

UNIVERSITY OF SCIENCE AND TECHNOLOGY OF HANOI
UNDERGRADUATE SCHOOL



Research and Development
BACHELOR THESIS

By
PHAM GIA PHUC
Information and Communication Technology

Title:
CROP ANALYSIS ON UAV IMAGES

Supervisors: Dr. TRAN GIANG SON
ICTLab

Hanoi, July 2022

ACKNOWLEDGEMENTS

My supervisor, Dr. TRAN Giang Son, has been nothing short of genuine and supportive, and I want to convey my sincere gratitude for that. I found encouragement from Dr. TRAN Giang Son as I struggled to complete this Bachelor's degree. He is the ideal role model and the embodiment of leadership. Without his direction from the first stage of the research, I would not have been able to complete this thesis and have a grasp of the topic. He set up some incredible experiences for me, and I'm grateful for the chances he gave me to advance professionally. Learning from Dr. TRAN Giang Son is an honour.

I am appreciative of my parents' unwavering love and encouragement, which keep me inspired and self-assured. They believed in me, and as a result, I achieved success. My sincere gratitude goes out to my siblings for helping to keep me grounded, reminding me of what's essential in life, and encouraging me on all of my experiences. I will always be grateful for the unwavering love and support I received daily and throughout the thesis-writing process.

Contents

ACKNOWLEDGEMENTS.....	1
LIST OF ABBREVIATIONS.....	3
LIST OF TABLES.....	4
LIST OF FIGURES.....	5
ABSTRACT	6
I/ INTRODUCTION	1
1.1. Context and Motivation	1
1.2. Objective.....	3
1.3. Thesis organisation	3
II/ BACKGROUND	4
2.1. Computer Vision.....	4
2.2. Semantic Segmentation	5
2.3. Convolution, Max Pooling and Transposed Convolution	8
III/ MATERIALS AND METHODS.....	11
3.1. Materials.....	11
3.1.1. Dataset	11
3.1.2. Hardware Infrastructure	16
3.1.3. Libraries and Tools	17
3.2. Methods	18
3.2.1. Data preparation.....	18
3.2.2. UNET Architecture and Training.....	19
3.2.3. Model optimization.....	21
IV/ RESULTS AND DISCUSSIONS.....	26
4.1. Results.....	26
4.1.1. Evaluation metrics	27
4.1.2. Testing with actual images.....	28
4.2. Discussion	29
V/ CONCLUSION.....	29
REFERENCES	31

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
UAV(s)	Unmanned Aerial Vehicle(s)
CNN	Convolutional Neural Network
UNET	U-shaped Convolutional Neural Network
ReLU	Rectified Linear Unit
IoU	Intersection over Union
RGB	Red Green Blue
NDVI	Normalized Difference Vegetation Index
DNN	Deep Neural Network
GSD	Ground Sample Distance
FoV	Field of View
CPU	Central Processing Unit
GPU	Graphics Processing Unit
RAM	Random Access Memory
ICT	Information and Communication Technology
USTH	University of Science and Technology of Hanoi

LIST OF TABLES

Table 1.1.1. Advantages and disadvantages of the existing remote-sensing platforms for agriculture2

Table 3.3.1. Dataset collection information. 12

Table 3.3.2. Multispectral sensor specification.14

Table 3.3.3. Two data collection campaigns. 15

Table 3.3.4. Datasets additional information.15-16

Table 4.1.2.1. Testing images. 28

Table 4.2.1. Model accuracy comparison. 29

LIST OF FIGURES

Figure 1.1.1. Unmanned Aerial Vehicles.	1
Figure 2.1.1. Human Vision and Computer Vision system.	4
Figure 2.1.2. Computer Vision Tasks.	5
Figure 2.2.1. Autonomous vehicles.	6
Figure 2.2.2. BioMedical Image Diagnosis.	6
Figure 2.2.3. Geo Sensing.	7
Figure 2.2.4. Precision Agriculture.	7
Figure 2.3.1. Convolution operation.	8
Figure 2.3.2. Max pooling operation.	9
Figure 2.3.3. LeNet 5.	10
Figure 3.1.2.1. One of the datasets used in this study.	11
Figure 3.1.2.2. Example UAV trajectory.	12
Figure 3.1.2.3. Dataset structure.	13
Figure 3.2.1.1. Dataset used structure.	18
Figure 3.2.1.2. Example of filtered dataset.	19
Figure 3.2.2.1. U-net architecture.	19
Figure 3.2.2.2. Detailed UNET Architecture.	20
Figure 3.2.3.1. Global and Local Minimum.	23
Figure 3.2.3.2. Performance Comparison on Training cost.	24
Figure 3.2.3.3. Intersection over Union.	25
Figure 3.2.3.4. IoU evaluation.	25
Figure 3.2.3.5. Dice Coefficient.	26
Figure 4.1.1.1. Model IoU accuracy and loss.	27
Figure 4.1.1.2. Model Dice Coefficient accuracy and loss.	27

ABSTRACT

Over the past two decades, Unmanned Aerial Vehicles (UAVs), sometimes known as drones, have undergone substantial development for a range of uses, including surveillance, geographic research, fire monitoring, security, military applications, search and rescue, agricultural, etc. In a couple of minutes, they may traverse large distances. UAV-based remote sensing technologies are being employed more and more in agriculture to gather important information that might be used for a variety of precision agriculture applications, including crop/plant classification.

Strong tools and algorithms, such as Deep Learning methods, are required to tackle the task in order to analyse these data effectively. Convolutional Neural Network (CNN) is the most advanced technology for vision applications, having recently emerged as a potent tool for image processing jobs with impressive results.

This study reviews the UNet CNN architecture for UAV-based remote sensing image analysis for crop/plant classification to assist researchers and farmers in selecting the appropriate algorithms for their investigated crops and the available hardware to accurately classify various crop/weed varieties.

Key words: *Unmanned aerial vehicles (UAVs), precision agriculture, Deep Learning, crop/plant classification, UNet Convolutional Neural Network, accurately classified.*

I/ INTRODUCTION

1.1. Context and Motivation

More than seven billion people are fed by agriculture, which was originally practised 13,000 years ago. There were not many people to feed at that time, but the United Nations (UN) projects that by 2050, there will be about 10 billion people on the planet [14], which would present a significant problem for farmers in terms of providing food for everyone. On the other hand, the loss of farmland due to a number of issues, including urbanisation and desertification, poses a significant danger to economic growth and food security. In order to tackle such issues, more dependable ways must be found to produce the necessary quantity of food with a minimum number of human resources.

Numerous academics have over the years suggested a variety of cutting-edge methods to increase agricultural output. The continual advancement of numerous sciences has enabled farmers to fully comprehend the dietary and water needs of plants as well as their sicknesses. By giving them precise and timely information about crop conditions, modern farming practices ensure a rise in yield.

By delivering precise and timely information about the crop state, efficient remote sensing technology makes it feasible to boost crop productivity. Accurate information on crops could aid in the better decision-making of farmers, government officials, and other stakeholders. To accurately classify crops using satellite-based remote sensing data, numerous studies have been conducted with excellent results [6, 7, 16, 17]. However, they have a number of drawbacks that could impair the quality of the data and a variety of weather situations that could make data collection difficult, which could hinder algorithm performance and lead to inaccurate crop analysis. Additionally, traditional methods for analysing different crop/weed types from aerial imagery have relied on well-established machine learning algorithms. These methods are based on manually extracting features using a variety of feature extraction techniques, which are time-consuming and ineffective when dealing with complex data.



Figure 1.1.1. Unmanned Aerial Vehicles [8]

Recent years have seen the emergence of new technologies that have the potential to significantly impact global agriculture and food productivity, especially deep learning and UAV-based remote sensing. They are useful, cutting-edge methods that ought to assist farmers in automating numerous chores, including the identification of plants and products. The advantages of UAVs with numerous sensors over other remote sensing platforms are listed in Table 1.1.1 below. These advantages include high flexibility, low cost, compact size, real-time data collection, and the optimal trade-off between spectral, temporal, and spatial resolution. While inspecting diverse crops, UAVs are a non-destructive technology. These qualities explain why they are often used for crop monitoring and classification.

Remote sensing technologies	Advantages	Dis-advantages
Satellite	Very large area coverage Very high spectral resolution	Very low spatial resolution Cloud sensitivity High cost Data not available all time
Manned Aircraft	Large area coverage High spectral resolution	Cloud sensitivity High cost Low spatial resolution Affected by weather conditions
Unmanned Aerial Vehicle	Low cost High spatial/spectral resolution Data available all time whenever it is needed Not sensitive to clouds Accurate position Difficult areas accessibility	Medium area coverage Low endurance Affected by weather conditions Require more time than satellite to cover very large areas Taught flying laws and regulations
Unmanned - Manned Ground Vehicle Other On-ground Technologies	Very high spatial resolution	Very low area coverage Direct impact on the field Require very long time to cover even small areas

Table 1.1.1. Advantages and disadvantages of the existing remote-sensing platforms for agriculture

However, even with various sensors, data collected by UAVs is not effective enough for precision agriculture, but needs to be processed. There are several ways to handle the data-processing issue, but the most effective, smartest and reasonable method

is applying machine learning and deep learning algorithms into the work. Together, UAVs and deep learning models should be able to provide information on crop status, soil type, and disease/pest attack that was previously impossible.

Some examples of how agricultural services with providers are streamlining their operations with UAVs can be counted: Planting, Irrigation Planning, Resource Distribution, Security and Tracking Sustainability Efforts. Undoubtedly, UAVs are a cutting-edge technology to watch out for as they have a significant impact on farming operations. Farmers can use drones to streamline current procedures and enhance crop management. These responsibilities could increase crop output, reduce costs, and free up a lot of time while giving farmers useful information that would help them make quick decisions.

