UNIVERSITY OF RWANDA

COLLEGE OF SCIENCE AND TECHNOLOGY

AFRICAN CENTRE OF EXCELLENCE IN INTERNET OF THINGS

KIGALI - RWANDA

HYBRID DEEP LEARNING MODELS FOR EARLY DETECTION OF FALL ARMYWORMS IN MAIZE FIELDS USING UNMANNED AERIAL VEHICLE-BASED SENSOR DATA: A CASE STUDY OF MAIZE FIELDS IN EAST AFRICA

PhD Thesis Submitted in The Fulfilment of Requirements of Award of The Degree of

Doctor of Philosophy in Internet of Things – Wireless Intelligent Sensor Network

By

Farian Severine Ishengoma

(219011988)

Farian S. Ishengoma

This page has been purposefully left blank.

UNIVERSITY OF RWANDA

COLLEGE OF SCIENCE AND TECHNOLOGY

AFRICAN CENTRE OF EXCELLENCE IN INTERNET OF THINGS

KIGALI - RWANDA

HYBRID DEEP LEARNING MODELS FOR EARLY DETECTION OF FALL ARMYWORMS IN MAIZE FIELDS USING UNMANNED AERIAL VEHICLE-BASED SENSOR DATA: A CASE STUDY OF MAIZE FIELDS IN EAST AFRICA

PhD Thesis submitted in the fulfilment of requirements of award of the Degree of

Doctor of Philosophy in Internet of Things – Wireless Intelligent Sensor Network

By

Farian Severine Ishengoma

(219011988)

Supervisor: Prof. Idris A. Rai

Co-Supervisor: Dr. Ignace Gatare

July 2023

Farian S. Ishengoma

This page has been purposefully left blank.

Declaration

I hereby declare that the dissertation entitled *"Hybrid deep learning models for early detection of fall armyworms in maize fields using unmanned aerial vehicle-based sensor data: A case study of maize fields in East Africa"* to be submitted for the Degree of Doctor of Philosophy is my original work and the dissertation has not formed the basis for the award of any degree, diploma, associateship, or fellowship of similar other titles. It has not been submitted to any other University or Institution for the award of any degree or diploma.

Farian S. Ishengoma

Signature. .. 

Date: 10 July 2023

Farian S. Ishengoma

This page has been purposefully left blank.

Farian S. Ishengoma, a Ph.D. student of UR-ACEIoT student ID 219011988, successfully defended the thesis entitled "*Hybrid deep learning models for early detection of fall armyworms in maize fields using unmanned aerial vehicle-based sensor data: A case study of maize fields in East Africa*", which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Supervisor: **Signature**

Prof. Idris A. Rai

State University of Zanzibar, Tanzania ……

Co-Supervisor:

Dr. Ignace Gatare

University of Rwanda, Rwanda ……

Viva Voce Members:

Chairperson:

Prof. Umaru Garba Wali ……

University of Rwanda, Rwanda

External members:

Prof. Udoinyang Godwin Inyang ……

University of Uyo, Nigeria

Prof. Jules Degila

University of Abomey-Calavi, Benin ……

Dr. Enan Nyesheja ……

University of Lay Adventists of Kigali, Rwanda

Date of Defense: 10/07/2023

Farian S. Ishengoma

This page has been purposefully left blank.

Dedication

I would like to express my heartfelt gratitude and extend my sincerest appreciation to the following individuals who have played an integral role in my academic journey and the completion of this thesis:

First and foremost, I am profoundly grateful to my late parents, *Ishengoma Severine and Prisca Paul*, for instilling in me the values of hard work, perseverance, and an insatiable thirst for knowledge. Your unwavering love and support have been my guiding light, and I am forever grateful for everything you have done for me.

To my beloved wife, *Priska S. Lohay*, you have been my rock, confidante, and greatest cheerleader. Your love, patience, and understanding have been essential in keeping me motivated and focused on my goals. I could not have accomplished this without you by my side.

To my incredible children, *Atugonza Glory*, *Byera*, and *Twesige Faithlyn*, you have been a constant source of inspiration and motivation. Your innocent joy, curiosity, and boundless energy have reminded me of the profound importance of education and the pursuit of knowledge. I dedicate this thesis to you and hope it will inspire you to fearlessly chase your dreams and aspirations.

Lastly, I would like to dedicate a special note of appreciation to my dear family members, *Desderius, Wares, Salvinar, Proscovia, Severusy, Edson, Egbert, Sospeter and others*, for your unwavering support and encouragement. Your words of wisdom, moral support, and practical assistance have been invaluable and deeply appreciated.

And to anyone currently facing challenges in pursuing their academic goals, I dedicate this thesis to you. May it serve as a reminder that with dedication, determination, and the unwavering support of loved ones, anything is possible.

Thank you all for being an integral part of this incredible journey. Your contributions have made a significant impact, and I am forever grateful.

Farian S. Ishengoma

This page has been purposefully left blank.

Acknowledgement

I would like to extend my heartfelt appreciation and sincere gratitude to my esteemed supervisors *Prof. Idris A. Rai, Dr. Said R. Ngoga, and Dr. Ignace Gatare* for their exceptional guidance, unwavering support, and invaluable mentorship throughout my thesis project.

Their extensive expertise, profound wisdom, and constructive feedback have shaped my research, improved my writing, and advanced my academic and professional growth. I am deeply grateful for their unwavering commitment to excellence, meticulous attention to detail, and high academic rigor, which have inspired me to strive for greatness and exceed my own expectations. Their dedication to intellectual excellence has taught me to think critically, be curious, and value knowledge. They were especially kind, patient, and understanding during the thesis's hardest parts. Even when faced with overwhelming challenges or self-doubt, their timely and insightful advice, words of encouragement, and unwavering motivation have kept me motivated and determined.

Furthermore, I would also like to acknowledge the significant contributions of my supervisors to my personal and professional development beyond the scope of the thesis. Their mentorship has instilled within me a lifelong passion for learning, a deep reverence for the academic community, and an unwavering commitment to excellence in all spheres of my life.

I would also like to thank the ACEIoT for allowing me to be a member of the excellent student cohort II and for providing me with a full scholarship for this exceptional study program. Furthermore, I want to thank the ACEIoT Centre's director, *Prof. Damien Hanyurwimfura*, the head of PhD students, *Dr. Omar Gatera*, and other staff members for their encouragement, support, and assistance.

I would also like to express my deepest gratitude to my mentors, professors, and advisors, whose guidance, expertise, and encouragement have played a pivotal role in shaping my academic and professional career. Your invaluable insights, critiques, and feedback have challenged me to think critically and push the boundaries of my abilities. To my esteemed colleagues, *Prof. Camilius A. Sanga, Dr. Sospeter C. Jibunge, Dr. Douglas Mushi, Hussein Mkwazu*, the laboratory companions *Dr. Omar Kombo, George Odong, Emmanuel Lule, Dr. Gibson Kimutai, Caroline Katushabe, Irene Mihigo, Elias Ntawuzumunsi*, I extend my heartfelt appreciation. Your support in various forms throughout this journey has been instrumental in keeping me motivated and focused.

Lastly, I would like to extend my appreciation to Sokoine University of Agriculture (SUA) for their continuous support and granting me permission to pursue the PhD studies program.

Abstract

Maize is the main food crop grown in Sub-Saharan Africa and is consumed by people with varying food preferences. Maize production is continuously and severely affected by several threats, such as weeds, insects, bacteria, viruses, nematodes, fungi, drought, and climate change. In the year 2016 fall armyworms (faw) invaded the crops in Africa causing severe damage. The worm grows very fast and destroys the maize crops in a short period. Different initiatives have been explored to raise farmer awareness, such as the introduction of mobile applications such as the Food and Agriculture Organisation Fall Armyworm Monitoring and Early Warning System (FAO-FAMEWS), Plant-Village-NURU and others. All the smartphone apps mentioned directly assist small-scale farmers in identifying and being aware of the faw. Moreover, they lack internet of things (IoT) and machine learning (ML) components that would allow them to cover a larger region and correct more data in a shorter period of time while analyzing it in real time.

In this study, we design a system to assist large-scale farmers in identifying the faw-caused pattern and sending a report to the farmers highlighting the size of the infection and the location for prompt action. In the proposed system we integrated unmanned aerial vehicle (UAV) techniques and ML algorithms to detect early indicators of faw in big farms and swiftly analyze the acquired data. The UAV, outfitted with IoT sensors and ML algorithms, allows capturing, analyzing, and immediately notifying the farmer of infected areas for further action.

The study is divided into three stages. First, the captured UAV images are cropped to 150x150 pixels, and the Shi-Tomas corner detection algorithm is applied to the cropped images. These images are then trained on convolution neural network (CNN) based models, specifically the VGG16, VGG19, InceptionV3 and MobilenetV2 to distinguish between healthy and infected images. The results show that this approach outperforms other CNN-based models in terms of accuracy, sensitivity, specificity, precision, and F1-score. Both InceptionV3 and MobileNetV2 outperformed other models by achieving an accuracy of 100%, a sensitivity of 1.00, a specificity of 1.00, a precision of 1.00, and an F1-score of 1.00. Despite its ability to achieve the highest precision and sensitivity in a real-time classification environment, the proposed algorithm can introduce noise to background images, leading to misclassification.

The second part of the study entails developing an autonomous system that classifies the UAV images into four groups, namely healthy, infected, weed and redundant counting infected images and sending a report to a farmer. The system is

divided into three parts: data collecting using UAV, data processing and analysis with CNN-based models (Hybrid CNN, VGG16, InceptionV3, Resnet50, and XceptionNet), and reporting to end users via a short messaging service. Moreover, the proposed real-time monitoring system's accuracy and training time are enhanced by using the hybrid CNN (HCNN) model. As a result, the system achieved 96.98% accuracy and reduced the training time by 16% up to 44%, making it quicker at detecting infected maize plants.

The third stage entailed developing two algorithms that improve the system designed in the previous step by developing a model that enhances the contrast of images captured in other farms to correlate with those used for training the proposed model to improve accuracy; and developing an algorithm that counts infected images, pinpoints the location of infected images, and estimates the infected areas on each UAV image. The first algorithm reduces the percentage error of images captured from other farms from 17.1% of the health group to 0.68% and -57.82% of the infected group to 6.64%, while the second algorithm improves the system by pinpointing infected images and displaying the size of infection on each UAV image.

In conclusion, the proposed method improves faw detection in large-scale farms, saving time and money. It uses UAV technology to quickly scan fields and upload images to the cloud for analysis with CNN-based models. The system reports infected areas, allowing farmers to target only affected areas when spraying pesticides.

Key words

Digital agriculture, maize, fall armyworms, computer vision, image processing, unmanned aerial network, internet of things, machine learning, and convolution neural network.

Table of Contents

Farian S. Ishengoma

This page has been purposefully left blank.

List of Figures

Farian S. Ishengoma

This page has been purposefully left blank.

List of Tables

Farian S. Ishengoma

This page has been purposefully left blank.

List of Acronyms

Abbreviation	Definition
4G	Fourth generation mobile network
5G	Fifth generation mobile network
Ac	Accuracy
AF	Activation function
AI	Artificial intelligence
ANN	Artificial neural network
BPNN	Back-propagation neural networks
CLDQHE	Contrast limited dynamic quadri-histogram equalization
CNN	Convolution neural network
CPU	Central processing unit
CV	Computer vision
DA	Digital agriculture
DBN	Deep belief networks
DL	Deep learning
DNN	Deep neural network
Faw	Fall armyworms
FC	Full connected layers
FN	false negative
FP	False positive
GAN	Generative adversarial networks
Gap	Global average pooling
GDP	Gross domestic product
GLCM	Grayscale co-occurrence matrix
GMMCE	Gaussian mixture model-based contrast enhancement
GPS	Global positioning system
GPSc	Global positioning system coordinates
GPU	Graphics processing unit
GSD	Ground sample distance
HCNN	Hybrid convolution neural network
IoT	Internet of things
KNN	K-nearest neighbor
LReLU	Leaky rectified liner unit
LSTM	Long short-term memory
MAF	Multi-activation function
MCNN	Multichannel convolutional neural network
ML	Machine learning
MLP	Multilayer perceptron
PCA	Principal component analysis
PR	Pattern recognition
Pr	Precision
PReLU	Parametric rectified liner unit
QMC	Quality metric
RBFN	Recurrent belief propagation networks
RBM	Restricted Boltzmann machine
ReLU	Rectified liner unit
RF	Random forest
RGB	Red, green, and blue
RICE	Robust image contrast enhancement
RNN	Recurrent neural networks

Sconv	Separable convolution
SGD	Stochastic gradient descent
SGD	Stochastic gradient descent
SMS	Short message service
SOMS	Self-organizing maps
SSA	Sub Saharan Africa
SVM	Support vector machine
TL	Transfer learning
TN	True negative
TP	True positive
UAV	Unmanned aerial network
VGG	Visual geometry group
VI	Vegetation index
YOLO	You only look once

Chapter 1

General Introduction

1.1. **Background**

The global population is projected to increase from 7.7 billion people in 2020 to 8.5 billion people in 2030, making it a challenge to provide sufficient food for the world's population. Most of the world's population increase may be attributed to emerging nations, most notably the continent of Sub-Saharan Africa (SSA), which is projected to see the highest annual growth rate of 2.5% [1]. Since 1990, the population of SSA has increased by 96%, which is significantly more than the average global growth rate of 38% [2]. To help farmers maximize their yield while decreasing their environmental impact, digital agriculture should be seriously considered. Botswana and South Africa's economies rely on agriculture for 3% of their Gross Domestic Product (GDP), but Ethiopia and Chad depend on it for 42% and 53%, respectively. The average contribution of agriculture to total GDP in SSA is 15%, although this number varies widely across the region. Around 80% of all farms in SSA are smallholder farms, and approximately 175 million people are directly employed on these farms. Most people in regions including SSA rely on cereals such as sorghum, millets, wheat, maize, and rice as their primary sources of nutrition [3].

In many nations across SSA, maize is the most important crop for food production and consumption [3]. Of the world's 22 countries, Africa is home to 16 nations that count maize as their primary source of nutrition. In eastern and southern Africa, maize accounts for over half of the calories and protein consumed. Still maize is responsible for one-fifth of the calories and protein ingested in West Africa. It is estimated that more than 33 million hectares of the total cultivated land in SSA, which totals 200 million hectares, is used for growing maize [3], [4]. More than 300 million people in SSA rely on maize as their primary source of nutrition and income [5], [6]. South Africa, Nigeria, Ethiopia, Tanzania, Malawi, Kenya, Zambia, Uganda, Ghana, Mozambique, Cameroon, Mali, Burkina Faso, Benin, Democratic Republic of Congo, Angola, Zimbabwe, Togo, and Cote d'Ivoire produce the most maize in SSA. This is 96% of SSA's maize harvest [3], [7].

Since 2016, the maize crop has been affected by various diseases and pests, but the fall armyworm (faw) has been the most damaging pest regarding economic losses [5]. Researchers believe faw first arrived in Africa by stowaways in cargo containers and

aeroplane holds, before being dispersed by the wind to other parts of the continent [5], [8]. Beginning in the West African tropics in 2016, faw eventually reached the South African Highveld and spread throughout at least 12 African countries (Togo, Ghana, Zambia, Zimbabwe, South Africa, Malawi, Mozambique, Namibia, Kenya, Rwanda, Uganda, and Tanzania). Damage caused by this armyworm in 2016 alone was estimated at 300,000 hectares in SSA [8]. A poll conducted in Ghana and Zambia in July 2017 anticipated a national mean maize loss of 45% and 40%, respectively [6].

1.2. **The Biological Concept of Faw**

The scientific name for faw is *Spodoptera frugiperda*. This moth feeds on many different types of plants and flowers. Researchers believe it first appeared in the Americas. A faw goes through four phases during its lifetime: *egg, larva, pupa,* and *adult* (see Figure 1.1). In addition to maize, rice, sorghum, and sugarcane, it also eats cabbage, beets, groundnuts, soybeans, onions, cotton, pasture grasses, millet, tomato, and potatoes, making it one of the most destructive agricultural pests.

Each adult moth may lay up to 2,000 eggs during its two to three weeks life cycle, faw can quickly and easily spread over large areas via prevailing winds [8], [9]. During its lifetime on a maize plant, the faw goes through the following four stages: Stage one (the egg) lasts for three to five days and involves the laying of 100 to 200 eggs at the base of the maize plant, at the point where the leaf meets the stem. Larvae (stage 2) develop for three to six days. The caterpillars' rapid feeding, especially at night on the undersides of leaves causes them to become partially transparent shortly after hatching. Older plants like the leaves surrounding the cob silks, while younger plants favor the spiral. The pupa (stage 3) lasts for between 7 and 14 days. When infected, leaves have holes as a result of faw penetrating the leaf's protective whorl. Over-fertilization of immature plants can kill the growth point, preventing the development of new leaves and cobs. After reaching maturity, the adult caterpillar (stage 4) spends between 14 to 23 days underground, digging a hole 2-8 cm deep. The caterpillar uses leaf litter to protect itself from the harsh environment if the soil is too dense for the pupation process. The adult moth later emerges after 8 or 9 days, and the cycle continues [9].

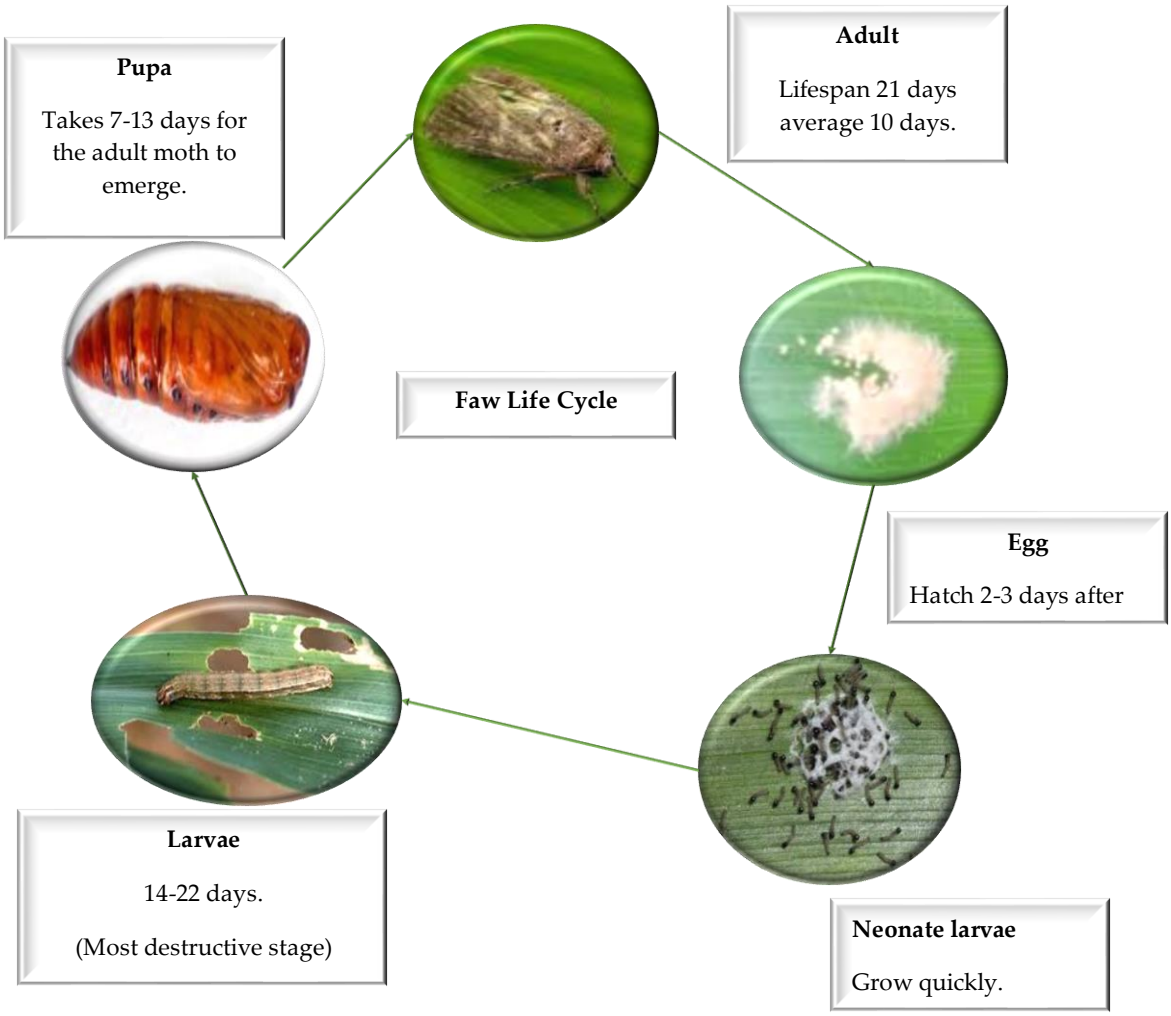


Figure 1.1: The life cycle of faw

1.3. **Problem Statement**

Presently, the process of detecting patterns caused by faw in large farms is a laborious task. Farmers are required to physically visit the farm and manually inspect small sections, relying either on capturing images and uploading them through a mobile application for classification or using their naked eyes if they are familiar with the patterns caused by faw [10].

Since 2016, several mobile applications have been developed for the detection and monitoring of faw symptoms and signs on various plants. Examples of such applications include Food and Agriculture Organisation Fall Armyworm Monitoring and Early Warning System (FAO-FAMEWS), Plant-Village-NURU, KALRO Fall Armyworm, and FALL ARMYWORM. For instance, FAO-FAMEWS serves as an

early warning system and provides general crop protection information. It operates by having farmers capture pictures using their smartphones, which are then uploaded to a server for classification when an internet connection becomes available. The application analyzes images and issues warnings to communities in their respective areas through local government channels. Moreover, Plant-Village-NURU and KALRO Fall Armyworm offer general information on preventing various pests and diseases in different crops. They also allow farmers to upload pictures to the database if they require assistance from experts. In addition, FALL ARMYWORM provides farmers with valuable awareness by illustrating the appearance, symptoms, and signs of faw, as well as offering guidance on scouting, monitoring, and managing the infestation.

In contrast, Kasinathan and Uyyala. proposed an approach for detecting faw insects using the Mask Region Convolutional Neural Network (Mask R-CNN) to ensure crop quality and safety. The performance of the Mask R-CNN detection model was compared to other methods, including Region CNN (R-CNN), Faster R-CNN, RetinaNet, and Single Shot Multi-Box Detector (SSD). The results of faw insect detection demonstrated a mean average precision of 94.21% using the Mask R-CNN model with the Resnet-101 backbone. This highlights the efficacy of the model in accurately segmenting faw insects in field crops [11].

Furthermore, Hanyurwimfura et al. introduced an automated system utilizing the Internet of Things (IoT) for monitoring and detecting faw in maize fields. The system incorporated Long Range (LoRa) communication technology for data transmission and Global Positioning System (GPS) for accurately identifying the location of the affected maize crops. Various sensors integrated within the system could assess the color and shape characteristics of the maize plants [12]. In addition, Sena et al. presented a method that utilizes digital images acquired by a digital camera to identify maize plants affected by faw damage. Their approach involved developing an algorithm capable of analyzing digital color images of maize plants under controlled lighting conditions to detect faw-inflicted damage. The proposed algorithm achieved a 94.72% accuracy in correctly labeling 720 instances [13]. The primary objective of all these methods is to assist small-scale farmers in detecting and preventing pest infestations. However, for medium and bigger farms, it is essential to explore low-cost alternative real-time scouting and reporting systems to avoid substantial losses and lower operating expenses.

The goal of this research is to assist farmers in reducing operational costs and time spent on farms identifying and treating faw by utilizing cutting-edge technologies such as unmanned aerial vehicles (UAV) outfitted with IoT sensors and machine-learning (ML) approaches. UAVs can quickly survey large regions using IoT-enabled sensors, and ML algorithms paired with computer vision techniques may detect patterns of maize leaf damage caused by faw. With these advancements, farmers can quickly check wide areas, identify damaged sections, and apply pesticides before the faw spreads over large areas of a farm, lowering costs without sacrificing yield.

