is more suited for images and videos. In fact, the RPN was constructed in order to find region of interests in images which is more described in detail later in this paper. The different networks can be combined for suitable applications together.

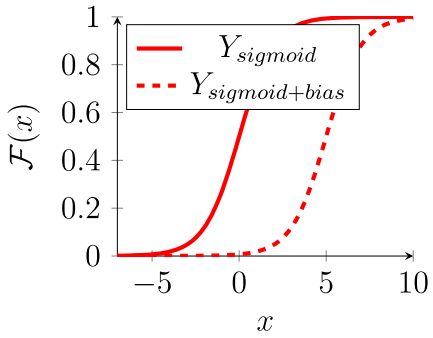


Figure 2.11: The dotted sigmoid function has been moved with the help of a bias.

The deepness, shallowness, thickness and narrowness of the neural network is of importance and there are some studies that has shown that a shallow network can fit any function if it is really thick. A thick network will however cause a large increase in neurons in each hidden layer and in turn a large increase in calculation size between the hidden layers. A deep but narrow neural network has also been proven to fit any function, despite having more hidden layers (with fewer neurons per layer) it has been proven to outperform shallower neural networks, (Lin et al. [2015]), (He et al. [2015]).

2.3.2 Transfer learning

Constructing a deep neural network and training it is very time consuming, so in order to get a good model without the large time consumption transfer learning is used. Early research showed that it is possible to transfer features from the lower levels of features, see figure 2.12, in a neural network and apply on a new task, (Caruana [1995]).

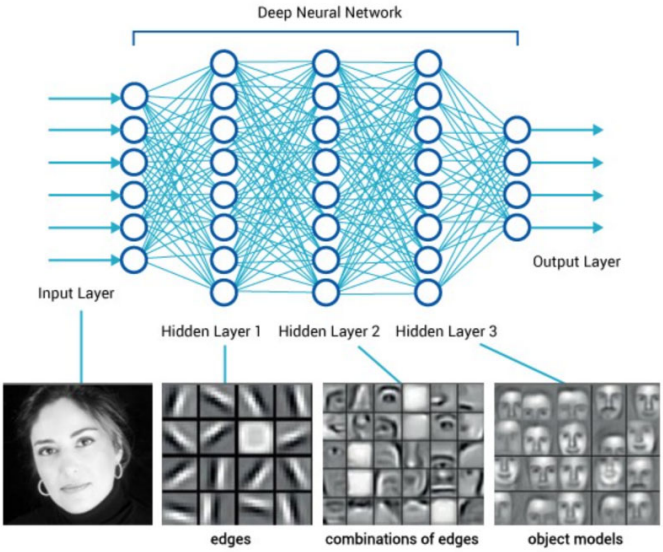


Figure 2.12: A simplified deep neural network with hidden layers. How features can look like, lower level features in the earlier hidden layers, higher level features in the later hidden layers. The lower level features consist of rough shapes such as edges and corners, whilst higher level features consist of more real life objects such as rough face shapes. Image credit: (Aashay Sachdeva [2017]).

More recent research has further shown that transfer learning can achieve superior results for some tasks compered to not using transfer learning, (Donahue et al. [2013]), (Yosinski et al. [2014]), (Razavian et al. [2014]). Moreover there are two major transfer learning alternatives for CNNs:

- Removing the last fully connected layer (the classifier) to retrain it on the new dataset. Basically meaning to have a CNN as a fixed feature extractor for the new dataset.
- Removing the classifier and retrain it as well as fine tuning and retraining the weights through backpropagation. Either all of the weights can be fine-tuned or only the weights associated with the layers of higher features.

Fine-tuning a CNN can be tricky depending on the size of the new dataset. So some rule of thumbs to keep in mind and help during the fine-tuning are:

- **If the new dataset is small and similar to the original dataset:** Fine-tuning can lead to overfitting. The best way is to train the linear classifier on the CNN.
- **If the new dataset is large and similar to the original dataset:** Fine-tuning all of the weights should be fine, and overfitting should be avoided.
- **If the new dataset is small and very different from the original dataset:** Training a linear classifier with start somewhere early in the network should aid in getting more data-specific features later in the neural network.
- **If the new dataset is large and very different from the original dataset:** Train a CNN from scratch, or fine-tune all of the weights.

Being similar to the original dataset means that they have a similar approach, e.g. the original dataset is ImageNet which focuses on many classes and natural scenery, the new dataset also have many classes in natural scenery. If the new dataset on the other hand contains few classes and for example are microscopic images it is considered to be very different from the original dataset (ImageNet in this example). Due to the effectiveness of transfer learning there are many open source frameworks available for implementation of instance segmentation tasks, such as Mask R-CNN, CFS-FCN, U-Net, and CUMedVision2 (DCAN).

2.3.3 The Mask R-CNN framework

The Mask Region based-Convolutional Neural Network (Mask R-CNN) framework is created for instance segmentation tasks and it uses the Faster R-CNN framework, used for object detection tasks, as a foundation, (He et al. [2017]). The Mask R-CNN framework is illustrated in figure 2.13.

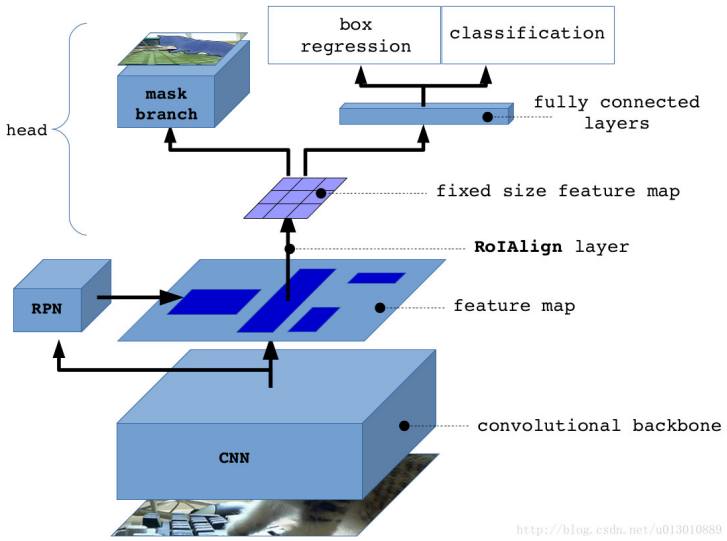


Figure 2.13: An illustration of the structure of the Mask R-CNN framework. Image credit: (Ren [2017]).

The Mask R-CNN framework can be divided into four blocks as follows:

- **Backbone:** A standard CNN of some sort.
- **Regional proposal network (RPN):** A lightweight CNN with sliding-window effect which creates regions of interests (RoIs).
- **RoI classifier and bounding box regressor:** A CNN which is cropping and resizing the RoIs.
- **Segmentation mask branch:** A CNN which converts RoIs into binary masks.

A more detailed description of the four blocks is given below.

Backbone

This block takes an image as input and produces a feature map which acts as an input to the next block.
There are many different network architectures that can be used as a backbone such as AlexNet, ZFNet, VGGNet, GoogLeNet, ResNet, ResNeXt, DenseNet, and SENet.

A suitable choice of CNN is a CNN with a significant depth and some type of ResNet, since it uses skip connections which enables a deeper network structure (up to 152 layers; ResNet152) with higher accuracy than other shallower networks, despite losing computational speed would be a suitable choice, (He et al. [2015]).

To improve the accuracy even more a Feature Pyramid Network (FPN) lays on top to extract features from more layers, see figure 2.14. The authors of the paper (Lin et al. [2017]) claims that in using a FPN they "achieves state-of-the-art single-model results on the COCO detection benchmark without bells and whistles, surpassing all existing single-model entries including those from the COCO 2016 challenge winners".

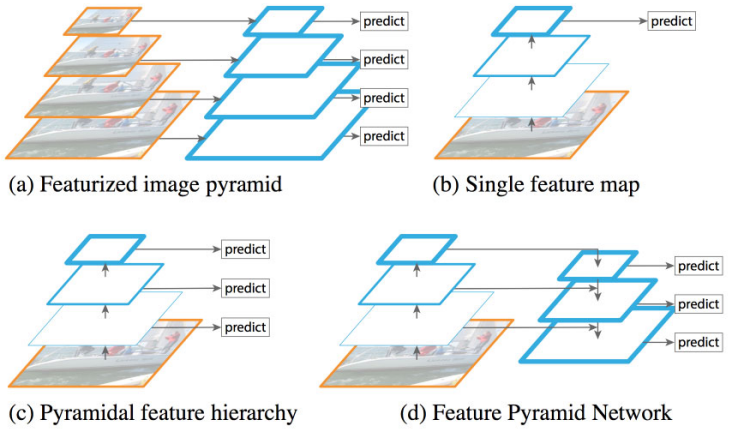


Figure 2.14: Different feature maps. Image (d) shows how the different features can transcend between the different layers increasing accuracy compared to (a)-(c). Image credit: (Lin et al. [2017]).

Regional proposal network (RPN)

Takes an feature map as input and produces anchor boxes (RoIs), containing center x-,y-coordinate of the box as well as the height and with of the box, as output.

A lightweight CNN is used to scan over the feature map (produced by previous block) like a sliding-window, which finds patches (areas) of interest in the feature map. In each sliding-window, multiple region proposals are predicted, called anchors, where k is the maximum number anchors as seen in 2.15. An anchor box is centered in the sliding window and is associated with a scale and aspect ratio.

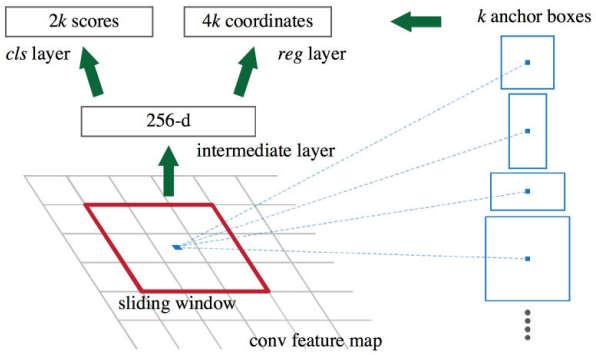


Figure 2.15: A RPN illustrating how the sliding-window generates anchor boxes. Image credit: (Ren et al. [2016]).

Each anchor is classified as positive (likely to contain an object) or negative (not likely to contain an object) using bounding box intersection over union (IoU) scores. A score higher than $IoU_{threshold}$ is positive, and if an ground-truth box has more than one positive anchor, the anchor with the highest score is kept (known as non-maximum suppression).

The positive anchors may not be placed perfectly over the object, so therefore they have to be refined, known as bounding box regression. This is done by shifting both the x-,y-coordinate of the center-point of the anchor box as well as adjusting the height and width of the anchor box. At last the refined anchor boxes finalizes in region of interests (RoI) proposals, (consisting of the final x-,y-coordinates and the width and height of the the box) illustrated in 2.16 with corresponding IoU score for each box.

Figure 2.16: RPN anchor box proposals (more commonly called Region of Interests (RoIs)) showing the individual intersection over union (IoU) score at each box as well. Image credit: (Ren et al. [2016]).

RoI classifier and bounding box regressor

Takes the bounding box proposals as inputs and classifies them into specific labels (given by the network input specifications).

This classification stage differs from the one in the previous stage in that this network is deeper and by that has more ability to classify more classes than just positive or negative, i.e. a softmax classifier is used.
Since the RoIs provided by the steps before most likely varies in size so they have to be cropped and resized in order for the classification to work. This is called RoI warping or bounding box regression and this layer is applied on the feature maps. The RoI warping layer has a pre-defined output size and can take any input size.

Segmentation mask branch

Takes the RoI from the classifier as an input and generates segmentation masks for them.
The segmentation mask block converts the RoIs into binary masks and by pixel-wise binary classification the CNN determines which pixel belongs to which category. If multiple classes are present, each class gets its own binary mask, so there is no competition between them. The generated masks consists of float numbers which makes it possible to be more precise compared to integer numbers.

2.3.4 Evaluation methods

There are many ways to evaluate a neural network and its performance, two of which will be presented here; loss functions and intersection over union.

