HAMM-LIPPSTADT
UNIVERSITY OF APPLIED SCIENCES

Generating Synthetic Images for Plant Disease in Sugar Beet using Augmentation Techniques

Department Lippstadt 2

Bachelor Thesis
for optainment of the degree of
Bachelor of Engineering

submitted by

Olaniyi Bayonle Alao

Electronic Engineering
Mat.Nr.: 2180561
olaniyi-bayonle.alao@stud.hshl.de

April 30, 2022

First supervisor:	Prof. Dr. Stefan Henkler
Second supervisor:	Prof. Dr. Achim Rettberg
Second supervisor:	Dipl.-Wirt.-inf. Kristian Rother

Abstract

This bachelor thesis aims to create synthetic leaf disease image datasets for sugar beet from available public datasets using image augmentation techniques. The lack of public image datasets for training disease detection models in sugar beet motivated this research. In order to create the synthetic dataset, we used publicly available sugar beet plant seedlings dataset and diseased image dataset of a plant that shares the same genus with a known disease in sugar beet from the PlantVillage dataset. We need to concatenate the extracted diseases in other plants to leaf areas of healthy seedling images of sugar beet plants. First, we will build a pipeline that can automatically overlay diseased areas on healthy areas of leaf images. Then, we present and evaluate the resulting images of the experiment by training convolutional neural network-based models. The experiment showed promising results to form a basis for further research on synthetic image generation for training disease detection models. Finally, the thesis was concluded with a comprehensive overview of the chapters and discussed limitations encountered during the experiment with recommendations for future works.

Contents

1 Introduction

Currently, in the 21st century, computers have become ubiquitous in our everyday lives. Previously mundane tasks like switching on/off light switches, vacuuming, counting objects, and accurately predicting sales, to safety-critical tasks like driving vehicles and trains are currently performed efficiently by computers. Efforts are being made to extend the capabilities of computer systems to perform more intelligent tasks like disease detection, speech-to-text transcribing, anomaly detections that humans otherwise perform. However, the challenging part is transferring human knowledge to computers to perform such analysis and inference tasks. The approach humans use in learning is by transferring knowledge of a system to another related system. For example, humans can effortlessly differentiate between a cat and a dog in an image. This ability to differentiate is possible because specific characteristics are present in dogs but absent in cats, irrespective of their breeds. However, manually programming computers to make such differentiation will prove impractical or possibly amount to several lines of code. Likewise, it is unrealistic to manually program a computer to detect diseases in an image or make driving decisions in the case of autonomous vehicles.

These shortcomings for computers have motivated researchers to trial the use of machine learning techniques to allow computers to analyse and interpret objects in images or videos. An exciting use case in agriculture is training models to monitor and detect plant leaf diseases or nutrient deficiencies through image data. The use of computers to precisely monitor and give respective treatment to particular plants in plantations to get increased average yields and reduce operational farming costs is known as precision farming.

Traditionally, disease detection techniques used to involve using naked eyes to visually inspect the leaves and branches of plants for any sign of infection. However, this plant health monitoring and disease detection method are time-consuming. For example, it is labour tasking and financially draining to visually inspect plant leaves spanning several hectares of land. Thanks to the advancements in the technological sector, researchers have shown the possibilities of plant health monitoring using machine learning techniques to classify plant leaves captured with RGB (Red Green Blue) cameras and hyperspectral sensors as healthy or unhealthy. Unfortunately, one major challenge in using these technologies is the unavailability of sufficient datasets for model training.

1.1 Goals

Plant diseases cause significant hindrances to the sustainability and profitability of agriculture [HRR+14]. In order to prevent the losses that come with diseases in plants, different methods have been developed to monitor and diagnose these diseases before they lead to a significant infestation. One of such methods is the use of established knowledge in

molecular biology and immunology to precisely identify the causal agents of diseases in plants and develop treatments for eradicating them before they become catastrophic to the plants. However, such an approach requires experts in the disease domain to cut down the plants and perform tests to identify the specific disease affecting the plants. Moreover, this disease detection technique is invasive; it is likewise tedious and expensive [WSW17].

These shortcomings in the traditional disease detection motivated research in the direction of non-invasive plant disease detections using data from sensors like RGB, hyperspectral imaging and spectroscopy with machine learning algorithms [MHS16, AAM+16, ZHD+19]. The research results on the use of collected real-time data from plants with machine learning techniques showed promising results that are more accurate in plant disease detection and classification than domain experts in such fields, thereby potentially minimising costs for farmers [BPVM20]. Standard algorithms used for plant disease detection and classification tasks are linear regression, logistic regression, random forest, clustering, Gaussian models, decision trees (DT), Naïve Bayes (NB), K-nearest neighbours (KNN), and support vector machines (SVM) among others.

Despite the success of plant disease detection, using sensor data with classical machine learning (ML) algorithms was time-consuming due to manual feature extractions and not being robust enough for use on datasets containing variations from the original training dataset [Fer18]. Nevertheless, the understandings from ML laid a solid foundation for current research in the direction of deep neural networks for automating the identification of disease from the data captured from sensors to allow precise plant treatment in a field. Furthermore, advancements in computer vision and artificial intelligence have led to solutions that were found to accurately solve complex decision-making tasks like yield forecasting [LCT+17], plant recognition [GLS17], and nutrition deficiency [YKU+20] in precision farming.

Consequently, deep neural networks require a large amount of image data with appropriate diversity representing different conditions that can likely occur to create a high-quality model for the trained tasks. However, capturing all the variability of conditions possible is a cumbersome task, hence the need to augment the existing dataset using computer vision and deep learning techniques to create robust image datasets for plant disease detection tasks in sugar beet. Furthermore, there are no publicly accessible datasets for sugar beet disease detection, and acquiring these datasets requires several months and professionals to ensure the quality and diversity needed [BPVM20, JCB+20].

This thesis proposes the approaches below to solve the lack of a publicly available dataset for disease detection in sugar beet plants:

- Use image augmentation techniques to overlay diseased areas in a plant on top of healthy areas in another plant. However, there should be a common disease occurrence in both plants.
- Validate the generated images by training convolutional neural network (CNN) based disease detection models using the synthetic images and the original healthy sugar beet images.

The results from creating these synthetic datasets will create an avenue for transferring knowledge in this domain to create more datasets for other plants, hence creating more than enough datasets for model training.

1.2 Sketch of Approach

Our approach to artificial dataset creation is based on segmenting the leaf areas of the diseased and healthy plants from their background in the images and then concatenating the extracted diseased areas intelligently on the healthy leaf areas. First, we propose segmenting foreground and background leaf areas by converting the images to their respective HSV (Hue Saturation Value) values to find the leaves' upper and lower bound values. We then propose using thresholding algorithms to create binary masks of the leaf areas and the background and finally use boolean operations to create the new image. Furthermore, we propose bringing the infected areas of the diseased leaf to the foreground and then masking out healthy areas to become the background. Figure 1.1 showcases a visual representation of the proposed artificial dataset creation pipeline. The extracted disease areas in the left side of figure 1.1 are concatenated with segmented healthy leaf areas of sugar beet in the right side of the figure to create synthetic disease images.

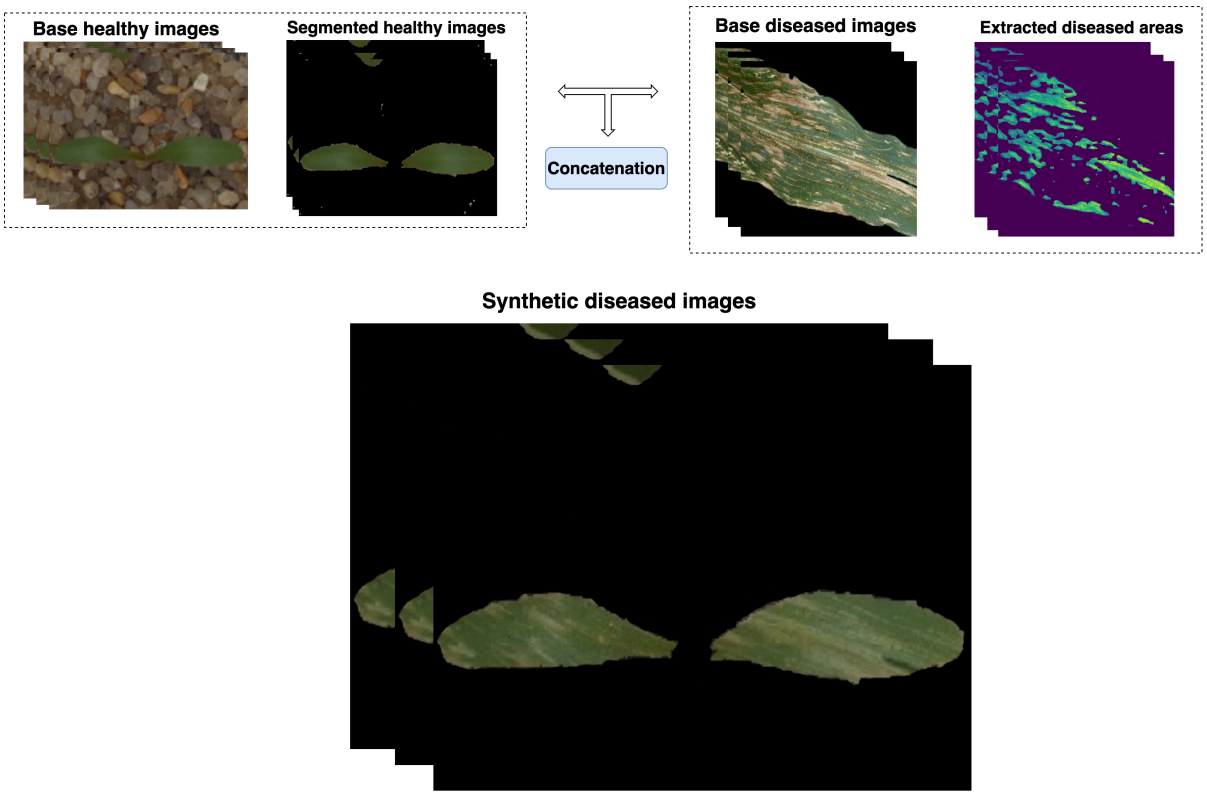

Figure 1.1: Overview of proposed appraoch for generating artificial dataset.

With the resulting images from the previously proposed approach, we suggest potentially automating the process of creating more randomly unique datasets using generative adversarial neural networks (GAN). On a high level, GANs work by using two neural networks competing against each other to create synthetic datasets. First, the neural generator network creates new synthetic image data, which are then fed to the discriminator neural network to ascertain how close the generated image is to the original images.

1.3 Organisation of Paper

In section 1.1, we described plant disease monitoring and disease detection and the goals of the thesis to address the lack of sufficient datasets for model training. Then, in section 1.2 we gave an overview of the sketch of approaches for implementing the proposed approach to solving the lack of public dataset in sugar beet for disease detection tasks. Next, chapter 2 explains the relevant fundamentals necessary for the experiment performed in this research. Then, in chapter 3 we highlight related works to the thesis. Next, based on the knowledge from the related works and fundamentals described in previous chapters, chapter 4 presents the prerequisites and a high-level overview of the approach design and its implementation. Next, chapter 5 presents and discusses the results of the generated synthetic image datasets. Finally, chapter 6 concludes the research work by summarising the thesis and giving recommendations for future work.

2 Fundamentals

This chapter explains the relevant fundamentals necessary for the experiment performed in this research.

2.1 Digital Image Formation

A digital image is a two-dimensional pixels array(matrix) composed of picture elements. Each pixel represents the numeric representation of the brightness of the corresponding picture element in the x and y-axis. Furthermore, a digital colour image stores colour information for each pixel in three intensity components (channels). A pixel in a colour image represents the value of the combination of three colours Red, Green, and Blue (RGB). Standard formats of storing the colour information are RGB (red, green, blue channels), HSV or HSB (hue, saturation, brightness) and YCbCr (where Y is the luma component, Cb and Cr are the blues and red components relative to the green component). For example, an RGB image supports 16,777,216 different colours because each channel occupies a maximum of 8-bit colour variations (i.e. between 0 and 255). A digital image can be converted from one colour space to another for easy manipulation purposes.