1.4. **Study Area**

As shown in Figure 1.2, the research area was in Morogoro district, Tanzania, and comprised four distinct 7-hectare farms. Three farms are located at Mikese with coordinates (6°45'57" S, 37°54'44" E), (6°46'2" S, 37°54'33" E), and (6°45'46" S, 37°54'8" E), while one farm is located at Mkambarani with coordinates (6°46'21" S, 37°48'48"E).

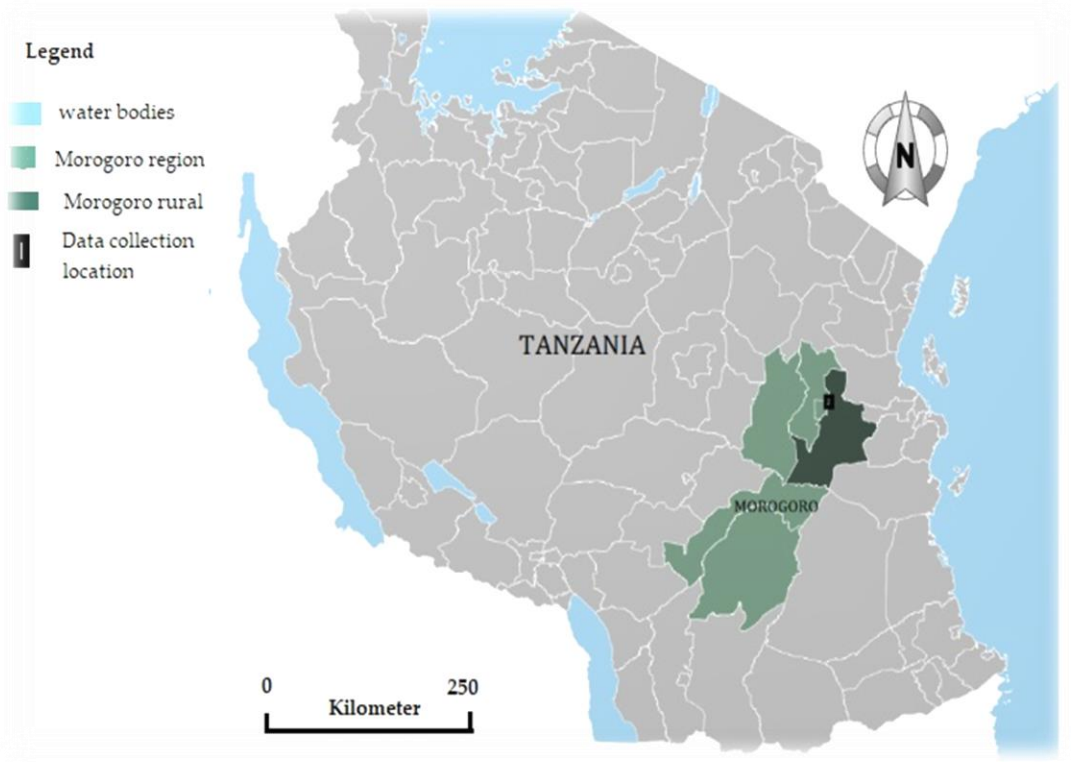


Figure 1.2: Study area location

During data collection, the research clearance permission was followed through all levels. Plant pathologists assisted in the search for and identification of pests, as well as in categorizing the images after cropping them. We spent seven days total in each field; five days looking for maize plants with infected leaves, and two days flying the

drone three times over each field to check for infection. The presence of *small holes*, *semitransparent areas*, and *flesh frass* on leaves is indicative of infection [9], as shown in Figure 1.3. The data was collected in three different seasons: March 2020, March 2021, and May 2022.

(a) (b)

Figure 1.3: a) Maize leaves image showing short holes taken by a digital camera (b) Maize leaves showing semitransparent patches taken by a digital camera.

1.5. **Research Aims**

The primary objective of this thesis is to create a method for the early detection of faw patterns on maize leaves by utilizing UAV technology and ML methods. The following objectives are defined based on the study questions stated in Section 1.6 and the gaps identified in Section 1.3:

1. To identify the pattern caused by faw on maize leaves and apply ML algorithms for decision-making (i.e., distinguish between infected leaves and healthy leaves).
2. To develop a system that combines ML techniques and a UAV outfitted with IoT sensors to report the status of the maize field. (i.e., show percentages of infected, healthy, and weed leaves).
3. To design, develop and deploy an autonomous system for early detection of the patterns caused by faw on maize leaves in the field.

1.6. **Research Questions**

When it comes to understanding the technical feasibility, effectiveness, and sustainability of a system that uses a variety of technologies, such as ML techniques and UAVs, there are still a lot of questions that need to be answered. One example would be; *1) what ML approaches are utilized to detect the pattern generated by faw on maize leaves readily? 2) which technologies can be utilized to develop a system that*

can recognize faw patterns on maize fields and communicate this information to farmers? and 3) how can drone technologies be combined with IoT devices and ML techniques to provide an autonomous reporting system? These questions seek answers to the following general question: how can drone technologies be combined with IoT devices and ML techniques to provide an autonomous reporting system that displays infested plants, their location, and the size of each UAV image covered on the ground? The answers to these questions are essential to gaining a deeper understanding of the topic at hand and determining whether the application of ML strategies and UAV technology can make it possible to detect faw infestations in maize plants at an earlier stage**.**

1.7. **Methodology**

This study is conducted systematically throughout four phases. During the initial phase, data is collected and then analyzed using the Python programming language. In the second phase, Deep Learning (DL) models are developed to recognize the faw patterns on maize leaf images captured by UAVs flying over the fields. During its training, the DL model is trained to differentiate between two groups of maize leaves, namely *infected* and *healthy*. The Shi-Tomasi corner detection method is then used to improve the detection of faw patterns on maize leaves. The third stage entails improving DL models for classifying UAV images as *healthy, infected, weed*, or *redundant*. The images captured by the UAV are segmented into four distinct groups to enable autonomous detection and reduce misclassification. In turn the model reduces the computation time and improves precision and sensitivity in real-time environment. The fourth phase comprises of building an independent system that would receive data, preprocess it, recognize faw-caused patterns, and provide a report to the farmer specifying the number of infected images and the position of each infected UAV image. The system utilizes a combination of the UAV and DL algorithms for data analysis.

1.8. **Thesis Overview and Contributions**

The research in this thesis is organized into six chapters as shown in Figure 1.4, each dedicated to a specific goal. **Chapter 1** begins with a general introduction that discusses the study's framework. Then **Chapter 2,** covers a literature review on computer vision in agriculture, the use of artificial intelligence in agricultural improvement, a summary of deep learning models, and an explanation of the state-of-the-art methods for detecting disease on maize.

Chapter 3 provides a method for detecting faw patterns using the Shi-Tomasi corner detection method. This addresses *objective 1*. The model categorizes UAV images as infected or healthy, ignoring background images such as weeds, maize tassels, stems, and soil. The performance of the original and Shi-Tomasi corner-detected images is compared using four convolutions neural network (CNN-based) transfer-learning models: InceptionV3, MobileNetV4, VGG16, and VGG19.

Chapter 4 proposes a CNN-based hybrid model trained on UAV images to categorize them into four groups: infected, healthy, weed, and redundant. The new model was developed because the previous chapter's Shi-Tomasi corner detection method was producing more noise in the background images, leading to misclassification. The proposed system can classify images autonomously by categorizing them into four groups, which addresses *objective 2*. The hybrid model's performance is compared to four CNN-based models: XceptionNet, Resnet50, VGG16, and InceptionV3.

Chapter 5 addresses two concepts that improve the system created in the previous chapter. First, the chapter discusses how to enhance the contrast of images captured from various farms or the same farm but at different seasons to match training images, lowering real-time misclassification. Second, it describes how to detect the global positioning system coordinate (GPSc) of infected images and determine the size of the infected area. This process enables farmers to take immediate action in damaged areas. These ideas contribute to *Objective 3*.

Final **Chapter 6** reviews Chapters 2-5 and gives the conclusion, recommendations, and directions of the research's future work.

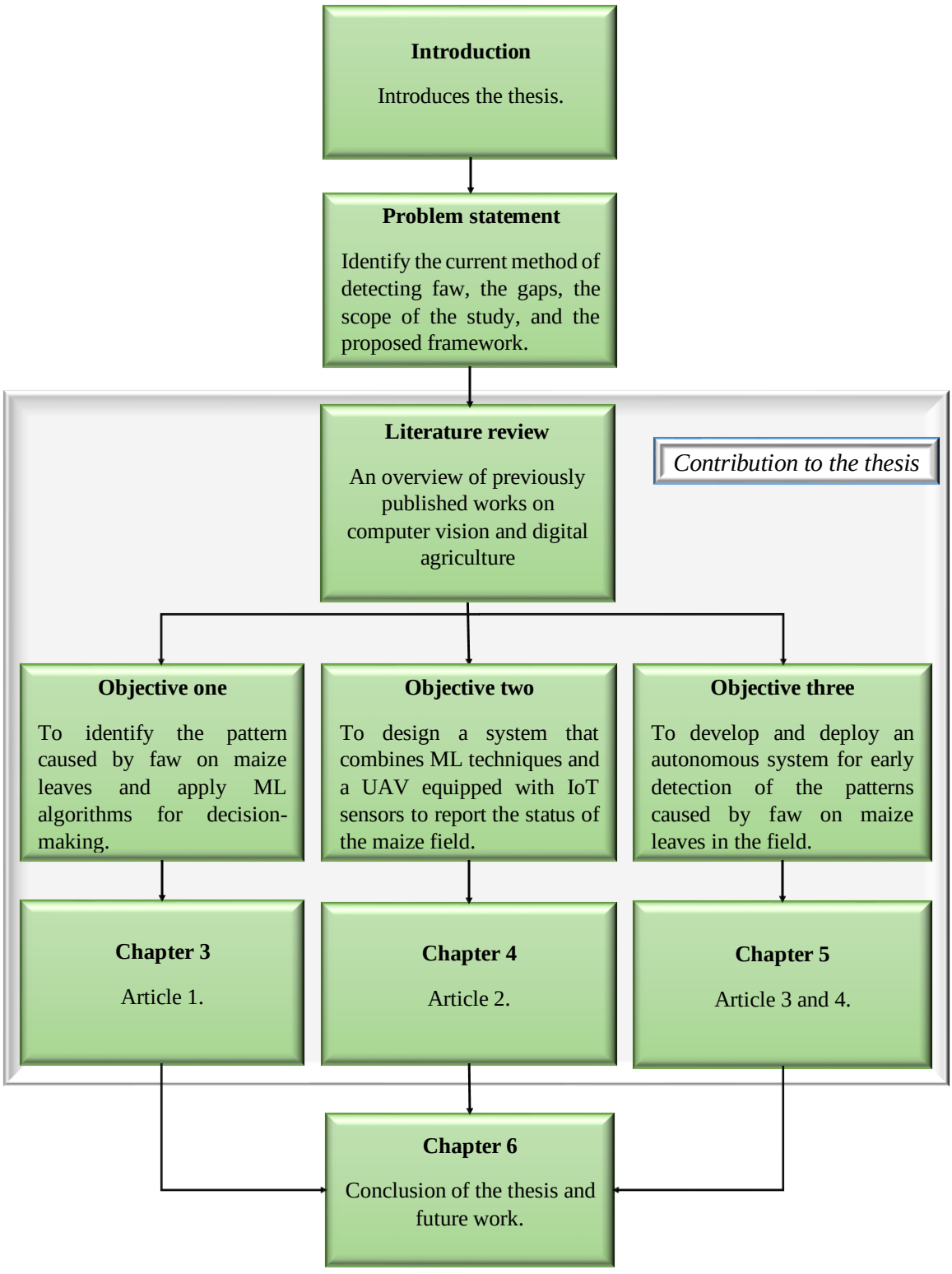


Figure 1.4: Organisation of the thesis

Farian S. Ishengoma

This page has been purposefully left blank.

Chapter 2

Literature Review

2.1. **Introduction**

This chapter provides a literature review on the application of digital agriculture, computer vision in remote sensing, and artificial intelligence (AI) in improving agricultural practices. In addition, we emphasize the contributions made and examine the gaps in the previously published material.

2.2. **Digital Agriculture**

The term "digital agriculture" (DA) refers to the implementation of cutting-edge and emerging technologies into a unified framework to assist farmers and other stakeholders in the enhancement of their goods and procedures [14]. Farm management can benefit from DA by boosting traceability, security, and automation. With the growth of AI and ML, data processing, analysis, and manipulation have evolved significantly [15]. Farmers can increase their farm's yield with the help of DA, saving money and reducing risk [16]. DA is the future of the agricultural business since everything can be monitored in real-time [14]. This increases crop value while reducing costs and the amount of energy required on the farm. DA can be broken down into two distinct farming approaches, which are referred to respectively as *smart farming* and *precision farming* [14].

The goal of smart agriculture is to make farming operations more predictable and effective by focusing on the collection of data and its interpretation using computational technologies. It uses various tools that enable farmers to check the state of their fields without physically going into the fields themselves [16]. The power to make decisions for an entire farm, a vast area, or even a single plant is granted to them because of precision farming, which also makes crop and livestock farming more accurate, optimized, and regulated [16], [17].

2.3. **Computer Vision in Agriculture**

Computer vision (CV) is a subfield of artificial intelligence that enables computers to reason and interpret visual input similarly to humans [18], [19]. CV technology can detect, identify, characterize, anticipate, and evaluate specific objects in a wide range of views,

from still photos to live video. Its objective, especially with DL, is to train AI to the point where it can do more than automate processes [18], [19], [20]. Monitoring crop health and growth, preventing and controlling weeds, diseases, and pests, etc, are just a few of the many agricultural uses that have been the focus of numerous studies that employ CV, AI, and UAV [14], [18].

2.3.1. Computer Vision Applications in Agriculture

How we process, analyze, and modify agriculture data have been fundamentally altered because of recent developments in AI and ML, particularly those in the fast-evolving field of DL [18]. This is largely because of the recent uptick in interest in DL as a new frontier in AI, in which the most representative and discriminative characteristics are learned end-to-end and hierarchically [18], [21]. DL approaches have been widely successful in typical CV tasks like target detection, visual identification, and robotics. They have also achieved extensive success in a wide variety of other practical applications [20], [22].

Most studies have implemented a CNN algorithm that is derived from DL [22], [23], [24]. CNN can distinguish distinct objects in an image by using the input image, learnable weights, and biases. Compared to other classification algorithms, CNN requires a substantially lower amount of pre-processing [23]. CNN can learn these filters with sufficient training, despite the original approaches requiring hand-engineered filters [23], [25]. CNN can be trained from the ground up or using the transfer-learning method. On the other hand, Transfer Learning (TL) uses a previously trained model as the foundation for a new model applied to a different task. CNN trained from scratch requires more data, computation resources, and training time to make the model generalize, whereas TL uses a model that has already been trained [25]. For the model's weights to become more generalized, they are trained on many images before being applied to a different task. Using network weights is a method that contributes to improved generalization while requiring fewer data, less computational resources, and time spent on training [25], [26].

In this thesis, we used CNN-based models that were trained utilizing the fine-tuning TL method. These models were VGG16, VGG19, InceptionV3, MobilenetV2, XceptionNet, and Resnet50. The restricted computational resources and data led us to use these models. Some researchers have employed CV and DL to monitor the health of plants in the fields and their growth, as well as to control various diseases, pests, and weeds, and to monitor farms by employing remote sensing technologies, particularly UAV [25].

2.3.1.1. Crop Health and Growth Monitoring

Monitoring the health and development of the crop is essential for maximizing production, identifying patterns in farm operations, and establishing the optimal time to apply fertilizer.

For instance, using the photos captured by the digital camera, M.P. Rico-Fernández et al. investigated the effects of botanical indicators and color space by employing various ML methods. They found variations in crop types, leaf color, and other characteristics. Comparisons were made between various existing technologies, such as ML and threshold processing. Based on the data, a novel formulation that uses the CIELUV color space and a support vector machine (SVM) was presented. It has achieved results that are encouraging in the field of agricultural monitoring. This technique is adaptable to various climatic conditions and farming species. However, image analysis and improving the efficiency and accuracy of analysis while the images are being seen in backlight settings remains an incredibly tough task [27]. In addition, Rodrigo Pérez-Zavala et al. employed a camera sensitive to the visible spectrum to locate grape bunches and identify grape berries. The suggested approach is determined by the aggregated pixel regions' shape, texture, and segmentation. The findings of the study demonstrate that there was an increase in the accuracy of grape monitoring. This approach is reliable even when applied in environments with different degrees of illumination and occlusion [28]. Moreover, Sun et al. proposed a method that used continuous imaging to examine how the shape and color of rice leaves changed over time in response to variations in nitrogen (N), phosphorous (P), and potassium (K) treatments. They assessed the usefulness of dynamic characteristics for identification. All images were processed in MATLAB to extract the morphological and color features for dynamic analysis, and scans were performed on the top four leaves (the first incomplete leaf and the top three fully developed leaves) every three days. When assessing the efficiency of dynamic indices, the mean impact value was afterward used. Results showed that N shortage had the most significant influence on leaf growth, followed by P deficiency, and finally, K insufficiency; increased nutrient availability led to faster leaf extension and a lower growing rate of chlorosis [29].

On the other hand, orthophoto processing utilizing the MATLAB image-processing technique was carried out on palm oil plantations by F. Fahmi et al. using UAV. They employed a Grayscale Co-occurrence Matrix (GLCM) approach to categorize fertile,

sterile, and dead palm oil plants. The criteria they generated were based on four directions and specific degrees of 0 degrees, 45 degrees, 90 degrees, and 135 degrees. Experiments showed that UAV monitoring could provide information more quickly and accurately than traditional approaches [30].

2.3.1.2. Crop Diseases, Insect, and Weed Control

Controlling and preventing the spread of crop diseases, insects, and weeds are essential in producing high-quality agricultural goods and generating high yields. It is possible to avoid yield losses caused by diseases and weeds by controlling them automatically and promptly using modern farming practices.

For instance, Liu and Chahl, created a multispectral computer vision system for detecting invertebrate pests widely seen on wild green leaves. This system was able to identify the pests. During the studies, a satisfactory degree of accuracy was established when identifying twelve prevalent invertebrate crop pests. In addition to having a high degree of accuracy, the system can provide robots with decisions regarding their actions in real-time. It is feasible to make intelligent predictions if one uses accurate and real-time monitoring; this will be very important for the prevention and control of agricultural diseases [31]. Moreover, Zhong Y et al. devised and put into action a vision-based monitoring system that, when tested with You Only Look Once (YOLO) and SVM techniques, was shown to have excellent performance. The overall accuracy rate of the classification process was 90.18%, while the overall accuracy rate of the system was 92.50%. The system can serve as a full-service platform for predicting the occurrence probability and development trend of pests, both of which are highly significant, in addition to providing identification data that is effective and accurate. Both factors are incredibly vital [32].

On the other hand, Toseef and Khan suggested an intelligent crop disease diagnosis technique that can act as the critical back-end decision engine. To provide an output in the form of a disease diagnostic, the system takes crop symptoms as its input and then uses an inference engine to decide. The method has a diagnostic accuracy of 99% for the most important diseases, and it may be used on the two most important crops, which are cotton and wheat. The system initially generates a tiny data set, can monitor two crops simultaneously, is straightforward to implement, and possesses a great deal of untapped potential [33]. Likewise, Sabzi et al. presented a CV expert system that is based on neural

networks. This system would recognize potato plants and three other types of weeds (Secale cereale L., Polygonum aviculare L. and Xanthium strumarium L.) for on-site spraying. The 126 color features and 60 texture features retrieved from each object are listed below. The experiment results showed that the proposed expert system had an accuracy of 98.38% and an average execution time of less than 0.8 seconds. Nevertheless, when there was an exceptionally high plant density, the usage of the system was restricted [34]. By merging aspects of computer vision and multitasking, Chang and Lin were able to design a compact intelligent agricultural machine capable of automatically weeding cultivated area and adjusting the amount of irrigation that is applied. The machine can weed and water while keeping a deep soil moisture content of $80 \pm 10\%$ and an average herbicidal rate of 90% since it can distinguish between plants and weeds in real-time. This strategy has a lot of potential because it incorporates multitasking and uses all available resources to their maximum capacity [35].

2.3.1.3. Farm Monitoring with UAV

Precision agriculture requires real-time agricultural data management and correct analysis [36]. As rapidly developing technology, UAVs have made it possible to get agricultural data with high resolution, cheap cost, and speedy solutions in recent years [37]. The UAV is a form of aircraft that can fly missions without the need for a human pilot. Unmanned aircraft systems comprise three components: an aircraft component, sensor payloads, and a ground control station [38]. They can be controlled by electronic equipment on board the vehicle or on the ground. It is a remotely piloted vehicle when controlled remotely from the ground. For it to work, it needs to be able to communicate wirelessly in a reliable manner [38]. UAVs can be built on either a fixed-wing or a rotary-wing platform. The ability of fixed wing UAVs to fly at high speeds for extended periods while utilizing simpler aerodynamic elements is one of the advantages of these vehicles. When taking off and landing, some do not even require a runway or a launcher. The ability of rotary-wing UAVs to take off, land, and hover vertically over a target is a significant advantage [38], [39]. Using UAVs for remote sensing has increased agricultural output while simultaneously lowering costs [40].

Multiple studies have been conducted using UAVs, ML techniques, and other technology to monitor and improve farming practices. For instance, Han et al carried out a comprehensive study on the biomass estimation of maize, an important crop. In this

study, UAV remote sensing was combined with ML to provide spectral information that could be used to estimate maize biomass. The researchers used an Octocopter DJI Spreading Wings S1000 UAV platform with two cameras. They also proposed a method for extracting plant height from UAV images and showing volume in trials, which revealed that the method might contribute to increases in precision and demonstrated that the technique should be improved. Because of this, the integration of ML and UAV remote sensing provides an exciting new alternative [41]. In addition, the point cloud collected by the UAV-RGB (Quadrotor UAV-RGB remote sensing system DJI Phantom 4 Pro) was used by Yaxiao Niu et al. to derive data on maize biomass and elevation. The outcomes demonstrate the benefits of utilizing this approach to create a high-performance ML estimate model [42].

Moreover, Souza et al. proposed a method for mapping sugarcane hoppers based on object analysis of UAV images (OBIA) utilizing an eBee Ag equipped with a Canon Power Shot S110 compact. This study demonstrates the high degree of automation and flexibility of the OBIA method, which may be applied to collecting essential information for agricultural decision-making and monitoring [43]. Xiang Shi et al. developed a decision support system for variable-rate irrigation using UAV multispectral remote sensing (VSI-DSS). The vegetation index (VI) is calculated from multispectral images taken by UAVs, and the system subsequently provides users with recommendations based on the results. The findings confirm that the approach is rational and in line with the anticipated outcomes but also highlight the need for further improvements in terms of reliability [44]. Due to the widespread availability of image data, UAV platforms are used in nearly all-precision agriculture. Many farming tasks, including crop monitoring, protection, and management, can benefit from deploying of UAVs [16], [40]. With its adaptability, timeliness, and stability, this method improves the scientific quality of assessment and helps with the strategically manage of agricultural resources [45].

The crop monitoring and management described in this thesis were carried out with the assistance of a quad copter drone known as a Phantom 4 Pro v2. The quad copter drone known as UAV is outfitted with a standard built-in camera that has a standard length of the camera's focus that measures 8.6mm, a sensor width that measures 12.83mm, a resolution that measures 5472 by 3078 pixels, a lens that has a field of view that measures 84, and an aperture that measures F/2.8 at infinity.

2.3.2. Pattern Recognition and Matching

Pattern recognition (PR) is a branch of AI and ML that focuses on identifying and interpreting data patterns or regularities. It refers to the process of identifying and categorizing objects or events based on their underlying patterns or structures [46], [47][48]. PR techniques are used to analyze and extract meaningful information from large amounts of data, such as images, signals, text, or numerical data [49] [50]. The goal of PR is to create algorithms and models that can recognize and categorize patterns automatically in an accurate and efficient manner. Visual features, statistical distributions, temporal sequences, and any other discernible characteristics within the data may be included in these patterns[49] [50] [51]. Computers can make informed decisions, predictions, or classifications based on identified patterns by recognizing patterns [47].