Loss functions

One way to evaluate the performance of a neural network is to evaluate the loss-functions during the training period, (Changhau [2017]). Evaluating the loss function is done both on the training set as well as the validation set.
There is a variety of different loss functions such as Mean Squared Error (MSE), Mean Squared Logarithmic Error(MSLE), L2, Mean Absolute Error (MAE), Mean Absolute Error (MAE), Mean Absolute Precentage Error (MAPE), L1, Kullback Leibler (KL) Divergence, Cross Entropy, Negative Logarithmic Likelihood, Poisson, Cosine Proximity, Hinge, Squared Hinge. They all have different applications and specific areas of interest for the developer. The purpose of a loss function is however the same despite which function is selected, namely to minimize the loss of both the training loss and validation loss and to make them converge to one another.

The different loss functions in the Mask R-CNN:
The multitask loss function for the Mask R-CNN combines the losses from classification, bounding box regression and segmentation mask, and is defined as following in (He et al. [2017]):

$$\mathcal{L} = \mathcal{L}_{cls} + \mathcal{L}_{box} + \mathcal{L}_{mask} \tag{2.1}$$

in which the two losses $\mathcal{L}_{cls}$ and $\mathcal{L}_{box}$ where first defined in (Girshick [2015]).

Symbol	Description
u	True class label, $u \in$ 0,1,...,K; the background class has $u = 0$
p	Discrete probability distribution (per RoI) over K + 1 classes: $p = (p_0, p_1, ..., p_k)$, computed by a softmax over the $K + 1$ outputs of a fully connected layer
v	True bounding box, $v = (v_x, v_y, v_w, v_h)$
t^u	Predicted bounding box correction, $t^u = (t^u_x, t^u_y, t^u_w, t^u_h)$
p_i	Predicted probability of anchor i being an object
p^*_i	Binary ground truth label of whether anchor i is an object
t_i	The four parameterized predicted coordinates
t^*_i	Ground truth coordinates
N_{cls}	Normalization term, set to be mini-batch size
N_{box}	Normalization term, set to the number of anchor locations
λ	A balancing parameter, so that both $\mathcal{L}_{cls}$ and $\mathcal{L}_{box}$ terms are roughly equally weighted

Table 2.1: The different symbols used in the equations. Table credit: (Weng [2017]).

Further on these are given as following:

$$\mathbb{1}[u \geq 1] = \begin{cases} 1 & \text{if } u \geq 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\mathcal{L}(p, u, t^u, v) = \mathcal{L}_{cls}(p, u) + \mathbb{1}[u \geq 1]\mathcal{L}_{box}(t^u, v) \tag{2.2}$$

$$\mathcal{L}_{cls}(p, u) = -\log p_u \tag{2.3}$$

$$\mathcal{L}_{box}(t^u, v) = \sum_{i \in \{x,y,w,h\}} L_1^{smooth}(t^u_i - v_i) \tag{2.4}$$

$$L_1^{smooth}(x) = \begin{cases} 0.5x^2 & \text{if } |x| \geq 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

$$\mathcal{L}_{mask} = -\frac{1}{m^2} \sum_{1 \leq i,j \leq m} [y_{ij} \log y_{ij}^{\hat{k}} + (i - y_{ij}) \log(1 - y_{ij}^{\hat{k}})] \ . \tag{2.5}$$

The loss functions for the RPN is the following:

$$\mathcal{L}(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i \mathcal{L}_{cls}(p_i, p_i^*) + \frac{\lambda}{N_{box}} \sum_i p_i^* L_i^{smooth}(t_i - t_i^*) \tag{2.6}$$

$$\mathcal{L}_{cls}(p_i, p_i^*) = -p_i^* \log p_i - (1 - p_i^*) \log(1 - p_i) \ . \tag{2.7}$$

All of these five loss equations contribute to the total loss function for the whole training process of the network. The same calculations are used to calculate the validation loss functions, with the change that drop-out is not used, i.e. all neurons in the whole network is used to calculate the loss functions. Compared to when in training mode, and drop-out is used to drop random neurons in order to add some noise and avoid over-fitting.

Intersection over Union (IoU)

Another method to evaluate a neural networks performance is to use the Jaccard index, coined by Paul Jaccard in (Phytoiiogist [1912]), or as it is more commonly known today; the Intersection over Union (IoU) index. This index measures the similarity between finite sample sets, and is defined as

$$\mathbf{J}(\mathbf{A}, \mathbf{B}) = \frac{|A \cap B|}{|A \cup B|} = \mathbf{IoU} \ . \tag{2.8}$$

To implement this, the nominator in equation 2.8 consists of the intersection between the ground-truth bounding box and the predicted bounding box and the denominator is represented by the union between the same boxes. A representation of the boxes is shown in figure 2.17.

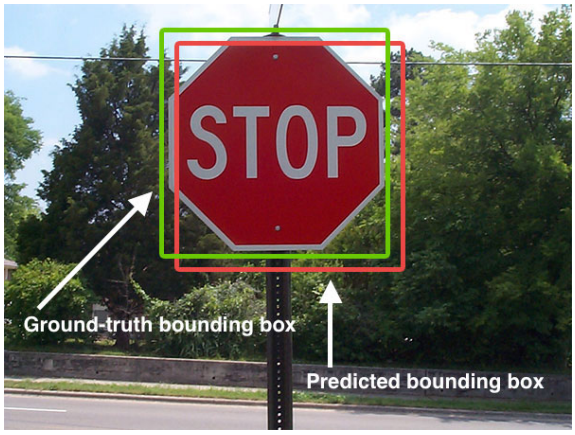


Figure 2.17: Showing the ground truth bounding box and the predicted bounding box. Image credit: (Rosebrock [2016]).

However to get an even more accurate IoU for segmentation tasks one can calculate the area, pixel-wise, of the segmentation masks instead of the bounding boxes. The equation is basically the same, however it calculates the area of a ground truth mask and a predicted output mask instead as:

$$\mathbf{IoU_{mask}} = \frac{\text{Intersection of ground truth mask and predicted mask}}{\text{Union of ground truth mask and predicted mask}} . \tag{2.9}$$

Visualization of the ground truth mask and the predicted output mask are shown as yellow fields in figure 2.18. And the intersection and union of the two masks are represented in figure 2.19.

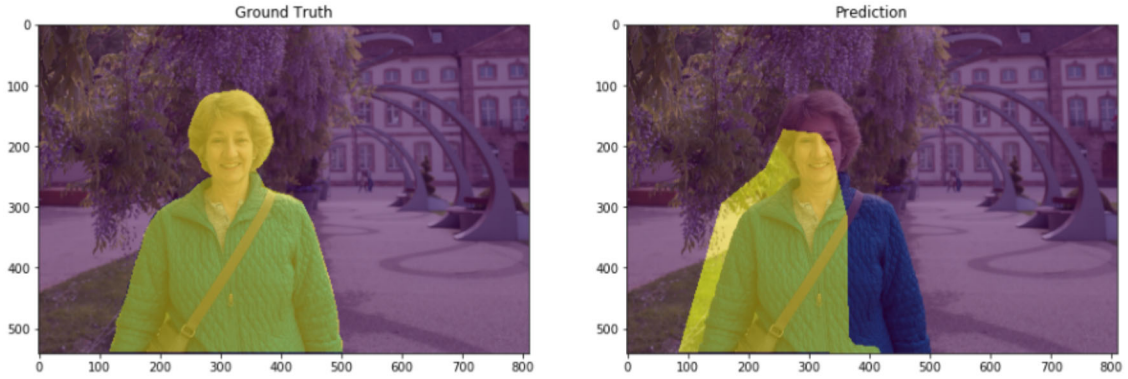


Figure 2.18: R.F. The yellow field in the image represents the ground truth mask. **L.F.** The yellow field represents the predicted neural network output. Image credit: (Jordan [2018]).

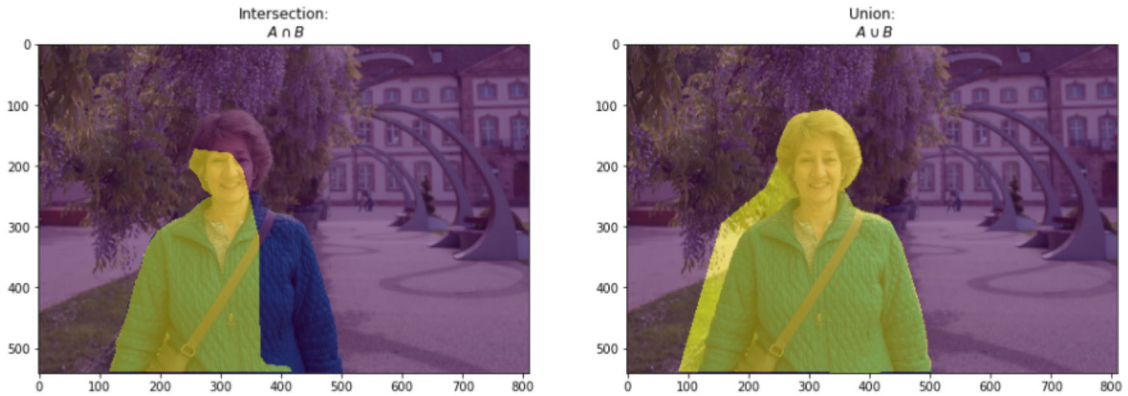


Figure 2.19: R.F. The yellow field represents the intersection between the ground truth and the predicted output. **L.F.** The yellow field represents the union between the ground truth and the predicted output. Image credit: (Jordan [2018]).

3
Methods

In this chapter, the methods used to solve the task in hand are presented.

3.1 Data acquisition setup

In order to get a data set images of the required object are needed and in order to get those images the object is necessary. In this case the dataset should be constructed for basil leaves and therefore basil plants are required in order to get the images needed.

3.1.1 Plants

The basil plants where positioned in a growth chamber and was monitored during the 28-day growth period. In order to get different data the basil plants where placed on 4 different trays positioned in different ways on each tray as seen in figure 3.1. Each pot had 8 plants per pot except for pot number 20, 21, 22, 49 and 50 whom had 3 plants per pot. During the growth period the plants received sufficient lighting and nutrients in order to grow. During each time of watering, the plants where also examined and all of the leaves of each plant where counted in order to follow the growth pattern, a leaf is counted as a leaf when the physical length of it is larger or equal to 2 cm and the result can be found in appendix A.1 and A.2. In order to aid in future work the height of the plants, during harvest, where also measured and can also be found in the same appendix.

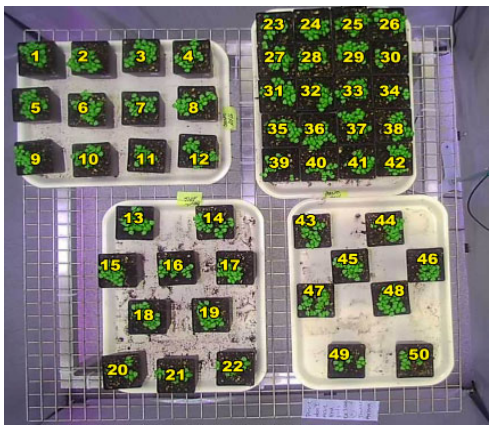
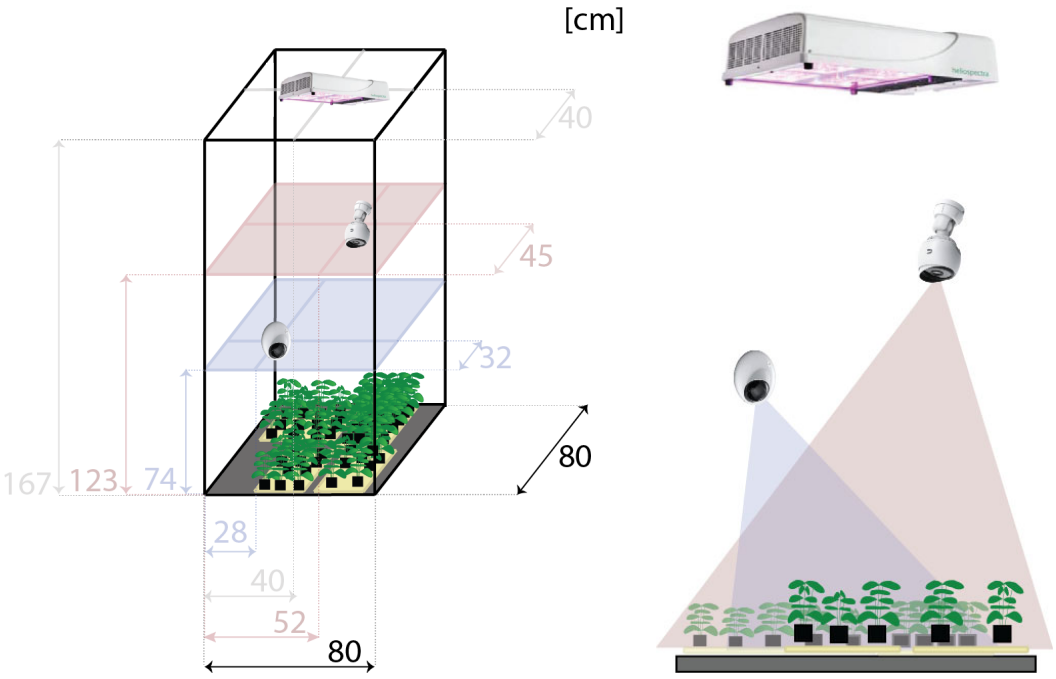


Figure 3.1: The plant setup shown from the G3 camera. Each plant was numbered in order to follow its growth pattern. The setup was kept the same throughout the whole growth process.