In light of this, agricultural service providers may be able to lower costs, optimise crop yield, improve worker safety, respond to disasters, and survey land by utilising remote sensing technology with UAVs and applying deep learning algorithms. This might help to support the sustainable and rapid development of global agriculture.

1.2. Objective

This study's primary objective is the development of a deep learning model for crop/weed analysis and mapping that utilises multispectral pictures processed by UNET, a Fully Convolutional Neural Network (CNN), using Unmanned Aerial Vehicles (UAVs). The implemented model will be able to assist scientists and farmers in deciding which algorithms should be chosen to employ for accurately analysing different crop/weed locations.

1.3. Thesis organisation

As the first part of this Thesis is the introduction of the study, the rest of this Thesis is structured as follows:

- **Chapter II: Background** presents some important theoretical bases for Deep Learning methods which are related to this study.
- **Chapter III: Material and Method:** describes the materials and the methodology of solving this study's problem.
- **Chapter IV: Results and Discussions:** shows the experimental results of work done in chapter III.
- **Chapter V: Conclusion:** presents the conclusion and future works of this study.

II/ BACKGROUND

For a better understanding of the field of knowledge and theoretical foundations in this study, some significant background information linked to this topic will be discussed in this session.

2.1. Computer Vision

Computer Vision [5] is a field of Artificial Intelligence (AI) that deals with how computers can be made to gain high-level understanding from digital images, videos and other visual inputs.

From the perspective of engineering, it seeks to automate tasks that the human visual system can do. Figure 2.1.1. below shows a simple visualisation of the Human Vision System compared to Computer Vision System. Human sight has the advantage of lifetimes of context to train how to tell objects apart, how far away they are, whether they are moving and whether there is something wrong in an image.

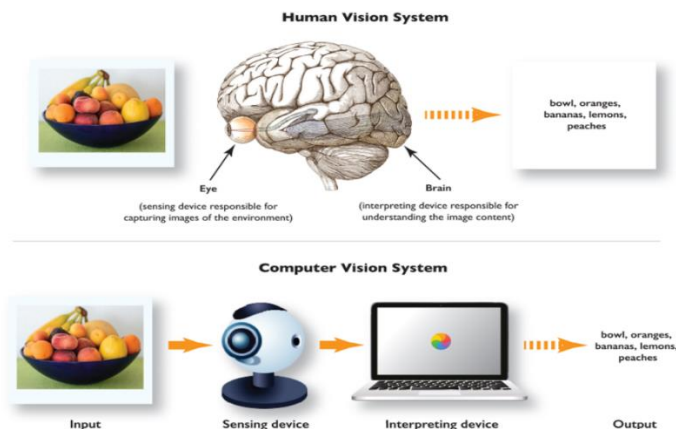


Figure 2.1.1. Human Vision and Computer Vision system ¹

Basically, a digital image is a 2-dimension matrix with a lot of elements within, which is often known as pixels. Each element in the image is a set of pixels which have the different colour with other sets. Therefore, when doing the image processing, these things are converted to the equations based on matrices and vectors.

In terms of video, the essence of a video is many pictures run continuously together, such as 10 fps, 25fps, 30fps, 60fps, 120fps or more, thus applying computer vision to video is essentially running algorithms on a series of images continuously at high speed.

¹ https://zkteco.in/news_detail/635.html

Some operations commonly used in computer vision based on a Deep Learning perspective include:

- Convolution
- Pooling
- Non-Linear Activations

In essence, computer vision is used for tasks involving teaching computers to comprehend both digital images and visual information from the outside environment. This may entail taking information from such sources, processing it, and analysing it to make judgments. Large-scale formalisation of challenging problems into well-known, defensible problem statements was a hallmark of the development of machine vision. Researchers from all over the world were able to recognize issues and effectively address them thanks to the topical division into well-defined areas with appropriate nomenclature.

The most common computer vision tasks (some is presented in Figure 2.1.2. below) that frequently encountered in AI terminology include:

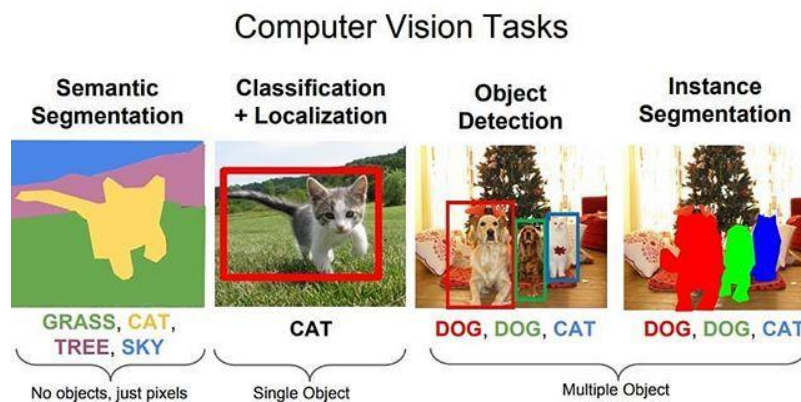


Figure 2.1.2. Computer Vision Tasks²

- Image segmentation
- Object detection
- Face and person recognition
- Edge detection
- Image restoration
- Feature matching
- Scene reconstruction
- Video motion analysis

2.2. Semantic Segmentation

Semantic Image Segmentation [3] is a computer vision task in which specific regions of an image will be labelled according to what's being shown.

² <https://blog.xpress.ai/mask-r-cnn-single-class-instance-segmentation-tutorial/>

The goal of semantic image segmentation is to categorise each pixel in a picture according to the concept it is meant to convey. Since every pixel in the image is predicted, this approach is referred to as dense prediction. One key thing to remember is that it only takes into account the category of each pixel, not other instances of the same class. In other words, the segmentation map does not automatically distinguish two objects from the same category in the input image as separate ones. Different objects belonging to the same class can be distinguished using a different class of models called instance segmentation models.

Segmentation models are useful for a variety of tasks, including:

- Autonomous vehicles

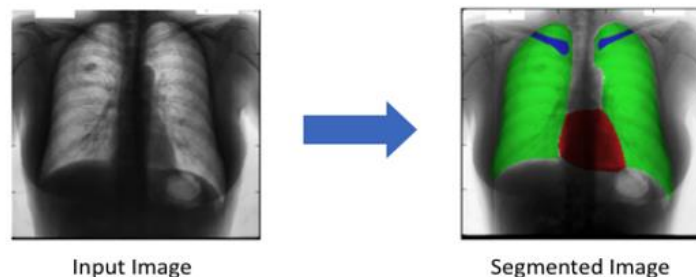
The challenging robotics task of autonomous driving calls for sensing, planning, and execution within ever-changing conditions like showing in Figure 2.2.1. below. Since safety is of the utmost significance, this task also needs to be carried out with absolute precision. Semantic segmentation can identify lane lines and traffic signs, as well as provide information about open area on the highways.



*Figure 2.2.1. Autonomous vehicles*³

- BioMedical Image Diagnosis

The amount of time needed to perform diagnostic tests can be significantly decreased by using machines to supplement radiologists' analysis like shown in Figure 2.2.2. below.



*Figure 2.2.2. BioMedical Image Diagnosis*⁴

³ https://blog.csdn.net/weixin_42392454/article/details/118460055

⁴ <https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47>

- Geo Sensing

Each pixel is assigned to one of a number of different item classes in semantic segmentation issues, which are also known as classification problems. Thus, charting land use using satellite photography has a use case. Information about land cover is crucial for several purposes, including tracking urbanisation and deforestation.

Land cover classification like showing in Figure 2.2.3. below can be thought of as a multi-class semantic segmentation job because it identifies the kind of land cover (such as urban, agricultural, or water areas, etc.) for each pixel on a satellite picture. For traffic management, city planning, and road monitoring, road and building detection is an essential research area.



Figure 2.2.3. Geo Sensing ⁴

- Precision Agriculture

The number of herbicides that need to be sprayed on the fields can be decreased by using precision farming robots, and semantic segmentation of crops and weeds helps them in real time to initiate weeding actions. These cutting-edge image vision (like shown in Figure 2.2.4. ⁴ below) systems for agriculture can lessen the need for manual monitoring.



Figure 2.2.4. Precision Agriculture ⁴

2.3. Convolution, Max Pooling and Transposed Convolution

It is very important to understand the different operations that are typically used in a Convolutional Network. The terminology used is listed below.