2.3.3. Images Processing Methods

Digital image processing primarily focuses on signal processing research, which includes picture contrast adjustment, image coding, filtering, and denoising [52]. To improve the visual information in an image for either human interpretation or autonomous machine perception, image processing modifies the image's characteristics. Two primary techniques used in this area are preprocessing and feature extraction [30], [52]. Preprocessing is necessary before images can be used for model training or inference. It includes transformations such as adjusting brightness, hue, saturation, edge sharpness, scale, and orientation [53]. Feature extraction, on the other hand, aims to reduce the size of the initial data collection by analyzing the variations across input patterns based on a set of shared characteristics or feature [30]. Pixels that collectively indicate edges, lines, and other shapes are organized into more sophisticated features [52].

2.3.3.1. UAV Image Processing

When assessing DL-based algorithms applied to remote sensing data, three significant tasks emerge: scene-wise categorization, object detection, and semantic and instance segmentation [22], [54], [55]. Scene-wise categorization involves assigning a class label to each image within a scene. On the other hand, object detection refers to the process of labeling specific regions of an image using bounding boxes [54], [55]. Object detection is a complex task as it involves locating and labeling objects within an image [55]. Semantic segmentation, an alternative approach to drawing bounding boxes, involves delineating

regions or structures around object boundaries to differentiate their classes at the pixel level. However, semantic segmentation may struggle to distinguish multiple objects of the same category, as each pixel is assigned only one class label [54]–[57]. To circumvent this restriction, the instance segmentation task was introduced. It combines semantic segmentation with object detection, enabling the recognition of multiple objects through pixel-level masks, each tagged with a class label [56], [58]. However, Instance segmentation allows for the precise identification of objects on a pixel-by-pixel basis while maintaining their distinction [59].

2.3.3.2. Real-time UAV Image Processing

Using of UAVs in tandem with DL-based methods accelerates processing and classification, but the underlying algorithms are computationally intensive [54], [55]. Even though DL is often promoted as a quick way to extract information from data after training, its architecture's complexity can be a bottleneck for real-time applications. Therefore, real-time DL methods necessitate the creation of more efficient algorithms or specialized graphic processing unit (GPU) processors [20], [22], [23], [54].

In remote sensing, high-resolution images captured by UAV provide a large amount of information with minimal variation between classes, as stated by Xu et al, [60]. Besides their proximal sensing capabilities, UAVs have an advantage over other types of sensing platforms due to their ability to capture images at a higher resolution. Because of these variations and the tremendous volume of data, it is more challenging to extract valuable characteristics using traditional CNNs, which give equal weight to all regions. DL architectures typically require low-resolution input images because they reduce computation time and aid the model's generalization ability. For example, it took days for models running on a GPU to converge, even when using the input dimension of 256x256 pixels, as proposed by Krizhevsky et al. [61].

In this thesis, the photos were resized to 150x150 pixels for easier classification and training using a central processing unit (CPU). ML techniques were employed to categorize the images into four distinct groups on a CPU-powered computer. The system successfully and instantly identified the images, which initially had a resolution of 5078x3000 pixels, and assigned them to their respective categories.

2.4. **Machine Learning**

ML employs data and algorithms to teach computers how to learn like humans, allowing them to improve at tasks such as prediction, classification, and insight discovery [62]. It works in three stages: first, data and algorithms are used to make predictions and categorize data sets; second, an error function helps evaluate the accuracy; and third, the data points are optimized to suit the model best [62], [63]. ML Based on their respective learning strategies, machine learning may be broken down into three distinct categories. In Figure 2.1, we can see the connection between AI, ML, and DL.

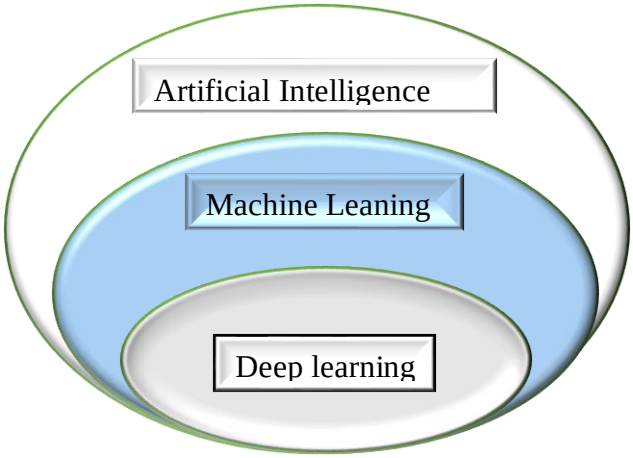


Figure 2.1: The artificial intelligence overview

a) Supervised Machine Learning

Supervised learning, also known as supervised machine learning, involves using labeled datasets to train algorithms for accurate data classification and prediction. Here, machine learning algorithms are fed with pairs of input and output data from the past, with processing in between each pair allowing the algorithm to adjust the model to provide outputs as close to the desired outcome as feasible. Typical supervised learning methods include neural networks, decision trees, linear regression, and support vector machines [20], [54], [62].

b) Unsupervised Machine Learning

In contrast to supervised learning, unsupervised learning does not require all data and training sets to be labeled in the same way. It analyzes and organizes unlabeled data sets

with the help of machine learning algorithms. These algorithms enable the automatic discovery of previously unknown relationships between data points. Exploratory data analysis, cross-selling techniques, consumer segmentation, and picture and pattern recognition all benefit greatly from this method's capacity to identify patterns and similarities in large amounts of data. Methods like Hidden Markov models, k-means, hierarchical clustering, and Gaussian mixture models are all examples of popular unsupervised learning techniques [20], [54].

c) Reinforcement Machine Learning

Unlike supervised learning, where an algorithm is trained with examples, reinforcement learning does not require pre-existing data. Instead, it uses the same process of trial and error that humans do to acquire knowledge. It gains knowledge when the agent or algorithm engages with its environment and responds to stimuli, positive or negative. For reinforcement learning to work, the environment must be static or have a vast amount of relevant data. It is believed that it is simpler to work with than supervised learning when dealing with unlabeled data sets because it requires less management. The temporal difference, deep adversarial networks, and Q-learning are only a few of the most popular methods [64], [65].

2.4.1. Common Terms Used in Machine Learning

2.4.1.1. Underfitting

An underfit model fails to forecast test data reliably and does not generalize well. Poor results on training data will reveal a machine-learning model to be unfit for its purpose. When a model is underfitting, it does not learn enough from the training data; hence, its predictions are inaccurate; poor model fitting results in low variance and significant bias. Figure 2.2 demonstrates that the model does not adequately represent the plotted data. Increasing the training duration and the number of features offered helps prevent underfitting [52].

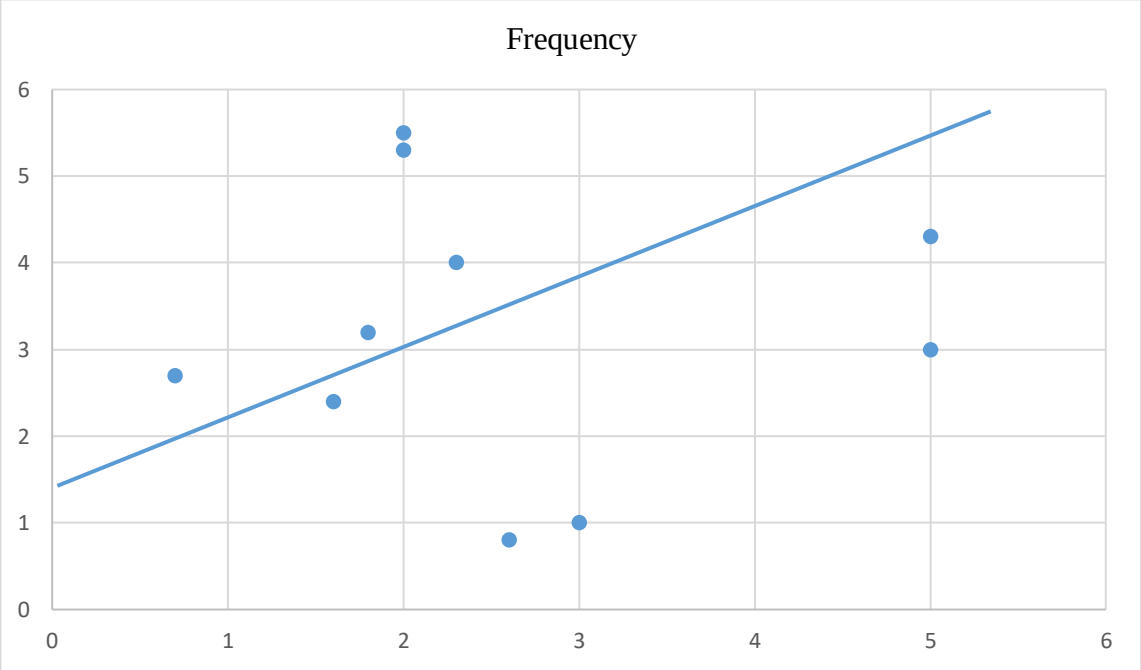


Figure 2.2: Graph describing the underfitting state.

2.4.1.2. Overfitting

When a machine-learning model attempts to account for every observation in a dataset, this is known as overfitting. Caching noise and wrong values in the dataset causes the model to become less effective and accurate. The variance of the biased overfit model is quite large. In supervised learning, overfitting is by far the most frequent issue. As can be seen in Figure 2.3, the model tries to account for all the points in the scatter plot [52].

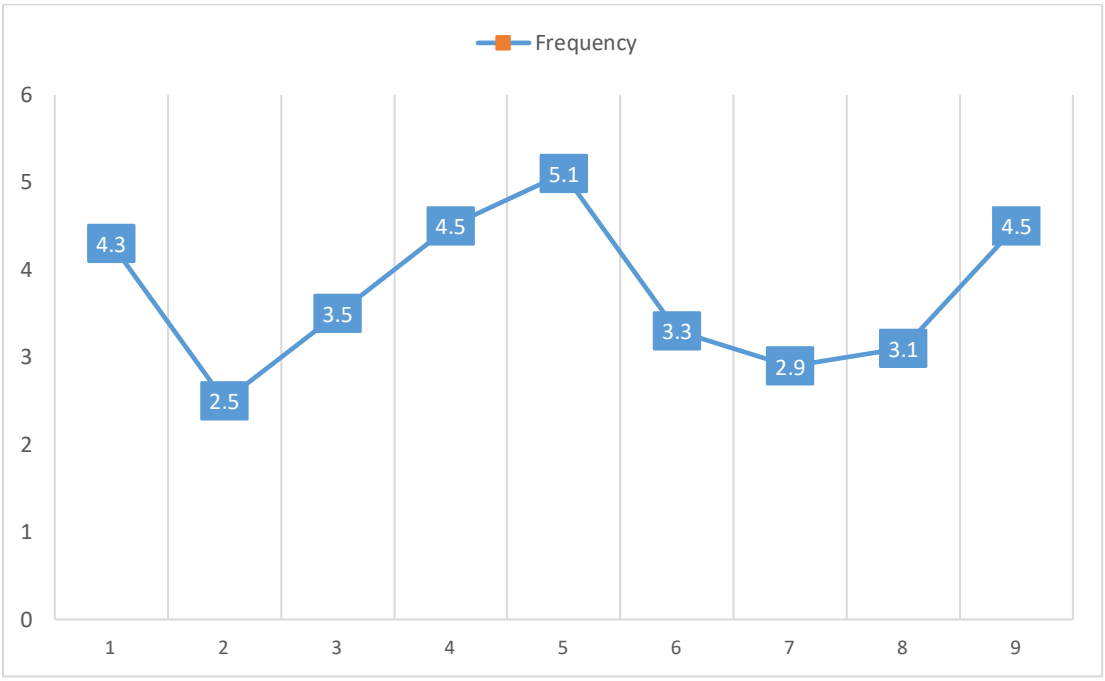


Figure 2.3: Graph describing the overfitting state.

The performance of a model suffers from overfitting when it takes in too much information from the training data, including irrelevant noise. This means the model can identify noise or random oscillations in the data and interpret them as concepts during the training. Problematically, the models' generalizability is hindered by the fact that these notions do not apply to novel data [52]. Both overfitting and underfitting decrease the ML model's effectiveness. However, cross-validation, training with more data, deleting features, stopping training early, regularization, and ensemble can all help prevent overfitting, the root reason.

2.4.1.3. Loss functions

The quality of a program's dataset representation is evaluated using loss functions. If the prediction is wrong, the loss function increases otherwise it will drop. Loss function changes reflect algorithm improvements [65].

2.4.1.4. Optimizer

Optimizers are functions or algorithms used to find the optimal values for a neural network's weights and learning rate. As a result, it helps to improve accuracy while decreasing overall loss [66]. Because deep learning models typically have millions of parameters, picking the best weights for them can be difficult [24], [54]. Optimizer work

involves using of concepts such as epochs, batch size, and learning rate [66]. The batch size that defines how many samples will be utilized to update the model parameters. At the same time, the number of epochs is the number of times the algorithm iteratively runs through the full training dataset [24], [55], [52]. As an alternative, the learning rate (η) is a parameter that instructs the model as to how often it should update the model weights to achieve a local minimum [52], [55]. In this thesis, we use several optimization algorithms to evaluate the efficacy of our models, including Gradient descent, Stochastic gradient descent with momentum, Mini-batch gradient descent, Adagrad, RMSprop, Adadelta, Adam, Adamax, and Nadam [66]–[69]. Appendix II has a full description of the optimization algorithm.

2.4.1.5. Weights

The weight of a neural network is a parameter that modifies the input data at the network's hidden layers [70]. Neurons, or nodes in a neural network, perform several functions, including storing a set of inputs, a weight, and a bias value. As shown in Equation 1, each node in a neural network receives an input, which is multiplied by some weight value and then either observed or sent on to the next layer [23], [54], [70], [71]. It is common practice for a neural network's weights to be stored in its hidden layers. The strength of the signal (or link) between two neurons is controlled by weights [55]. A weight quantifies the significance of the input in producing the desired result. A constant bias is an extra input into the following layer, always with the value one [23]. Since bias units don't get any information from the layer below them, they can act independently by sending information to the world via connections with their weights [55]. The bias module ensures that the neuron works even when there are no inputs.

$$Y = \sum (weight * input) + bias \qquad 1$$

2.4.1.6. Activation Functions

The activation function (AF) evaluates the input against a target value. If the input value is greater than the threshold value, the neuron is enabled and its output is passed on to the next or hidden layer; otherwise, the neuron is inhibited [68]. Lacking an activation function, a neural network's output signal is a simple linear function, a polynomial of degree one. While it's true that linear equations are easy to work with, they can only handle very straightforward mappings and cannot learn or detect complex patterns from data [71].

Without an activation function, a neural network typically performs as a weak Linear Regression Model [71]. Instead of learning and computing a linear function, a neural network is best used to model complex data including images, videos, audio, speech, and text [68]. Binary step function, linear, sigmoid, tanh, rectified linear unit (ReLU), leaky ReLU, parameterized ReLU, exponential linear unit, swish, and softmax are some of the most used activation functions [68], [72].

i.) Rectified Linear Unit Function

ReLU is an activation function in neural networks that is approximately linear [68]. All input elements undergo a threshold operation, and all values below zero are transformed into zero. The ReLU activation function has been shown to outperform the Sigmoid and Tanh activation functions in deep learning [73]. Using the ReLU function has the benefit that not all neurons are stimulated simultaneously. This means the linear transformation output must be zero before a neuron is silenced. The gradient value can be zero during the training of a neural network, in which case the weights and biases will not be changed during the backpropagation phase. Equation 2 is the mathematical representation of ReLU.

$$f(x_i) = \max(0, x_i) = \begin{cases} x_i, & if\ x_i \geq 0 \\ 0, & if\ x_i < 0 \end{cases} \qquad 2$$

This function forces less-than-zero inputs to zero, solving the vanishing gradient problem from previous activation functions. ReLU is utilized in deep neural network hidden units with additional activation function in the output layers for object categorization [74]. ReLU ensures speedier processing because they do not compute exponentials and divisions [73]. ReLU introduces sparsity in hidden units by squashing zero-to-maximum values. ReLU overfits readily compared to the sigmoid function, despite the dropout approach and corrected networks improving deep neural network (DNN) performance [75]. The ReLU and its variants have been used in various DL architectures, including restricted Boltzmann machines [76] and CNN architectures [77], [74], [78]. According to Nair and Hinton. the ReLU has been used in various architectures due to its simplicity and dependability [76]. ReLU's weakness during training can kill gradients. Some neurons die, inhibiting future weight updates and stifling learning [79]. Leaky ReLU was offered as a dead neuron solution.

a) Leaky ReLU

The leaky ReLU (LReLU) was introduced as an activation function in 2013 to keep the weight updates alive throughout the propagation process [80]. This AF involves the introduction of a modest negative slope to the ReLU. The alpha parameter was implemented to fix the problem of dead neurons in ReLUs. This guarantees that the gradients are never zero while training. Since the LReLU uses a small constant value, approximately 0.01 for the negative gradient, to compute the gradient, the LReLU AF may be computed using Equation 3, "a" stand for a small constant set to a value less than 1.

$$f(x_i) = \max(ax_i, x_i) = \begin{cases} x_i, & if\ x_i > 0 \\ ax_i, & if\ x_i \leq 0 \end{cases} \qquad 3$$

The results of the LReLU and tanh functions are the same as those of the regular ReLU [80], except that the LReLU and tanh functions have non-zero gradients throughout the time. This indicates that the result is not significantly improved, perhaps in terms of sparsity and dispersion. Automatic speech recognition data was used to evaluate the LReLU.

b) Parametric Rectified Linear Units (PReLU)

The parametric ReLU (PReLU) is a variant of the ReLU AF developed by He et al. in which the negative portion of the function is adaptively learned. In contrast, the positive part is linear [74]. Equation 4 represents the PReLU.

$$f(x_i) = \begin{cases} x_i, & if\ x_i > 0 \\ a_i x_i, & if\ x_i \leq 0 \end{cases} \qquad 4$$

Where a_i is the negative slope controlling parameter can be learned through back-propagation training. If a_i= 0, the PReLU becomes ReLU. The PReLU can be expressed as shown in Equation 5.

$$f(x_i) = max(0, x_i) + a_i min(0, x_i) \qquad 5$$

The authors reported that PReLU outperformed ReLU in large-scale image recognition and that PReLU was the first to surpass human-level performance on a visual recognition challenge [74].

c) Randomized Leaky ReLU (RReLU)

A dynamic variant of the leaky ReLU, randomized leaky ReLU trains its network with a uniformly distributed random number *(l, u).* Equation 6 is used to calculate the ReLU under random conditions.

$$f(x_{ji}) = \begin{cases} x_{ji}, & if\ x_{ji} \geq 0 \\ a_{ji}x_{ji}, & if\ x_{ji} < 0 \end{cases} \quad 6$$

Where $a_i \sim U(l,u)$, $l < u$ and $l,u \in [0.1]$

The test phase uses of the same averaging method as the training phase but without the dropout used to converge the learnt parameter to a deterministic value $a_i = \frac{l+u}{2}$. As a result, Equation 7 is used to obtain the actual test output.

$$y_{ji} = \frac{x_{ji}}{\frac{l+u}{2}} \quad 7$$

Xu et al., demonstrated that LReLU, RReLU, and PReLU outperform the ReLU on classification tasks [81], and the RReLU has been tested and compared against the other variations of the ReLU on standard classification datasets.

ii.) Sigmoid

In some literature, the Sigmoid activation function $f(x)$ is also called the logistic function or the squashing function [68]. In feed-forward neural networks, the Sigmoid is a popular non-linear activation function. It takes in any actual number and outputs a number between 0 and 1. When the input is high, the output approaches 1.0, when it is small, it approaches 0. Equation 8 describes the relationship that characterizes the Sigmoid function.

$$f(x) = \frac{1}{1 - e^{-x}} \quad 8$$

The sigmoid function is often utilized in the output layer of deep learning networks to make probabilistic predictions. It has been effectively implemented in various neural network settings, including binary classification issues, modeling logistic regression tasks, etc. Neal [82] argues that sigmoid functions' primary benefits lie in the fact that they are easily grasped and are typically used in shallow networks. When training a neural network with modest random weights, advise avoiding the Sigmoid [83]. Sharp damp gradients during backpropagation from deeper hidden layers to input layers, gradient saturation, slow convergence, and non-zero centered output (which allows gradient updates to propagate in various directions) are only a few of the major shortcomings of the Sigmoid activation function [68].

iii.) Softmax

Softmax functions are another kind of activation function used in neural computation. It accepts a vector of real numbers as input and spits out a probability distribution. The Softmax function $f(x_i)$ produces a continuous variable with values between 0 and 1, with 1 representing the cumulative probability. As shown in Equation 9, softmax [79] can be computed by using that formula.

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \qquad 9$$

When used with multi-class models, the Softmax function assigns the highest probability to the target class, returning probabilities for each class. DL architectures typically use the Softmax function in the last layer [84], [85]. In contrast to the Softmax activation function, which is utilized for multivariate classification, the Sigmoid activation function is only used for binary classification.

2.5. **Deep Neural Network**

DL is a subfield of ML that makes use of multilayer neural networks [20], [25], [54], [55], [70], [86]. With limited success so far, these neural networks attempt, to simulate the functioning of the human brain by allowing it to acquire knowledge through exposure to large datasets [19], [20]. Single-layer neural networks are suitable for approximations, but multi-layer networks allow for optimization and fine-tuning accuracy. DNN prediction and optimization abilities are built up layer by layer from a series of interconnected nodes [25], [54]. It is the process through which calculations are transmitted from one node to another in a network. The only layers of a DNN typically on display are the input and output layers. A DL model takes in data at its "input" layer and "outputs" a prediction or classification at its "output" layer [19], [25], [54]. By working backward through the layers, backpropagation trains a model by using methods like gradient descent to calculate prediction errors and then modifying the function's weights and biases [24], [55]. Together, forward and backpropagation enable a neural network to learn from experience and produce highly accurate predictions [87]. CNNs, recurrent neural networks (RNNs), long short-term memory (LSTM), generative adversarial networks (GANs), recurrent belief propagation networks (RBFNs), multilayer perceptron (MLPs), self-organizing maps (SOMs), restricted Boltzmann machine (RBM), deep belief networks (DBNs), and auto-encoders are some of the most popular deep learning algorithms [24], [54]. While CNNs and RNNs

are the most frequent DNNs, they fall within the umbrella of supervised networks [54], [88]. CNN is the most popular choice among remote sensing picture architectures, followed by RNN.

2.5.1. Convolution Neural Network Architecture

A CNN is a DL algorithm for object recognition that takes an input image as well as learnable weights and biases [24], [55]. Compared to other classification algorithms, CNN takes much less pre-processing time [23], [24]. CNN's superior training over more conventional methods that require hand-engineered filters allows it to learn these filters automatically [22], [70]. The structure of the human brain's Visual Cortex serves as an architectural model for CNNs, which mimic how neurons in the brain communicate [24]. Each neuron has a limited "receptive field" to process visual information. Compared to other neural networks, CNN excels when given inputs of images, voices, or sounds. Each consists of three layers: a convolutional layer, a collection layer, and a fully connected layer [22], [23], [25], [54], [55], [70], [86].

As can be seen in Figure 2.4, the first layer of a convolutional network is the convolutional layer. More convolutional layers or pooling layers often follow convolutional layers, but the fully connected layer is always the last in the chain. CNN gets more complicated with each additional layer, which allows it to recognize more significant portions of the image. The earlier layers concentrate on the more fundamental characteristics, such as colors and edges. As the image data progresses through the CNN layers, it starts to recognize larger components or shapes of the object. Eventually, it can identify the one that is sought after.