3.1.2 Camera setup

Two UniFi cameras was placed inside the unit at different heights as seen in figure 3.3a. The camera to the left in figure 3.3a is a UniFi Video Camera G3 Dome (G3 Dome) and the camera to the right in the same figure is a UniFi Video Camera G3 (G3). The field of view from the G3 Dome camera can be seen in figure 3.2.
The UniFi cameras are used for surveillance and are therefore easy to use and access via Wi-Fi. To obtain the raw data a python script was used in order to acquisition 1 image per camera per minute during the 28-day growth period.
The images was categorized and stored automatically in order of date and time of the capture. Both cameras produce images of the size 1920x1080px which equals to a pixel count of roughly 2.07 MP.

Figure 3.2: The plant setup shown from the G3 Dome camera.



(a) A 3D illustration of the set up inside the growth unit with measurements. At the top a Heliospectra lamp (grey layer), in the middle a UniFi Video Camera G3 Dome (red layer) and at the bottom a UniFi Video Camera G3 (blue layer).

(b) A side view illustration of the set up. The blue and red areas show the camera G3 Dome's and the camera G3's field of view respectively.

Figure 3.3: Illustrating the setup for obtaining the raw data.

3.2 Creating the datasets

To create the datasets, the images had to be manually annotated first. A suitable choice for an annotating tool was the VGG Image Annotator (VIA) tool[1], since it can be used online on any computer and does not keep the data created using the tool. As the goal is to perform instance segmentation the dataset needs to contain annotated images containing instance segmentation masks around the objects of interest, the basil leaves in this case. In every image a polygon train had to be created on the outline of each basil leaf such as in figure 3.4b, and this was done using the polygon region shape in the VIA tool shown in figure 3.4a. The polygon train around each leaf is called an instance and the more accurate the instance follows the shape of the leaf, the better the neural network will perform in the end.

(a) Polygon region shape.

(b) One polygon train created around one basil leaf.

Figure 3.4: Using the VIA image annotating tool to create a polygon train around each basil leaf. The polygon region shape in the VIA tool is used to create the polygon trains.

During the growth process leaves physically smaller than 2 cm were not considered a leaf and were not counted, because of this reasoning it is only logical that leaves physically smaller than 2 cm are not annotated in the images as well, see figure 3.5. Since it is images there is no ability to measure and therefore the number of annotations shall be compared to the real number of leaves, in order to be more precise.
When all of the leaves in an image have a corresponding instance, the instances (as well as the non-instance areas of the image) have to be categorized, i.e. the pixels in the image needs to be told which category they belong to.

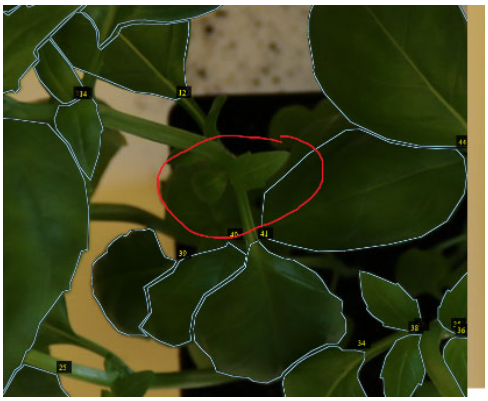


Figure 3.5: Leaves physically smaller than 2 cm are not considered to be leaves and therefore are not annotated. Too small leaves are denoted by the red circle.

[1]VGG Image Annotator (VIA) is an image annotation tool that can be used to define regions in an image and create textual descriptions of those regions. VIA is an open source project developed at the Visual Geometry Group and released under the BSD-2 clause license.

Since the neural network should be able to find a basil leaf a suitable name for the category is "basilleaf". In the VIA tool this is done by adding an object name to the instance, shown in figure 3.6, and this has to be done for each instance (each leaf) in the image.
When every instance have an object name (i.e. belongs to a category) the VIA tool generates a JSON-file, where each entry in the file corresponds to one instance and its attributes. Some of the attributes being the name of the image file, the x-,y-coordinates of each red-dot in the polygon trains (see figure 3.4b) and what category the pixels confined by the train belongs to. The file also contains information about all other areas whom are not included in the instances and these are assigned to the background "BG" category.

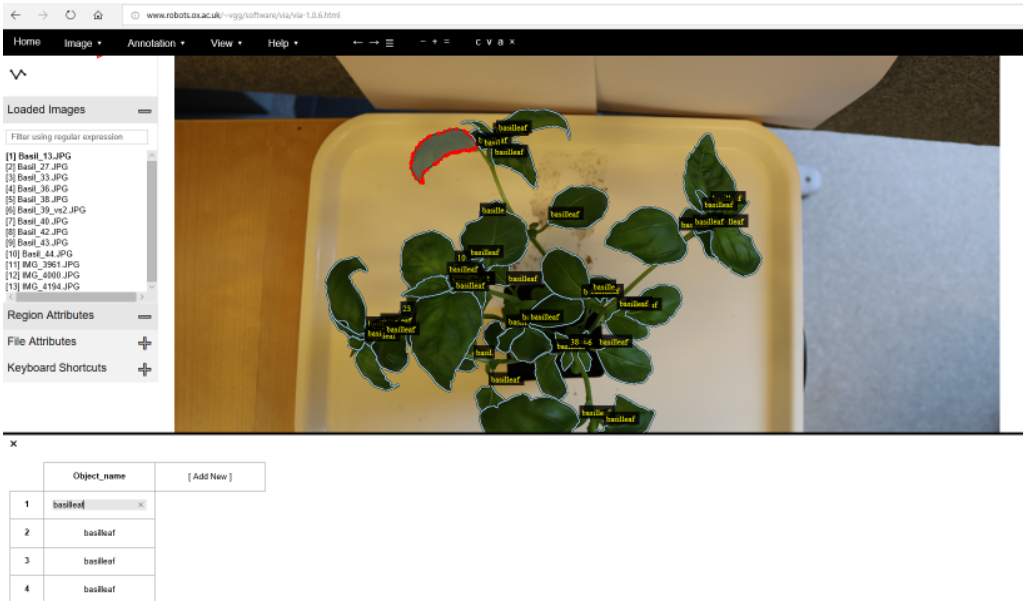


Figure 3.6: Using the VIA tool to add object names to each instance in order to confine them to a category, the category being "basilleaf" in this case. Instances not confined to a category is automatically categorized as "BG" (background).

A dataset should contain more than just one image and all images shall be annotated at the same time in order to create just one JSON-file containing the information for all of the images. The annotating procedure was repeated for all of the images to create the three datasets (training, validation and test) each containing images and a corresponding JSON-file. Also keeping in mind that the total amount of data should be distributed over the three datasets accordingly; ~60% training dataset, ~20% validation dataset and ~20% test dataset.

3.2.1 Augmenting images

Manual annotation of images is very time consuming demonstrably mentioned in section 2.2, so in order to overcome this issue, augmentation was used. Augmentation can be done by either rotating the image, flipping it, shearing it, scaling it or a combination of said methods.

The easiest method of augmentation is to rotate each image in the dataset a couple of times in order to gain a greater dataset. However the images cannot just be rotated, the instances needs to be rotated in the same manner as the image itself, see figure 3.7. This was simply done using a MATLAB script, which as an input takes an image with its corresponding JSON-file. The script in return produces an output image with the rotated version of the image, as well as adding new entries in the JSON-file containing the rotated instances.

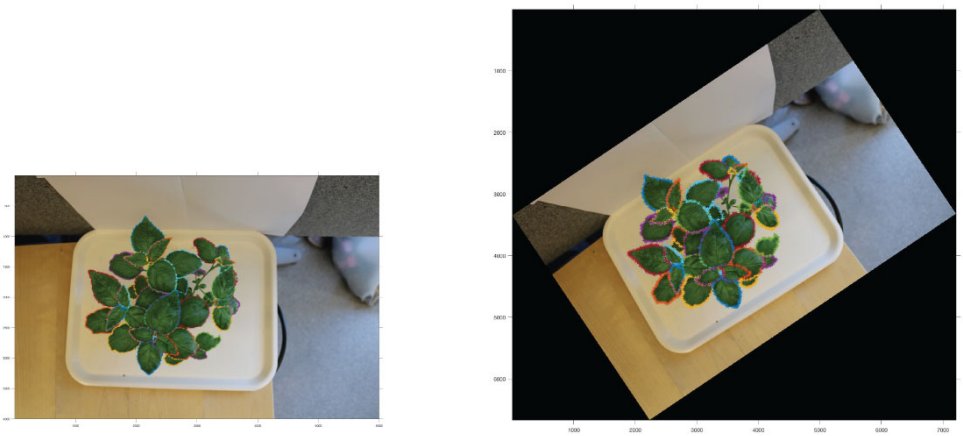


Figure 3.7: Illustrating how the augmentation was done. The left image is the input image with corresponding instances visualized by colored polygon trains. The right image is a rotated version of the input image, the instances have been rotated in the same manner as the image.

3.3 Training the neural network

Transfer learning was used in order to train the neural network. Since the task is to preform instance segmentation, a good starting point is with a neural network trained on the MS COCO dataset since the dataset has more labelled instance segmentations compared to ImageNet and MNIST. The Mask R-CNN framework was chosen as the framework to use, and further on an implementation of the framework made by the company Matterport[2], available on GitHub[3], was used to work with as they have done an implementation with the MS COCO dataset.
As described in section 2.3.2, the rule of thumbs shall be considered when applying transfer learning. In this case the new dataset is quite small and the original dataset is chosen to be MS COCO there are two choices left to consider: is the new dataset similar or very different. Since the new dataset consists of images taken of basil plants, there is a slight chance that there exist basil plants in the MS COCO dataset

[2]Matterport is a company developing and selling three-dimensional camera systems one can use to create realistic, fully immersive experiences, (Abdulla [2017]).

[3]GitHub is a development platform used for anything from open source to business. It has 36 million developers and people can host and review code, manage projects, and build software.

already, and therefore the new dataset shall be considered as *similar to the original dataset.*
And following the rule of thumbs the best way to train the neural network is to train just the linear classifier i.e the head branch in this case. This is done by freezing all of the weights not associated with the head branches and retrain the head branch on the new dataset.
Most of the network parameters in Matterport's implementation is the same as in the ones presented in the Mask R-CNN article however some varies due to compatibility issues towards TensorFlow and Keras and some needs to be changed to match the new dataset. The parameters of interest is mentioned below and the other parameters can be reviewed at Matterport's GitHub page, (Abdulla [2017]).

Network parameters

Backbone

The RestNet101 has been widely used since the introduction and it won 1st place in the ILSVRC[4] image classification, detection and localization competition as well as the MS COCO detection and segmentation competition 2015[5].
The FPN which lays on top to extract features from more layers has a size of 256 with the following strides: 4, 8, 16, 32 and 64, (Lin et al. [2017]).

Regional proposal network (RPN)

The anchor box has a scale and aspect ratio whom both by default is 3, which yields $k = 9$ at each sliding-window adding up to ~200'000 anchor boxes in total for a feature map with an input image size of 1024x1024xRGB.
The $IoU_{threshold}$ is set to 0.7 and an anchor is classified as positive if the score is higher, and if a ground-truth box has more than one positive anchor, the anchor with the highest score is kept.
The total number of RoIs per image is kept to a maximum of 2000 per image.

RoI classifier and bounding box regressor

The classifier only needs to decide between two different classes, "basilleaf" or "BG" in this case. Therefore it is not necessary to use the softmax function, the sigmoid function will suffice.
The function RoIAlign is not yet implemented in TensorFlow[6] and therefore, as the Matterport implementation suggests, the crop_and_resize function in TensorFlow is used as a substitute function.

