Faculty of Engineering and the Built Environment,
Department of Mechanical Engineering Science

A
Masters Research Dissertation
Titled

Development of a Smart Weed Detector and Selective Herbicide Sprayer

By

Ukaegbu Uchechi Faithful

Submitted in Fulfilment of The Requirements for the Completion of Study Towards the Postgraduate Degree - M.ENG.
In
Mechanical Engineering

Supervisor: Professor L.K. Tartibu
Co-Supervisor: Dr. O.T. Laseinde

Date of Submission: January 4, 2021

Dedication

To God Almighty, the source of my strength, my close confidant and a faithful Father.

Declaration of Originality

I, Ukaegbu Uchechi Faithful, hereby declare that this master's research dissertation is wholly my own work and has not been submitted anywhere else for academic credit either by myself or another person. I understand what plagiarism implies and declare that this dissertation is my ideas, words, phrase, arguments, graphics, figures, results, and organization except where reference is explicitly made to another work. I understand further that any unethical academic behavior, which includes plagiarism, is seen in a serious light by the University of Johannesburg and is punishable by disciplinary action as stipulated by the university rules and regulations.

Full name: …UKAEGBU UCHECHI FAITHFUL….

Student No: …219120451……

Signature: ……

Date: ……4/01/2021……

Acknowledgement

It is with utmost joy I express my profound gratitude to God Almighty for His loving kindness, goodness and guidance. I specially thank my mentor and spiritual father, the Most Rev Prof. Daddy Hezekiah, the International Mayor of Peace for his selfless love, unending sacrifices and fervent prayers on my behalf. There were moments during the course of this research when it seemed I had lost direction but his words to me 'positive efforts yields positive results' kept me going.

I remain ever grateful to my amiable research supervisor, Prof L.K Tartibu for providing his invaluable support both morally and academically in the pursuit of my research idea-I couldn't have asked for a better supervisor. To my co-supervisor, Dr. O.T Laseinde, I say a big thank to you for your concern and advice during the course of this research.

Also, I will not fail to acknowledge my late friend and colleague, Mr. Emmanuel Ndubuisi Asomugha. He was a promising and intelligent young man who shared a similar research dream as mine. I mentioned him here to show that he is remembered.

Furthermore, I would like to thank my loving parents, Engr. H.N Ukaegbu for their immeasurable contributions as regards my academic pursuit. I also thank the family of Mr. and Mrs. Nnamdi Iwenofu for their invaluable support and assistance during my days as a masters student. I will not forget to thank my wonderful siblings; Amara, Nenye and Victor; my friends and Ijeoma Okafor for their moral support.

Finally, I would like to appreciate the department of Mechanical and Industrial Engineering, University of Johannesburg, South Africa for the wonderful platform to carry out this research study. I am also grateful to the University Research Committee (URC) and the Global Excellence and Stature (GES) scholarship board for their financial contributions to the realization of this research work.

Abstract

The fourth industrial revolution has brought about tremendous advancements in various sectors of the economy including the agricultural domain. Aimed at improving food production and alleviating poverty, these technological advancements through precision agriculture has ushered in optimized agricultural processes, real-time analysis and monitoring of agricultural data. The detrimental effects of applying agrochemicals in large or hard-to-reach farmlands and the need to treat a specific class of weed with a particular herbicide for effective weed elimination gave rise to the necessity of this research work.

This research study involved the real-time detection of weeds and selective spray of herbicides on broadleaf and grass weed by applying deep learning algorithms on an embedded system. The deep learning algorithm deployed was a convolutional neural network model re-trained through transfer learning on the ResNet50 model using the soybean weed dataset and evaluated with the random forest classifier. The proposed smart selective herbicide sprayer was developed by assembling a quadcopter kit with the sprayer module and embedded system incorporated.

The convolutional neural network model trained yielded training and validation accuracies of 99.98% and 98.4% respectively thus outperforming the random forest classifier. Also, training and validation losses of 0.0039 and 0.0323 respectively were obtained giving a clear indication that the model was appropriate. The test conducted on the smart herbicide sprayer, weighing about 3kg, indicates that the proposed models was able to detect broadleaf and grass weeds accurately and selectively spray herbicides on the weeds in less than a second. The results obtained from this work create the opportunity for more research to be carried out in this field of precision agriculture as regards weed and pest detection.

Table of Contents

List of Figures

List of Tables

Nomenclature

Symbol	Description
x_1, x_2, x_3	Input data
w_1, w_2, w_3	Weights
b	Bias
y	Weighted sum
z	Output
T_i	True output of the ith sample
Z_i	Estimated output of the ith sample
m	Number of outputs generated
γ	Learning rate
h	Hidden layer
x	Visible layer
W_h	Hidden weight
W_y	Output weight
W_x	Input weight
C	Cell value
σ	Sigmoid function
H	Output value of the hidden layer
A	Ampere
V	Volt
N	Newton

List of abbreviations

UAV	Unmanned Aerial Vehicle
SVM	Support Vector Machines
DL	Deep learning
CNN	Convolutional Neural Network
RF	Random forest
GPS	Global Positioning System
ESC	Electronic speed controller
ILSVRC	Imagenet Large Scale Visual Recognition Challenge
GPIO	General Purpose Input/output
DC	Direct current
VTOL	Vertical Take-off and Landing
PWM	Pulsating Width Modulation
Li-S	Lithium-Sulphur
Li-Po	Lithium-Polymer
Tanh	Hyperbolic tangent
ReLU	Rectified Leaky Unit
AE	Autoencoder
RBM	Restricted Boltzmann Machines
DBN	Deep Belief Networks
DNN	Deep Neural Network
GAN	Generative Adversarial Network
ANN	Artificial Neural Network

RNN	Recurrent Neural Network
BPTT	Backpropagation through Time
LSTM	Long Short-Term Memory
KNN	K-Nearest Neighbor
RGB	Red Green Blue
SD	Secure Digital
HDMI	High Definition Multimedia Interface
USB	Universal Serial Bus
CSI	Camera Serial Interface
DSI	Display Serial Interface
LCD	Liquid Crystal Display
CUDA	Compute Unified Device Architecture
NIR	Near-Infrared
GPU	Graphical Processing Unit
FPGA	Field Programmable Gate Array
ASIC	Application Specific Integrated Circuit
MAC	Multiply and Accumulate
TPU	Tensor Processing Unit
MAV	Micro Aerial Vehicle
WRN	Wide Residual Network
ARM	Advanced RISC Machine
YOLO	You Only Look Once
LED	Light Emitting Diode

CPU	Central Processing Unit
HOG	Histogram of Oriented Gradients
COCO	Common Object in Context
NOOBS	New out-of-the Box
PC	Personal Computer
IP	Internet Protocol
SFTP	Secure File Transfer Protocol
VCC	Voltage Common Collector

Chapter 1-Introduction

1.1 Research background

The task of obliterating weeds on farmland has been a challenge faced by farmers hence this research focuses on proffering solutions aligned with the fourth industrial revolution, to the nodus of weeds and its attendant consequences. In a bid to increase food production by 2050 to feed a population of about 9 billion, the concept of precision agriculture through the application of technological solutions is imperative.

According to Kamilaris and Prenafeta-Boldú (2018), precision or smart agriculture involves optimizing agricultural processes, real-time analysis, and monitoring of agricultural data. It also involves automation, making intelligent predictions based on algorithms and various approaches aimed to maximize economic returns while preserving resources, and protecting the environment (Ojha, Misra, and Raghuwanshi, 2015). The word "smart" according to Miranda *et al.,* **(**2019**)** relates to the ability of a system to integrate control and actuation tasks in analyzing situations to make real-time decisions given available data. Unmanned Aerial Vehicles(UAV), also known as drones, are one of the most economically important sectors of precision agriculture as they find application in weed detection, soil and field analysis, crop monitoring, nutrient deficiency detection, crop spraying, aerial planting, etc. (Maes and Steppe, 2019). Furthermore, the application of computer vision technology and machine learning in areas of image analysis and prediction, as regards weed, pest, and nutrient deficiency detection, has grown in recent years to meet the ever-increasing demand for fast and precise methods of monitoring. When applied to drones, they are used to collect data that could be analyzed and used to provide insights to improve yield. The widely used techniques for analyzing images include machine learning (K-means, Support Vector Machine (SVM), artificial neural networks (ANN), deep learning (DL), etc.), wavelet-based filtering, vegetation indices, and regression analysis. Of all the aforementioned techniques, deep learning is gaining momentum because of its high performance and precision level (Mohanty, Hughes, and Salathé, 2016).

Deep learning algorithms are mostly theories and hence the implementation of these algorithms requires a hardware-software interface known as embedded systems. Embedded systems (e.g. Nvidia Jetson TX series, Raspberry Pi, Intel Edison, etc.) are elucidated according to Jha *et al.* (2019) as hardware built systems comprising of memory chips with custom software programmed

on it and their applications could be seen in driverless cars (Bojarski *et al.*, 2016), monitoring systems (Rao and Sridhar, 2018), weed detection (Vikhram *et al.*, 2018), etc. Furthermore, real-time inferencing of deep learning algorithms using embedded systems could be carried out either on a cloud server or on-board (Hadidi *et al.*, 2018).

Weeds are unwanted plants which compete with crops for water, light, nutrient, space and also increases the risk of pest and disease outbreak on farmland. This has caused adverse effects on crop yield, growth, and development. Also, traditional weed control methods which involve manual removal of weeds and the use of herbicides on the whole agricultural field is labor-intensive, a waste of herbicides, and also a source of environmental pollution. Several pieces of research have been conducted for detecting weeds using deep learning techniques and smart spraying of herbicides using other techniques deployed on embedded systems. According to Ferreira *et al.* (2017), the application of herbicides yields better results if the treatment is targeted to the specific class of weeds (grass or broadleaf). Hence, a convolutional neural network (CNN) was trained on the Caffe framework to detect broadleaf and grass weeds on soybean farmland. Alsalam *et al.* (2017) recommended the use of deep learning technique for more precise weed detection in their study and implemented a UAV that could change its planned path, approach a target and perform a specific action such as spraying of herbicides. Vikhram *et al.* (2018) implemented a smartweed detection and herbicide sprayer robot which detected weeds using morphological thresholding, erosion, and dilation. The detection algorithm was deployed on a raspberry pi 3B, which had a sprayer module attached to it to carry out a selective spray of herbicides.

However, none has been able to carry out a spray of a particular herbicide on a specific class of weed and also mount a DL inference-embedded device on a UAV. Hence, taking a leaf from the aforementioned reviews, the proposed study focuses on real-time detection of weeds and the selective spray of a particular herbicide on a specific class of weed(broadleaf or grass) using deep learning algorithms deployed on a raspberry pi 3B and mounted on a UAV. The following section describes the problem this research is addressing, point out the significance and the motivations behind this study, highlight the limitations and delimitations of this study and briefly discuss the approach adopted to conduct this research.

1.2 Rationale and Motivation

The motivation for this study is based on the fact that grass weeds are resistant to herbicides which are effective for a broadleaf weed hence the need to develop a smart sprayer that selectively sprays herbicides based on the type of weed.

1.3 Problem Statement

The spraying of agrochemicals on large farmlands or hard-to-reach areas has proven to be time-consuming, risky, and detrimental to human health. Also, excessive use of herbicides gives rise to waste, due to chemical residues, and emissions to the air. Furthermore, to make for an effective weed elimination, the need to treat a specific class of weed with a particular herbicide is imperative as suggested by Ferreira *et al.* (2017). This creates the need for a smartweed detector and selective herbicide quadcopter sprayer which sprays specific herbicides on a broadleaf weed and grass weed accordingly.

1.4 Research Questions

The questions this research sets out to answer are as follows:

1. How can a weed detection CNN model be trained on a pre-trained ResNet50 model?
2. Would a CNN model outperform a random forest classifier in weed classification?
3. What are the procedures for deploying a trained CNN model on a raspberry pi 3B?
4. What are the procedures for building a quadcopter?
5. What is the significance of developing a selective herbicide sprayer?

1.5 Research objectives

The proposed study aims to develop a smart selective herbicide sprayer. The objectives geared towards achieving this aim are:

- To train a CNN model using a pre-trained residual network(ResNet50) model on TensorFlow framework and compare its performance with a random forest (RF) classifier
- To deploy the trained model on a raspberry pi 3B and incorporate a sprayer module.
- To build a quadcopter from an already existing quadcopter kit.
- Mount the Raspberry Pi together with the sprayer module on the quadcopter and test run.

1.6 Significance of the research

The selective spray of herbicides would reduce environmental pollution and air emissions (which may contain greenhouse gases). The spray of herbicides according to the class of weed eliminate weeds thus improving crop yield and giving rise to a corresponding decrease in production cost. Also, it would be easy to access hard-to-reach areas, especially in large farms when a UAV is being used. Furthermore, it is expected that the proposed project would bring about an increase in food production thus alleviating poverty in Africa. Finally, the proposed study, which generally involves applying precision agriculture technologies in identifying and selective spraying of weeds would make for effective protection of crops in the field. This project could potentially contribute to the propulsion of Africa into the 4^{th} industrial revolution.

1.7 Research methodology

This section gives the procedures set out throughout the course of the project. A quantitative approach is employed to develop a quadcopter which differentiates between two classes of weeds using deep learning and sprays herbicides accordingly. Although chapter 3 provides a detailed methodology of this study, a summary is given as follows:

- **Preliminary study**: To gain insight to the research work, extensive literature review was conducted and two research papers were written. The first involved the application of deep learning on an embedded device in detecting potassium deficiency in red grape vines. While the second involved a review of hardware accelerators in the agricultural domain.
- **Building the quadcopter**: A quadcopter kit has been purchased with an emphasis on its weight and power specifications. The kit would consist of a lithium-polymer battery, brushless motors, electronic speed controllers (ESC), flight controller, Global Positioning System (GPS), frame, propellers, receiver and transmitter, and other accessories. These components would be assembled and tested.
- **Training the CNN model**: Soybean weed dataset which consists of soybean, grass weed, and broadleaf weed images has been obtained from a publicly available source, Kaggle. They were divided in a ratio of 7:2:1 into training, validation, and test dataset respectively. A total of 5349 images with an input shape of 224×224×3 and batch size of 20 has been used to train, through transfer learning, a CNN model. The ResNet50 model (winner of the

Imagenet Large Scale Visual Recognition Challenge (ILSVR) 2015) is the CNN model selected and by applying transfer learning, its last layer bearing the softmax activation function has been modified to a 3-neuron layer. The model has been trained on the TensorFlow framework for 10 epochs. Also, the datasets has been trained on the random forest classifier to evaluate the performance of the CNN model.

- **Weed detection and spraying actuation**: The trained model has been converted to the TensorFlow Lite format and deployed on a raspberry pi3 using the FileZilla client software. Python codes has been developed to access the General Purpose Input/output (GPIO) pins of the raspberry pi for a fixed time interval when a grass or broadleaf weed is detected by the raspberry pi camera. When a weed is detected, a specific GPIO pin sends voltage combined with a 5V from another GPIO pin to turn on a relay which in turn connects the spray pump to a 12V DC power source. The spray pump draws herbicide from the tank and pumps it through the nozzle. The system comprising the raspberry pi3 and the sprayer module have been incorporated on the assembled quadcopter for a final test-run.

1.8 Delimitation and Limitation

The proposed study sets out to build a quadcopter from an already existing quadcopter kit. A pre-trained ResNet50 model will be re-trained on a Tensorflow framework and deployed on a raspberry pi 3B to detect broadleaf and grass weeds. A sprayer module has been incorporated to spray accordingly.

1.9 Main contribution from this research

The main contributions of this research work are listed as follows:

- A smart herbicide sprayer for the real-time and selective spray of weeds has been developed.
- Publication of two research papers involving the application of DL on an embedded system and a review on DL hardware accelerators in agriculture. The references to the research papers are as follows:
    - Ukaegbu, U.F, Tartibu L.K, Laseinde T, Okwu M.O, Olayode I.O. (2020). ‘"A deep learning algorithm for detection of potassium deficiency in a red grapevine and spraying actuation using a raspberry pi3," *2020 International Conference on*

Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD), Durban, South Africa, 2020, pp.(1-6), doi: 10.1109/icABCD49160.2020.9183810.

- Ukaegbu U.F, Tartibu L.K and Okwu M.O.(2020).'Deep Learning Hardware Accelerators for High Performance in Smart Agricultural Systems: An Overview'. Proceedings of the 31st Annual South African Institute for Industrial Engineering Conference, South Africa Volume: ISSN: 2308-8265,pp (558-570).

1.10 Research Structure

This dissertation is structured into five chapters to comprehensively describe the research study undertaken. Chapter two provides a review of existing literature that used deep learning for weed classification and also executed deep learning algorithms on embedded systems. Also, it discusses theories such as the unmanned aerial vehicle, deep learning, computer vision, embedded systems, and sprayer module to aid a better understanding of the study.

Chapter three describes the application of deep learning on an embedded system by detecting potassium deficiency on a red grape vine and prompting a spraying actuation. This preliminary study gives a summary of the methodology and results obtained.

Chapter four is concerned with the materials used and methodology implemented in building the prototype. It describes how the image datasets were obtained and talked about the software used for training the CNN model. It further gives a breakdown of the raspberry pi 3B as well as its setup procedures. The calculations involved in selecting and the procedures for assembling the quadcopter components.

Chapter five analyzes and discusses the results obtained. It outlines the accuracies obtained from training the CNN model and RF classifier. It also points out some observations from carrying out a test-run on the prototype. Chapter six gives a summary of the achievements of this research work and possible recommendations for the improvement of the proposed work.

Chapter 2-Literature Review

2.1 Introduction

This chapter reviews related works focusing on the use of DL for weed control and application of DL on embedded systems. It starts by describing the unmanned aerial vehicle with an emphasis on the quadcopter. The concepts of deep learning and computer vision have been expounded together with an elucidation of the types of embedded systems and sprayer module.

2.2 The Unmanned Aerial Vehicle (UAV)

According to Yanushevsky (2011) as well as Chao, Cao, and Chen (2010), an unmanned aerial vehicle can be defined as a power-driven and reusable vehicle that flies without a human pilot on-board and could be navigated using a remote control or flown autonomously with the aid of an autopilot. The use of the UAV dates back to the 1950s when the Q-2 was built by Ryan Aeronautical and flown for military reconnaissance (Sullivan, 2006). The UAV ranges from nano, micro, or small to large scale aerial vehicles, and since it was developed for military purposes, developing or maintaining one was quite cost-prohibitive. However, the advent of high power density batteries and miniaturized equipment has made the UAV reasonably priced and in high demand, as it finds applications in areas of surveillance, photography, rescue mission, weather monitoring, precision agriculture, payload deliveries, etc. Also, it has important advantages of minimized operational cost, reduced human error as well as its ability to be operated under hazardous conditions (Vogeltanz, 2016). Based on its wing types (as shown in Fig. 2.1), the UAV can be classified into three categories:

1. Fixed-wing configuration
2. Rotary wing configuration
3. Flapping wing configuration

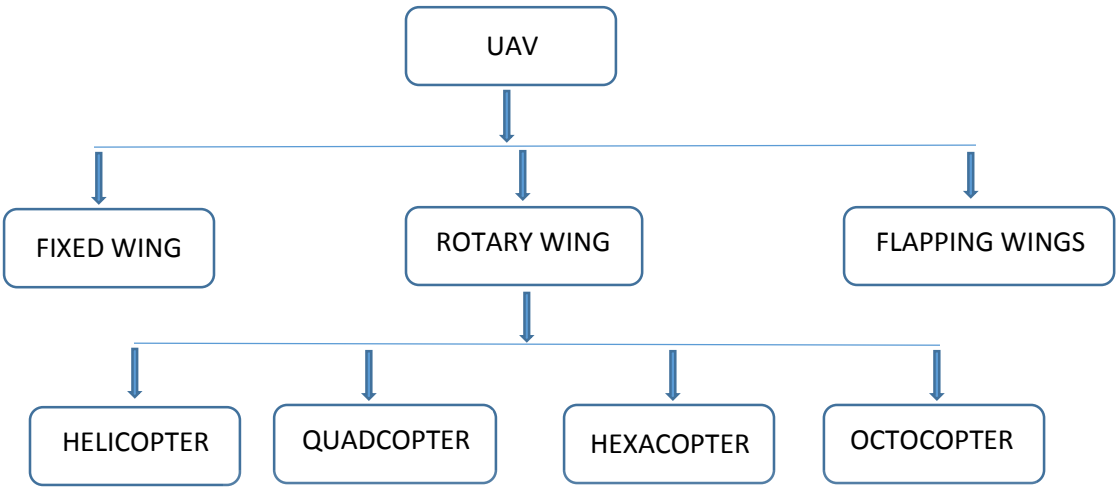


Fig. 2.1: Classification of a UAV

2.2.1 Fixed Wing Configuration

The fixed-wing configuration, as shown in Fig. 2.2, are UAVs with stationary wings made up of airfoils that create lift when a certain speed is attained (Marinello *et al.*, 2016). Similar to an airplane, they have six degrees of freedom and four main control surfaces which according to Zhao *et al.* (2018), are ;

i. Two ailerons- control the roll angle
ii. Elevator- controls the pitch angle
iii. Rudder- controls the yaw angle
iv. Throttle- controls the motor speed

They have a longer time of flight and can cover large distances however, they cannot hover over a place and are difficult to pilot (Probst, Pedersen and Dakkak-Arnoux, 2017; Hafsal, 2016).

Fig. 2.2: A Typical Fixed Wing UAV (adapted from Rioku, 2018)

2.2.2 Flapping Wing Configuration

Flapping wing UAVs try to mimic the mode of flight for insects and birds as could be seen in Fig. 2.3. They have low power consumption, are cost-effective, and can carry out vertical take-off and landing however they have a particularly low payload and endurance capabilities (Zhang, Liu, and Han, 2015; Guerrero *et al.*, no date).

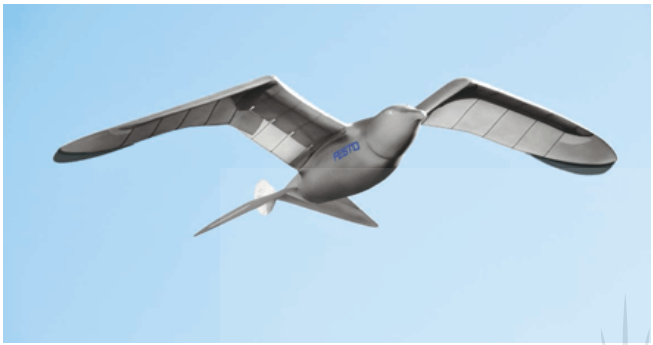


Fig. 2.3: A Typical Flapping Wing UAV (adapted from Mackenzie, 2012).

2.2.3 Rotary Wing Configuration

Rotary wing UAVs, also known as vertical take-off and landing (VTOL) rotorcrafts and are usually used on operations requiring hovering flight. They are more advantageous than their fixed-wing counterpart in terms of being less susceptible to turbulence, possessing higher degrees of freedom, lower flying speeds, and suitability for indoor usage (Shraim, Awada, and Youness, 2018). They are further categorized as helicopter, quadcopter, hexacopter, and octocopter. The helicopter has a single set of rotor blades attached to the central mast to produce thrust and an anti-torque tail rotor for change in direction (Ghazbi, 2016). It is quite difficult to control because according to Hoang and Poon (2013), it suffers from torque issues due to its main rotor. The quadcopter, hexacopter, and octocopter are UAVs with four, six, and eight rotors respectively. They are easy to pilot and can hover over a place however, a quadcopter is widely used due to its higher power-to-weight ratio, maneuverability, and cyclic design(Patel *et al.*, 2013; Garratt *et al.*, 2020). A typical rotary wing configuration is shown in Fig. 2.4.

Fig. 2.4: A Typical Rotary Wing UAV (adapted from Cannon, 2018)

2.3 Quadcopter

Hoang and Poon (2013) defined a quadcopter as an aerial vehicle controlled by the rotational speed of four rotors for lift, steering, and stability with VTOL. The first quadcopter, a manned vehicle though, was built in 1907 and as research progressed, there came the advent of the unmanned quadcopter by the Breguet brothers in the late 20th century (Garratt *et al.*, 2020). The quadcopter is structured such that a brushless motor, which drives each rotor, is connected to an ESC which in turn receives signals from the flight controller powered by a battery. Based on its rotor arrangement relative to its body co-ordinate system, a quadcopter has two main configurations: The cross and the plus configurations (shown in Fig. 2.5) with the former considered to be more stable than the latter (Zhang *et al.*, 2014).

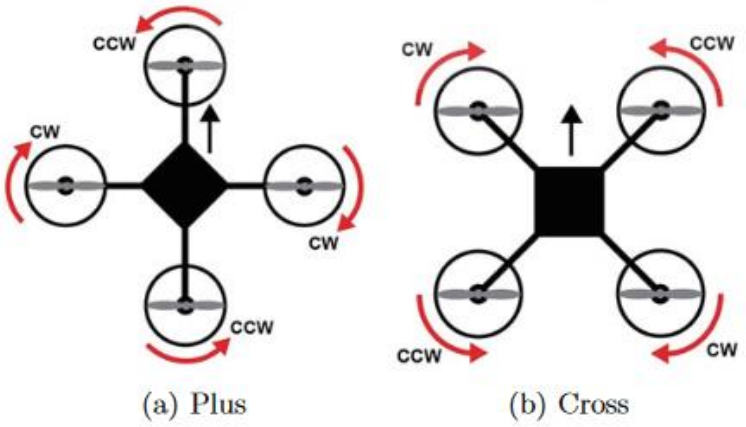


Fig. 2.5: Configurations of a Quadcopter (adapted from Maurya, Behera and Verma, 2019)

The four rotors are arranged in such a way that the two opposite rotors rotate clockwise while the other two rotate in the anticlockwise direction. Each rotor produces a torque about the quadcopter's center and hence stability is achieved by two pairs of counter-rotating motors which give rise to a net moment/torque of zero at its center (Patel *et al.,* 2013). Thus its principle of operation is such that by varying the angular velocity of one or more rotors relative to the others, the quadcopter carries out the roll, pitch, or yaw movements (Gupte and Conrad, 2012; Gopalakrishnan, 2017).

2.3.1 Quadcopter Movement

Pitch up: The quadcopter carries out forward motion when the speed of the front rotor is increased while decreasing that of the rear rotor simultaneously with the other two rotors at equal speeds.

Pitch down: The quadcopter carries backward motion when the speed of the rear rotor is increased while decreasing that of the front rotor simultaneously with the other two rotors at equal speeds.

Roll left: The quadcopter rolls to the left when the left rotor's thrust is lower than that of the right rotor with the front and rear rotors at equals speeds

Roll right: The quadcopter rolls to the right when the right rotor's thrust is less than that of the left with the front and rear rotors at the same speed/thrust.

Yaw left: A quadcopter yaws to the left when the two rotors which move counter-clockwise are at reduced speeds when compared with the rotors rotating clockwise

Yaw right: A quadcopter yaws to the right when the two rotors which rotate clockwise are at a reduced speed when compared with the other two rotors rotating counter-clockwise.

Thrust: When all four rotors attain a high speed and are equal, the quadcopter increases in altitude.

Hover; When all four rotors attain a low speed and are equal, the quadcopter hover or decreases in altitude depending on the particular speed.

Fig. 2.6 illustrates the typical movement of a quadcopter about different axis.