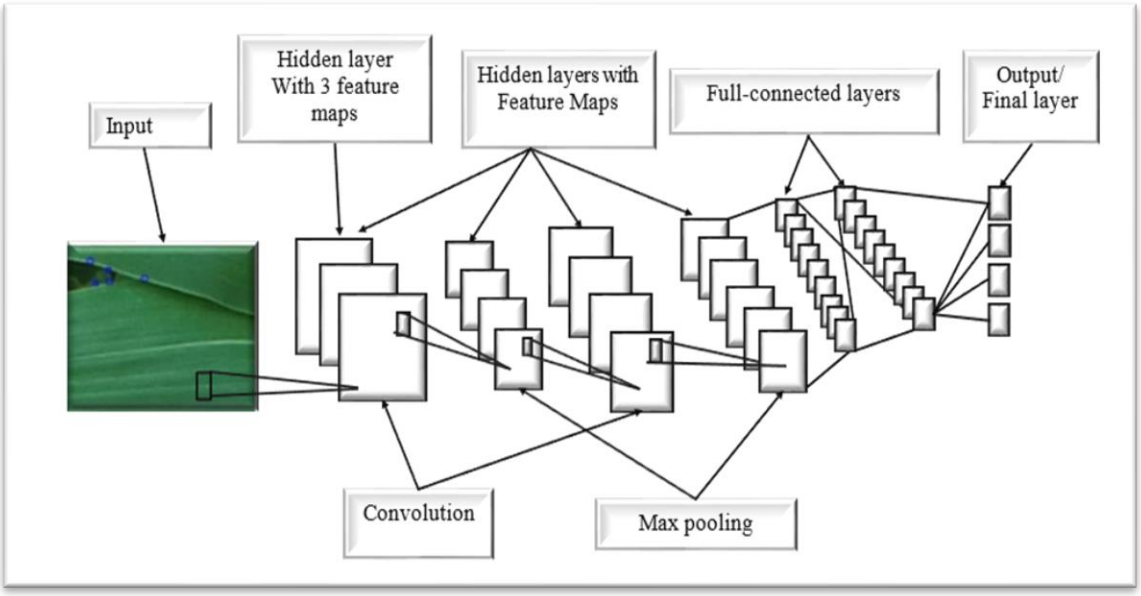


Figure 2.4: Convolution neural network

a) Layer of Convolution

The convolutional layer is the basis of a CNN and is where most of the computation occurs. It requires a few different components, such as input data, a filter, and a feature map, among other things [23], [25]. For example, suppose the input is a color image composed of a three-dimensional matrix of pixels. In that case, this indicates that the input will have three dimensions corresponding to the RGB color space present in an image [27], [28]. These dimensions are height, width, and depth. When searching for a specific feature in an image, a feature detector (also known as a kernel or filter) will scan across the image's receptive fields.

Moreover, the feature detector is a two-dimensional (2D) weighted array used to represent a specific image region. Although the filter size can vary, typically it is a 3x3 matrix, which determines receptive field size [23]. When a filter is applied to an area of an image, the next step is to compute the dot product of the input pixels and the filter. An output array then processes this dot product. The steps are repeated until the entire image is covered by the kernel, at which point the filter moves forward by one-step. The output of a filter is a sequence of dot products based on the input data and the shape of the filter. These dot products are collectively referred to as a feature map, activation map, or convolved feature [22], [23], [55].

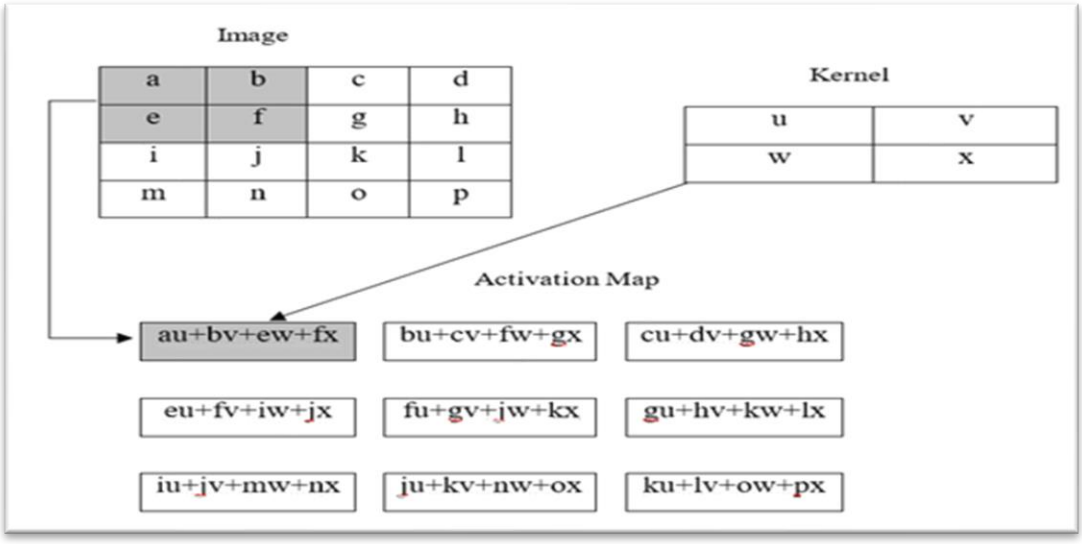


Figure 2.5: Convolution layer

Figure 2.5 shows that it is not required for every feature map output value to correspond to a pixel value in the input image. The only requirement is to be linked to the receptive field, where the filter is used. Partial-connected layers, such as the convolutional and pooling layers, are examples of popular neural network architectures [23]. This is because not every input value needs to be represented exactly in the output array. This feature is also referred to as "local connectivity." It is critical to note that the feature detector's weights remain constant as it moves across the image; this phenomenon is known as parameter sharing. Backpropagation and gradient descent are two methods for adjusting parameters while the model is being trained [24]. To begin training the neural network, three hyper-parameters affecting the size of the output volume must first be set. The following are some examples:

i.) The output depth depends on how many filters are used. For instance, if you used four different filters, you would get four different feature maps, giving you a depth of four.

ii.) Stride is the number of pixels that the kernel moves from one side of the input matrix to the other, as shown in Figure 2.6. Even though two or more stride values do not happen very often, a larger stride makes the output smaller.

iii.) When the filters do not fit the image being sent in, zero padding is usually used, as shown in Figure 2.7.

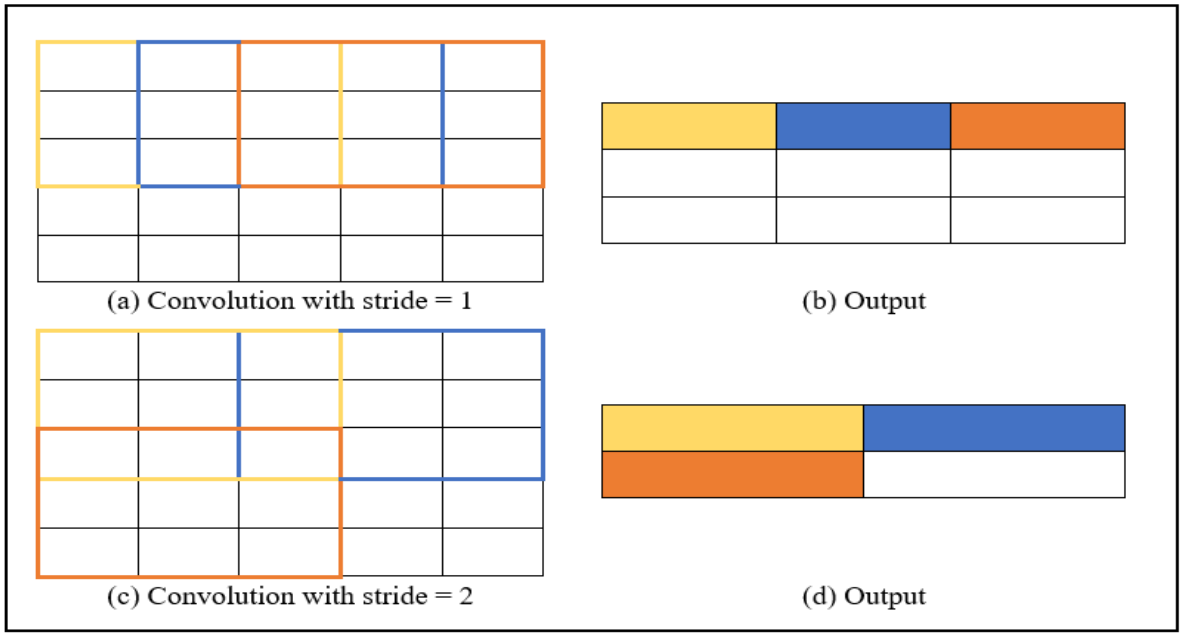


Figure 2.6: Convolution stride input and outputs a) convolution stride=1 b) output of convolution stride=1 c) shows convolution stride=2 d) shows the output of convolution stride=2

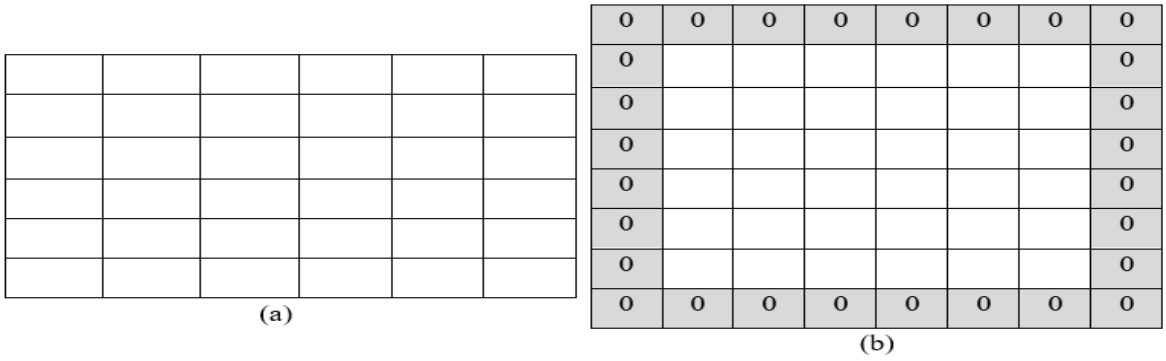


Figure 2.7: Padding (a) 6 x 6 image (b) 6 x 6 image with one layer of zero padding

b) Pooling Layer

The process of pooling layers, which is also referred to as down sampling, performs dimensionality reduction to cut down on the total number of parameters contained in the input [22], [23], [55]. The pooling operation, much like the convolutional layer, will sweep a filter over the entirety of the input; however, unlike the convolutional layer, this filter will not have any weights [22]. In its place, the kernel uses of an aggregation function, which it applies to the values of the receptive field, to populate the output array [22], [23]. There are two different kinds of pooling: *maximum pooling* and *average pooling*.

i.) Maximum pooling is when the filter moves across the input and chooses the pixel with the highest value to send to the output array [22], [55]. This is done as it moves across the input. As a side note, this method is used more frequently than the standard pooling technique [22], [24].

ii.) Average pooling happens as the filter moves through the input, calculating the average value in the receptive field to send to the output array [24].

While the pooling layer can cause significant data loss, it does provide some benefits for CNN [22], [24]. They help simplify things, boost performance, and mitigate overfitting concerns [23].

c) Fully Connected Layer

In full-connected layers, each output node connects directly to the previous node. This layer classifies the previous layers' features and filters [22], [24], [55]. FC layers use a softmax activation function to produce a 0 to 1 probability, while convolutional and pooling layers use ReLU functions [22].

2.5.2. Training Convolution Neural Network

Models for CNNs can be trained either from scratch or through TL. The model trained from scratch takes more time, data, and more powerful computation resources, such as parallel GPUs to generalize [22]. This is incredibly challenging to achieve in image processing because of the high cost of computing resources and the large amount of time and money required to collect millions of images from various classes. On the other hand, TL creates an easy model because the trained model's weights are reused after being tweaked [22], [55]. In the following section, we will delve even further into the significance of TL.

2.5.3. Transfer Learning

TL is altering previously trained network weights on many images before applying them to a new task. This is done to generalize the network's capabilities [26], [55]. Better generalization can be achieved by applying the technique of using network weights [55]. CNN employs TL to reduce the time and money spent on computation [22], [55]. While working on this thesis, we used CNN-based models such as VGG16, VGG19, InceptionV3, Resnet50, MobilenetV2, and XceptionNet that had been trained using the fine-tuning method of TL. These models have been extensively utilized and proven accurate [89][90].

2.5.3.1. Visual Geometry Group

The visual geometry group (VGGNet) consists of sixteen convolutional layers and keeps a uniform architecture by employing filters of the same size (3x3) across all its convolutional layers [91]. With two layers of 3x3, each neuron has a 5x5 receptive field. Two 3x3 layers stacked together had 18 parameters, but a 5x5 layer had 25 parameters, which cut the number of parameters. If you have two convolutional layers, you can connect two ReLU layers instead of just one. So, in both VGG16 in Figure 2.8 and VGG19 in Figure 2.9, the network comprises only 3x3 convolution layers and 2x2 pooling layers. A softmax activation layer is used as a classifier after two fully connected layers with 4,096 nodes each. The image size that goes into the connected layers is 224x224 pixels.

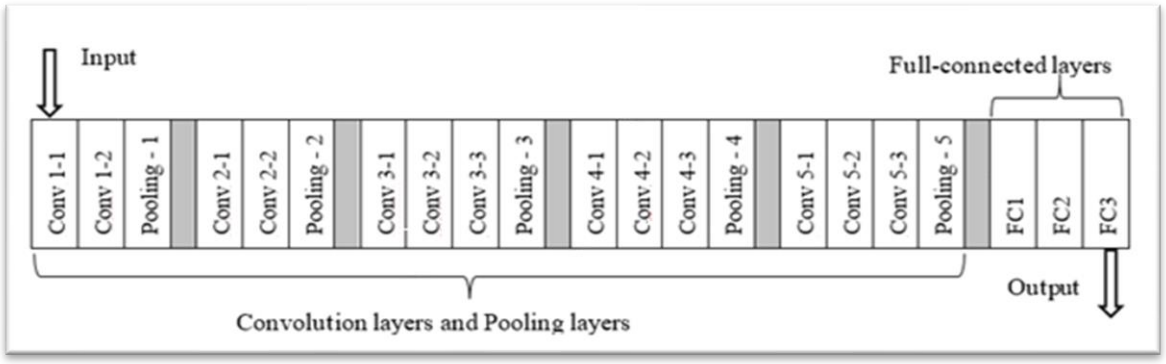


Figure 2.8:VGG16 model architecture

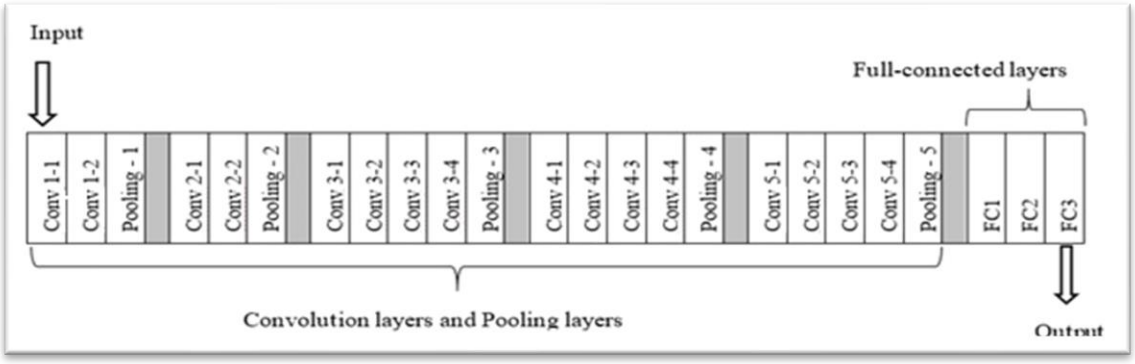


Figure 2.9: VGG19 model architecture

2.5.3.2. InceptionV3

GoogLeNet offers an alternative CNN by providing an inception module with extra layers. GoogLeNet replaces fully connected layers at the top of the convolution network with average pooling, significantly reducing the number of parameters [92]. The structure of GoogLeNet as shown in Figure 2.10, was designed to consume as little energy and memory as possible. In this study, we also examine InceptionV3, which was intended to succeed InceptionV1 and InceptionV2 [93].

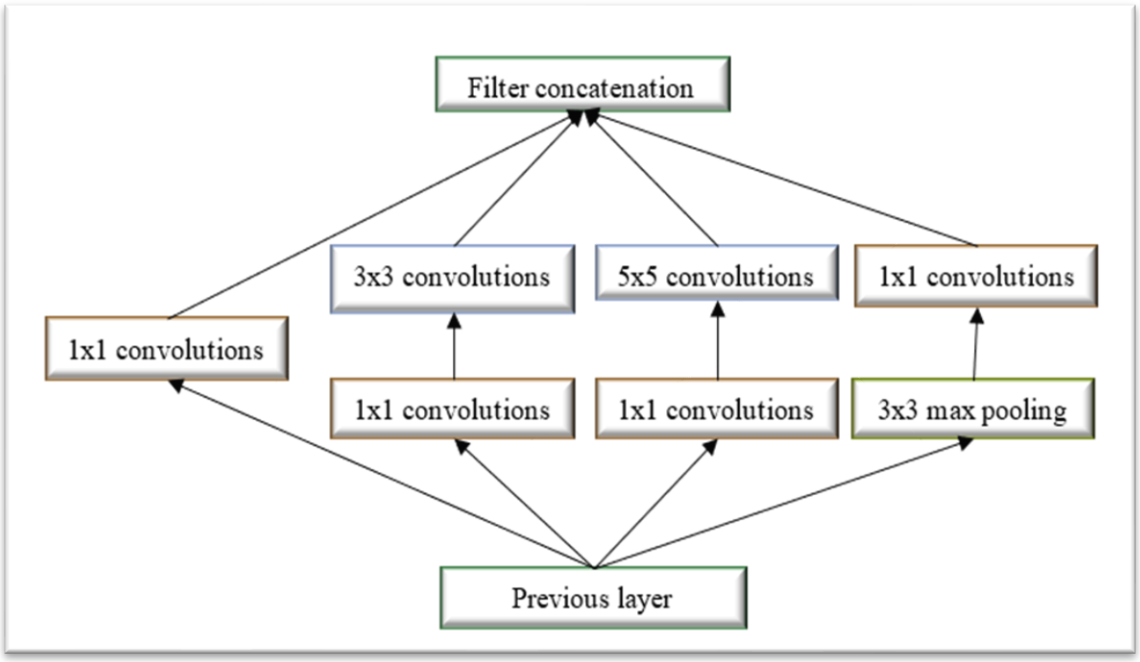


Figure 2.10:InceptionV3 model architecture

2.5.3.3. Residual Neural Network

The deep residual network with 50 layers (ResNet-50) was proposed by Ebrahimi and Abadi [94]. It is a deep CNN with fifty layers and residual blocks that serve as shortcut connections between layers. These residual connections aid in combating the problem of vanishing gradients that occur in multi-layer networks. As a result, network training and accuracy levels are improved. ResNet's architecture is shown in Figure 2.11.

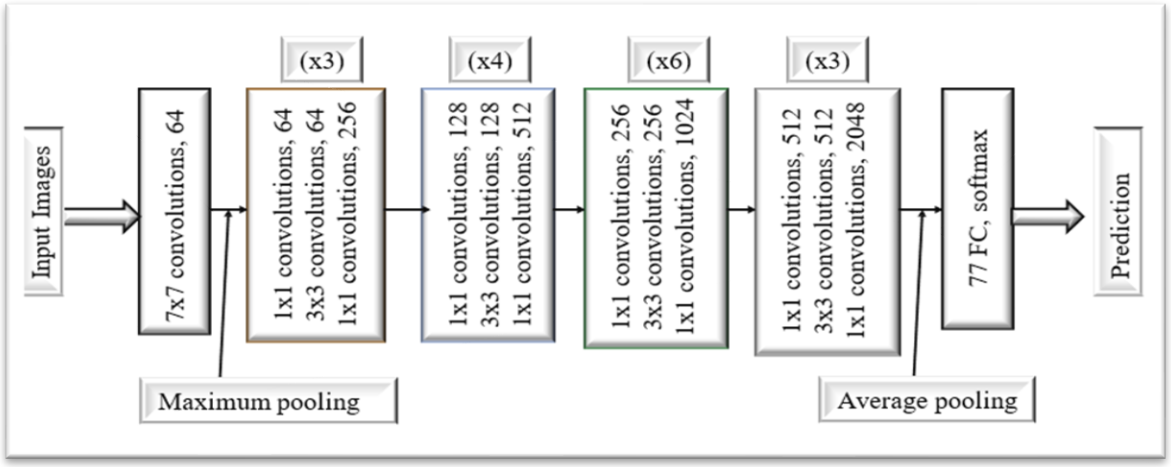


Figure 2.11: ResNet model architecture

2.5.3.4. MobileNet

MobileNet [95] is a lightweight model proposed by Google for mobile and embedded systems. Because it uses depth-wise separable convolutions, the model minimizes the number of parameters necessary to train a network. Additionally, it assists in the construction thin deep neural networks in compared to the GoogleNet and Inception V3 models, as shown in Figure 2.12.

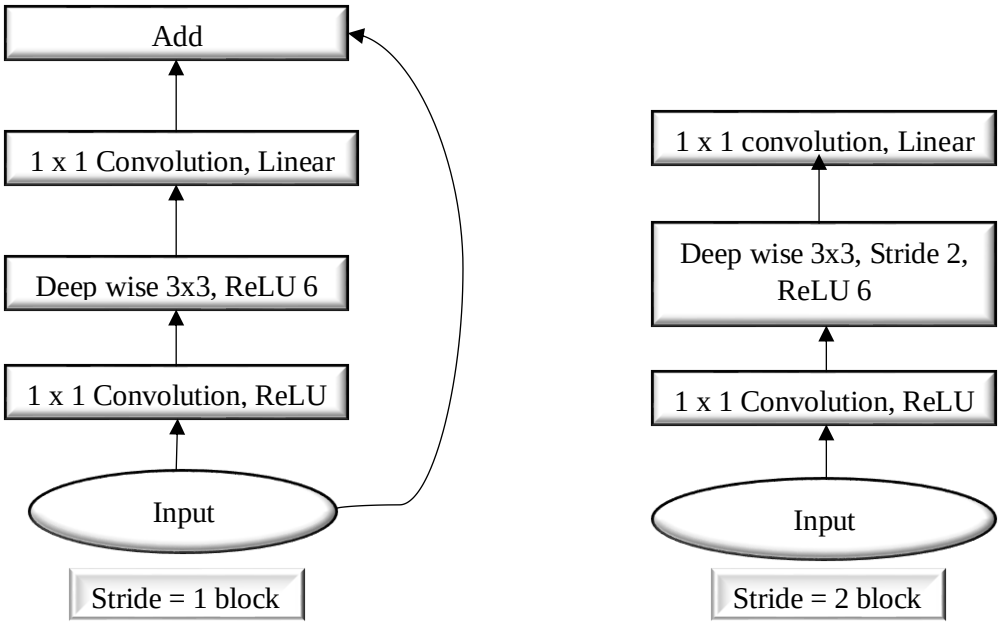


Figure 2.12: MobileNet model architecture

2.5.3.5. Extreme Inception

The extreme inception, or XceptonNet, is built on the inception principle as shown in Figure 2.13. To reduce the input data's size, Inception uses various filter types and 1x1 convolutions on the various depth spaces. When using Xception, however, the input space is compressed with 1x1 convolution across the depth before the filters are applied to the depth maps [96]. In the Inception model, both operations are followed by a ReLU non-linearity, whereas in the Xception model, there is no such non-linearity. Furthermore, despite having the same number of parameters as InceptionV3, XceptionNet outperformed InceptionV3 on the largest dataset.

