Copy and paste synthetic dataset generation in agriculture

An investigation in reducing the need for manual annotation using generated datasets.

Master Thesis

Martin L. Jepsen and Mikkel D.B. Jeppesen

Aalborg University
Electronics and IT

Copyright © Aalborg University 2021

Electronics and IT
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Generate synthetic data for training

Theme:
Computer Vision

Project Period:
Spring Semester 2021

Project Group:
Group 1040

Participant(s):
Martin L. Jepsen
Mikkel D.B. Jeppesen

Supervisor(s):
Thomas B. Moeslund
Christoffer B. Rasmussen

Copies: 1

Page Numbers: 55

Date of Completion:
July 3, 2021

Abstract:

The project investigates if the use of relatively simple methods to generate synthetic datasets in the are of weed detection can yield comparable results to training models on conventionally annotated datasets. The problem analysis considers different datasets to base the work on, and explores the structure of the chosen dataset to then later use this structure information when generation several different synthetic datasets using a Cut, Paste and Learn approach by Dwibedi et al. The problem analysis also briefly discusses the choice of segmentation model for testing. Following the analysis, the Design and Implementation of the dataset generation and segmentation model is described. Testing and results then describe how the methods explained are applied to generate different datasets with different blending techniques, and present the results, where surprisingly fully synthetic datasets outperformed the conventionally annotated dataset.

The content of this report is freely available, but publication (with reference) may only be pursued due to agreement with the author.

Contents

Preface

We would like to take this opportunity to thank our supervisors Thomas B. Moeslund and Christoffer B. Rasmussen, and also thank CLAAS for the opportunity to work on this project.

Aalborg University, July 3, 2021

Martin L. Jepsen
<mjepse16@student.aau.dk>

Mikkel D.B. Jeppesen
<mjeppe15@student.aau.dk>

Chapter 1

Introduction

Machine learning has emerged as a promising new tool in the field of agriculture. As the paper by Liakos et al. [1] explores, machine learning has been used in many different fields of agriculture, especially in crop management, where it has for example been used for yield prediction, disease detection, crop quality, weed detection and more.
In Mishra et al. [2] and their work on spectral imaging of plants, they mention how different forms of machine learning, such as CNN based methods are good approaches for utilizing the sensor data they gather.
The work done by Osorio et al. [3] is an example where such image data was processed with the use of machine learning. They implemented three different deep learning based approaches for estimating the amount of weed in a field.

Another use of machine learning in agriculture was suggested by Christiansen et al. [4]. They developed a deep neural network for detecting anomalies in the context of robotic farming, allowing automated farming equipment to reliably detect and avoid unexpected obstacles.

As seen machine learning in agriculture is an active field and for good reason. The agricultural industry is vital for our way of life, providing the food we need to survive. The more efficient the the field of agriculture can operate, the more food we can produce for the same impact on our environment. This project focuses on the task of using machine learning in weed detection, as weeds are a major threat to crop production, and being able to selectively/mechanically remove them saves on expensive and environmentally problematic herbicides [5].

Training a network to distinguish between crops and weeds is a difficult task as they often are very similar, and for a model to be successful a large good quality dataset to train on is necessary.
Creating such datasets is a large and expensive task, as several thousand examples have to be manually annotated. Depending on the task the model needs to solve,

this annotation can vary from simply labeling each sample, for example with image classification, or can be as complex as annotating all pixels in an image in the case of image segmentation. The larger the dataset, and the more complex the required annotation, the more time and money has to be spent on creating a good dataset to train a model on.

With the current method, a representative dataset has to be collected. A dataset that has as many variations of the desired classes such that the chosen model can learn as many variations of the different classes as possible. This collection task is itself a sizeable task, however this data then has to be further annotated.

One method to reduce this task of annotation is to attempt to automate the annotation task. Saripalli et al. [6] developed an AI assisted annotation tool, that would attempt to automatically annotate new data based on previous expert annotated data through reinforcement learning, and with further expert input would continously improve. Maninis et al. [7] developed a method (DEXTR) that would generate detailed polygon annotations given just the extreme points of an object. A system like This reduces the amount of work an annotator has to put in per image, and produced high quality annotations.

A different strategy used to reduce or eliminate the need for manual image annotation, is the synthetic generation of a dataset. Generating a dataset entirely allows the annotation to be perfectly generated along with the samples. Henriksen et al. [8] developed a system to generate point-cloud data for a virtual ToF sensor in a virtual sewer system. This system allowed entirely automated generation of a point-cloud dataset for sewer inspection. Their work was based off of the work by Prakash et al. [9] on Structured Domain Randomization (SDR), which showed that a fully synthetic dataset could be generated and used to train a network which performed competitively with conventional datasets without having to generate photo realistic images.

While SDR showed that photo-real data isn't always necessary, there are times where the task at hand doesn't fit the approach. In these cases photo-real datasets are of interest, and different methods for generating these have also been explored. One such method was proposed by Ghorbani et al. in their DermGAN paper [10] where they used a U-net based GAN to transfer rare skin conditions to a variety of skin types with the goal of augment otherwise sparse datasets.

Similarly Dwibedi et al. [11] proposed a method of copy pasting objects into background scenes and blending them into the image, to quickly create a large dataset of images for training object detection models. By copy pasting the images they achieve photo-realistic objects by default. Instead the work was in blending

these objects into the background in a way that wasn't visible to a machine learning model.

Our project aims to investigate how synthetically generated datasets could help in the field of agriculture. If synthetically generated datasets could help speed up the training or re-training of models used to detect and control weed on farm land.

1.1 Initial problem statement

What possibilities are there for easing the process of acquiring datasets in agriculture by synthetically generating it?

Chapter 2

Problem Analysis

2.1 Methods for generating synthetic datasets

As discussed in the introduction, there are several different ways to produce synthetic datasets. This section discusses three different approaches. Photo realistic renderings, style transfer using generative networks and the simple Cut, Paste and Learn method by Dwibedi et al. [11]

2.1.1 3D Rendering for synthetic datasets

One method of generating a synthetic dataset is to model the scene in enough detail to generate photo-realistic renders. Cicco et al. [12] developed such a method for full 3D rendering of a sugar-beet and weed dataset with physics modeling of the leaves as well. While this method, if implemented with a sufficient level of detail, can perform very well, it requires a lot of work in modelling the desired classes, as not just the shape, but also the material properties and light interactions have to be properly modeled.

2.1.2 Synthetic images using Generative networks

Lin et al. [13] developed a network, (SYN-MTGAN), for generating a more balanced dataset of street signs. Their project aimed to generate more samples of under represented classes, as the frequency of difference classes were very different. Their approach for generating samples uses their network to transfer the style of real images to fake images, and in this way generate more samples for use in training. Using their method they managed to improve model performance on an object detector model compared to using purely conventional datasets.

2.1.3 Cut, Paste and Learn

In the paper by Dwibedi et al. [11] a method of dataset generation is proposed that avoids the complexities of computer rendering, or the training of generative networks. Instead they achieve photo-realistic images by using real images and inserting them onto real backgrounds. This simple approach allows them to generate large datasets from a collection of background images, and a collection of images of their desired classes. Simply pasting the images onto the backgrounds however isn't sufficient, and their work focuses on the process of blending these objects into the background in such a way that a neural network trained on these images would correctly learn the different classes and not pasting artifacts.

Their testing showed that while their synthetic datasets didn't outperform a conventional dataset on it's own, it did outperform a conventional dataset if they mixed 10% of the conventional dataset into their synthetic dataset.

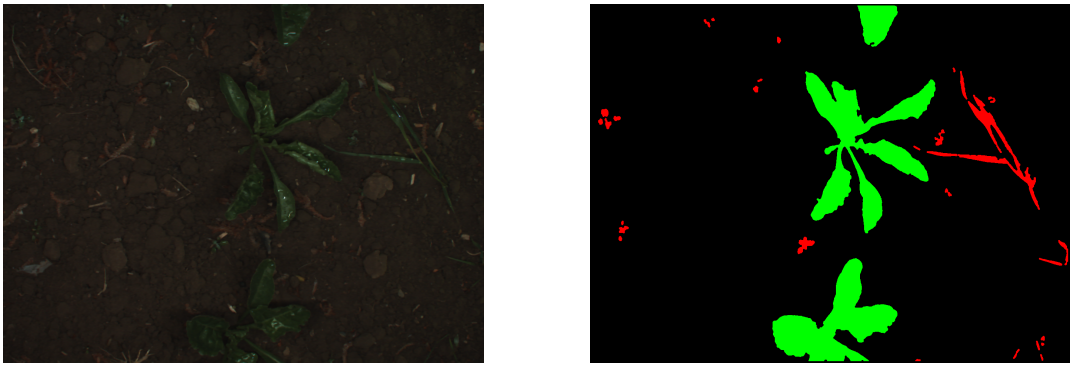
2.2 Datasets in agriculture

In order to evaluate the use of generated synthetic datasets in agriculture, a convectional dataset to compare against has to be selected. In the field of agriculture multiple public dataset have been made to use in computer vision tasks, where available datasets have been compiled in a survey by Lu and Young [14] and more datasets have been listed in the survey by Kamilaris and Prenafeta-Boldú [15].

Some other dataset available in agriculture is the Weed-AI platform from Precision Weed Control Group and Sydney Informatics Hub [16]. The dataset available there is all bounding box notations of weed, the downside of this type of annotations is the additional work need to segment the weed if extraction of only the weed and not the ground i needed. Additionally only the weed is annotated so to get crop extracted from the dataset these also need to to be annotated. These dataset were therefor not suitable for our use case.

The survey [14] divided the datasets up into 3 different categories, weed control, fruit detection and other. In addition to these categories the survey also provides the size of the dataset, type of annotation, how the data it acquired and what type of data is given for each dataset. This make it easier to identify relevant datasets for our use case. As the focus of the project is mechanical weed control as stated earlier in the introduction chapter 1, datasets suitable for this will be looked into. This means image segmentation datasets, as the provide very accurate localization of the different objects. From this survey two datasets with a large amount of segmentation annotated images was found. The dataset by Chebrolu et al. [17]

with 12,340 images and by Sa et al. [18] with 10,196 title images.



(a) RGB image **(b)** Corespondent color mask

Figure 2.1: Example of image from [17].

The dataset from [17] was chosen, as it has a higher resolution of the crops and weeds as each image has a resolution of 1296×966 pixels over a area of 24 cm × 31 cm which will give a resolution of approximately 3 pixel/mm as it can be seen on figure 2.1, where [18] highest resolution of 8.2 cm/pixel which is 0.01 pixel/mm.

2.3 Distribution of crops and weed in data-set

As the data-set is a series of images of sugar beets, there's a large class imbalance. The largest class is the background dirt, especially on images in the early stages of the crops growth. Similarly, as weeds aren't intentionally planted, the weed class has very few samples compared to either the crop or the dirt class. Because of this large class imbalance, a model trained with no class weighing would reach close to 100% accuracy by just predicting all pixels as dirt.