2.2 Grayscaling

A grayscale image is a black and white image with shades of grey in-between. There are different techniques for converting a colour image to grayscale.

[Pra07] developed a color to grayscale image technique using weighted combination of RGB channels in equation 2.2. Also, a colour image can be grayscaled using equation 2.1 by taking the maximum of the RGB channels [AR05] or taking the average of the three channels [Jac07] as denoted in equation 2.3.

$$G_{value} = \max(R, G, B) \tag{2.1}$$

$$G_{luminance} = 0.3R + 0.59G + 0.11B \tag{2.2}$$

$$G_{intensity} = \frac{R + G + B}{3} \tag{2.3}$$

Using grayscale images for extracting descriptors is computationally cheaper and simplifies the algorithm compared to colour images due to working with only one channel (luminance information) as compared to three channels in colour images [KC12]. For example, a 24-bit

RGB image becomes 8-bit when converted to grayscale. However, the above mentioned and other colours to grayscale algorithms have different time computation, and appropriate use case scenarios in image recognition tasks [Ĉad08]. According to Kanan and Cottrell [KC12], equation 2.2 is good for object and face recognition tasks.

2.3 Thresholding

Thresholding is a popular and computationally straightforward technique in computer vision for foreground and background segmentation of images [LCP90]. Thresholding works by converting grayscale images to binary images according to the pixels' threshold value [NA19]. This means that each pixel's brightness/grey value in the interested objects/regions of the input image must be known. The grey values of objects and background can be obtained by drawing a grayscale image histogram. As the operator of the threshold algorithm goes through every pixel in the image, if the value of the respective pixel is greater than the threshold, it is set to the maximum pixel value (255, i.e. white). On the other hand, if the pixel value is less than the threshold, the value is set to (0, i.e. black). Hence the final image becomes a white and black image showing a segmented region of interest. For example, figure 2.1 shows a thresholded image, where the original image (a) has been transformed into a grayscale image (b), and (c) is a thresholded image where all pixels above the value of 150 are set to 255 and pixels below the threshold value are set to 0.

Figure 2.1: (a) Original image (b) Grayscale image (c) Thresholded image.

Sezgin and Sankur [SS04] categorises thresholding operators based on the parameters used for manipulating grayscale images into the following six categories.

- histogram shape-based methods, in this method, the peaks, valleys and curvatures of the smoothed histogram are analysed.
- clustering-based methods, where the grey-level samples are clustered in two parts as background and foreground objects, or alternately are modelled as a mixture of two Gaussians.
- entropy-based methods result in algorithms that use the entropy of the foreground and background regions, the cross-entropy between the original and binarised image.

- object attribute-based methods search a measure of similarity between the grey-level and the binarised images, such as fuzzy shape similarity edge coincidence.
- the spatial methods use higher-order probability distribution and correlation between pixels.
- dynamic/local methods adapt the threshold value on each pixel to the local image characteristics.

Furthermore, according to Guruprasad [Gur20], the main forms of thresholding methods are:

- Threshold Binary

$$K(x,y) = \begin{cases} maxVal & if \quad L(x,y) > threshold \\ \\ 0 & otherwise \end{cases}$$

- Threshold Binary, Inverted

$$K(x,y) = \begin{cases} 0 & if \quad L(x,y) > threshold \\ \\ maxVal & otherwise \end{cases}$$

- Threshold to zero

$$K(x,y) = \begin{cases} 0 & if \quad L(x,y) > threshold \\ \\ L(x,y) & otherwise \end{cases}$$

- Threshold to zero, Inverted

$$K(x,y) = \begin{cases} L(x,y) & if \quad L(x,y) > threshold \\ \\ 0 & otherwise \end{cases}$$

- Truncate

$$K(x,y) = \begin{cases} threshold & if \quad L(x,y) > threshold \\ \\ L(x,y) & otherwise \end{cases}$$

  Where L(x,y) is the source image matrix, K(x,y) is the output image matrix, and maxVal is the maximum pixel value.

Otsu's Thresholding

The Otsu thresholding method [Ots79] is named after Nobuyuki Otsu. It is a common thresholding technique for automatically finding the optimal threshold value that gives the

best separation between an object and its background for every pixel in the image. Otsu's thresholding is a clustering-based adaptive thresholding method compared to uniform thresholding forms where the threshold value is manually calculated. Using Otsu's method, the optimal threshold t_{opt} is calculated using equation 2.4 by maximising the between-class variance (for class C_0 and C_1, i.e. background and objects) of the two classes.

$$\sigma_B^2(t_{opt}) = \max_{1 \leq k \leq L_{max}} \sigma_B^2(t) \tag{2.4}$$

The covariance(i.e. between-class variance) is given by

$$\sigma_B^2(t) = \frac{[\mu_T \omega(t) - \mu(t)]^2}{\omega(t)\,[1 - \omega(t)]} \tag{2.5}$$

Where t is the threshold, μ_T is the total mean level of the image, L_{max} is the maximum grey level, $\mu(t)$ and $\omega(t)$ is the zero- and first-order cumulative moments of the normalised histogram up to the threshold level t.

2.4 Gaussian filtering

In images, smoothing techniques are used to blur out noise and edges that occur due to sampling and processing in the source camera by sliding a kernel of low-pass filter over the image [NA19]. Noises can occur in images during acquisition, sampling, transmission or processing in the source camera. Examples of noise in images are light fluctuations, sensor noise (camera noise), and quantisation effects. Gaussian smoothing filter is one of the types of image smoothing methods commonly used in computer vision for edge detection due to its optimal performance [Bas02].

The Gaussian smoothing operator in 2-D has the form:

$$G_{(x,y)} = \frac{1}{2\pi\sigma^2} e^{-} \frac{x^2 + y^2}{2\sigma^2} \tag{2.6}$$

Where σ is the standard deviation of the distribution and x, y are the values of each pixel in the x, and y coordinate [HCH07]. The Gaussian function in equation 2.6 gives the coefficient of the Gaussian kernel, which is slid over an image when smoothing.

2.5 Active contours

Contour is a continuous curve drawn around the edges of detected object/s in an image. The curve traces the exterior outline around objects to be detected at an arbitrary starting point in either a clockwise or an anti-clockwise direction and stops on the boundary of the object [WM18]. Drawing contours around objects in an image allows for getting the measurements and angle of orientation of the detected obeject/s, which can be used for other image processing operations like translation and rotation.

2.6 Machine Learning

Machine learning is a subset of artificial intelligence that uses statistical algorithms to enable computers to learn independently without being explicitly programmed to do so. Machine learning algorithms can be grouped as supervised, unsupervised or reinforcement learning depending on how the algorithm learns the relationships between the input and the expected output data [GBC16]. On the other hand, artificial intelligence is a method that trains computers to solve tasks by mimicking the way the human brain works through the use of mathematical and computer science methods.

Furthermore, deep learning is an evolution of machine learning, enabling computers to train themselves in making accurate decisions on tasks through neural networks.

The accuracy of models trained by previously mentioned methods depends heavily on the training dataset's quality to represent the problem statement adequately. Convolutional neural networks (CNN), deconvolutional neural networks (DNN) and recurrent neural networks (RNN) are common types of deep learning algorithms. This thesis will focus on generative adversarial networks that use CNN and DNN to generate artificial datasets.

2.7 Generative Adversarial Networks (GANs)

According to [GPAM+14], "[generative adversarial network is] a new framework for estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G; the training procedure for G is to maximise the probability of D making a mistake." This approach has been successful in generating artificial images in computer vision tasks [DCSF15].

To learn a generator distribution p_g over data data x, according to [MO14] the generator builds a mapping function from a prior noise distribution $p_z(z)$ to data space as $G(z;\theta_g)$. And the discriminator, $D(x;\theta_d)$, outputs a single scalar representing the probability that x came from training data rather than p_g. G and D are both trained simultaneously: we adjust parameters for G to minimize $log(1 - D(G(z))$ and adjust parameters for D to minimize $logD(X)$, as if they are following the two-player min-max game with value function $V(G, D)$.

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[logD(x)] + E_{z\sim p_z(z)}[log(1 - D(G(z)))] \tag{2.7}$$

Extensions of the GANs architecture includes deep convolutional GANS (DCGANs), conditional GANs (CGAN), information maximising GANs (InfoGANs), amongst others [RMC15, CDH+16, MO14]. Figure 2.2 shows an overview of the GANs architecture. The architecture begins with a random noise which is inputed to generator for generating fake images. The synthetically generated images are then fed into the discriminator network which also takes original images which fakes ones are generated from as input. The discriminator network determines how close the generated image is to the original image

Different GAN-based architectures have been developed due to the instability in training

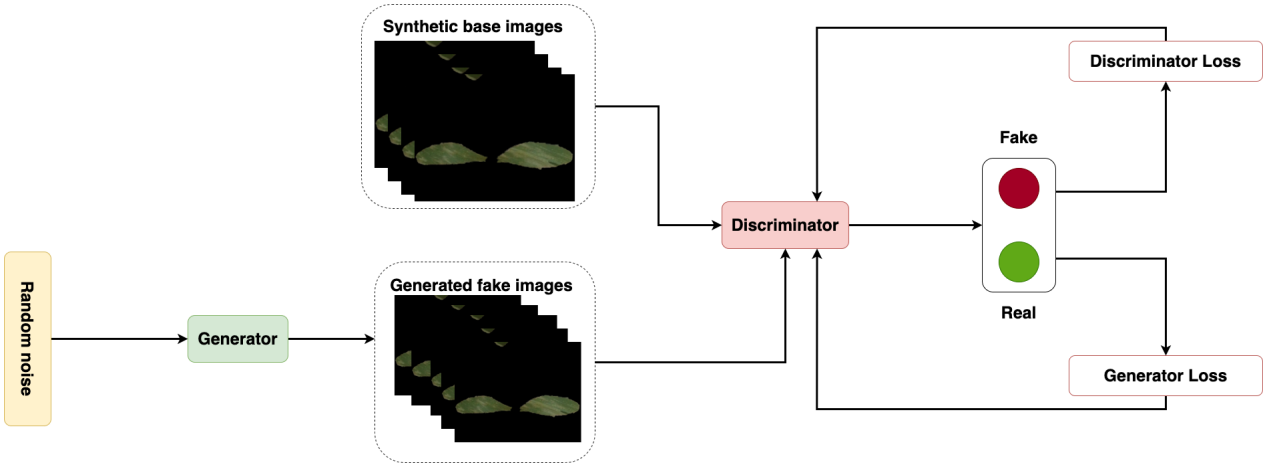

Figure 2.2: Generative adversarial network (GAN) architecture.

models using standard GANs, which often result in ridiculous output by the generator network [RMC15]. Therefore, researchers have proposed using other GANs based architectures to create synthetic image datasets to augment currently limited available datasets for training purposes. For example, Gandhi et al. [GNYP18] used deep convolutional GAN (DCGAN [RMC15]) for creating synthetic images to augment the limited number of local images available for training disease detection models for plants in India. Likewise, Barth et al. [BHVH20] in their experiment used Cycle GAN for creating 10 500 synthetic images of bell pepper for their segmentation task, Zhu et al. [ZAK+18] created synthetic images from the CVPPP 2017 LSC plant dataset using conditional GAN (CGAN).