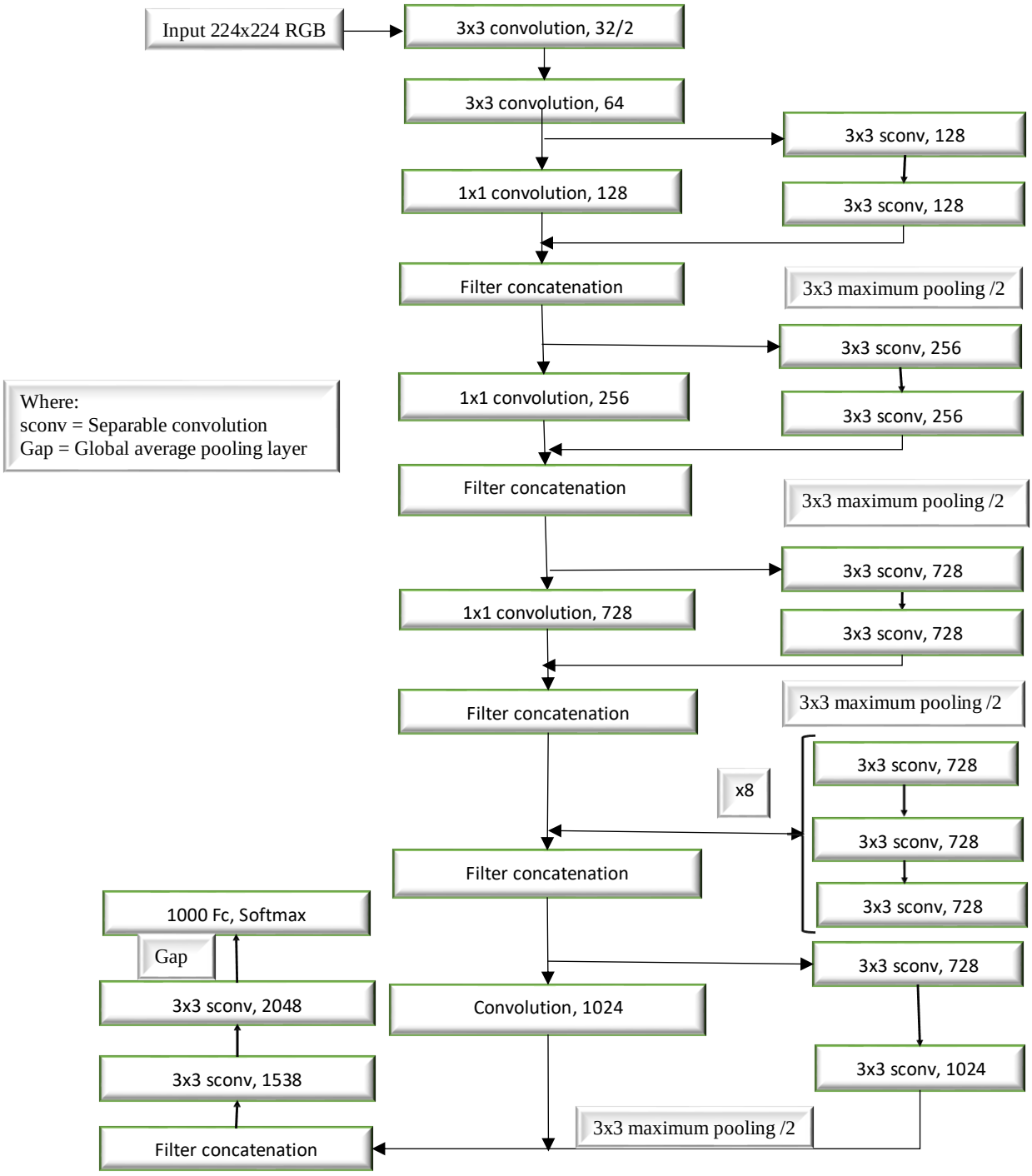


Figure 2.13: XceptonNet model archtecture

2.5.3.6. Hybrid CNN Model

This thesis proposes the hybrid CNN (HCNN) model by combining the benefits of the VGG16 model and the InceptionV3 model. HCNN has shared inputs and outputs and uses both the VGG16 and the InceptionV3 network. The HCNN model uses only the bottom layers of VGG16 and InceptionV3; these layers are then concatenated, sent through the full-connected dense layers of size 128, and finally processed using softmax to produce output in four categories. In-depth information on the design's structure is shown in Figure 2.14.

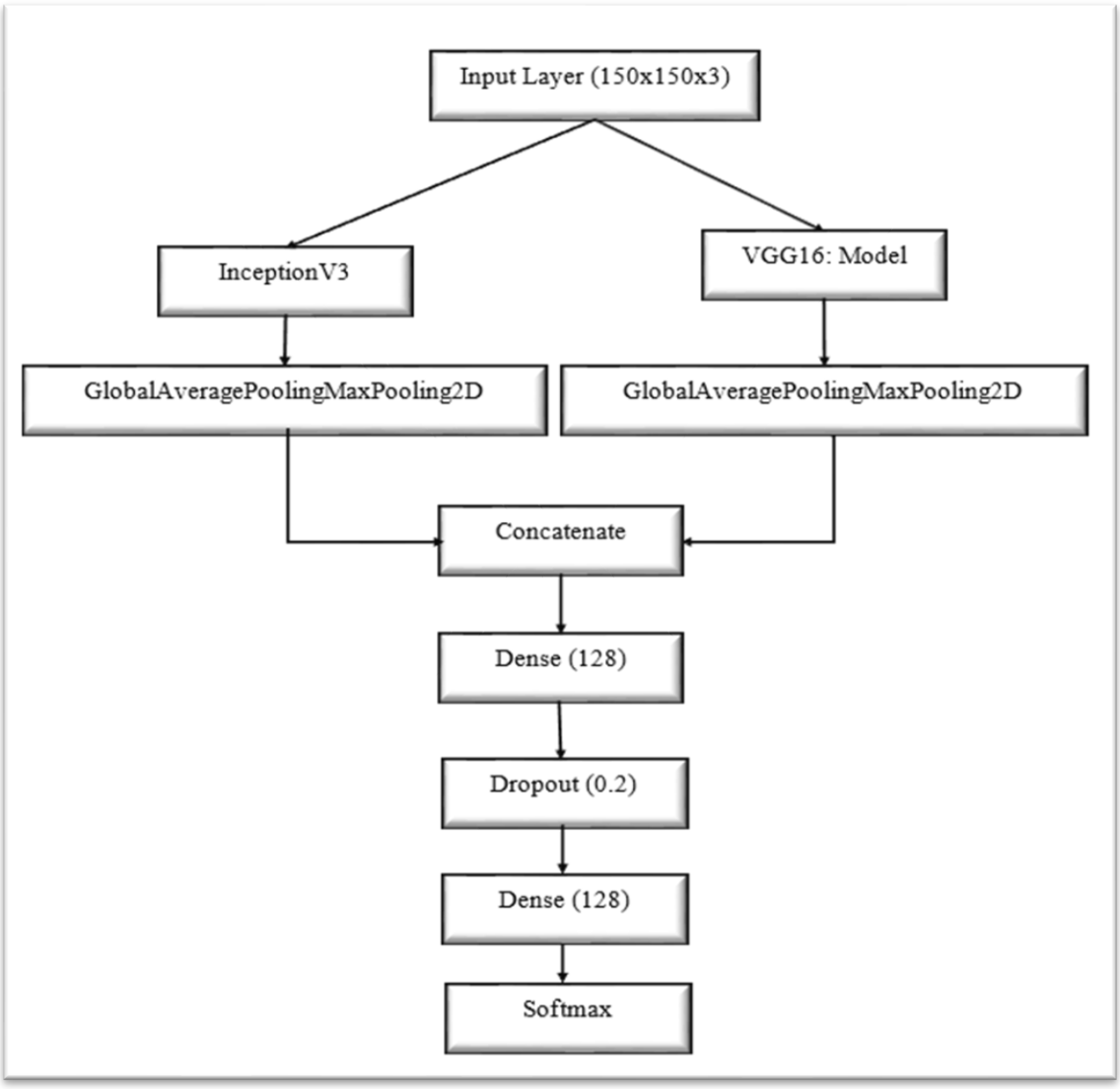


Figure 2.14: Hybrid CNN model architecture

2.5.4. Disease Detection Methods in Maize Plants

Leaf diseases in maize can present themselves in various of ways, which has led to the development of a wide range of methods for the accurate and speedy diagnosis of leaf diseases [97]. In the past, successful methods for detecting and categorizing leaf diseases have included the use of digital image processing, the IoT, SVM, and deep neural network techniques [12], [98] – [100]. When these methods are combined with other approaches like computer vision and remote sensing, the accuracy of detection is improved to identify maize plants that have been afflicted with a variety of diseases [23], [55], [70], [86].

2.5.4.1. Sensor-based Disease Detection

The IoT is a new way of thinking that has brought traditional ways of living up to the same technological level as other fields of work. Sensors can extract and gather many features from end devices and transmit them to be further processed, which significantly increases the efficiency of the system when connected devices are used in tandem.

For instance, Hanyurwimfura, D. et al. [12] proposed using an automated system based IoT to monitor and detect faw in the maize field. LoRa for communication and GPS for identifying the position of the affected maize crop were installed in the system. The different sensors in the system could determine the color and shape of the maize. Because it is expensive to install sensors on every leaf of a maize plant to detect its color and shape, the system faces various challenges during its implementation in large and medium-sized farms, respectively.

2.5.4.2. Detecting Disease with ML and Images Taken from the Ground

Ground-based digital camera images are more detailed and localized than the remote-sensing images. With the help of CV and DL, crop diseases can be quickly and accurately identified in a large volume of images [26].

For instance, to boost the precision of conventional AI techniques, Zhang et al., [101] suggested optimizing a CNN with a Multi-Activation Function (MAF) module for detecting maize leaf diseases such maculopathy, rust, and blight. The accuracy of the proposed method was 97.41% in the validation set. More so, on ResNet50, the best results were achieved by combining Sigmoid, ReLU, and Mish, which led to a 2.33%-point increase in accuracy. Likewise, Waheed et al., [102] developed an improved version of the dense CNN architecture (DenseNet) for detecting maize leaf diseases like common rust,

grey leaf spot, and northern leaf blight. Regarding trainable parameters and computation time, the proposed model was compared to other CNN-based models like EfficientNet, VGG19Net, NASNet, and XceptionNet. The optimized DenseNet model was found to be 98.6% accurate, with fewer parameters and a shorter computation time. For the diagnosis of Cercospora leaf spot, Common rust, and Northern leaf blight in maize, Panigrahi et al. proposed a Random Forest (RF) algorithm. For disease detection in maize plants, they compared the RF model to other supervised machine learning methods, including Naive Bayes (NB), Decision Tree (DT), K-Nearest Neighbor (KNN), and SVM. The RF algorithm achieved 79.23% precision [103]. Similarly, a CNN-based architecture referred to as modified LeNet was proposed by Ahila Priyadharshini et al. [98] for maize leaf disease classification. Images of maize leaves taken from the PlantVillage collection were used in the testing. In the modified version of LeNet, the images are preprocessed using Principal Component Analysis (PCA) whitening. This reduces the degree to which the features are correlated with one another, which speeds up the process of extracting features from maize leaves. With an accuracy of 97.89%, modified LeNet could distinguish between four distinct classifications of maize leaf conditions: healthy, common rust, grey leaf spot, and northern leaf blight.

Bhatt et al. [99] proposed a method for identifying maize leaf diseases by using CNN based models such as VGG16, InceptionV2, ResNet50, and MobileNetV1 and an ensemble of Adaptive Boosting cascaded with a decision tree-based classifier. The method aimed to differentiate between leaf blight and leaf spot classes of diseases, which appeared to be similar. According to the random forest test results, the accuracy provided by InceptionV2 was the highest at 98%. The image classifier separated the images into four groups: healthy, common rust, late blight, and leaf spot. Similarly, Zhang et al. [104] proposed the use of the GoogLeNet and Cifar-10 models to categorize eight diseases that can affect maize leaves: Northern leaf blight, grey leaf spot, southern leaf blight, brown spot, rust, round spot, dwarf mosaic, curvularia leaf spot, and grey leaf spot. The VGG and Alexnet models have more parameters than the proposed architecture for Google's GoogLeNet network, which has 22 layers. On the other hand, the Cifar-10 model was enhanced by adding additional layers and implementing the ReLU function. The dataset consisted of 500 images, and image augmentation was utilized to increase the dataset’s size. The improved Cifar-10 and GoogLeNet models attained accuracy levels of 98.8% and 98.9%, respectively.

Similarly, Lin et al. proposed a new multichannel CNN (MCNN). It is made up of the following fundamental layers: an input layer, five convolutional layers, three subsampling layers, three fully connected layers, and an output layer. The first and second subsampling layers are connected directly to the first fully connected layer in video saliency detection using a method that mimics human visual behavior. Furthermore, ReLU and dropout were developed and implemented to prevent overfitting and gradient diffusion. The model correctly diagnosed five diseases with 92.31%. Leaf blight, sooty blotch, brown spot, rust, and purple leaf sheaf were the diseases in question [105]. Likewise, Alehegn., [106] on the other hand, introduced KNN and artificial neural network (ANN) methods to categorize three different forms of maize leaf diseases: common rust, leaf blight, and leaf spot. The images' color, morphology, and texture features were extracted and fed into the KNN and ANN classification models. The accuracy of the proposed algorithms was measured at 94.4% for ANN and 82.5% for KNN, respectively. Similarly, Kai et al., [107] proposed using back-propagation neural networks (BPNN) to categorize three distinct maize leaf diseases. The first step of BPNN is to transform images into YCbCr format.

Finally, some texture features are derived from the grey-level co-occurrence matrix, some texture features are derived. With BPNN, we were able to achieve a 98% success rate in accuracy. This article describes how Song et al. used an SVM multi-classifier approach to classify various maize leaf diseases. They were able to get an accuracy rate of 89.6%. In addition, Sena et al. proposed a technique that employs digital photos captured by a digital camera to recognize maize plants with faw damage. They proposed an algorithm to analyze digital color images of maize plants in simplified lighting conditions and identify those that the faw had damaged. The proposed algorithm correctly labelled 94.72% of 720 [13].

2.5.4.3. Detect Disease with ML and UAV Images

Proximal sensing, which refers to images captured by UAVs close to the ground, contrasts with remote sensing, which refers to images captured by satellites in space and used to create maps of Earth's surface [15], [60], [108]. The images captured by UAVs are more detailed and compact than those captured by satellites[15]. Using machine learning methods and UAV images, numerous pieces of data can be gleaned from photos. One such example is the Resnet-34 model proposed by Wu et al. [109] which uses CNNs to detect Northern leaf blight on maize leaves in UAV images. Features were extracted using the pre-trained ResNet

model, and the results were fed into a straightforward linear classifier. The model's accuracy was 95.1%.

2.6. **Conclusion**

This chapter delves into various topics related to DA, focusing on smart farming and precision farming. Specifically, the chapter explores CV applications in agriculture, including crop health and growth monitoring, as well as weed control. Furthermore, the chapter provides a comprehensive explanation of the common terms used in ML. Lastly, the chapter explores the techniques of DNN, particularly CNN, and discusses their application in disease detection.

The subsequent chapter delves into the initial objective of identifying and recognizing the patterns arising from faw infestation on maize leaves.

Chapter 3

Identification of Maize Leaves Infected by Fall Armyworms Using UAV-based Imagery and Convolutional Neural Networks

This chapter is based on

F. S. Ishengoma, I. A. Rai, and R. N. Said, "Identification of maize leaves infected by fall armyworms using UAV-based imagery and convolutional neural networks," Computer Electronic Agriculture, vol. 184, no. October 2020, p. 106124, 2021, doi: 10.1016/j.compag.2021.106124.

3.1. Introduction

Precision farming technologies are essential for ensuring that healthy food is always available [13], [26], [98]. Farmers harvest a small amount of crops each year due to pests and diseases [98], [99], [107], [110]. The automatic detection of crop health from images helps farmers increase yield and profit while decreasing input costs and time [110], [111], [112].

The goal of this chapter is to precisely detect maize leaves infected with faw using automatic recognition algorithm models based on CNN. We remotely captured photos of maize leaves from farms using UAVs, specifically quadrotor drones. Then, four improved CNN-based models, namely VGG16 [91], VGG19 [91], InceptionV3 [93] and MobileNetV2 [110], are used to process, detect, and classify images. In all four models, we adjust the weights of the uppermost layers while holding the weights of the lower layer's constant. In our analysis, we consider two scenarios: the first is when we train the models using the original images, and the second is when we train the models with modified images to improve their detection performance. During the preprocessing stage, we apply the Shi-Tomasi corner detection technique to the original images thus result into modified images. Our results demonstrate that models trained on the modified images outperform the previously proposed methods. Specifically, we observed an improvement in accuracy of 3.92%, 6.59%, 3.25%, and 1.75% for VGG16, VGG19, InceptionV3, and MobilenetV2, respectively.

The remaining sections of this chapter are organized as follows: Section 3.2 provides a review of the materials and procedures employed in this study. Sections 3.3 and 3.4 present the experimental setup and analyze the results of the experiment, respectively. In section 3.5, we discuss the results and finally, in section 3.6 conclude the chapter.

3.2. Material and Method

3.2.1. Data Description

The study took place in Morogoro, a region located in Tanzania, East Africa. It spanned three years and was divided into three phases, as detailed in section 1.4. The initial phase of data collection involved two farms, Mkambarani (6°46'21" S 37°48'48" E) and Mikese (6°45'57" S 37°54'44" E), which together covered a land area of 3 hectares. The in-field observations were carried out between March 15 and March 21, 2020. The first five days were spent looking for fields where the maize plant shows positive signs of infection on

the leaves. The last two days were spent flying the drone over the areas three times on each field. More infections were found in the maize field at Mkambarani compared to the field at Mikese. As a result, most of the photos depicting the infected leaves come from Mkambarani. At the time that the data was being collected, the maize plant had been growing for a total of three months.

The UAV images, as shown in Figure 3.1 were captured using an RGB sensor and stored in the Joint Photographic Experts Group (JPEG) format. They have a resolution of 5472 x 3078 pixels and portray maize plants, weeds, and soil. Taken from a height of 5 meters above the ground, these photos offer a comprehensive view of the environment. To ensure comprehensive coverage, each image had a 60% overlap both in the forward and sideways directions.

The UAV images were cropped and classified into four groups that are healthy, infected, redundant and weed as shown in Figure 3.2. All the healthy images have a negative faw sign, whereas all the infected ones have a positive one as described in section 1.4. Furthermore, the redundant images consist of a maize stalk, a shadow, a tassel, and soil, while the weed images feature a variety of weeds. On the other hand, Figure 3.3 provides a clear description of the data divisions for the training, validation, and test sets.
This logical flow enables a comprehensive understanding of the image categories and the data distribution across different sets.

Figure 3.1: The UAV image

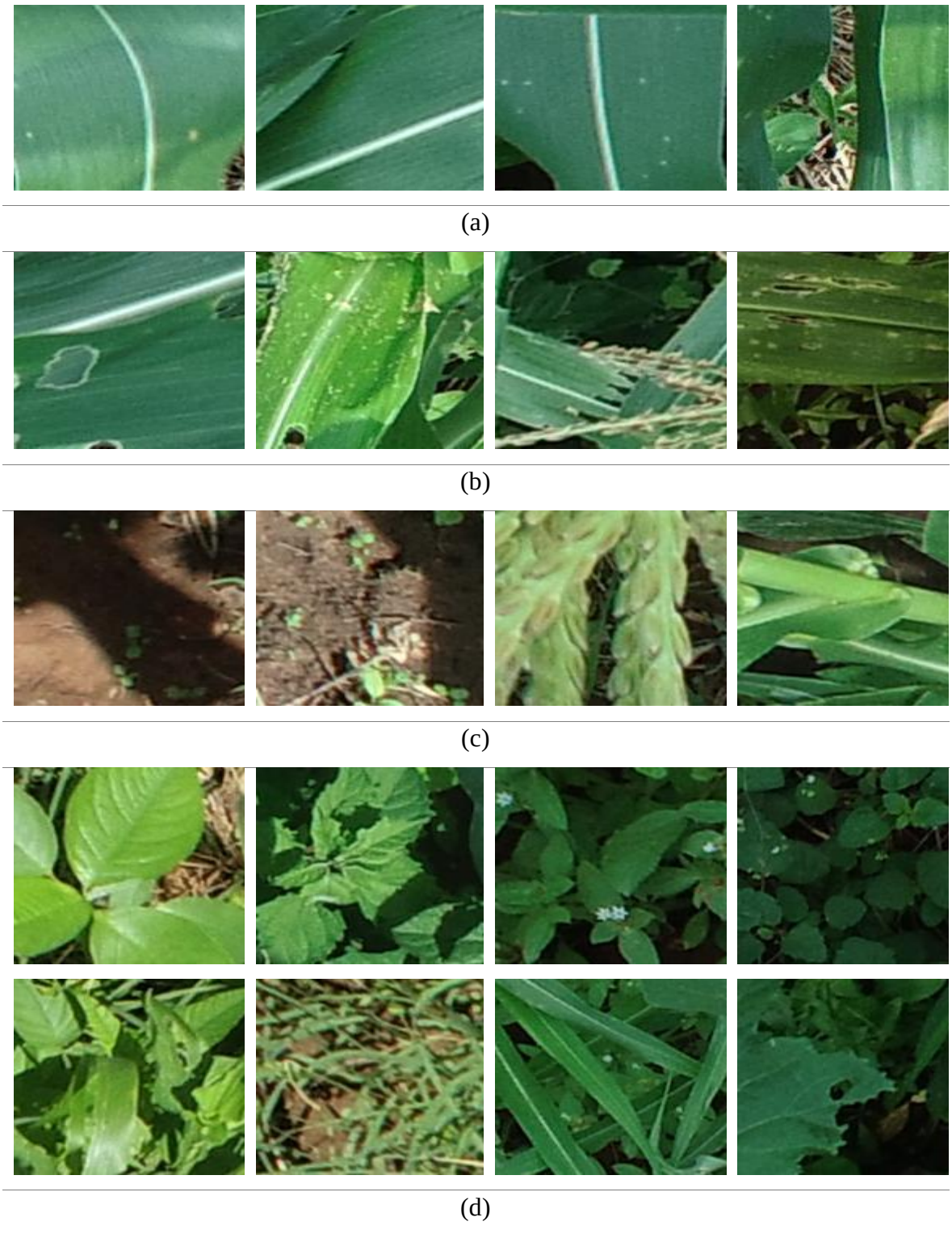


Figure 3.2: Shows the structure and nature of the dataset used in the study; a) Heathy images, b) Infected images c) Redundant images that are shadow, soil, maize tassel and maize stem d) Different types of weeds

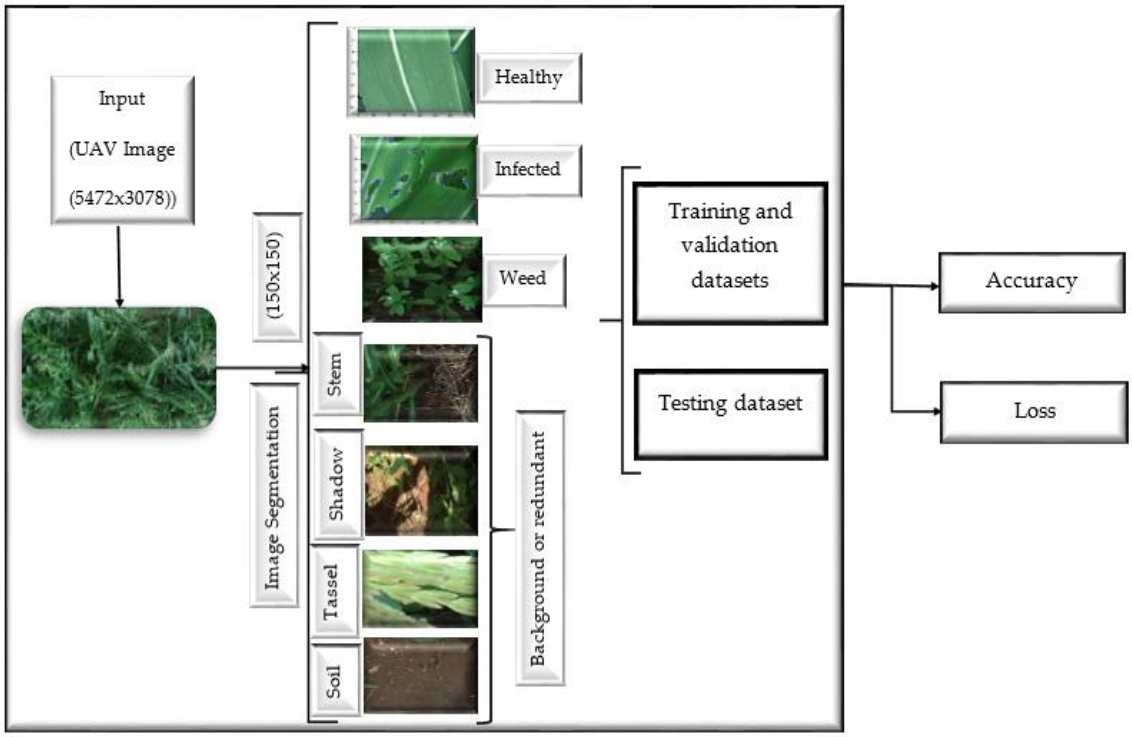


Figure 3.3: Cropped images categories

3.2.2. Shi-Tomasi Corner Detection Method

J.Shi and C.Tomasi developed a method for tracking the corner in an image [113]. The method was based on the fundamental concept that corners could be located by searching for significant variations in all directions. Open-Source Computer Vision (OpenCV) includes a function "cv.goodFeaturesToTrack()" that employs the Shi-Tomasi method (or Harris Corner Detection) to determine where the strongest corners in an image are located [113], [115]. The Shi-Tomasi method for detecting faw-caused patterns on maize leaves is summarized in Algorithm 1.