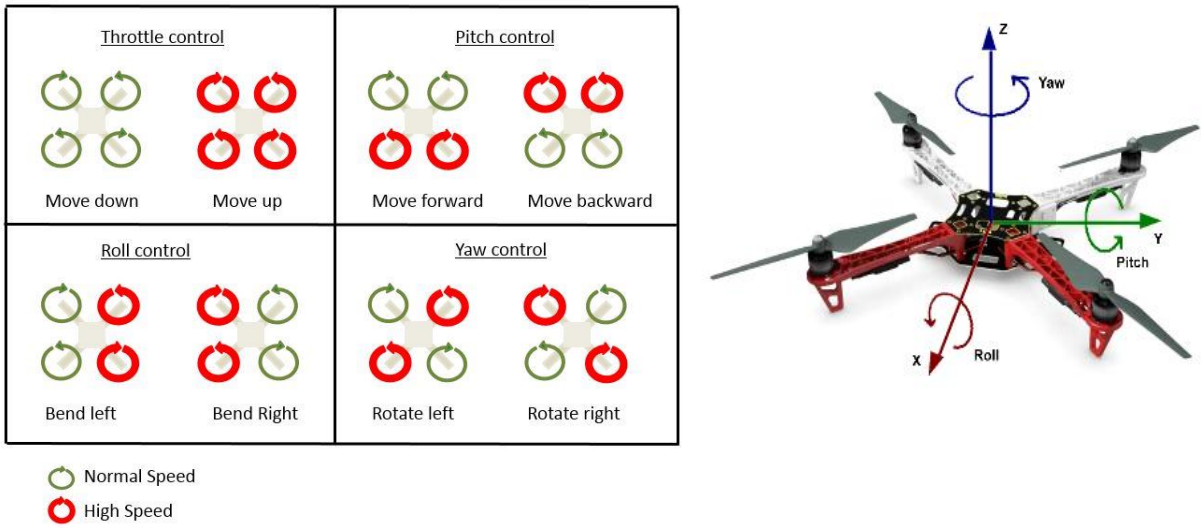


Fig. 2.6: Movement of a Quadcopter in Different Axis (adapted from Harsha, Balaji, and Kamath, 2014; Reséndiz and Rivas-Araiza, 2016)

2.3.2 Parts of the Quadcopter

As shown in Fig. 2.7, a typical quadcopter has the following parts according to Dryden and Barbaccia, (2014); Gopalakrishnan, (2017); Rehman *et al.* (2016); Abbe and Smith (2016); Gao 2015 and Shivaji *et al.* (2017):

i. **Frame**: It provides a physical structure, houses the electric motors and other components.
ii. **Electric motors**: They are two types of motors- brushed and brushless motors. For a quadcopter, a brushless motor is preferred due to its high thrust-to-weight ratio. The brushless motor spins the propeller which results in lift
iii. **Propellers**: They come in different sizes and materials and are measured by diameter and pitch in the format, diameter × pitch. Pitch is a measure of how many "travels" the propeller undertakes in one revolution while the diameter is the length of the propeller from tip to tip.
iv. **Electronic speed controller**: These are electric devices that collect PWM (Pulsating Width Modulation) signals from the flight controller and sends them to the electric motors thus regulating their signals appropriately.
v. **Batteries**: The battery supplies direct current (DC) power to the electric motor and all other components of the quadcopter. They come in different shapes and sizes, some of which are

alkaline, lead-acid, nickel-cadmium, lithium sulphur (Li-S), etc. Lithium polymer batteries (especially Li-S) are extensively used due to their high energy density

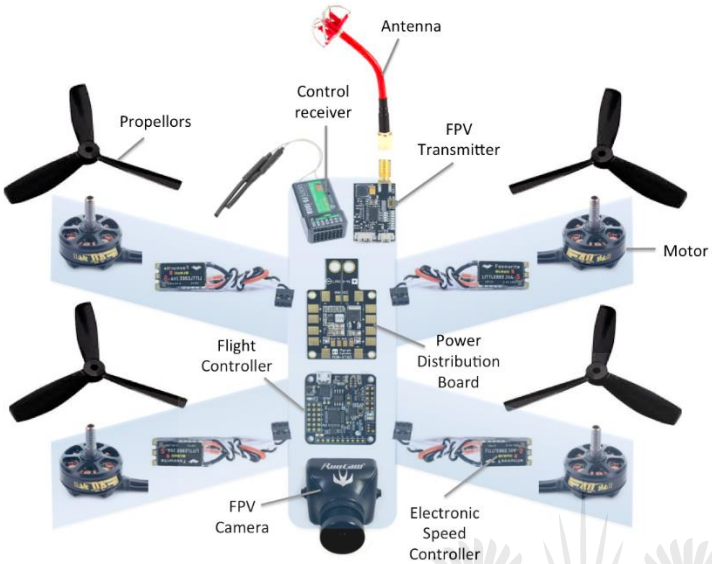


Fig. 2.7: Selected Parts of a Quadcopter (adapted from Nixon, 2017)

2.4 Deep Learning

The application of DL in solving real-life problems has stirred a great deal of recognition with significant impacts made in areas such as cancer prognosis (Murtaza et al., 2019), image analysis (Aradi, 2020), self-driving cars (Noda et al., 2015), speech recognition (He et al., 2016), natural language processing (Hassan and Mahmood, 2018) and prediction of natural disasters (Nevo, 2019), etc. These advancements were believed to have been initiated by Hinton, Osindero, and Tey (Hinton, Osindero, and Teh, 2006) who introduced the concepts of layer-wise greedy-learning and deep belief networks. DL's ability to analyze big data, automatically extract features and its short testing times have made it an undoubted preference to other conventional machine learning methods (Liu et al., 2017). However, it is computationally intensive requiring long training times but thanks to the high processing speed and parallelism of DL hardware accelerators which has greatly ameliorated this drawback (Ma et al., 2019).

Deep learning is a subset of machine learning, comprising of multiple processing layers, that transforms and learns a representation of data with various features in a hierarchical way (Kamilaris and Prenafeta-Boldú, 2018). It is made up of the input, several hidden and the output layers with nodes in each layer connected to nodes in the corresponding layer thus mimicking the

neuron structure of the human brain (Marcus, 2018). A weighted sum of the input is transformed by an activation function which generates non-linear outputs fed as input to the adjacent units of the succeeding layer until it reaches the output layer (Saleem and Chishti, 2019). The forward and backpropagation procedures are iterated until the weights and biases are optimized and then the result of the output layer becomes the solution to the problem. The activation functions mostly used are the sigmoid, hyperbolic tangent (tanh), Rectified Leaky Unit (ReLU), and Identity functions because they make it easier to compute the loss function required for weight optimization (Pillai, 2018; Zhang, Wang and Liu, 2018). The architecture of an artificial neuron, deep learning architecture, mathematical equations for the loss function, gradient descent, activation functions are shown in Fig. 2.8, Fig. 2.9, and equations 2.1-2.7.

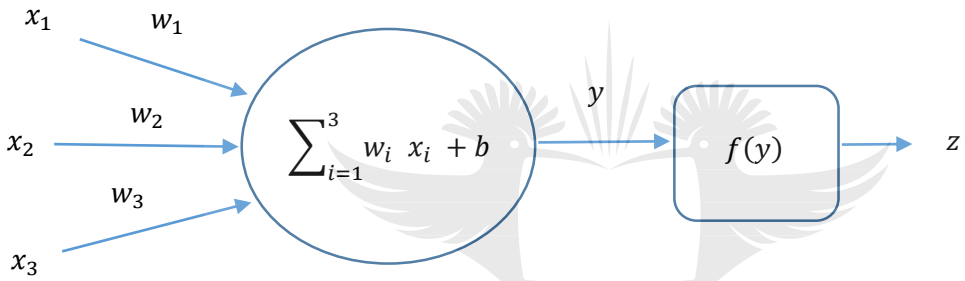


Fig. 2.8: Structure of an Artificial Neuron

Where x_1, x_2, x_3 = Input data

w_1, w_2, w_3 = Weights

b = Bias

y = Weighted sum

z = Output

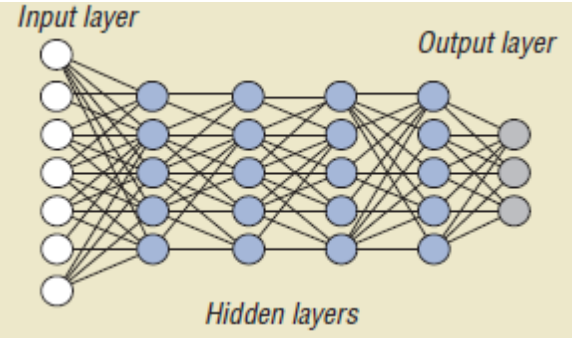


Fig. 2.9: Deep learning architecture (adapted from Lane *et al.* 2017)

$$L = \frac{1}{2m}\sum_{i=1}^{m}(T_i - Z_i)^2 \qquad \text{Equation 2.1}$$

Where, T_i = True output of the i^{th} sample,

Z_i = Estimated output of the i^{th} sample,

m = Number of outputs generated.

Equation 1: Loss Function of a Neural Network

$$w_i \rightarrow w_i - \gamma\frac{dL}{dw_i} \qquad \text{Equation 2.2}$$

$$b_i \rightarrow b_i - \gamma\frac{dL}{db_i} \qquad \text{Equation 2.3}$$

Where w_i = Weight of the i^{th} sample Equation 2.4

γ = Learning rate

b_i = Bias of the i^{th} sample

$$\text{Sigmoid}(y) = \frac{1}{1 + e^{-y}} \qquad \text{Equation 2.5}$$

$$\text{Tanh}(y) = \frac{1 - e^{-2y}}{1 + e^{-2y}} \qquad \text{Equation 2.6}$$

$$\text{ReLU}(y) = \{0, y\} \qquad \text{Equation 2.7}$$

$$\text{Identity}(y) = y \qquad \text{Equation 2.8}$$

DNN architectures have three classes of learning models and they are:

- Supervised learning model
- Unsupervised learning model
- Semi-supervised learning model

In supervised learning, the data used during the training procedure of the architecture is fully labeled in contrast to unsupervised learning where the architecture extracts relevant information from the data which are unlabeled. The semi-supervised learning model combines the

functionalities of both the supervised and unsupervised learning models whereby the input is a mixture of labeled and unlabeled data. Furthermore, deep learning architectures are of two categories: Discriminative architecture which generally supports the supervised learning models, and Generative architecture which supports the unsupervised learning models (Mohammadi *et al.*, 2018).

2.4.1 Generative Deep Learning Architectures

a) **Autoencoder (AE):** This is a neural network that consists of the input, hidden, and output layers. In this architecture, input data is transformed using unsupervised learning to an abstract form in a lower dimension and then reconstructed to produce output by fine-tuning with backpropagation (Shrestha and Mahmood, 2019). Dimension reduction and input reconstruction are carried out by encoder and decoder blocks to generate outputs that are as similar as possible to the input. AE is advantageous because it facilitates the extraction of relevant features and since the learning efficiency is improved as the input data is converted to a representation in a lower dimension (Zamini and Montazer, 2018). Denoised and Sparsed Autoencoders are variants with improved feature extraction capabilities. Denoised AE introduces noise deliberately in its training data while in sparse AE, some hidden units are made inactive. Fig. 2.10 below shows the AE architecture adapted from Zamini and Montazer (2018).

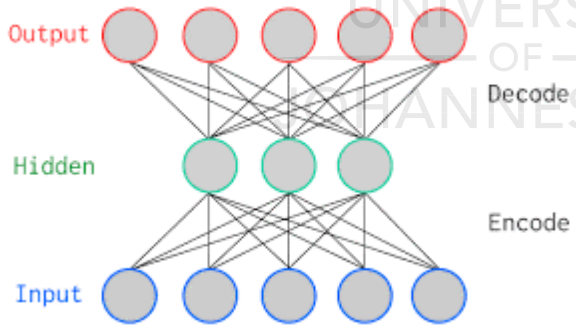


Fig. 2.10: Autoencoder architecture

b) **Restricted Boltzmann Machines (RBM)**: This is a type of artificial neural network (ANN) that consists of the visible and hidden layers with each neuron connected to all units in the adjacent layer but there is no connectivity within the same layer. This architecture uses unsupervised learning to build an RBM structure that probabilistically reconstructs the input (Shrestha and Mahmood, 2019). Also, variants of the RBM were proposed to

boost dimensionality reduction, collaborative filtering, and feature extraction functions. They are the discriminative RBM, conditional RBM, and FE-RBM introduced by Larochellle & Bengio (2008), Mnih et al (2012), and Elfwing (2015) respectively. Fig. 2.11 below shows the RBM architecture adapted from Mu and Zeng (2019).

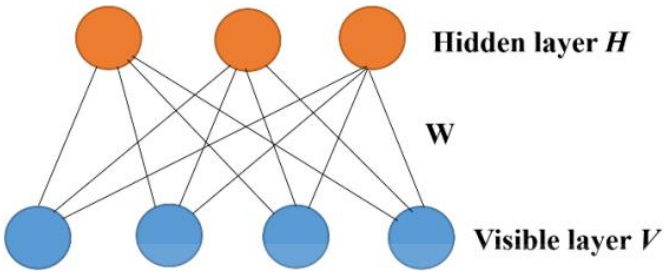


Fig. 2.11: Restricted Boltzmann Machines Architecture

W represents the weight between the layers

c) **Deep Belief Networks (DBN)**: This is a kind of ANN (proposed by Geoffrey Hinton) designed by stacking several RBMs, consists of the visible layer which accepts the input and the hidden layers responsible for extracting features (Mu and Zeng, 2019). With the output of a preceding RBM used to train the next RBM layer, the training phases of a DBN is carried out in 2 stages: pre-training and fine-tuning stages (Liu, 2017; Pandey and Janghel, 2019). In the pre-training stage, the DBN applies an unsupervised learning process to extract features from the input data while in the fine-tuning stage, supervised learning using the backpropagation algorithm is used to modify the network parameters. Fig. 2.12 below shows the DBN architecture adapted from Pandey and Janghel (2019).

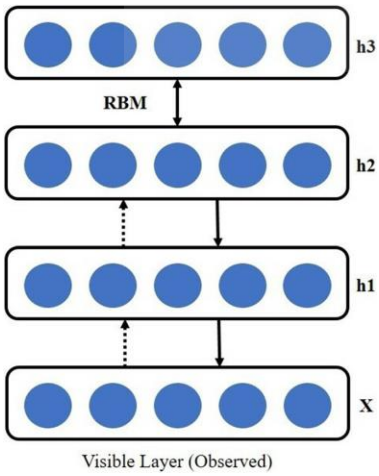


Fig. 2.12: Deep Belief Network Architecture

Where,

h, x represents the hidden and visible layer

d) **Generative Adversarial Network (GAN):** This is a deep neural network (DNN) structure with 2 networks: The Generator and Discriminator. The generator produces synthesized data derived from a data distribution while the discriminator functions to discriminate between the true data distribution and the data from the generator. To attain optimization, the GAN is trained so that the generator produces data which is identical to the true distribution so much that the discriminator has difficulty in differentiating between the true distribution and synthesized data (Amanullah, 2020). Fig. 2.13 shows the structure of GAN.

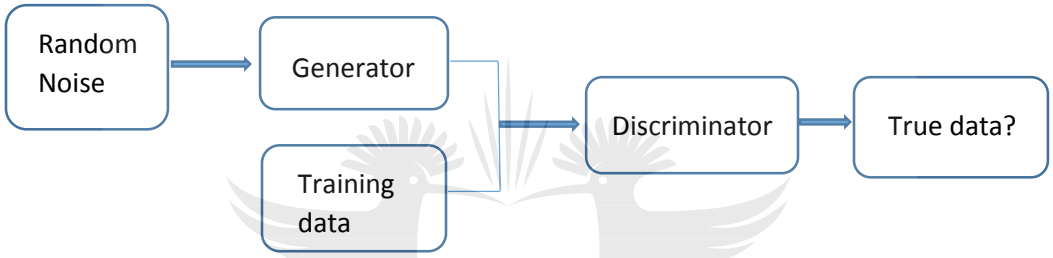


Fig. 2.13: Generative Adversarial Network Structure

2.4.2 Discriminative Deep Learning Architectures

Recurrent Neural Network(RNN): This is a kind of DNN consisting of the input, hidden and output layers applied in language modeling, machine translation, speech recognition, etc (Mu and Zeng,2019; Young et al., 2018).]. It is used to model sequential information and possesses an internal memory that captures information about previous computations as shown in Fig 2.14. It takes in two inputs (the present and recent past inputs) and applies the backpropagation through time(BPTT) algorithm in training the network such that the output at time step 't' is dependent on the output at time step 't-1'(Tang et al., 2018; Pandey and Janghel, 2019)

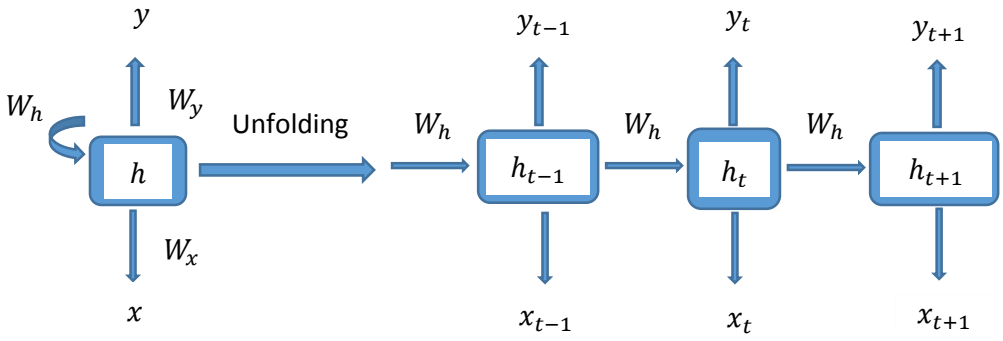


Fig. 2.14: Recurrent Neural

Where,

W_h, W_y , W_x represents the hidden weight, output weight, and input weight

x, h, y represents the input, hidden, and output values.

a) **Long Short-Term Memory (LSTM):** This is a special variant of the RNN which was proposed to make up for the drawback in RNN such as sensitivity to change in parameters, vanishing, and exploding gradient. It consists of the input, hidden as well as output layers and used for applications involving long dependencies in time such as handwriting generation, video descriptor, etc. The hidden layer of the LSTM comprises the memory cell (which captures information for a certain time-frame) and gates (input, forget, and output gates). The input gate determines the new information to be stored in the LSTM cell, the forget gate decides on which information should be forgotten and the output gate controls the flow of information to the network (Sengupta, S. *et al.* 2020). Fig. 2.15 shows the hidden layer of the LSTM adapted from Mu and Zeng (2019).

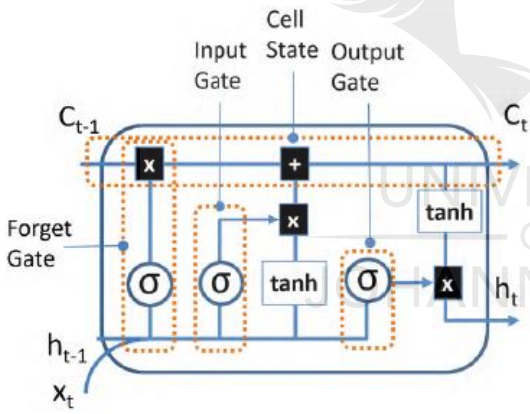


Fig. 2.15: The Long-Short Memory Hidden Layer

Where,

x represents the input value

C represents the cell value

H represents the output value of the hidden layer

σ represents the sigmoid function

b) Convolutional Neural Network (CNN): This is a class of DNN inspired by the human visual mechanism and capable of extracting hierarchical features from a 2-dimensional input (image, text, or audio signal) using a sequence of layers. It consists of the input, convolutional, pooling, fully connected, and output layers. The convolutional layer consists of a set of kernels (arrays of weights) that extract features (edges, contours, strokes, textures, orientation, color, etc.) from input data. These kernels are convoluted on the image by computing the sum of their dot products to generate feature maps. An activation function (most commonly the ReLU) is applied to introduce non-linearity and prevent network saturation (Pouyanfar, Chen, and Shyu, 2017). The pooling layer, which is either a max or average pooling function, reduces the spatial dimensions of the feature map and computation load in the network while retaining relevant information (Huang et al., 2018). Thereafter, the fully connected layer performs the classification tasks to produce a list of probable outputs and then passed through a softmax function that selects the output with the highest probability as the prediction for a given input. Similar to a conventional neural network, the goal of a CNN is to optimize the loss function in a network and it achieves this by applying a backpropagation algorithm via gradient descent to train the kernels and modify the weights. Fig. 2.16 shows the architecture of the CNN while Fig. 2.17 illustrates the convolution and max-pooling processes adapted from Park et al. (2019)

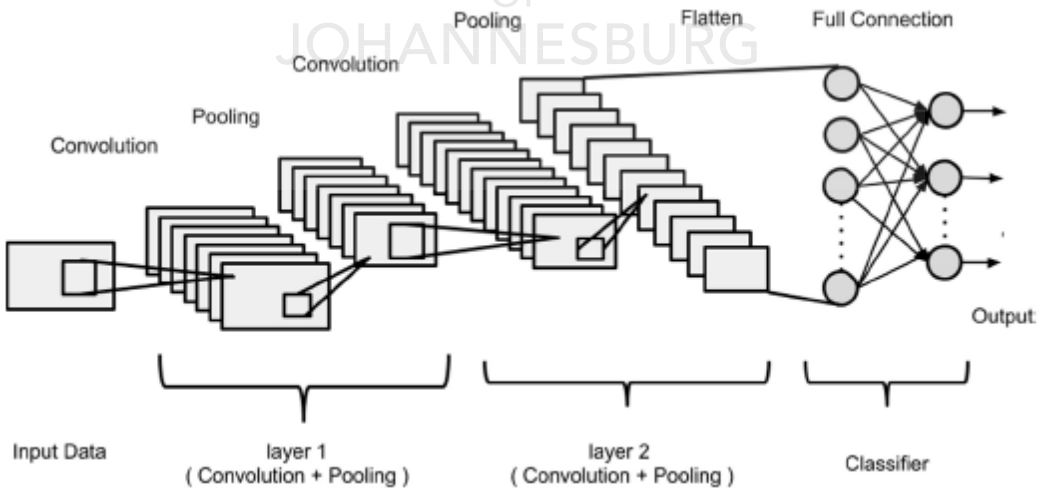


Fig. 2.16: CNN Architecture (adapted from Sengupta *et al.*, 2020)

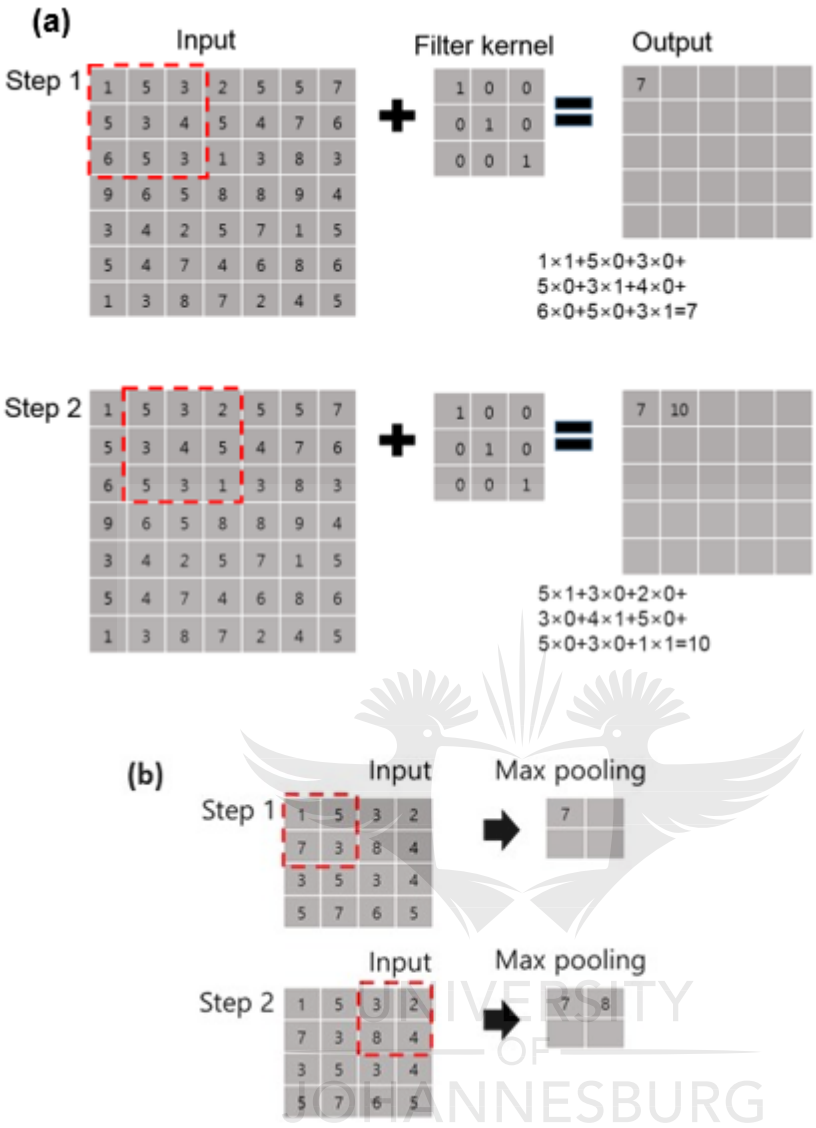


Fig. 2.17: Illustration of the convolution and max-pooling processes in CNN

2.5 Transfer Learning

The CNN is widely used for precision agriculture and computer vision applications involving image recognition, classification, and detection due to its sparse interaction, equivalent representation as well as weight capabilities (Pouyanfar, Chen and Shyu, 2017). It has been proven to yield better results than other DNN classes especially when transfer learning is applied (Pouyanfar *et al.*, 2018; Kamilaris and Prenafeta-Boldú, 2018).

According to Pan and Yang (2010), transfer learning can be defined as 'the ability of a system to recognize and apply knowledge and skills learned in previous tasks to new domains which share some commonalities'. It involves training deep neural networks by utilizing weights of pre-trained models in a situation where there is a deficit in training data to improve model performance and avoid over-fitting. There are two transfer learning approaches and they include deep feature extraction and fine-tuning (Coulibaly *et al.*, 2019). In deep feature extraction, features of a pre-trained network are used to train input data of a new dataset while in fine-tuning, the top layers of the pre-trained model are fixed but the final layer is modified to learn the properties of the new datasets. Examples of pre-trained CNN models are ResNet (He et al., 2016), VGGNet (Simonyan and Zisserman, 2014), AlexNet (Krizhevsky, Sutskever, and Hinton, 2012), MobileNet (Howard *et al.*, 2017), GoogleNet (Szegedy *et al.*, 2015), etc. Also, deep learning architectures are implemented on frameworks such as Tensorflow (Allaire *et al.*, 2016), Theano (Al-Rfou *et al.*, 2016), Keras (Chollet *et al.*, 2015), Caffe (Jia *et al.*,2014), Pytorch (Paszke *et al.*, 2017), Microsoft Cognitive Toolkit (Yu *et al.*, 2014), etc.

2.6 Computer Vision

Several works of literature (Krizhevsky, Sutskever, and Hinton, 2012; Szegedy *et al.*, 2015) have shown the competence of DL in solving computer vision problems as it outperforms other machine learning classifiers. Tremendous advances in image recognition, classification, and detection have been brought about by combining DL and computer vision with driverless cars and autonomous aerial vehicles as vivid examples (Ding *et al.*, 2020). The agricultural domain is not left out as several vision-based systems enabled with deep learning are used for identification and classification of pests (Cheng *et al.*, 2017), diseases (Ferentinos, 2018), and weed (Dos Santos Ferreira, 2017).

Computer vision can be defined as the branch of automation which enables computer systems to identify, see, and understand the physical world similar to the human vision. It also develops methods for the task of getting information from images. Computer vision technology can be classified into three stages: Low-level processing, intermediate-level processing, and high-level processing (Smith *et al.*, 1979; Baratella and Gomes De Melo, 2017; Bhargava and Bansal, 2018; Taheri-garavand *et al.*, 2019).

i. **Low-level processing**: Image recognition and pre-processing take place in this stage. Images are captured using devices such as video cameras, scanners, color spectrum analysis, thermal sensors, etc., and then pre-processed by resizing, noise removal using filters as well as correcting geometrical distortion.

ii. **Intermediate-level processing**: In this level, segmentation and feature extraction operations are carried out. Segmentation separates an image into distinct components. The two commonly used segmentation techniques are thresholding (segmentation by region) and clustering (segmentation by contours). Thresholding involves segmenting an image by partitioning its grayscale format into several regions. The pixel of the image is then classified into interest and background area each having a distinct gray level. The Otsu method is an example of the thresholding technique and it generates a gray level histogram that optimizes threshold from a grayscale image. The thresholding technique is disadvantageous as it requires more computation time and this makes the clustering technique a better option. In the clustering technique, however, pixels with similar characteristics form a cluster and are classified into a hierarchical and partition-based method. Clustering is of two types: soft clustering and hard clustering. Feature extraction involves extracting color, textural, and morphological features from an image with aim of enhancing the rate of recognizing the input.

iii. **High-level processing**: In this stage, information extracted from the acquired images are used to make decisions. These decisions are made based on features noticed or by applying machine learning techniques such as SVM, K-Nearest Neighbour (KNN), ANN, DNN, etc. for a more effective interpretation.

2.6.1 Types of Sensors in Computer Vision (McCarthy, Hancock and Raine, 2010)

i. **Monocolor Vision**: This is the simplest form of sensing whereby a scene visible to the human eye is captured with an RGB (Red Green Blue) camera. Here, objects extracted are easily identifiable by humans.

ii. **Stereo Vision & 3D Structure**: This type of sensor is used to monitor plant parameters such as height, leaf shape, leaf area, etc., and also differentiate between plant species.

iii. **Multi-spectral Imaging**: This type of sensor simplifies imaging analysis by imaging a part of the electromagnetic spectrum which shows relevant features in contrast to the broad visible band produced by an ordinary RGB camera. When using the electromagnetic spectrum, plant materials can be differentiated based on color (visible), cellular structure (near-infrared), thermal (mid-infrared), or hardness (X-ray) properties. It provides information invisible to humans.

iv. **Hyperspectral Imaging**: This is a combination of spectroscopy and imaging techniques used to generate spectral information for each pixel of a spatial image.

v. **Range sensing**: These are sensors that require variations in ambient lighting to function. It provides information about the canopy of plants.

2.7 Embedded Systems

The application of deep learning on embedded systems is a step in the ongoing technological revolution and this could be evident in the development of autonomous cars, drones, smart homes, cities, intelligent transportation, healthcare, video surveillance, etc. These embedded systems are hardware-software interfaces and they can be used to inference deep learning models. Example of embedded systems are:

i. Raspberry Pi
ii. Nvidia Jetson Series
iii. Intel Edison
iv. Intel UP
v. Ordroid U3+, etc.