3 Related Work

Precision agriculture uses innovative technologies to optimise agricultural production through the use of site-specific management. Particularly in crop production, this aims at crop-specific nutrient deficiencies and disease monitoring, application of fertiliser or pesticides, prediction of yields, automated counting of crops, remote/automatic control of agricultural vehicles like drones and tractors. The underlining principle behind precision farming is to use innovative technologies to economically optimise agricultural production as well as reduce harmful outputs into the environment by applying only the needed amount of fertiliser or pesticides needed during a plantation season [Aue01]. Applying the specific amount of resources needed by individual plants will help reduce the impact of chemical by-products or hazardous materials ending up in the environment and economically improve farmers' financial costs. Furthermore, in order to achieve environmental protection of waters and soils, there is a need for reduction in the use of fertilisers and pesticides in crop production [OVSC05]. However, the most significant limiting factors in precision farming is the interpretation of collected data from sensors [OW17] and adequate datasets. Nevertheless, with the current advancement of data manipulation techniques in data science, there is a possibility of addressing the data interpretation shortcomings to enable a more successful implementation of precision farming.

Based on the research results conducted in our previous work [Ala22], there are many approaches to disease detection in plants, especially in sugar beet. However, there are not enough datasets around prompting researchers to use image augmentation techniques like random cropping and flipping to increase the data size and create randomness in the datasets. Likewise, some of the datasets used in the paper reviewed were not publicly accessible. Hence, the approach in this research is to synthetically generate a dataset using a combination of techniques in image processing. While there are no approaches that precisely match the one described in this thesis, the idea of synthetic image generation for model training already exists. This chapter reviews different approaches documented in research works in synthetically generating datasets for varying reasons like data privacy and increasing datasets.

Ward et al. [WMH18]proposed a method of meeting the large amount of data required to train models using state-of-the-art machine learning approaches for leaf instance segmentation tasks. The proposed framework aims to use the synthetically generated images of the *Arabidopsis* plant in their research to augment existing natural plant datasets. Their synthetic image dataset generation begins with manually tracing a randomly chosen *Arabidopsis* leaf image in Blender to produce a 3D mesh of an inspiration leaf based on the original leaf image. Then, more 3D leaf images are created by randomly scaling to model leaves of different shapes and sizes. Next, the synthetically generated plant leaves are circularly arranged close to each other at a similar height. Finally, random background, camera and lightning are added to the generated plant leaves, which are then rendered

together as a 2D synthetic image and corresponding segmentation mask. Their proposed framework's success is evidenced by its average accuracy of 90% in leaf segmentation.

Da Silva et al. [dSBG+19] proposed three methods for generating synthetic defoliation images used in training CNN-based models to estimate soybean leaf defoliation. Their proposed methods remove leaf-belonging pixels in different ways to simulate actual defoliation in a pre-processed image and returns a new defoliated leaf image along with its level of defoliation. The first method simulates defoliation using random sizes of polygons formed in random pixels in the leaf area. The second method makes circles with random radii to remove leaf-belonging pixels in the leaf region. Then, secondary circles with different radii are generated in the circumference of the main circle. The third method is similar to the second method except that the secondary circles are generated within the main circle. They were able to generate 10,000 synthetic defoliation images with each method. Their models were trained only with the synthetically generated images and evaluated using natural images. Their experiments produced an impressive result in estimating soybean leaf defoliation despite being trained using synthetically generated datasets.

Datasets are also synthetically generated in domains other than precision farming. For example, Björklund et al. [BFA+19]proposed a framework for generating challenging synthetic license plate images to avoid collecting and annotating the thousands of images required to train a CNN model. Likewise, Silvano et al. [SRG+21] described a combination of techniques to generate large random synthetic images of license plates to supplement their small volumes of available authentic images for training deep learning-based automated license plate recognition systems.

In the field of biomedical image analysis, Svoboda and Ulman [SU12] generated synthetic static and time-lapse image sequences of fully 3D fluorescence microscopy images showing the motion of objects of various sizes. Han et al. [HHR+18] generated synthetic multi-sequence brain Magnetic Resonance (MR) images using Generative Adversarial Networks (GANs). Prokopenko et al. [PSS+19] investigated approaches for generating synthetic Computed Tomography (CT) images from actual Magnetic Resonance Imaging (MRI) data using GAN to enable single-modality radiotherapy planning in clinical oncology.

4 Design and Implementation

This chapter explains the implementation details of foreground and background image segmentation, region of interest (ROI) segmentation, image rotation and colour transformation for artificial disease image dataset. Figure 4.1 shows an abstract block diagram of the flow of approaches for implementing the proposed methodology, with each block in the diagram discussed in the sections below. The operations in the first three blocks in figure 4.1 were performed on both the diseased and healthy leaf dataset.

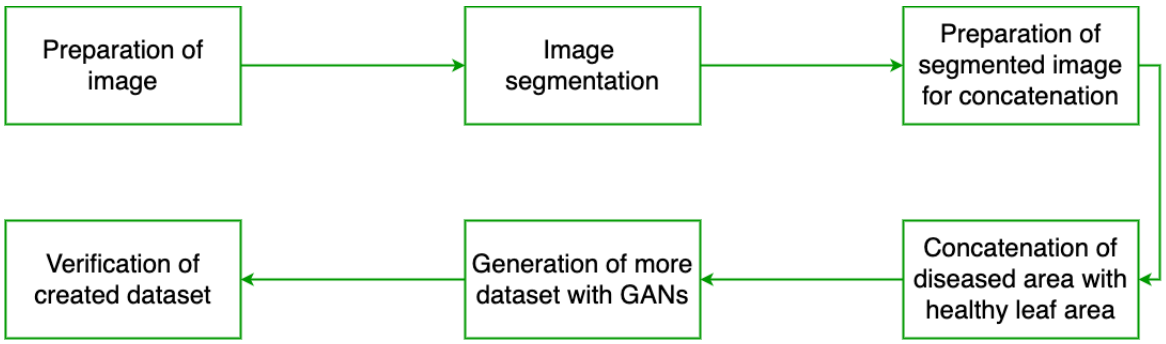

Figure 4.1: Abstract flow of the overall system block diagram of the overall system.

4.1 Prerequisites

Programming language and libraries

OpenCV 2 library and Python 3 programming language are used to implement image processing in this research work. The reason for using Python is its comprehensive and efficient use in image processing tasks due to its excellent libraries and tools collections. Likewise, OpenCV was used due to its optimised implementation of several scientifically proven algorithms for image processing and machine learning tasks. Furthermore, through its library, OpenCV provides easy to use methods in Python for image processing tasks like image segmentation, colour transformations and performing thresholding on images. The Sugar beet disease detection models were trained using the generated synthetic images, and fast.ai [H+18] Python library, which provides easy to use functions and methods for training CNN based architectures. All CNN models were trained on an Ubuntu 20.04.3 LTS Linux distribution server with 2x AMD EPYC 7452 32-Core Processor, 1 TB of RAM, 7x Nvidia A100-PCI-E-40G Band 1x Nvidia Quadro RTX 5000 graphic cards. The code for the implementation of techniques described in this research work can be found on Github: `https://github.com/neyoalao/synthetic_leaf_image_generation`.

Datasets

The publicly available PlantVillage [HS+15] and Plant seedlings [GDJ+17] datasets were used for the experiment in this thesis. The Plant Seedlings dataset contains images of approximately 960 unique plants belonging to 12 species at several growth stages. The dataset was captured at the Aarhus University Flakkebjerg Research station in collaboration with the University of Southern Denmark and Aarhus University. In contrast, the PlantVillage dataset consists of 54,323 images divided into 38 diseased and healthy plants based on 14 different plant species. All images are taken as a single leaf on a solid background labelled only by a class name.

For creating the artificial dataset, we used 463 images of sugar beet in the plant seedlings dataset as a healthy leaf dataset and 513 images of foreground segmented corn leaves infected with grey leaf spot (GLS) in the PlantVillage dataset as the diseased dataset. Corn leaves infected with GLS (caused by *Cercospora zeae-maydis and Cercospora zeina*) were used since it was the only disease in the PlantVillage dataset sharing the same genus (*Cercospora*) of foliar *Cercospora leaf spot (CLS)* caused by *Cercospora beticola*. Figure 4.2 and 4.3 show some images of *Corn* leaves infected with GLS in the PlantVillage dataset and sugar beet leaves in the Plant seedling dataset, respectively. It can be seen from the images that there are varying leaf orientations across the entire dataset.

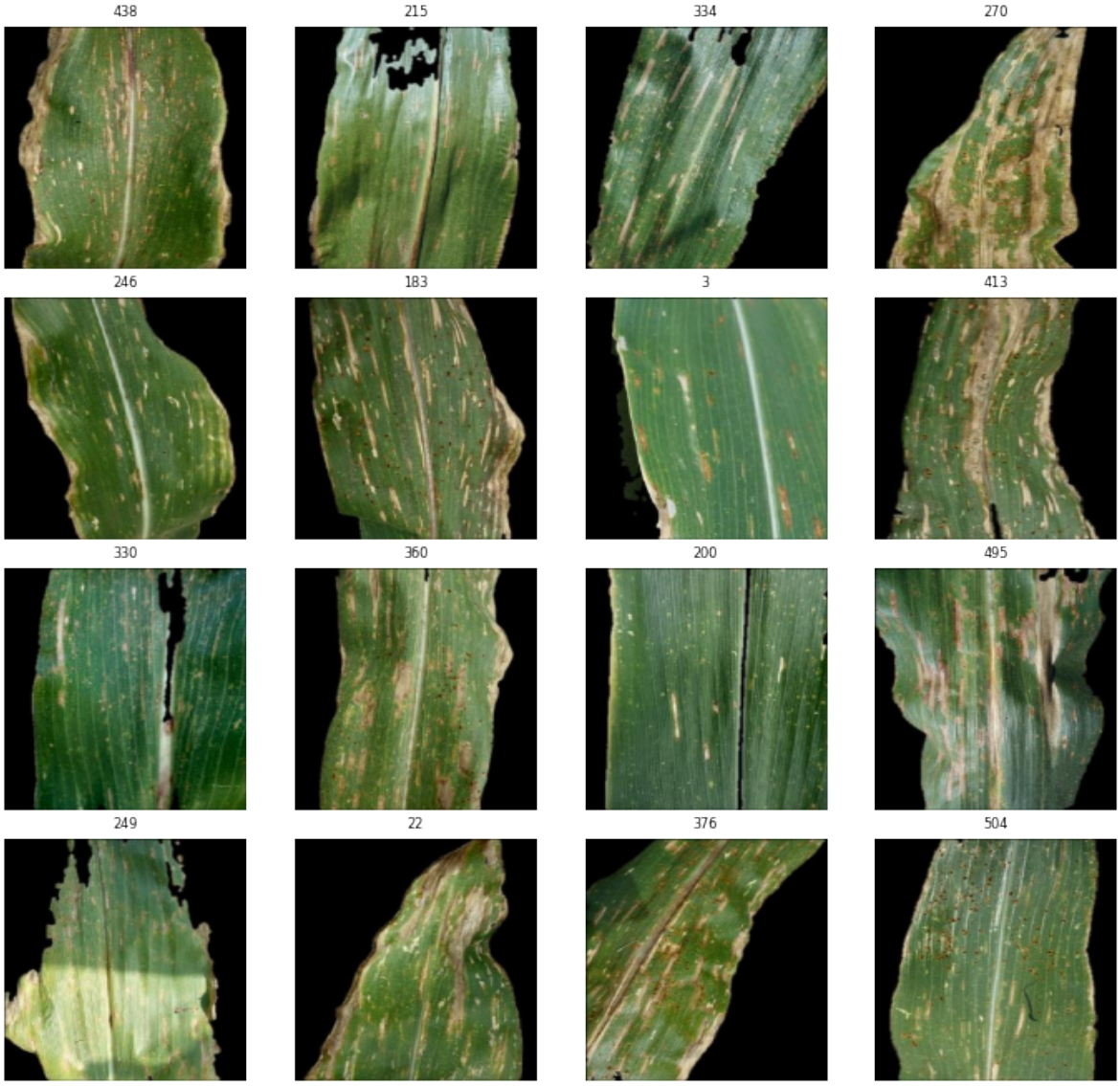

Figure 4.2: Random images in the PlantVillage dataset.