The grayscale images should be used as input, and the function should include parameters such as the number of corners, a quality level, and the shortest Euclidean distance between the detected corners. The number of corners represents the desired corners based on the nature of the image, whereas a quality level is represented by a number between 0 and 1. The quality level specifies the minimum acceptable corner quality. Any corner that does not meet this criterion will be rejected.

Furthermore, the shortest Euclidean distance between detected corners aids in avoiding the detection of nearby corners that are too close together. The function locates corners in an image by rejecting corners that do not meet the quality level specified. The remaining

corners are then sorted in decreasing order of quality. Starting with the highest quality corner, the function selects the N strongest corners. It discards any nearby corners that are closer than the minimum distance specified and returns the N strongest corners [114].

Algorithm 1: Shi-Tomasi corner detection method on maize leaves.	
Step 1:.	*Load the UAV images and resize them to a size of 450 x 300 pixels.*
Step 2:.	*Convert the resized images to grayscale (gray-image).*
Step 3:.	*Apply the corner detection method using the OpenCV function.* New_{image} *= goodFeaturesToTrack (gray-image, Ω, Ψ, Φ), where letters Ω, Ψ, and Φ, indicate the maximum number of corners to be detected and sorted, the quality level of the corners (ranging from 0 to 1), and the minimum Euclidean distance between detected corners respectively.*
Step 4:.	*Convert the resulting corner coordinates to integers by using the "numpy.int0" function.* New_{image} *= numpy.int0(*New_{image}*)*
Step 5:.	*Transform the corner coordinates into a continuous array and place circles on each corner that fulfill the specified criteria in Step 3.*
Step 6:.	*Repeat Step 5 for all the corners in the image, circling each one*

When using the Shi-Tomasi corner detection method, an estimate is made for each pixel regarding the image gradient along the x and y axes (I_x and I_y, respectively) [114] After that, the correlation matrix, denoted by the letter *(M*), for each pixel is constructed, as shown in Equation 10.

$$M = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \quad 10$$

We can improve their detection accuracy by applying a Gaussian function to the matrix's elements [114]. Equation 11 is used to estimate the response of each pixel; corners with the minimum eigenvalue below a threshold (R) are discarded; and the remaining corners are ranked from best to worst [52], [115], [116].

$$R = min(\lambda_1, \lambda_2) \quad 11$$

Eigenvalues of matrix M are denoted by λ_1, λ_2 and the threshold value R.

3.2.3. Augmentation

CNNs require a significant quantity of training data; data augmentation methods are one way to increase the size of the training dataset. Without altering the original data's structure, data augmentation creates new information. The most used augmentation

techniques are mirroring [117], rotating [118], shifting [119], and various photometric transformations [120]. A random 90º rotation, and horizontal and vertical flips, were used to enlarge the dataset in this research [114], [121]. Using these techniques, we increased the dataset size, reducing the likelihood of the model being over-fit during training [122]. Figure 3.4 (c and d) depicts the augmented method used in this study for infected and uninfected images.

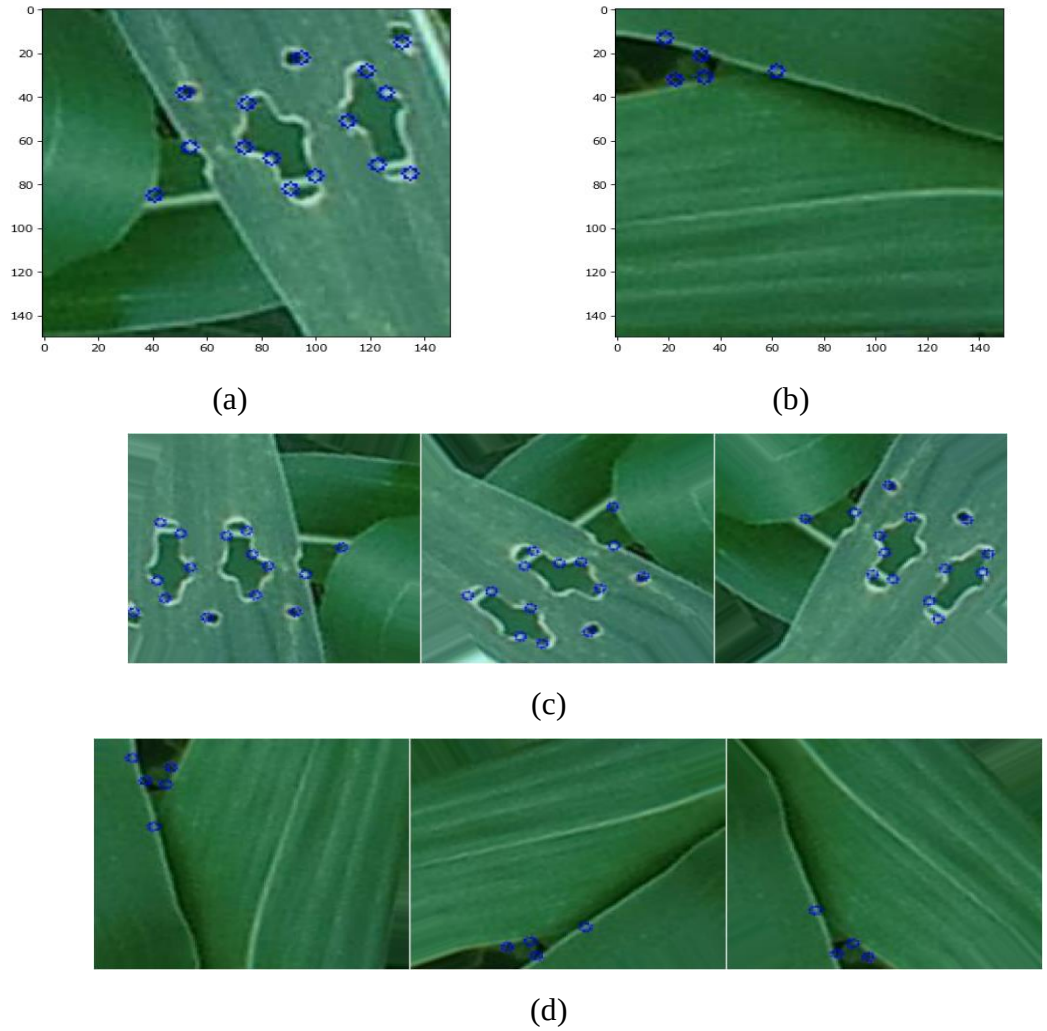


Figure 3.4. (a) Shows infected images (b) shows non-infected image (c) shows augmented infected images 90 degrees random rotation, vertical flip, and horizontal flip (d) shows augmented non-infected images 90 degrees random rotation, vertical flip, and horizontal flip.

3.3. Convolutional Neural Network

In recent years, machine learning techniques, and especially neural networks, have become increasingly popular as computing power has improved. The earliest form of neural network was ANNs, which are formed by fully connected layers in which neurons

from one layer are connected to the next, resulting in the generation of a vast amount of training data [123]. Convolution, pooling, and fully connected layers are the three main types of layers used to build a CNN (see illustration in Figure 3.5). Convolutional and pooling layers extract features, then mapped to the output by a fully connected layer [123].

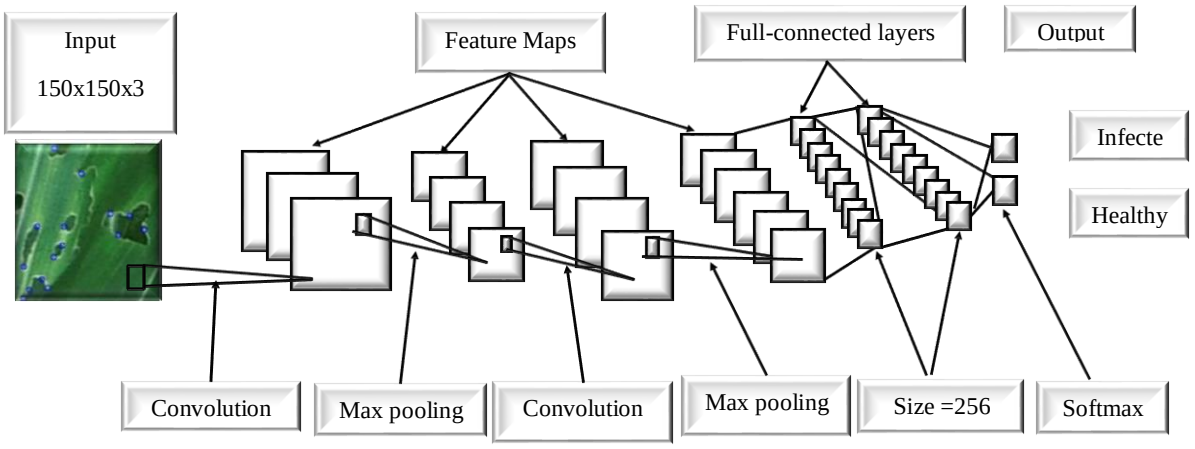


Figure 3.5. Convolution neural network architecture

3.4. Results

In this section, we describe the experimental setup, define the parameters used to evaluate the models, and then present the results.

3.4.1. Experimental Setup

The photos were obtained with the help of a quadcopter drone, the Phantom 4 pro v2, which was outfitted with a standard built-in camera with a resolution of 5472 x 3078 pixels. The lens on the camera had a field of view of 84, and its aperture was set to F/2.8 at infinity. At a height of five meters above the ground, a total of 189 photos were taken across both fields. When taking pictures, the drone was set up to stay at the same height the whole time.

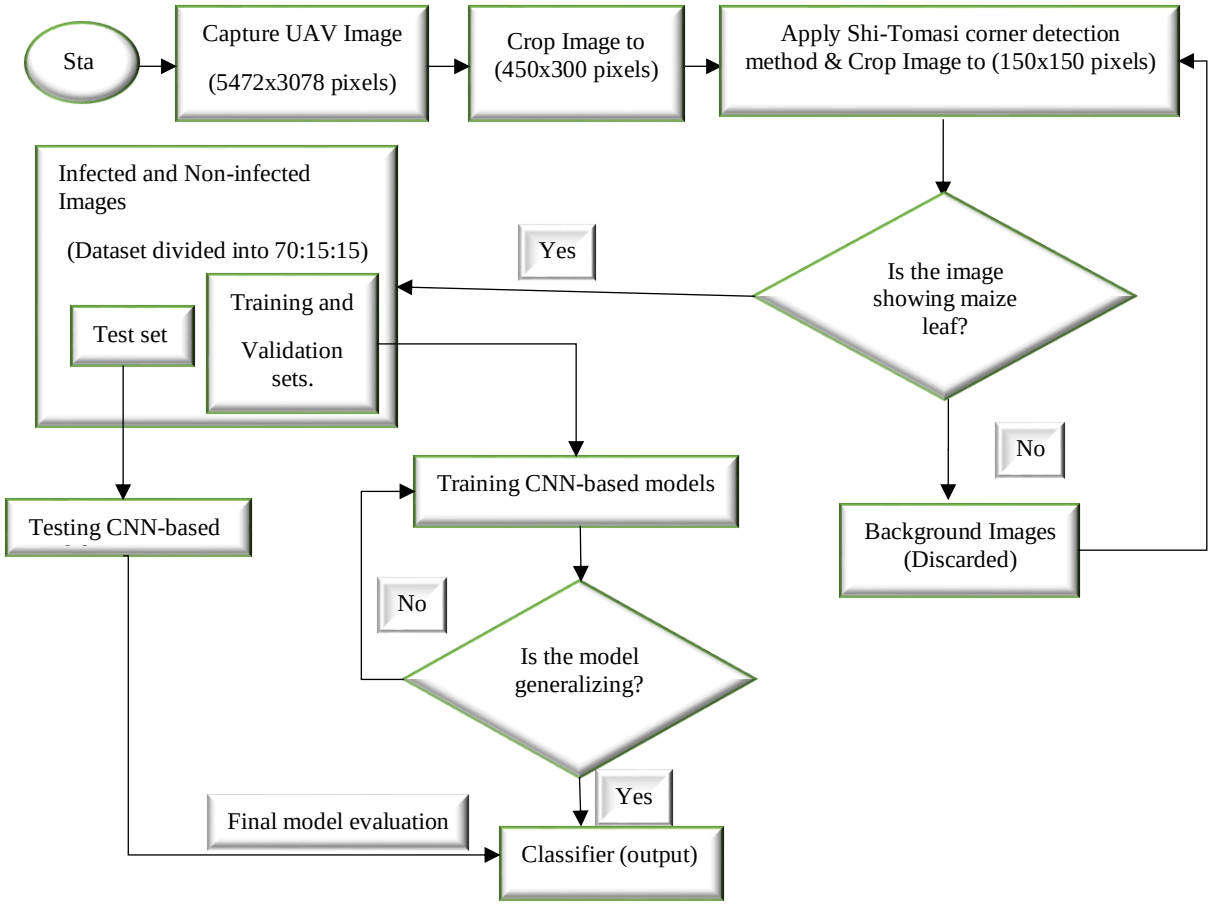


Figure 3.6: Illustration of the method used in this study.

Python codes were used to crop the collected photos down to smaller sizes, with each image now measuring 450 by 300 pixels (python v3.7 and pillow v6.0). The Shi-Tomasi corner detector method, which is available in OpenCV [113], [115] was applied to the images after they were cropped. Then the images were further cropped to have a size of 150x150 pixels so that they could be processed and classified more easily. The images were cropped automatically to have a size of 150x150 pixels and were separated into three different groups manually: infected, non-infected, and background. The infected group consisted of images that showed a positive sign of the disease, the non-infected group showed a negative sign of the disease (healthy), and the background group represented grass, the trunk, and the soil. Images from the second group were thrown out after being examined. Figure 3.6 shows the steps obtained from the images in this chapter.

Only two groups of images were considered during training, that is, infected and healthy (non-infected) were considered. The total number of images in these two groups was

11280: infected images 5640 and healthy images 5640. The dataset was divided into training (7896), validation (1692), and test sets (1692) at a ratio of 70:15:15.

The models were built with the help of the Python programming language and the open-source software framework TensorFlow 2.0. Despite being a low-level library, TensorFlow has many useful features and can be easily modified, but Python offers a wider variety of libraries and a more manageable learning curve [124]. The parameters used to determine that Stochastic Gradient Descent (SGD) was the best optimizer were a momentum of 0.9, a learning rate of 0.01, and a dropout rate of 0.2. On the other hand, intel Core i5-7200u CPU 2.50GHz 2.71GHz and 16 GB of memory were used for the training. Applying the Shi-Tomasi corner detector method to the dataset also enables the detection of additional images during the preprocessing stage [52], [115], [116]. Therefore, we used both the original and modified images with the corner detection method applied to them for model training.

Time is always a concern when training a model. It took 5 hours to prepare the MobileNetV2 model, 7 hours for the InceptionV3, 8.5 for the VGG16, and 9 for the VGG19. All four models were trained in the same amount of time regardless of whether they were trained on the original images or the modified versions. Due to the extensive use of parameters, training a model based on VGGNets takes significantly longer than training other models. To ensure accuracy, each model went through 30 iterations, with early stopping, patience, and a model checkpoint. Early stopping was used to monitor validation loss, with a patience of 25 epochs to ensure that the local minima were reached. A model checkpoint was also used to save the improved validation loss. After training the model, we compared their performance on the training and validation sets.

3.4.2. Evaluation Parameters

The classification models were evaluated using the metrics accuracy (Ac), sensitivity (Se), specificity (Sp), precision (Pr), and F_1-score (F_1). The term "accuracy" describes the degree to which a sample statistic corresponds to a parameter of the larger population [125]. Divide the total number of predictions by the proportion of correct ones to get the accuracy rate. Equation 12 represents the concept of accuracy.

$$Accuracy = \frac{TN + TP}{TN + TP + FN + FP} \quad (12)$$

In this context, "True Positive" (TP) refers to an outcome in which the model correctly predicts the positive class, "True Negative" (TN) refers to an outcome in which the model correctly predicts the negative class, "False Positive" (FP) refers to an outcome in which the model incorrectly predicts the positive class, and "False Negative" (FN) refers to an outcome in which the model incorrectly predicts the negative class. A test's sensitivity can be calculated by dividing the ratio of true positives by the sum of the true positives and the number of false negatives. It is an expression of the ability to locate all instances of relevance within a dataset [125]. The concept of sensitivity can be represented by Equation 13.

$$Sensitivity = \frac{TP}{TP + FN} \quad 13$$

"Specificity" refers to the proportion of tests with a false negative result [125]. Specificity can be represented by Equation 14.

$$Specificity = \frac{TN}{TN + FP} \quad 14$$

According to Kosmopoulos et al., [126] precision is calculated by dividing the number of true positives by the sum of the number of true positives and the number of false positives, as shown by Equation 15.

$$Precision = \frac{TP}{TP + FP} \quad 15$$

Finally, the F1-score is required when a trade-off between precision and sensitivity is desired. It's more informative about the number of misclassified cases than the Accuracy Metric [127]. F_1-score can be represented by Equation 16.

$$F_1 - Score = 2\left(\frac{Pr * Se}{Pr + Se}\right) \quad 16$$

The sensitivity measures how well a model can identify infected images, while the specificity measures how well a classifier can rule out such images. A high precision value indicates a small number of false positives when evaluating classification accuracy. Finally, a high F_1-score suggests that the model is accurate at making classifications. In a nutshell, if the values of all these performance parameters converge to one, indicating that the model has stabilized, the model is performing well.

3.4.3. Experimental Results

Figures 3.7 show the level of accuracy achieved by the models that were trained using the original images. As can be seen in Figures 3.7(a) and 3.7(b), there is not a significant difference between the accuracy of the VGG16-based model and the VGG19-based model

(b). Based on the information provided, it was determined that the average accuracy of the VGG16 model was 92.26%, while the average accuracy of the VGG19-based model was 92.32%. Furthermore, as shown in Figures 3.7(c) and 3.7(d), there was no imaginary difference between the accuracy of the InceptionV3 model and the MobileNetV2 model.

The average accuracy of the model based on InceptionV3 was computed to be 96.75%, whereas the average accuracy of the model that was based on MobileNetV2 was 97.93%. The validation curve (red line) was located relatively close to the training curve (blue line) in each of the four models. A validity curve is a measurement tool that evaluates how well the models perform.

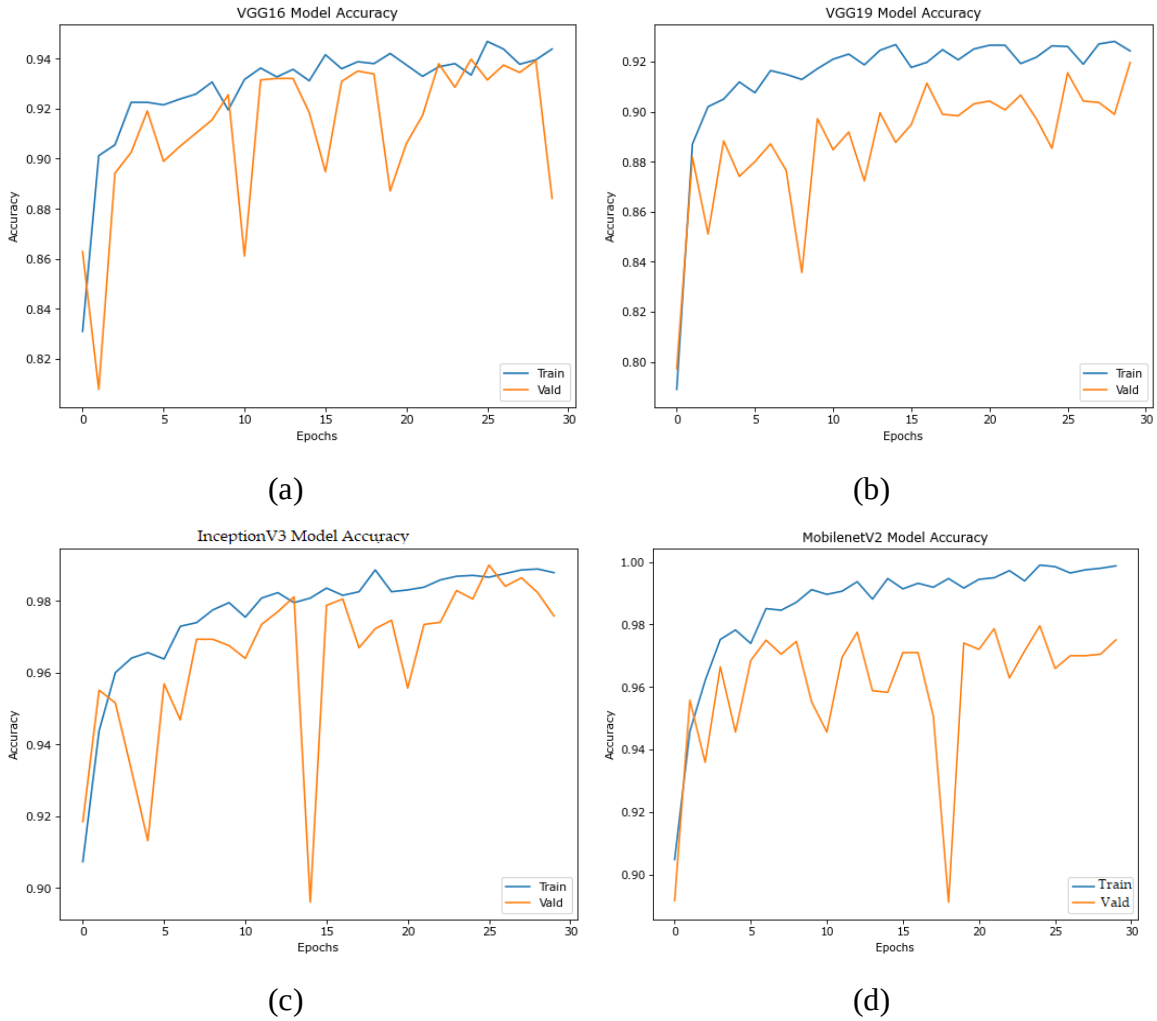
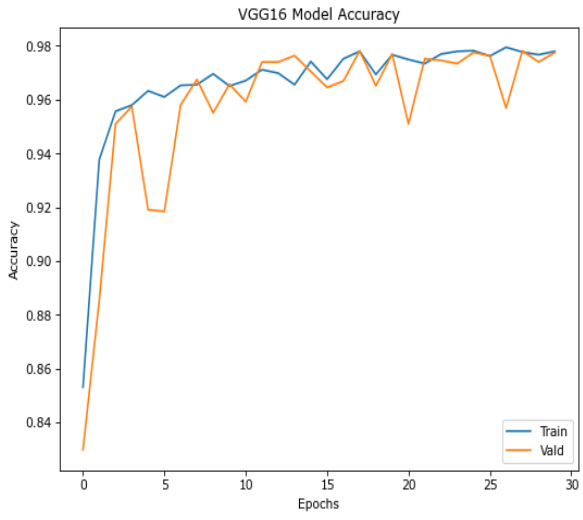


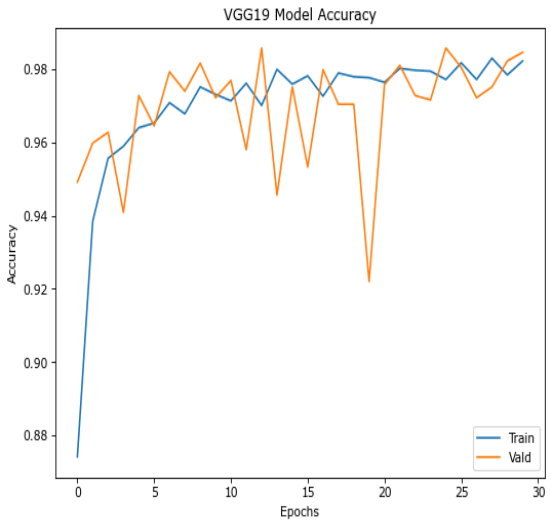
Figure 3.7. Accuracy of models trained on original images. Part (a) to (d) shows different models trained on actual images. The blue line indicates the accuracy of the training set and the red line indicate the accuracy of the validation set.