2.7.1 Raspberry Pi

This is a credit-card-like single-board computer that makes use of the open-source Linux operating system and is used to provide access to the internet and connect automation systems (Suriansyah, Sukoco, and Solahudin, 2016). It was initially developed by the raspberry pi foundation in the United Kingdom to assist students in learning basic programming skills. It has evolved through several models since its first release in 2012 with significant improvements being noticed in each model released. The available models are model A, model B, model B+, model 2B, model zero, model zero W, model 3B, model 3B+, model 4B(2GB), and model 4B(4GB) with a typical

example shown in Fig. 2.18 (Leroux *et al.*, 2017). Table 2.1 below shows some selected parameters for the raspberry pi.

Table 2.1: Selected Parameters for the Raspberry Pi (Hattersley, 2018)

Category	Model	Dimension	CPU	GPU	Memory Size	Ethernet	Onboard WIFI & Bluetooth
Raspberry Pi 1	B	85.5 × 56.5mm	700MHz ARM 11 Processor	250MHz Broadcom VideoCore 4	512MB	Present	Absent
	A	85.5 × 56.5mm	700MHz ARM 11 Processor	250MHz Broadcom VideoCore 4	512MB	Absent	Absent
	B+	85.5 × 56.5mm	700MHz ARM 11 Processor	250MHz Broadcom VideoCore 4	512MB	Present	Absent
	A+	65 × 56.5mm	700MHz ARM 11 Processor	250MHz Broadcom Video core 4	512MB	Absent	Absent
Raspberry Pi 2	B	85 ×56.5mm	900MHz quad-core ARM Cortex-A7	250MHz Broadcom Video core 4	1GB	Present	Absent
Raspberry Pi Zero	Zero	65 × 30mm	1 GHz ARM 11	250MHz Broadcom	512MB	Absent	Present

				VideoCore 4			
	W	65 × 30mm	1 GHz ARM 11	Absent	512MB	Absent	Present
Raspberry Pi 3	B	85 × 56mm	1.2GHz 64 bit Quad-Core ARM Cortex-A53	400MHz Broadcom VideoCore 4	1GB	Present	Present
	A+	67 × 56mm	1.4GHz 64 bit Quad-Core ARM Cortex-A53	400MHz Broadcom VideoCore 4	512MB	Absent	Present
	B	82 × 56mm	1.4GHz 64 bit Quad-Core ARM Cortex-A53	400MHz Broadcom VideoCore 4	1GB	Present	Present
Raspberry Pi 4	B	88 × 58mm	1.5GHz 64 bit Quadcore ARM Cortex-A72	400MHz Broadcom Video core 4	2GB	Present	Present

	B	85 × 56mm	1.5GHz 64 bit Quadcore ARM Cortex-A72	400MHz Broadcom Video core 4	4GB	Present	Present

The raspberry pi according to Gondchawar and Kawitkar (2016) generally consists of:

- Universal Serial Bus (USB) Ports-Used for attaching peripherals such as the keyboard, mouse, webcam, etc.
- Secure Digital (SD) Card Slot- It houses the SD card where the operating system and data is stored.
- General Purpose Input Output (GPIO) Pins- Used to control external devices
- High Definition Multimedia Interface (HDMI) port- Used to connect to a monitor or television
- Audio Jack Port- Used to connect to speakers
- Micro USB Power Port- Used to connect to the power source
- Ethernet Port: Used to establish an internet connection from an external source.
- Camera Serial Interface (CSI) - This is the interface where the PiCamera is attached.
- Display Serial Interface (DSI) - This is used to connect a liquid crystal display (LCD) panel.

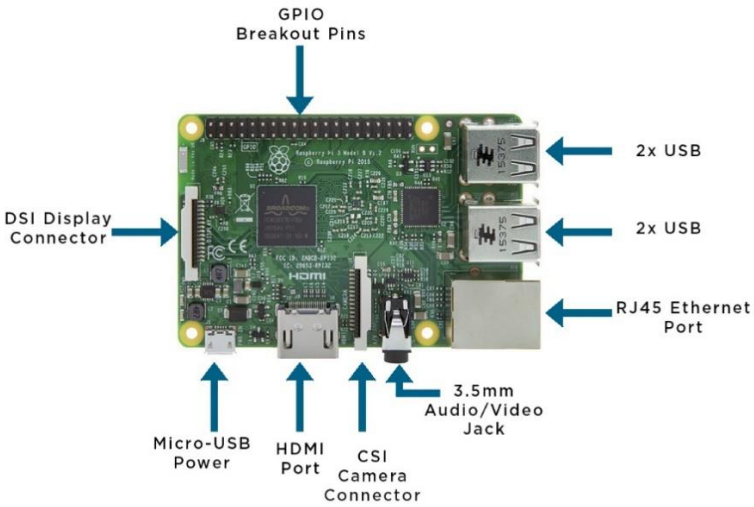


Fig. 2.18: Typical Parts of a Raspberry Pi 3B Model (adapted from Sam, 2016)

2.7.2 Nvidia Jetson Series

The Nvidia Jetson (shown in Figs. 2.19, 2.20, 2.21) is a powerful board computer and has similar applications as the raspberry pi but has unique features that make it the fastest in terms of speed among other embedded systems. A few of such features according to Mittal, (2019) are:

- Its low weight and power consumption.
- Its high performance per watt
- It's Compute Unified Device Architecture (CUDA)-programmability which facilitates efficient DNN inferencing.

Table 2.2 below gives an overview of the existing Nvidia Jetson series and some selected parameters

Table 2.2: Selected Parameters for the Nvidia Jetson Series (Leroux *et al.*, 2017; Kim *et al.*, 2018)

Nvidia Jetson Model	**Size**	**CPU**	**GPU**	**Memory**	**Storage**	**Power Under Load**
TK1	28nm	'4 plus-1' 2.32GHz	192-Core Kepler	2GB	16GB	10W

		ARM Quad-Core Cortex-A15				
TX1	20nm	1.73GHz ARM Quadcore Cortex-A57	256-Core Maxwell @ 998MHz	4GB	16GB	1-15W
TX2	16nm	2GHz ARM Quadcore Cortex-A57 + Nvidia Denver2 Dual-Core @ 2GHz	256-Core Pascal @ 1300MHz	8GB	32GB	7.5-15W

Fig. 2.19: A typical Nvidia Jetson TK1 (adapted from Howard, 2014)

Fig. 2.20: A Typical Nvidia Jetson TX1 (adapted from Franklin, 2015)

Fig. 2.21: A Typical Nvidia Jetson TX2 (adapted from Franklin, 2017)

Some Selected parameters of other embedded systems are shown in table 2.3 below while Fig. 2.22 and Fig. 2.23 shows their typical appearance:

Fig. 2.22: A Typical Intel Edison (adapted from Tameni, 2015)

Fig. 2.23: A Typical Intel UP Core (adapted from Allan, 2017)

Table 2.3: Selected Parameters for other Embedded Systems (Leroux *et al.*, 2017; Mittal, 2019)

Embedded System	**Dimension**		**CPU**	**GPU**	**RAM**	**Storage**
Intel Edison	35.5 × 25mm		Dual-Core, dual threaded Intel Atom CPU @ 500MHz	Absent	1GB	4GB
Intel UP			Intel Atom X5-Z8350 Quad-Core, 64 bits @ 1.92GHz	Intel HD 400 Graphics @500MHz	4GB	16/32/64GB

2.8 Sprayer Module

The use of agricultural drones in spraying chemicals across farms is widespread due to its speed and effectiveness of the spraying process. The Sprayer module is a system that functions to spray agricultural inputs such as fertilizers, pesticides, and herbicides on plants. It may consist of the following:

- **Nozzle**- for spraying the agricultural input;
- **Tank**- to store the liquid for spraying;

- **Pump**- to pump the liquid from the tank to the nozzle;
- **Pressure gauge**- to prevent the agricultural input from dripping;
- **Controller**- controls the speed of the pump.

The rate at which fertilizers/pesticides are dispensed by the nozzle could be controlled either through variable-rate spraying or PWM spraying (Escola 2013; Dryden and Barbaccia, 2014). Other types of sprayers are the electrostatic sprayer and knapsack-type electric sprayer (Zhang *et al.*, 2017; Qin *et al*., 2018). The variable rate sprayer employs the use of an electromagnetic high-frequency solenoid valve which varies the flow rate through the nozzle based on the amount of signal the controller supplies. The pulse width modulation is regulated by a PWM controller which generates pulse width signals to control the speed of the pump based on control signals from a data acquisition device.

2.9 Related Works

Most research works in literature as regards weed control were concerned only with detection/ classification using traditional machine learning and deep learning algorithms. The traditional machine learning methods include SVM (Akbarzadeh *et al.*, 2018), ANN (Bakhshipour *et al.*, 2017), RF (Lottes *et al.*, 2016), naïve Bayesian (De Rainville, *et al.*, 2014), Bayesian classifier (Garci and Pajares, 2017), Adaboost(Ahmad *et al.*, 2018), KNN (Kazmi *et al.*, 2015), threshold-based methods (Liu, Lee, and Saunders, 2014), etc. As deep learning methods outperform other machine learning algorithms, a few works which applied deep learning for detection/classification tasks would be reviewed.

Potena *et al.* (2017) presented an approach for detecting weeds in real-time with an unmanned ground vehicle which involved applying input RGB+ near-infrared (NIR) images to two CNN models. The first model was a SNet lightweight CNN which was used to extract the relevant pixels that represent the vegetation. The second CNN model selected a subset of original images that contained the most relevant features to boost the manual labeling process without the classification performance being compromised. Andrea *et al.* (2017) also developed an algorithm for classifying weeds from maize plants. The datasets obtained were trained on four CNN architectures (LeNet, AlexNet, CNet, and SNet) and the CNet proffered the best training accuracy of about 94%. The authors also proved that reducing the number of CNN filter layers and training the datasets on a

Graphical Processing Unit (GPU) would reduce the classification time. Furthermore, Milioto, Lottes, and Slachnoir, (2017) developed a system for identifying weeds from sugar beet plants. The authors trained the datasets with a custom CNN model consisting of three convolutional layers on a low-cost Nvidia GeForce GTX 940MX. They achieved a testing accuracy of about 97% for datasets of plants in an early growth stage and 89% for datasets of plants in a more advanced growth stage. Dos Santos Ferreira *et al.* (2017) trained a CNN model pretrained on an AlexNet architecture using the CaffeNet framework to detect two classes of weeds (broadleaf and grass) in a soybean field. Their results were compared with and were seen to outperform the SVM, AdaBoost as well as Random forest methods with an accuracy of 98%.

Computer vision techniques could also be deployed on embedded systems but for a more precise detection task, Alsalam et al. (2017) recommended the use of deep learning techniques in their work. He developed a quadcopter controlled using computer vision to detect weed on-board and approach the detected weed to perform a specific action such as the application of herbicides. An embedded system, Ordroid U3+ was used to process the images obtained from the on-board camera to check if weed was detected and also send information about the weed position to the pixhawk autopilot for navigation. The pixhawk autopilot guides the UAV to the new location of the weed, descends to a lower height, sprays herbicides, and then continues to the next waypoint thereafter. Also, Vikhram (2018) developed a smartweed detector and herbicide sprayer robot for an E. Ragi plant which carried out weed detection using computer vision techniques (morphological thresholding, erosion, and dilation). The robot was designed in such a way that weed images captured by a picamera were processed with computer vision algorithms written in python and deployed on a raspberry pi 3. A motor driver (L293D integrated circuit) was used as a bridge between the robot wheel and the spray pump motor with the raspberry pi. Such that when weed was detected, the spray pump was actuated to spray herbicides for four seconds, and if otherwise, the robot wheel moves for four seconds. Some weed misclassifications were observed by the author hence recommended the need for better classification methods.

The proliferation of embedding intelligence on embedded devices has given rise to diverse applications involving deep learning. Also, real-time execution operations of deep learning models on embedded systems have received a lot of attention in recent years. However, while executing deep learning models on embedded systems, constraints such as memory, energy/power

consumption, and computation must be taken into account (Alippi *et al.*, 2018). Deep learning models generally require a large number of computational resources to run and thus expends more power, occupies more memory with increased inference times. Furthermore, the embedded systems are usually capacity constrained in terms of memory and computation speed to handle most computational workloads (Dey *et al.*, 2019). Several techniques have been put forward to reduce computational load, memory size, and power consumption of deep learning models. A well-known approach for accelerating DNN inference called Pruning/model compression involves compressing the deep learning model with the assumption that some structures and representation in the model are redundant (Bhattacharya *et al.*, 2017; Han *et al.*, 2016; Howard *et al.*, 2017). However, this technique comes with a repercussion in terms of accuracy loss. Some pieces of literature proposed cloud computing such that some computations should be offloaded to a cloud server but this is seen to have a downside due to privacy constraints and unreliability of network connection (Kang *et al.*, 2017; Teerapittayanon, Mcdanel, and Kung, 2017) (Taylor *et al.*, 2018). Hinton, Vinyals, and Dean (2015) proposed the knowledge distillation approach whereby a 'student' (small) network is trained to replicate the output of a larger network known as the 'teacher' network hence knowledge is transferred from the teacher to the student network. This method however has issues of accuracy loss. Down-sampling of input images which involves reducing the image size was investigated by Lammie *et al*. (2019) and this boosted the real-time performance of DNN although with a slight decrease in validation accuracy which is a reasonable trade-off. While the search for an ideal approach to boost embedded system performance continues, GPUs, FPGAs (Field Programmable Gate Array) and Application Specific integrated circuits (ASICs) proffer good results while the other aforementioned approaches would also produce good results if their drawbacks are effectively managed.

Graphical processing units are specialized flexible electronic chips comprising multi-core processors that carry out parallel processing of data and perform mathematical operations at high speed. Its high processing speed and easy programmability makes it the most commonly used accelerator for training/inferencing deep learning models. Generally, GPUs are capable of performing millions of computations in the shortest possible time but have the disadvantage of not being energy efficient. However, the embedded GPUs by Nvidia such as the Jetson TX series are more power-efficient(requiring low power) but with a lower processing capacity than conventional

GPUs (Nikam, 2018). Examples of GPUs are the Nvidia-DGX series, Nvidia-AI GPU series, Nvidia-HGX server framework, Nvidia-Drive, Nvidia Jetson series.

Field programmable gate arrays are reconfigurable integrated circuits that can be reprogrammed an infinite number of times to accelerate computationally intensive tasks (Véstias, 2020). They are known for their good processing capabilities and are more energy-efficient than GPUs. They are however not widely used for accelerating computations because programming an FPGA requires expertise. Also, apart from being not easily programmable, FPGAs have a long synthesis time for complex designs. Examples of FPGAs are Spartan-6, Microsoft's project brain wave, Stratix-10, Stratix-IV, Zynq-7000, VirtexII-600, Virtex-7, Cyclone-IV, etc.

Application-specific integrated circuits are specialized integrated circuits designed to carry out specific algorithms such as the multiply and accumulate (MAC) operations for convolutional neural networks. They offer high efficiency, high processing speed, with easily programmable logic, and are more cost-effective than FPGAs (Misra, 2019). However, because they are design-specific algorithms, a slight change in the deep learning model would render the ASIC redundant (Nikam, 2018). Common examples of the ASICs are the Intel Movidius Neural Compute Stick(NCS), Google's Tensor Processing Unit(TPU), Snapdragon series 6, Krin 900 series, Ascend 910, etc.

GPUs, FPGAs, and ASICs are well suited for DNN computational tasks due to their high speed of processing and parallelism however, their presence in embedded systems are quite limited. A few instances of literature concerned with accelerating DNN models on GPUs, FPGAs, and ASICs would be reviewed below. Ukaegbu, Tartibu, and Okwu (2020) carried out an investigation on the impact of model optimization techniques with an emphasis on DL hardware accelerators being used in the agricultural domain and reviewed research works of literature published between 2017 and 2020. From the study conducted, it was evident that GPUs were the most prominent for acceleration DL models. Also, it was discovered that the application of energy-efficient hardware accelerators (such as FPGAs, ASICs, embedded GPUs) especially for real-time agricultural applications is still in its nascent stages as only a handful of research works have proposed such ideas. Furthermore, real-time practicability of DL models in areas of monitoring, pest and disease control, and weed control would be a formidable avenue for improving food production. It was observed that reduced training and inference runtimes were achieved through hardware

acceleration but recommended that combining hardware acceleration with other optimization techniques such as pruning, quantization, etc., would further reduce training/inference time. Sa *et al.* (2018) presented a technique for classifying weeds using multispectral images over a field with varying herbicide levels obtained by a micro aerial vehicle (MAV). Only crop, only weeds, both crops and weeds were the three kinds of image dataset obtained and these were sent to the ground station through the MAV at a frequency of 1Hz for processing. Semantic segmentation was carried out by a SegNet model on a modified Caffe framework using Nvidia's Titan X GPU module on a desktop computer for about 40 000 iterations which lasted for 12 hours. Dryman *et al.* (2017) developed a model pretrained on the DetectNet (a combination of a modified GoogleNet architecture and a clustering function) with 18 541 weed annotations on an Nvidia Titan X GPU. He obtained a precision of about 87% and a recall of about 43% despite having an occlusion of the cereal crop and weeds. Tang *et al.* (2017) proposed a method for identifying weeds in a soybean field by combining K-means unsupervised learning and CNN. The K-means clustering algorithm was used as a pre-training process to learn the weights of the network thus reducing errors. 820 images were trained on the K-means pre-trained CNN model using a dual-core @ 2.5GHz CPU and testing accuracy of about 93% was achieved which is 1.82 % more than that obtained when just a CNN weight initialization method is used. Lammie *et al*. (2019) in a bid to reduce high power consumption that is associated with accelerating DNN models on a GPU investigated the use of FPGA- accelerated DNN. VGG-16, DenseNet, and Wide Residual Network (WRN) were trained using an Intel DE1-Soc FPGA development board on an open-source framework using a modified version of Deep Weeds (known as DeepWeedX). The results obtained were compared with some models trained on an Nvidia Titan V GPU and AMD Ryzen 2700X@4.10GHz CPU. Based on his results, the inference times and power requirements for the three models were much lower on the FPGA when compared to its GPU and CPU counterparts. In a bid to develop an artificially intelligent recognition system on a low-power embedded system, Li and Yang (2020) developed a system that detects cotton pests. A CNN model was used to extract the features trained by the triplet loss and then implemented on a PYNQ-Z2 development board which consists of an Advanced RISC Machine(ARM) and FPGA. Even though a limited amount of dataset was used, a testing accuracy of above 95% was obtained with an inference runtime of 2FPS. The authors recommended that parameter quantization and further parallelism could be employed to improve inference speed. An Internet-of-things(IoT) device, powered by solar energy, for detecting codling

moth in apple orchards was developed by Brunelli et al. (2019). The insect datasets obtained were utilized to train through transfer learning, a pre-trained VGG-16 model on the TensorFlow framework for 10epochs. The trained model, with a classification accuracy of about 81%, was converted to a graph model and deployed on a Movidius compute stick for accelerated inferencing. The IoT device functioned to preprocess images of insects on the raspberry pi3, carry out prediction using the Movidius compute stick and notify the farmer, using a Long-Range Wide Area Network(LoRaWAN) protocol, when a codling moth was detected. A system was proposed by Shadrin et al. (2019) for detecting seed germination. The datasets collected were trained on a custom CNN, made up of 2 convolutional and 2 fully connected layers, for 50 epochs and a validation accuracy of 97% was recorded. The trained model was deployed on both a raspberry pi accelerated with a Movidius neural compute stick and a desktop GPU(Geforce GTX 1050Ti). Although the raspberry pi had a slower running speed of 4.5FPS, it is portable and its power consumption was significantly lower(2.5watts) than that of the desktop GPU(24.01watts).

To the best of the author's knowledge, no literature was found which executed deep learning models in real-time on embedded systems incorporated on a UAV applied for (weed control) agricultural purposes. The only work which seems forthcoming is that by Chechlinski, Siemiatkowska, and Majewski (2019) who proposes to develop a weeding machine to be mounted on a tractor with a weeding tool to be controlled by segmentation output from a custom CNN on a raspberry pi3B+ to be carried out in 2020. However, outside the agricultural domain, there are some examples of literature where deep learning was deployed for several applications. Manderson *et al.* (2018) developed a vision-based controller that guides an aqua robot to swim near coral reefs avoiding collisions and coral-free areas. He made use of a ResNet-18 architecture which takes as input the image data and makes decisions as regards the relative steering angles of the robot. The CNN architecture was deployed on an Nvidia Jetson TX2 which processed the image at 10Hz with an accuracy of 41% for classification. Gu *et al.* (2018) also developed a tennis ball collector that detected tennis balls with a DNN technique known as YOLO (You Only Look Once). The tennis collector carried out path-planning to collect the ball using the 'pointer network' algorithm thus solving the 'traveling salesman problem'. These algorithms were deployed on an Nvidia Jetson TX1 and the results obtained were evident that the robot was able to recognize the ball and perform path-planning efficiently in getting the ball. Finally, Bechlel *et al*. (2018) built a low-cost autonomous car known as the DeePicar which operates in such a way that input image from a

camera is used to generate steering angle value as an output in real-time thus navigating the car. It makes use of a CNN architecture with five convolutional networks and four fully connected layers (same as that used in Nvidia's DAVE-2 driverless car). The CNN architecture was used to train data on recorded time-stamped videos and control commands with an Nvidia Titan Xp GPU. The trained network was then deployed on a raspberry pi 3 and operates such that image frames from the web camera are processed by the network to produce steering angle output which is converted into PWM values to control the car's steering motor.

2.10 Conclusion

Overall, the studies presented in this chapter highlighted the need to implement deep learning algorithms on an embedded system for weed control. This chapter discussed the unmanned aerial vehicle, deep learning techniques, and computer vision. It also explained the types of embedded systems and sprayer module. Finally, a review of literature, concerning weed identification/classification using deep learning techniques and execution of deep learning algorithms on embedded systems for real-time applications, was carried out. The succeeding chapter would discuss the preliminary study conducted to gain insight into this research work

Chapter 3 - Application of Deep Learning for potassium deficiency detection

3.1 Introduction

This study described the application of artificial intelligence on an embedded system. It involved the application of the DL algorithm in detecting potassium deficiency in a red grapevine. A CNN model, which was retrained on the ResNet50, was deployed on a raspberry pi3 and a spraying action was prompted by the GPIO pin being activated to light up a Light Emitting Diode (LED).

3.2 Significance of this preliminary study

This study is significant based on the fact that grapevine contributes to about 40% of South Africa's overall export earnings. It is also because the adverse effect of nutrient deficiency could be alleviated by prompt detection using deep learning techniques.

3.3 Objective of this preliminary study

The objectives of the study were:

- To train a ResNet-50 model and evaluate its performance using an SVM model
- To deploy the CNN model on a raspberry pi and test for spraying actuation.

3.4 Methodology

3.4.1 Acquisition of dataset and training of the model

The red grapevine image datasets were provided by Rangel et al. (2016). They consisted of 50 images made up of healthy and potassium deficient leaves for six varieties of the red grapevine. They were divided in a ratio of 6:2:2 into training, validation, and test dataset. Thereafter, the training data was used to retrain, through transfer learning, a pre-trained ResNet50 model on the Keras framework on top of TensorFlow using a 4GB RAM CPU Intel core B160 processor for 5 iterations. Furthermore, the image datasets were also divided in a ratio of 7:3 into training and test data to train the SVM classifier on features initially extracted using the histogram of oriented gradients (HOG). Fig. 3.1 and Fig. 3.2 show an example of the image dataset and the flowchart of the methodology respectively adapted from Ukaegbu et al. (2020).

Fig. 3.1: Example of Image dataset made up potassium deficient and healthy red grape leaf

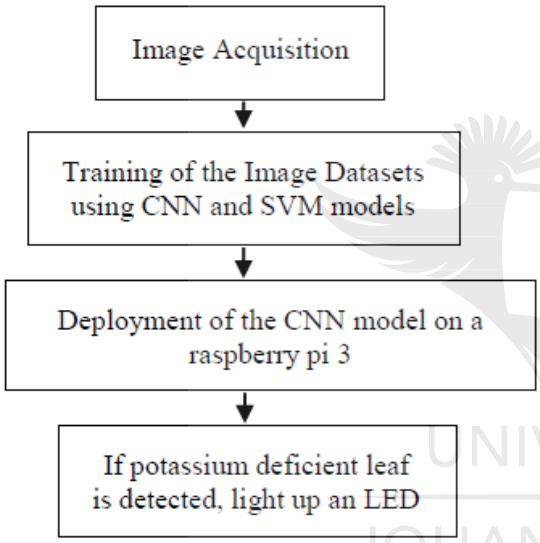


Fig. 3.2: Flowchart of the Methodology

3.4.2 Principle of operation

This system was structured such that the DL algorithm deployed on the raspberry pi3 analyzed the video stream by the pi camera for features of potassium deficiency. When a potassium deficient leaf was detected, GPIO pin 7 was activated to light up an LED for 3 seconds otherwise, no GPIO pin was be activated. Programming codes written in the python language were used to access the deployed CNN model, link the video stream to the deployed model, and regulate the timing of the LED. Fig. 3.3 gives an illustration of the working principle adapted from Ukaegbu et al. (2020).

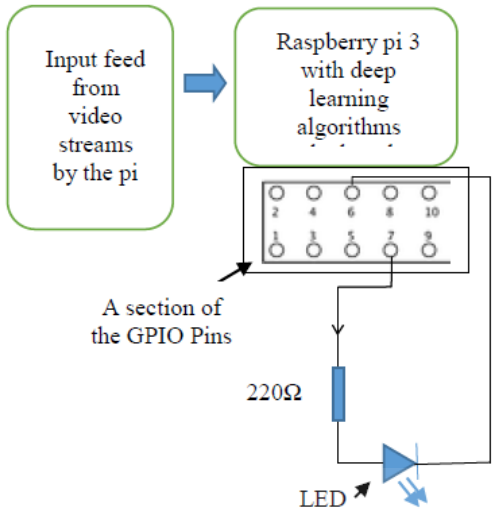


Fig. 3.3: Block diagram for detection and actuation

3.5 Results

The image datasets with an input shape of 224×224×3 generated 2048 feature maps when retrained on the ResNet-50 model. Training, validation, and testing accuracies of 89%, 81%, and 80% respectively were obtained. Also, training and validation losses of 0.2205 and 0.4720 were obtained respectively which depicted that the model learned the features well. In contrast, an accuracy of 66.7% was obtained from the SVM classifier. The CNN model, which was the better performing model, was deployed onto the raspberry pi with the aid of the FileZilla Client software. An LED was lit when the image of a potassium deficient leaf was brought close to the pi camera. Fig. 3.4 and Fig. 3.5 shows the graphs of the training and validation accuracies as well as the losses.

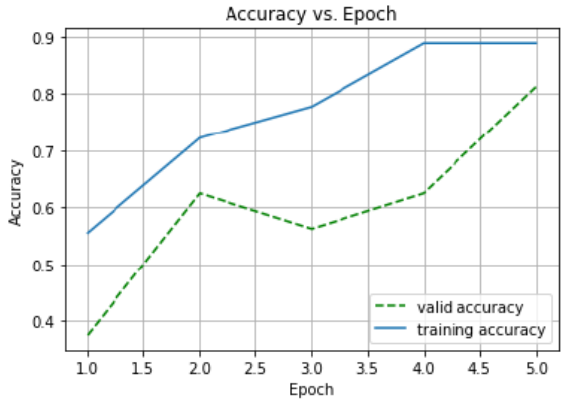


Fig. 3.4: Graph of Accuracy versus Epoch

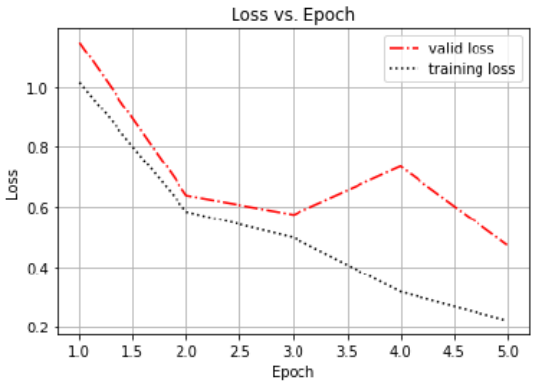


Fig. 3.5: Graph of Loss versus Epoch

In summary, this research study confirmed the fact that DL algorithms surpass other conventional machine learning models, especially in image classification tasks. It also acknowledged that the limited image dataset was a challenge as it affected the model accuracy. Finally, a prospect of incorporating a sprayer module instead of the LED used in this study. Also, mounting the proposed smart sprayer module on a ground robot or unmanned aerial vehicle for the real-time detection and spraying of fertilizers was envisaged

3.6 Conclusion

This preliminary study described and developed a deep learning-based potassium deficiency detector and triggered a prompt for spraying actuation. Although it was evident that limited image datasets posed a serious challenge as regards the accuracy of the CNN model, this work envisages solving the problem of poor harvest due to potassium deficiency in grape vineyards. Potential future work could consider the incorporation of a sprayer module in place of the LED and mounting the entire system on a land robot or unmanned aerial vehicle for real-time detection and spray of fertilizers.