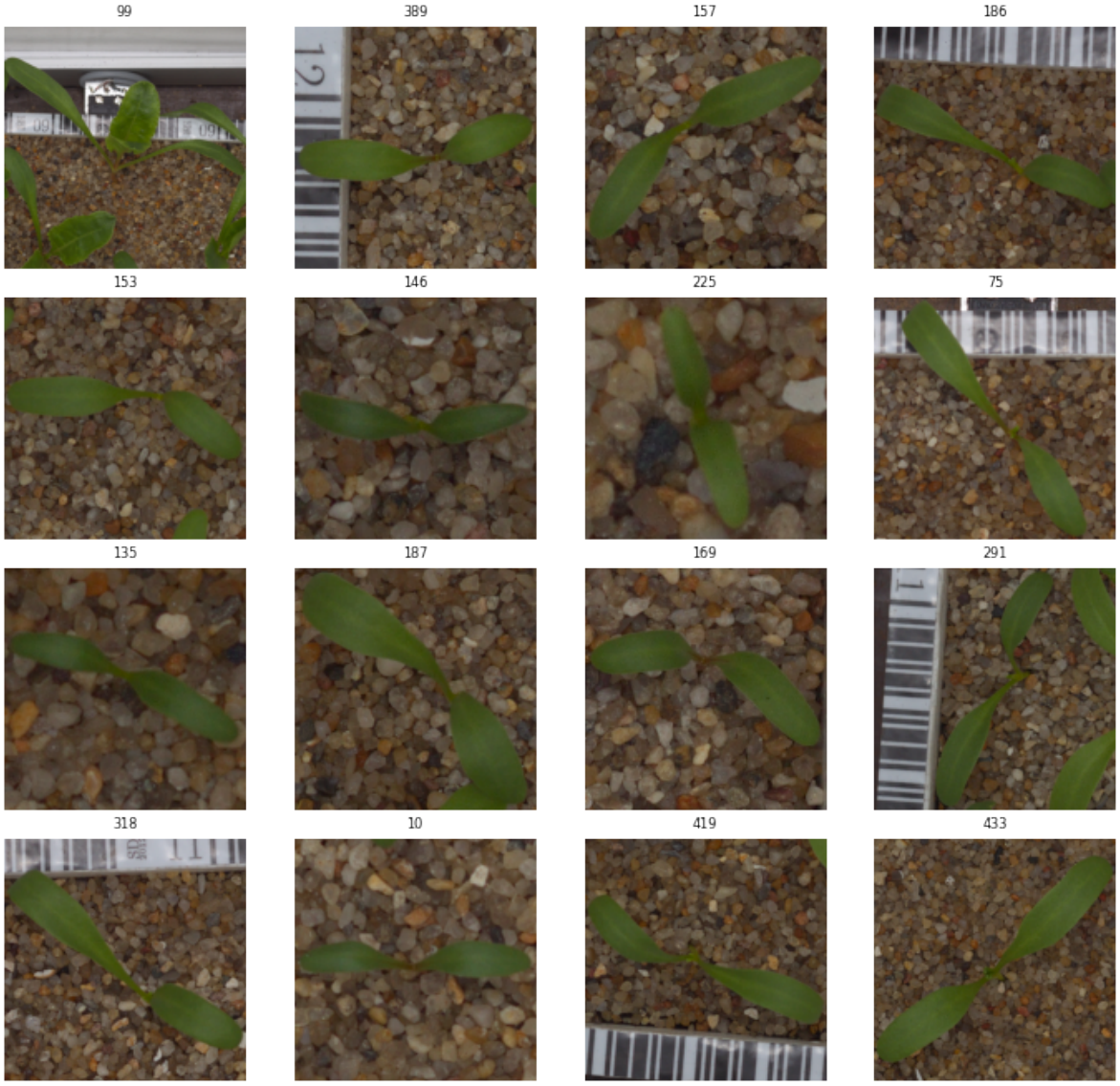

Figure 4.3: Random images in the sugar beet plant seedling dataset.

4.2 Image Preparation

Since the datasets consist of images of different sizes, resizing the images to the same size needs to be done to ensure uniformity in the datasets and allow for a successful addition of the healthy and diseased image dataset since images have to be of the same size. The activity diagram in figure 4.4 shows the complete control flow of operations in preparing the images for image segmentation.

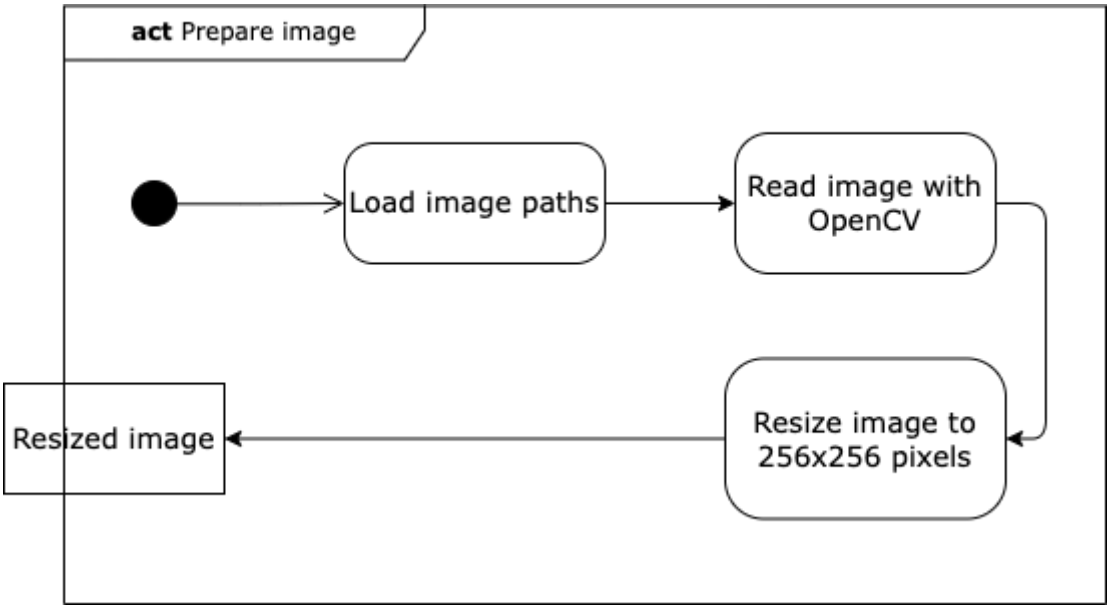

Figure 4.4: Activity diagram for image preparation.

The image path of the image to be resized is read using OpenCV. Reading the image with OpenCV allows the capability of resizing the image to 256 x 256 pixels. The images were resized to 256 x 256 pixels based on the result of research conducted by Rukundo [Ruk21] which found the size performant for training deep learning models. Furthermore, the resized image is passed on to the image segmentation block in figure 4.1 for further image manipulations.

4.3 Image Segmentation

The leaf areas in images of the diseased and healthy datasets were segmented from their respective backgrounds to reduce the computational complexity of manipulating images with non-uniform backgrounds. Figure 4.5 shows the flow of actions in segmenting the leaf areas from the background.

Firstly, the resized image from the previous action is colour transformed from BGR (blue-green-red) into HSV colour space. By default, OpenCV orders the pixels of images in BGR format. Next, the images were transformed to HSV colour representation because it helps detect green leaf and disease areas in the image. Using the HSV information of the image, the values of the lower and upper boundary representing the green leaf and disease areas in the image were found by splitting and plotting random pixels in the image into their respective hue and saturation values. Figure 4.7 show plots of random pixels

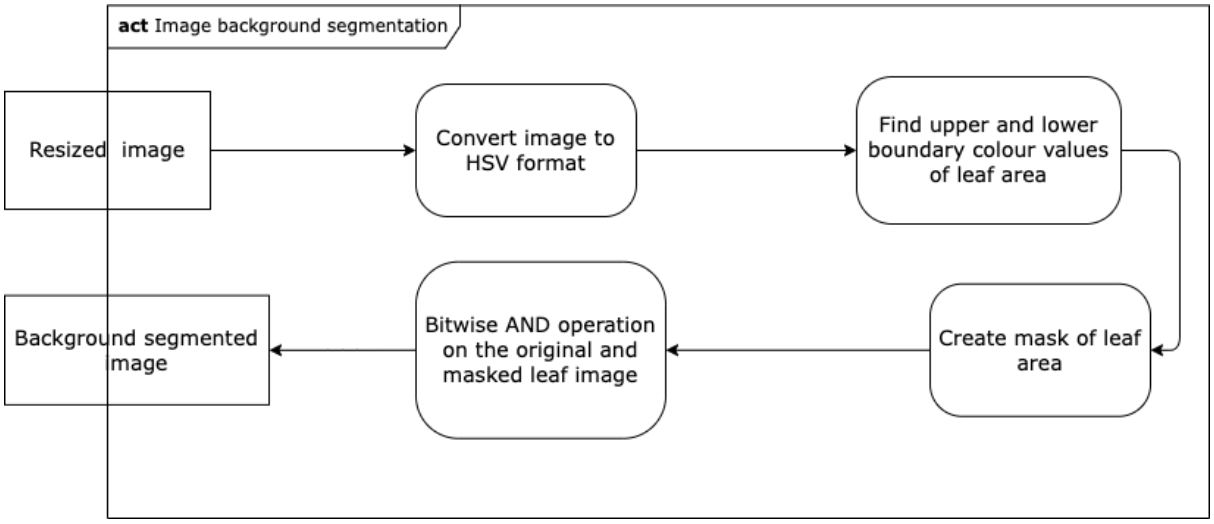

Figure 4.5: Activity diagram showing the flow of actions for image segmentation.

and the hue and saturation values of random pixels in the PlantVillage dataset. The green clusters in the right-hand image represent the leaf area, the brown clusters represent the disease areas, and other colour clusters represent other things like specks of dirt and dark background present in the image.

Figure 4.6: Segmenting sugar beet leaf areas from the background in the plant seedlings dataset.

A mask of the leaf area is created with the found lower and upper boundary HSV value using OpenCV *inRange*() function. Then, the masked image is used to create a segmented image from the original image using the *bitwise_AND* operator. For example, figure 4.6 shows an example of a sugar beet leaf image in the plant seedlings dataset and the resulting background segmented image. Finally, the segmented image is passed on to the next block of actions for further image processing in the section below.

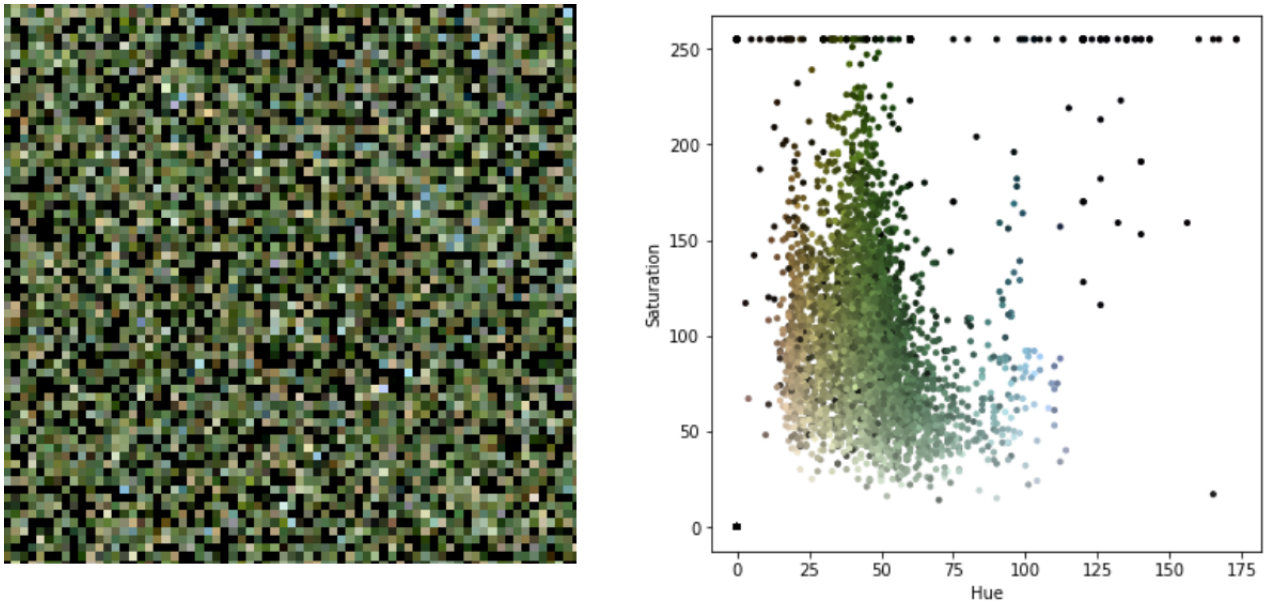

Figure 4.7: Plots of random pixels, hue and saturation values for images in the PlantVillage dataset.