○ Convolution operation

There are two inputs to a convolutional operation

- A 3D volume (input image) of size ($n_{in} \times n_{in} \times \text{channels}$)
- A set of “k” filter (also called as kernels or feature extractors) each one of size ($f \times f \times \text{channels}$), where f is typically 3 or 5
- The output of a convolutional operation is also a 3D volume (also called as output image or feature map) of size ($n_{out} \times n_{out} \times k$)
- The relationship between n_{in} and n_{out} is as follows”

$$n_{out} = \left\lfloor \frac{n_{in} + 2p - k}{s} \right\rfloor + 1$$

n_{in} : number of input features
 n_{out} : number of output features
 k : convolution kernel size
 p : convolution padding size
 s : convolution stride size

Following Figure 2.3.1. is an illustration of a convolution operation:

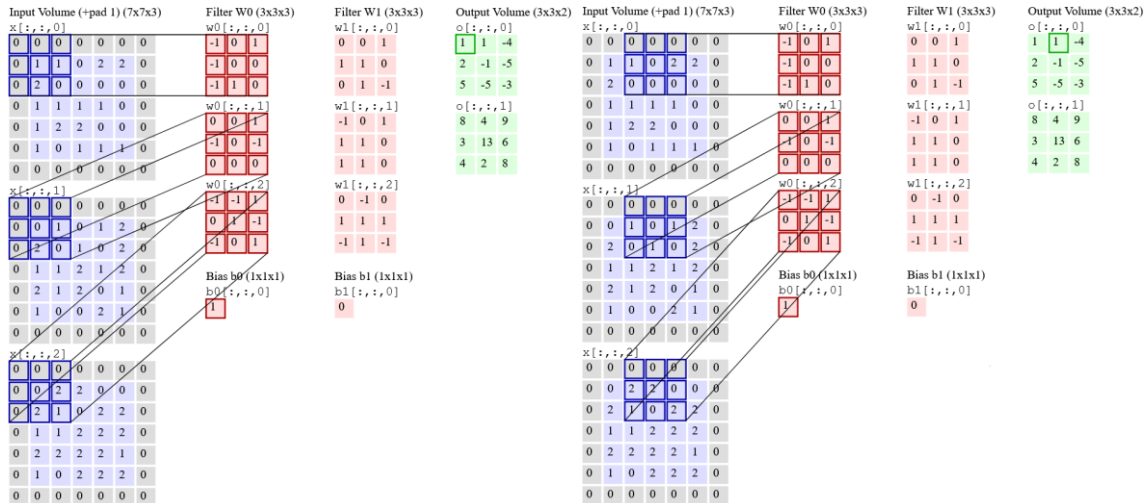


Figure 2.3.1. Convolution operation⁵

⁵ <https://cs231n.github.io/convolutional-networks/>

In the figure 2.3.1. above, there is an input volume of size $7 \times 7 \times 3$. Two filters each of size $3 \times 3 \times 3$. Padding = 0 and Strides = 2. Hence the output volume is $3 \times 3 \times 2$.

The Receptive field is a crucial phrase that is frequently used. This area of the input volume is being examined by a specific feature extractor (filter). The receptive field is represented in the above picture as the 3×3 blue region in the input volume that the filter at any one time covers. The context is another name for this.

To put in very simple terms, the receptive field (context) is the area of the input image that the filter covers at any given point of time.

○ Max pooling operation

The goal of pooling, to put it simply, is to make the feature map smaller so that the network has fewer parameters.

For example, in the Figure 2.3.2. below:

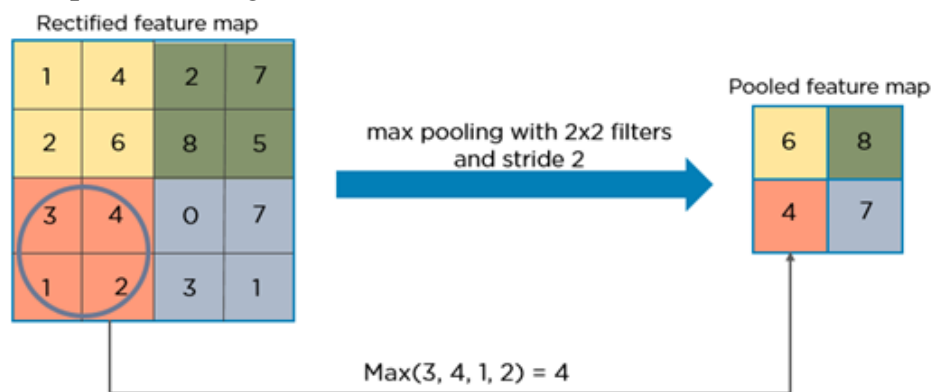


Figure 2.3.2. Max pooling operation

In essence, the maximum pixel value from each 2×2 block of the input feature map is chosen to create a pooled feature map. Keep in mind that strides and filter size are two crucial hyper-parameters for the max pooling operation.

The idea is to eliminate unnecessary information and save only the most crucial details (pixels with maximum values) from each zone.

It is very important to note that both convolution and, in particular, the pooling procedure, lower the size of the image. This is called down sampling. The image size in the example above is 4×4 before pooling, and 2×2 after pooling. In reality, down sampling essentially involves reducing the resolution of a high-resolution image.

In other words, after pooling, the information that was in a 4×4 image previously is now (nearly) present in a 2×2 image.

The filters in the following layer will now be able to view a wider context when the convolution process is applied again; that is, as one moves further into the network, the size of the image decreases but the receptive field expands.

Below Figure 2.3.3. is an example of the LeNet 5 [\[4\]](#) architecture:

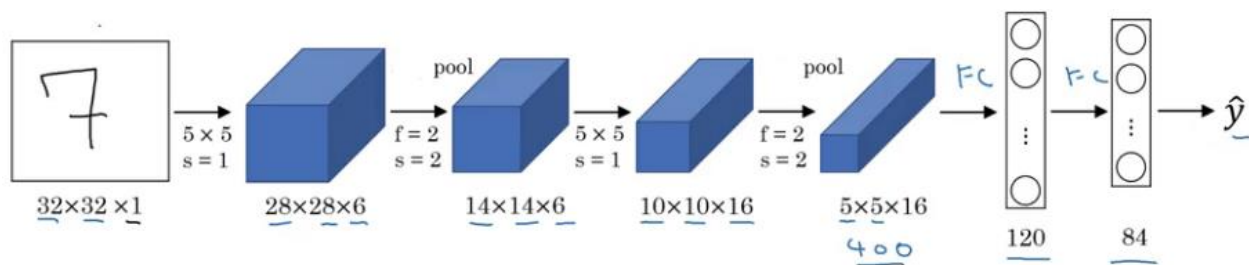


Figure 2.3.3. LeNet 5

Observe how the height and width of the image rapidly decrease in a typical convolutional network (down sampling, due to pooling), allowing the filters in the deeper layers to concentrate on a larger receptive field (context). To extract more complicated features from the image, the number of channels/depth (number of filters utilized) constantly increases.

The pooling operation leads logically to the following conclusion. The model gains a better understanding of "WHAT" is there in the image by down-sampling, but it loses information about "WHERE" it is present.

○ Need for up sampling

Semantic segmentation produces more than just a class label or a set of bounding box parameters. In actuality, the output is a comprehensive, high-resolution image with pixel classification.

In order to avoid losing the "WHERE" information and only retain the "WHAT" information, using a typical convolutional network with pooling layers and dense layers is inconsequential.

Hence in order to recover the "WHERE" information, it is necessary to up sample the image, or change a low-resolution image to a high-resolution image.

There are several methods for up-sampling an image in the literature. Unpooling, transposed convolution, nearest neighbour interpolation, bi-linear interpolation, and cubic interpolation are a few of them. However, transposed convolution is the favoured option for up-sampling a picture in the majority of cutting-edge networks.

○ Transposed Convolution

Transposed convolution is a technique for performing up-sampling of an image with learnable parameters. It is often referred to as deconvolution or fractionally strided convolution.

In essence, it is the exact reverse of a conventional convolution in that the input volume contains low quality images, while the output volume contains high resolution images.

III/ MATERIALS AND METHODS

The resources and methodology needed to implement the crop/weed segmentation problem are described in this session.

3.1. Materials

3.1.1. Dataset

The 2018 Weed Map Dataset [\[10\]](#) is used in the validation and training processes. The sections that follow include information regarding sensor requirements, file name conventions, tiled and orthomosaic data, as well as the datasets for sugar beet and weeds.

The figure 3.3.1. below is an example of one of the datasets used in this study. The centre and right photos show zoomed-in views of each region, while the left image is an orthomosaic of the entire map. The yellow, red, and cyan boxes depict different fields in the image, according to clipped perspectives. These facts clearly show how large the agricultural area is and highlight how difficult it is to visually distinguish the crops from the weeds (limited number of pixels and similarities in appearance).

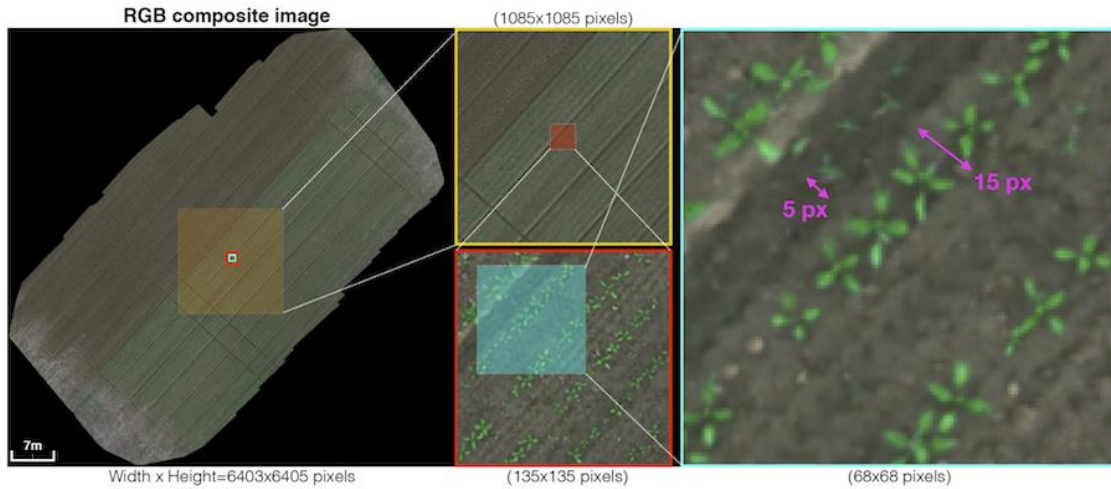


Figure 3.1.2.1. One of the datasets used in this study [\[10\]](#)

The following figure 3.3.2. displays a sample UAV trajectory for a 1,300 square metre sugar beet field. Each yellow frustum, which stands for the place where an image capture was made, is connected to its co-visible numerous views by the green lines, which symbolise rays. The 2D feature points from the proper subplots were successfully collected and matched to produce an accurate orthomosaic map, which is evident in terms of quality. A comparable coverage-type flying path is used for dataset collecting.