Chapter 4 - Methodology

4.1 Introduction

This chapter gives a robust elucidation of the materials and methods employed during the course of this research work. First of all, it describes how the datasets were acquired, provides information on the software used as well as the packages installed, and also further explains the base neural network used. It goes further to give an overview and setup procedure of the raspberry pi 3B. Finally, it discusses the calculation for the compatibility of selected components, lists out the specifications of the smart herbicide sprayer components and procedures for assembling them.

4.2 Acquisition of datasets

The dataset used in this study is about one-third of the soybean weed dataset gotten from the public dataset source, Kaggle. It consists of soil, soybean, grass weeds, and broadleaf weeds which sums up to a total of 5349 images. The image datasets, created by Dos Santos Ferreira et al. (2017), were segmented with the aid of the SLIC algorithm on the pynovisao software. A DJI Phantom 3 professional UAV, equipped with an RGB camera was used to capture the images from a height of 4m above the ground. Fig. 4.1 to Fig. 4.4 show examples of the soybean weed datasets.

Fig. 4.1: An example of the broadleaf weed image dataset

Fig. 4.2: An example of the grass weed image dataset

Fig. 4.3: An example of the soybean mage dataset

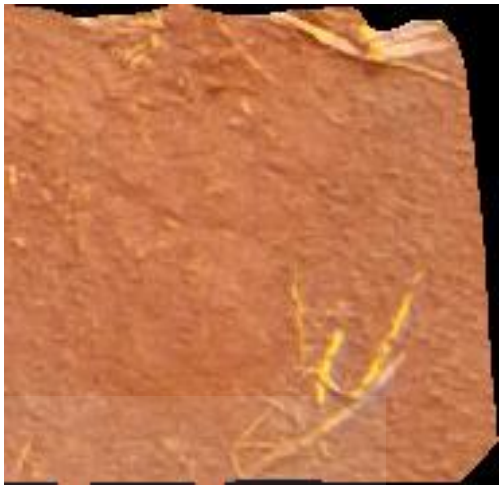


Fig. 4.4: An example of the soil image dataset

4.3 Model development software and packages installed

Anaconda individual edition is a data science toolkit/software which consists of about 7500 open-source packages and libraries used for machine learning tasks. It is a platform that supports Python and R programming languages founded by Wang and Oliphant in 2012. Also, it is one of the three products of Anaconda Inc. with the anaconda team edition and anaconda enterprise edition being the other two. The anaconda individual edition is the particular product employed in this research work and it includes the anaconda navigator (as shown in Fig. 4.5) which allows for the launch and installation of conda packages and virtual environments. The conda is an open-source package manager for python programs used to install, run, and update packages in anaconda.

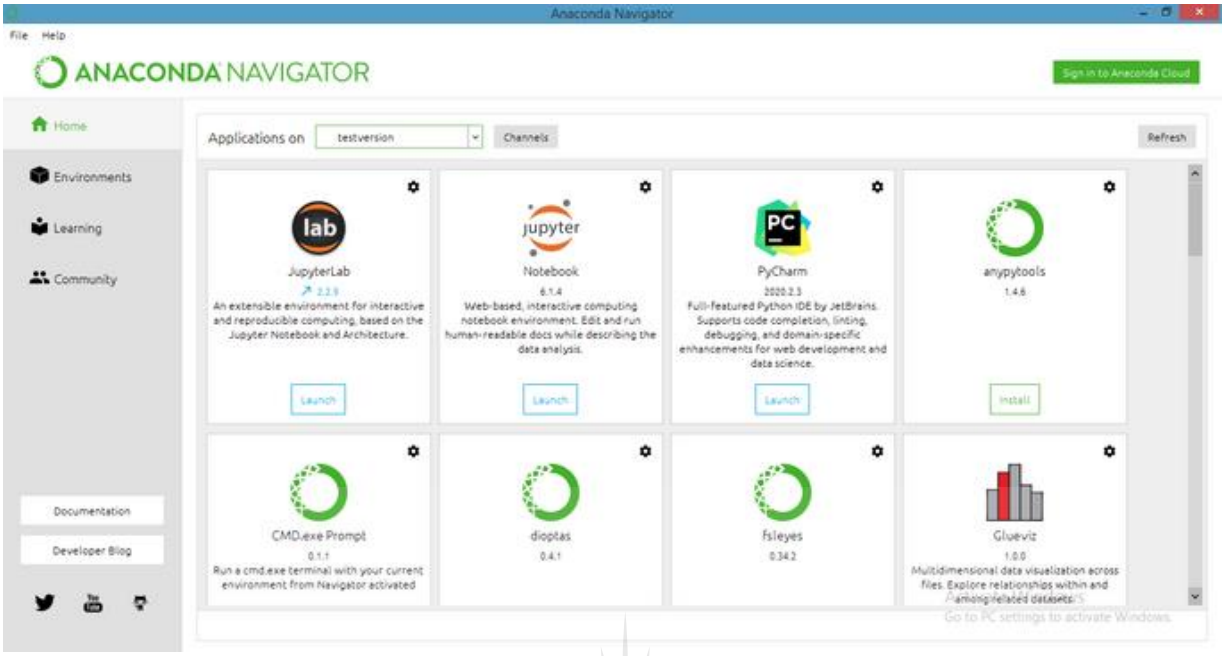


Fig. 4.5: Home page of the Anaconda navigator

After downloading the anaconda individual edition, the applications which are present by default on the navigator are:

1. Jupyter notebook
2. Jupyter lab
3. Spyder
4. Glue
5. Orange
6. Rstudio
7. Visual Studio Code, etc

The Jupyter notebook is a web-based interactive environment, usually with the ipynb extension. It consists of an organized sequence of input and output cells where codes, text, plots, etc can be written. For this research work, a virtual environment titled 'testversion' was created to run on python 3.6 and the Jupyter notebook was used in writing the python codes used to train the random forest classifier. Also, the Jupyter notebook in google colaboratory was used to write DL algorithms for training the CNN model. Fig. 4.6 shows the Jupyter notebook environment.

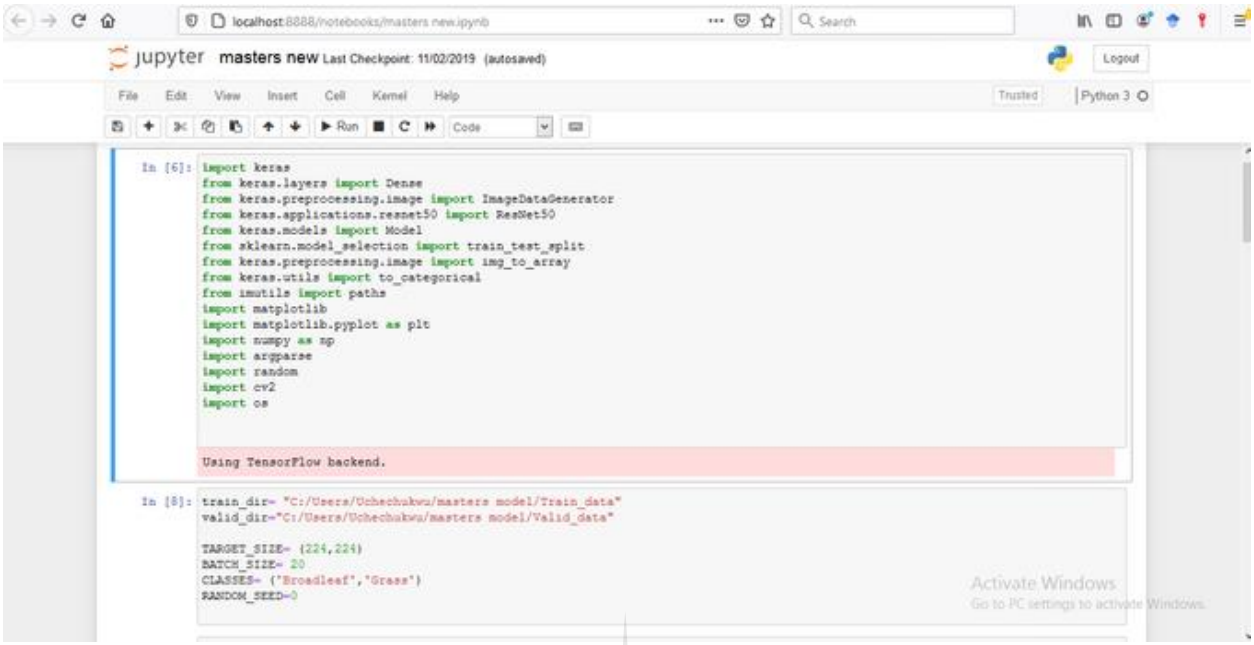


Fig. 4.6: The Jupyter notebook

4.3.1 Packages/library installed

The packages /libraries installed during the course of this project are briefly described as follows:

- Tensorflow: This is an open-source neural network platform developed by Google for solving machine learning tasks by processing data in different nodes/neurons. The TensorFlow 2.3.0, which is the default version in the google colaboratory, was utilized. The TensorFlow 2.3.0 version provides support for the Keras application programming interface which was used in this work.
- Numpy: This is a python library which enables support for arrays and matrices.
- Pillow: This is a python library that enables support for accessing, editing, and saving several image file formats.
- Scikit-learn: This is an open-source machine learning library that features several machine algorithms such as SVM, random forest, K-means, gradient boosting, etc.
- Matplotlib: This is an open-source python library that enables graph plotting function.
- Scikit-image: This is a python package used for the pre-processing of images.
- Open Source Computer Vision (OpenCV): This is a python library designed for computer vision application

4.4 Base Neural Network-ResNet50

ResNet, also known as deep residual network produces a good performance for deep networks as it creates a way of passing information from an earlier layer to a much deeper layer in a model. . Generally, the vanishing gradient poses a serious challenge in deep networks and this affects the accuracy of the model. This challenge is however curbed in residual networks as it employs skip or shortcut connections which assists a network in learning global features. It achieves this by skipping irrelevant layers during training. This results in the most appropriate tuning of the layers and hence faster training of the network. The ResNet50 is made up of the following layers:

- 7×7 convolution layer consisting of 64 kernels
- 3×3 max-pooling layer with a stride of 2
- 16 residual building blocks
- 7×7 average pooling layer with a stride of 7
- A fully connected layer with 1000 nodes
- Output layer

The CNN model used in this research work was a ResNet50 re-trained through transfer learning. This is because the earlier layers of the base neural network were frozen and its output layer bearing the softmax activation was modified to a 4-neuron layer to account for the 4 classes of the soybean weed dataset.

4.5 Training the CNN model and random forest classifier

The soybean weed dataset was divided in a ratio of 7:2:1 into training, validation, and test dataset. With an input shape of 224×224×3 and a batch size of 20, the datasets were used to train the ResNet50, through transfer learning, for 10 epochs on google colaboratory. The CNN model was trained on the TensorFlow framework carried out a 4GB RAM Intel® Celeron® CPU 1007U @ 1.50GHz. During compilation of the model, the Adam optimizer, categorical crossentropy loss, and accuracy metric were employed. After the training procedure, the CNN model was converted to a TensorFlow lite format to be efficiently readable by the raspberry pi interpreter. Also, an optimization technique known as quantization was carried out to optimize the CNN model for both latency and size hence improving its performance. The random forest classifier was used to evaluate the performance of the CNN model and was selected because it is the best machine

learning classifier Fernandez-Delgado et al. (2014). The algorithm for training the CNN model as well as the random forest classifier, conversion to TensorFlow lite format, and quantization are shown in appendix B.

4.6 Calculation for component compatibility

4.6.1 ESC calculation

The amperage rating of the ESC should be 20-50% more than the amperage rating of the electric motor.

$$\min \text{ESC amperage} = 1.2 \times \min \text{amp rating of a motor} \quad \text{Equation 4.1}$$

$$\max \text{ESC amperage} = 1.5 \times \max \text{amp rating of motor} \quad \text{Equation 4.2}$$

4.6.2 Battery calculation

$$\text{Discharge current} = \text{battery capacity} \times \text{C rating} \quad \text{Equation 4.3}$$

The higher the rate of current discharge rate, the higher the capacity of the battery to withstand overheating. Also, the ESC amperage rating should not exceed the battery discharge current.

$$\max \text{current drawn by the motors} = \text{no. of motors} \times \max \text{current drawn by 1 motor} \quad \text{Equation 4.4}$$

4.6.3 Thrust calculation

For a quadcopter to be able to lift off the ground, the total thrust to total weight ratio should be greater than 1.

$$\text{Thrust provided by each motor} = \frac{2}{4} \times \text{total weight of quadcopter} \quad \text{Equation 4.5}$$

$$\text{Thrust prvided by the propellers} = \frac{\pi \times D^2 \times v \times \Delta v \times p}{4} \quad \text{Equation 4.6}$$

Where $D =$ diameter of propeller

$v =$ velocity of air

$\Delta =$ velocity of accelerated air;

with the assumption that it equals $11.61\ m/s$ at 78% ideal efficiency

$p =$ density of air

From the calculations described previously, the electric motors draw a maximum current of 80A which is less than the battery discharge current. Also, the ESCs are compatible with the electric motor since it has a current rating of 30A which is higher than that of the electric motor (12A). Furthermore, the ESC is designed to handle batteries ranging from 2 to 3 cells and the battery being used in this work has 3 cells. From the thrust calculations, the thrust of each motor obtained is seen to be more than 2/4 of the weight of the quadcopter hence lift is assured. The frame chosen is made up of carbon fiber which is known for its lightweight and low inertia. Table 4.1 shows the results from the calculations performed.

Table 4.1: Calculations performed

Minimum ESC amperage	14.4A
Maximum ESC amperage	18A
Discharge current	247.5A
Maximum current drawn by the motors	80A
Thrust provided by the propellers	340.47N

4.7 Raspberry pi

4.7.1 Overview

The embedded system being used in this dissertation is the raspberry pi 3B. It consists of the quadcore 64-bit ARM cortex @ 1.2GHz, 1GB RAM, 4 USB 2.0 ports, HDMI ports, camera serial interface, display serial interface, micro USB power port 40 GPIO pins, and SD card slot. The raspbian is the operating system recommended and hence being employed for use on the raspberry pi 3B. Fig. 4.7 provides a pictorial representation of the raspberry pi 3B.

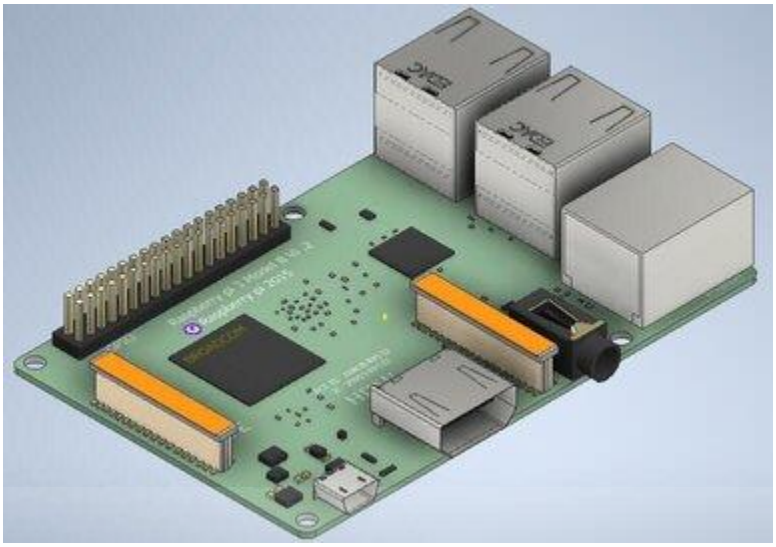


Fig. 4.7: Layout of the raspberry pi 3B

4.7.2 Setup for the raspberry pi

The components required for the setup are:

- A 2.5A 5.1V micro USB power supply
- An SD card ranging from 16 to 32GB
- Keyboard and mouse
- HDMI cable
- A pi camera or USB camera
- A computer or television screen.

4.7.3 Setup for the SD card

The New out of Box (NOOBS) installation manager is the easiest way of installing the raspbian operating system on the SD card using a personal computer (PC) and an SD card reader. The procedure for installing the operating system image on the SD card is highlighted as follows:

- Download a zip file of NOOBS from https://www.raspberrypi.org/downloads/
- Format the SD card using an SD card formatting software, most preferably the SD card formatter.
- Extract the NOOBS from the zip file already downloaded.
- Copy the extracted NOOBS files into the SD card using an SD card reader inserted in a PC.

4.7.4 Booting the raspberry pi

After connecting all the components according to the setup listed out in 4.7.2 and inserting the SD card into the SD card slot on the raspberry pi, the following steps should be carried out to boot the raspberry pi.

- Supply power (2.5A, 5.1V) to the raspberry pi through the micro USB power port. A red LED would light up on the raspberry pi and the computer screen would be rainbow-colored.
- Thereafter, the initial setup which includes time zone setting, setting up the Wi-Fi, user profile completion, etc. is required.
- The raspbian desktop (as shown in Fig. 4.8) would then appear on the screen after the initial setup is completed and a reboot request would pop-up to fully complete the setup.

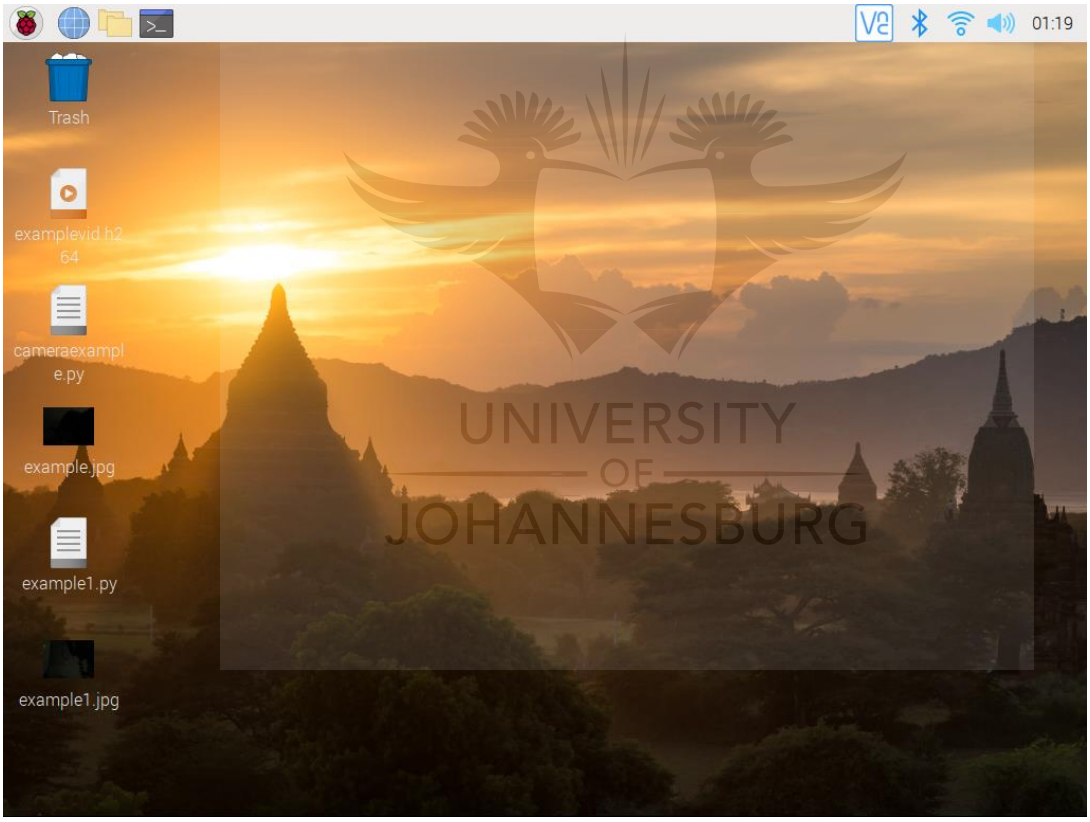


Fig. 4.8: Typical raspberry pi desktop

4.7.5 Setting up the raspberry pi for DL application

Similar packages installed on the PC for training the CNN model are installed on the raspberry pi. Also, a virtual environment was created where the required packages (TensorFlow, Keras,

OpenCV, etc.) were installed. The trained model from the PC was transferred to the raspberry pi using the FileZilla Client software. The procedure for file transfer from a PC to the raspberry pi is given as follows:

- After downloading the FileZilla client software and with an internet connection on the PC, click on the FileZilla software to launch it.
- On the File menu, click on 'site manager'. Make sure the protocol is set to SFTP-SSH File Transfer Protocol. Also, type in the internet protocol (IP) address and password of the raspberry pi. Type in 'pi' as the user and then click on 'connect'.
- If the above was correctly done, it should show the folders/directories present in the raspberry pi on the right-hand side with the directories of the PC on the left-hand side as shown in Fig. 4.9.
- Open the particular raspberry pi folder (on the right-hand side) you intend to send the file to. Right-click on the file to be sent(on the left-hand side) and click on 'upload'

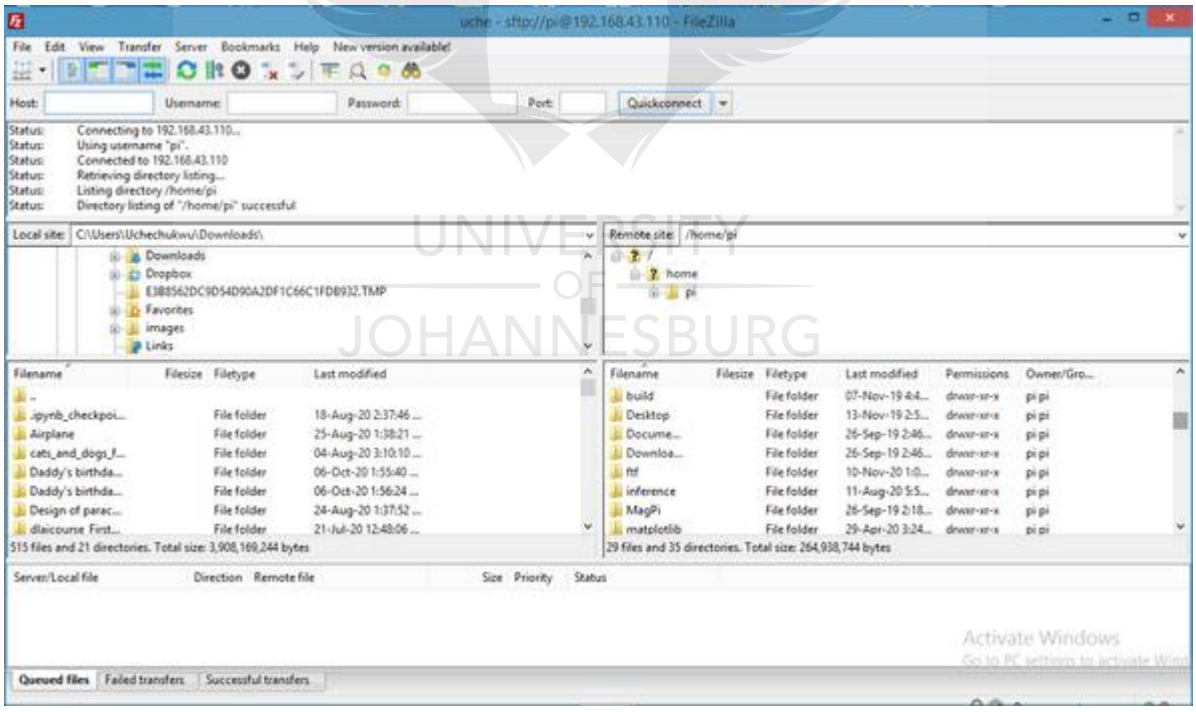


Fig. 4.9: FileZilla software with raspberry pi connected

4.8 Assembling the smart herbicide sprayer

Table 4.2 gives the specifications of some components of the smart herbicide sprayer.

Table 4.2 Specification of selected smart herbicide sprayer components/accessories

S/N	Name of Component	Specification
1.	XXD A2212 brushless outrunner electric motor	KV: 1000 Current rating: 12A/60s No-load Current: 10V: 0.5A Number of cells: 2-3 lithium polymer battery Dimensions: Ø27.5 × 30mm Shaft diameter: Ø3.17mm Weight: 47g
2	Electronic speed controller	Continuous current: 30A Burst current: 40A Input voltage: 2-3 cell lithium polymer battery BEC: 2A/5V(linear mode) Size: 45mm(L)×24mm(W)×11mm(H) Weight: 25g
3	APM 2.8 flight controller	Made up of 3- axis gyro, accelerometer, and barometer. 4-megabyte dataflash chip. Optional off-board GPS LEA-6CH module with a compass.
4	Propeller	Diameter: 10″ Pitch: 4.5″
5	Flysky FS-I6 CH transmitter/receiver set	Transmitter: Channel: 6 Modulation type: GPSK RF range: 2.408-2.475GHz Bandwidth: 500KHz

		Receiver: FS-IAS 6-channel
6	BMT mini subm. water pump	Working voltage: 12V DC Working current: 400mA Max flow: 240L/H Pump life: >30000 hours Inlet diameter: 8mm Outlet diameter: 8mm
7	ZOP power 11.1V 3S lithium polymer battery	Capacity: 5500MAH Continuous discharge rate: 45C Size: 40x46.5x138
8	B6 V3 Smart Balance Charger	DC Input voltage:11-18v Charge power:80W Discharge power:10W Charge current range:0.1-6A Discharge current range:0.1-2A
9	Single / 1 channel 5VDC 10A relay module development board	Control Voltage: 5V DC Max Control Capacity:10A@250VAC or 10A@30VDC Size:.41 x 16 x 16mm Weight:12g

Figs. 4.10 to 4.20 show details of the components and accessories used for the construction of the smart herbicide sprayer.