4.4 Prepare segmented image for Concatenation

In this section, the actions depicted in figure 4.8 are carried out. The main goal of the actions performed in this section is to rotate the leaf or disease areas in the images vertically. Rotating the leaf/disease area in the vertical direction allows for the consistent positioning of the disease/leaf areas in the datasets since they have different orientations. In addition, having the disease and leaf areas in the same direction ensures that a more significant part of the segmented disease area land on the sugar beet leaf area during image concatenation.

As a first step, the segmented image is converted to grayscale. Converting the image to grayscale reduces the dimension of the image to an 8-bit single-channel image, which allows easier manipulations than RGB images, which holds the information of all three channels. Equation 4.1 is used for the colour to grayscale algorithm implementation in OpenCV.

$$Y = 0.299R + 0.587G + 0.114B \tag{4.1}$$

Furthermore, the methods *cv2.ADAPTIVE_THRESH_MEAN_C* and *cv2.THRESH_BINARY_INV* available in OpenCV was used to create an adaptive threshold binary image of the segmented input image. Binarizing the image enabled us to find the curve joining all the continuous areas along the external boundary of the leaf area since they have the same intensity and colour, i.e. white. Finding the contours of the object of interest in the image gives access to the object's width, height, length, and orientation angle with respect to the whole image.

The values of the contours of the object of interest were then used to calculate the object's width, height, length and orientation angle by drawing a minimum area rectangle around

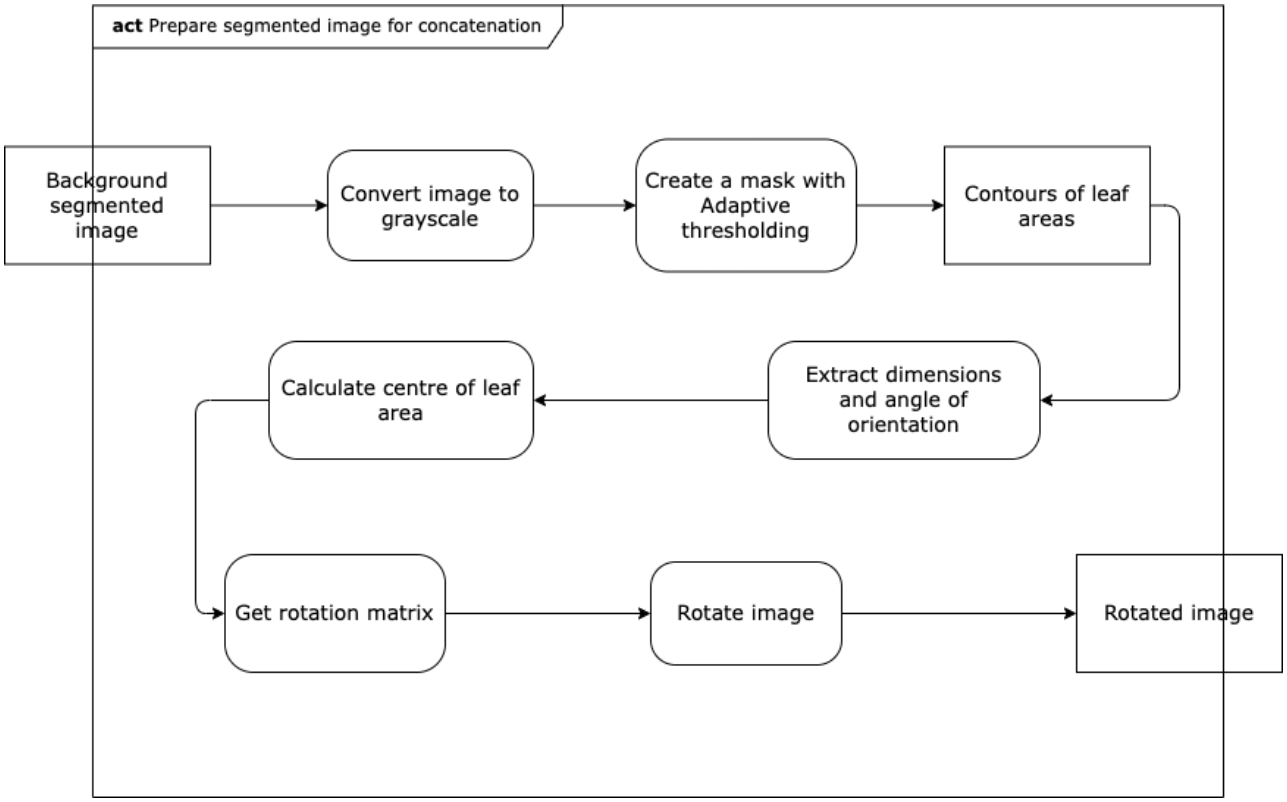

Figure 4.8: Activity diagram showing the flow of actions for preparing segmented images for concatenation.

the found leaf area. Since detecting the specific location of the leaf area is an interesting task, the implemented solution is further explained in detail in section 4.6.

Non vertically oriented leaf areas were vertically rotated using wrap affine transformation on the transformation matrix M gotten from the OpenCV function *cv2.getRotationMatrix2D()*. Rotation in opencv is counterclockwise (i.e. positive degrees specify counterclockwise rotation while negative degrees specify clockwise rotation).The transformation matrix M is defined by equation 4.2

$$M = \begin{bmatrix} \alpha & \beta & (1-\alpha).c_x - \beta.c_y \\ -\beta & \alpha & \beta.c_x + (1-\alpha).c_y \end{bmatrix} \tag{4.2}$$

where $\alpha = scale.\cos\theta$, $\beta = scale.\sin\theta$, c_x and c_y are the x and y coordinates of the centre of the image and $scale$ is the isotropic scale factor.

Finally, the vertically rotated image is passed on to the next block for concatenation operation discussed in the next section.

4.5 Concatenation of disease area with healthy leaf area

Before concatenating the segmented disease and healthy leaf image, there needs to be a confirmation that the images are of the same dimensions (i.e. 256 x 256). Inequality in

the size of the images will lead to an error. Figure 4.10 shows the activity diagram of operations in this section.

A region of interest (ROI) is created on the rotated sugar beet image using the total rows and columns of the disease image. The created ROI of interest has the same size as the disease image since they have the same sizes. Using the disease image, disease areas only corresponding to the leaf regions were extracted using *bitwise_and* operation as shown in figure 4.9b. The bitwise addition decides the pixels to be displayed using an AND operation on each pixel of the images (i.e. a bitwise AND operation is true if and only if both pixels are greater than zero). Next, a mask of the disease is created by converting the diseased image to grayscale and then thresholding the grayscale image using *thresh_binary*. Then an inverse of the mask is created using the *bitwise_not* operation. The inverse mask is used to create areas where the disease will fall on in the leaf area. Then, this region of disease on the sugar beet leaf is blacked out using the inverse mask and *bitwise_operation* as shown in figure 4.9a. Finally, the blackout ROI on the leaf and extracted disease region were added together.

Figure 4.9: (a) Blacked out leaf area for addition of disease (b) Region of disease area which will be overlayed on the leaf area in (a).

The pseudo-code for implementing the algorithm in generating synthetic disease images is highlighted in listings 4.1. The function begins by using a for loop to go through each disease in the disease dataset path. For each disease, disease areas are segmented and rotated. The function continues in line 7 by looping through the healthy leaf images which are concatenated with the segmented disease area. More realistic synthetic diseased leaf datasets can be created using generative adversarial networks. This idea is to train a GANs based model using the generated images from the image manipulation techniques previously discussed. Then, the trained GANs model can be used to create arbitrary values of unique diseased leaf images that could be used in the training phase of a plant disease classifier model.

Of the GANs based architectures, styleGANs have shown promising results in creating higher resolution images of size 256 x 256, ideal for training a CNN-based model. In addition, the StyleGAN architecture combines the proGAN and neural style transfer networks for creating new datasets.

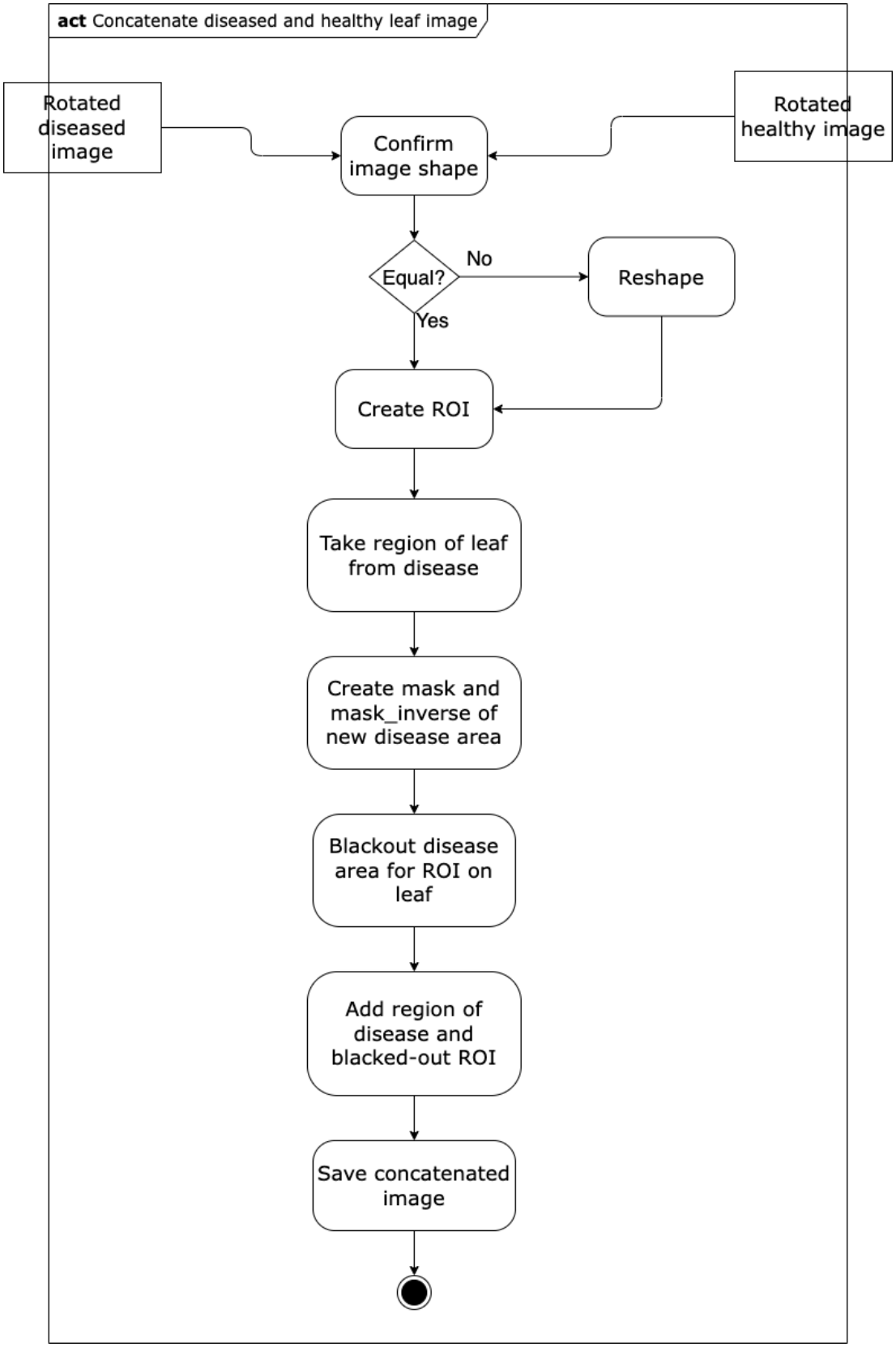

Figure 4.10: Activity diagram showing the flow of actions for concatenating disease are with leaf area.

```
def createSyntheticImages():
    for disease in diseasePath:
        open disease
        segment disease
        find orientation
        rotate disease
        for leaf in leafPath:
            open leaf
            resize image
            segment leaf area
            find orientation
            rotate leaf
            add disease and leaf area
            save image
    return 'done'
```

Listing 4.1: Pseudo code for generating synthetic disease images.