The distribution of classes on the data-set can be seen in table 2.1

Background	Weed	Crop
98.5%	0.2%	1.3%

Table 2.1: The distribution of the different classes in the data-set

In order to train a model on this data-set, a method for dealing with the class imbalance is necessary. While over-sampling under-represented classes or under-sampling over-represented classes is a common approach in other learning tasks, it doesn't apply well to our data-set as context around the pixels is important. Instead the loss is weighed for each pixel depending on its class, giving more importance

to under-represented samples. To find these class weights, several approaches can be used. The first and simplest idea is to weigh each class inversely related to it's frequency in the data-set as shown in equation (2.1).

$$W_{C_i} = \frac{N}{C * N_{C_i}} \tag{2.1}$$

where:

- W_{C_i} is the weight of class C_i
- N is the total number of samples
- C is the total number of classes
- N_{C_i} is the total number of samples in class C_i

However using this lead to very unstable behaviour and caused the model training to fail. This is likely due to the very large weights for the under-represented classes.
Another method of weight balancing was chosen, *median frequency weighing* as described in the paper by Eigen et al. [19]. This was chosen as recent studies have found that inverse frequency weighing isn't optimal, and instead a smoothed version performs better [20]. To calculate this weight equation (2.2) and equation (2.3) is used.

$$W_{C_i} = \frac{median\_freq}{freq(C_i)} \tag{2.2}$$

$$freq(C_i) = \frac{N_{C_i}}{N_{P_{C_i}}} \tag{2.3}$$

where:

- W_{C_i} is the weight of class C_i
- N_{C_i} is the total number of samples in class C_i
- $N_{P_{C_i}}$ is the total number of pixels in images where class C_i is present
- $median\_freq$ is the median of the frequencies for all classes

Additionally, as described in the paper by Cui et al. [21], we re-scale the weights such that the sum of the weights come out to the number of classes C.

2.4 Choice of detection model

In order to evaluate how the different datasets compare to each other, they have to be used to train a machine learning model. As the dataset is annotated for segmentation the model has to be capable of image segmentation.

While there are a lot of networks available for image segmentation, the absolute performance isn't the important metric. The main goal is to compare how well the different datasets used to train the model impacts the performance of the model.

As such the choice of model was made after looking at options such as Mask R-CNN [22], Deeplab V3 [23] and U-net [24] and the U-net network was chosen.

U-net was chosen, as it is a popular network having been cited over 25000 times according to google scholar as of 2021-07-01. Compared to the other networks, its structure is fairly simple, and as such is quicker to implement. As the focus isn't on the absolute model performance, this is an important benefit. Additionally it is a single network that has to be trained compared to Mask R-CNN which consists of two networks, a Region Proposal Network (RPN) that finds regions of interest in the image, and then a second network that then performs the image segmentation.

2.5 Success criteria

In order to know if the synthetic generated data would be a viable alternative to the real data, some kind of criteria is needed. These success criteria will be based on the result from [17] as it is the method for generating synthetic images that the solution in this report will be based on. As [17] was able to get results of 5 % lower than the real dataset, this same limit will be considered sufficient model performance in our implementation.

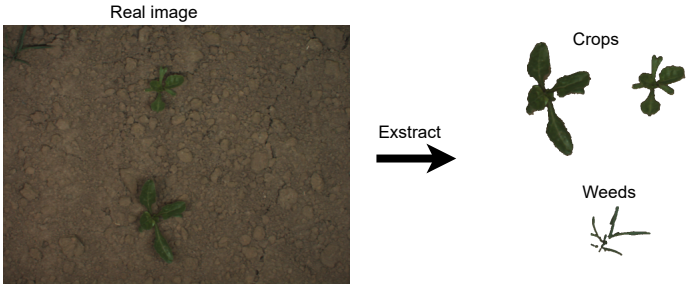
2.6 Problem statement

> *Can the use of a simple cut and paste method for generating synthetic data for machine learning perform close to or better than conventionally collected and annotated datasets in crop/weed image segmentation?*

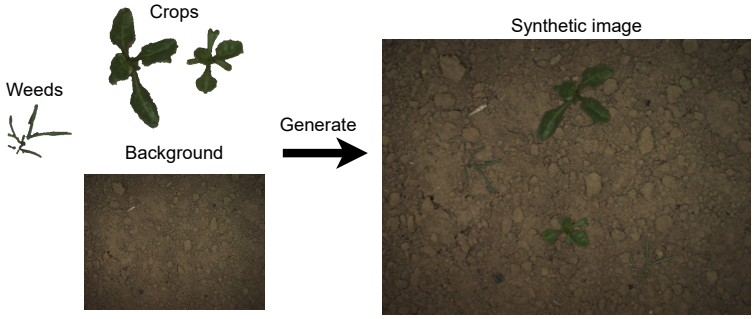
Chapter 3

Design and implementation

As stated from the problem statement in section 2.6 our project will implement the cut and paste method, the first step of the design and implementation is the extraction of weed and crops from the dataset [17] as illustrated in figure 3.1a, where a synthetic dataset then will be made from these extracted weeds and crops, as it can be seen on illustration in figure 3.1b, Lastly U-net in which the synthetic dataset needs to be tested on need to be designed. Each of these step will be described in detail trough out this chapter.



(a) Illustration of extraction of weed and crop.



(b) The concept of generation synthetic image from background and cut out crops and weeds

Figure 3.1: Illustration of the process of generating synthetic data.

3.1 Synthetic image generation

In order to generate a synthetic dataset using the copy paste method described in section 2.1.3 a source of the class images and backgrounds have to be chosen. As our project aims to investigate the use of synthetically generated datasets for detection of weeds and crops, the source for our synthetic dataset, and the dataset we'll be comparing to, is the "SugarBeets2016" [17] dataset. This dataset has a large collection of RGB images of crops and weeds with annotation masks.

3.1.1 Background images

In order to get some clear backgrounds for pasting in crops and weeds, the dataset was parsed for images where no crop or weed was present. 13 such images were found and could be used as the background images for pasting instances of the crop and weed classes onto.

3.1.2 Crops and weeds

The original paper used simple one object images of the objects they wanted to place in the scene, and then used a machine learning model to segment and extract the objects, however as the dataset we use already has full class segmentation masks these can instead be used to segment the objects from the images. To do this, the two color segmentation maps were first split into binary masks for each class as shown on figure 3.2.

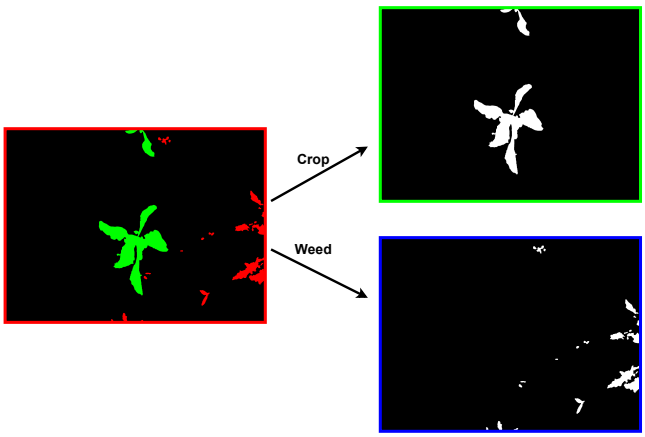


Figure 3.2: Illustration of how a color map is divided into a weed and crop mask

With the crop and weed masks split into their own masks, these could be used to extract the crops and weeds. A diagram of the method used for extracting these can be seen on figure 3.3.

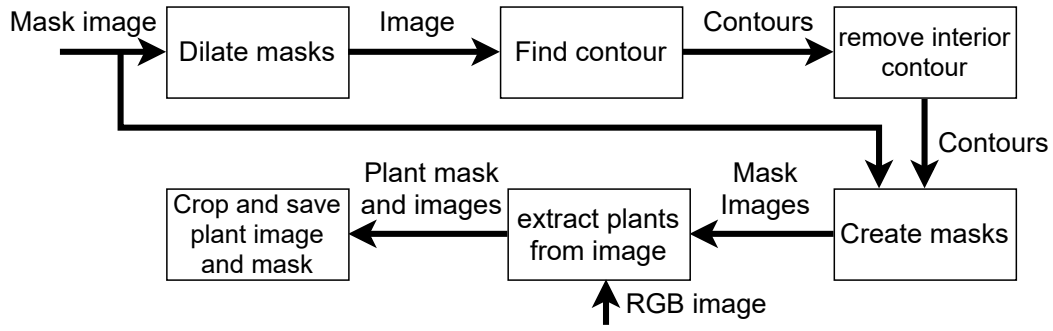


Figure 3.3: Diagram of the extraction of plant and corresponding mask.

As the masks for the plants can have some leaves show up as separate closed contours while they still belong to the same plant, the masks are dilated slightly to ensure that one continuous mask is found for a single plant. The smaller non-dilated masks are still retained, as these are what will be used to segment the crops later, however the larger continuous mask is used to represent the plant for positioning and distance measurements. This larger plant contour can have voids in it, as the contours of the individual leaves are only dilated enough to slightly over-lap. These voids are smaller contours fully contained within the larger plant contour, and are removed, as they represent soil around the stem of the plant.

Once the plants have been localized with this outer contour, the original leaf contours are used to make a new mask for each plant. This way the algorithm can segment individual plants made up of separate leaves.

The end result is a collection of crops and weeds exported to individual files to later be used to generate a synthetic dataset.

Each step of the extraction process from figure 3.3 is shown in figure 3.4 for the crops part of the annotation from figure 3.2.

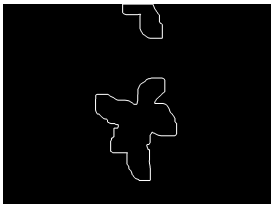
Contour

In order to find the contour of a plant, the function *findContour()* from OpenCV is used, which takes a binary image and finds all the contours in it. The algorithm used by OpenCV to find the contours is proposed by Satoshi Suzuki and Keiichi Abe in "*Topological structural analysis of digitized binary images by border following* [25]".

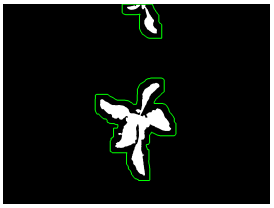
Dilate mask

As the mask image is a binary image, dilation can be used. Dilating is part of mathematical morphology, which is a method where a kernel is used on each pixel of the mask/image [26]. Morphology have multiple operations that can me used

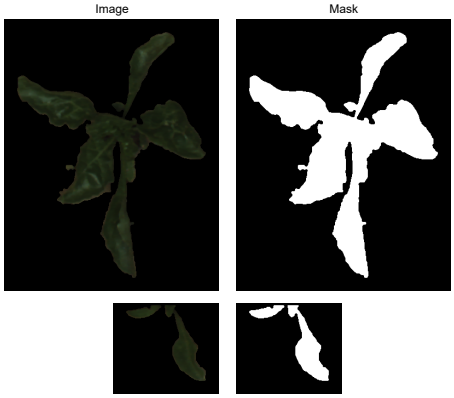
(a) The crop mask extracted in figure 3.3 is dilated



(b) The contour of the dilated mask is found and as there is no contours inside, none are removed.



(c) The masks inside the dilated contour are all considered one plant, and used to extract it.



(d) The extracted crops image and mask

Figure 3.4: Here is an example of the process of extracting crops from the original dataset.

on the mask, but as only dilation is used, only this will be explained.

Each pixel in a binary image is represented by ether '0' or '1', where '1' is the masking bit. By applying a kernel on all pixel a dilated mask will be created. This kernel is denoted as structuring element (SE) and equation (3.1) show how it is applied.

$$g(x,y) = f(x,y) \bigoplus SE \tag{3.1}$$

figure 3.5 shows an example where the structuring element is 3×3.

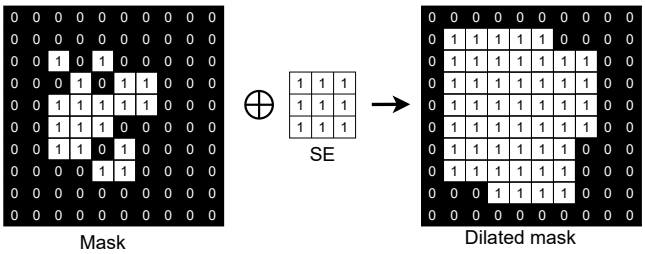
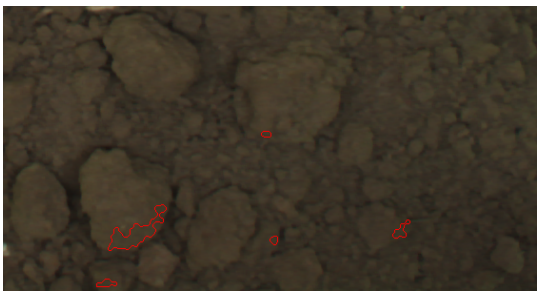


Figure 3.5: Illustration of a mask being dilated by a 3×3 SE.