[4]The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) evaluates algorithms for object detection and image classification at large scale, (*ImageNet Large Scale Visual Recognition Challenge* [2015]).

[5]Looking at the general problem of visual recognition, the underlying datasets and the specific tasks in the challenges probe different aspects of the problem,(*COCO - Common Objects in Context* [2015]), (*ImageNet and MS COCO Visual Recognition Challenges Joint Workshop 2015* [2015]).

[6]At the time of writing this report a beta version of RoIAlign has been released for TensorFlow.

Segmentation mask branch

The mask size is set to 28x28 and the minimum probability value of which to accept a detection is set to 90% in order to trying to get a high accuracy.
The implementation of Matterport also uses a maximum of 100 ground truth instances, which is quite small due to the fact that there are probably more than 100 leaves visible when having an image of 3 basil plants (on average one plant had ~43 leaves). So this number is set to 400 to make sure every ground truth area is kept.

3.4 How the result shall be evaluated

During the training of the neural network the program produces an output of how the training loss and validation loss is going. As mentioned in section 2.3.4 both of the loss functions are composed of five different losses that contribute to the total loss function. The progress of the loss functions shall be presented in a graph form in order to follow its development during the training process. The goal is to minimize the loss functions and see that both the training loss and validation loss finds the same steady-state value.
The IoU_{mask} result can only be calculated when the neural network is done training and ready to perform the segmentation task. The evaluation is performed on the test dataset in order to ensure a plausible outcome of the network performance.

4
Results

This chapter presents the results obtained throughout the work.

4.1 Data acquisition setup

4.1.1 Plants

Plants are living things and do not always behave as expected. During the growth process some of the basil plants dried out and were nearly dead for several days, so some of the images have nearly dead plants in them. This is also reflected in the real number off leaves counted since some leaves fell of the plants.

4.1.2 Setup

The python script used for obtaining the images was supposed to run from the start of the growth until the harvest day, however it crashed several times during, so instead of 40 320 images from each camera, approximately 23 160 images where obtained from each camera instead.

Camera	**PPI**	**Pixel dimension[WxH]**	**Pixel count[MPixel]**
G3	255	1920x1080	2.07
G3 Dome	255	1920x1080	2.07
Canon 750D	258	6000x4000	24.00

Table 4.1: Camera resolutions for the different cameras.

In table 4.1 it can be seen that both the G3 and the G3-Dome camera have the same PPI[1] as well as the same pixel dimension and pixel count resolution, however, since the G3 camera was placed ~40cm above we get a larger field of view which results in a blurrier image from it compared to the G3-Dome camera, as shown in figure 4.1. When the plants where ready for harvest, a Canon 750D camera was used to photograph each plant in close up (from about ~50 cm height) in order to get images with higher pixel count resolution. The height from were the images was obtained by the Canon camera varied quite a lot, since the camera unfortunately was held by hand, and not placed in a stationary position. To gain some diversity of

[1]PPI stands for pixels per inch, and describes how many pixels that can be placed in a square with sides of 1 inch (2.54 cm).

images, a couple of different constellations of plant images were also taken in close up as well, such as the ones presented in figure 4.1.

(a) Image from G3 camera.

(b) Image from G3 Dome camera.

(c) Image from Canon 750D camera.

Figure 4.1: Images of the same basil plants, cropped and scaled down into the same sizes to show what the difference in camera placement does, as well as what role camera resolution have. The left and middle image have the same camera resolution, however the left camera is placed ~40cm above the middle camera. The right image have a camera resolution of roughly 11 times more than the other two cameras.

4.2 Annotating images and creating the datasets

Four different training datasets where created, compiled of a different amount of images. The first dataset contains 7 images in the training set and 3 images in the validation set, visualized in figure 4.2. The same images are present in the other datasets, since the dataset grew over time as the annotation progressed. The second dataset contains of 9 images in the training set and 3 images in the validation set, visualized in appendix B.3. The third dataset contains 40 images in the training set and 10 in the validation set, visualized in appendix B.4, B.5 and B.6. To create the fourth dataset augmentation was used, and each image in the third dataset was rotated 3 times each, thereby creating 3 times more images for the last dataset, i.e. 160 images in the training set and 40 images in the validation set.

After creating the four training datasets a test dataset was also created, containing 10 images in total. These images are not present in the four training datasets, in order to get a plausible result when evaluating the performance later on.

Annotating one image took between 20 minutes up to 3.5 hours depending on the amount of leaves present in the image.

Dataset	Number of training images	Number of validation images
Dataset #1	7	3
Dataset #2	9	3
Dataset #3	40	10
Dataset #4	160	40
Test dataset	10 images	

Table 4.2: The setup of the four different training datasets as well as for the test dataset, visualization of the datasets can be seen in appendix B.

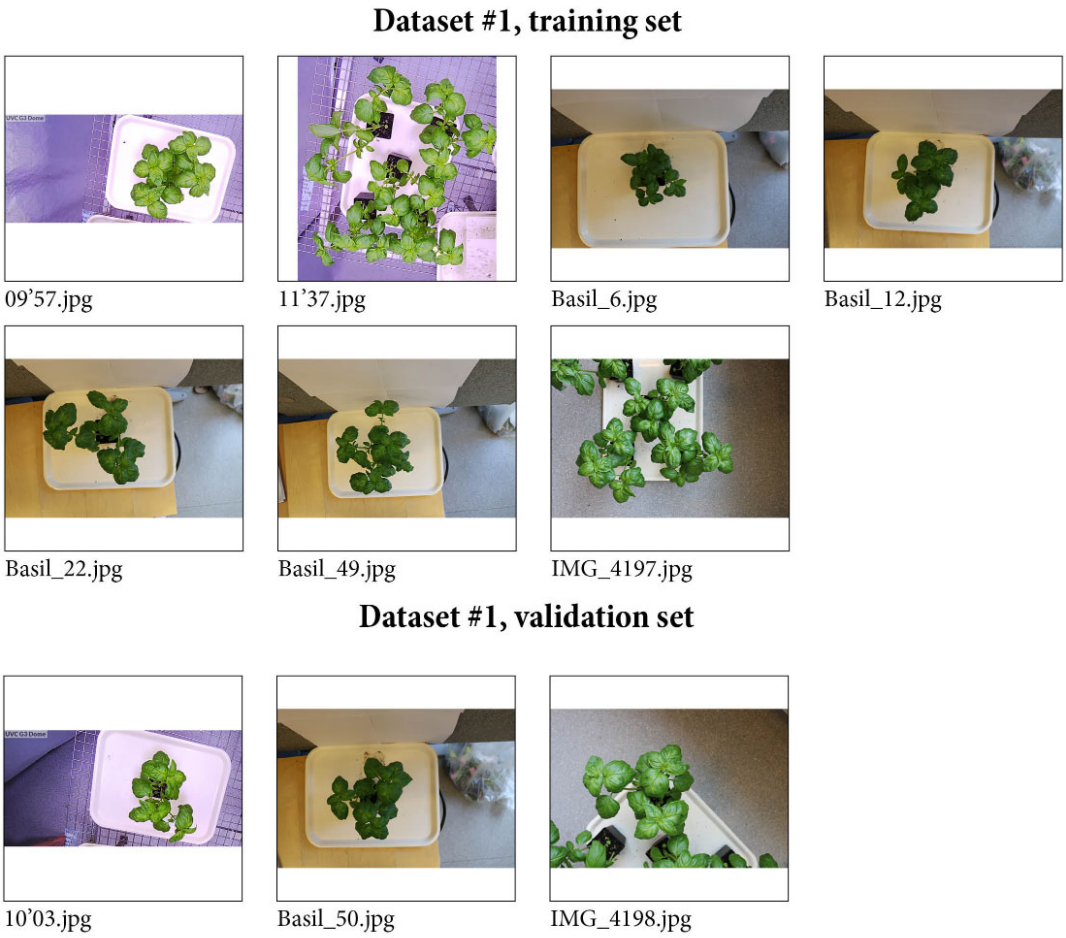


Figure 4.2: Showing the images in dataset #1, both the training set as well as the validation set. Images starting with a number was taken with the G3 Dome camera, other images were taken with the Canon 750D camera. Note that the annotations are not visible in the images.

4.3 Network evaluation

This section presents the result regarding the neural network, both the result from the training of the neural network as well as the results regarding the performance of the neural network.

4.3.1 Training of the neural network

The training of the neural network was done using different network setups, presented in table 4.3. The nomenclature seen in the table 4.3 is used throughout this chapter when describing which setup was used.
During the training of the neural network the loss functions are calculated as described in section 2.3.4. The goal during training is to minimize the loss functions whilst getting a similar value for the training loss and validation loss. The loss functions are then combined to produce a total loss function for the training as well as for the validation.

Nomenclature for network setup	**Dataset**	**Number of epochs**	**Number of runs per epoch**	**Time consumption for training**
10e 7im 10runs	Dataset #1	10	10	∼6h
10e 7im	Dataset #1	10	100	∼20h
10e 9im	Dataset #2	10	100	∼21h
30e 9im	Dataset #2	30	100	∼64h
30e 40im	Dataset #3	30	100	∼64h
30e 160im	Dataset #4	30	100	∼72h

Table 4.3: The different network setups during the training of the network.

In figure 4.3 the loss functions from the setups run with 10 epochs is presented and in figure 4.4 the result from the network setups run with 30 epochs is presented. The loss functions where calculated at the end of each epoch and summed to a total training loss and a total validation loss for each network setup.
Looking at figure 4.3, one can see that the setup with *10e 7im 10runs* is almost saturated at 10 epochs and the validation loss and training loss are beginning to converge towards one another. Further on, looking at the other two setups with 10 epochs they have a similar pattern, i.e. the validation loss are beginning to increase while the training loss are decreasing, thus they are moving further apart. Having a loss function that is significantly increasing in value during training is most likely a sign of an overfitted network which preforms worse for each epoch trained.
The same pattern, however much less significant, can also be seen in the network setups *30e 9im* as well as in *30e 40im* in figure 4.4. Moving on to the last network setup *30e 160im*, the training loss and validation loss are both decreasing in value whilst converging to one another, which is a great sign that the network will perform well.

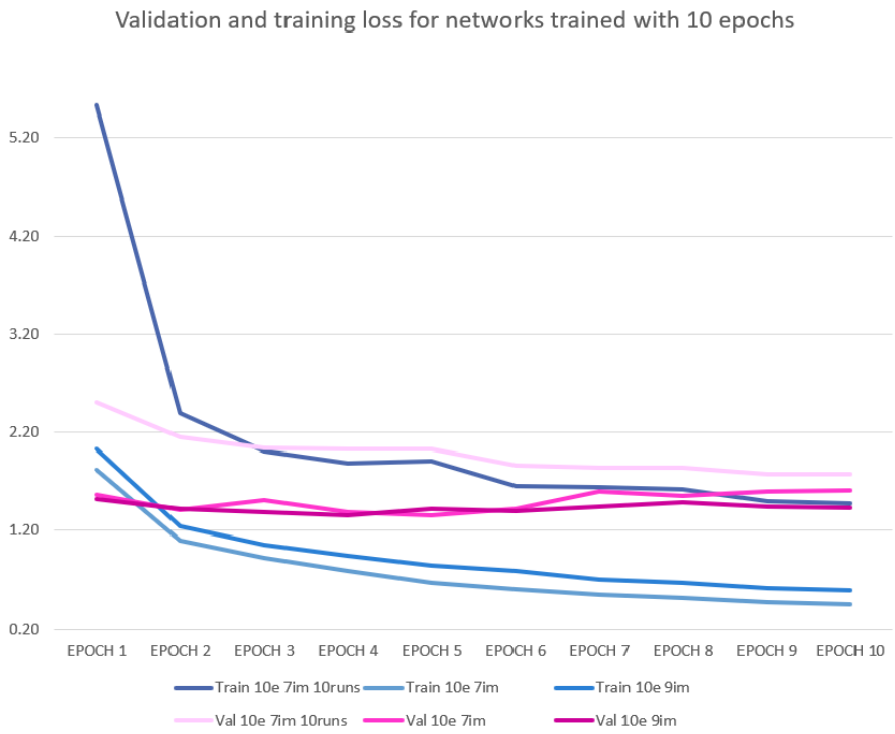


Figure 4.3: Validation loss and training loss for the network setups with 10 epochs.