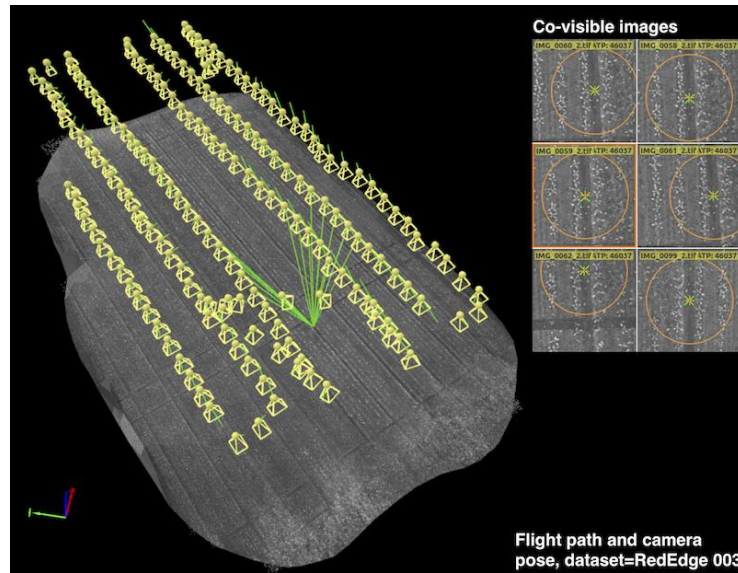


Figure 3.1.2.2. Example UAV trajectory [10]

Over the course of five months, the dataset was gathered from sugar beet fields in Eschikon, Switzerland, and Rheinbach, Germany, using two commercial quadrotor UAV systems from MicaSense, the RedEdge-M and Sequoia (as shown in the Table 3.3.1 below).

Description	1 st campaign	2 nd campaign
Location	Eschikon, Switzerland	Rheinbach, Germany
Date, Time	5-18/May/2017, around 12PM	18/Sep/2017, 9:18-40 AM
Aerial platform	Mavic pro	Inspire 2
Sensor ^a	Sequoia	RedEdge
# Orthomosaic map	3	5
Training/Testing multispectral image ^b	227/210	403/94
Crop	Sugar beet	
Altitude	10m	

Table 3.3.1. Dataset collection information

The original datasets are divided into 129 directories and 18746 picture files as shown in Figure 3.3.3. of structure as below:



Figure 3.1.2.3. Dataset structure

After file extraction, there are two files called Orthomosaic and Tiles that each contain orthomosaic maps and the corresponding tiles (portions of the region in an image with the same size as that of the input image). An orthomosaic map is covered in a window that is moved across a number of its tiles.

The RedEdge and Sequoia subfolders of the orthomosaic folder house eight orthomosaic maps. Each of the subfolders, which are indexed 000-007, houses the composite-png, groundtruth, and reflectance-tif folders. Each of the eight tiles includes a groundtruth, mask, and tile folder that resembles the Sequoia folder.

The datasets are indexed from 000 to 007, and it is expected that most file names follow straightforward rules. For instance, an orthomosaic image of dataset 000 can be found in the groundtruth folder from Orthomosaic Rededge 000, (specifically, RedEdge dataset). Although groundtruth, mask, and tile require further explanation, the Tiles folder followed the same standards (000-007 denoting dataset indices, for example).

The groundtruth folder contains both the original RGB (or CIR) shots and photos with hand-written annotations. Color and indexed pictures are the two categories of labelled images. XXX frameYYYY GroundTruth color.png is the file naming convention for the former, whilst XXX frameYYYY GroundTruth iMap.png is used for the latter. XXX denotes the dataset index (i.e., 000–007), and the four-digit frame number (YYYY) begins at 0000, not 0001.

The index image (XXX frameYYYY GroundTruth iMap.png) encodes these classes as integers background=0, crop=1, weed=2, and non-class=10000. The color groundtruth images (XXX frameYYYY GroundTruth color.png) display each class in color; background=black, crop=green, and weed=red. The color groundtruth image has a file format of 3 channel, 8-bit PNG, 480360, whereas the index image is a grayscale image with a 16-bit resolution (width x height).

Binary masks of the tile pictures can be found in the mask folder (white: valid, black: invalid). The file names have the format "frameYYYY," where YYYY is the frame number that was previously mentioned. The tile folders have 8 and 6 subfolders,

respectively (RedEdge and Sequoia respectively). The files are named using the same structure as mask, with "frameYYYY" designating the current frame, and each folder corresponds to a single channel.

The Table 3.3.2. below provides more information on the demands for multispectral sensors:

Description	RedEdge	Sequoia	Unit
Pixel size	3.75		um
Focal length	5.5	3.98	mm
Resolution (width x height)	1280 x 960		pixel
Raw image data bits	12	10	bit
Ground Sample Distance (GSD)	8.2	13	cm/pixel (at 120m altitude)
Imager size (width x height)	4.8 x 3.6		mm
Field of View (Horizontal, Vertical)	47.2, 35.4	61.9, 48.5	degree
Number of spectral bands	5	4	n/a
Blue (Centre wavelength, bandwidth)	475, 20	N/A	nm
Green	560, 20	550, 40	nm
Red	668, 10	660, 40	nm
Red Edge	717, 10	735, 10	nm
Near Infrared	840, 40	790, 40	nm

Table 3.3.2. Multispectral sensor specifications

According to table 3.3.3. below, the two data gathering campaigns span 16,554 square metres in total. The two used cameras have 5 and 4 raw image channels respectively, and the R, G, and B channels are stacked to create RGB images (RedEdge-M) and R, G, and NIR images to create CIR images (Sequoia). The Normalised Difference Vegetation Index is also extracted (NDVI). For the RedEdge-M and Sequoia cameras, these procedures provide 12 and 8 channels, respectively. As a result of treating each channel as an image, a total of 1.76 billion pixels—1.39 billion for training and 367 million for testing—are produced (10,196 images).

The input image size refers to the resolution of data received by the Deep Neural network (DNN). Since most CNNs downscale input data due to the difficulties associated with memory management in GPUs, the input image size is defined to be the same as that of the input data. This way, the down-sizing operation is avoided, which significantly

degrades classification performance by discarding crucial visual information for distinguishing crop and weeds.

Description	RedEdge	Sequoia
# Orthomosaic map	5	3
Total surveyed area (ha)	0.8934	0.762
# channel	12 ^a	8 ^b
Input image size (tile size)	480 x 360	
# training data	# images = 403 x 12 = 4,836 ^c	#images = 404 x 8 ^b = 3,232
	# pixel = 835,660,800	# pixel = 558,489,600
# testing data	94 x 12 = 1,128	125 x 8 = 1,000
	# pixel = 194,918,400	# pixel = 172,800,000
Total data	# image = 10,196	
	# pixel = 1,761,868,800	
Altitude	10 m	
^a 12 channels of RedEdge data consists of R(1), Rededge(1), G(1), B(1), RGB(3), CIR(3), NDVI(1), and NIR(1). The number in parentheses indicates the number of channels.		
^b 8 channels Sequoia data consists of R(1), Rededge(1), G(1), CIR(3), NDVI(1), and NIR(1).		
^c Each channel is treated as an image.		

Table 3.3.3. Two data collection campaigns

Additional information about datasets is provided in the table 3.3.4. below. Given a sensor, pixel size, picture resolution, altitude, and camera focal length as determined by its Field of View, the Ground Sample Distance (GSD) specifies the distance between two pixel centres when projecting them on the ground (FoV). Given the characteristics of the camera and flight height, a GSD of around 1 cm might be obtained. This is consistent with the dimensions of weeds and crops (15–20 pixels) (5-10 pixels).

Sensors	RedEdge					Sequoia		
Dataset name	000	001	002	003	004	005	006	007
Resolution	5995	4867	6403	5470	4319	7221	5601	6074
(col/row)	x	x	x	x	x	x	x	x
(width/height)	5854	5574	6405	5995	4506	5909	5027	6889
Area covered (ha)	0.312	0.1108	0.2096	0.1303	0.1307	0.2519	0.3316	0.1785
GSD (cm)	1.04	0.94	0.96	0.99	1.07	0.85	1.18	0.83

The resolution (row/col) pixels	360/480							
# effective tiles	107	90	145	94	61	210	135	92
# tile in row / # tile in col	17x13	16x11	18x14	17x12	13x19	17x16	14x12	20x13
Padding info (row/col) pixels	$\frac{266}{245}$	$\frac{186}{413}$	$\frac{75}{317}$	$\frac{125}{290}$	$\frac{174}{1}$	$\frac{211}{459}$	$\frac{13}{159}$	$\frac{311}{166}$
Attribute	train	train	train	test	train	test	train	Train
# channels	5						4	
Crop	Sugar beet							

Table 3.3.4. Datasets additional information

The number of images that actually have any valid pixel values other than entirely black pixels is known as the number of effective tiles. This happens because the tiles from the farthest upper left or lower right corners of orthomosaic maps are wholly black images. The amount of tiles in a row or column, or the number of 480x360 photos, is indicated by the number of tiles in that row or column. In order to match the size of the orthomosaic map with a particular tile size, padding information indicates the number of additional black pixels in rows and columns. This data is used to create a segmented orthomosaic map from the tiles, along with a ground truth map for it.