3.1.3 Annotation quality

After extracting crops and weeds from the dataset, it was noted that several extracted crops or weeds were flawed, which lead to finding out that several annotations in the dataset were inaccurate. The errors that were found can be divided into two different types. One error has weed annotated as part of a crop (figure 3.6a) another annotated soil as weed (figure 3.6b). In order to ensure high quality annotations in the synthetic dataset, all extracted crops and plants were inspected and faulty annotations were removed. Depending on how many such flawed annotations are present in the real dataset, it could impact the model performance when trained on it.

(a) Weed and crop notated as one crop.



(b) Ground notated as weed.

Figure 3.6: Examples of two different bad notations. Green notation as crops and Red notating as weeds

3.2 Generation of synthetic dataset

The method used to generate synthetic images is based on the cut and paste method described in section 2.1.3. Before one or more of the extracted objects are placed on a background image the location to place them have to be found. Where the original paper [11] placed the objects randomly, we aim to more closely mimic the parent dataset. In order to do that the placement of plants and crops have to be determined.

3.2.1 Analyse real dataset

When creating a synthetic dataset, the position of the weed and crops are necessary to place the plants in a realistic manner. Crops are usually planted in a row with a fixed interval and a fixed space between rows as it can be seen on 3.7. Weeds on the other hand aren't intentionally planted, and could appear anywhere, however their distance to the crops is a metric that can be used to characterize their positioning. Lastly the amount of crops and weeds in an image is also an important metric.

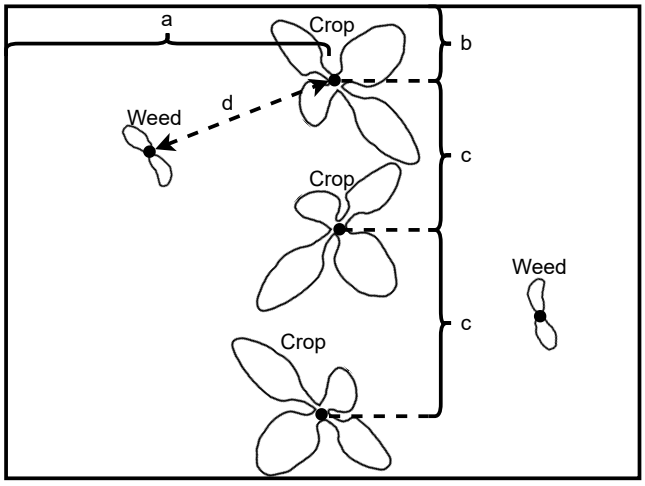


Figure 3.7: Illustration of crops and weeds positions and relevant metrics used to characterize the dataset. (a) is the x-axis position of the crop centroids. (b) is the crops distance from the top of the image. (c) is the distance between crops along the rows and (d) is the distance from a weed to the nearest crop.

To analyse the crops, the dilated mask contours from section 3.1.2 representing the plants are used. First the centroids for each contour is found and represents the centroid of the plant. All plant horizontal positions are then collected and plotted

in a histogram as seen on figure 3.8. The distribution can be approximated by a normal distribution, so the median and standard deviation is found.

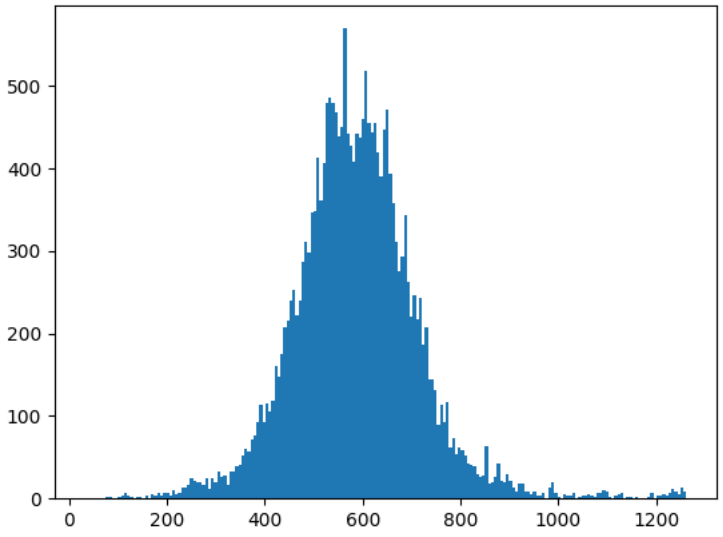


Figure 3.8: Histogram of all the crops position in the x-axis.

To calculate the standard deviation (σ), equation (3.2) is used, and the mean is calculated with equation (3.3) [27].

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2} \tag{3.2}$$

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{3.3}$$

where:

- μ is the mean value of the dataset
- n is the size of the
- x_i is the x value of crop i

With equation (3.2) and equation (3.3) the mean and standard deviation of the crops position in the x axis is calculated to be 586.971 and 128.016 respectively.

Next the distance between the crops is characterized.

To find the distance between the the crops, a distance between all the crops in each image is calculated and the the $n-1$ smallest distances is taken, where n is the number of crops in an image. This is because a distance further away than the $n-1$ closest is not distance to the closest crop but the crops further away.

From this method the distribution of the distance to the next crop is found and a histogram can be seen on figure 3.9.

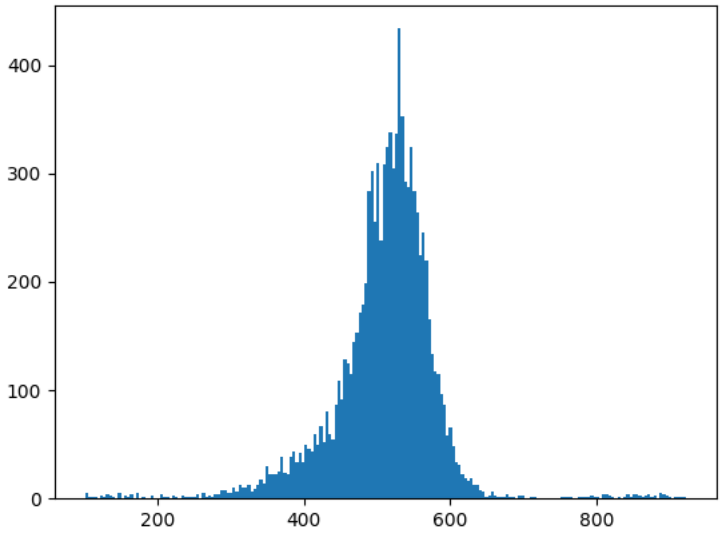


Figure 3.9: Histogram showing the distribution of the distance between two crops in a image

As the distribution of the distance between the crops looks like a normal distribution equation (3.3) and equation (3.2) can be used again to find the mean and standard deviation. With these equations the mean is found to be 509.512 and the standard deviation to be 72.082.

With the distance between the crops and the crops horizontal position characterized, the amount of crops in an image, and how far from the edge they typically appear needs to be characterized.

As there will be no crops on a generated image, another metric than distance to crops is needed to place the first crop. For this the distance to the top of the image is used. This distance to the top of the image is measured perpendicular to the top and the distribution of distance can be seen on figure 3.10

Figure 3.9 shows a histogram of the crops distance to the top of the image.

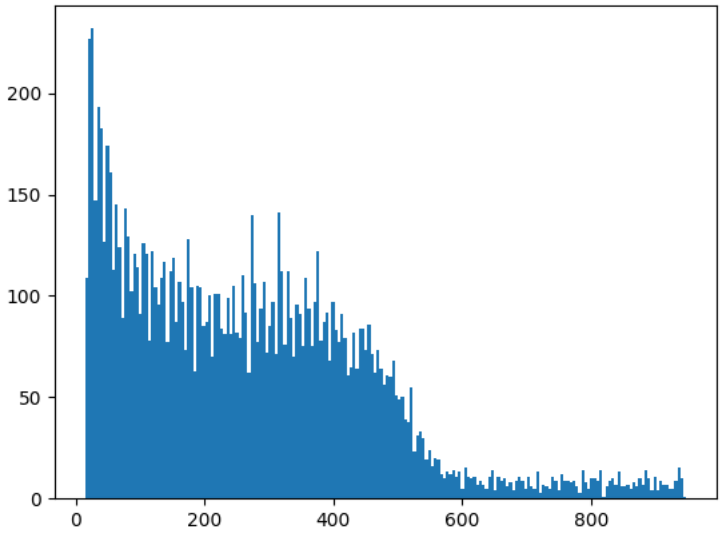


Figure 3.10: Histogram showing the distribution of closet crops distance to the top of the image.

As this distribution does not look like a symmetric normal distribution but more like half of a normal distribution, the normal distribution parameters will be found with equation (3.2) but where μ is the smallest value.

For the position of the weed compared to the crops, the distance between the weed and the crop is used. To get this distance the centroid of the crop and the centroid of the weed is used as the points to measure between. By getting the distance of all the weed to all the crops in a image the distribution on the histogram in figure 3.11 can be found.

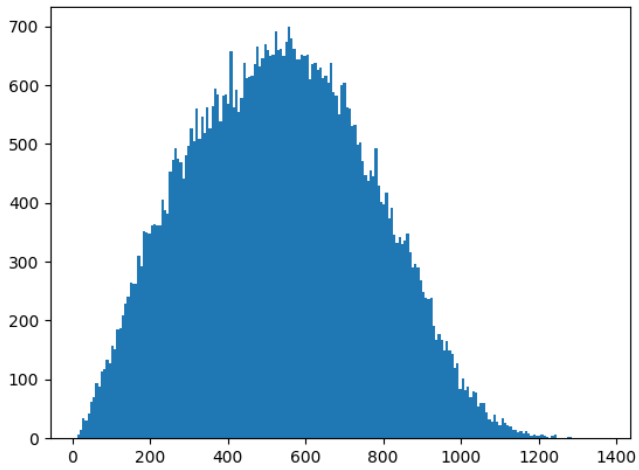
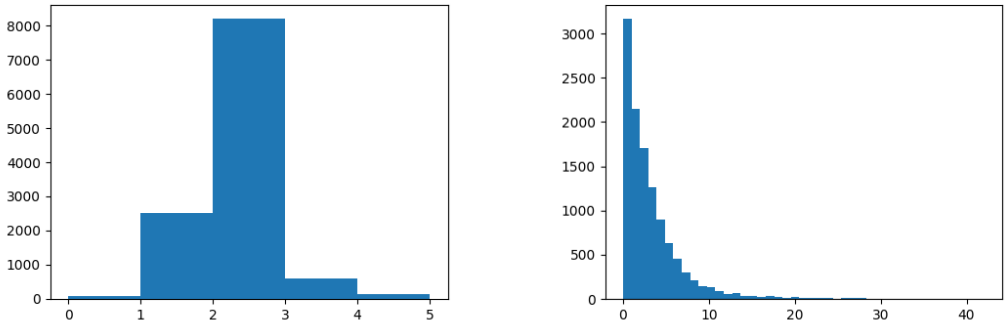


Figure 3.11: Histogram showing the distribution of weed distance to the crops.

Lastly the number of weeds and crops in the image will be analysed by counting the number of contours in the weed masks and the crop masks respectively. The distribution of the dataset can me sen on figure 3.12, where equation (3.2) and equation (3.3) will be used on 3.12a and equation (3.2) with μ set to the smallest value, will be used on figure 3.12b.