Fig. 4.10: Battery Charger

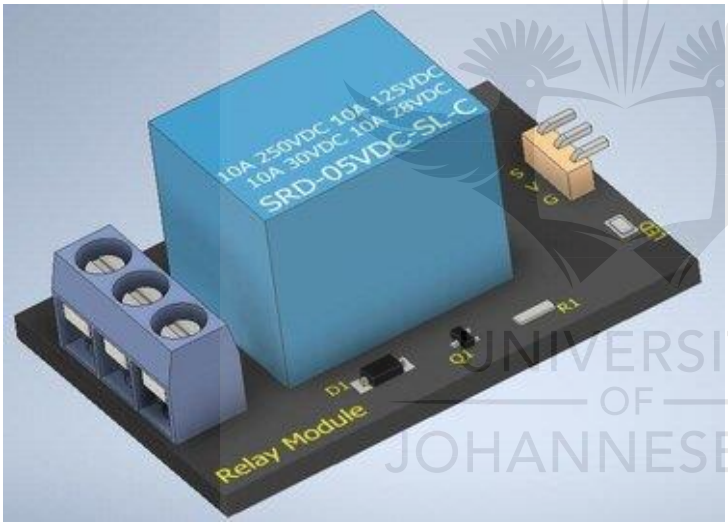


Fig. 4.11: Relay module

Fig. 4.12: Spray nozzle

Fig. 4.13: Brushless electric motor

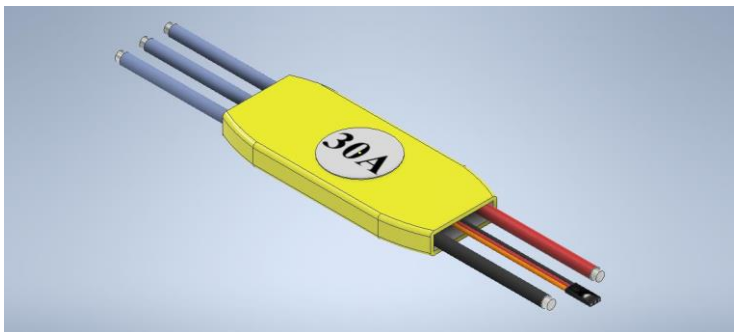


Fig. 4.14 Electronic speed controller

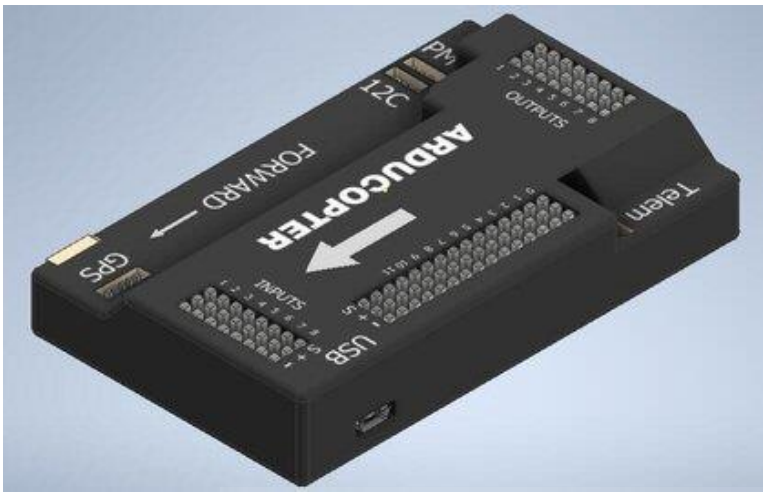


Fig. 4.15: Flight controller

Fig. 4.16: Frame

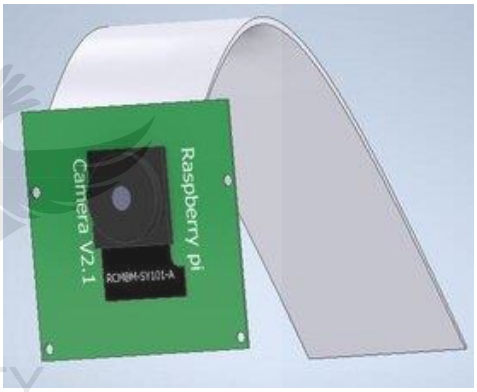


Fig. 4.17: PiCamera

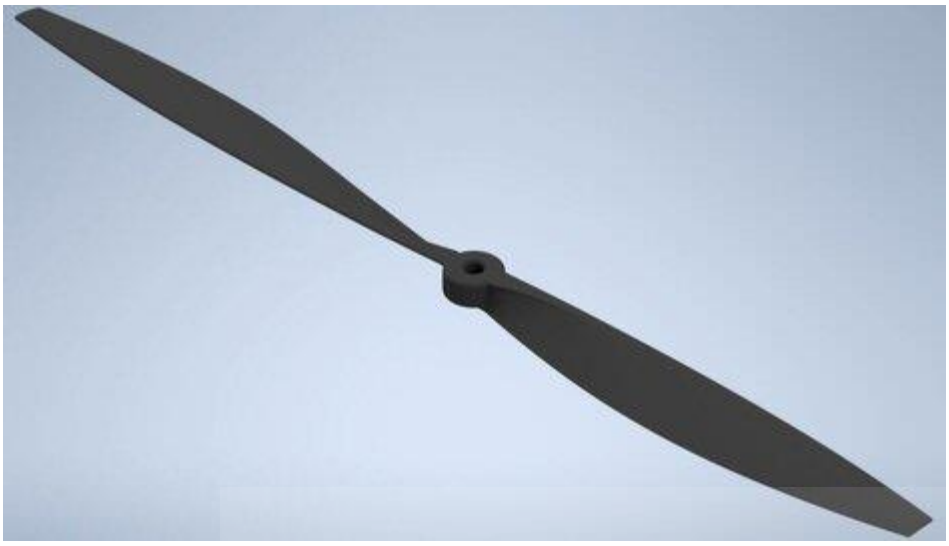


Fig. 4.18: Propeller

Fig. 4.19: Radio receiver

Fig. 4.20: Radio transmitter

Fig. 4.21: Spray pump

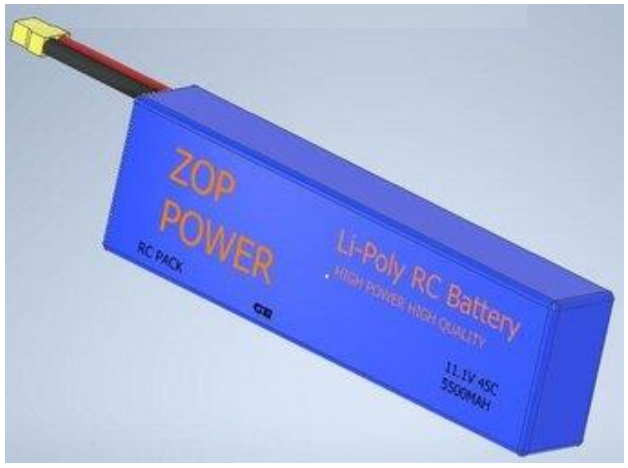


Fig. 4.22: Lithium Polymer

4.8.1 Tools required

The following tools were required to aid with assembling the smart herbicide sprayer:

- Soldering iron and soldering flux
- Screwdriver
- Component vice for keeping the components in place while working
- Wire strippers
- Screws
- Heat gun/lighter
- Scissors
- Needle-nose plier
- Multimeter

4.8.2 Procedures for Assembling

- Solder the ESCs to the bottom power distribution board. The red wire of the ESC should be soldered on the positive (+) contact while the black wire should be soldered on the negative (-) contact. Similarly, wires of the battery should be soldered on the power distribution board.
- Fix the four arms of the frame and landing gears to the bottom power distribution board with screws. Thereafter, the motors should be mounted on the frame using screws. It should be ensured that the screws do not touch the copper wires in the motor.
- Solder the male bullet plugs on each of the three wires of the brushless motors and female bullet plugs on each of the three wires of the ESCs. Then put a heat shrink tube on the bullet plugs and shrink the tube using a lighter or heat gun.
- Mount the top board onto the frame using screws. Then fix the APM 2.8 flight controller, already placed in a vibration absorber plate, onto the top board of the frame. The M8N GPS should also be mounted on the GPS holder placed alongside the flight controller.
- Connect the ESC cable set to channels 1 to 4 at the input section of the flight controller. Also, connect the 3-wired cable set from channels 1 to 4 on the receiver to channels 1 to 4 at the output region of the flight controller. Thereafter, connect the GPS to the flight controller.

- Connect the ESCs to the motors and then calibrate the accelerometer, radio, GPS, and transmitter by connecting the flight controller to a PC with the mission planner software already installed.
- Calibrate the ESCs by connecting each ESC to the throttle channel (channel 3) of the receiver and also connect it to the battery. Then using the transmitter, take the joystick to full throttle and wait till a beep sound is made by the ESC before disconnecting.
- Reconnect all components of the quadcopter and check if all motors are rotating in the desired direction. Ideally, two motors should rotate in the clockwise direction while the other two opposite motors should rotate in the anticlockwise direction. If they are not moving as expected, swap any of the two wires of the ESC connected to the motors.
- Fix the propellers on the shaft of the motor using the motor accessories and connect the battery fastened onto the frame with the battery straps.
- Connect the GPIO 17 and 18 of the raspberry pi, each to the signal contacts of the relay by soldering. Also, connect pin 2 and pin 4 each to the VCC contacts of the relay. Pin 6 and pin 34 should be connected to the ground contact of each relay. Then connect the positive wire of the spray pumps to the 'normally open' contacts of each relay and connect the 12V DC power source to the common of the relay. The ground or negative end of both the 12V power source and spray pump should be connected.
- Ensure that the picamera is well placed on the CSI port of the raspberry pi. Also, connect an 8mm water hose from the spray pump to the tank.
- Fasten the raspberry pi, relay, spray pumps, and tanks on the assembled quadcopter. A setup design of the smart selective herbicide sprayer using Autodesk Inventor is shown in Fig. 4.23.

Fig. 4.23 Overview of the smart herbicide sprayer

4.8.3 Pre-flight checks

Before any flight operation, the following check should be carried out:

- Inspect the frame for any physical damage or crack
- Inspect the motors and propellers for damage or presence of debris and ensure that both are secured correctly.
- Inspect all electrical components for correct functioning.
- Inspect the installation of parts that are removable such as the battery.

4.9 Principle of operation

The smartweed detector would operate in such a way that pictures taken by the picamera, connected to the raspberry pi which is mounted on the quadcopter, would be analyzed by the deployed CNN model. When a broadleaf weed is detected, the GPIO 18 is activated to supply 3.3V for 3 seconds. This voltage (from GPIO 18) and the 5V voltage common collector (VCC) would be input to a DC relay module. When the signal voltage from GPIO 18 is supplied to the relay, it

clicks on to connect the spray pump1 with 12V DC power. The spray pump in turn draws a broadleaf weed herbicide from the tank1 and pumps it through the nozzle. On the other hand, when a grass weed is detected, the GPIO 17 is activated to supply 3.3V for 3 seconds. This voltage is also combined with 5V from the raspberry pi's VCC to turn on a relay which in turn connects spray pump2 to a 12V DC power. The spray pump2 then draws grass weed herbicides from tank2 and pumps it through the nozzle. However, when a soybean plant is detected, no GPIO pin is activated and hence no spraying action occurs. Fig. 4.24 shows a block diagram for the principle of operation.

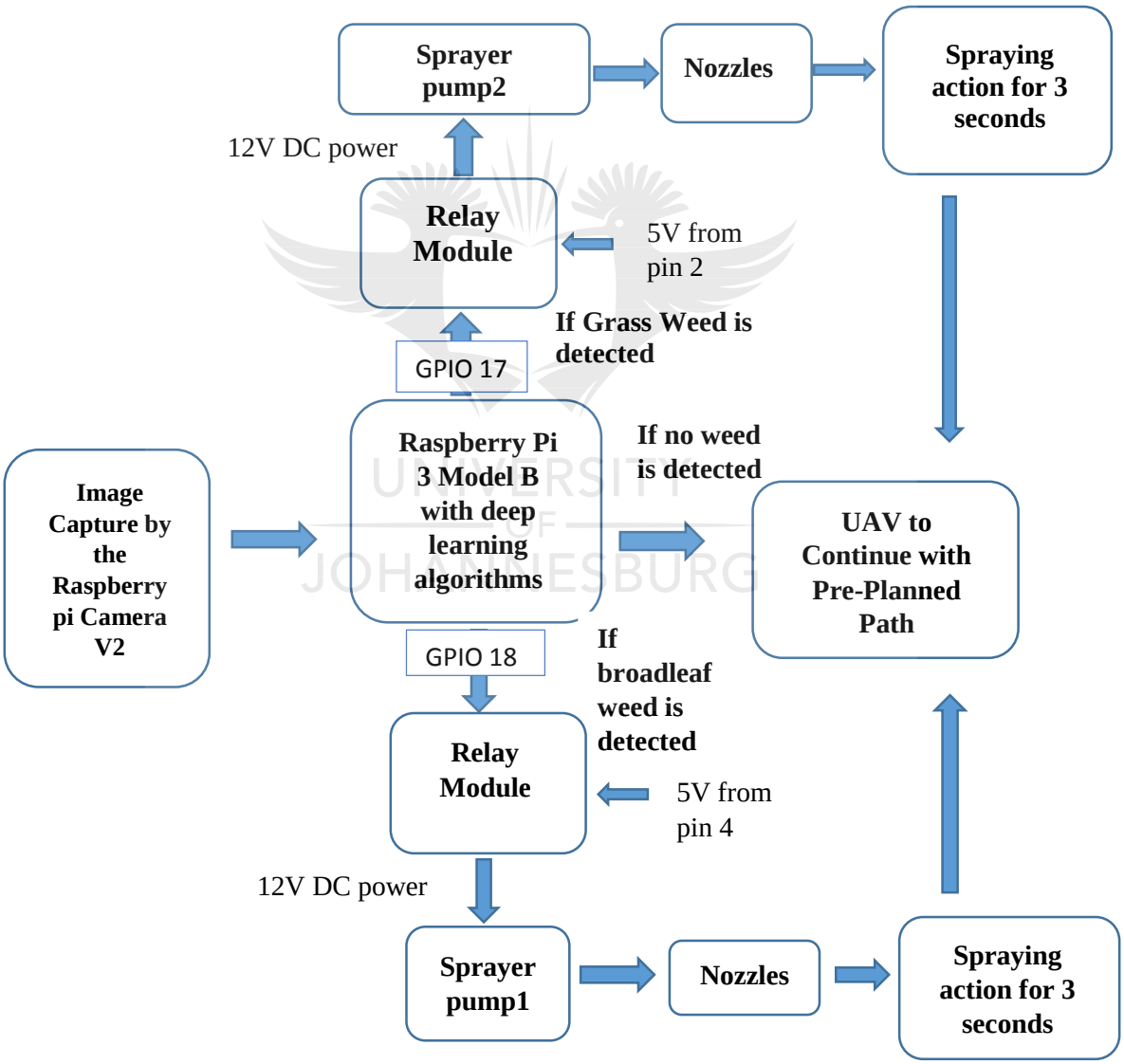


Fig 4.24: Schematic diagram explaining the working principle of the smart herbicide sprayer

4.10 Conclusion

In this chapter, the materials used and methods were discussed. It described how the datasets were obtained. The setting of the software/libraries are discussed and an overview of the setup procedure of the raspberry pi 3B are provided. Furthermore, this section gives details about the compatibility testing and the calculations undertaken to ensure the assembly model of the smart herbicide sprayer was appropriate. Most importantly, details about the assembly model and its mode of operation are disclosed. The ensuing chapter would describe the results obtained from training the CNN model and test running the smart herbicide sprayer.

CHAPTER 5 - RESULTS AND DISCUSSION

5.1 Introduction

This chapter presents and discusses the results obtained from this research study. It provides details of the metrics used to measure the accuracy of the proposed model. In addition, performance of the sprayer module with the raspberry pi is provided. Lastly, evidence of the quadcopter performance are included in this section.

5.2 Outcome of the Model Training Procedure

The CNN model generated 8196 trainable parameters from training 6109 images of input shape 224×224×3. After the training procedure, training, and validation accuracies of 99.98% and 98.4% respectively were obtained. Also, training and validation losses of 0.0039 and 0.0323 were obtained as well. However, a classification accuracy of 95.3% was recorded for the RF classifier. Fig. 5.1 and Fig. 5.2 shows the accuracy and loss graphs of the CNN model.

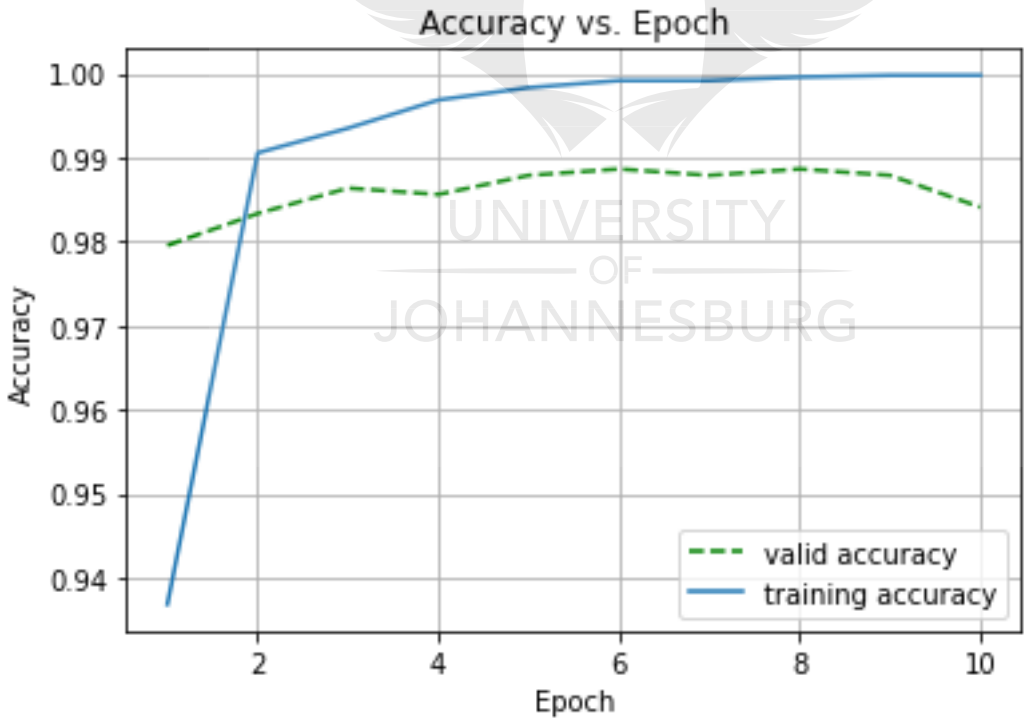


Fig. 5.1: Accuracy vs epoch for the CNN model

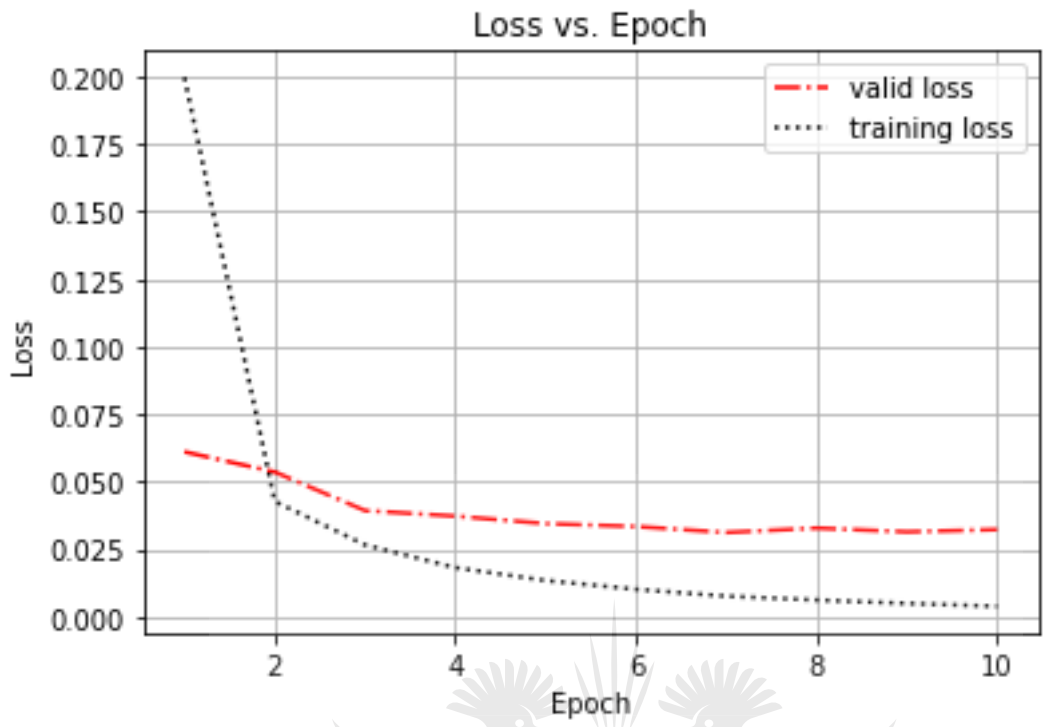


Fig. 5.2: Loss versus epoch for the CNN model

According to the accuracy versus epoch and loss versus epoch graphs above, it was noticed that the best accuracy was achieved at the 9th epoch. A low variance error, which is the difference between the training and validation accuracies, of 1.1% was recorded hence there was no over-fitting of the training data. Also, the training and validation losses were below 2.5% and 5% respectively depicting that the model learned the features perfectly well. Fig. 5.3 and Fig. 5.4 gives the confusion matrix for both the CNN model and RF classifier.

From the confusion matrix of the CNN model, there was just one case of misclassification where a few broadleafs were misclassified for grass weed (Fig. 5.3). This is unlike the situation in the confusion matrix of the RF classifier where there were about five cases of misclassification (Fig. 5.4). In essence, the CNN model outperformed the random forest classifier.

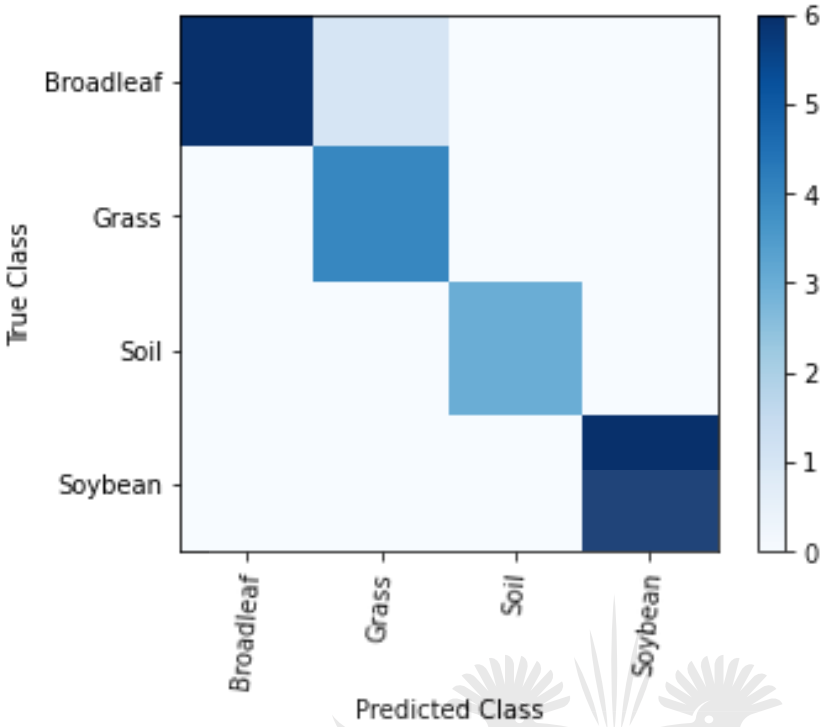


Fig. 5.3 Confusion matrix for the CNN model

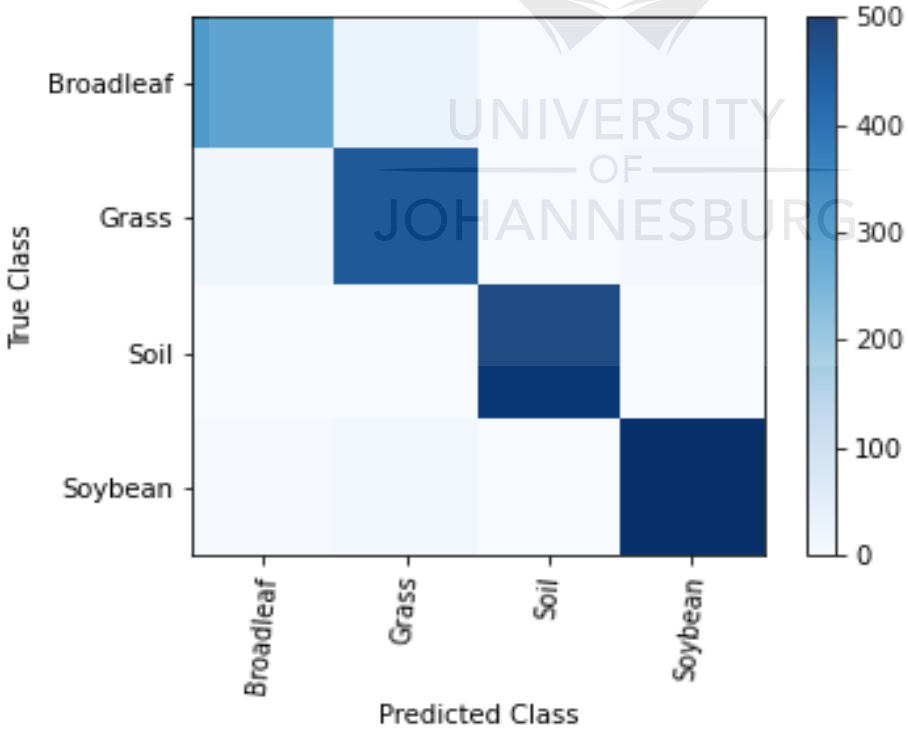


Fig. 5.4: Confusion matrix for the random forest classifier

5.3 Outcome of incorporating the sprayer module with the raspberry pi

The raspberry pi 3 with the CNN model already deployed was linked with the sprayer module and tested for spraying as shown in Fig. 5.5. The positive end of the spray pump was connected to 'normally open' and a 12V DC power source connected to the common of the relay. Also, when broadleaf weed was detected, GPIO 18 was activated to supply 3.3V together with the VCC from the raspberry pi which served as inputs to the relay to turn on spray pump1. Similarly, when grass weed was detected, GPIO17 was activated to supply 3.3 V together with the VCC from the raspberry pi which served as inputs to the relay to turn on spray pump2.

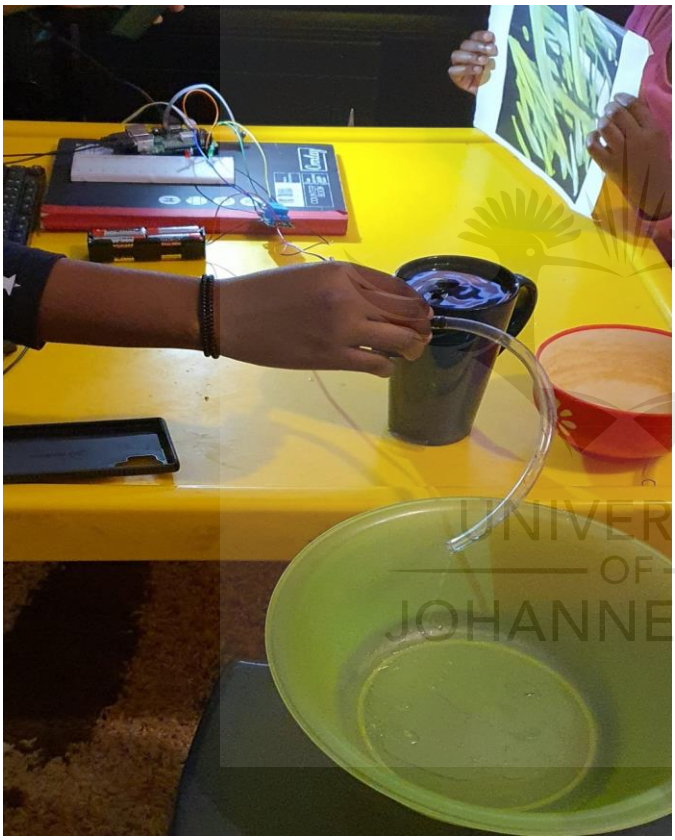


Fig. 5.5: Testing the sprayer module

5.4 Assembling the Quadcopter kit

The procedures for assembling a quadcopter kit as explained in chapter 4 was strictly followed during the quadcopter building. Fig. 5.6, Fig. 5.7, and Fig. 5.8 show some steps followed during the course of building the quadcopter. The quadcopter was first tested and was found to be in good

condition with a time of flight of about 15 minutes. Fig. 5.9 shows the quadcopter during its first test-run.

Fig. 5.6: Calibrating the ESC

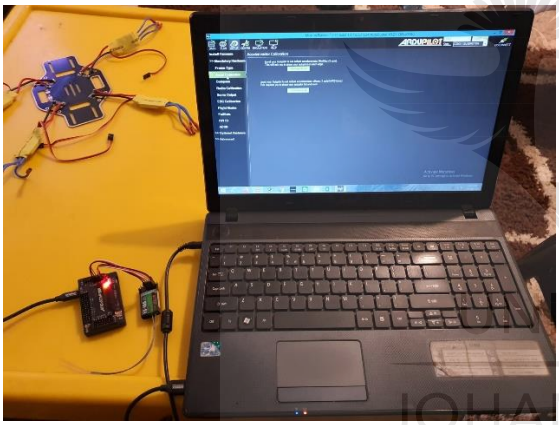


Fig. 5.7: Calibrating the flight controller

Fig. 5.8: Assembled quadcopter

Fig. 5.9: First test-run of the quadcopter

5.5 Test-running the smartweed detector and selective herbicide sprayer

The sprayer module together with the raspberry pi was incorporated on the assembled quadcopter to form the smartweed detector and selective herbicide sprayer as shown in Fig. 5.10. A final test-run was carried out on the smart herbicide sprayer and it was observed that it could detect broadleaf weed or grass weed accurately in less than a second at a height of 50cm above the ground and spray herbicides accordingly. However, spraying of herbicides occurs after two weed detection attempts. This means that the sprayer has to detect a weed twice consecutively before spraying the appropriate herbicide. Although the smart sprayer commences spraying operation immediately after a broadleaf or grass weed is detected, it sprays for 10 seconds instead of the envisaged 3seconds. Fig. 5.12 displays the smart selective herbicide sprayer during the test-running operation.

Fig. 5.10: Quadcopter with the sprayer module incorporated

Fig. 5.11: Test-running the smartweed detector and selective herbicide sprayer

CHAPTER 6- CONCLUSION AND RECOMMENDATIONS

6.1 Conclusion

This research work provides details about the development of a smartweed detector and selective herbicide sprayer which detects and sprays on weeds accordingly. This was achieved by deploying deep learning algorithms on an embedded system mounted on a quadcopter. As a step in the right direction in the domain of precision agriculture, it envisages solving the problem of poor harvest due to competition for nutrients by weeds and effective elimination of weeds through its selective spraying ability. This chapter gives a detailed report on the achievements of this research work and also proffer noteworthy recommendations for prospects. The achievements recorded in this research work are given as follows:

- **Results from preliminary study conducted**: From the preliminary study conducted, a system was developed to detect potassium deficiency in red grape vine and prompt a spraying actuation. Training and validation accuracies of 89% and 81% respectively was obtained. Also, training and validation losses of 0.2205 and 0.4720 were obtained respectively. Thereafter, an LED lit up when a potassium deficient leaf was brought close to the picamera which served as the spraying actuation.
- **Training a CNN model and deploying it on a raspberry pi 3**: For the main research work, the CNN model was trained through transfer learning on the soybean dataset. The CNN model was evaluated with the RF classifier, converted to TensorFlow lite format, and deployed on the raspberry pi 3. Training and validation accuracies of 99.98% and 98.4% were obtained respectively with an insignificant variance error which also surpassed the accuracy gotten from the RF classifier. Also, training and validation losses of 0.0039 and 0.0323 respectively reveals that the proposed model was appropriate.
- **Building a sprayer module**: A sprayer module which consists of a relay, raspberry pi 3, spray pump, 12V DC source, water hose, and the tank was built. It operated in such a way that when a weed is detected based on the deep learning algorithms deployed on the raspberry pi, GPIO 17 or GPIO 18 were activated to supply 3.3V which turned on a DC relay to spray herbicides accordingly .
- **Building the quadcopter**: The quadcopter components were assembled with light-weight and strong materials being used to improve its efficiency. It consisted of the electric motor,

ESC, propellers, frame, li-po battery, flight controller, GPS, receiver. In this research work, a quadcopter was built and was able to perform as pre-planned.