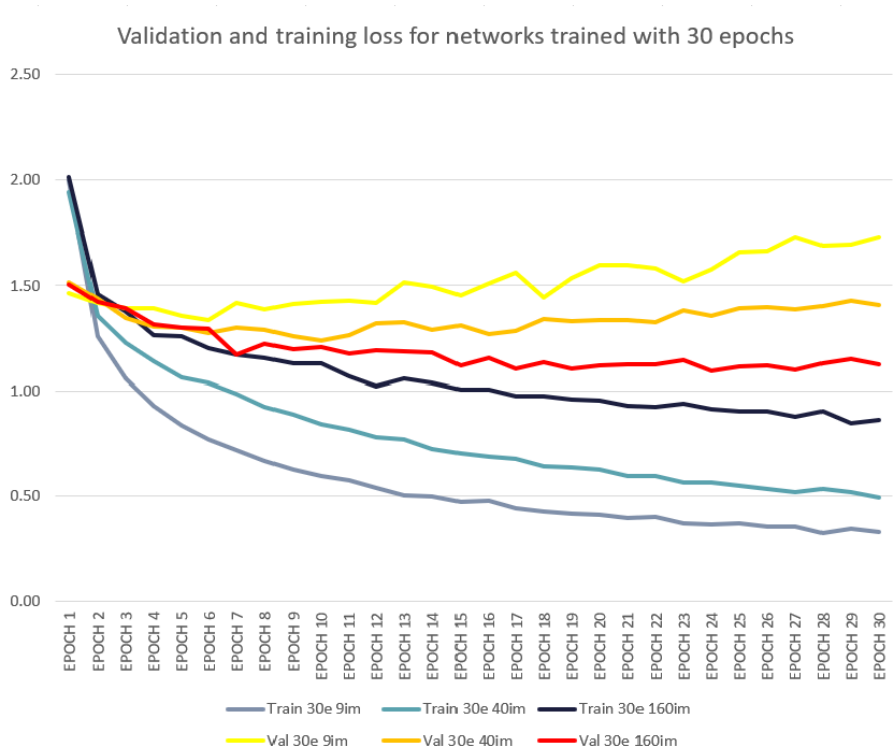


Figure 4.4: Validation loss and training loss for the network setups with 30 epochs.

4.3.2 Evaluating the network performance

To render an image with instance segmentation masks (i.e. leaves) an input image was fed into the trained neural network. In general the neural network then finds the leaves and produces an output image containing the input image with a filter on top showing the leaves in different colors in order to emphasize the instance segmentation. The neural network performed this evaluation for each image in the test dataset and for each training setup. This step took the neural network about 30 seconds to process each image in the test dataset.

The evaluation of the performance for the neural network was done, as described in section 2.3.4, by calculating the IoU_{mask} for each instance in an image. More explicitly this was done by letting the neural network also produce a file containing the meta-data for the masks, which then was imported via a script into MATLAB. The script also performed the calculation of the IoU_{mask} as well as producing a visual representation of the result, seen in figures 4.6, 4.7, 4.9 and 4.10 for the input images shown in figure 4.5 and figure 4.8 respectively. This evaluation took about 1 minute per image to calculate.

In the IoU_{mask} images (figure 4.6, 4.7, 4.9 and 4.10) the different colors represents the following; blue is the correctly predicted ground truth, red is ground truth that has been missed and green is falsely predicted areas.

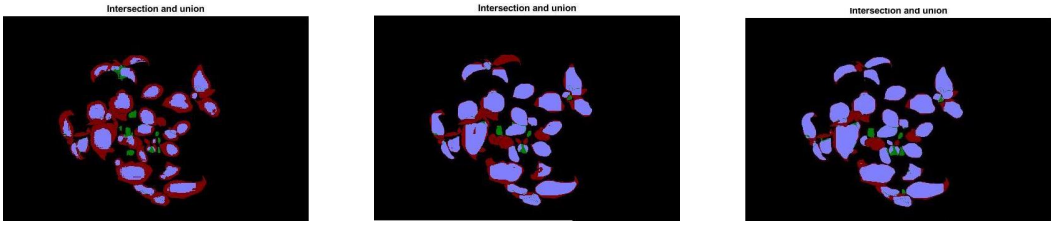
Looking at the figures 4.6 and 4.9, from (a) to (c), an increase of the blue areas as well as a decrease in the red and green areas can be seen. The figures shows the result of increasing the number of steps per epoch as well as increasing the number of images in the training dataset when the number of epochs is consistent. This increase clearly effects the neural network performance in a good way and the same pattern can be seen throughout the three datasets trained with 10 epochs, presented by the three first average data points in figure 4.11.

When solely looking at the results trained on 30 epochs instead, figures 4.7 and 4.10, the same pattern is occurring when the number of images in the training set is increasing. This can also be seen in the last three average data points in figure 4.11.

When comparing figure 4.9c with figure 4.10a, a large decrease of the blue areas as well as an increase of the red areas are shown. This change is not as significant in figure 4.6b and figure 4.7a, however over all the training dataset this change can be clearly seen as the drop between the average data points 3 and 4 in figure 4.11.

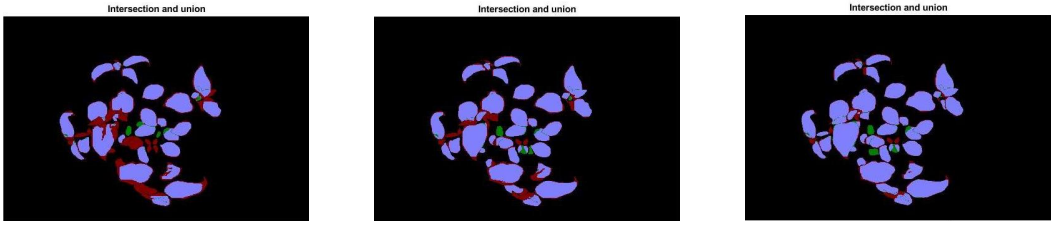
Over all the data setup with 30 epochs and 100 runs per epoch, trained on 160 images have the highest average IoU_{mask} result of 85%.

Figure 4.5: The input image, Basil-13, in the test dataset.



(a) Neural network setup: 10e 7im 10 runs. **(b)** Neural network setup: 10e 7im. **(c)** Neural network setup: 10e 9im.

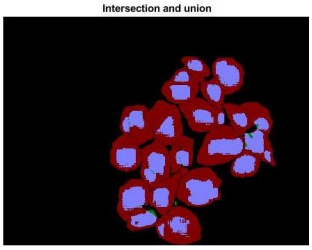
Figure 4.6: The input image, Basil-13, with different neural network setups trained on 10 epochs.



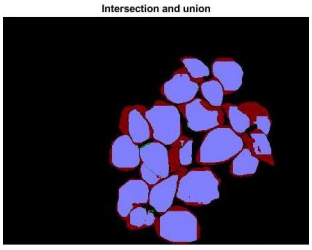
(a) Neural network setup: 30e 9im. **(b)** Neural network setup: 30e 40im. **(c)** Neural network setup: 30e 160im.

Figure 4.7: The input image, Basil-13, with different neural network setups trained on 30 epochs.

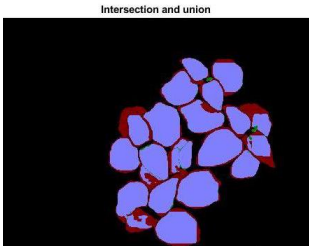
Figure 4.8: The input image, IMG-4000, in the test dataset.



(a) Neural network setup: 10e 7im 10 runs.

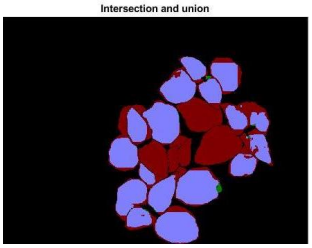


(b) Neural network setup: 10e 7im.

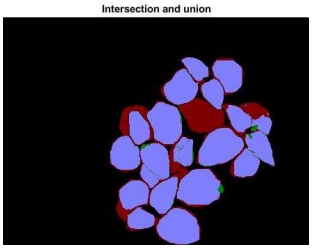


(c) Neural network setup: 10e 9im.

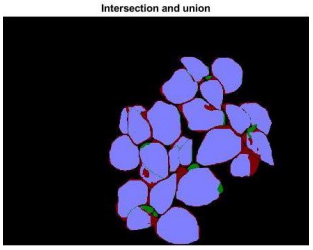
Figure 4.9: The input image, IMG-4000, with different neural network setups trained on 10 epochs.



(a) Neural network setup: 30e 9im.



(b) Neural network setup: 30e 40im.



(c) Neural network setup: 30e 160im.

Figure 4.10: The input image, IMG-4000, with different neural network setups trained on 30 epochs.

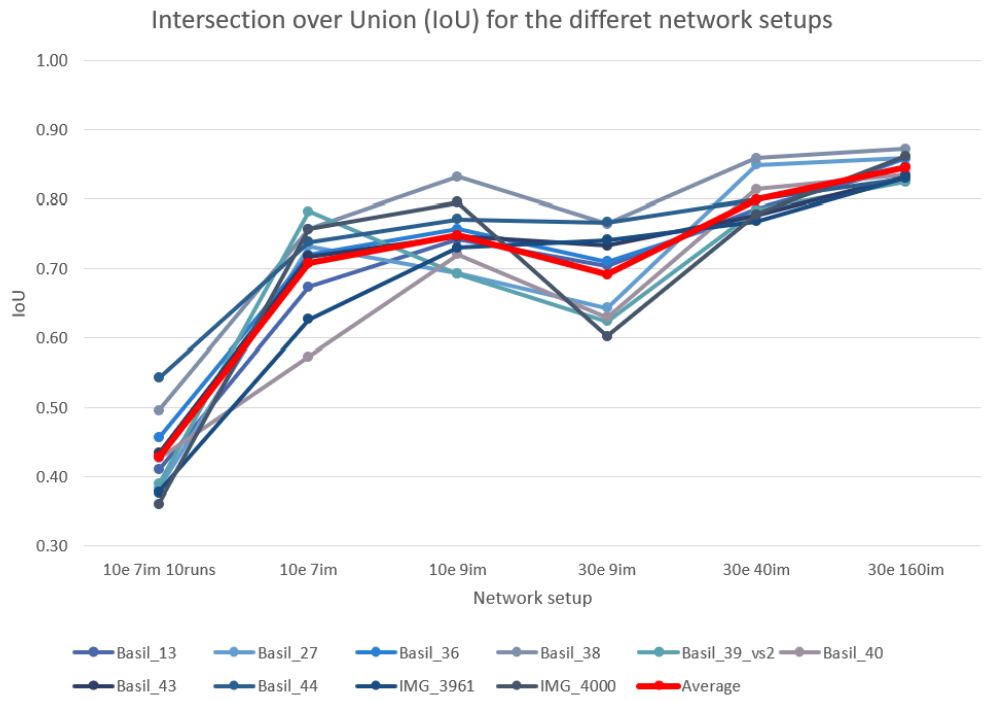


Figure 4.11: The mask intersection over union (IoU_{mask}) results for the different network setups. The IoU_{mask} result was calculated over the whole test dataset. An average for each network setup was calculated, represented as the red line in the graph.

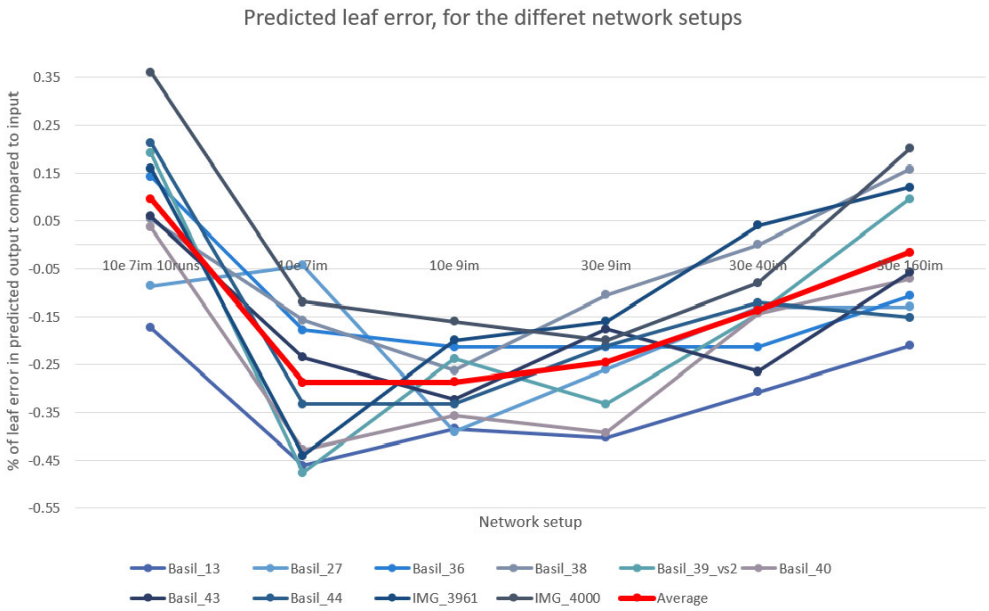


Figure 4.12: Leaf count for the different network setups. The leaf count result was calculated over the whole test dataset. An average for each network setup was calculated, represented as the red line in the graph.