Figure 3.8 shows how accurately the models were trained on modified versions of the original images. As can be seen for the case of the original models, the accuracy values for the VGG16-based model and the VGG19-based model did not vary an excessive amount, as can be seen in Figures 3.8(a) and 3.8(b). Therefore, the VGG16-based model's average accuracy was calculated to be 98.17%, while the VGG19-based model's average accuracy was 99.00%. In addition, there was no imaginary difference between the accuracy of the InceptionV3-based model and the MobileNetV2-based model, as demonstrated in Figures 3.8(c) and 3.8(d). While the average accuracy of the MobileNetV2-based model was also 100%, the average accuracy of the InceptionV3-based model was also 100%. In light of this, there was an increase in the accuracy of the models trained on modified images compared to those that were trained with the original images.

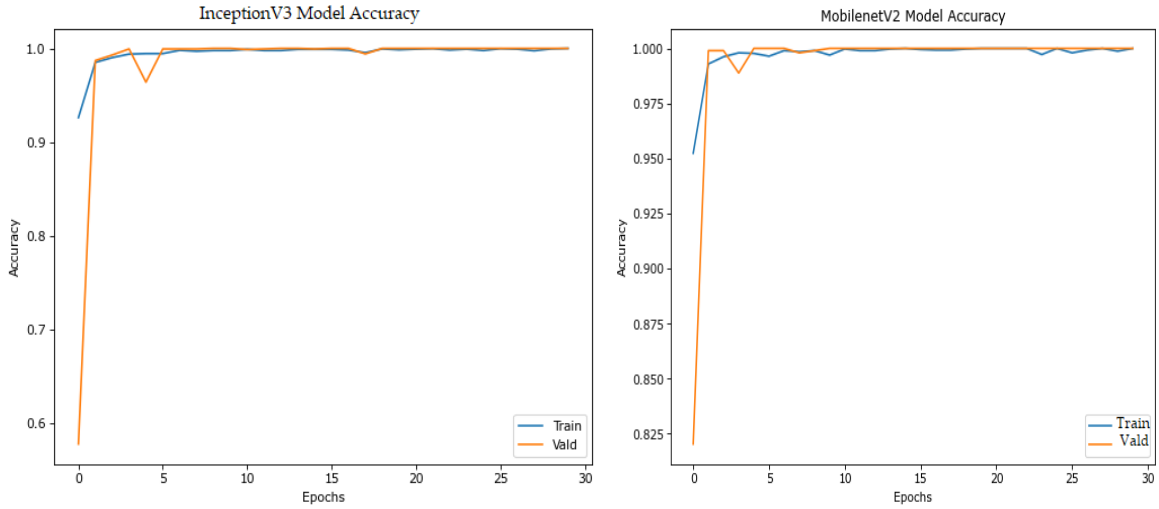
To begin, the accuracy of the model based on VGG16 improved from 92.26% to 98.17%, while the accuracy of the model that was based on VGG19 enhanced from 92.32% to 99.00%, respectively. Second, the accuracy of the model built using InceptionV3 increased from 96.75% to 100%, and the accuracy of the model built using MobileNetV2 rose from 97.93% to 100%, respectively. It is also important to note that the accuracy values reach steady-state after only a few epochs, which indicates that even though the modes were run for a considerable amount of time, it is not necessary to do so when using modified images because the Shi-Tomasi corner detection method was able to identify short holes and windowpane easily. This makes it easier for the classifier to generalize the images in a short amount of time.



(a)



(b)



(c) (d)

Figure 3.8. Accuracy of models trained on modified images. Part (a) to (d) shows different models trained on altered images. The blue line indicates the accuracy on the training set and the red line indicates the accuracy on the validation set.

Table 3.1 provides a summary of the models' performance in terms of classification. Images that had been modified were used as input images for the models labeled as VGG16-mod, VGG19-mod, InceptionV3-mod, and MobileNetV2-mod. Original versions of the images were used for the models labeled as VGG16, VGG19, InceptionV3, and MobileNetV2. As seen in the table, the accuracy of the VGG16-mod, VGG19-mod, InceptionV3-mod, and MobileNetV2-mod models is 98.17%, 99.00%, 100%, and 100%. This represents a significant improvement in performance in comparison to the models' performances when trained with the original images, which performed respectively as VGG16 (92.26%), VGG19 (92.32%), InceptionV3 (96.75%), and Mobile (97.93%). It has been discovered that the accuracy of all four models has increased, but it has been found that networks based on InceptionV3-mod and MobileNetV2-mod have achieved the best performance with the highest accuracy of (100%). It is important to point out that these models can achieve such excellent results with such a limited number of epochs because they have very few trainable parameters (See Figure 3.8).

On the other hand, the performance of VGG16, VGG19, InceptionV3, and MobileNetV2 in terms of sensitivity varied by 5%, 7%, 4% and 2% respectively, from the other four models. This shows the improvement of models trained on modified images. Similarly, way, the table shows that the performance of the models with modified images is 1.00 (InceptionV3 and MobileNetV2), 0.98 (VGG16), and 0.99 (VGG19) in terms of

sensitivity, specificity, precision, and F1-score, which is higher in every case than the values that were found for the original images. After reaching a steady state, the values for the InceptionV3 model and the MobileNetV2 model were retrieved for analysis. This again supports the claim that utilizing modified images ensures that the models will identify and categorize infected images with the highest level of precision possible.

Table 3.1: Confusion matrix

Class	Infected	Healthy	Accuracy (%)	Sensitivity	Specificity	Precision	F_1 score
VGG16							
(Infected)	837	9	92.26	0.93	0.93	0.93	0.93
(Healthy)	122	724					
VGG16-mod							
(Infected)	**828**	**18**	**98.17**	**0.98**	**0.98**	**0.98**	**0.98**
(Healthy)	**13**	**833**					
VGG19							
(Infected)	746	100	92.32	0.92	0.93	0.93	0.93
(Healthy)	30	816					
VGG19-mod							
(Infected)	**833**	**13**	**99.00**	**0.99**	**0.99**	**0.99**	**0.99**
(Healthy)	**4**	**842**					
InceptionV3							
(Infected)	841	5	96.75	0.96	0.97	0.97	0.97
(Healthy)	50	796					
InceptionV3-mod							
(Infected)	**846**	**0**	**100**	**1.00**	**1.00**	**1.00**	**1.00**
(Healthy)	**0**	**846**					
MobileNetV2							
(Infected)	827	19	97.93	0.98	0.98	0.98	0.98
(Healthy)	16	830					
MobileNetV2-mod							
(Infected)	**846**	**0**	**100**	**1.00**	**1.00**	**1.00**	**1.00**

(Healthy)	**0**	**846**

3.5. Discussion

In this chapter, we proposed using a CNN-based TL method for identifying faw-infected images. The same parameters were used to train four deep CNN-based models on both the original and altered images (VGG16, VGG19, InceptionV3, and MobileNetV2). The maize leaf images fine-tuned the models' upper layers, while the weights of the lower layers were held constant. To retrain VGG16 and VGG19, a 0.2 dropout was applied to the two fully connected layers. The final convolutional block and full-connected layer were retrained with a dropout of 0.2 for both InceptionV3 and MobileNetV2. Models trained on the altered images outperformed those trained on the original images regarding accuracy, precision, and F_1-score (see Table 3.1).

When comparing the performance of models trained on original and modified images, it is important to note that the latter shows a more erratic performance increase over time. Table 3.2 compares the accuracy performance of the models proposed in our work to that of other models already existing in the literature. The table clearly shows that the models presented in this work have better results than any other methods that have been proposed before. Applying the Shi-Tomasi corner detection method during image preprocessing made it to identify short holes and windowpane simple, allowing the proposed models to reach steady-state performance quickly. While both the existing and proposed works utilized machine-learning techniques, it is important to note that they did not share a standard simulation environment and focused on different diseases affecting maize leaves.

Table 3.2. Comparison of our proposed method with other methods.

Method		**Detection (%)**
Proposed	VGG16-mod	**98.17**
	VGG19-mod	**99.00**
	InceptionV3-mod	**100**
	MobileNetV2-mod	**100**
Existing	BPNN [107]	98
	CNN-AdaBoost [99]	98
	LeNet [98]	97.89
	ResNet-34 [63]	95.1
	SVM [111]	89.6

3.6. Conclusions

The goal of this chapter is to evaluate and compare four improved deep CNN models for classifying faw-infected images. VGG16, VGG19, InceptionV3, and MobileNetV2 models are used in this analysis. The Shi-Tomasi corner detection technique is used to enhance the 450x300 pixel maize images before analysis. This made it simple to identify small cracks and broken glass. High levels of accuracy, sensitivity, specificity, precision, and F_1-score are achieved using this method to identify the maize images. Compared to other models, InceptionV3 and MobileNetV2 performed best because they both achieved F_1-scores of one, 100% accuracy, 100% sensitivity, 100% specificity, 100% precision, and 100% F_1-scores, respectively.

The upcoming chapter introduces real-time monitoring by improving the existing models discussed in this chapter. The primary aim is to facilitate prompt and automated disease detection and weed control. To achieve this, a new image preprocessing algorithm is introduced, allowing the system to capture images in real-time without human intervention. Moreover, to minimize misclassification, two classes (weed and redundant) are included.

Farian S. Ishengoma

This page has been purposefully left blank.

Chapter 4

Hybrid Convolution Neural Network Model for a Quicker Detection of Fall Armyworms Infested Maize Plants Using UAV-Based Images

This chapter is based on

F. S. Ishengoma, I. A. Rai, and S. R. Ngoga, "Hybrid convolution neural network model for a quicker detection of infested maize plants with fall armyworms using UAV-based images," Ecol Inform, vol. 67, no. July 2021, p. 101502, 2022, doi: 10.1016/j.ecoinf.2021.101502.

4.1. Introduction

Identifying plant diseases visually across a vast region is laborious, and the results are often inaccurate because of the subjectivity of human assessments [102]. Although numerous automatic disease detection algorithms outperform visual methods in terms of detection time and accuracy, they are not yet practicable for real-time monitoring.

In this chapter, we provide a novel approach that employs a HCNN model to improve the identification of faw-infested maize leaves while minimizing misclassification in real-time environment. Our suggested method combines the use of UAV technology for capturing maize leaf photos with the HCNN model for autonomous classification.

The HCNN is built with a parallel structure that takes advantage of the strengths of both the VGG16 and InceptionV3 models. While the models from the previous chapter (VGG16 and InceptionV3) are maintained for this study, the remaining models (VGG19 and MobileNetV2) are not since VGG19's performance is comparable to VGG16 and MobileNetV2 is specifically optimized for mobile and embedded devices with limited computational resources. Furthermore, because this chapter requires real-time image identification, using a tablet or mobile phone to handle UAV images is difficult. As a result, two new models, XceptionNet and Resnet50, have been added for comparison. Moreover, to improve the precision and specificity of the infected class in real-time, we divide maize leaf UAV images into four categories healthy, infected, weed, and redundant. This method enables real-time classification of each UAV image segment, thereby reducing misclassifications between the infected, weed, and redundant classes.

The following factors are considered to implement the suggested system: One technique for improving accuracy while reducing misclassification is to increase the number of image categories. The second technique uses HCNN model with a parallel structure to shorten the training period, in which the lower-layer weights used to extract features are shared across models and only the top layers are retrained. The proposed hybrid system combines the VGG16 and InceptionV3 models' bottom layer weights and retrains the top layer weights using images of faw-infected maize leaves. VGG16 and InceptionV3 were chosen because of their higher accuracy over XceptionNet and Resnet50. In comparison to state-of-the-art approaches, the suggested HCNN model saves more time during training without reducing accuracy, according to our findings. The model improves accuracy to 96.98% while reducing training time by 16.4%. In addition, by omitting preprocessing algorithms

like Shi-Tomasi during real-time testing of the HCNN model, time is saved. This decision is particularly advantageous because the algorithm has been found to cause the misclassification of infected classes with weed and redundant classes. As a result, excluding these preprocessing steps not only improves the efficiency of the testing process but also ensures more accurate classification results.

The remaining portions of this chapter are organized as follows. Section 4.2 presents the materials and methods, while Section 4.3 discusses the experimental setting. Section 4.4 contains the experimental data and commentary, while Section 4.5 concludes the chapter.

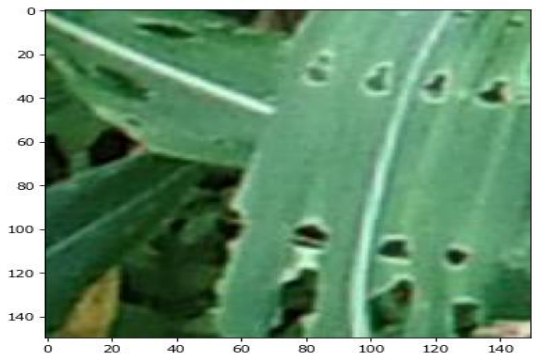
4.2. Materials and Method

4.2.1. Data Description

The training set images for the model were acquired from Mkambarani and Mikese as outlined in section 3.2.1. The second phase of data collection occurred on March 5 and 6, 2021, specifically at Mikese (6°47'7.6"S 37°54'44.3" E), covering a two-hectare area. These captured images were used for real-time system testing to ensure optimal performance. During the testing phase, the drone conducted two flights over the field, capturing a total of fifteen photos. The settings and properties of these photos were the same as those in section 3.3.1, except the overlap was adjusted to 40% to ensure a broader coverage of the area.

4.2.2. Augmentation

Data augmentation is being used in many studies because it improves learning overall [128], [129], and weed classes had the fewest images after UAV image cropping. To achieve this, we used data augmentation to supplement our original dataset images drawn from "healthy" and "redundant" classes. In Figure 4.1, twelve of the fifteen-cropped images were classified as "healthy" or "redundant." For this reason, we enhance the dataset by randomly inverting it horizontally and vertically and rotating it by 90 degrees [114], [121]. Figure 4.1(b) demonstrates how these tactics help increase the dataset's size, reducing the likelihood of model over-fitting during training [130].



(a)

(b)

Figure 4.1: (a) Indicates the original infected image (b) Indicates augmented images from the left side - horizontal flip, vertical flips, and random rotation by 90°

4.2.3. The Proposed Hybrid Convolution Neural Network

The proposed HCNN model uses a parallel setup to combine VGG16 and InceptionV3, applying the input image to both models simultaneously and then using their combined outputs to train a classifier on the full-connected layers. VGG16 and InceptionV3 were selected because of their higher accuracy compared to the XceptionNet and Resnet50 models. Figure 4.2 depicts the HCNN architecture, which involves feeding a 150x150x3 image to VGG16 and InceptionV3 models and then combining their outputs to feed into full-connected layers of size 128 before being classified into four classes by a softmax activation layer. Stochastic gradient descent (SGD) was used to retrain the fully connected layer with parameters including a dropout of 0.25, a momentum of 0.9, and a learning rate of 0.01.

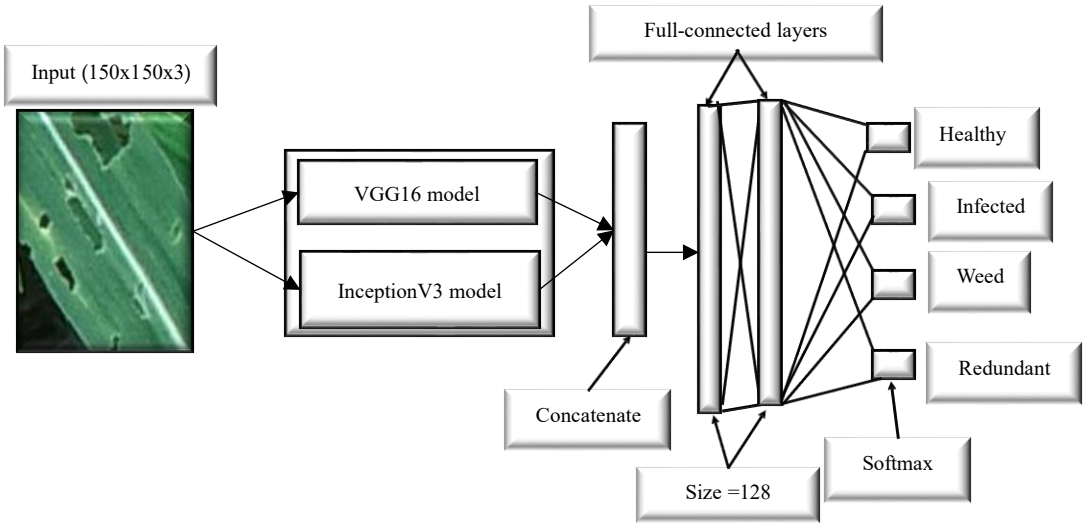


Figure 4.2: Proposed architecture

The proposed model uses a parallel structure known to classify images more quickly than a serial one [131]. The system allows the top layers to be retrained without retraining each model and uses shared weights for the lower layers to expedite feature extraction. Moreover, four distinct types of images are selected to minimize false positives and maximize precision.

4.3. Results

Within this section, we will outline the experimental configuration, establish the parameters employed for model evaluation, and subsequently showcase the outcomes.

4.3.1. *Experimental Setup*

The use of a Phantom 4 Pro v2 drone, as detailed in section 3.5.1, enabled the collection of 189 photos. Among these, 80 non-overlapping photos were chosen using a random selection method. Using Python v3.7 and pillow v6.0, the photos were cropped to 150x150 pixels. Figure 4.3 shows cropped images classified as healthy, infected, weed, and redundant. Health images depict a negative disease sign, and infected images depict a positive disease sign, weed images depict various types of weeds, and redundant images depict tassel, trunk, and soil. The redundant class was added to reduce misclassification, which is crucial for UAV image testing. The 80 images were cropped, and 10,000 images were chosen, resulting in a dataset that was split 70:15:15 into a training set (7,000),

validation set (1500), and test set (1500). The classes of training set consisted of 1750 images, whereas both the validation and test sets comprised 375 images each.

The training was done on a computer with an Intel Core i5-7200u CPU and 16 GB of memory. Training CNN models takes time; in this chapter, we trained five models in more than 30 hours. VGG16 took 8.14 hours to train, InceptionV3 9.21 hours, XceptionNet 7.39 hours, ResNet-50 6.1 hours, and HCNN 5.13 hours. To ensure accuracy, each model went through 30 iterations, with early stopping, patience, and a model checkpoint. Early stopping was used to monitor validation loss, with a patience of 15 epochs to ensure that the local minima were reached. A model checkpoint was also used to save the improved validation loss.

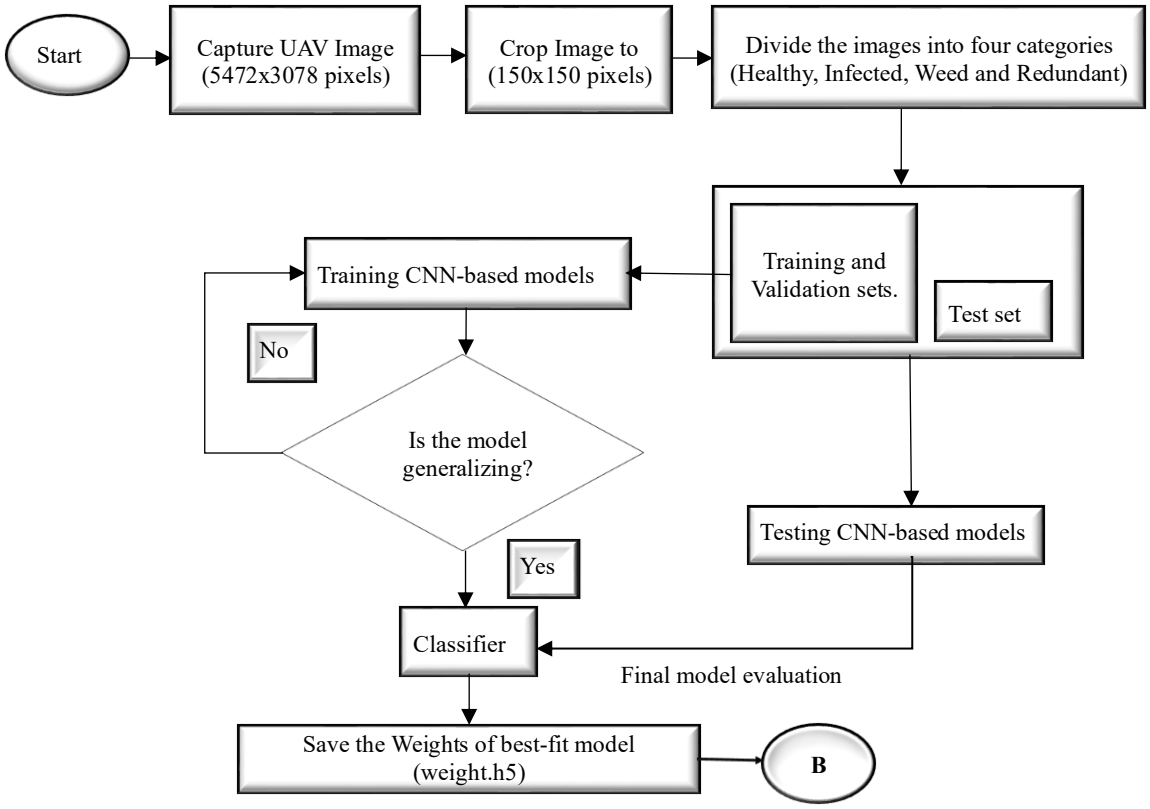


Figure 4.3: Training and testing process of the models

It should be noted that after a successful classification, the weights of the best-fit model (HCNN model) were transferred to the server denoted by B in Figure 4.3; these weights are then used for system evaluation, as shown in Figure 4.4. The framework of the proposed method, shown in Figure 4.4 below, was used to evaluate the HCNN model's efficacy. Fifteen images were taken by the UAV and uploaded to the cloud using the smartphone and a fourth-generation (4G) connection. A total of 10800 images were

obtained after being uploaded to the server and cropped to 150x150 pixels. After the images had been cropped, they were split into 60 groups and fed into the HCNN model's weights for analysis.