4.6 Detecting Location of Leaf Area in Image

Contours are curves joining all the continuous areas along the external boundary of objects in an image with the same intensity and colour (i.e. white and black for binary image). For example, the red lines drawn around the detected leaf area in figure 4.11 shows an example of contours. Contours are helpful for object detection and shape analysis tasks in image processing.

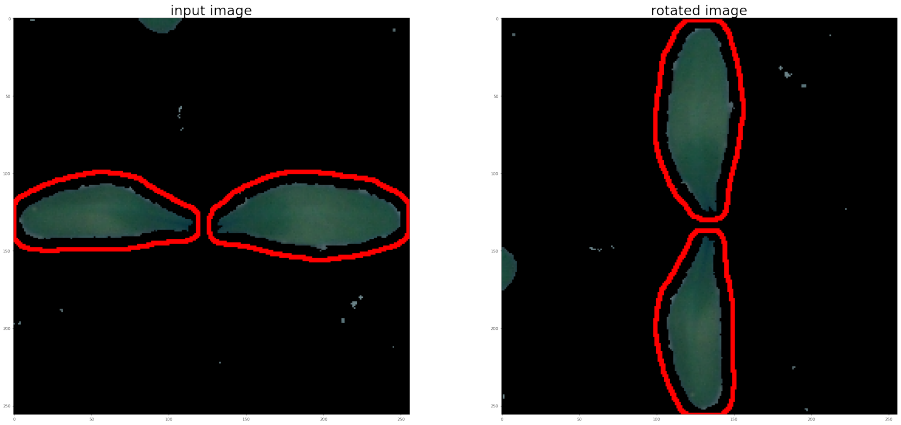

Figure 4.11: Contours drawn around detected leaf areas.

The $cv2.findContours()$ function provided by the OpenCV python library provides an implementation for easily detecting objects in an image. The function takes a binary image, contour retrieval mode and approximation method as arguments. Note that Opencv requires that objects of interest in the binary image are white and the background is black. For this experiment, the following were passed in as arguments; the binary image of the leaf, $RETR\_EXTERNAL$ as the retrieval mode (this extracts just the outer contours of the leaf) and $CHAIN\_APPROX\_SIMPLE$ as the approximation method (stores only the coordinates of the object boundary shape). Figure 4.13 shows the programmatic implementation for finding the contours using Opencv 2 and python 3. The function takes

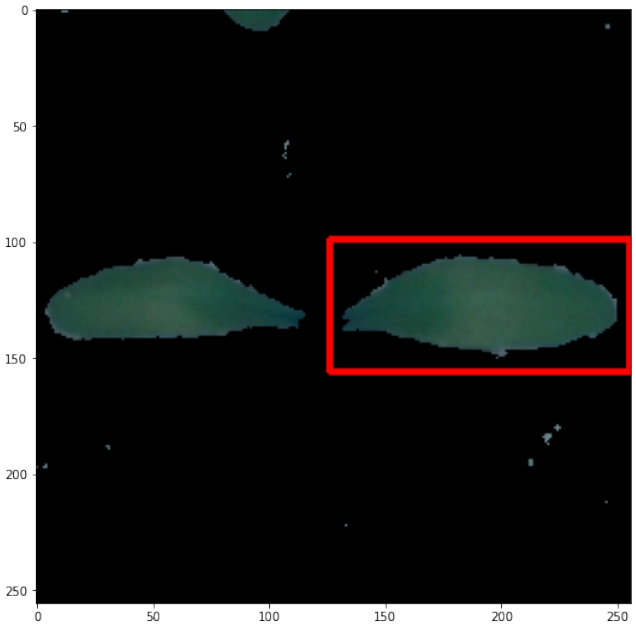

Figure 4.12: Minimum area rectangle drawn around one of the leaf area using its contour values.

an image as input parameter. Then, the input image is transformed into grayscale which is then used to create a binary image mask

```
def detect_objects(image):
    # Convert Image to grayscale
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Create a binary image Mask with adaptive threshold
    mask = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY_INV, 19, 5)

    # Find contours
    contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    objects_contours = []

    for cnt in contours:
        area = cv2.contourArea(cnt)
        if area > 2000:
            objects_contours.append(cnt)

    return objects_contours
```

Figure 4.13: Sample image from the image folder

The contours of binary images are calculated in OpenCV using the algorithm described in [S+85]. The *findContours*() function in OpenCV returns the modified binary image, hierarchy and contours as a python list (which we are interested in) where each contour is an x and y coordinates of boundary points. Every contour whose area is less than 2000 is dropped from the contours list since those values represent non-ROI.

The values of each contour in the new list can be looped through to create a visual line around the boundaries of the detected leaf object using the *cv2_polylines*() function in OpenCV. The lines around the detected leaf region are marked as red in figure 4.11.

Figure 4.14 shows a code snippet of the helper function that is used to find the angle

of orientation of the detected leaf area. The function uses the first contour information passed in as arguments to find its minimum area rotated rectangle. The minimum area rotated rectangle is a bounding rectangle drawn around the detected object to get access to the following details about the object: centre (x, y), (width, height), angle of rotation. Figure 4.12 show the minimum area rectangle of one of the contours of the leaf area. The angle returned by the *findOrientation*() helper function in figure 4.14 is then used to rotate the image about its origin using the *rotateImage*() function shown in figure 4.14 and the resulting rotated image in figure 4.11.

```
def findOrientation(contours):
    for cnt in contours:
        #get rectangle of first contour object
        rect = cv2.minAreaRect(cnt)

        #extract properties from rectangle
        (x, y), (w,h), angle = rect
        print("angle={}, x={}, y={}".format(angle,x,y))

        #display rectangle
        box = cv2.boxPoints(rect)
        box = np.int0(box)
        box_img = cv2.polylines(sb10_img, [box], 1, (255, 0, 0), 2)
        return angle,x,y
```

```
def rotateImage(image, angle):
    center=tuple(np.array(image.shape[0:2])/2)
    rot_mat = cv2.getRotationMatrix2D(center,angle,1.0)
    return cv2.warpAffine(image, rot_mat, image.shape[0:2],flags=cv2.INTER_LINEAR)
```

Figure 4.14: Code snippet for finding leaf orientation and rotating the image using the found angle.

5 Evaluation

Section 5.1 of this chapter presents the results of the experiment using the proposed approach. Finally, in section 5.2, we validated the synthetically generated images by training a CNN-based disease detection model.

5.1 Results of generate images

Every image in the 513 diseased leaf collection was overlaid on the 463 sugar beet images for the experiment. As a result of the experiment, 237 520 synthetic images were created. Figure 5.1 and 5.2 are random images from the generated dataset. The generated images are labelled according to their corresponding number in the original image path to easily find the original images used to form the synthetic image. An image could be name for example *cgls_445_sb_342.png*. *Cgls_445* is short for disease image number 445 in corn's grey leaf spot image folder, while *sb_342* correlates to sugar beet image number 342.

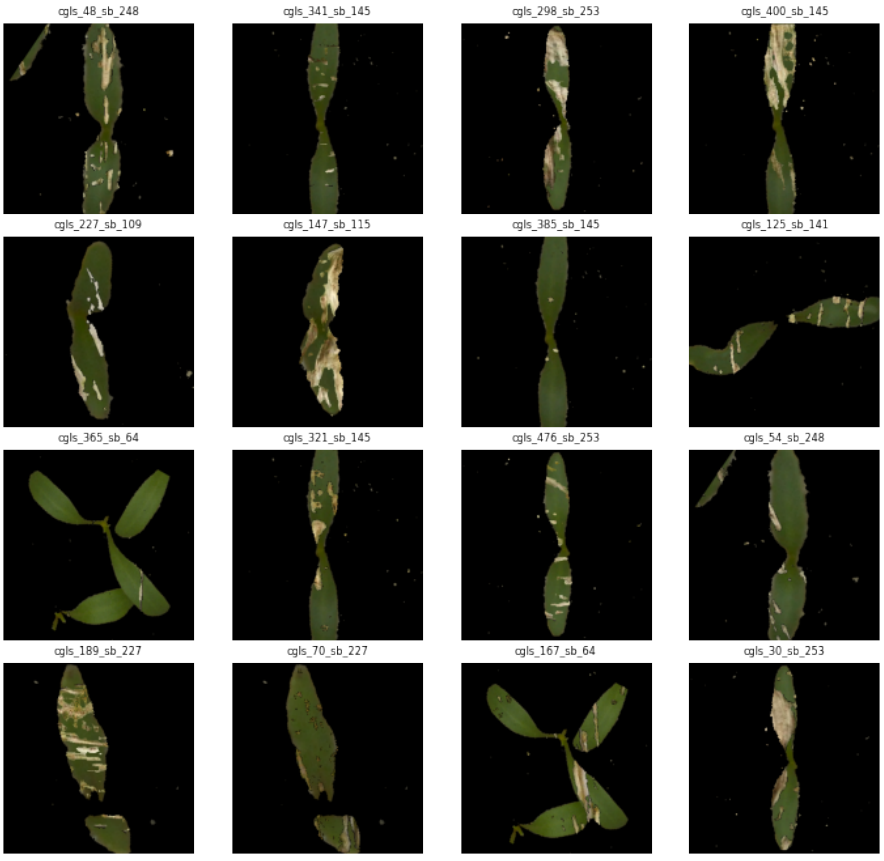

Figure 5.1: Random images from the synthetically generated datasets.