- **Incorporating the sprayer module on the quadcopter and test-running the smart herbicide sprayer**: The sprayer module was mounted on the quadcopter and from the test-running operation carried out, broadleaf and grass weed were accurately detected and spraying of herbicides according to the weed type occurred in less than a second.

Although the aim of the research was achieved, there were a few highlighted variations from what was anticipated. This has to do with the time interval for spraying the herbicides and spraying of the appropriate herbicide at the second attempt when there is an immediate change in weed type detected. The latter is due to a 'clean-up' issue on the raspberry pi as it correctly detects the right weed but fails to accurately activate the appropriate GPIO pin at the first attempt. Nevertheless, the results obtained from this work create the opportunity for more research to be carried out in this field of precision agriculture for further improvements.

6.2 Recommendations

In a bid to establish a basis on which improvements could be made in this research work, the following recommendations were made:

- The use of an RGB camera with a higher resolution quality would enable a better picture capture from a higher height above the ground and hence a more accurate detection at higher heights.
- Employing the use of DL hardware accelerators such as the Intel Movidius compute stick would lead to a reduced inference time which is an important criterion for real-time detection.
- Implementing DL algorithms on better and more efficient embedded systems such as the Nvidia Jetson TX2, FPGA, etc would improve the spraying actuation challenge experienced with the raspberry pi in this work.
- The field of DL is broad and evolving with swift advancement in technology hence extensive research on the use of better DL detection algorithms, such as YOLO, is imperative.

- Powering the smart herbicide sprayer with a renewable energy source such as solar would increase the flight time as the estimated time of flight with batteries is about 15-20 minutes.
- Making the smart herbicide sprayer autonomous by planning its flight path using a grand control station.

References

1. Abbe G. and Smith H. (2016). Technological Development Trends in Solar-Powered Aircraft Systems. *Renewable and Sustainable Energy Reviews*; 60, pp. 770-783.
2. Abdel-zaher, A. M. and Eldeib, A. M. (2016) 'Breast cancer classification using deep belief networks', *Expert Systems With Applications*. Elsevier Ltd, 46, pp. 139–144. doi: 10.1016/j.eswa.2015.10.015.
3. Ahmad, J. *et al.* (2018) 'Computers in Industry Visual features based boosted classification of weeds for real-time selective herbicide sprayer systems', *Computers in Industry*. Elsevier B.V., 98, pp. 23–33. doi: 10.1016/j.compind.2018.02.005.
4. Akbarzadeh, S. *et al.* (2018) 'Plant discrimination by Support Vector Machine classifier based on spectral reflectance', *Computers and Electronics in Agriculture*, 148(March), pp. 250–258. doi: 10.1016/j.compag.2018.03.026.
5. Alippi, C. *et al.* (2018) 'Moving Convolutional Neural Networks to Embedded Systems : the AlexNet and VGG-16 case', pp. 212–223. doi: 10.1109/IPSN.2018.00049.
6. Allaire, J.J., (2016). TensorFlow for R.
7. Allan A. (2017). First Thought on the New UP Core. Viewed on 4th March 4, 2020 from https://hackaday.com/2017/06/02/first-thoughts-on-the-new-up-core/
8. Al-Rfou, R., (2016). 'Theano: A Python framework for fast computation of mathematical expressions'. *arXiv preprint arXiv:1605.02688*
9. Alsalam, B.H.Y., et al (2017). 'Autonomous UAV with vision based on-board decision making for remote sensing and precision agriculture'. In *2017 IEEE Aerospace Conference* (pp. 1-12). IEEE.
10. Amanullah, M.A. *et al.* (2020) 'Deep learning and big data technologies for IoT security', *Computer Communications*. Elsevier B.V., 151(October 2019), pp. 495–517. doi: 10.1016/j.comcom.2020.01.016.
11. Amodei D. *et al.* (2016). 'Deep Speech 2 : End-to-End Speech Recognition in English and Mandarin' *Proceedings of The 33rd International Conference on Machine Learning*.48.pp173-182
12. Andrea, C.C., Daniel, B.B.M. and Misael, J.B.J., (2017), October. Precise weed and maize classification through convolutional neuronal networks. In *2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM)* (pp. 1-6). IEEE.

13. Aradi, S., (2020). Survey of Deep Reinforcement Learning for Motion Planning of Autonomous Vehicles. *arXiv preprint arXiv:2001.11231*

14. Bahdanau D., Cho, K., Bengio Y. (2015). 'Neural Machine Translation by Jointly Learning to Align and Translate'. *In Proceedings of International Conference on Learning Representations (ICLR)*. Pp.1-15.
15. Bakhshipour, A. *et al.* (2017) 'ScienceDirect Weed segmentation using texture features extracted from wavelet sub-images', *Biosystems Engineering*. Elsevier Ltd, 157, pp. 1–12. doi: 10.1016/j.biosystemseng.2017.02.002.
16. Baratella A. and Gomes De Melo M. (2017). 'Computer Vision and Artificial Intelligence Techniques Applied to Robot Soccer'. *International Journal of Innovative, Computing, Information and Control*. Vol 13(3), pp 991-1005.ISSN 1349-4918.
17. Bechtel, M.G.,et al. (2018). 'Deeppicar: A low-cost deep neural network-based autonomous car'. In *2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)* (pp. 11-21). IEEE.
18. Bhargava, A. and Bansal, A. (2018) 'Fruits and vegetables quality evaluation using computer vision : A review', *Journal of King Saud University - Computer and Information Sciences*. The Authors. doi: 10.1016/j.jksuci.2018.06.002.
19. Bhattacharya, S. *et al.* (2017) 'Low-resource Multi-task Audio Sensing for Mobile and Embedded Devices via Shared Deep Neural Network Representations', *2017 ACM Interactive Mobile Wearable Ubiquitous Technologies* 1(3).http://doi.org/10.1145/3131895
20. Bojarski, M., et al (2016). End to end learning for self-driving cars.arXiv, Cornell University:1604.07316.
21. Brunelli, D., et al. (2019). 'Energy Neutral Machine Learning Based IoT Device for Pest Detection in Precision Agriculture'. *IEEE Internet of Things Magazine*, *2*(4), pp.10-13.
22. Cannon L. (2018). 'Drones Deliver Data on Oyster Reef Health'. viewed on 4th 2020 from https://ncspacegrant.ncsu.edu/2018/09/25/drones-deliver-data-on-oysters/
23. Chao, H., Cao, Y. and Chen, Y. (2010) 'Autopilots for Small Unmanned Aerial Vehicles : A Survey', 8, pp. 36–44. doi: 10.1007/s12555-010-0105-z.
24. Chechliński, Ł., Siemiątkowska, B. and Majewski, M., (2019). 'A System for Weeds and Crops Identification—Reaching over 10 FPS on Raspberry Pi with the Usage of

MobileNets, DenseNet and Custom Modifications'. *Sensors*, *19*(17), p.3787.

25. Cheng, X., et al (2017). 'Pest identification via deep residual learning in complex background'. *Computers and Electronics in Agriculture*, *141*, pp.351-356.
26. Chollet, F. *et al.* (2015) keras, GitHub.URL: https://github.com/fchollet/keras
27. Coulibaly, S. *et al.* (2019) 'Deep neural networks with transfer learning in millet crop images', *Computers in Industry*. Elsevier B.V., 108, pp. 115–120. doi: 10.1016/j.compind.2019.02.003.
28. De Rainville, F., *et al.* (2014) 'Bayesian classification and unsupervised learning for isolating weeds in row crops'. *Pattern Analysis and Application* **17,** 401–414. https://doi.org/10.1007/s10044-012-0307-5
29. *Dey, S. et al. (2019)* 'Embedded Deep Inference in Practice : Case for Model Partitioning*', 1st Workshop on Machine Learning on Edge in Sensor Systems (SenSys-ML 2019)*, pp. 25–30. https://doi.org/10.1145/3362743.3362964
30. Ding, L. *et al.* (2020) 'Automation in Construction Computer vision applications in construction safety assurance Weili Fang', *Automation in Construction*. Elsevier, 110(September 2019), p. 103013. doi: 10.1016/j.autcon.2019.103013.
31. Dos Santos Ferreira, A., et al (2017).'Weed detection in soybean crops using ConvNets'. *Computers and Electronics in Agriculture*, *143*, pp.314-324.
32. Dryden J. and R. Barbaccia (2014). Quadcopter Design Project
33. Elfwing, S., Uchibe, E. and Doya, K., (2015). 'Expected energy-based restricted Boltzmann machine for classification'. *Neural networks*, *64*, pp.29-38
34. Escola A. (2013). Variable rate sprayer. Part 1– Orchard Prototype: Design, Implementation and validation. *Computers and Electronics in Agriculture* 95, pp. 122-135.
35. Feng, X. *et al.* (2019) 'Computer vision algorithms and hardware implementations : A survey', *Integration, the VLSI Journal*. Elsevier Ltd, (June).doi: 10.1016/j.vlsi.2019.07.005.
36. Ferentinos, K.P., (2018). 'Deep learning models for plant disease detection and diagnosis'. *Computers and Electronics in Agriculture*, *145*, pp.311-318.
37. Fernández-Delgado, M. et al (2014). Do we need hundreds of classifiers to solve real world classification problems? *The journal of machine learning research*, *15*(1), pp.3133-3181.
38. Franklin D. (2015). 'Nvidia Jetson Supercomputer-on-Modules-Drive Next Wave of

Autonomous Machines. Viewed on 4th March, 2020 from https://devblogs.nvidia.com/nvidia-jetson-tx1-supercomputer-on-module-drives-next-wave-of-autonomous-machines/

39. Franklin D., (2017). 'Nvidia Jetson TX2 Delivers Twice the Intelligence to the Edge'. Viewed on 4th March, 2020 from https://devblogs.nvidia.com/jetson-tx2-delivers-twice-intelligence-edge/
40. Gao, X. (2015). Reviews of methods to extract and store energy for solar-powered aircraft. *Renewable and Sustainable Energy Reviews* 44, pp. 96-108.
41. Garci, D., Pajares, G. (2017) 'ScienceDirect On-line crop / weed discrimination through the Mahalanobis distance from images in maize fields', *Biosystems Engineering*, Elsevier Ltd, 166, pp.28-43. doi: 10.1016/j.biosystemseng.2017.11.003.
42. Garratt, M. A. *et al* (2020) 'Towards the use of fuzzy logic systems in rotary wing unmanned aerial vehicle : a review', *Artificial Intelligence Review*. Springer Netherlands, 53(1), pp. 257–290. doi: 10.1007/s10462-018-9653-z.
43. Ghazbi, S. N. (2016) 'Quadrotors Unmanned Aerial Vehicles : A Review', *International Journal on Smart Sensing & Intelligent Systems*, 9(1), Pp. 309–333.
44. Gondchawar, N., Kawitkar, R.S. (2016) 'IoT based Smart Agriculture', *International Journal of Advanced Research in Computer and Communication Engineering*, 5(6), pp. 838–842. doi: 10.17148/IJARCCE.2016.56188.
45. Gopalakrishnan E. (2017). *Quadcopter Flight Mechanics Model and Control Algorithms* (masters thesis). Department of Control Engineering, Lulea University of Technology, Czech Technical University
46. Gu, S., et al (2018). 'A deep learning tennis ball collection robot and the implementation on nvidia jetson tx1 board'. In *2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)* (pp. 170-175). IEEE.
47. Guerrero, J. E., *et al.*, (no date) 'Preliminary design of a small-sized flapping UAV . II . Aerodynamic Performance and Flight Stability', pp. 1–10.
48. Gupte, S. and Conrad, J. M. (2012) 'A Survey of Quadrotor Unmanned Aerial Vehicles', *2012 Proceedings of IEEE Southeastcon*. IEEE, pp. 1–6. doi: 10.1109/SECon.2012.6196930.
49. Hadidi, R., et al (2018). Musical chair: Efficient real-time recognition using collaborative

iot devices. *arXiv preprint arXiv:1802.02138*.

50. Hafsal L. P. (2016). *Precision Agriculture with Unmanned Aerial Vehicles for SMC estimations – Towards a more Sustainable Agriculture* (masters thesis). Department of Applied Ecology and Agriculture, Hedmark University of Applied Sciences.
51. Han, S. *et al.* (2016) 'EIE : Efficient Inference Engine on Compressed Deep Neural Network', *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture*. pp 243-254. doi: 10.1109/ISCA.2016.30.
52. Harsha, S., Balaji and Kamath D. (2014). Quadcopter. Viewed on 4th March 4, 2020 from: http://socialledge.com/sjsu/index.php/S14:_Quadcopter.
53. Hassan, A. and Mahmood, A., (2018). 'Convolutional recurrent deep learning model for sentence classification'. *IEEE Access*, *6*, pp.13949-13957.
54. Hattersley, L. (2018)**,** 'Raspberry Pi 3B+ Specs and Benchmarks' viewed 5th February 2020 from https://magpi.raspberrypi.org/articles/raspberry-pi-3bplus-specs-benchmarks
55. Hattersley, L. (2018), 'Raspberry Pi 4, 3A+, Zero W - specs, benchmarks & thermal tests'. Viewed 5th February 2020 from https://magpi.raspberrypi.org/articles/raspberry-pi-specs-benchmarks
56. He K. *et al* (2016). 'Deep Residual Learning for Image Recognition' *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* pp. 770-778.
57. Hinton, G., Vinyals, O. and Dean, J. (2015) 'Distilling the Knowledge in a Neural Network', pp. 1–9.
58. Hinton, G.E., Osindero, S. and Teh, Y.W., (2006). 'A fast learning algorithm for deep belief nets'. *Neural computation*, 18(7), pp.1527-1554.
59. Hoang, M. and Poon, T. W. (2013) *Final Report Design , Implementation , and Testing of a UAV Quadcopter*. (Bachelors of Engineering Project). Department of Electrical and Computer Engineering, University of Manitibia
60. Howard C., (2014). 'NVIDIA Jetson TK1 Development Kit brings Mobile Supercomputer to Avionics Systems'. Viewed on 4th March 4, 2020 from https://www.militaryaerospace.com/computers/article/16718650/nvidia-jetson-tk1-development-kit-brings-mobile-supercomputer-to-avionics-systems
61. Howard, A. G. *et al.* (2017) 'MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications', pp 1-9.

62. Huang, L. et al (2018). 'Micro-seismic event detection and location in underground mines by using Convolutional Neural Networks (CNN) and deep learning'. *Tunneling and Underground Space Technology*, *81*, pp.265-276.
63. Jha, K. et al (2019). A comprehensive review on automation in agriculture using artificial intelligence. *Artificial Intelligence in Agriculture*, *2*, pp.1-12.
64. Jia Y. *et al.*(2014) 'Caffe: Convolutional Architecture for Fast Feature Embedding', In *Procceeding of the 22nd ACM International Conference on Multimedia* pp. 675–678. https://doi.org/10.1145/2647868.2654889
65. Kamilaris, A. and Prenafeta-Boldú, F. X. (2018) 'Deep learning in agriculture: A survey', *Computers and Electronics in Agriculture*. Elsevier B.V., pp. 70–90. doi: 10.1016/j.compag.2018.02.016.
66. Kang, Y. *et al.* (2017) 'Neurosurgeon : Collaborative Intelligence Between the Cloud and Mobile Edge'. pp 615-629.DOI: http://dx.doi.org/10.1145/3037697.3037698.
67. Kaya, A. *et al.* (2019) 'Analysis of transfer learning for deep neural network based plant classification models', *Computers and Electronics in Agriculture*, 158(January), pp. 20–29. doi: 10.1016/j.compag.2019.01.041.
68. Kazmi, W. *et al.* (2015) 'Exploiting affine invariant regions and leaf edge shapes for weed detection', *Computers and Electronics in Agriculture*. Elsevier B.V., 118, pp. 290–299. doi: 10.1016/j.compag.2015.08.023.
69. Kim, C. E. *et al.* (2016) 'A Comparison of Embedded Deep Learning Methods for Person Detection'.
70. Krizhevsky, A., Sutskever, I. and Hinton, G.E., (2012). 'Imagenet classification with deep convolutional neural networks'. In *Advances in neural information processing systems* (pp. 1097-1105).
71. Lammie, C. et al (2019). 'Low-Power and High-Speed Deep FPGA Inference Engines for Weed Classification at the Edge'. *IEEE Access*, *7*, pp.51171-51184.
72. Lane, N.D., et al (2017). 'Squeezing deep learning into mobile and embedded devices'. *IEEE Pervasive Computing*, *16*(3), pp.82-88.
73. Larochelle, H. and Bengio, Y., (2008). 'Classification using discriminative restricted Boltzmann machines'. In *Proceedings of the 25th international conference on Machine learning* (pp. 536-543).

74. Leroux, S. *et al.* (2017) 'The cascading neural network: building the Internet of Smart Things', *Knowledge and Information Systems*. Springer London, 52(3), pp. 791–814. doi: 10.1007/s10115-017-1029-1.
75. Li, B., Najafi, M. H. and Lilja, D. J. (2016) 'Using Stochastic Computing to Reduce the Hardware Requirements for a Restricted Boltzmann Machine Classifier', pp. 0–5.
76. Li, Y. and Yang, J., (2020). 'Few-shot cotton pest recognition and terminal realization'. *Computers and Electronics in Agriculture*, *169*, p.105240.
77. Liu, H., Lee, S. H. and Saunders, C. (2014) 'Development of a Machine Vision System for Weed Detection During Both of Off-Season and In-Season in Broadarce No-Tillage Cropping Lands', *American Journal of Agricultural and Biological Sciences*9(2), pp. 174–193. doi: 10.3844/ajabssp.2014.174.193.
78. Liu, W., et al (2017). 'A survey of deep neural network architectures and their applications'. *Neurocomputing*, *234*, pp.11-26.
79. Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y. and Alsaadi, F.E., (2017). A survey of deep neural network architectures and their applications. *Neurocomputing*, *234*, pp.11-26.
80. Lottes, P. *et al.* (2016) 'An Effective Classification System for Separating Sugar Beets and Weeds for Precision Farming Applications', *2016 IEEE International Conference on Robotics and Automation (ICRA) Stockholm, Sweden.* pp. 5157–5163. doi: 10.1109/ICRA.2016.7487720.
81. Ma, X., et al (2019). 'A Survey on Deep Learning Empowered IoT Applications'. *IEEE Access*, *7*, pp.181721-181732.
82. Mackenzie, D., (2012). 'A flapping of wings'. [Online].AAAS Database 335(6075)(March). pp 1430-1433. Available from: https://science.sciencemag.org/content/335/6075/1430. Accessed[4th March 4, 2020]
83. Maes, W.H. and Steppe, K., 2019. Perspectives for remote sensing with unmanned aerial vehicles in precision agriculture. *Trends in plant science*, *24*(2), pp.152-164.
84. Manderson, T., et al (2018). 'Vision-based autonomous underwater swimming in dense coral for combined collision avoidance and target selection'. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 1885-1891). IEEE.

85. Marcus, G., (2018). 'Deep learning: A critical appraisal'. pp1-27 *arXiv preprint arXiv:1801.00631*.
86. Marinello F. *et al.* (2016). Technical Analysis of Unmanned Aerial Vehicles (Drones) For Agricultural Applications.
87. Maurya, H.L., Behera, L. and Verma, N.K., (2019). Trajectory Tracking of Quad-Rotor UAV Using Fractional Order PID Controller. In *Computational Intelligence: Theories, Applications and Future Directions-Volume I*.pp. 171-186. Springer, Singapore.
88. McCarthy C. L., Hancock N. H. and Raine S. R. (2010). 'Applied machine vision of plants – a review with implications for field deployment in Automated farming operations'. *Intelligent Service Robotics,*3 (4), pp. 209-217. ISSN 1861-2776
89. Milioto, A., Lottes, P. and Stachniss, C., (2017). Real-time blob-wise sugar beets vs weeds classification for monitoring fields using convolutional neural networks. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *4*, p.41.
90. Miranda, J. et al, (2019). Sensing, smart and sustainable technologies for Agri-Food 4.0. *Computers in Industry*, *108*, pp.21-36.
91. Misra, A., (2019). 'Deep Learning Acceleration on the Edge'.
92. Mittal, S. (2019) 'A Survey on optimized implementation of deep learning models on the NVIDIA Jetson platform', *Journal of Systems Architecture*. Elsevier B.V., 97(December 2018), pp. 428–442. doi: 10.1016/j.sysarc.2019.01.011.
93. Mnih, V., Larochelle, H. and Hinton, G.E., (2012). 'Conditional restricted boltzmann machines for structured output prediction'. *arXiv preprint arXiv:1202.3748*.
94. Mohammadi, M. *et al.* (2018) 'Deep Learning for IoT Big Data and Streaming Analytics : A Survey'. IEEE, 20(4), pp. 2923–2960.
95. Mohanty, S. P., Hughes, D. P. and Salathé, M. (2016) 'Using Deep Learning for Image-Based Plant Disease Detection', *Frontiers in Plant Science*. doi: 10.3389/fpls.2016.01419
96. Mu, R. and Zeng, X., (2019). 'A Review of Deep Learning Research'. *TIIS*, *13*(4), pp.1738-1764
97. Murtaza, G., et al (2019). 'Deep learning-based breast cancer classification through medical imaging modalities: state of the art and research challenges'. *Artificial Intelligence Review*, pp.1-66.
98. Nevo, S., e al (2019). 'ML for flood forecasting at scale'. *arXiv preprint arXiv:1901.09583*.

99. Nikam, A.D., (2018). *Enabling Architecture Research on GPU Simulator for Deep Learning Applications* (Doctoral dissertation, The University of North Carolina at Charlotte).
100. Nixon A. (2017). 'Three Great FPV Drones for $100, $200 and $300'. Viewed on 5th March, 2020 from https://bestdroneforthejob.com/blog/three-great-fpv-racing-dronekits-assembly-required/
101. Noda, K., et al (2015). 'Audio-visual speech recognition using deep learning'. *Applied Intelligence, 42*(4), pp.722-737.
102. Ojha, T., Misra, S. and Raghuwanshi, N.S., (2015). Wireless sensor networks for agriculture: The state-of-the-art in practice and future challenges. *Computers and Electronics in Agriculture, 118*, pp.66-84.
103. Pan, S. J. and Yang, Q. (2010) 'A Survey on Transfer Learning'. *IEEE Transactions on Knowledge and Data Engineering*. IEEE, 22(10).pp. 1345-1359.
104. Pandey, S.K. and Janghel, R.R., (2019). 'Recent deep learning techniques, challenges and its applications for medical healthcare system: A review'. *Neural Processing Letters, 50*(2), pp.1907-1935.
105. Park, J., et al (2019). 'Algal morphological identification in watersheds for drinking water supply using neural architecture search for convolutional neural network'. *Water, 11*(7), p.1338.
106. Paszke A. *et al.* (2017). 'Automatic Differentiation in Pytorch' *In Proceeding of 31st Conference on Neural Information Processing System CA, USA.* Pp1-4.
107. Patel, D. C. *et al.* (2013) 'Design of Quadcopter in Reconnaissance', *International Conference on Innovations in Automation and Mechatronics Engineering.* pp. 21–23.
108. Pillai, T.R., et al (2018). 'Credit Card Fraud Detection Using Deep Learning Technique'. In *2018 Fourth International Conference on Advances in Computing, Communication & Automation (ICACCA)*.pp. 1-6. IEEE.
109. Pool, J. and Dally, W. J. (no date) 'Learning both Weights and Connections for Efficient Neural Networks', pp. 1–9.
110. Potena, C., Nardi, D. and Pretto, A., (2016). 'Fast and accurate crop and weed identification with summarized train sets for precision agriculture'. In *International Conference on Intelligent Autonomous Systems* (pp. 105-121). Springer, Cham.

111. Pouyanfar, S. *et al.* (2018) 'A Survey on Deep Learning : Algorithms , Techniques '. *ACM Computing Surveys* 51(5). pp. 1-36. https://doi.org/10.1145/3234150.
112. Pouyanfar, S., Chen, S.C. and Shyu, M.L., (2017). 'An efficient deep residual-inception network for multimedia classification'. In *2017 IEEE International Conference on Multimedia and Expo (ICME)* (pp. 373-378). IEEE.
113. Probst L., Pedersen B. & Dakkak-Arnoux L. (2017). Drones in agriculture. A report prepared for the European Commission by consortium composed of PwC, IDATE and ESN.*Procceeding of the 22nd ACM International Conference on Multimedia* pp. 675–678. https://doi.org/10.1145/2647868.2654889
114. Qin, W.C. *et al.* (2018) Droplet deposition and efficiency of fungicides sprayed with small UAV against wheat powdery mildew. *International Journal of Agricultural and Biological Engineering*, 11(2), pp 27-
115. Rangel, B.M.S., et al (2016). 'KNN-based image segmentation for grapevine potassium deficiency diagnosis'. In *2016 International Conference on Electronics, Communications and Computers (CONIELECOMP)* (pp. 48-53). IEEE.
116. Rao, R.N. and Sridhar, B., (2018), January. IoT based smart crop-field monitoring and automation irrigation system. In *2018 2nd International Conference on Inventive Systems and Control (ICISC)* (pp. 478-483). IEEE.
117. Rehman M. *et al.* (2016). Quadcopter for Pesticide Spraying. *International Journal of Scientific & Engineering Research*, Vol 7(5), pp 238-240. ISSN 2229-5518.
118. Reséndiz, V.M.A. and Rivas-Araiza, E.A., (2016). 'System Identification of a Quad-rotor in X Configuration from Experimental Data'. *Research in Computing Science*, *118*, pp.77 86.
119. Rioku (2018). 'Long Endurance Fixed Wing Drone' viewed on 4th March 4, 2020 from http://www.regimage.org/long-endurance-fixed-wing-drone/
120. Sa, I. et al (2017). 'Weednet: Dense semantic weed classification using multispectral images and MAV for smart farming'. *IEEE Robotics and Automation Letters*, *3*(1), pp.588-595.
121. Sak, H., Senior, A. and Beaufays, F. (no date) 'Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling Has'.
122. Saleem, T. J., Chishti, M. (2019) 'Deep Learning for Internet of Things Data Analytics

Deep Learning for Internet of Things Data Analytics', *Procedia Computer Science*. Elsevier B.V., 163, pp. 381–390. doi: 10.1016/j.procs.2019.12.120.

123. Sengupta, S., *et al.* (2020). A review of deep learning with special emphasis on architectures, applications and recent trends. *Knowledge-Based Systems*. Elsevier B.V.pp.105596. https://doi.org/10.1016/j.knosys.2020.105596.
124. Shadrin, D., et al (2019). 'Designing Future Precision Agriculture: Detection of Seeds Germination Using Artificial Intelligence on a Low-Power Embedded System'. *IEEE Sensors Journal*, *19*(23), pp.11573-11582.
125. Shivaji C. P. *et al.* (2017). Agriculture Drone for Spraying Fertilizer and Pesticides. *International Journal for Research Trends and Innovation*, 2(6), ISSN: 2456-3315
126. Shraim, H., Awada, A. and Youness, R. (2018) 'Feature Article : A Survey on Quadrotors : Configurations , Modeling and Identification , Control , Collision Avoidance, Fault Diagnosis and Tolerant Control', *IEEE Aerospace and Electronic Systems Magazine*. IEEE, 33(10), pp. 14–33. doi: 10.1109/MAES.2018.160246.
127. Shrestha, A. and Mahmood, A., (2019). 'Review of deep learning algorithms and architectures'. *IEEE Access*, *7*, pp.53040-53065.
128. Simonyan, K. and Zisserman, A., (2014). 'Very deep convolutional networks for large scale image recognition'. *arXiv preprint arXiv:1409.1556*.
129. Smith, P. *et al.* (1979) 'A Threshold Selection Method from Gray-Level Histograms', 20(1), pp. 62–66.
130. Sullivan, J. M. (2006) 'Evolution or Revolution ? The Rise of UAVs'. IEEE Technology and Society Magazine, vol. 25, no. 3, pp. 43-49.
131. Sun Y. *et al.* (2014). 'Deep learning Face Representation by Joint Identification Verification'. *Proceeding of Neural Information Processing System Conference.* Pp. 1-9.
132. Suriansyah, M. I., Sukoco, H. and Solahudin, M. (2016) 'Weed detection using fractal based low cost commodity hardware Raspberry Pi', *Indonesian Journal of Electrical Engineering and Computer Science*, 2(2), pp. 426–430. doi: 10.11591/ijeecs.v2.i2.pp426 430.
133. Szegedy, C., et al., (2015). 'Going deeper with convolutions'. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).
134. Taheri-garavand, A. *et al.* (2019) 'Meat quality evaluation based on computer vision

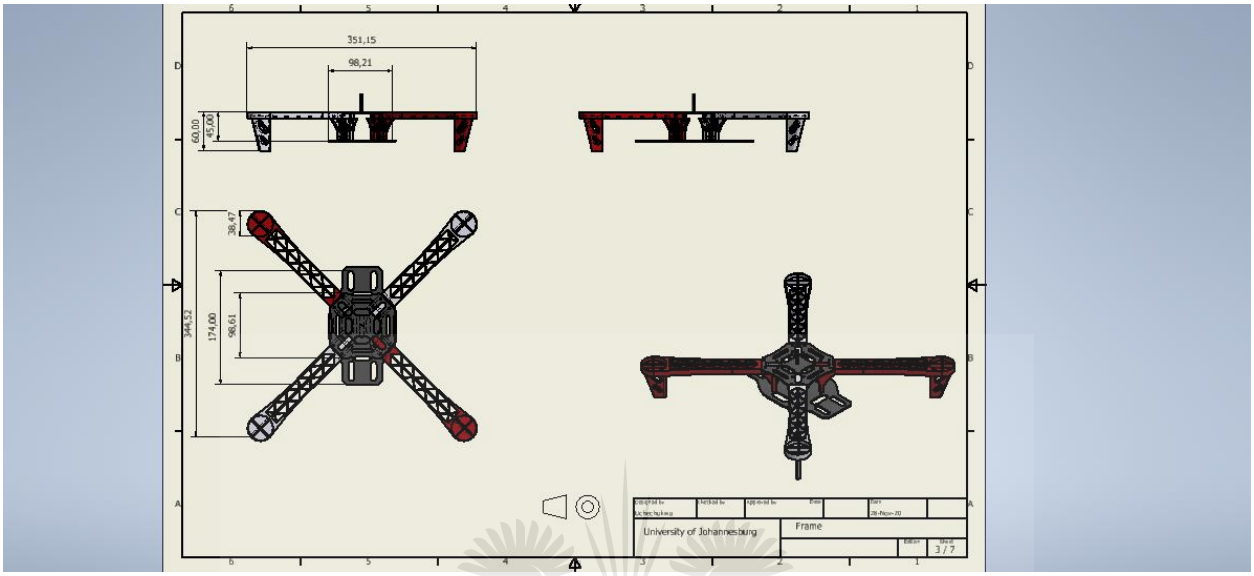
technique : A review', *Meat Science*. Elsevier, 156(June), pp. 183–195. doi: 10.1016/j.meatsci.2019.06.002.