(a) Histogram showing crops.

(b) Histogram showing weed.

Figure 3.12: Distributions of crops and weed per image.

All the results from analysing of the real dataset can be seen on table 3.1.

With a description of the crop and weed placements, construction of the synthetic datasets can begin.

	Mean	Standard deviation
crops horizontal position	586.971	128.016
Crops distance between each other	509.512	72.082
Crops distance to top of image	14.000	319.126
Weed distance from crops	535.795	229.102
Weed count distribution	0.000	4.852
Crop count distribution	1.842	0.577

Table 3.1: Mean and standard deviation of crops and weed position from real dataset

3.2.2 Placement of plant on image

After finding a general description of the distribution of crops and weeds in the dataset, construction of a synthetic dataset can begin. First, the amount of crops and weed to place in the image is randomly chosen to be within the range found in the original dataset. After this the location of the first crop is randomly picked based on the normal distributions found in the original dataset. The X position of each crop is always sampled like this, however the Y position is based on the distance from the top for the first crop placed, and on the distance to the previous crop for any further crops. The process of finding the position of the crops is illustrated in figure 3.13 where the position of the crops can be seen on figure 3.7.

The placement of the weeds are randomly placed based on the distance from crops as observed in the dataset, as they are not intentionally planted.

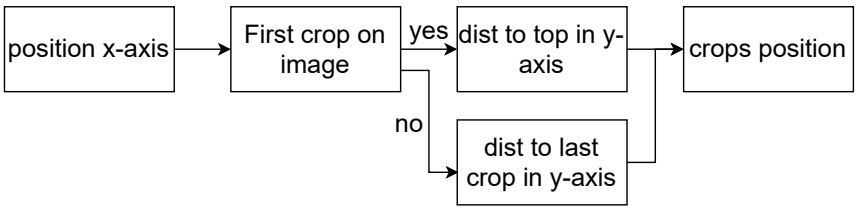


Figure 3.13: Diagram showing how the crops position is found

After the location of the plants and weeds have been determined, the image has to be constructed. The previously extracted weed and crop images are randomly picked and placed on the coordinates. The weeds are placed first, as they are smaller and can be partially or fully occluded by large crops.

To achieve more variation in output images, the background images are randomly flipped, effectively augmenting the amount of backgrounds the network sees. Additionally, as the dataset entirely consists of images taken from directly above the plants, the plants can also be randomly rotated to augment the selection

of plants.

After placing the weeds and crops there will be heavy artifacts around the edge of the placed objects, requiring blending to properly merge them with the background.

3.2.3 Blending

As can be seen on figure 3.14 an object that is pasted into an image will generate artifacts at the border between the object and background. To remove these artifacts blending is used. To reduce the sharp tranasition along the edge of the plat, Gaussian blur is used to blend the border with the background. Furthermore Poisson local illumination is used to better blend the object into the scene. the effect of both Gaussian blur and Poisson local illumination can be seen on figure 3.15.

Figure 3.14: Example of crop pasted on a background without blending.

(a) Gaussian blur applied to the edge. (no Poisson)

(b) Poisson local illumination change applied. (No Gaussian)

Figure 3.15: Examples of the the types of blending used on the pasted plant.

Gaussian blur

Blurring also called smoothing is done by applying a kernel on the image. To apply this blurring equation (3.4) is used [28].

$$g(x,y) = \sum_{i=-R}^{R} \sum_{j=-R}^{R} h(i,j) \cdot f(x-i, y-j) \tag{3.4}$$

Where:

- g is the blurred image
- x and y is the ordinate where the filter is applied
- h represent the applied kernel
- f represent the image
- R is the radius of the kernel

The kernel used in the blurring is a Gaussian kernel. The reason for this choice is the closer a pixel is to the center pixel the higher importance it has. To get this kernel equation (3.5) is used [28].

$$h(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.5}$$

where:

- σ is the variance
- x and y is the distance from the center

As it is only the edge that needs to be blurred, the contours of all the masks are drawn on a binary image which will be used as a mask to indicate the part of the image that will need to be blurred. The contour width was decided to be 5 by trying different sizes and found this to be optimal visually.

The edge is then blurred by creating an image where the kernel has been applied and then using the created mask to paste the blurred part onto the original image. The optimal kernel size used to blur the image was found to be 5×5, which was again decided by visually inspection a blurred edge on a generated image. The final result of the Gaussian blur can be seen on figure 3.15a

Poisson local illumination

To make the pasted plant better blend in to the scene Poisson local illumination changes was used, as proposed in the paper *"Poisson Image Editing"* [29]. The advantage of using local illumination is to lessen specular reflections and also in case of under-exposure correct it by change the dynamic range over the designated area evenly. To do this equation (3.6) is used to get the gradient field over the area of the mask object.

$$v = \alpha^{\beta} |\nabla f^*|^{-\beta} \nabla f^* \tag{3.6}$$

Where:

- v is the transform vector field
- α is 0.2
- β is 0.2
- f^* represent the area of the original/background image

By solving the Poisson equation in equation (3.7) with the boundary conditions defined in equation (3.8), the illumination will be changed within the boundary area.

$$\Delta f = \text{div } v \tag{3.7}$$

$$f|_{\delta\Omega} = f^*|_{\delta\Omega} \tag{3.8}$$

Where:

- f^* and v is the same a previous
- f represent the mask region of the image
- $\delta\Omega$ is the boundary around the mask

The end result of using this local illumination change can be seen on figure 3.15b where it can be seen the illumination of the crop is more even over the hole image.

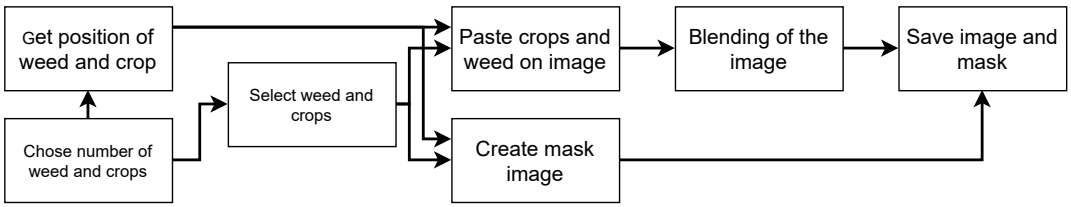
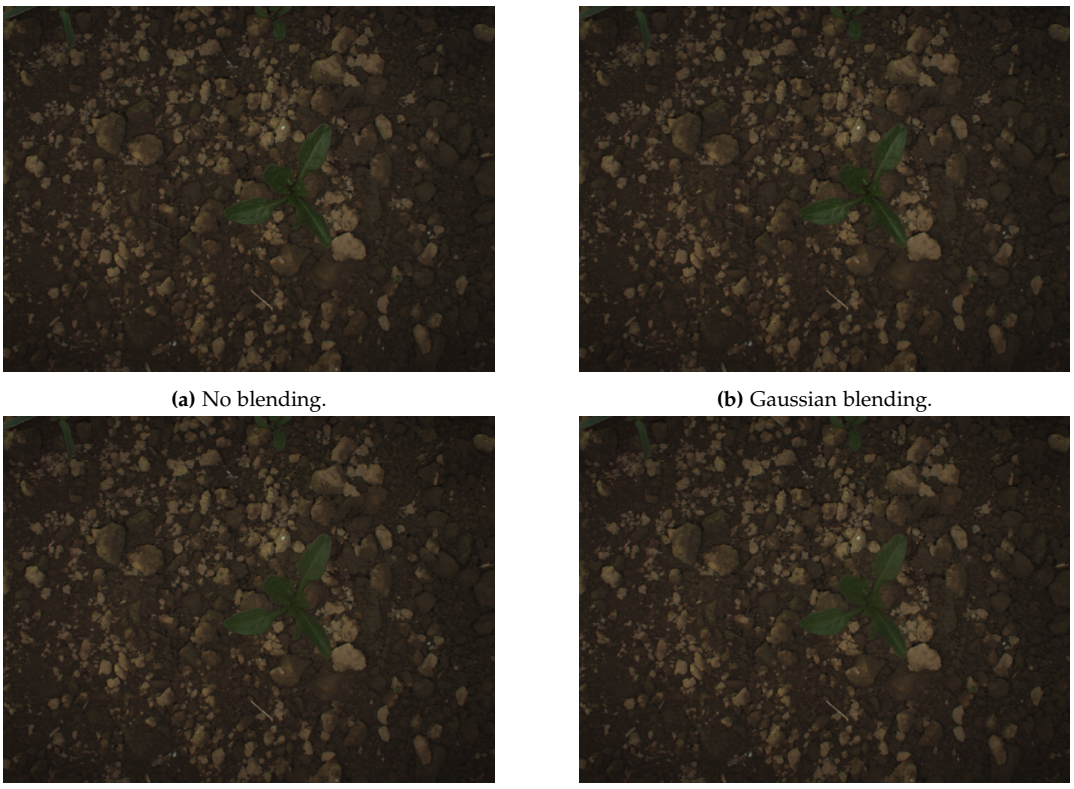


Figure 3.16: Diagram of the steps involved in placing the plants in a image

3.2.4 output of synthetic image generator

On figure 3.16 a diagram of the full process of generating a synthetic image can be seen.

The end result of the generated images can be seen on 3.17, where the same example image is shown with all the different combinations of blending or no blending is used. A larger version of the images can also be seen in appendix A with the corresponding mask. 6 other examples of generated image can also be seen in appendix B



(a) No blending.

(b) Gaussian blending.

(c) Poisson blending.

(d) Poisson and Gaussian blending

Figure 3.17: Examples of the generated images with different types of blending.

3.3 Neural network theory

Before discussing the architecture of the chosen model for testing the data, this section will cover some of the basic theory in neural networks.

3.3.1 Neurons

An Artificial Neural Network (ANN) is computer model meant to simulate the human brains ability to learn behaviour that it wasn't explicitly programmed for. The base unit of a neural network is called a neuron. A neuron is a basic component, consisting of an arbitrary amount of weighted inputs, an activation function that processes these inputs, a bias and an output from the neuron [30]. An illustration of the neuron can be seen on figure 3.18 and it is described by equation (3.9)

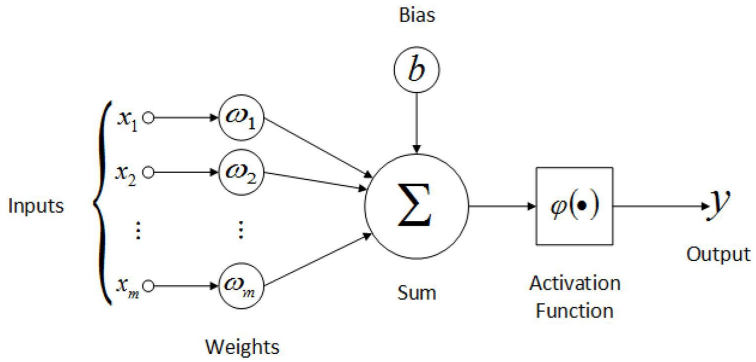


Figure 3.18: A model of a single artificial neuron [31]

$$y = \phi((\sum_{1}^{n} x_i * \omega_i) + b) \tag{3.9}$$

3.3.2 Constructing a network

While a single neuron is the simplest possible network, it can't do much work on its own. As such a neural network is constructed. This is done by arranging multiple neurons next to each other forming a layer, and having several layers one after another as seen on figure 3.19

This example network has an input layer with three inputs, and an output layer with a single neuron. In between are two so-called hidden layers with 4 neurons each. These are called hidden layers, as they don't directly interact with the users info. Input data is passed to the input layer, passed through the layers of the model, and a result is then presented on the output layer. The hidden layers, their type size and connectivity, affects how complex behaviour the model can learn.