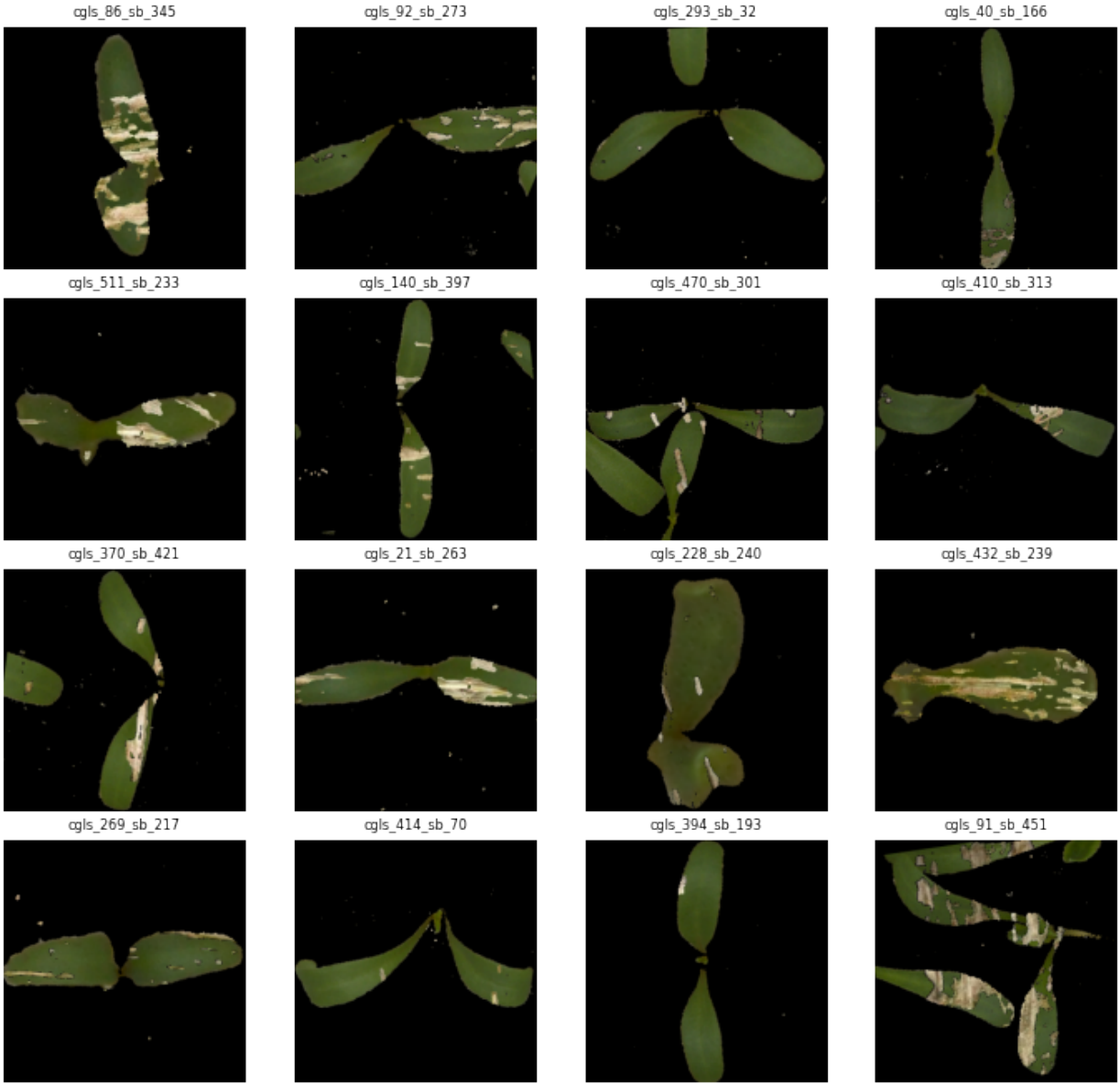

Figure 5.2: Random images from the synthetically generated datasets.

5.2 Validation

CNN-based model training was performed using the generated dataset and the original healthy sugar beet images dataset to validate the generated dataset's efficacy. Four sets of datasets were created for training. The first dataset contains the complete synthetically generated and healthy sugar beet image data. Nine hundred eighty-five randomly chosen synthetic images and all healthy sugar beet images comprise the second dataset. The third dataset consists of 463 synthetic images and 417 healthy sugar beet images. The fourth dataset comprises 453 healthy sugar beet plant images that were not used in synthetic data generation and 1000 synthetically generated images from 10 random sugar beet and 100 random disease plant images. The background of the healthy sugar beet images was segmented across all datasets to avoid the neural network from learning only to make predictions based on the background. Likewise, random image augmentation techniques

like random flipping and clipping were used during model training to avoid the problem of overfitting. The pre-trained versions of Alexnet ResNet 34 and 152 CNN architectures were used for model training to classify input images as healthy or diseased.

Table 5.1 shows the average training time per epoch and accuracy of each architecture trained on the four datasets for five epochs except datasets 2 and 4, which were trained for ten epochs. Models trained on the first dataset had the highest accuracy, which is understandable because of the wide gap between the datasets in the healthy and diseased plant images. The average accuracy of the models across all datasets was above 90%, which shows the success of the synthetically generated dataset for training disease detection models.

Dataset	Architecture	Training time per epoch	Accuracy
Dataset 1 (237, 982 images)	ResNet-34	6 min 7 s	100%
	ResNet-152	2 min 5 s	99.98%
	AlexNet	33 min 29 s	99.99%
Dataset 2 (1448 images)	ResNet-34	5 s	93.05%
	AlexNet	3 s	90.27%
	ResNet-152	31 s	97.91%
Dataset 3 (880 images)	ResNet-34	6 s	93.75%
	AlexNet	4 s	94.88%
	ResNet-152	18 s	91.47%
Dataset 4 (1453 images)	ResNet-34	5 s	98.62%
	AlexNet	4 s	93.10%
	ResNet-152	15 s	98.27%

Table 5.1: Accuracies and training time of CNN models trained of four different sets of data.

6 Summary and Outlook

This chapter begins by giving a summary of the topics discussed in this thesis in section 6.1. Then, section 6.2 gives an outlook for future research works in synthetic dataset generation for plant disease monitoring tasks.

6.1 Summary

This bachelor thesis aimed to determine whether image processing techniques can be used as augmentation techniques to generate more datasets for training disease detection models for precision farming. Previous research showed that basic augmentation techniques like random cropping and rotation were used to increase the training dataset to prevent the problem of overfitting. However, this study is the first known to use the proposed approach to create a synthetic plant disease dataset using image processing techniques from two different plant images. Furthermore, the result of this study confirmed that new datasets could be generated using the proposed techniques.

The results of the trained models using the synthetically generated images in this research showed the tremendous result of the proposed techniques. The proposed approach is an excellent step by step guide that can be used to synthetically generate an arbitrary number of image datasets using healthy and diseased leaf images. Likewise, this approach of dataset generation can significantly reduce the time needed to acquire a dataset for plant disease model training.

This thesis has described a complete approach for generating synthetic disease image datasets for sugar beet plants. The proposed approach builds upon ideas in image processing to provide a general, easy to use method for generating synthetic images. The main contribution is the extraction of the disease sections in diseased plant images and concatenating them on the leaf surface of healthy leaves. The thesis began by highlighting the problem statement of the research work and precision farming regarding disease monitoring and detection. Then we explained the relevant theories in image processing used to augment insufficient datasets in sugar beet. Likewise, we reviewed research papers in precision farming geared towards plant disease detection and monitoring. Finally, the results from the experiments were presented and analysed with the generated fake images.

6.2 Outlook

Although the proposed approach produced tremendous results, there were some shortcomings worth noting.

It can be seen in figure 5.2 that some of the leaves have little or no disease falling on them. This occurrence is mainly attributed to the different orientations in the sugar beet leaf area and disease areas in the images making it difficult to orient them vertically. Likewise, some of the images in the sugar beet dataset had more than two leaves which increased the complexity in re-orienting.

Furthermore, the segmented disease image dataset used for synthetic image generation has some challenging images, as shown in figure 6.1. It can be seen that some of the images in the figure have little or no disease and shadows. Likewise, images 85 and 315 in figure 6.1 are not helpful due to the greyish colour in more than half of the image area. These sorts of images in the disease image path reduced the quality of the generated images. Therefore, it is essential to inspect the quality of base images before using them for dataset generation to avoid the limitations encountered in this experiment.

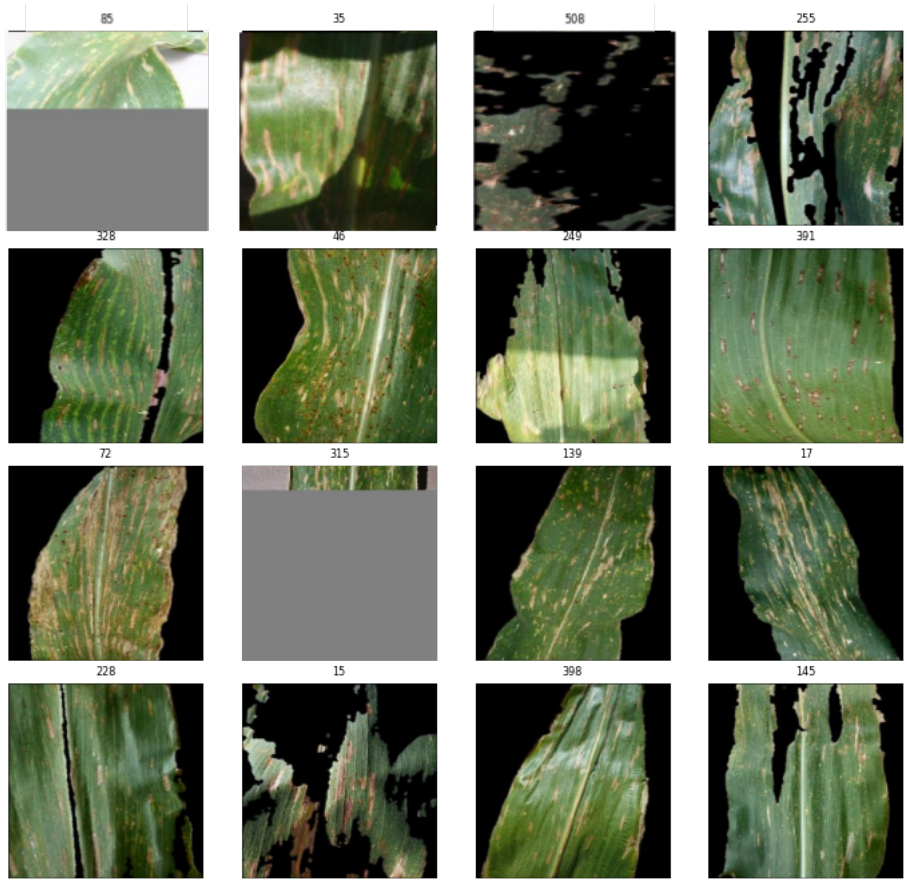

Figure 6.1: Synthetic plant leaf images generated from the PlantVillage dataset using styleGAN. [AKS+19].

Based on the limitations we discussed, we can think about the following next steps to improve the proposed approach. An improvement on the proposed approach is to ensure that the disease is intelligently placed on the leaf areas irrespective of the diversities in the orientation angle of the desired disease and plant leaf dataset.

Likewise, as an extension of this research, it will be interesting to use neural network-based techniques to generate more synthetic datasets in addition to the one generated in this experiment. This technique will briefly be described using scientific references as it was

not implemented in this thesis. However, the understandings from this section can be used as a foundation for implementation in future research.

Of all the GAN-based architecture documented in research papers for creating synthetic plant disease images, styleGAN has proven to have the best result [AKS+19]. StyleGAN proposed by Karras et al. [KLA19] combines their previously proposed progressively growing GAN (ProGAN [KALL17]) and neural style transfer (NST [GEB15]) for creating higher resolution (up to 1024 x 1024 pixels) images with features close to the plant leaves. One of the significant advantages of styleGAN over standard GAN is its ability to influence details in the style of generated images by changing the values of the style vectors and noise of the network.

The StyleGAN architecture uses its discriminator and generator network to progressively generate synthetic images from a very resolution (4 x 4 pixels) until it reaches the desired target image resolution. The incremental learning allows the network to learn more details about the original images over time. StyleGAN replaces the nearest neighbour upsampling layers in the original ProGAN architecture with bilinear sampling. It also uses eight fully connected layers with 512 neurons to process its initial random input.

Figure 6.2 shows the 256 x 256 pixels synthetic images generated from the PlantVillage dataset by Arsenovic et al. [AKS+19] using styleGAN. Based on the resulting images, it can be concluded that styleGAN can be used for generating more synthetic images from the ones created in this thesis.