135. Tameni M. (2015). An Introduction to Intel Edision for IOT. Viewed on 4th March, 2020 from https://www.sitepoint.com/introduction-intel-edison-iot-developers/
136. Tang, J., et al (2017). 'Weed identification based on K-means feature learning combined with convolutional neural network'. *Computers and electronics in agriculture*, *135*, pp.63-70.
137. Tang, T. A. *et al.* (2018) 'Deep Recurrent Neural Network for Intrusion Detection in SDN-based Networks', *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*. IEEE, (NetSoft), pp. 202–206. doi: 10.1109/NETSOFT.2018.8460090.
138. Taylor, B. et al. (2018) 'Adaptive Deep Learning Model Selection on Embedded Systems', *Proceedings of 19th ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES'18). ACM, New York, USA,* pp. 31 43. https://doi.org/10.1145/3211332.3211336
139. Teerapittayanon, S., Mcdanel, B. and Kung, H. . (2017) 'Distributed Deep Neural Networks over the Cloud , the Edge and End Devices', *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, pp. 328–339. doi: 10.1109/ICDCS.2017.226.
140. Ukaegbu U.F, Tartibu L.K and Okwu M.O.(2020). 'Deep Learning Hardware Accelerators for High Performance in Smart Agricultural Systems: An Overview'. Proceedings of the 31st Annual South African Institute for Industrial Engineering Conference, South Africa Volume: ISSN: 2308-8265,pp(558-570).
141. Ukaegbu, U.F, et al (2020). 'A deep learning algorithm for detection of potassium deficiency in a red grapevine and spraying actuation using a raspberry pi3'. In *2020 International Conference on Artificial Intelligence, Big Data, Computing and Data* doi: 10.1109/icABCD49160.2020.9183810.IEEE.
142. Vikhram, G.R., et al (2018). Automatic weed detection and smart herbicide sprayer robot. *International Journal of Engineering & Technology*, *7*(3.6), pp.115-118.
143. Véstias, M.P. et al (2020). 'Moving Deep Learning to the Edge'. *Algorithms*, *13*(5), p.125.

144. Vogeltanz, T. (2016) 'A Survey of Free Software for the Design , Analysis , Modelling ,and Simulation of an Unmanned Aerial Vehicle', pp. 449–514. doi: 10.1007/s11831 015-9147-y.
145. Wang, F. *et al*.(2019) 'A Survey on Deep Learning Empowered IoT Applications', *IEEE Access*. IEEE, 7, pp. 181721–181732. doi: 10.1109/ACCESS.2019.2958962.
146. Yanushevsky R. (2011). *Guidance of Unmanned aerial vehicles*. CRC Press, Taylor and Francis Group, Boca Raton Florida. ISBN 9781439850954.
147. Young, T., (2018). 'Recent trends in deep learning based natural language processing'. *IEEE Computational intelligence magazine*, *13*(3), pp.55-75.
148. Yu, D., et al (2014). *An introduction to computational networks and the computational network toolkit*. Microsoft Technical Report MSR-TR-2014–112.
149. Zamini, M. and Montazer, G., (2018).'Credit card fraud detection using autoencoder based clustering'. In *2018 9th International Symposium on Telecommunications (IST)* (pp. 486-491). IEEE.
150. Zhang Y.et al (2017). Design and test of a six-rotor unmanned aerial vehicle (UAV) electrostatic spraying system for crop protection. *International Journal of Agricultural and Biological Engineering* 10(6), pp.68-76.
151. Zhang, L., Wang, S. and Liu, B., (2018). 'Deep learning for sentiment analysis: A survey'. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *8*(4), p.e1253.
152. Zhang, X. *et al.* (2014) 'A Survey of Modelling and Identification of Quadrotor Robot', *Abstract and Applied Analysis.* Hindawi Publishing Corporation. pp. 1-16. http://dx.doi.org/10.1155/2014/320526
153. Zhang, Y., Liu, F. and Han, J. (2015) 'Amn-Pso Method For Jamming Unmanned Aerial Vehicle Network', 8(4), pp. 2042–2064.
154. Zhao, S. *et al.* (2018) 'Curved Path Following Control for Fixed-wing Unmanned Aerial Vehicles with Control Constraint', pp. 107–119. doi: 10.1007/s10846-017-0472-2.

Appendix A: Engineering drawing of selected components of the smart herbicide sprayer

A.1. Engineering drawings of the Frame and Propeller



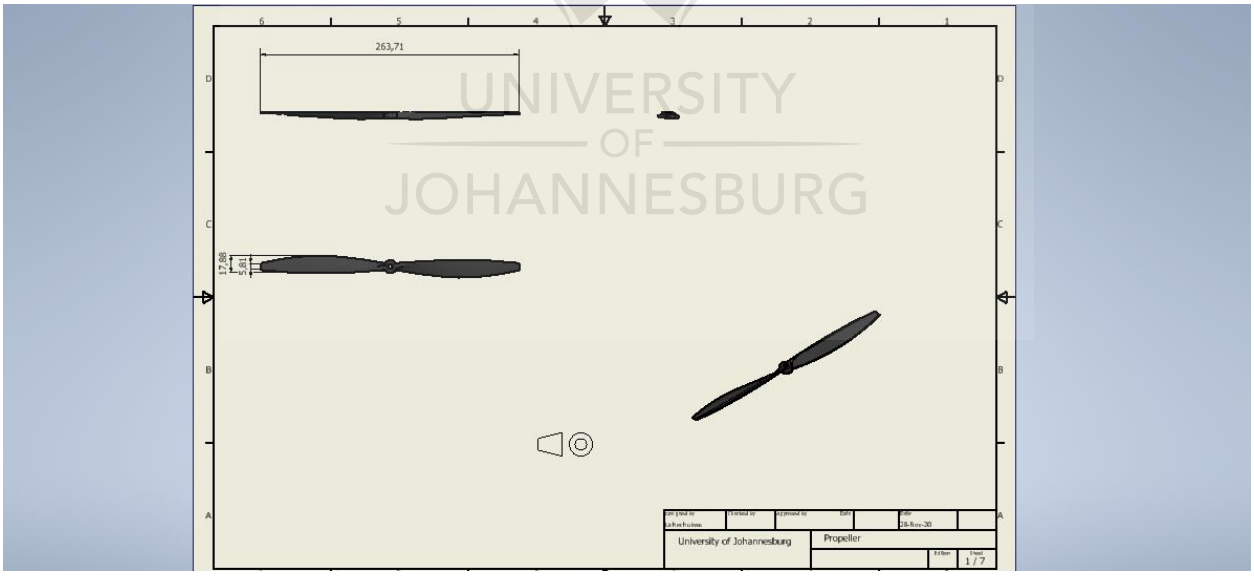


Fig A.1: Engineering drawings of the frame propeller

A.2. Engineering drawings of the electric motor and electronic speed controller

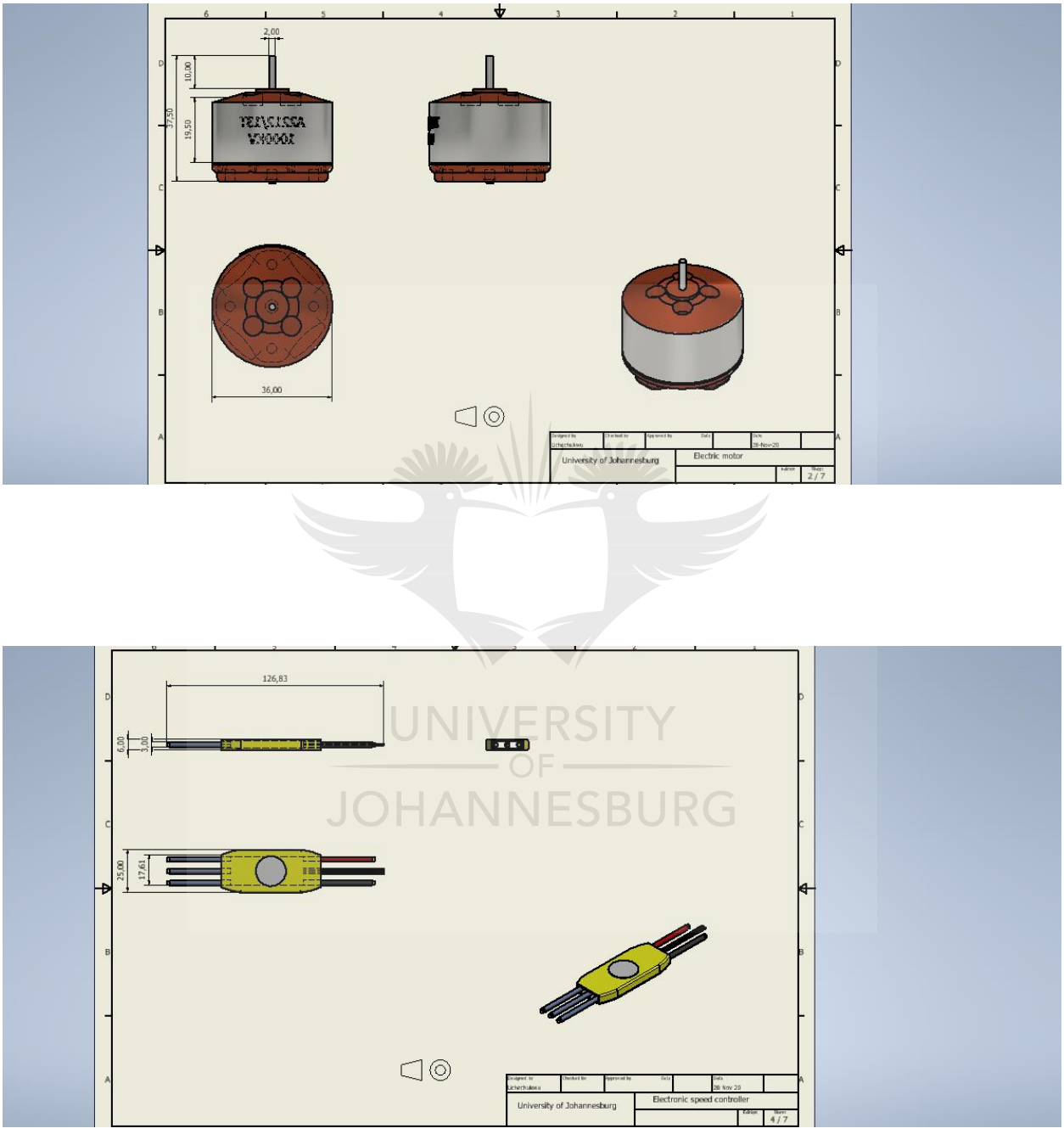


Fig A.2: Engineering drawings of the electric motor and electronic speed controller

A.3. Engineering drawings of the spray pump and nozzle

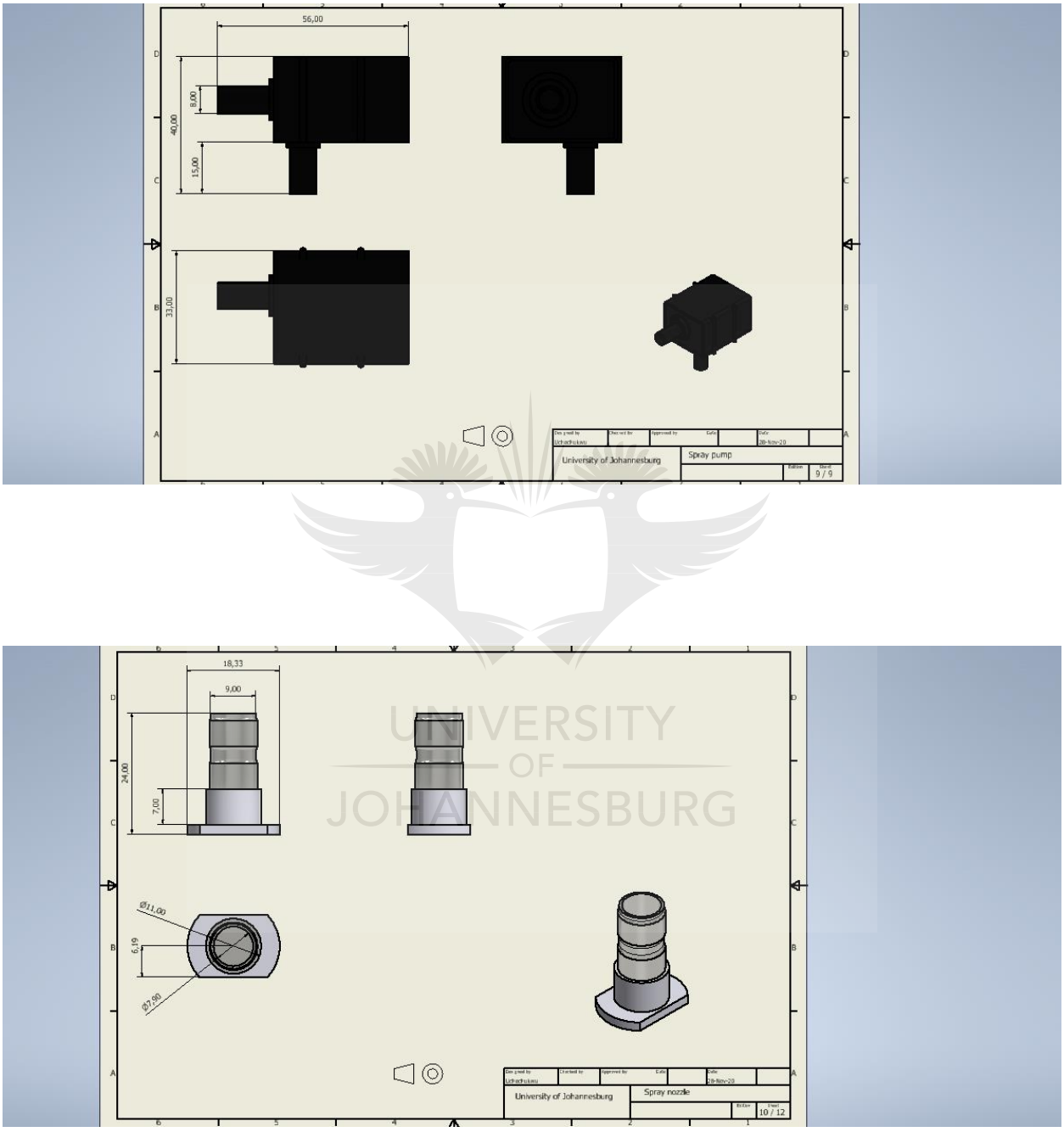


Fig A.3: Engineering drawings of the spray pump and nozzle

A.4. Engineering drawings of the raspberry pi and relay module

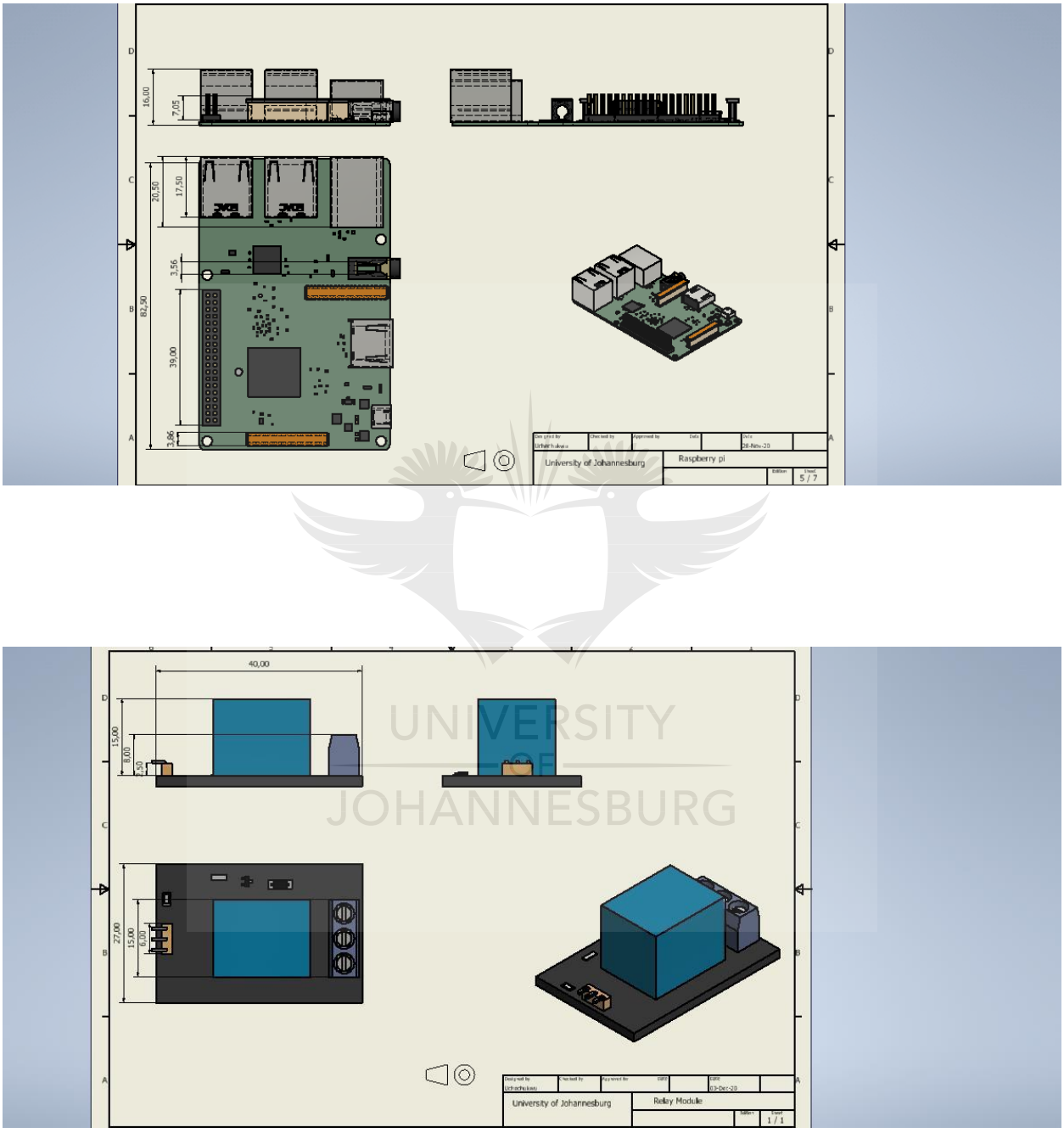


Fig A.4: Engineering drawings of the raspberry pi and relay module

A.5. Engineering drawings of the transmitter and receiver

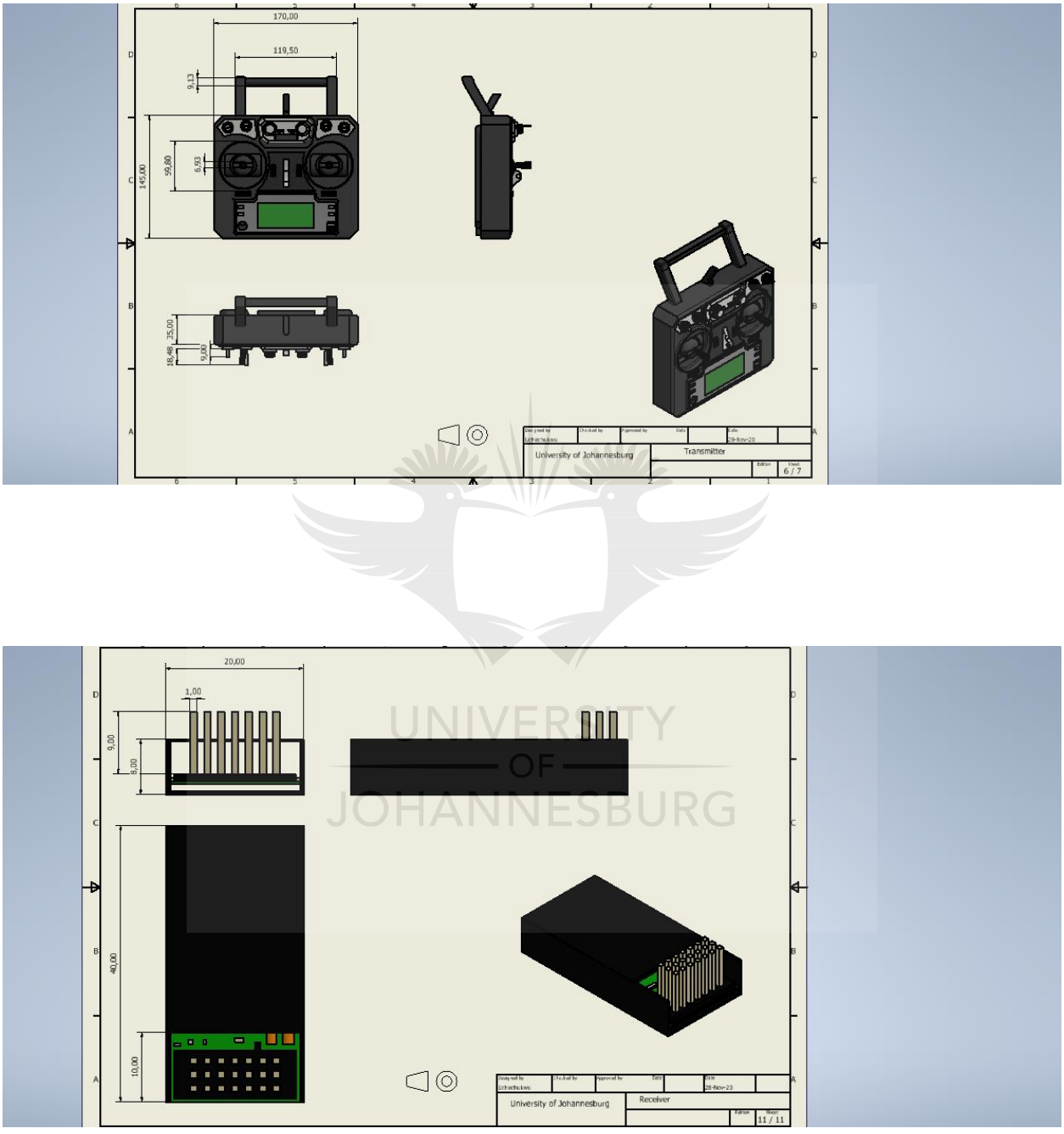
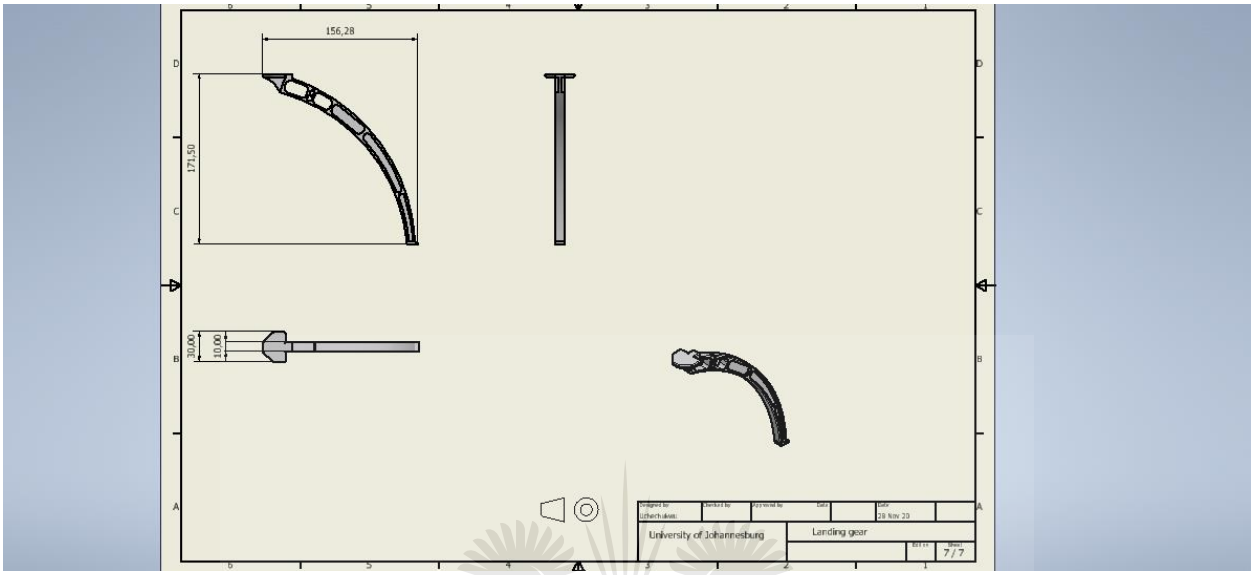


Fig A.5: Engineering drawings of the transmitter and receiver

A.6. Engineering drawings of the landing gear and prototype



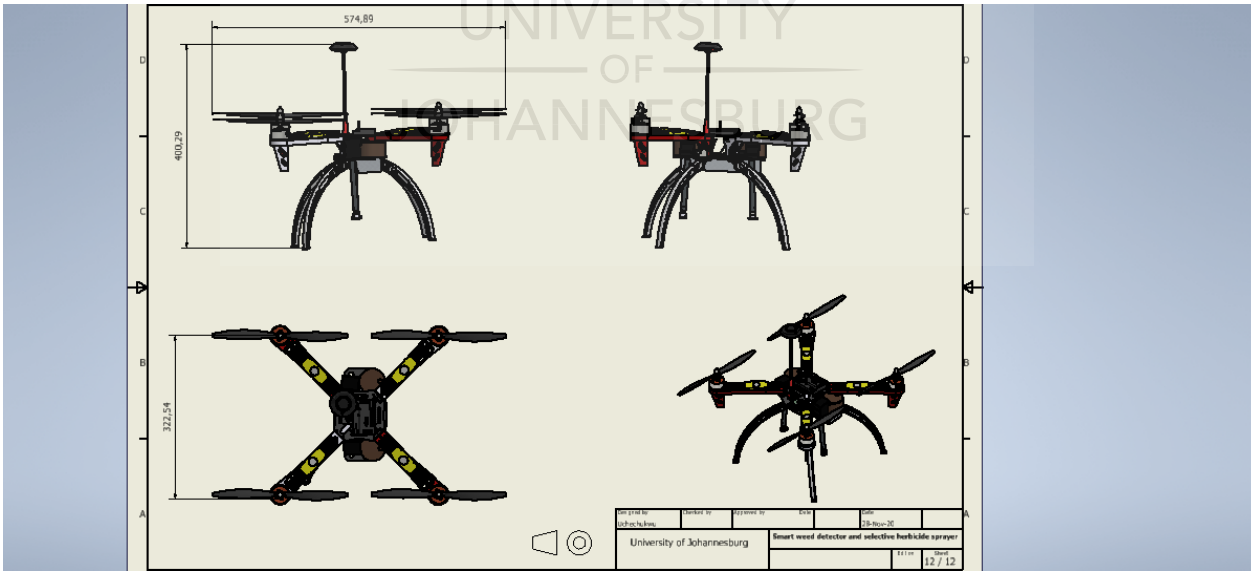
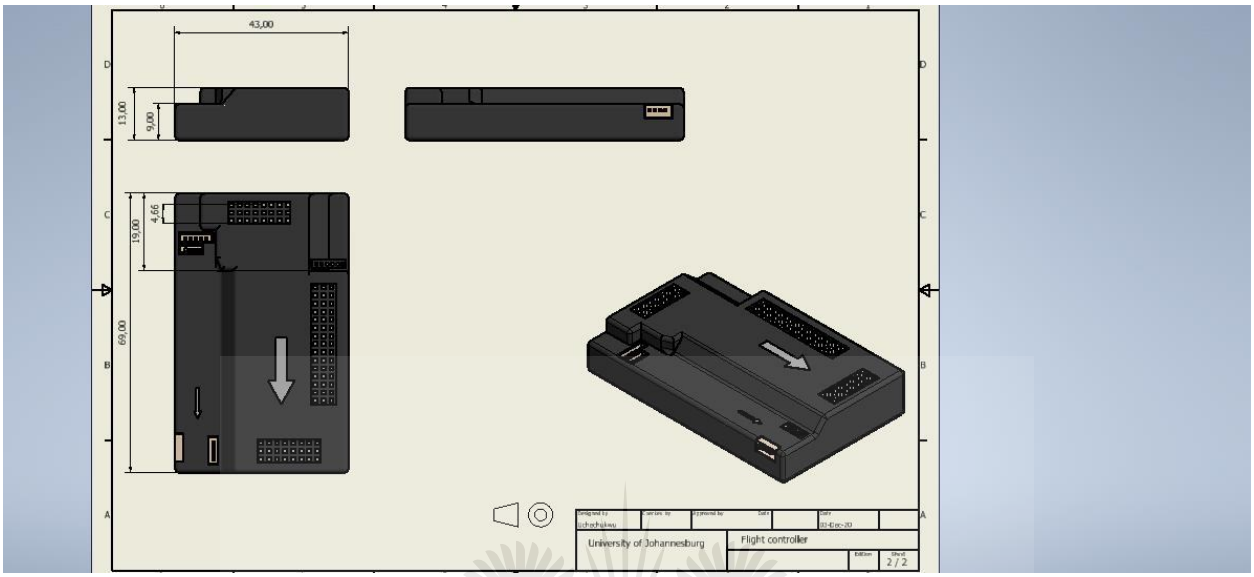