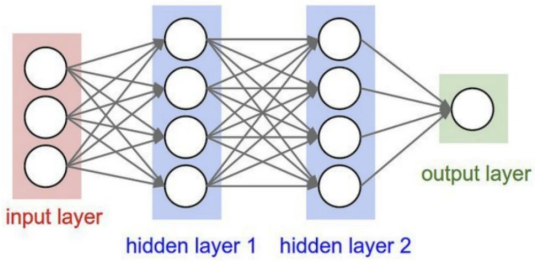
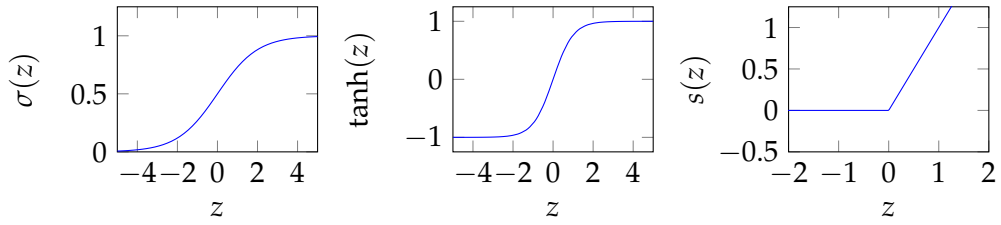


Figure 3.19: An illustration of a simple neural network[31]

In the example all layers are fully connected. Every neuron output is connected to all neurons in the next layer. This isn't the only way to construct a network. Layer types such as convolutional layers for example rely on a particular way of connecting between layers.

Examples of common activation functions are Sigmoid, Tanh and Rectified Linear Unit (ReLU), where ReLU currently is the most common, as it is quicker to train. The different activation functions can be seen on figure 3.20



(a) Sigmoid **(b)** Tanh **(c)** ReLU

Figure 3.20: Sigmoid Tanh and ReLU activation functions

3.3.3 Initializing the network

In order to train the network, optimal connection weights have to be found. This is done through the training process.

First the network is initialized. This means all weights in the network are given an initial value that can later be improved. While an initial thought might be to simply initialize all the weights randomly following normal distribution this will cause issues as you propagate deeper through the network, as the standard deviation would grow through the layer, causing very large activations that could cause issues while training. Either by causing large values that round to infinity, or by causing very negative values that would get clamped in the activation function. Instead an initialization method called Xavier initialization or Glorot uniform initialization is used. [32]

With the network initialized, training is then done by an optimizer using a form of gradient decent to efficiently find and optimal combination of weights that minimize the loss function used while training.

3.3.4 Loss function

A loss function for a network is any function that evaluates the output of the network, compares it to a desired state, and then produces a loss value describing how far off the network is from the desired result. During training the goal is to minimize the loss function [27].

Examples of loss functions could be the mean square error (MSE) loss function which is a typical loss function for regression problems, that is where the goal is to predict a value like the price of a house. MSE is simply the mean of the squared errors in the training samples (equation (3.10)).

$$loss = \frac{1}{n}\sum_{i}(y_i - \hat{y}_i)^2 \tag{3.10}$$

where:

- y_i is the true value for sample i
- $\hat{y}_i$ is the predicted value for sample i
- n is the amount of samples

Another kind of tasks, classification, benefits from using a different loss function like cross entropy loss, seen in equation (3.11)

$$loss = -\sum_{i} y_i * \ln(\hat{y}_i) \tag{3.11}$$

Assuming that a prediction will only ever correspond to one class, this simplifies to simplifies to equation (3.12) as y_i will be 0 for all other classes.

$$loss = -\ln(\hat{y}_i) \tag{3.12}$$

The reason categorical cross entropy is preferred, is that it grows really quickly when the model prediction far from the desired output, which would be a 1.0, or 100% confidence in the desired class. This means the derivative of the loss for bad predictions is really large, which is important for the method used to train the network, gradient decent.

3.3.5 Gradient Decent

In order to train the network, the weights of the network have to be adjusted. As mentioned, the goal while training is to reduce the loss function, however the only way this can be changed is by adjusting the weights in the network. Gradient decent is the method used to efficiently calculate which adjustments to make to the different weights of the network, based on their current state and the loss function [27]. In fact, gradient decent calculates the gradient of the loss function, with respect to weights in the in the network, and uses this gradient to adjust the weight, and this is where the name is from. The algorithm uses the gradient of the loss function to descend towards a minima of the loss function, hopefully the global minima.

An illustration of gradient decent can be seen on figure 3.21

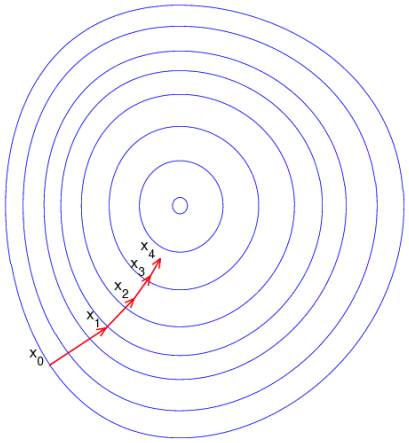


Figure 3.21: Illustration of Gradient Decent[33]

In order to control how fast the model learns, and avoid making too large changes, the algorithm works by calculating the gradient for the loss function with respect to a weight, it then multiplies the weight, the gradient, and a learning rate. The learning rate is a tune-able parameter, global to the model and is typically in the range of 0.1 to 0.001, and this learning rate is what moderates the size of change that can be made.

With this explained, it's now clear why a loss function that has a large gradient when the model performs poorly is desired, as this will speed up the initial steps of learning, and slow down when approaching a minima.

3.3.6 Backpropagation

As mentioned, gradient decent uses the gradient of the loss function with respect to the weights of the network, however it is not immediately obvious how this is

calculated. In order to calculate this gradient, backpropagation is used [27].

Backpropagation is the process of feeding back the error through the network after a forward pass of the network. It solves the complicated issue of calculating the gradient of the loss function with respect to any weight in the network, such that the weights can be updated in an efficient manner. This is also where the name stems from. As the error of the network is observed at the last layer in the network, this error has to be propagated backwards by recursively applying the chain rule.

3.3.7 Optimizers

While gradient decent offers a method for adjusting the weights in a neural network in an efficient manner, it suffers from the fact that it only updates the weights of the networks after having found the true gradient. The gradient calculated with all data. This is memory in-efficient, and while providing better changes to the weights, it does so less frequently than other optimizers.

A variation of gradient decent, Stochastic Gradient Decent (SGD), instead calculates the weight changes based on only a few samples. In fact, following the strict definition of SGD it only uses one sample, however using a batch of samples is more common, and are known as mini-batches [27]. Using only a subset of the data both means it scales better for larger models and results in a lot more model updates over the same amount of training compared to gradient decent. While the updates aren't as optimal as they'd be using pure gradient decent, the higher frequency of adjustments mean the model trains faster than with gradient decent.

SGD works well and provides good results for finding a minima without overshooting, however it learns very slowly if there's areas with low gradient like saddle-points or plateaus. It can also produce noisy changes as it only samples a subset of the data, however a method known as momentum can help smooth this out. The general idea is to also consider the direction of previous updates, and weight changes that don't follow this direction less. Like a ball rolling down a hill increasing momentum and becoming less sensitive to changes in direction.

Adam [34] is a modern and popular optimizer that improves on previous optimizers by combining the benefits of SGD with momentum, and another optimizer called RMSProp [35] which uses the moving average of the squares of the gradient, and normalizes this to work well on gradients with large differences in magnitude.

Adam, like a lot of optimizers after SGD is an adaptive learning rate optimizer. This means it calculates and maintains separate learning rates for each parameter, allowing the update to take larger steps for weights that are far from an optimal value, while also making smaller adjustments for weights that are close to a desired value.

3.4 U-net segmentation model

As mentioned in section 2.4 the chosen netowrk to train on our data is U-net. The network expands on the work done by Ciresan et al. [36] where a sliding window of large patches were fed into a network to accurately predict pixel class and location.
U-net instead uses a mostly symmetrical Convolutional Neural Network (CNN) network, consisting of a contractive path (feature extractor) and a expanding path (predictive path).
The contractive path consists of two 3x3 convolutional layers, each with a ReLU activation layer, followed by a 2x2 max-pooling layer. These blocks are then repeated with decreasing resolution and increasing filters.
The expansive path is similar, however instead of the max pooling layers it uses the transpose convolution to upscale the layers, while reducing the amount of filters. It is also after this up-scaling that the center portion of the corresponding layer from the contractive path is copied and concatenated with the result of the up-scaling. This is then followed by two 3x3 convolutional layers again with ReLU activation. This is repeated until the output layer is reached, where a final convolutional layer is used to produce c channels where each channel represents a prediction for each class, and c is the number of classes.

Moving away from a sliding window approach avoids the redundancy of training over a lot of the same pixels due to overlapping patches as in the sliding-window approach, as well as speeds up the class prediction. It still only predicts classes for pixels where it has full context, as in it uses un-padded convolutions and relies on the input images to be prepared with padding to allow predictions over the entire image. The network also incorporates features from multiple layers, allowing both high resolution filters and higher complexity filters to work together to achieve high localization accuracy and accurate predictions.

Due to this structure, the resulting network is shaped like the letter U, hence the name.

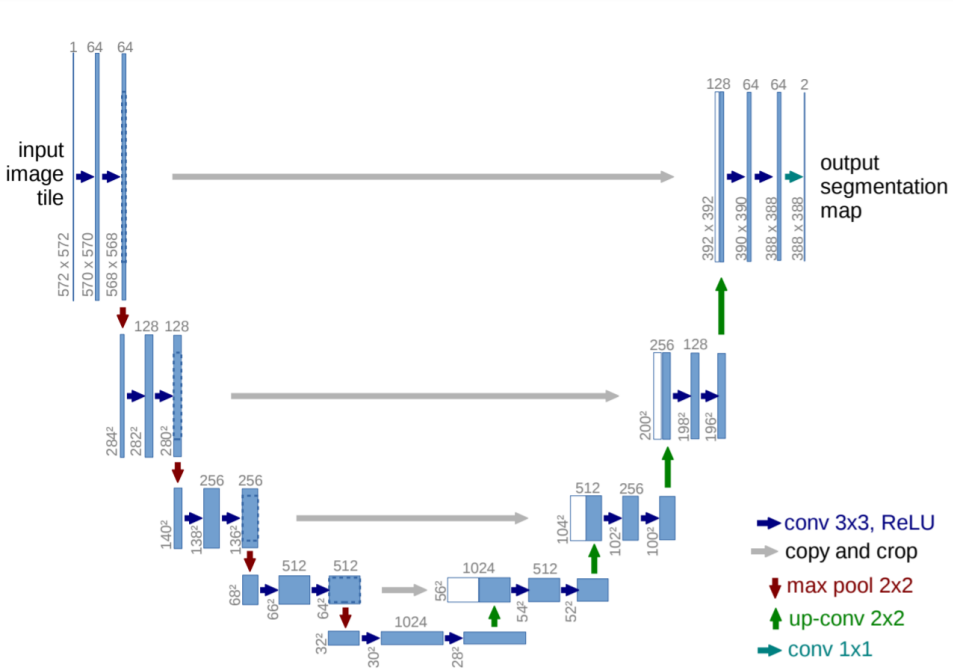


Figure 3.22: The U-net model as described in the paper[24]

3.4.1 Input image tiling

As seen on figure 3.22, the model has an input size of 572x572, and produces predictions on the center 388x388 pixels in the image. This is because the network uses unpadded convolutions to only predict on pixels with full context. To provide predictions over an arbitrarily large image, a tiling approach is used as seen on figure 3.23.