3.1.2. Hardware Infrastructure

Since graphics cards play a significant role in quick, effective, and high-performance processing, they must be mentioned while discussing the development of machine learning. A high complexity problem can be solved much more quickly because of the daily improvements in GPU processing speed. The training experiments for this internship were carried out using the high-performance computer infrastructure at USTH ICTLab, and the specifics are as follows:

- CPU: Intel (R) Xeon (R) CPU E5-2620 v3 @ 2.40Ghz
- GPU: Tesla K80
- RAM: 128 GB

3.1.3. Libraries and Tools

Python, a high-level programming language that facilitates more effective system integration, is used in the internship project. Among programmers, Python is well-liked. It was designed by Guido van Rossum and published in 1991. It is used in server-side web development, software development, mathematics, and system programming.

Python can be used on a server to build web applications, used in conjunction with other programs to build processes, and it links to database systems including the ability to read and edit files. Python can also be used for rapid software development, handling large amounts of data, performing sophisticated mathematics, and handling big data.

It is compatible with a variety of operating systems (Windows, Mac, Linux, Raspberry Pi, etc.), has a straightforward syntax resembling English, allows programmers to write programs with fewer lines of code than with some other programming languages, and operates on an interpreter system, allowing code to be executed immediately after it is written. In light of this, prototyping can be done quickly and, in a procedural, object-oriented, or functional manner. Python 3, which will be utilised in this study, is the most recent major version of the language.

In addition to the fundamental Python libraries, the data pre-processing and model training phases in this study also make use of the following open-source tools:

- **Matplotlib version 3.5.2:** a complete library in Python for the creation of static, animated, and interactive visualisations. Matplotlib makes difficult things possible and simple things easy
- **Numpy version 1.23.1:** a library for working with arrays in Python. Additionally, it has matrices, fourier transform, and functions for working in the area of linear algebra.
- **Opencv-python version 4.6.0.66:** a huge open-source image processing, machine learning, and computer vision library. Programming languages including Python, C++, Java, and many others are supported by OpenCV. It can analyse pictures and movies to find items, faces, or even human handwriting.
- **Scikit-image version 0.19.3:** an open-source image processing library for the Python programming language. It contains algorithms for feature identification, analysis, filtering, morphology, segmentation, geometric transformations, colour space manipulation, and more. It is made to work with Python's NumPy and SciPy scientific and mathematical libraries.
- **Pandas version 1.4.3:** a Python-based open source data analysis and manipulation tool that is quick, strong, adaptable, and simple to use.
- **Scikit-learn version 1.1.1:** the best and most reliable Python machine learning library. Through a Python consistency interface, it offers a variety of effective tools for statistical modelling and machine learning, including classification, regression, clustering, and dimensionality reduction. This library is based on NumPy, SciPy, and Matplotlib and was written primarily in Python.

- **Garbage collector:** An interface to the optional garbage collector is provided by this module. The collector may be turned off, the collection frequency can be adjusted, and debugging parameters can be configured. Additionally, it gives access to things that the collector discovered but was unable to liberate
- **Keras version 2.9.0:** a Python interface for artificial neural networks provided by an open-source software package. For the TensorFlow library, Keras serves as an interface.
- **Tensorflow version 2.9.1:** an artificial intelligence and machine learning software library that is free and open-source. It can be used for a variety of applications, but focuses particularly on deep neural network training and inference.
- **Google colab:** allows writing and executing Python in browser.

3.2. Methods

3.2.1. Data preparation

In this study, the original dataset is downloaded and stored in the storage directory of the USTH ICT Laboratory server as well as in Google drive storage. The used dataset for training and validating the model in this study is manually filtered for only RGB images with mask and color (groundtruth images) directories included.

The used dataset folder structure is shown in the Figure 3.2.1.1. as follow:

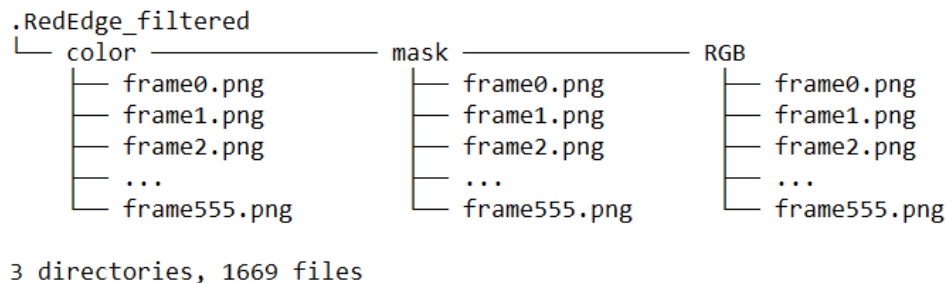


Figure 3.2.1.1. Dataset used structure

The images of the dataset in use are 480 pixels wide by 360 pixels high. The bit depth for the RGB and groundtruth images is 24 whereas the bit depth for the mask images is 8. During the training period, the groundtruth masks and RGB input images are scaled to 256 by 256 pixels.

Figure 3.2.1.2 below shows the examples of a RGB image, mask image and groundtruth image respectively.

In the RGB image of Figure 3.2.1.2, the target field is indicated by the black area, and the remaining portion contains the ground base, which is represented by the color brown, crop, and weed, which is represented by the color green, making it simple to understand the realistic context of the field; The black color of the mask image indicates where the field region is, while the white one indicates a position outside the context; The

ground base is removed from the color or groundtruth image, indicated by the black interval, while the crop and weed are given the colours of green and red, respectively.

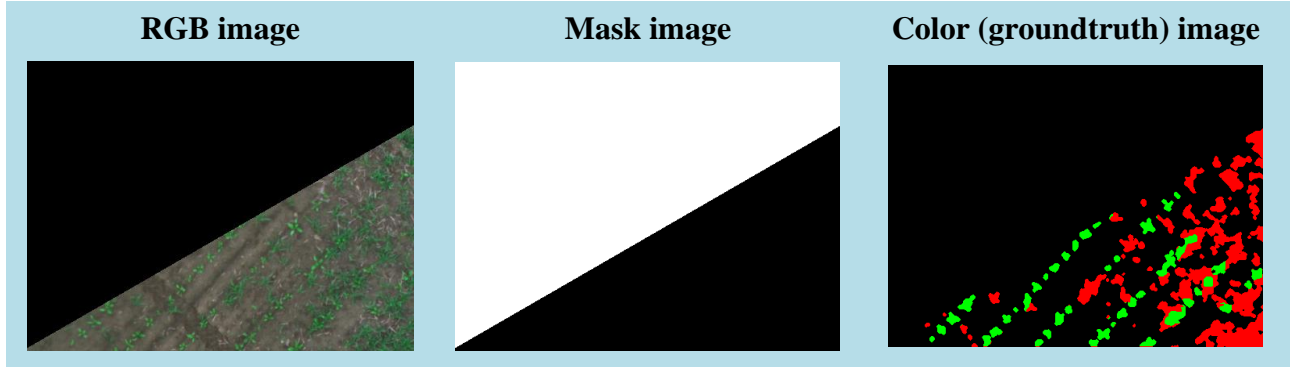


Figure 3.2.1.2. Example of filtered dataset

3.2.2. UNET Architecture and Training

A convolutional neural network called U-Net [9] was first introduced for biological image segmentation at the University of Freiburg's Computer Science Department. With a tweaked and extended architecture that allows it to operate with less training photos and provide more accurate segmentation, it is based on fully convolutional neural networks.

The main idea is to employ a contracting network followed by an expanding network, where upsampling operators in place of pooling operations in the expansive network. The output's resolution is raised by these layers. A large convolutional network may also learn to put together a precise output depending on encoded data.

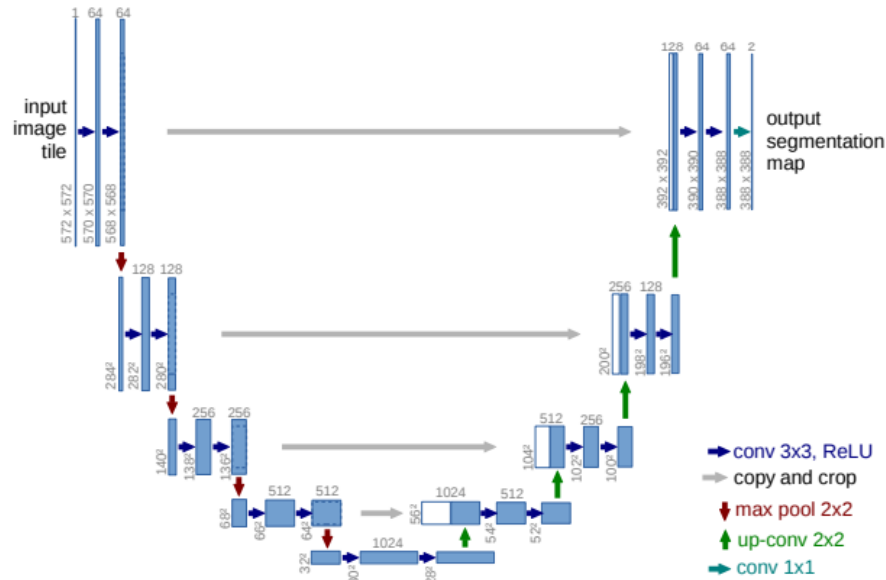


Figure 3.2.2.1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations. [9]

The network has a U-shaped architecture because it has expanding and contracting paths (left and right, respectively). The contracting path uses a standard convolutional network with repeated convolutions, Rectified Linear Units (ReLU), and max-pooling operations after each convolution. Spatial information is decreased while feature information is boosted during the contraction. Through a series of up-convolutions and concatenations with high-resolution features from the contracting pathway, the expansive pathway merges the features and spatial information.