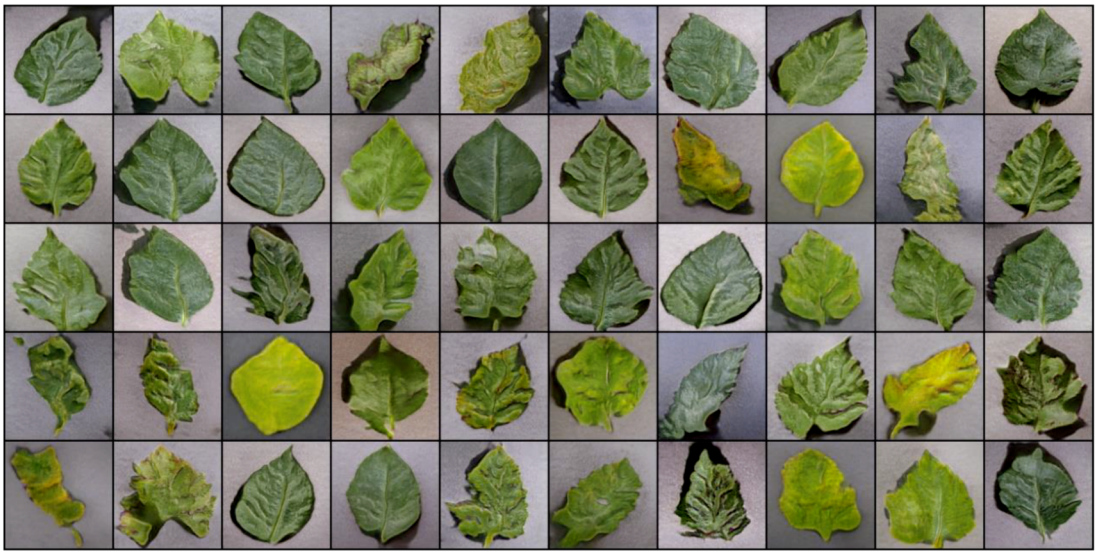

Figure 6.2: Synthetic plant leaf images generated from the PlantVillage dataset using styleGAN. [AKS+19].

Bibliography

[AAM+16] Davoud Ashourloo, Hossein Aghighi, Ali Akbar Matkan, Mohammad Reza Mobasheri, and Amir Moeini Rad. An investigation into machine learning regression techniques for the leaf rust disease detection using hyperspectral measurement. *IEEE journal of selected topics in applied earth observations and remote sensing*, 9(9):4344–4351, 2016.

[AKS+19] Marko Arsenovic, Mirjana Karanovic, Srdjan Sladojevic, Andras Anderla, and Darko Stefanovic. Solving current limitations of deep learning based approaches for plant disease detection. *Symmetry*, 11(7):939, 2019.

[Ala22] Olaniyi Bayonle Alao. Disease detection in sugar beet plant leaves. to appear, 2022.

[AR05] Tinku Acharya and Ajoy K Ray. *Image processing: principles and applications*. John Wiley & Sons, 2005.

[Aue01] Hermann Auernhammer. Precision farming—the environmental challenge. *Computers and electronics in agriculture*, 30(1-3):31–43, 2001.

[Bas02] Mitra Basu. Gaussian-based edge-detection methods-a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 32(3):252–260, 2002.

[BFA+19] Tomas Björklund, Attilio Fiandrotti, Mauro Annarumma, Gianluca Francini, and Enrico Magli. Robust license plate recognition using neural networks trained on synthetic images. *Pattern Recognition*, 93:134–146, 2019.

[BHVH20] Ruud Barth, Jochen Hemming, and Eldert J Van Henten. Optimising realism of synthetic images using cycle generative adversarial networks for improved part segmentation. *Computers and Electronics in Agriculture*, 173:105378, 2020.

[BPVM20] Abel Barreto, Stefan Paulus, Mark Varrelmann, and Anne-Katrin Mahlein. Hyperspectral imaging of symptoms induced by rhizoctonia solani in sugar beet: Comparison of input data and different machine learning algorithms. *Journal of Plant Diseases and Protection*, 127(4):441–451, 2020.

[Ĉad08] M Ĉadík. Perceptual evaluation of color-to-grayscale image conversions. In *Computer Graphics Forum*, volume 27, pages 1745–1754. Wiley Online Library, 2008.

[CDH+16] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 2180–2188, 2016.

[DCSF15] Emily Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. *arXiv preprint arXiv:1506.05751*, 2015.

[dSBG+19] Lucas Abreu da Silva, Patrik Olã Bressan, Diogo Nunes Gonçalves, Daniel Matte Freitas, Bruno Brandoli Machado, and Wesley Nunes Gonçalves. Estimating soybean leaf defoliation using convolutional neural networks and synthetic images. *Computers and electronics in agriculture*, 156:360–368, 2019.

[Fer18] Konstantinos P Ferentinos. Deep learning models for plant disease detection and diagnosis. *Computers and electronics in agriculture*, 145:311–318, 2018.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[GDJ+17] Thomas Mosgaard Giselsson, Mads Dyrmann, Rasmus Nyholm Jørgensen, Peter Kryger Jensen, and Henrik Skov Midtiby. A Public Image Database for Benchmark of Plant Seedling Classification Algorithms. *arXiv preprint*, 2017.

[GEB15] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

[GLS17] Min Gao, Lang Lin, and Richard O Sinnott. A mobile application for plant recognition through deep learning. In *2017 IEEE 13th International Conference on e-Science (e-Science)*, pages 29–38. IEEE, 2017.

[GNYP18] Rutu Gandhi, Shubham Nimbalkar, Nandita Yelamanchili, and Surabhi Ponkshe. Plant disease detection using cnns and gans as an augmentative approach. In *2018 IEEE International Conference on Innovative Research and Development (ICIRD)*, pages 1–5. IEEE, 2018.

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[Gur20] Prathima Guruprasad. Overview of different thresholding methods in image processing. In *TEQIP Sponsored 3rd National Conference on ETACC*, 2020.

[H+18] Jeremy Howard et al. fastai. `https://github.com/fastai/fastai`, 2018.

[HCH07] Pei-Yung Hsiao, Shin-Shian Chou, and Feng-Cheng Huang. Generic 2-d gaussian smoothing filter for noisy image processing. In *TENCON 2007-2007 IEEE Region 10 Conference*, pages 1–4. IEEE, 2007.

[HHR+18] Changhee Han, Hideaki Hayashi, Leonardo Rundo, Ryosuke Araki, Wataru Shimoda, Shinichi Muramatsu, Yujiro Furukawa, Giancarlo Mauri, and Hideki Nakayama. Gan-based synthetic brain mr image generation. In *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*, pages 734–738. IEEE, 2018.

[HRR+14] Celia A Harvey, Zo Lalaina Rakotobe, Nalini S Rao, Radhika Dave, Hery Razafimahatratra, Rivo Hasinandrianina Rabarijohn, Haingo Rajaofara, and James L MacKinnon. Extreme vulnerability of smallholder farmers to agricultural risks and climate change in madagascar. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 369(1639):20130089, 2014.

[HS+15] David Hughes, Marcel Salathé, et al. An open access repository of images on plant health to enable the development of mobile disease diagnostics. *arXiv preprint arXiv:1511.08060*, 2015.

[Jac07] K Jack. Ntsc and pal digital encoding and decoding. *Video Demystified. 5th ed. Burlington: Newnes*, pages 388–465, 2007.

[JCB+20] Sylvain Jay, Alexis Comar, Rafael Benicio, Julie Beauvois, Dan Dutartre, Gaetan Daubige, Wenjuan Li, Jeremy Labrosse, Samuel Thomas, Nicolas Henry, et al. Scoring cercospora leaf spot on sugar beet: comparison of ugv and uav phenotyping systems. *Plant Phenomics*, 2020, 2020.

[KALL17] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[KC12] Christopher Kanan and Garrison W Cottrell. Color-to-grayscale: does the method matter in image recognition? *PloS one*, 7(1):e29740, 2012.

[KLA19] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

[LCP90] Sang Uk Lee, Seok Yoon Chung, and Rae Hong Park. A comparative performance study of several global thresholding techniques for segmentation. *Computer Vision, Graphics, and Image Processing*, 52(2):171–190, 1990.

[LCT+17] Scarlett Liu, Steve Cossell, Julie Tang, Gregory Dunn, and Mark Whitty. A computer vision system for early stage grape yield estimation based on shoot detection. *Computers and Electronics in Agriculture*, 137:88–101, 2017.

[MHS16] Sharada P Mohanty, David P Hughes, and Marcel Salathé. Using deep learning for image-based plant disease detection. *Frontiers in plant science*, 7:1419, 2016.

[MO14] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[NA19] Mark Nixon and Alberto Aguado. *Feature extraction and image processing for computer vision*. Academic press, 2019.

[Ots79] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66, 1979.

[OVSC05] N Otero, L Vitoria, A Soler, and A Canals. Fertiliser characterisation: major, trace and rare earth elements. *Applied geochemistry*, 20(8):1473–1488, 2005.

[OW17] Roger Nkoa Ondoua and Olga Walsh. Precision agriculture advances and limitations: Lessons to the stakeholders. *Crops & Soils*, 50(4):40–47, 2017.

[Pra07] WK Pratt. Digital image processing: Piks scientific inside. wiley-interscience, john wiley & sons, inc. 2007.

[PSS+19] Denis Prokopenko, Joël Valentin Stadelmann, Heinrich Schulz, Steffen Renisch, and Dmitry V Dylov. Unpaired synthetic image generation in radiology using gans. In *Workshop on Artificial Intelligence in Radiation Therapy*, pages 94–101. Springer, 2019.

[RMC15] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[Ruk21] Olivier Rukundo. Effects of image size on deep learning. *arXiv preprint arXiv:2101.11508*, 2021.

[S+85] Satoshi Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, 30(1):32–46, 1985.

[SRG+21] Gilles Silvano, Vinícius Ribeiro, Vitor Greati, Aguinaldo Bezerra, Ivanovitch Silva, Patricia Takako Endo, and Theo Lynn. Synthetic image generation for training deep learning-based automated license plate recognition systems on the brazilian mercosur standard. *Design Automation for Embedded Systems*, 25(2):113–133, 2021.

[SS04] Mehmet Sezgin and Bülent Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic imaging*, 13(1):146–165, 2004.

[SU12] David Svoboda and Vladimír Ulman. Generation of synthetic image datasets for time-lapse fluorescence microscopy. In *International Conference Image Analysis and Recognition*, pages 473–482. Springer, 2012.

[WM18] Jana Wäldchen and Patrick Mäder. Plant species identification using computer vision techniques: A systematic literature review. *Archives of Computational Methods in Engineering*, 25(2):507–543, 2018.

[WMH18] Daniel Ward, Peyman Moghadam, and Nicolas Hudson. Deep leaf segmentation using synthetic data. *arXiv preprint arXiv:1807.10931*, 2018.

[WSW17] Guan Wang, Yu Sun, and Jianxin Wang. Automatic image-based plant disease severity estimation using deep learning. *Computational intelligence and neuroscience*, 2017, 2017.

[YKU+20] Jinhui Yi, Lukas Krusenbaum, Paula Unger, Hubert Hüging, Sabine J Seidel, Gabriel Schaaf, and Juergen Gall. Deep learning for non-invasive diagnosis of nutrient deficiencies in sugar beet using rgb images. *Sensors*, 20(20):5893, 2020.

[ZAK+18] Yezi Zhu, Marc Aoun, Marcel Krijn, Joaquin Vanschoren, and High Tech Campus. Data augmentation using conditional generative adversarial networks for leaf counting in arabidopsis plants. In *BMVC*, page 324, 2018.

[ZHD+19] Xin Zhang, Liangxiu Han, Yingying Dong, Yue Shi, Wenjiang Huang, Lianghao Han, Pablo González-Moreno, Huiqin Ma, Huichun Ye, and Tam Sobeih. A deep learning-based approach for automated yellow rust disease detection from high-resolution hyperspectral uav images. *Remote Sensing*, 11(13):1554, 2019.

List of Figures

List of Tables

Affidavit

I Olaniyi Bayonle Alao herewith declare that I have composed the present paper and work by myself and without use of any other than the cited sources and aids. Sentences or parts of sentences quoted literally are marked as such; other references with regard to the statement and scope are indicated by full details of the publications concerned. The paper and work in the same or similar form has not been submitted to any examination body and has not been published. This paper was not yet, even in part, used in another examination or as a course performance .
Lippstadt, April 30, 2022

Olaniyi Bayonle Alao