To find the required amount of reflection padding the missing amount of pixels required to fit an integer amount of tiles is calculated. Then the difference in input and output size is added all around the image. This is done using reflection padding as described in the paper. After padding the image with this padding, the image can be tiled. Because we are interested in predicting for all of the input image, the padded image tiled into overlapping tiles with a step-size equal to the output size, or 388x388px. This causes the tiles to be overlapping, as the tile size is the input size of the network, or 572x572. And example of this tiling can be seen on figure 3.24

The U-net model is implemented as described in the original paper[24], except that the input layer is 3 channels to accept color images, and the output is three channels to predict our three classes.

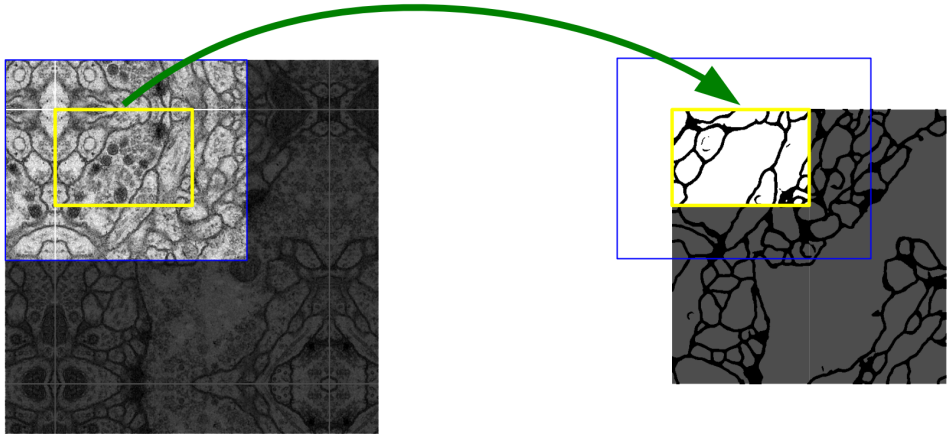


Figure 3.23: The tiling strategy used in U-net. Reflecting the edges of the input image and tiling the image to fit through u-net(fig2 from u-net paper[24])

Figure 3.24: Example of calculating padding to tile an image. The white area is the padding required to have an integer amount of tiles. Grey area is the padding required to account for the discrepancy in input and output size. The image is tiled in step sizes of 388, producing overlapping input tiles of 572x572 (blue) and tiled output tiles of 388x388 (red)

Chapter 4

Testing and results

In order to test how the different methods of blending the plants into the images performed, a selection of datasets were generated as described in section 3.2.

ID	Dataset	Description
1	Real	A conventionally annotated dataset
2	Synthetic no blending	Synthetic dataset with no blending operations applied to the plants
3	Synthetic Gaussian blending	Synthetic dataset with Gaussian blending operations applied to the plant edges
4	Synthetic Poisson blending	Synthetic dataset with Poisson blending operations applied to the plants
5	Synthetic combined blending	Synthetic dataset with both Poisson blending and Gaussian blending operations applied to the plant and plant edges.
6	Test set	A conventionally annotated dataset

Table 4.1: Table of the different datasets used

The different datasets generated can be seen on table 4.1. All datasets, except the test set, consist of 1250 images. This was chosen to have 1000 images to train on, and 250 images for an 80/20 train/validation split. As suggested by roboflow [37]. The two conventionally annotated datasets, the real dataset and the test dataset, are both subsets of the sugar beats dataset, and do not share any images.

While the chosen dataset has plenty more images than 1000 images, and while generating several thousand images for the synthetic datasets wasn't an issue, the lower amount of images was chosen due to the training time of the model. With

1000 images with a resolution of 1296 x 966px and the tiling method described in section 3.4.1, the model ends up training on 12000 572 x 572px tiles which takes 9-10h training on a Google Colaboratory instance with either a Tesla P100 or V100.

The U-net model was trained on each dataset for 40 epochs with sample weighing as described in section 2.3, and then evaluated on the test set. The metric used to evaluate the performance on the models was *weighted mean Intersection over Union*

4.1 Weighted mean Intersection over Union

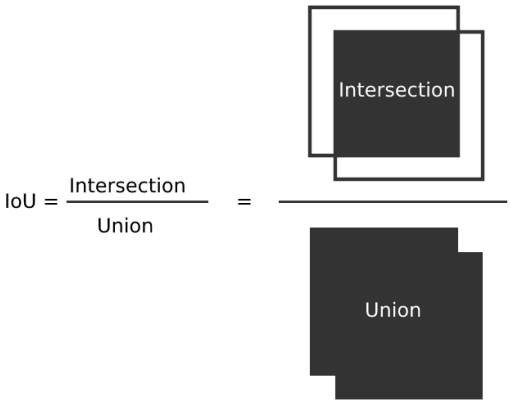


Figure 4.1: Geometric illustration of how IoU calculates a score

Mean intersection over union (mIoU) is a commonly used metric for both object detectors and image segmentation models. It produces a score between 0 and 1 where 1 is perfect. The score is fairly intuitive too as a perfect prediction will overlap the ground truth perfectly. Only when the two areas perfectly cover each other will the intersection equal the union. An illustration of IoU can be seen on figure 4.1. MIoU is the mean of the IoUs calculated for each class.

Lastly, as the dataset is highly imbalanced, we use weighted mIoU, which applies the class weighing when calculating the score. This metric is then used to evaluate the performance of the different datasets.

4.2 Results

The resulting weighted mIoU score of the different data-sets can be seen in table 4.2

As expected, the synthetic dataset utilizing both blending methods performed the best out of the synthetic data. What wasn't expected was that all synthetic datasets performed better than the manually annotated source data-set, as can be seen on table 4.2. Additionally looking at the the predictions on a test image

Dataset	wmIoU
Real	0.634
No Blending	0.662
Gaussian	0.694
Poisson	0.701
Gaussian + Poisson	0.767

Table 4.2: Results of evaluating the models trained on the different datasets

for each model, it was clear that models trained on real data couldn't find the difference between weeds and crops. Two examples can be seen on figure 4.2 and 4.3. Examples for all models can be found in appendix C.

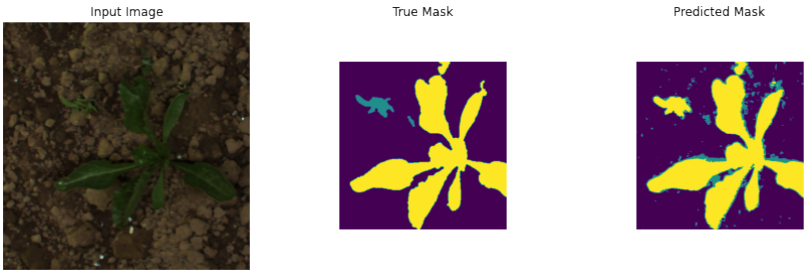


Figure 4.2: Prediction with model trained on dataset 1 (real)

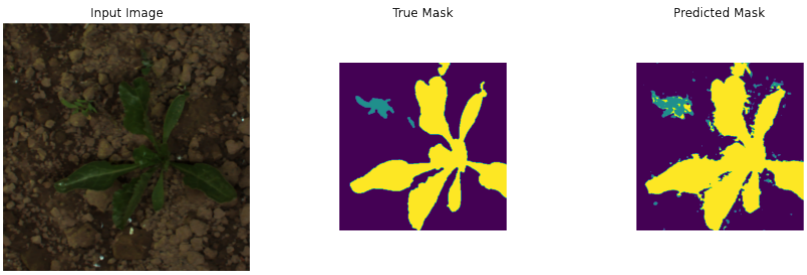


Figure 4.3: Prediction with model trained on dataset 5 (Poisson + Gaussian)

After seeing the models inability to tell the difference between weed and crops when trained on the real dataset further tests were run to see what effect the use of the real dataset had on the training results. Datasets were prepared based on dataset 5 (see table 4.1 with 10, 30 and 50% of the data replaced with samples from dataset 1. After training the model on these datasets, it was clear that adding data from the conventionally annotated dataset made the model unable to distinguish weed and crops, with the effect becoming worse the more that was added. The results of the mixed datasets can be seen on table 4.3

Dataset	wmIoU
Mix 10%	0.735
Mix 30%	0.729
Mix 50%	0.701

Table 4.3: Results of evaluating the models trained on the different mixed datasets

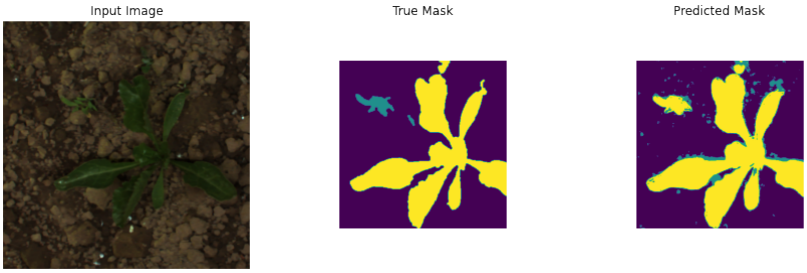


Figure 4.4: With just 10% of the dataset being from dataset 1 already shows poor weed prediction

These results suggest that the problem found in section 3.1.3 is sufficiently widespread that it impacts the model performance to such a degree that it can't distinguish weeds from crops. If too many examples of weed are annotated as crops, it makes sense that the model will predict weeds as crops.

While the results are better than the model trained purely on real data, they are still worse than the model trained on the purely synthetic data-set. Furthermore, while the models trained on mixed data have a fairly high mIoU, they still fail to predict any weeds just like the model trained purely on the real data, as shown on figure 4.5 and 4.6

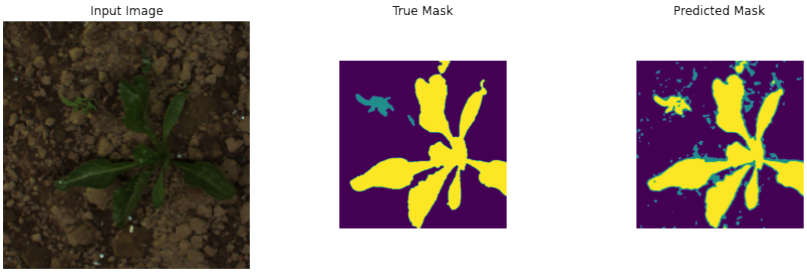


Figure 4.5: Prediction with model trained on mixed data-set with 10% from the real data-set

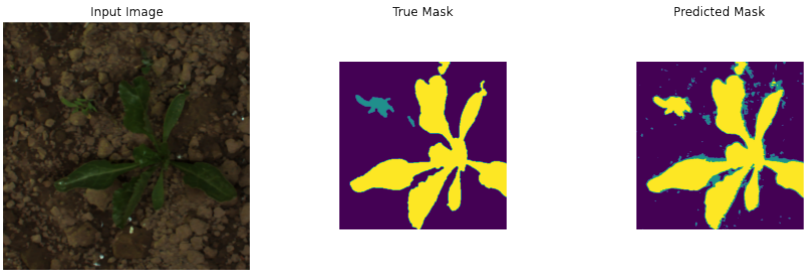


Figure 4.6: Prediction with model trained on mixed data-set with 50% from the real data-set

Chapter 5

Discussion

The results found in section 4.2 shows that the simple Cut, paste and learn method can be used to generate high quality datasets for machine learning in agriculture. One particular benefit of this approach that was discovered is the much smaller need for hand annotated data allowing for more in depth inspection of the annotations. While the general idea is to greatly reduce the need for manual annotations, this also turned out to be a benefit in the quality of the annotations that could be achieved with much less effort. This increased quality turned out to be vital for the model to perform properly and predict correct classes.