While performing the instance segmentation the neural network also always performed a leaf count for each image. The leaf count was based on how many masks the neural network found and the neural network then presented an output of the sum. This result was then compared to the real leaf count number as following:

$$\frac{\text{(network calculated leaf count)-(real leaf count)}}{\text{(real leaf count)}}\,, \tag{4.1}$$

and the result for each network setup is presented in figure 4.12. The network setup that on average performed the best was the last one, dataset #4 (30 epochs and 100 runs trained on 160 images), with a deviation of -0.9% compared to the real leaf count. However the first setup (10 epochs and 10 runs trained on 7 images) with a deviation of 10% from the real leaf count is also very good.

The result is clearly effected by the fact that the neural network sometimes counted two or more leaves as one, see figure 4.13. On the other hand the neural network sometimes compensated by finding a leaf that in fact was too small to actually be counted as a leaf, see figure 4.14.

(a) A neural network result performed on input image Basil-36 and with network setup: 30e 160im.

(b) A neural network result performed on input image IMG-4000 and with network setup: 10e 7im 10runs.

Figure 4.13: Each mask is denoted by a different color to emphasize the different segmentations. The neural network segments two leaves with one mask (denoted by the red circle) in (a) or segments one leaf into several masks as in (b).

Figure 4.14: A neural network result performed on input image IMG-4000 and with network setup: 30e 160im. Each mask is denoted by a different color to emphasize the different segmentations. The neural network have segmented a leaf that is too small to be counted as a leaf (denoted by the red circle). This in turn effects the number of leaves counted.

5

Discussion

Any discussion developed during the work is presented in this chapter in order of appearance.

5.1 Data acquisition setup

The bad resolution of the images taken during the growth process could have been discovered earlier in order to gain better images during the growth process.
The placement of the plants as well as the number of plants was a good idea in theory, however the number of leaves present in the images taken during the growth process made it too extensive to annotate. Despite that, the number of plants was good in order to gain a greater dataset and diversity of plant images at the end of the growth process. Perhaps the images taken with the Canon 750D camera could have been performed a couple of times during the growth process rather than just at the end in order to gain an even better dataset.

5.1.1 Annotating images and creating the dataset

Annotating images containing more than one basil plant in it was extremely time consuming and therefore that kind of images are sparse in the datasets. This might have caused the neural network to be a less robust system.
There where some other type of difficulties when annotating the images as well. The major difficulty being when stems and leaves were overlapping other leaves, this problem was solved in different ways depending on how the situation looked like. In some cases the stem was included in the "leaf-area" as shown in figures 5.1 and 5.2.
In other cases the leaf had to be divided into two or more sections as in figure 5.3. Both of these methods are in general quite bad, since areas that should be classified as background are being included and classified as a leaf. Another bad thing is dividing a leaf into two or more sections since it is actually one leaf. This can provide a false input for the neural network while training which in hand can perform bad and produce a bad output.
Another less significant difficulty regarded blurry leaves, which made it hard to distinguish the outline of a leaf as seen in figures 5.4 and 5.5. This may not affect the neural network as much as the previously mentioned difficulty, however this made it tricky and more time consuming to annotate.

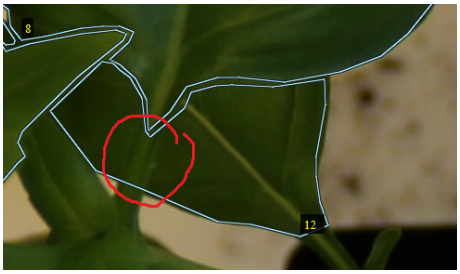
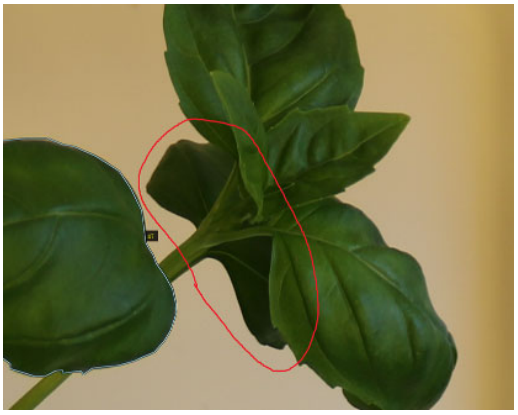
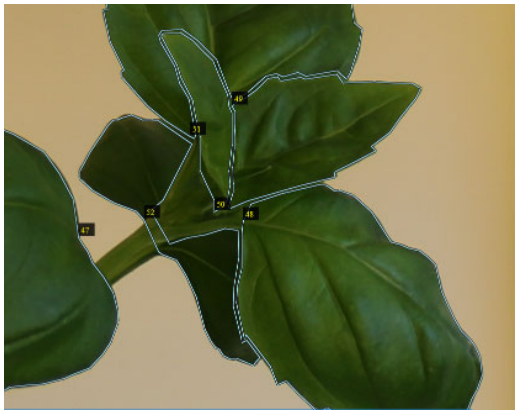


Figure 5.1: A stem from another leaf (denoted by the red circle) overlapping a leaf. The stem is included in the underlying leaf, in order for the underlying leaf to not be divided into two instances.



(a) Before annotating.



(b) After annotating.

Figure 5.2: A large stem covering the underlying leaf (denoted by the red circle). A small portion of the stem is included into the underlying leaf in order for the leaf not to be divided into two instances.

Figure 5.3: A leaf overlapping another leaf. The underlying leaf, (denoted in red) had to be divided into two separate instances.

(a) Before annotating. **(b)** After annotating.

Figure 5.4: Hard for the eye to distinguish between two separate leaves (denoted by red circle). The underlying leaf is also very blurry, making it even harder to distinguish, thus the annotations are done approximately.

(a) Before annotating. **(b)** After annotating.

Figure 5.5: Hard for the eye to distinguish between two separate leaves (denoted by red circle), thus the annotations are done approximately.

5.1.2 Augmenting images

More augmenting could have been performed in order to gain larger datasets. Apart from being rotated, the images could also have been sheared, scaled and flipped to get a more diverse dataset.

5.2 The neural network

It would have been nice to test different neural network parameter setups to see if a greater and more sufficient network could have been achieved. Small changes may include changing the training parameters even more to get a better performance.

Larger changes might include changing the backbone of the neural network or even change the whole framework.

5.2.1 Training the neural network

Training the neural network was very time consuming, and in order to have been more efficient the training should have been done on a computer containing a graphics processor unit (GPU). In comparison to a standard central processor unit (CPU), the GPU can make parallel calculations and can be up to 50–100 times faster when computing neural networks, (Olena [2018]).

5.2.2 Evaluating the network performance

The network performance regarding the segmentation worked very well, especially with the 30e 160im setup. The leaf count result however, despite it looking quite good, is misleading. This is due to the fact that the neural network sometimes divided one leaf into two or more leaves, as well as included leaves smaller than 2 cm in some cases.

5.3 Relation to medical field

The methods used regarding instance segmentation, constructing datasets and collecting the metadata can be applied onto almost any image application. And transfer learning can be used to train the neural network for the application needed.
Since instance segmentation is used (amongst other) within the areas of ex-rays, MRI and CT-scans in the medical field, the methods can be applied for a medical application, (Ronneberger et al. [2015]), (Novikov et al. [2017]).
Some newly developed applications using the Fast R-CNN framework combined with a modified version of U-Net called SkinNet in order to find, localize and segment skin lesions in images, where recently released, (Sulaiman Vesal [2018]).

5.4 Future work

Further progress of this work would be to add images from the whole growth process to the databases (training, validation and test) in order to create a more robust neural network. This can be developed further in order get the neural network to determine how old a plant is.
An interesting investigation is to see if it is possible to approximate the green area the leaves cover in an image in order to use that as a crop control parameter. It may or may not be combined with the prospects of determining how old a plant is. Since the segmentation of the images is done and some measurements were taken during the image gathering, it should be possible to determine the pixel-to-(real)cm-ratio. Since these measurement images where taken, and the placement of the cameras were done, it might also be possible to determine the height of the plants via stereo.

And finally, a development of the datasets would be to extend the datasets so they include more than just one class (background class excluded) i.e. include other plant types in the datasets. This enables for many other inventions as well, one could be for the neural network to identify and separate the different plant species.

6
Conclusion

Image analysis in its own way has been proven to be useful when applied on plants. The demanding and time-consuming part is to create the datasets in order to train and configure the neural network and get a desired result.

The neural network by itself have a high accuracy ($\text{IoU}_{mask} = 85\%$) when trained with dataset #4. Despite this, the leafcount-functionality by itself is not as sophisticated as one could hoped and as of today it will not aid the grower in a sufficient and accurate way.

One downside with the neural network (as it is implemented today) is its incapability to work in real time on a live video-feed. In order to aid the grower in monitoring the plants and their growth process, this should be a next step of implementation.

Exploring other possibilities such as determining leaf area for the plant can however prove to be fruitful since the instance segmentation for the the leaves is quite good. This however requires the implementation of a transition from pixels to cm.

So in conclusion, the thesis by itself cannot aid the grower today, but it has laid out the ground work for an image platform for Heliospectra to develop further.

Bibliography

Aashay Sachdeva [2017], 'Deep Learning for Computer Vision for the average person'. Last accessed on 2019-02-21.
URL: *https://medium.com/diaryofawannapreneur/deep-learning-for-computer-vision-for-the-average-person-861661d8aa61*

Abdulla, W. [2017], 'Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow'. Last accessed on 2019-04-05.
URL: *https://github.com/matterport/Mask_RCNN*

Andrew Ng, Kian Katanforoosh, Y. B. M. [2019], 'Improving Deep Neural Networks: Hyperparameter tuning, Regularization and Optimization'. At 4:23 in the video.
URL: *https://www.coursera.org/lecture/deep-neural-network/train-dev-test-sets-cxG1s*

Artificial Neural Network - Basic Concepts [2019]. Last accessed on 2019-03-10.
URL: *https://www.tutorialspoint.com/artificial_neural_network/artificial_neural_network_basic_concepts.htm*

Box, G. E. and Meyer, R. D. [1986], 'An Analysis for Unreplicated Fractional Factorials', *Technometrics* **28**(1), 11–18.
URL: *https://www.tandfonline.com/doi/abs/10.1080/00401706.1986.10488093*

Caruana, R. [1995], Learning many related tasks at the same time with backpropagation, *in* 'In Advances in Neural Information Processing Systems 7', Morgan Kaufmann, pp. 657–664.

Changhau, I. [2017], 'Loss Functions in Neural Networks'. Last accessed: 2019-02-01.
URL: *https://isaacchanghau.github.io/post/loss_functions/*

COCO - Common Objects in Context [2015]. Last accessed on 2019-01-30.
URL: *http://cocodataset.org//#detection-2015*

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. and Fei-Fei, L. [2009], ImageNet: A Large-Scale Hierarchical Image Database, Technical report.
URL: *http://www.image-net.org.*

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E. and Darrell, T. [2013], 'Decaf: A deep convolutional activation feature for generic visual recognition', *CoRR* **abs/1310.1531**.
URL: *http://arxiv.org/abs/1310.1531*

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J. and Zisserman, A. [2010], 'The Pascal Visual Object Classes (VOC) Challenge', *International Journal of Computer Vision* **88**(2), 303–338.