In the original paper [9], the input picture is 572x572x3, but this session is going to use one that's 128x128x3. As a result, the size may vary from that in the original paper at different locations, but the essential elements will always be the same.

Below is the detailed explanation of the architecture:

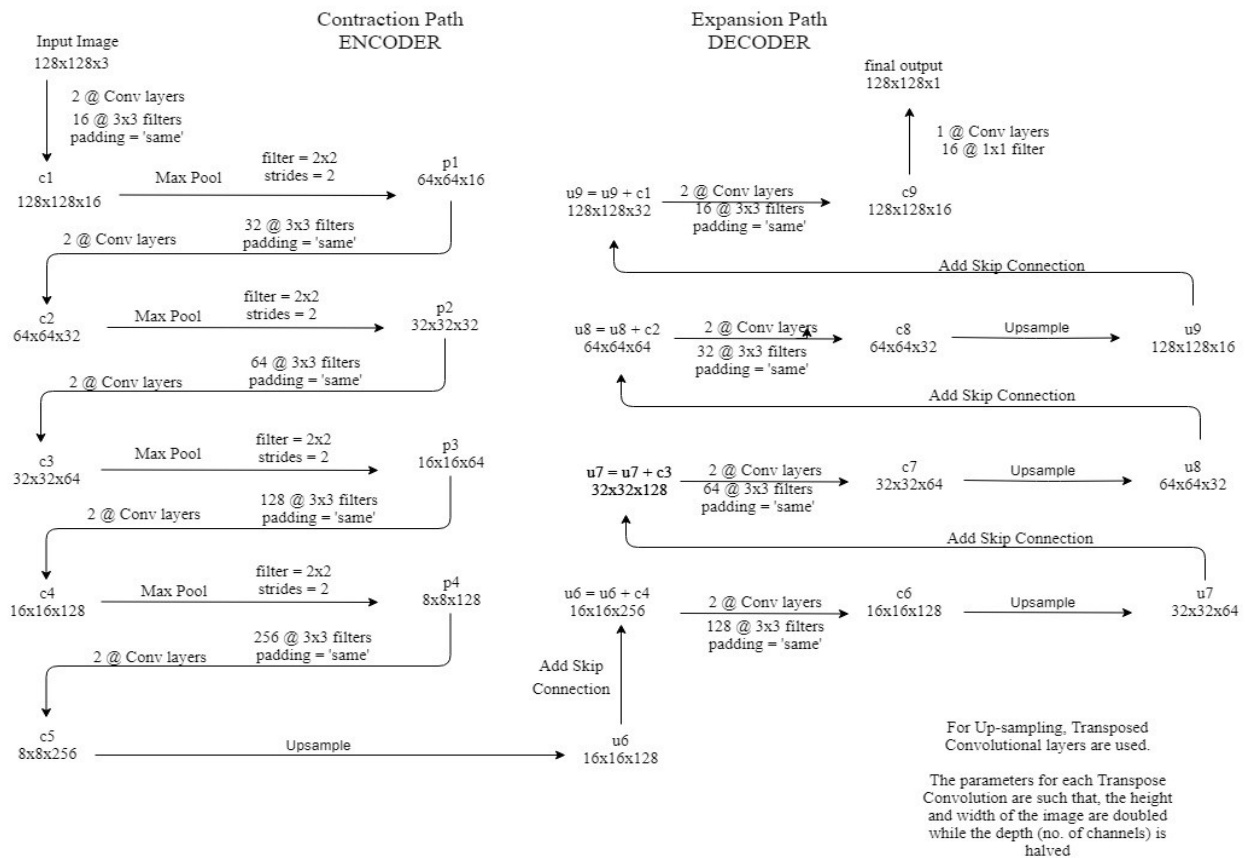


Figure 3.2.2.2. Detailed UNET Architecture ⁶

Points to note:

- 2 @ Conv layers means that two consecutive Convolution Layers are applied.
- c1, c2, ..., c9 are the output tensors of Convolutional Layers
- p1, p2, p3 and p4 are the output tensors of Max Pooling Layers
- u6, u7, u8 and u9 are the output tensors of up-sampling (transposed convolutional) layers

⁶ <https://medium.com/@madeshselvarani/unet-model-with-keras-2020-c6de560cac8b>

- The left-hand side is the contraction path (Encoder) where regular convolutions and max pooling layers are applied.
- In the Encoder, while the depth steadily increases, the size of the image gradually decreases, starting from 128x128x3 to 8x8x256.
- This basically means the network learns the image's "WHAT" information, but it has forgotten the "WHERE" information.
- The right-hand side is the expansion path (Decoder) where transposed convolutions along with regular convolutions are applied.
- In the Decoder, while the size steadily increases, the depth of the image gradually decreases, starting from 8x8x256 to 128x128x1.
- By gradually using up-sampling, the Decoder intuitively retrieves the "WHERE" information (precise localization).
- In order to obtain more accurate locations, connections are skipped at each stage of the decoder by joining the feature maps from the encoder at the same level with the output of the transposed convolution layers:
 - $u_6 = u_6 + c_4$
 - $u_7 = u_7 + c_3$
 - $u_8 = u_8 + c_2$
 - $u_9 = u_9 + c_1$
 - Again, performing two successive regular convolutions after each concatenation so that the model might learn to assemble a more exact output.
- The design is thus given a symmetric U-shape, hence the name UNET.
- On a high level, the relationship is:
 - Input (128x128x1) \Rightarrow Encoder \Rightarrow (8x8x256) \Rightarrow Decoder \Rightarrow Output (128x128x1)

3.2.3. Model optimization

Adam Optimizer

Adaptive Moment Estimation [1] is an algorithm for optimization technique for gradient descent. When dealing with complex problems requiring a huge number of variables or data, the approach is incredibly effective. It is productive and needs little memory. It appears to be a hybrid of the RMSProp method and the gradient descent with momentum algorithm.

Adam optimizer involves a combination of two gradient descent methodologies:

- **Momentum:** By using the "exponentially weighted average" of the gradients, this approach is used to speed up the gradient descent algorithm. The technique converges faster to the minima when averages are used.

$$w_{t+1} = w_t - \alpha m_t$$

where,
$$m_t = \beta m_{t-1} + (1 + \beta) \left[\frac{\delta L}{\delta w_t} \right]$$

- m_t : aggregate of gradients at time t [current]
(initially, $m_t = 0$)
- m_{t-1} : aggregate of gradients at time $t-1$ [previous]
- W_t : weights at time t
- W_{t+1} : weights at time $t+1$
- α_t : learning rate at time t
- δL : derivative of Loss Function
- δW_t : derivative of weights at time t
- β : moving average parameter (const, 0.9)

- **Root Mean Square Propagation (RMSP)**: Root mean square prop or RMSprop is an adaptive learning algorithm that tries to improve AdaGrad. It uses the "exponential moving average" as opposed to AdaGrad's method of computing the cumulative sum of squared gradients.

$$w_{t+1} = w_t - \frac{\alpha_t}{(v_t + \varepsilon)^{1/2}} * \left[\frac{\delta L}{\delta w_t} \right]$$

where,
$$v_t = \beta v_{t-1} + (1 - \beta) * \left[\frac{\delta L}{\delta w_t} \right]^2$$

- w_t : weights at time t
- w_{t+1} : weights at time $t+1$
- α_t : learning rate at time t
- δL : derivative of Loss Function
- δW_t : derivative of weights at time t
- v_t : sum of square of past gradients. [i.e $\sum(\delta L/\beta W_{t-1})$] (initially, $V_t = 0$)
- β : Moving average parameter (const, 0.9)
- ε : A small positive constant (10^{-8})

NOTE: Time (t) could be interpreted as an Iteration (i).

In order to provide a more optimised gradient descent, Adam Optimizer builds on the advantages or positive characteristics of the previous two approaches as shown in Figure 3.2.3.1. below.

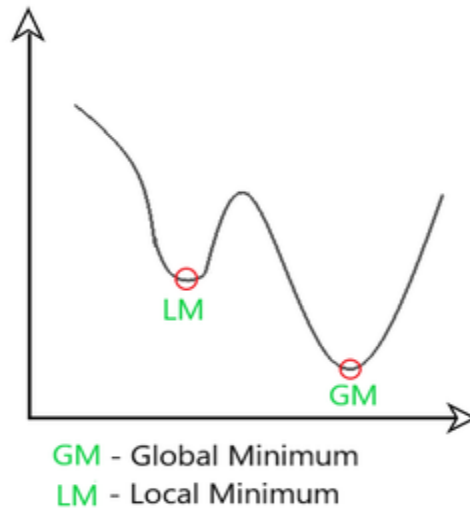


Figure 3.2.3.1. Global and Local Minimum ⁷

The rate of gradient descent is adjusted to attain the global minimum with the least amount of oscillation possible while progressing through the local minimum obstacles with large enough steps (step-size). Consequently, utilising the advantages of the aforementioned techniques to effectively meet the global minimum.

Using the formulas from the two approaches mentioned above, it is able to determine:

$$\begin{aligned}
 m_t &= \beta_1 m_{t-1} + (1 - \beta_1) \left[\frac{\delta L}{\delta w_t} \right] v_t \\
 &= \beta_2 v_{t-1} + (1 - \beta_2) \left[\frac{\delta L}{\delta w_t} \right]^2
 \end{aligned}$$

ε : a small +ve constant to avoid 'division by 0' error when $(v_t \rightarrow 0)$. (10-8)

β_1, β_2 : decay rates of average gradients in the above two methods. ($\beta_1 = 0.9$ & $\beta_2 = 0.999$)

α : Step size parameter / learning rate (0.001)

⁷ <https://www.geeksforgeeks.org/intuition-of-adam-optimizer/>

Based on the aforementioned approaches, m_t and v_t were both initialised as 0, hence it is seen that they become "biased towards 0" because both β_1 & $\beta_2 \approx 1$. By calculating "bias-corrected" m_t and v_t , this optimizer resolves the issue. In order to avoid large oscillations when close to the global minimum, this is also done to manage the weights as they approach it. The formulas used are:

$$\widehat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad \widehat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

Intuitively, the gradient descent after every iteration is adapted so that it remains controlled and unbiased throughout the process, hence the name Adam.