Fig A.6: Engineering drawings of the landing gear and prototype

A.7. Engineering drawing of the flight controller and Exploded view/part list of the prototype



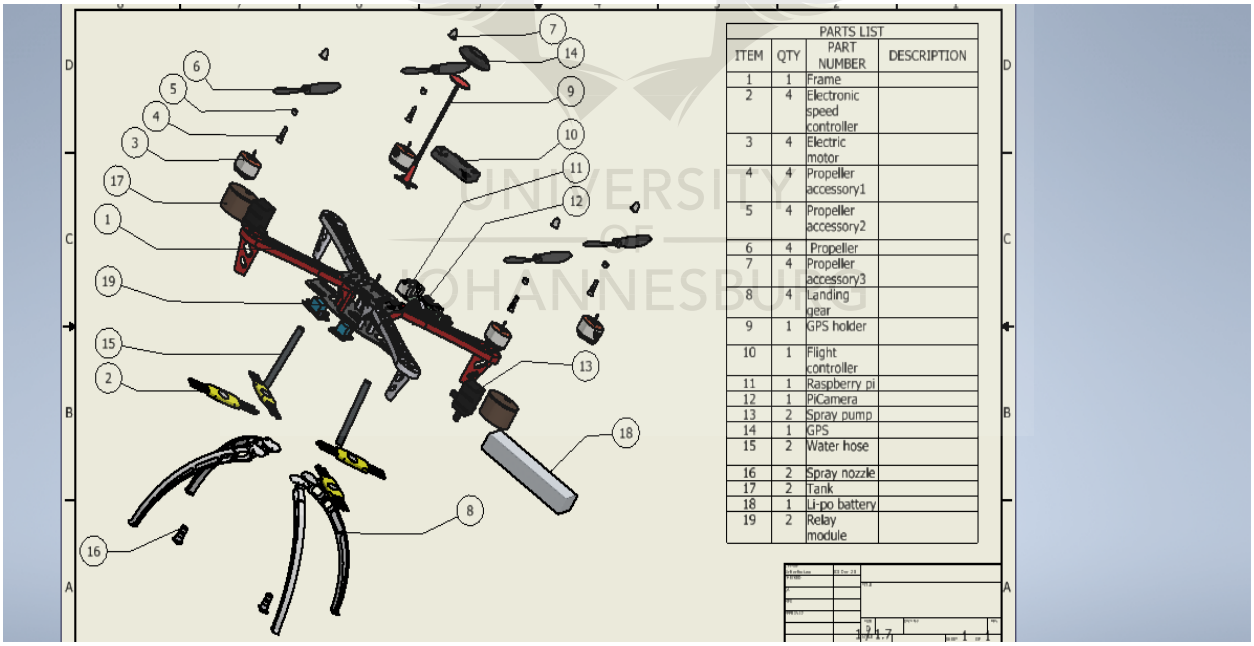


Fig A.7: Engineering drawing of the flight controller and Exploded view/part list of the prototype.

Appendix B: Python programming codes

B.1. Algorithms for training the RF classifier (Page1)

```
In [4]: from sklearn.preprocessing import LabelEncoder
        from sklearn.preprocessing import MinMaxScaler
        import numpy as np
        import mahotas
        import cv2
        import os
        import h5py
        import glob
        import matplotlib.pyplot as plt
        from sklearn.model_selection import train_test_split, cross_val_score
        from sklearn.ensemble import RandomForestClassifier
```

```
In [10]: # make a fix file size
         fixed_size  = tuple((224,224))

         #train path
         train_path = "C:/Users/ACTION/RF classfier/Train_d"

         # no of trees for Random Forests
         num_tree = 50

         # bins for histograms
         bins = 8

         # train_test_split size
         test_size = 0.30

         # seed for reproducing same result
         seed = 9
```

```
In [11]: # features description -1:  Hu Moments

         def fd_hu_moments(image):
             image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
             feature = cv2.HuMoments(cv2.moments(image)).flatten()
             return feature
```

```
In [12]: # feature-descriptor -2 Haralick Texture

         def fd_haralick(image):
             # conver the image to grayscale
             gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
             # Ccompute the haralick texture fetature ve tor
             haralic = mahotas.features.haralick(gray).mean(axis=0)
             return haralic
```

```
In [13]: # feature-description -3 Color Histogram

         def fd_histogram(image, mask=None):
             # conver the image to HSV colors-space
             image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
             #COPUTE THE COLOR HISTPGRAM
             hist  = cv2.calcHist([image],[0,1,2],None,[bins,bins,bins], [0, 256, 0, 256, 0, 256])
             # normalize the histogram
             cv2.normalize(hist,hist)
             # return the histog....
             return hist.flatten()
```


Fig B.1: Algorithms for training the RF classifier (Page1)

B.2. Algorithms for training the RF classifier (Page2)

```
In [14]: # get the training data labels
         train_labels = os.listdir(train_path)

         # sort the training labesl
         train_labels.sort()
         print(train_labels)

         # empty list to hold feature vectors and labels
         global_features = []
         labels = []

         i, j = 0, 0
         k = 0

         # num of images per class
         #images_per_class = 80
```

```
['Broadleaf', 'Grass', 'Soil', 'Soybean']
```

```
In [15]: # iterate the folder to get the image label name

         %time
         # loop over the training data sub folder

         for training_name in train_labels:
             # join the training data path and each species training folder
             dir = os.path.join(train_path, training_name)

             # get the current training label
             current_label = training_name
```



```
    # get the current training label
    current_label = training_name

    k = 1
    # loop over the images in each sub-folder

    for file in os.listdir(dir):

        file = dir + "/" + os.fsdecode(file)

        # read the image and resize it to a fixed-size
        image = cv2.imread(file)

        if image is not None:
            image = cv2.resize(image,fixed_size)
            fv_hu_moments = fd_hu_moments(image)
            fv_haralick   = fd_haralick(image)
            fv_histogram  = fd_histogram(image)
        #else:
            #print("image not loaded")

        #image = cv2.imread(file)
        #image = cv2.resize(image,fixed_size)

        # Concatenate global features
        global_feature = np.hstack([fv_histogram, fv_haralick, fv_hu_moments])

        # update the list of labels and feature vectors
        labels.append(current_label)
        global_features.append(global_feature)
```

Fig B.2: Algorithms for training the RF classifier (Page2)

B.3. Algorithms for training the RF classifier (Page3)

```
        i += 1
        k += 1
    print("[STATUS] processed folder: {}".format(current_label))
    j += 1

print("[STATUS] completed Global Feature Extraction...")
```

```
Wall time: 0 ns
[STATUS] processed folder: Broadleaf
[STATUS] processed folder: Grass
[STATUS] processed folder: Soil
[STATUS] processed folder: Soybean
[STATUS] completed Global Feature Extraction...
```

In [16]:
```
%time
# get the overall feature vector size
print("[STATUS] feature vector size {}".format(np.array(global_features).shape))

# get the overall training label size
print("[STATUS] training Labels {}".format(np.array(labels).shape))

# encode the target labels
targetNames = np.unique(labels)
le = LabelEncoder()
target = le.fit_transform(labels)
print("[STATUS] training labels encoded...{}")
# normalize the feature vector in the range (0-1)
scaler = MinMaxScaler(feature_range=(0, 1))
rescaled_features = scaler.fit_transform(global_features)
print("[STATUS] feature vector normalized...")
```

```
print("[STATUS] target labels: {}".format(target))
print("[STATUS] target labels shape: {}".format(target.shape))
```

```
Wall time: 0 ns
[STATUS] feature vector size (6071, 532)
[STATUS] training Labels (6071,)
[STATUS] training labels encoded...{}
[STATUS] feature vector normalized...
[STATUS] target labels: [0 0 0 ... 3 3 3]
[STATUS] target labels shape: (6071,)
```

In [17]:
```
global_features=  np.array(rescaled_features)
global_labels= np.array(target)
# split the training and testing data
(trainDataGlobal, testDataGlobal, trainLabelsGlobal, testLabelsGlobal) = train_test_split(np.array(global_features),
                                                                                          np.array(global_labels),
                                                                                          test_size=test_size,
                                                                                          random_state=seed)
```

In [18]:
```
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
# create the model - Random Forests
clf  = RandomForestClassifier(n_estimators=100)

# fit the training data to the model
clf.fit(trainDataGlobal, trainLabelsGlobal)
```

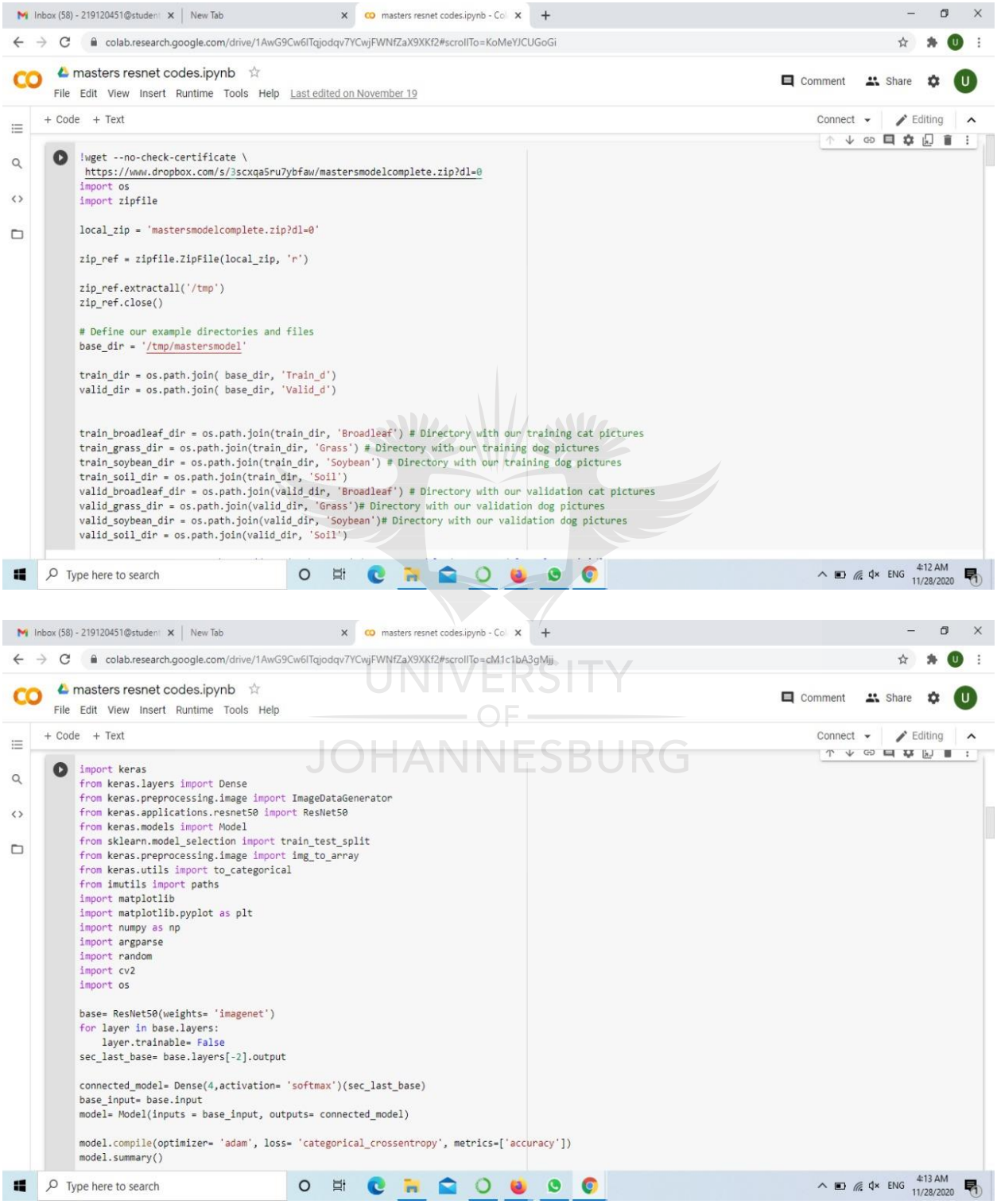
Fig B.3: Algorithms for training the RF classifier (Page3)

B.4. Algorithms for training the RF classifier (Page4)

```
In [18]: from sklearn.metrics import classification_report
         from sklearn.metrics import accuracy_score
         # create the model - Random Forests
         clf  = RandomForestClassifier(n_estimators=100)

         # fit the training data to the model
         clf.fit(trainDataGlobal, trainLabelsGlobal)

         #print(clf.fit(trainDataGlobal, trainLabelsGlobal))

         clf_pred = clf.predict(testDataGlobal)
         accuracy= accuracy_score(testLabelsGlobal, clf_pred)
         print('Model accuracy score is', accuracy)
         #testLabelsGlobal.shape
         #clf_pred.shape
         #trainDataGlobal.shape

         #clf_pred = clf.predict(global_feature.reshape(1,-1))[0]
         print(classification_report(testLabelsGlobal,clf_pred))

         Model accuracy score is 0.9533479692645445
                       precision    recall  f1-score   support

                    0       0.92      0.89      0.90       332
                    1       0.91      0.94      0.93       484
                    2       1.00      1.00      1.00       487
                    3       0.97      0.97      0.97       519

             accuracy                           0.95      1822
            macro avg       0.95      0.95      0.95      1822
         weighted avg       0.95      0.95      0.95      1822
```

Fig B.4: Algorithms for training the RF classifier (Page4)

B.5. Algorithms for training the CNN model (Page1)



```
!wget --no-check-certificate \
 https://www.dropbox.com/s/3scxqa5ru7ybfaw/mastersmodelcomplete.zip?dl=0
import os
import zipfile

local_zip = 'mastersmodelcomplete.zip?dl=0'

zip_ref = zipfile.ZipFile(local_zip, 'r')

zip_ref.extractall('/tmp')
zip_ref.close()

# Define our example directories and files
base_dir = '/tmp/mastersmodel'

train_dir = os.path.join( base_dir, 'Train_d')
valid_dir = os.path.join( base_dir, 'Valid_d')


train_broadleaf_dir = os.path.join(train_dir, 'Broadleaf') # Directory with our training cat pictures
train_grass_dir = os.path.join(train_dir, 'Grass') # Directory with our training dog pictures
train_soybean_dir = os.path.join(train_dir, 'Soybean') # Directory with our training dog pictures
train_soil_dir = os.path.join(train_dir, 'Soil')
valid_broadleaf_dir = os.path.join(valid_dir, 'Broadleaf') # Directory with our validation cat pictures
valid_grass_dir = os.path.join(valid_dir, 'Grass')# Directory with our validation dog pictures
valid_soybean_dir = os.path.join(valid_dir, 'Soybean')# Directory with our validation dog pictures
valid_soil_dir = os.path.join(valid_dir, 'Soil')
```



```
import keras
from keras.layers import Dense
from keras.preprocessing.image import ImageDataGenerator
from keras.applications.resnet50 import ResNet50
from keras.models import Model
from sklearn.model_selection import train_test_split
from keras.preprocessing.image import img_to_array
from keras.utils import to_categorical
from imutils import paths
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import argparse
import random
import cv2
import os

base= ResNet50(weights= 'imagenet')
for layer in base.layers:
    layer.trainable= False
sec_last_base= base.layers[-2].output

connected_model= Dense(4,activation= 'softmax')(sec_last_base)
base_input= base.input
model= Model(inputs = base_input, outputs= connected_model)

model.compile(optimizer= 'adam', loss= 'categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

Fig B.5: Algorithms for training the CNN model (Page1)

B.6. Algorithms for training the CNN model (Page2)

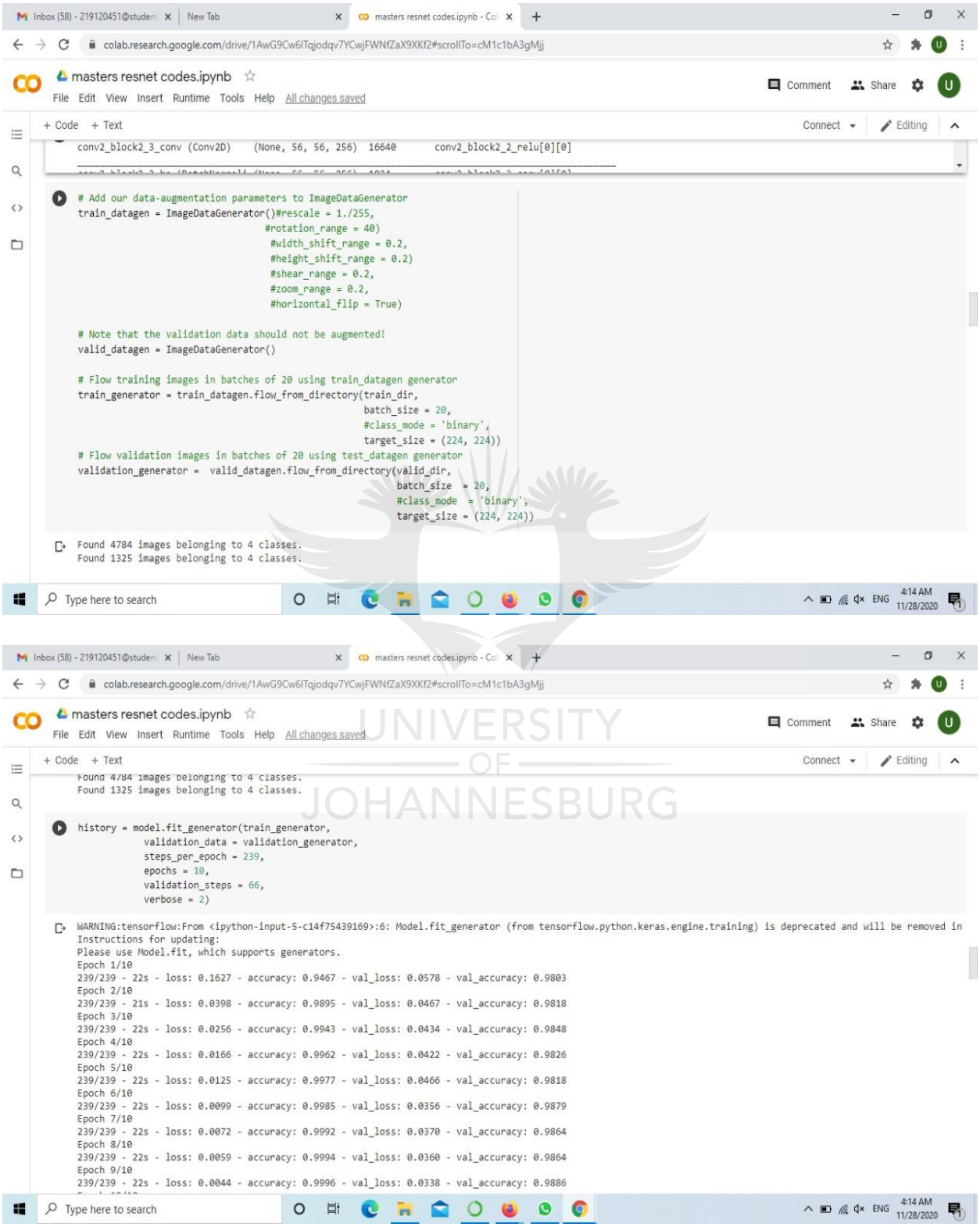


Fig B.6: Algorithms for training the CNN model (Page2)

B.7. Algorithms for conversion to tensorflow lite format

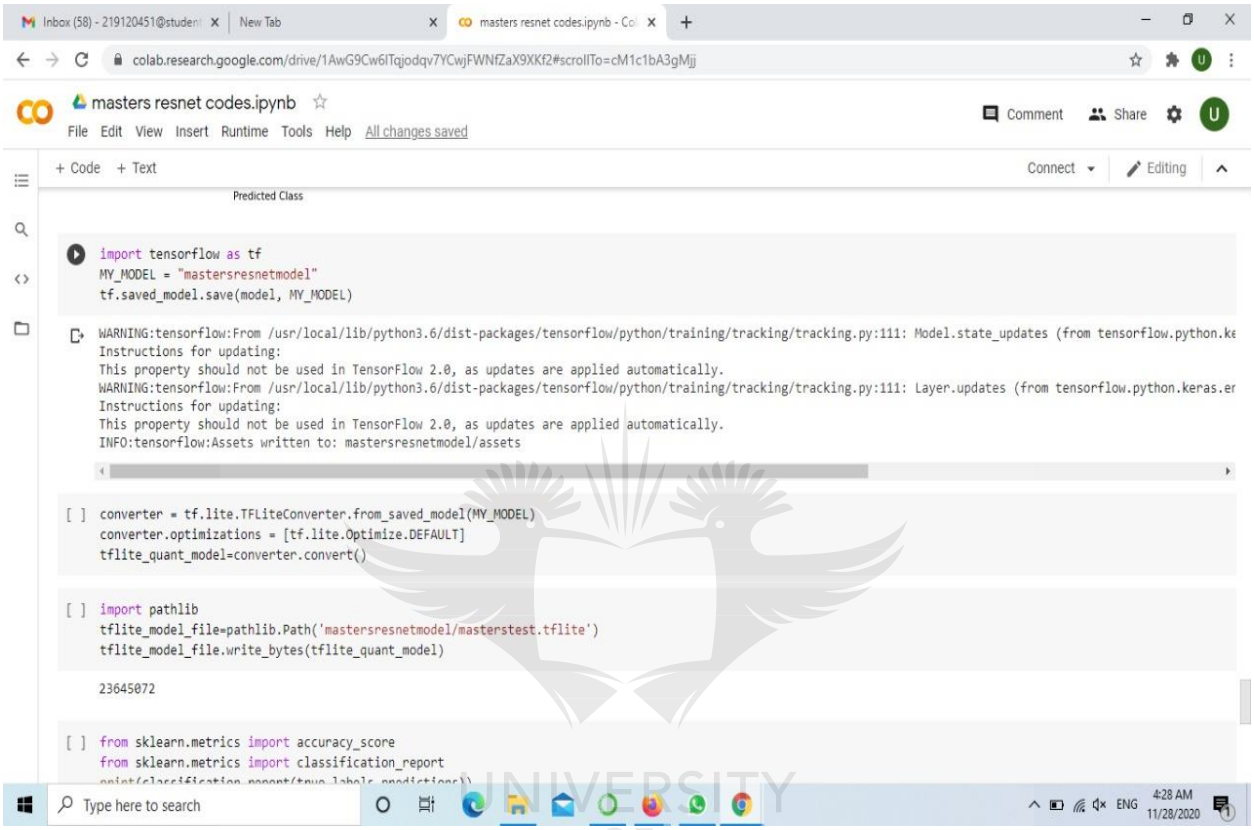


Fig B.7: Algorithms for conversion to tensorflow lite format

B.8. Algorithms for weed detection and spraying actuation (page1)

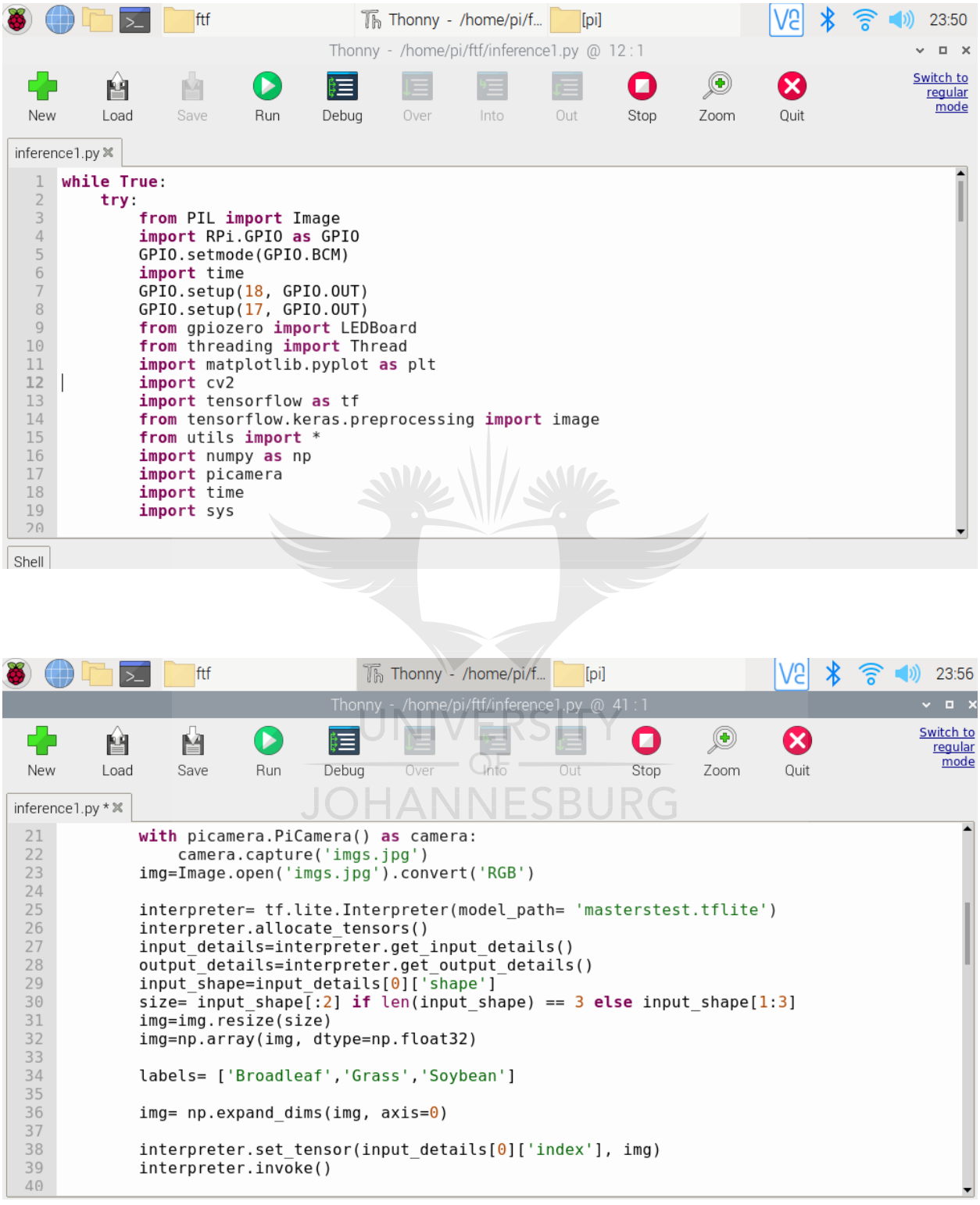


Fig B.8: Algorithms for weed detection and spraying actuation (page1)

B.9.Algorithms for weed detection and spraying actuation (page2)

Fig B.9: Algorithms for weed detection and spraying actuation (page2)

Appendix C: Cost analysis

C.1. Cost analysis

S/N	COMPONENTS PURCHASED	COST(IN RANDS)
1	Quadcopter Kit	3120
2	ZOP power 5500MAH 11.1V li-poly battery	2250
3	Balance smart battery charger	675
4	Raspberry pi 3B kit	1400
5	Raspberry pi camera V2	582
6	CMU Flysky FS-I6 6 channel transmitter/receiver set	1451
7	Single / 1 channel 5VDC 10A relay module development board	61
8	BMT mini subm. water pump	153
9	Anker PowerCore 13000MAH power bank	805
10	Universal tail landing gear skid for DJI F450	236
11	Pressure washer spray nozzle	311
12	Pixnor APM 2.6 MWC GPS compass antenna folding fixed mount bracket	214
13	Mirthhobby RC anti-vibration plate	328
14	Battery holder 8×AA with leads black	21
15	Battery connector male plug	19
16	AA size batter 1.5V alkaline 24pieces/pack	120
	Total	**11 746**

Table C.1: Cost analysis