Garbade, D. M. J. [2018], 'How to Create a Simple Neural Network in Python(18:n38)'. Last accessed on 2019-03-10.
URL: *https://www.kdnuggets.com/2018/10/simple-neural-network-python.html*

Gehan, M. A., Loraine, A., Fahlgren, N., Abbasi, A., Berry, J. C., Callen, S. T., Chavez, L., Doust, A. N., Feldman, M. J., Gilbert, K. B., Hodge, J. G., Hoyer, J. S., Lin, A., Liu, S., Lizárraga, C., Lorence, A., Miller, M., Platon, E., Tessman, M. and Sax, T. [2017], 'PlantCV v2: Image analysis software for high-throughput plant phenotyping'.
URL: *https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5713628/pdf/peerj-05-4088.pdf*

Girshick, R. [2015], Fast R-CNN, *in* 'Proceedings of the IEEE International Conference on Computer Vision'.

Grenzdörffer, G. J. [2014], 'CROP HEIGHT DETERMINATION WITH UAS POINT CLOUDS'.
URL: *https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XL-1/135/2014/isprsarchives-XL-1-135-2014.pdf*

He, K., Gkioxari, G., Dollar, P. and Girshick, R. [2017], 'Mask R-CNN', *Proceedings of the IEEE International Conference on Computer Vision* **2017-Octob**, 2980–2988.

He, K., Zhang, X., Ren, S. and Sun, J. [2015], Deep Residual Learning for Image Recognition, Technical report.
URL: *http://image-net.org/challenges/LSVRC/2015/*

Hinton Geoffrey E, Krizhevsky Alex, S. I. [2013], ImageNet Classification with Deep Convolutional Neural Networks, Technical report.
URL: *http://www.cs.toronto.edu/f̃ritz/absps/imagenet.pdf*

ImageNet and MS COCO Visual Recognition Challenges Joint Workshop 2015 [2015]. Last accessed on 2019-02-15.
URL: *http://image-net.org/challenges/ilsvrc+mscoco2015*

ImageNet Large Scale Visual Recognition Challenge [2015]. Last accessed on 2019-02-15.
URL: *http://image-net.org/challenges/LSVRC/*

Jordan, J. [2018], 'Evaluating image segmentation models.'. Last accessed on 2019-01-30.
URL: *https://www.jeremyjordan.me/evaluating-image-segmentation-models/*

LeCun Yann, Bottou Léon, B. Y. H. P. [1998], 'Gradient-Based Learning Applied to Document Recognition', *Proceedings of the IEEE* pp. 2278–2324.

Lin, T.-T., Lai, T.-C., Liu, T.-Y., Yeh, Y.-H., Liu, C.-C. and Chung, W.-C. [2012], 'An Automatic Vision-Based Plant Growth Measurement System for Leafy Vegetables'.
URL: *https://pdfs.semanticscholar.org/7c5c/6e3b284fd4102e48ad998a88bc30f3769680.pdf*

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B. and Belongie, S. [2017], 'Feature Pyramid Networks for Object Detection'.
URL: *https://arxiv.org/pdf/1612.03144.pdf*

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L. and Dolí, P. [2015], Microsoft COCO: Common Objects in Context, Technical report.
URL: *https://arxiv.org/pdf/1405.0312.pdf*

MNIST dataset introduction [2017]. Last accessed on 2019-03-15.
URL: *https://corochann.com/mnist-dataset-introduction-1138.html*

Multilayer perceptron — Wikipedia, The Free Encyclopedia [2019]. Last accessed on 2019-03-27.
URL: *https://en.wikipedia.org/w/index.php?title=Multilayer_perceptron&oldid=889400320*

Novikov, A. A., Major, D., Lenis, D., Hladuvka, J., Wimmer, M. and Bühler, K. [2017], 'Fully convolutional architectures for multi-class segmentation in chest radiographs', *CoRR* **1701.08816**.
URL: *http://arxiv.org/abs/1701.08816*

Olena [2018], 'GPU vs CPU Computing: What to choose?'. Last accessed on 2019-03-23.
URL: *medium.com/altumea/gpu-vs-cpu-computing-what-to-choose-a9788a2370c4*

Peyre, G. [2010], 'Color Image Processing'. Last accessed on 2019-03-30.
URL: *http://www.numerical-tours.com/matlab/multidim_1_color/*

Phytoiiogist, H. E. W. [1912], 'the Distribution of the Flora in the', *Flora* **XI**(2).

Radha, S. [2017], 'Leaf disease detection using image processing', *Journal of Chemical and Pharmaceutical Sciences* .

Razavian, A. S., Azizpour, H., Sullivan, J. and Carlsson, S. [2014], 'CNN features off-the-shelf: an astounding baseline for recognition', *CoRR* **abs/1403.6382**.
URL: *http://arxiv.org/abs/1403.6382*

Ren, S. [2017], 'mask rcnn'. Last accessed on 2019-04-05.
URL: *https://blog.csdn.net/u013010889/article/details/78588227*

Ren, S., He, K., Girshick, R. and Sun, J. [2016], 'Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks'.
URL: *https://arxiv.org/pdf/1506.01497.pdf*

Ronneberger, O., Fischer, P. and Brox, T. [2015], 'U-net: Convolutional networks for biomedical image segmentation', *CoRR* **1505.04597**.
URL: *http://arxiv.org/abs/1505.04597*

Rosebrock, A. [2016], 'Intersection over Union (IoU) for object detection - PyImageSearch'. Last accessed: 2018-12-07.
URL: *https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/*

Scharstein, D. and Szeliski, R. [2001], 'A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms'.
URL: *http://vision.middlebury.edu/stereo/taxonomy-IJCV.pdf*

SHARMA, S. [2017], 'The Fundamentals of Neural Networks'. Last accessed on 2019-03-25.
URL: *https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53*

Sulaiman Vesal, Shreyas Patil, N. R. A. M. [2018], 'A multi-task framework for skin lesion detection and segmentation.', *Conference: Workshop on skin image analysis, MICCAI 2018, At Granada, Spain* .

Sullivan, S. [2016], 'RGB vs CMYK – Colors for the web vs print'. Last accessed on 2019-03-30.
URL: *https://www.sumydesigns.com/rgb-vs-cmyk-colors-web-vs-print/*

Weng, L. [2017], 'Object Detection for Dummies Part 3: R-CNN Family'. Last accessed on 2019-01-13.
URL: *https://lilianweng.github.io/lil-log/2017/12/31/object-recognition-for-dummies-part-3.html*

Yosinski, J., Clune, J., Bengio, Y. and Lipson, H. [2014], 'How transferable are features in deep neural networks?', *CoRR* **abs/1411.1792**.
URL: *http://arxiv.org/abs/1411.1792*

A

Measuring data during growth process

Presenting the data regarding counting the leaves of each basil plant during the growth process. On harvest day the height of each plant was also measured. The pot number refers to the pot number shown in figure 3.1 in section 3.1.1.

Table A.1: Result of the growth pattern for each basil. The first column represents the pot number for each basil plant, in order to follow each basil's growth pattern. The number in each column represents the number of leaves physically larger or equal to 2 cm. The yellow colored rows represents the pots containing only 3 plants per pot, every other pot contained 8 plants per pot. On harvest the height of the plant was measured as well, seen in the last column.

# Leaves / Pot #	7/12/2018	7/16/2018	7/19/2018	7/20/2018	23/7/2018	25/7/2018	27/7/2018	30/7/2018	8/2/2018 HARVEST DAY	HEIGHT ON HARVEST DAY
1	0	10	12	16	26	26	34	42	44	18
2	0	14	16	16	28	30	38	46	60	19
3	0	14	14	16	26	30	36	40	50	20
4	0	10	14	16	28	30	36	44	50	19
5	0	12	14	14	28	28	32	42	48	16
6	0	10	16	22	26	30	42	50	65	25
7	0	10	12	18	26	30	40	43	48	26
8	0	10	17	21	29	33	43	47	62	22
9	0	12	16	16	26	28	34	40	46	13
10	0	8	16	18	24	28	38	42	49	17
11	0	8	14	16	24	26	36	36	44	15
12	0	14	16	16	24	28	34	dried out	42	10
13	0	12	18	18	26	30	40	46	54	20
14	0	6	16	16	24	26	37	38	50	20
15	0	9	14	16	24	27	35	37	48	19
16	0	8	16	16	18	30	34	36	48	23
17	0	10	16	16	26	30	40	42	50	20
18	0	8	16	16	24	28	35	38	47	20
19	0	16	16	16	30	32	38	38	42	16
20	0	6	6	6	12	14	18	22	32	20
21	0	6	6	8	12	18	18	24	38	20
22	0	4	6	8	10	16	16	22	42	23
23	0	6	12	12	24	24	29	30	42	19
24	0	10	14	16	26	28	33	33	37	19
25	0	8	12	12	20	24	24	32	32	22
26	0	8	16	18	24	28	31	dried out	35	11
27	0	4	12	16	22	26	28	33	38	21

Table A.2: Cont. Result of the growth pattern for each basil. The first column represents the pot number for each basil plant, in order to follow each basil's growth pattern. The number in each column represents the number of leaves physically larger or equal to 2 cm. The yellow colored rows represents the pots containing only 3 plants per pot, every other pot contained 8 plants per pot. On harvest the height of the plant was measured as well, seen in the last column.

28	0	6	10	15	21	21	24	27	29	21
29	0	8	14	16	22	25	29	34	35	24
30	0	4	10	10	16	18	20	24	24	20
31	0	10	16	16	26	32	34	37	46	20
32	0	8	11	13	22	24	24	28	28	23
33	0	4	12	16	26	28	34	38	42	21
34	0	4	8	8	14	16	20	20	22	22
35	0	8	16	16	24	26	34	36	40	20
36	0	6	14	14	22	26	27	34	36	22
37	0	10	14	14	26	26	34	36	38	22
38	0	6	13	16	22	28	30	32	41	21
39	0	2	12	14	22	30	32	34	36	13
40	0	4	12	16	22	26	28	33	34	19
41	0	6	12	16	24	28	31	34	41	20
42	0	12	16	16	30	30	37	43	45	17
43	0	10	15	15	28	30	40	44	58	23
44	0	14	16	16	28	30	36	48	54	21
45	0	6	17	17	25	31	33	34	40	21
46	0	2	16	16	21	26	33	38	46	19
47	0	6	16	18	30	30	39	40	52	20
48	0	14	14	18	28	36	38	44	58	23
49	0	6	6	6	12	16	18	30	42	24
50	0	6	6	6	12	18	18	24	38	22

B

Datasets

This appendix contains the datasets not presented in the report (dataset #1 is presented in section 4.2) and describes which images that are present in which dataset. Dataset #4 is not present in its entity, since it is composed of dataset #3 with all of the images rotated 3 times.

Image name in training dataset	Dataset #1	Dataset #2	Dataset #3	Dataset #4
09'57.jpg	x	x	x	x
09'58.jpg			x	x
10'00.jpg			x	x
10'01.jpg			x	x
11'37.jpg	x	x	x	x
Basil_1.jpg			x	x
Basil_3.jpg			x	x
Basil_4.jpg			x	x
Basil_5.jpg			x	x
Basil_6.jpg	x	x	x	x
Basil_7.jpg			x	x
Basil_9.jpg			x	x
Basil_10.jpg			x	x
Basil_11.jpg			x	x
Basil_12.jpg	x	x	x	x
Basil_13_vs2.jpg			x	x
Basil_14.jpg			x	x
Basil_15.jpg			x	x
Basil_16.jpg			x	x
Basil_17.jpg			x	x
Basil_18.jpg			x	x
Basil_20.jpg			x	x
Basil_22.jpg	x	x	x	x
Basil_23.jpg			x	x
Basil_24.jpg			x	x
Basil_26.jpg			x	x
Basil_28.jpg			x	x
Basil_29.jpg			x	x
Basil_30.jpg			x	x
Basil_32.jpg			x	x
Basil_34.jpg			x	x
Basil_35.jpg			x	x
Basil_45.jpg			x	x
Basil_47.jpg			x	x
Basil_48.jpg			x	x
Basil_49.jpg	x	x	x	x
IMG_3967.jpg		x	x	x
IMG_4195.jpg			x	x
IMG_4196.jpg		x	x	x
IMG_4197.jpg	x	x	x	x

Figure B.1: List of images in the different training datasets. The x marks if an image belongs to a dataset.