At this place, instead of normal weight parameters m_t and v_t , take the bias-corrected weight parameters $(\widehat{m}_t)_t$ and $(\widehat{v}_t)_t$. By including them in the larger calculation, it is possible to obtain:

$$w_{t+1} = w_t - \widehat{m}_t \left(\frac{\alpha}{\sqrt{\widehat{v}_t} + \epsilon} \right)$$

Building on the advantages of earlier models, Adam optimizer performs far better than those models and outperforms them in producing an optimised gradient descent. The graphic below clearly demonstrates how Adam Optimizer performs significantly better than the rest of the optimizers in terms of training cost (low) and performance (high).

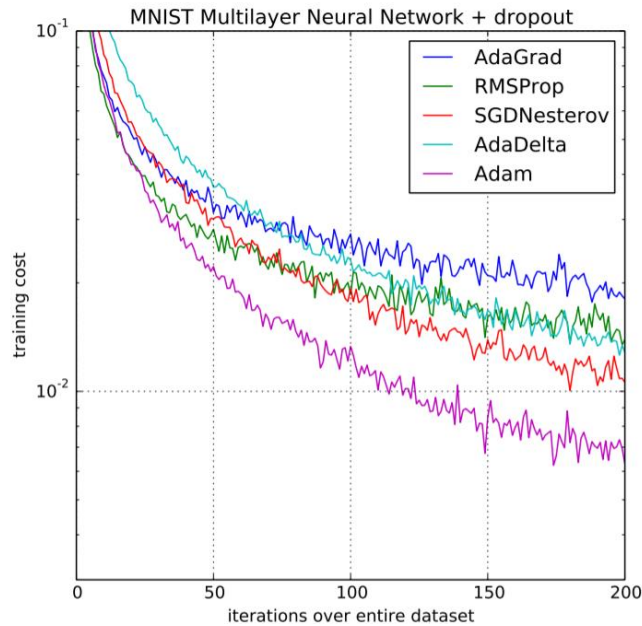


Figure 3.2.3.2. Performance Comparison on Training cost ⁸

⁸ <https://www.geeksforgeeks.org/intuition-of-adam-optimizer/>

Loss function

Assume that each output pixel in the image has the value p . It can be used to define the investigated loss functions in this situation as follow:

IoU Loss (Jaccard index): Intersection over Union loss or IoU loss [15], which has fewer hyperparameters than the other options, is the choice for imbalanced segmentation. But let's first get to know this metric in equation in Figure 3.2.3.3.:

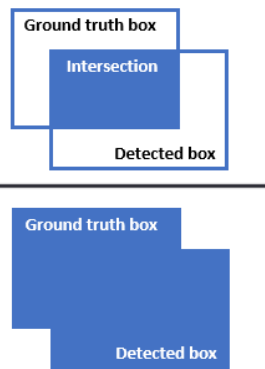
$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} = \frac{\text{Intersection}}{\text{Union}}$$


Figure 3.2.3.3. Intersection over Union ⁹



Figure 3.2.3.4. IoU evaluation ¹⁰

The predicted and ground-truth mask overlap acts as both the nominator and the denominator in the equation. By dividing these two figures, the IoU is determined, with values closer to one denoting more precise forecasts. As shown in the Figure 3.2.3.4 above, the higher IoU score, the better accuracy of the model.

The IoU has a value between 0 and 1, and since the goal of optimization is to maximise it, the loss function is defined as follows:

$$\text{IoU Loss} = 1 - \text{IoU}$$

⁹ <https://www.baeldung.com/cs/object-detection-intersection-vs-union>

¹⁰ <https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>

Dice Coefficient Loss (F1 Score): The Dice Coefficient loss [12] value is 1 minus Dice Coefficient which is calculated by dividing 2 times of the Area of Overlap to the total number of pixels in both images as the equation shown in the Figure 3.2.3.5. below:

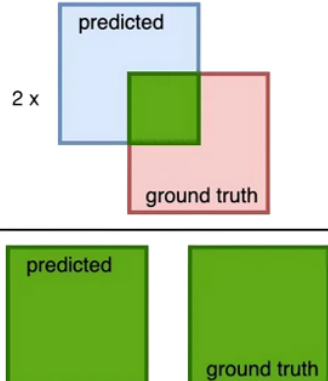
$$\text{Dice coefficient} = \frac{2 \times \text{area of overlapped (green)}}{\text{total area (green)}} = \frac{2 \times \text{area of overlapped (green)}}{\text{area of predicted (blue)} + \text{area of ground truth (red)} + \text{area of overlapped (green)}}$$


Figure 3.2.3.5. Dice Coefficient ¹¹

The IoU and the Dice coefficient are quite similar. They are positively associated, thus if one claims that model A is superior to model B at picture segmentation, the other will also claim the same. They both have a range of 0 to 1, like the IoU, with 1 denoting the greatest resemblance between predicted and truth.

In this study, the model is compiled with Adam optimizer and Intersection over Union (IoU) is used to measure the accuracy of weed detection. Otherwise, Keras callbacks are used to implement the learning rate decay if the validation loss does not improve for 5 continuous epochs, Early stopping if the validation loss does not improve for 10 continuous epochs with the batch size of 32.

It should be noted that there may be plenty of opportunity to fine-tune these hyperparameters and enhance the model's performance.

IV/ RESULTS AND DISCUSSIONS

The model is being trained with the RedEdge_filtered dataset (filtered 2018 Weed Map Dataset), which contains 1669 files and 3 directories. The dataset used is divided into two distinguishing sets with 1500 images for training and 169 images for validating the model.

4.1. Results

¹¹ <https://dev.to/erol/3d-brain-segmentation-with-unet-brats19-4cmj>

4.1.1. Evaluation metrics

The model is evaluated with IoU and Dice Coefficient scores.

As shown in the Figure 4.1.1.1. below, it is possible to see that the IoU score starts at around 0.0500, and gradually grows up until the epoch of 40. The model then became more stable starting from the epoch of 45, where the IoU score slightly increased as well as fluctuated from nearly and around 0.7000 up to 0.7679. This means that the model is not in a situation of overfitting or underfitting. On the other hand, the loss score is the same as IoU accuracy but with the opposite trend.

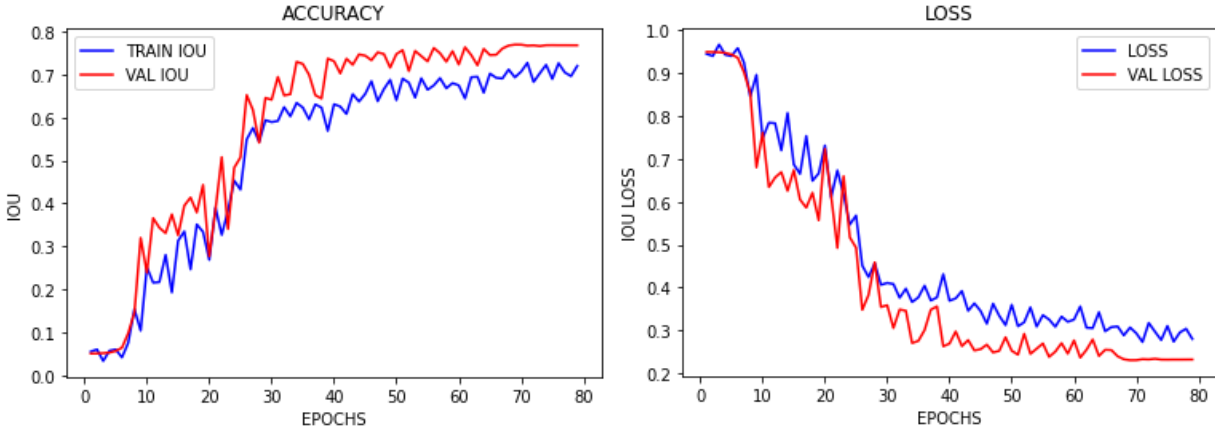


Figure 4.1.1.1. Model IoU accuracy and loss

On the left-hand side, the Figure 4.1.1.2. below shows the Dice Coefficient accuracy and loss diagram when training the model. Similar to IoU above, the Dice Coefficient score starts from around 0.0500 and peaks up to 0.7495 after becoming more stable at the middle of the training process. The Dice Coefficient loss is the same but with the opposite trend.