While the results suggest that the hand annotated data has too many flaws as described in section 3.1.3, this hasn't been specifically checked, as it would involve a detailed inspection of the entire dataset, or at least a large portion of it.

Other explanations for the improved performance of the synthetic dataset could be that sampling from the normal distribution fitted to the histograms produce better images for training. Selection of crops for pasting onto the images also didn't account for the size or growth-stage of the crops. This more random distribution of small and large crops could also make the models generalize better.

In general, it was also found that the datasets had a very imbalanced class representation, necessitating very different class weights while training. While this worked to generate results, this does mean the datasets had much fewer examples of the smallest class, weeds, making it hard to learn.

Chapter 6

Conclusion

In looking to reduce the amount of work required to train a neural network in agriculture, a modified Cut, paste and learn [11] method was implemented to generate synthetic datasets. In order to test these, a 3-class U-net network was implemented, and different datasets with different blending techniques were generated.

When the different datasets were used to train the U-net network, it was found that the synthetic datasets all outperformed the conventionally annotated real dataset, and as such is considered a success.
The results showed that using this method for generating new datasets could significantly reduce the need for manual annotation of the dataset. This much smaller need for manual annotation also meant that the quality of the annotations could be higher.

Chapter 7

Future work

With the synthetic datasets performing as well as they did, new types of synthetic datasets could be tested. Of particular interest could be the generation of datasets that don't follow the structure found in the normal datasets. A dataset could be generated that drastically over-represented weeds for example, to help counter the large class imbalance in the dataset.

It'd also be interesting to compare the performance of synthetic dataset generated with some of the other mentioned synthetic methods (section 3.2) on the same model, as well as testing our datasets on different segmentation models.

Looking in to the suspicious around poor annotation quality in the source dataset from section 3.1.3 would also be interesting, to confirm or deny the suspicion of poor annotation quality.

Acronyms

ANN Artificial Neural Network. 24, *Glossary:* Artificial Neural Network

CNN Convolutional Neural Network. 29, *Glossary:* Convolutional Neural Network

FCN Fully Convolutional Network. *Glossary:* Fully Convolutional Network

mIoU mean Intersection over Union. 33, 35, *Glossary:* mean Intersection over Union

ReLU Rectified Linear Unit. 25, 29, *Glossary:* Rectified Linear Unit

SDR Structured Domain Randomization. 2, *Glossary:* Structured Domain Randomization

SGD Stochastic Gradient Decent. 28, *Glossary:* Stochastic Gradient Decent

SYN-MTGAN SYNthetic Multi-Task Generative Adversarial Networks. 4, *Glossary:* Synthetic Multi-Task Generative Adversarial Networks

Glossary

Artificial Neural Network is computer model meant to simulate the human brains ability to learn behaviour that it wasn't explicitly programmed. 24

Convolutional Neural Network is neural network that utilizes convolution layers in their design, implementing image convolution for feature extraction. 29

mean Intersection over Union Is a metric used to evaluate how well a method predicts an object based on how well they overlap. It is the intersection of the ground truth and the prediction over the union of these same areas.. 33

Rectified Linear Unit is a popular activation function in machine learning as it's been found fast to train while giving good results.. 25

Stochastic Gradient Decent is an optimizer for weight updates that approximates gradient decent. 28

Structured Domain Randomization is a method for generating synthetic datasets that generalize well when used to train deep learning models, without having to achieve photo realism. 2

Synthetic Multi-Task Generative Adversarial Networks is a neural network developed to generate annotated test data for training object detectors. 4

Bibliography

[1] K. G. Liakos, P. Busato, D. Moshou, S. Pearson, and D. Bochtis, "Machine learning in agriculture: A review," *Sensors*, vol. 18, no. 8, 2018, ISSN: 1424-8220. DOI: 10.3390/s18082674. [Online]. Available: https://www.mdpi.com/1424-8220/18/8/2674.

[2] P. Mishra, G. Polder, and N. Vilfan, "Close range spectral imaging for disease detection in plants using autonomous platforms: A review on recent studies," *Current Robotics Reports*, vol. 1, no. 2, pp. 43–48, Jun. 2020, ISSN: 2662-4087. DOI: 10.1007/s43154-020-00004-7. [Online]. Available: https://doi.org/10.1007/s43154-020-00004-7.

[3] K. Osorio, A. Puerto, C. Pedraza, D. Jamaica, and L. Rodríguez, "A deep learning approach for weed detection in lettuce crops using multispectral images," *AgriEngineering*, vol. 2, no. 3, pp. 471–488, 2020, ISSN: 2624-7402. DOI: 10.3390/agriengineering2030032. [Online]. Available: https://www.mdpi.com/2624-7402/2/3/32.

[4] P. Christiansen, L. N. Nielsen, K. A. Steen, R. N. Jørgensen, and H. Karstoft, "Deepanomaly: Combining background subtraction and deep learning for detecting obstacles and anomalies in an agricultural field," *Sensors*, vol. 16, no. 11, 2016, ISSN: 1424-8220. DOI: 10.3390/s16111904. [Online]. Available: https://www.mdpi.com/1424-8220/16/11/1904.

[5] K. Hu, Z. Wang, G. Coleman, A. Bender, T. Yao, S. Zeng, D. Song, A. Schumann, and M. Walsh, *Deep learning techniques for in-crop weed identification: A review*, Mar. 2021.

[6] V. R. Saripalli, G. Avinash, D. Pati, M. Potter, and C. W. Anderson, "Ai assisted annotator using reinforcement learning," 2019. DOI: 10.1007/s42979-020-00356-z. eprint: arXiv:1910.02052.

[7] K.-K. Maninis, S. Caelles, J. Pont-Tuset, and L. V. Gool, *Deep extreme cut: From extreme points to object segmentation*, 2017. eprint: arXiv:1711.09081.

[8] K. S. Henriksen, M. S. Lynge, M. D. B. Jeppesen, M. M. J. Allahham, I. A. Nikolov, J. B. Haurum, and T. B. Moeslund, "Generating synthetic point clouds of sewer networks: An initial investigation," in *Augmented Reality, Virtual Reality, and Computer Graphics*, L. T. De Paolis and P. Bourdot, Eds., Cham: Springer International Publishing, 2020, pp. 364–373, ISBN: 978-3-030-58468-9.

[9] A. Prakash, S. Boochoon, M. Brophy, D. Acuna, E. Cameracci, G. State, O. Shapira, and S. Birchfield, *Structured domain randomization: Bridging the reality gap by context-aware synthetic data*, accessed: 2020-02-28, 2018. arXiv: 1810.10093 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1810.10093.

[10] A. Ghorbani, V. Natarajan, D. Coz, and Y. Liu, *Dermgan: Synthetic generation of clinical skin images with pathology*, 2019. eprint: arXiv:1911.08716.

[11] D. Dwibedi, I. Misra, and M. Hebert, *Cut, paste and learn: Surprisingly easy synthesis for instance detection*, 2017. arXiv: 1708.01642 [cs.CV].

[12] M. Di Cicco, C. Potena, G. Grisetti, and A. Pretto, "Automatic model based dataset generation for fast and accurate crop and weeds detection," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[13] Y. Lin, K. Suzuki, H. Takeda, and K. Nakamura, "Generating synthetic training data for object detection using multi-task generative adversarial networks," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. V-2-2020, pp. 443–449, 2020. DOI: 10.5194/isprs-annals-V-2-2020-443-2020. [Online]. Available: https://www.isprs-ann-photogramm-remote-sens-spatial-inf-sci.net/V-2-2020/443/2020/.

[14] Y. Lu and S. Young, "A survey of public datasets for computer vision tasks in precision agriculture," *Computers and Electronics in Agriculture*, vol. 178, p. 105760, 2020, ISSN: 0168-1699. DOI: https://doi.org/10.1016/j.compag.2020.105760. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169920312709.

[15] A. Kamilaris and F. X. Prenafeta-Boldú, "Deep learning in agriculture: A survey," *Computers and Electronics in Agriculture*, vol. 147, pp. 70–90, 2018, ISSN: 0168-1699. DOI: https://doi.org/10.1016/j.compag.2018.02.016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169917308803.

[16] (). "Weed-ai: A repository of weed images in crops," Precision Weed Control Group and Sydney Informatics Hub, the University of Sydney, [Online]. Available: https://weed-ai.sydney.edu.au/ (visited on 2021-07-02).

[17] N. Chebrolu, P. Lottes, A. Schaefer, W. Winterhalter, W. Burgard, and C. Stachniss, "Agricultural robot dataset for plant classification, localization and mapping on sugar beet fields," *The International Journal of Robotics Research*, 2017. DOI: 10.1177/0278364917720510.

[18] I. Sa, M. Popović, R. Khanna, Z. Chen, P. Lottes, F. Liebisch, J. Nieto, C. Stachniss, A. Walter, and R. Siegwart, "Weedmap: A large-scale semantic weed mapping framework using aerial multispectral imaging and deep neural network for precision farming," *Remote Sensing*, vol. 10, no. 9, 2018, ISSN: 2072-4292. DOI: 10.3390/rs10091423. [Online]. Available: https://www.mdpi.com/2072-4292/10/9/1423.

[19] D. Eigen and R. Fergus, *Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture*, 2014. eprint: arXiv:1411.4734.

[20] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. van der Maaten, *Exploring the limits of weakly supervised pretraining*, 2018. eprint: arXiv:1805.00932.

[21] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie, *Class-balanced loss based on effective number of samples*, 2019. eprint: arXiv:1901.05555.

[22] K. He, G. Gkioxari, P. Dollár, and R. Girshick, *Mask r-cnn*, 2017. eprint: arXiv:1703.06870.

[23] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, *Rethinking atrous convolution for semantic image segmentation*, 2017. eprint: arXiv:1706.05587.

[24] O. Ronneberger, P. Fischer, and T. Brox, *U-net: Convolutional networks for biomedical image segmentation*, 2015. eprint: arXiv:1505.04597.

[25] S. Suzuki and K. be, "Topological structural analysis of digitized binary images by border following," *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32–46, 1985, ISSN: 0734-189X. DOI: https://doi.org/10.1016/0734-189X(85)90016-7. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0734189X85900167.

[26] T. B. Moeslund, *Introduction to Video and Image Processing: Building Real Systems and Applications*, eng, ser. Undergraduate topics in computer science. London: Springer London, Limited, 2012, ISBN: 1447125029.

[27] C. Bishop, *Pattern recognition and machine learning.* eng, Berlin, 2006.

[28] R. Szeliski, *Computer Vision: Algorithms and Applications*, eng, ser. Texts in computer science. London: Springer London, Limited, 2010, ISBN: 9781848829343.

[29] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," *ACM Trans. Graph.*, vol. 22, no. 3, 313–318, Jul. 2003, ISSN: 0730-0301. DOI: 10.1145/882262.882269. [Online]. Available: https://doi-org.zorac.aub.aau.dk/10.1145/882262.882269.

[30] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[31] M. T. Gaudio, G. Coppola, L. Zangari, S. Curcio, S. Greco, and S. Chakraborty, "Artificial intelligence-based optimization of industrial membrane processes," *Earth Systems and Environment*, vol. 5, no. 2, pp. 385–398, Jun. 2021, ISSN: 2509-9434. DOI: 10.1007/s41748-021-00220-x. [Online]. Available: https://doi.org/10.1007/s41748-021-00220-x.

[32] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Y. W. Teh and M. Titterington, Eds., ser. Proceedings of Machine Learning Research, vol. 9, Chia Laguna Resort, Sardinia, Italy: PMLR, May 2010, pp. 249–256. [Online]. Available: http://proceedings.mlr.press/v9/glorot10a.html.

[33] Olegalexandrov. (2004). "Gradient descent." File: Gradient descent.png, [Online]. Available: https://commons.wikimedia.org/wiki/File:Gradient_descent.png.