Image name in validation dataset	Dataset #1	Dataset #2	Dataset #3	Dataset #4
10'03.jpg	x	x	x	x
Basil_2.jpg			x	x
Basil_8.jpg			x	x
Basil_19.jpg			x	x
Basil_21.jpg			x	x
Basil_25.jpg			x	x
Basil_31.jpg			x	x
Basil_46.jpg			x	x
Basil_50.jpg	x	x	x	x
IMG_4198.jpg	x	x	x	x
Image name in test dataset	**Dataset #1**	**Dataset #2**	**Dataset #3**	**Dataset #4**
Basil_13.jpg	x	x	x	x
Basil_27.jpg	x	x	x	x
Basil_36.jpg	x	x	x	x
Basil_38.jpg	x	x	x	x
Basil_39_vs2.jpg	x	x	x	x
Basil_40.jpg	x	x	x	x
Basil_43.jpg	x	x	x	x
Basil_44.jpg	x	x	x	x
IMG_3961.jpg	x	x	x	x
IMG_4000.jpg	x	x	x	x

Figure B.2: List of images in the different validation datasets at the top. List of images in the different testing datasets at the bottom. The x marks if an image belongs to a dataset.

Figure B.3: Showing the images in dataset #2, both the training set as well as the validation set. Images starting with a number was taken with the G3 Dome camera, other images were taken with the Canon 750D camera.

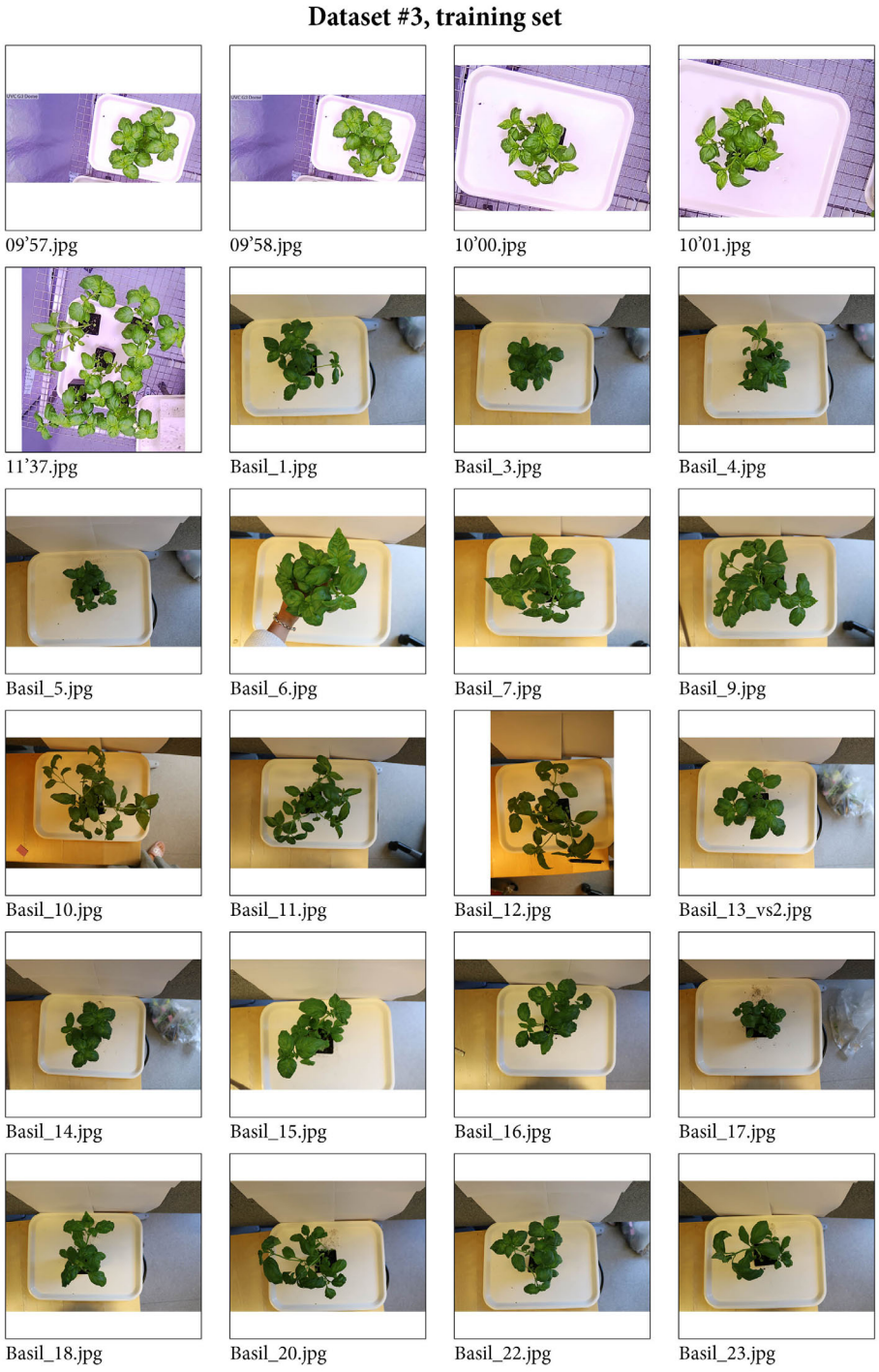


Figure B.4: Showing some of the images in the training set of dataset #3. Images starting with a number was taken with the G3 Dome camera, other images were taken with the Canon 750D camera.

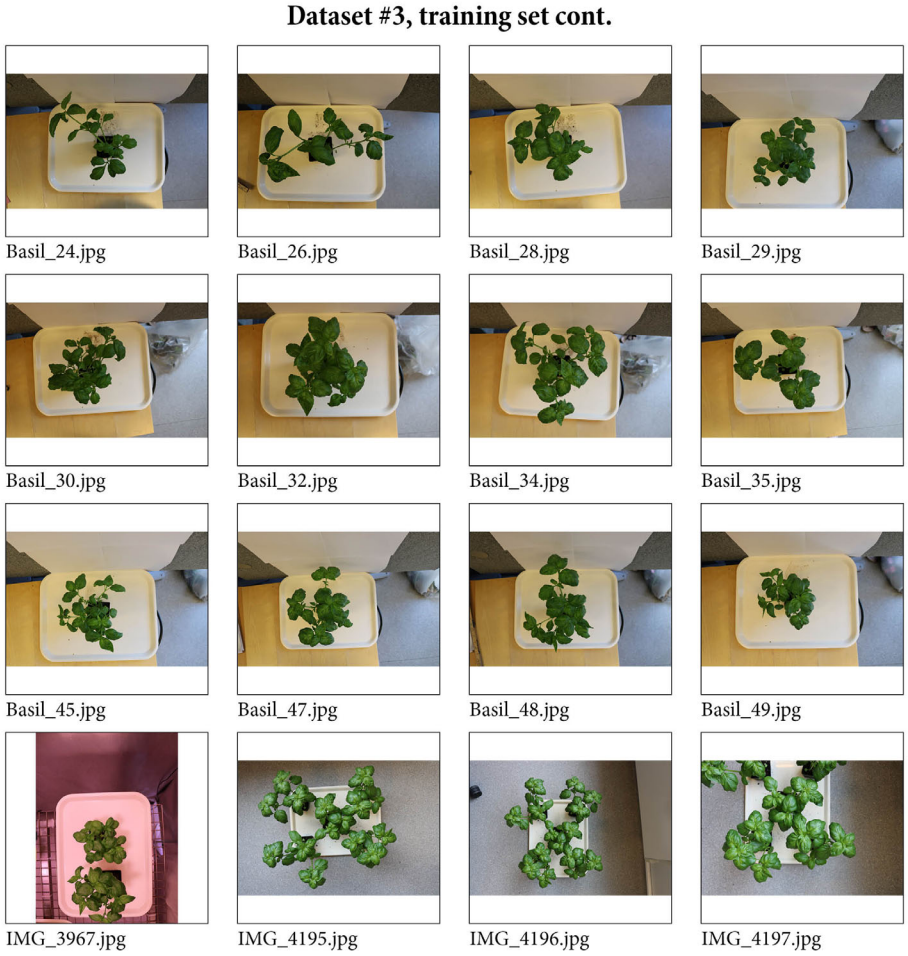


Figure B.5: Showing the rest of the images in the training set of dataset #3. All images were taken with the Canon 750D camera.

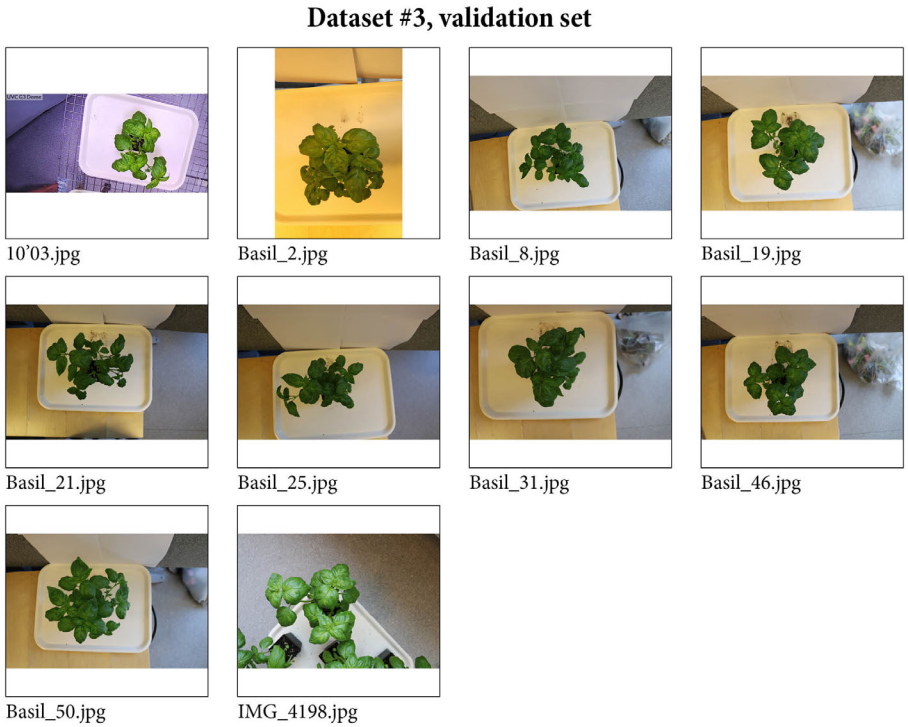


Figure B.6: Showing the images in the validation set of dataset #3. Images starting with a number was taken with the G3 Dome camera, other images were taken with the Canon 750D camera.

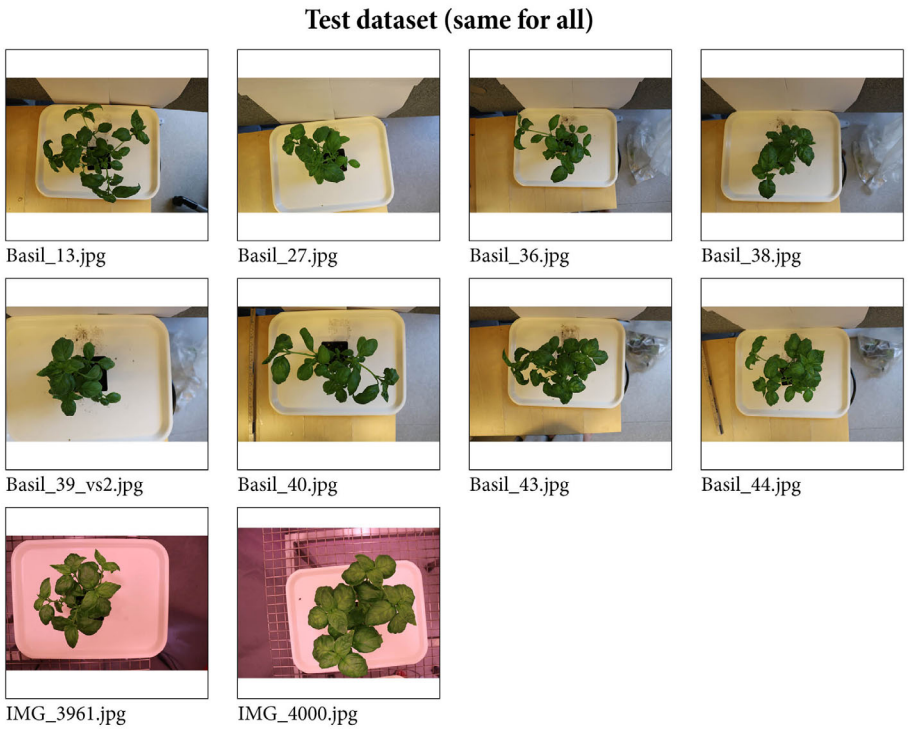


Figure B.7: Showing the test dataset, which is the same for all four datasets. All images were taken with the Canon 750D camera.