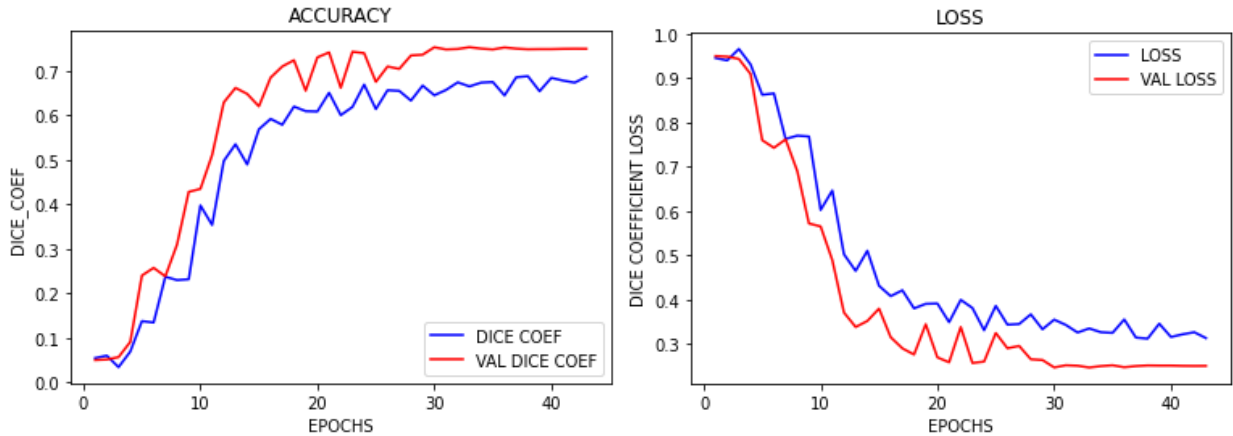


Figure 4.1.1.2. Model Dice Coefficient accuracy and loss

4.1.2. Testing with actual images

This study uses a few photos to assess the model's actual accuracy when input only consists of RGB images. The result is shown in the Table 4.1.2.1. below:


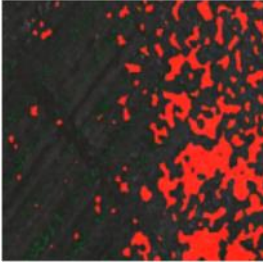
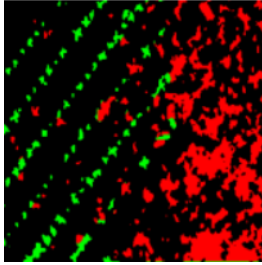
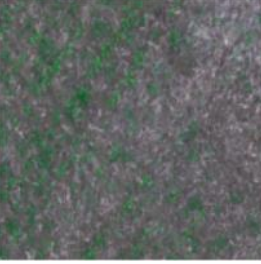
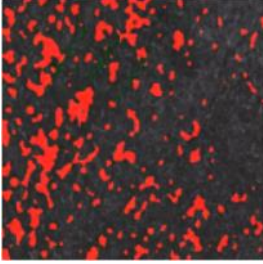
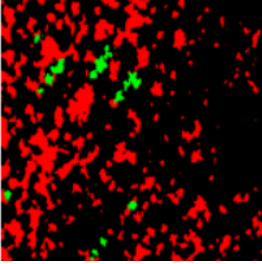
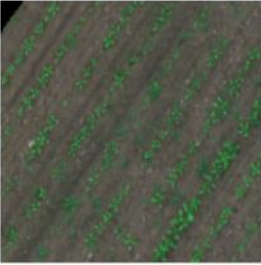
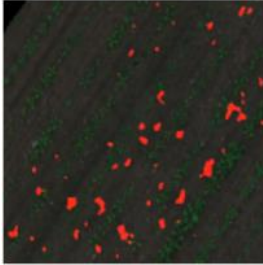
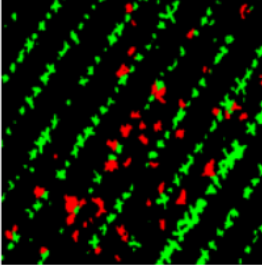
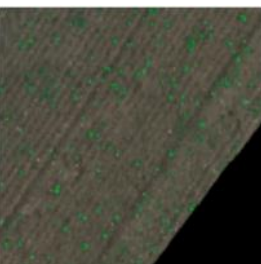
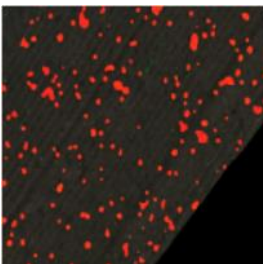
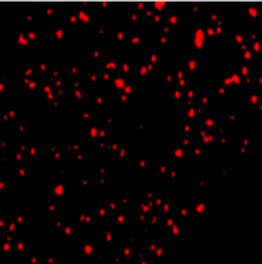


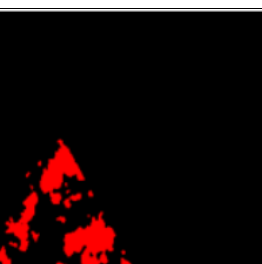
RGB	Predicted mask	groundtruth	IoU
			0.6945
			0.7121
			0.6217
			0.7339
			0.8691
Average IoU = 0.7262			

Table 4.1.2.1. Testing images

4.2. Discussion

In this study, the IoU accuracy of the model is around 0.7679. Table 4.2.1. below compares the best result of this study using UNet model with the existing works in the literature.:

Model	Device	Mean IoU (%)
The proposed UNet model (this study)	Tesla K80	76.79
FCN-Alexnet by Jizhong et al. [2]	GTX 1060	70.50
VGGNet-FCN by Simonyan et al. [11]	GTX 1060	72.80
GoogLeNet-FCN by Szegedy et al. [13]	GTX 1060	71.30
ResNet-FCN by He et al. [4]	GTX 1080 TI	77.20

Table 4.2.1. Model accuracy comparison

Compare to the 4 bottom models in Figure 4.2.1 above, the accuracy of this study's UNet model (76.79%) is better than FCN-Alexnet (70.50%), VGGNet-FCN (72.80%) and GoogLeNet-FCN (71.30%). This accuracy score may be considered as a pretty good starting point in terms of Crop/Weed image segmentation.

V/ CONCLUSION

Precision agriculture, which uses new technologies and inventions in farming to improve performance, has advanced significantly along with technological development. The organisation, growth in yield (both in number and quality), and rise in profit all benefit from precision agriculture, which also has a favourable impact on the performance of the agricultural sector.

In this thesis, we investigated and found a solution for the issue of crop/weed segmentation on UAV photos. Using three primary stages that include data preparation, model training, model testing, and model evaluation, the problem is implemented after researching precision agriculture, agriculture UAV photos, and segmentation approaches. The UNet base model with Python and the RGB input images are implemented during the

training model stage. Finally, the model testing and IoU score evaluation of the results are carried out.

The collected findings demonstrate that the trained model for crop/weed segmentation with 500 RGB images achieves the accuracy of about 76.79 percent after undergoing all three steps. The results of this internship study may obtain a very excellent point among the surveys, according to a comparison table we generate between the results and some of the publications we surveyed. It can be viewed as a significant first step in the use of UAVs in precision agriculture.

Only the Deep Learning UNet architecture was investigated and deployed due to the limitations of this internship. The system will be enhanced in the future for better accuracy and performance, and the size of the dataset utilised will also be raised. The model will also be trained using more datasets (s). Added to various model designs to discover the optimum approach to the crop/weed segmentation challenge. Additionally, following the improvements, the best model will be used in real-time segmentation of footage taken from a genuine UAV, which may have a favourable impact on the advancement of precision agriculture.

REFERENCES

1. Bock, S., Goppold, J., & Weiß, M. (2018). An improvement of the convergence proof of the ADAM-Optimizer. arXiv preprint arXiv:1804.10587.
2. Deng, J., Zhong, Z., Huang, H., Lan, Y., Han, Y., & Zhang, Y. (2020). Lightweight semantic segmentation network for real-time weed mapping using unmanned aerial vehicles. *Applied Sciences*, 10(20), 7132.
3. Goodman, W., & Minner, J. (2019). Will the urban agricultural revolution be vertical and soilless? A case study of controlled environment agriculture in New York City. *Land use policy*, 83, 160-173.
4. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).
5. Jarvis, R. A. (1983). A perspective on range finding techniques for computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2), 122-139.
6. Kussul, N., Lavreniuk, M., Skakun, S., & Shelestov, A. (2017). Deep learning classification of land cover and crop types using remote sensing data. *IEEE Geoscience and Remote Sensing Letters*, 14(5), 778-782.
7. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.
8. Mazzia, V., Comba, L., Khaliq, A., Chiaberge, M., & Gay, P. (2020). UAV and machine learning based refinement of a satellite-driven vegetation index for precision agriculture. *Sensors*, 20(9), 2530.
9. Ronneberger, O., Fischer, P., & Brox, T. (2015, October). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (pp. 234-241). Springer, Cham.
10. Sa, I., Popović, M., Khanna, R., Chen, Z., Lottes, P., Liebisch, F., ... & Siegwart, R. (2018). WeedMap: A large-scale semantic weed mapping framework using aerial multispectral imaging and deep neural network for precision farming. *Remote Sensing*, 10(9), 1423.
11. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.
12. Sudre, C. H., Li, W., Vercauteren, T., Ourselin, S., & Jorge Cardoso, M. (2017). Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *Deep learning in medical image analysis and multimodal learning for clinical decision support* (pp. 240-248). Springer, Cham.

13. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).
14. United-Nation (2020) Growing at a slower pace, world population is expected to reach 9.7 billion in 2050 and could peak at nearly 11 billion around 2100.
15. van Beers, F., Lindström, A., Okafor, E., & Wiering, M. A. (2019, February). Deep Neural Networks with Intersection over Union Loss for Binary Image Segmentation. In ICPRAM (pp. 438-445).
16. van Loon, M. P., Hijbeek, R., Ten Berge, H. F., De Sy, V., Ten Broeke, G. A., Solomon, D., & van Ittersum, M. K. (2019). Impacts of intensifying or expanding cereal cropping in sub-Saharan Africa on greenhouse gas emissions and food security. *Global change biology*, 25(11), 3720-3730.
17. Yang, S., Gu, L., Li, X., Jiang, T., & Ren, R. (2020). Crop classification method based on optimal feature selection and hybrid CNN-RF networks for multi-temporal remote sensing imagery. *Remote Sensing*, 12(19), 3119.