[34] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017. arXiv: 1412.6980 [cs.LG].

[35] V. Bushaev, *Understanding rmsprop - faster neural network learning*, Sep. 2018. [Online]. Available: https://towardsdatascience.com/understanding-rmsprop-faster-neural-network-learning-62e116fcf29a.

[36] D. C. Ciresan, A. Giusti, L. M. Gambardella, and J. schmidhuber, *Deep neural networks segment neuronal membranes in electron microscopy images*, 2012. [Online]. Available: https://people.idsia.ch/~juergen/nips2012.pdf.

[37] J. Solawetz, *The train, validation, test split and why you need it*, Dec. 2020. [Online]. Available: https://blog.roboflow.com/train-test-split/.

Appendix A

Synthetic generated dataset.

Figure A.1: Synthetic image with no blending

Figure A.2: Synthetic image with Gaussian blur blending.

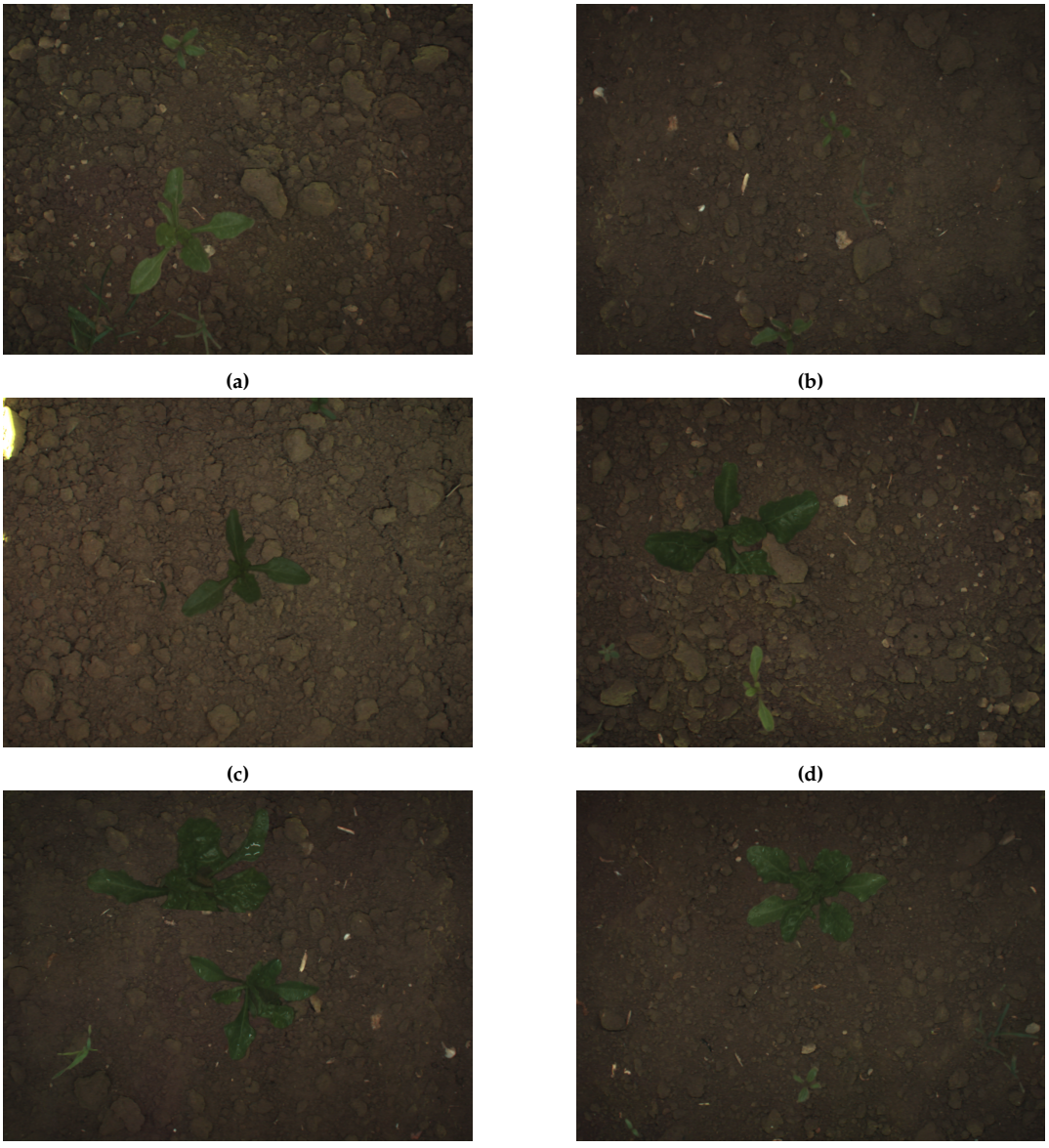
Figure A.3: Synthetic image with Poisson local illumination blending

Figure A.4: Synthetic images with Poisson and Gaussian blending

Figure A.5: Mask of the synthetic images.

Appendix B

Examples of generated images.



(a) **(b)**

(c) **(d)**

(e) **(f)**

Figure B.1: Six examples of synthetic generated images.

Appendix C

Model predictions

C.1 Initial datasets

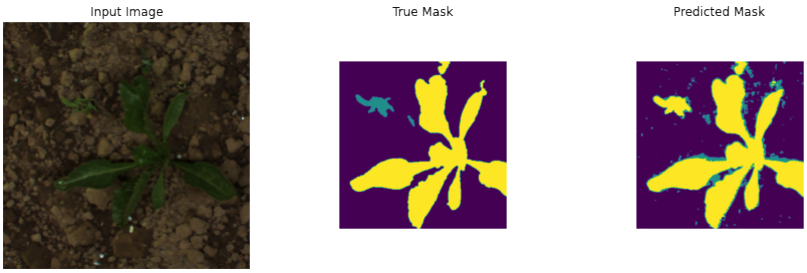


Figure C.1: Prediction with model trained on real data-set

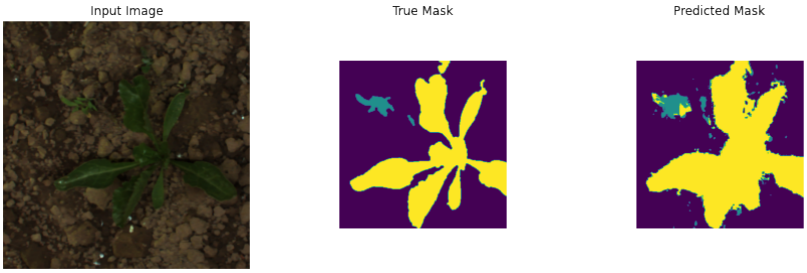


Figure C.2: Prediction with model trained on synthetic data-set with no blending

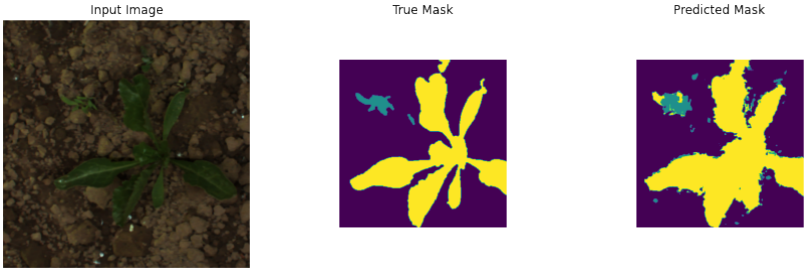


Figure C.3: Prediction with model trained on synthetic data-set with Gaussian blending

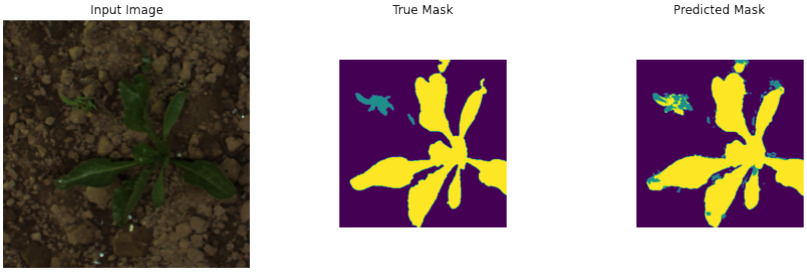


Figure C.4: Prediction with model trained on synthetic data-set with Poisson blending

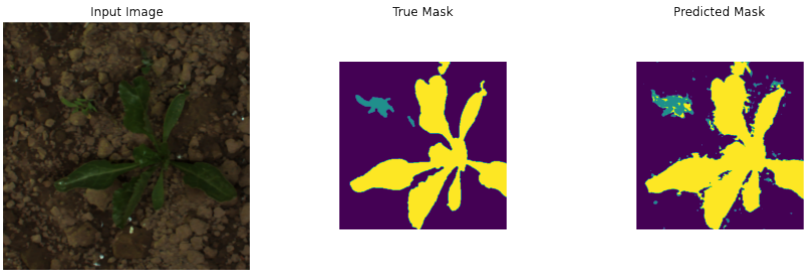


Figure C.5: Prediction with model trained on synthetic data-set with Poisson and Gaussian blending

C.2 Mixed datasets

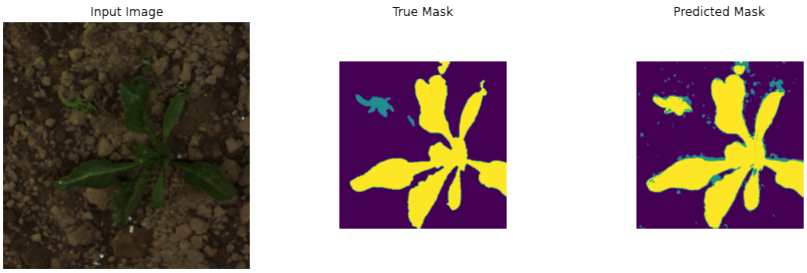


Figure C.6: Prediction with model trained on synthetic data-set with Poisson and Gaussian blending with 10% if the dataset being replaced with samples from the real dataset

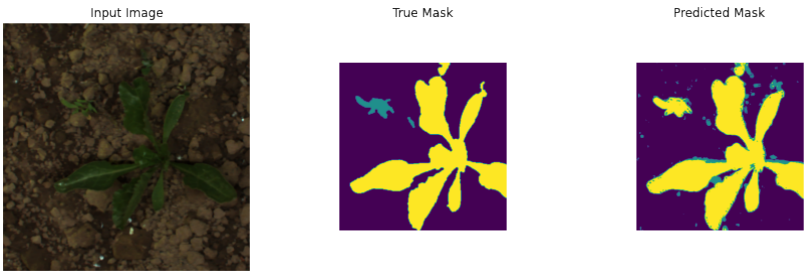


Figure C.7: Prediction with model trained on synthetic data-set with Poisson and Gaussian blending with 30% if the dataset being replaced with samples from the real dataset

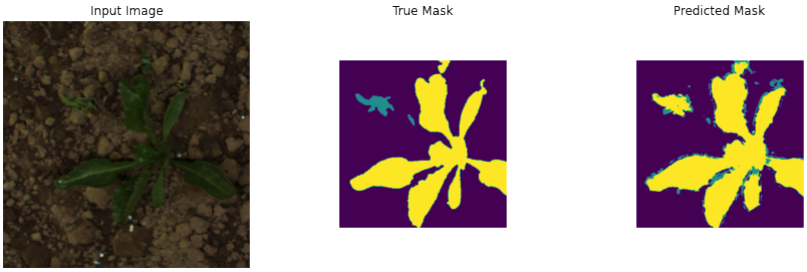


Figure C.8: Prediction with model trained on synthetic data-set with Poisson and Gaussian blending with 50% if the dataset being replaced with samples from the real dataset