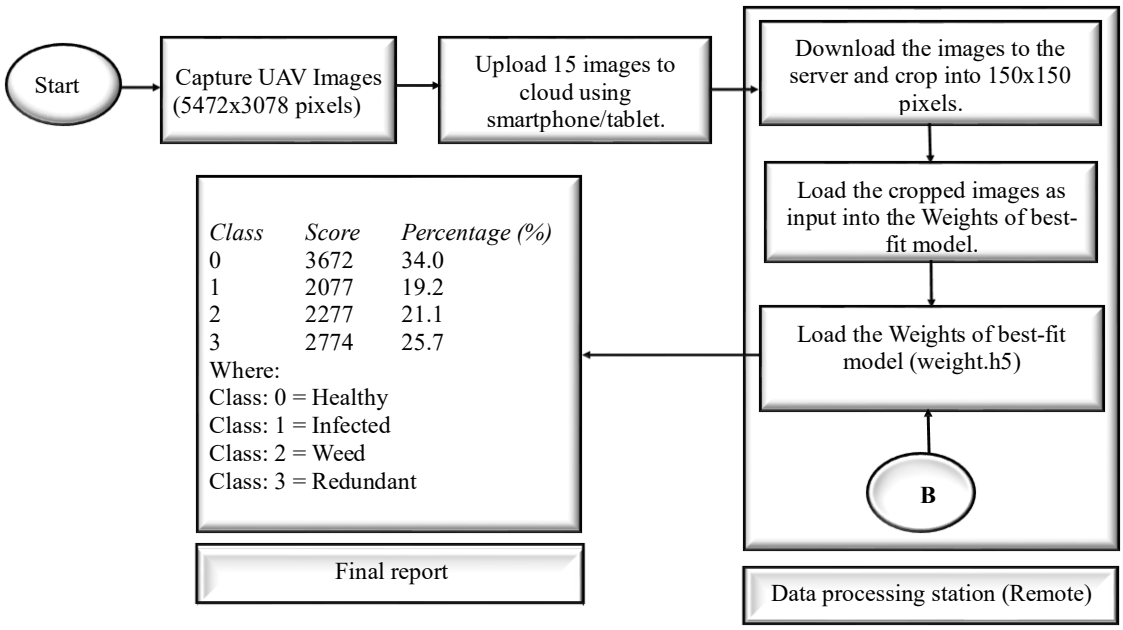


Figure 4.4: Framework of the proposed method

The duration of the test was 29.2 minutes, and the results showed that 34%, or 3672 images, were considered healthy, while 19%, or 2077 images, were considered infected, 21%, or 2277 images were considered a weed, and 25%, or 2774 images, were deemed redundant. Automatic short message service (SMS) and email delivery of the classification results to the farmer's electronic mail (e-mail).

4.3.2. Evaluation Parameters

To evaluate the classification models, we use four parameters listed in section 3.5.2 accuracy (Ac), precision (Pr), Sensitivity (Se), and F_1-score (F_1).

4.3.3. *Experimental Results and Discussion*

Figure 4.5 represents the accuracy level achieved by the HCNN model as a function of the total number of epochs. The training accuracy was calculated to be 97.29%, and the validation accuracy was estimated to be 96.67%. Because the training curve (represented by the blue line) was relatively close to the validation curve (represented by the orange line), the model did not suffer from overfitting while it was being trained.

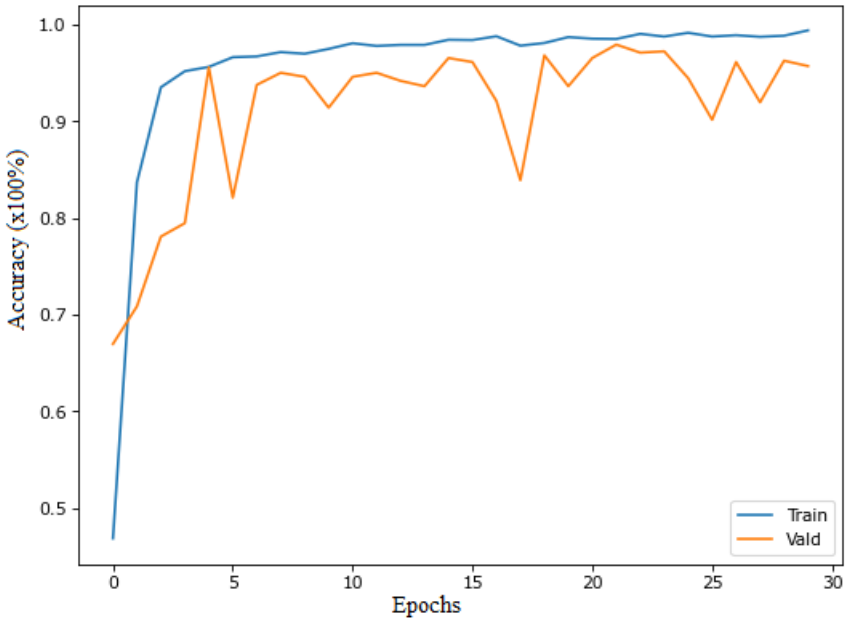


Figure 4.5: Accuracy of HCNN model.

The confusion matrix of the HCNN model, shown in Figure 4.6, is used to summarize the model's performance. The blue diagonals represent the accurate classifications, whereas the rest of the items represent incorrect classifications. The model classifier made an error in identifying three images from the healthy class, labeling 1 (0.27%) as infected and 2 (0.53%) as redundant. In addition, the classifier made an error in its classification of 23 (6.13%) of the images that belonged to the infected class, incorrectly labeling them as 21 (5.6%) healthy and 2 (0.53%) weed. Similarly, manner, the classifier got 7 (1.87%) of the images in the weed class wrong, labeling 3 (0.8%) of them as infected and 4 (1.07%) of them as redundant. Lastly, the classifier made an error in its classification of 11 (2.93%) of the images that belonged to the redundant class by labeling 4 (1.07%) as healthy, 2 (0.53%) as infected, and 5 (1.33%) as a weed.

As a result, the model can accurately identify any class with a level of certainty of at least 93.87%. In addition, the model is better at distinguishing between healthy and weed leaves. The findings, on the other hand, suggest that the possibility of incorrectly identifying infected leaves as healthy is relatively low, coming in at only 5.6%.

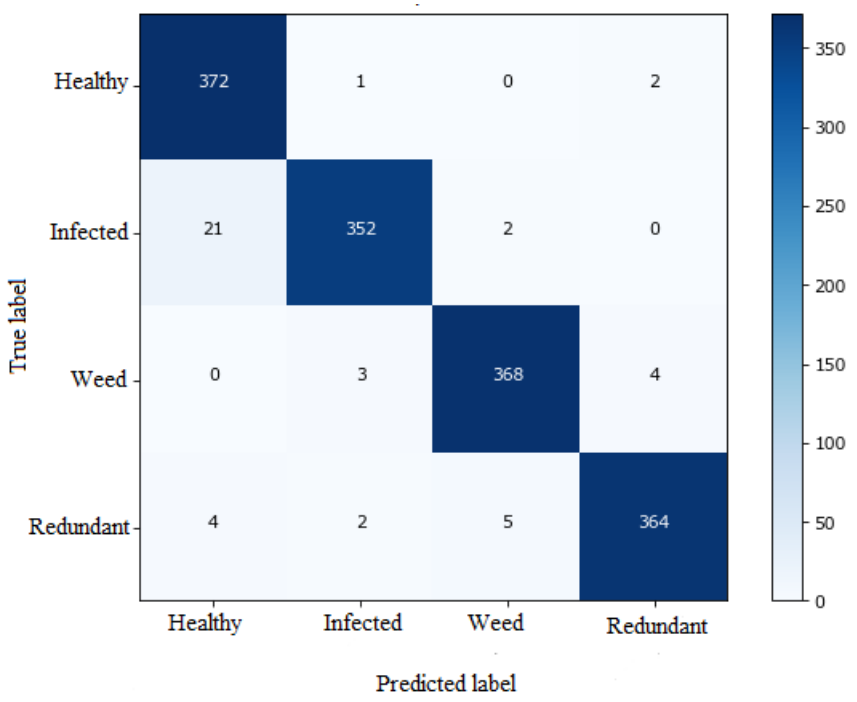


Figure 4.6: Confusion matrix of the HCNN model

The proposed hybrid model's classification report can be found in Table 4.1. It should be noted that the model has a precision of 0.98 for the infected, weed, and redundant classes, whereas the precision for the healthy class is only 0.94. In addition, the sensitivity of the model for the healthy, weed, and redundant classes are extremely close to one another, coming in at 0.99, 0.98, and 0.97, respectively; however, the sensitivity of the infected class is lower, coming in at 0.94. It is worth mentioning that the sensitivity of the infected class is 94% due to the misclassification of 21 images in that category as healthy (FN). Conversely, the precision of the healthy class is 94%, as 21 images are classified as healthy in the infected class, leading to FP.

In conclusion, the F_1-scores for the classes of healthy and infected individuals are the same, coming in at 0.96, while the F_1-scores for the classes of redundant individuals are 0.98. We can conclude that the model accurately predicts (precision) any class by a minimum of 94% and a maximum of 98% while simultaneously scoring between 94% and 99% for its true positive rate (sensitivity).

Table 4.1: Classification report

Models	Class	Accuracy (%) (Ac)	Precision (Pr)	Sensitivity (Se)	F_1-score (F_1)
HCCN	Healthy	96.98	0.94	0.99	0.96
	Infected		0.98	0.94	0.96
	Weed		0.98	0.98	0.98
	Redundant		0.98	0.97	0.98

Figure 4.7 compares the average degrees of accuracy achieved by each of the five models. The training and validation accuracies that were measured throughout all the epochs were used to calculate the models' average accuracy, which was then expressed as percentage. The illustration makes it abundantly clear that HCNN achieves the best results compared to all the other models, whereas Resnet50 achieves the lowest average accuracy. More specifically, the proposed hybrid model has the highest accuracy, clocking in at an average of 96.98%, while the average accuracies of VGG16, InceptionV3, XceptionNet, and Resnet50 are, respectively, 91%, 92.13%, 90.26%, and 89.8%. Therefore, compared to the other models considered for this study, HCNN exhibits an improvement in accuracy ranging from 4.85% to 7.18%.

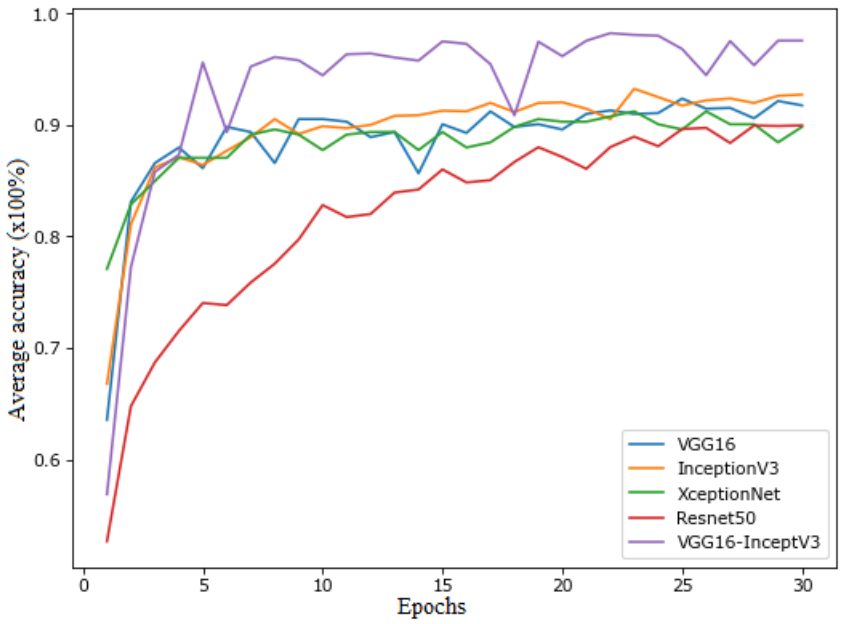


Figure 4.7: Comparison of accuracies of the models

Finally, we evaluate the proposed hybrid model against the other four models in Table 4.2 (accuracy, trainable parameters, and training time). The table shows that the proposed model has better results than the others. For example, the proposed model's trainable parameters are only 0.34 million, compared to VGG16's 8.15 million, InceptionV3's 11.39 million, XceptionNet's 7.1 million, and Resnet50's 6.1 million. This is because we only trained the top layers. The proposed HCNN model can accelerate the training process by reducing the total number of parameters used in the training process. Training time per epoch is decreased by as much as 44.3% when using the proposed model compared to VGG16, 33.3% compared to XceptionNet, and 16.0% compared to Resnet50.

Table 4.2: Comparison of the proposed model with other models

Models	Accuracy (%)	Total parameters (in millions)	Trainable parameters (In millions)	Training time (average minutes per epoch)	Training time reduction (%)
VGG16	91.0	15.78	8.15	16.28	37
InceptionV3	92.13	22.08	11.39	18.41	44.3
XceptionNet	90.26	21.14	7.1	15.30	33
Resnet50	89.8	23.86	6.1	12.20	16
Proposed	**96.98**	36.86	**0.34**	**10.25**	**-**

4.4. Conclusion

The aim of this chapter is to enhance the existing model from the previous chapter by constructing an efficient CNN model that reduces the time needed to categorize faw-infected UAV maize images while maintaining the accuracy of the model. The proposed HCNN model is based on the parallel combination of the VGG16 and InceptionV3 models, which is named as HCNN model. The hybrid model was evaluated and compared to the VGG16, InceptionV3, XceptionNet, and Resnet50 models. The experimental findings in terms of accuracy, trainable parameters, and the time required for training demonstrate that HCNN performs noticeably better than the other models examined in this research. In particular, the proposed HCNN model reduces the training time compared to all other models by 16% up to 44%, making it quicker at detecting infected maize plants with faw

while maintaining the best average accuracy of 96.98%. Furthermore, this reduction in training time occurs while the proposed HCCN model outperforms all other models in terms of accuracy, precision, and sensitivity.

In the upcoming chapter, two algorithms are presented to enhance the content covered in this chapter. The first algorithm aims to align the image contrasts of captured images from various farms with those of the trained images, thus reducing misclassification. On the other hand, the second algorithm enables the system to generate notifications displaying the infection density on each UAV image and the corresponding location of the infection.

Chapter 5

An Autonomous System for Early Detection of the Patterns Caused by Fall Armyworms on Maize Leaves in the Field

This chapter is based on

F. S. Ishengoma, I. A. Rai, and I. Gatare "Contrast Enhancement of UAV-Based Maize Plant Images for Automatic Detection of Fall Armyworm," in Data Science and Algorithms in Systems, R. Silhavy and P. Silhavy, (Eds.), Cham: Springer International Publishing, 2023, pp. 100–107, doi:10.1007/978-3-031-21438-7_8.

and

F. S. Ishengoma, I. A. Rai, and I. Gatare "Autonomous System for Locating the Maize Plant Infected by Fall Armyworm Using the UAV images" in Artificial Intelligence Application in Networks and Systems, R. Silhavy and P. Silhavy (Eds.), Cham: Springer International Publishing 2023, pp. 1–8, doi:10.1007/978-3-031-35314-7_10.

In this chapter, we propose an autonomous system based on the HCNN model presented in the previous chapter. To give the system autonomy, two distinct methods are employed. First, a contrast-enhancing method is applied, allowing the contrast of images captured in different farms or in the same farms but during different seasons to resemble the contrast of images used to train the HCNN model. These images may have been taken during the same or different seasons on the same farm. The second method is used to count the size of each infection on each UAV image and transplant the GPSc from UAV images into cropped images.

5.1. Contrast Enhancement of UAV-Based Maize Plant Images for Automatic Detection of Faw

5.1.1. Introduction

Agriculture extensively uses computer vision systems to increase yield, predict diseases, and decrease the time spent exploring farms. However, such systems necessitate extensive high-quality training data to make reliable predictions [132]. Furthermore, different image enhancement algorithms are required to improve classification accuracy on data captured in different locations, farms, or seasons due to differences in contrasts that can lead to poor performance [133]. Image *enhancement* is a method used to boost an image's quality for better viewing by humans and computers [134], [135]. On the other hand, *contrast* is the visual difference between an object and its background or other nearby objects. For this reason, increasing the contrast in an image can make it look better [136], [137]. Images with a higher contrast level have a more noticeable color scale difference than those with a lower one. Increasing the contrast of an image maximizes the effectiveness of the colors shown on screens, making details more apparent.

Due to severe weather conditions resulting in image noise, several methods have been proposed to automatically analyze such images with low contrast and scene brightness [138]. For instance, Huang et al. proposed an efficient image enhancement strategy called Contrast Limited Dynamic Quadri-Histogram Equalization (CLDQHE) that yields pleasing results while preserving brightness and structures [139]. Inspired by the illumination reflection model, Wang et al. proposed a technique that uses an adaptive local gamma transformation and color compensation. To estimate the Y component, the source image is first converted to YUV color space [140]. However, some authors have used the local gamma transform function to enhance the brightness of an image by adaptively

adjusting the parameters. For instance, Shakeri et al. proposed a contrast enhancement algorithm that uses of local histogram equalization to automatically calculate the total number of sub-histograms and divide them according to density [137].

Additionally, Gaussian Mixture Model-based Contrast Enhancement (GMMCE) was proposed by Abdoli et al. to enhance low-contrast images. This technique employs the Gaussian distribution, which stands for the picture's dominant intensity level, to handle the histogram of a low-contrast image. The overall contrast of the image is increased by boosting the sub-histogram split off from the main histogram based on the mean value of the Gaussian in the GMMCE. It has been demonstrated experimentally that GMMCE's shape-preserving method enhances image contrast [133]. As an alternative, Gu et al. proposed the Robust Image Contrast Enhancement (RICE) method, which incorporates an entire histogram modification framework and an automatic parameter selector. This framework combines the original image, the product of the histogram and the sigmoid transfer function, and the latter's visually pleasing inverted form. To solve the problems of over- and under-enhancement that plague most existing contrast enhancement techniques, they developed a saliency-preserving quality metric (QMC) for image contrast that is both efficient and effective. QMC also helps optimize the model parameters used by the RICE algorithm for maximum results [135].

Some recent research has proposed using UAVs equipped with ML algorithms to automatically detect plant diseases and present results to farmers in real-time, allowing them to take preventative measures sooner and mitigate the disease's negative effects. However, due to differences in contrast, UAV images captured from different farms cannot be used to train ML algorithms for disease detection in other farms reliably. Therefore, separately training ML algorithms for each farm is one approach to solving this issue. This is indeed the most thorough approach, but it also takes the longest. In this chapter, we propose a system that, without modifying the training set, can automatically adjust the testing set to suit any given farm's needs.

We would like to have a system that can monitor multiple farms for faw infestation in maize leaves and immediately notify the farmers of any problems it finds. To do this, use the Intensity Enhancement block to modify the HCNN model proposed by Ishengoma et al. [141], which has been shown to classify the maize leaves images captured by UAV effectively. This is done by adjusting the contrast of the cropped UAV images captured in different locations to match the average difference of the images used to train the model.

However, we believe that the contrast of the new images should be adjusted to resemble that used for training the ML model. In this study, we discourage using existing image enhancement methods because the machine learns and identifies the features based solely on the training and validation sets.

5.1.2. Proposed Solution

This section details the proposed methodology's flowchart, evaluation parameters, and algorithms. *Our solution is based on extending the previous method proposed in chapter 4.* The aim is to adjust the contrast of maize images captured in specific environments to match the contrast of healthy and infected images that were used to train the HCNN model. The model was trained to recognize four types of images: healthy, infected, weed, and redundant, but its primary goal was to identify faws-infested images.

We examined two types of images in this study, which we refer to as Category I and Category II images. The HCNN model was trained using images from Category I. These are assumed to be high-contrast images, producing the most accurate models. Category II consists of images captured from new farms or the same farm at different times of the year. It is assumed that these images have less contrast than Category I images.

Category I images were captured using a quadcopter drone Phantom 4 pro v2 between March 15 and 21, 2020, as described in sections 3.3.1. Alternatively, Category II images were obtained using a Mavic Air 2 quadcopter drone, positioned 6 meters above the ground. This phase of data collection occurred at Mikese in Morogoro, Tanzania (6°46'2" S 37°54'33" E) and covered a two-hectare farm field. The observation period spanned from May 22 to 29, 2022. The images were captured using an RGB sensor, saved in JPEG format, and had dimensions of 4000x2250 pixels. These images depict maize plants, weeds, and soil. To ensure complete coverage, each image had a 40% forward and sideways overlap, capturing all relevant details, and 40 photos were captured during the flight.

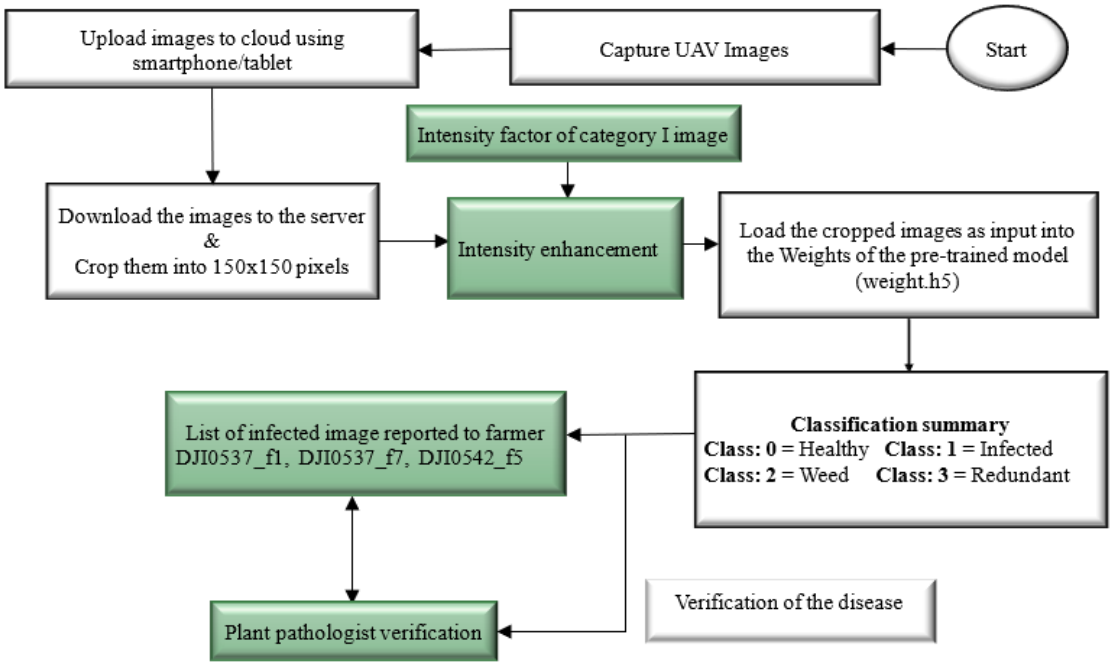


Figure 5.1: Workflow of the proposed system

The extensions of the system are shown in Figure 5.1 as green blocks. After downloading category II UAV images from the cloud, as depicted in the figure, the images are cropped to 150x150 pixels and passed through an intensity enhancement block that adjusts the contrast of the images based on the intensity values of the images. The cropped images are then loaded into the weights of the pre-trained HCNN model for classification into healthy, infected weed, and redundant. The infected images are then forwarded to the farmer and plant pathologist. The participation of a pathologist is essential for verifying and confirming the type of plant diseases.

5.1.3. Evaluation Parameter

To adjust the contrast of a given image relative to another image, we use the Intensity Factor (*IF*), defined as the ratio of total pixel intensity (t_i) values to maximum intensity (m_i) values in the image, to measure the brightness of an image. We also define the factor that balances the image's contrast, Alpha (α). Alpha is the gain that is obtained by taking a ratio of the IF values of category I images over the IF values of category II images. Finally, the percentage error (*PE*) parameter is used to determine how close the measured value is to the actual value when evaluating the category II images. In Equations 17-19, the parameters are expressed as follows:

17

$$IF = \frac{t_i}{m_i}$$

Given an image of column size w and downsize l, $m_i = 255wl$, where *w* and *l* are the widths and the length of the image respectively,

18

$$\alpha = \frac{IF(CategoryI)}{IF(CategoryII)}$$

19

$$PE = \frac{TrueScore - EstScore}{TrueScore} x100$$

values are calculated after cropping the images and categorizing them into four classes (Case I), whereas EstScore values are calculated after passing the cropped images through the HCNN model (Cases II through VII).

5.1.1. Proposed Algorithm

To use the training set captured under various environmental conditions, the contrast of the new images should be like that of the training images. As a result, whether the difference between the new images is high or low should be adjusted using the proposed *Algorithm 2*. The new images' contrast is adjusted to match the pre-trained models. To reduce misclassification, the following algorithm is proposed to improve image contrast to resemble those trained with the HCNN model. As a result, images with high contrast are highly encouraged to produce good results during the training process.

Algorithm 2: Adjusting the contrast of the captured images.	
Step 1:.	*Read the UAV images and crop them into 150 x 150 pixels.*
Step 2:.	*Compute the IF values for each cropped image (category II images).*
Step 3:.	*Calculate the alpha (α) value determining the gain to be multiplied for every cropped image.*
Step 4:.	*Process the cropped images' new contrast using the following equation.* $new_{image}(i,j) = \alpha * old_{image}(i,j) + \beta$ *, where i, j indicates that the pixels are in the i-th row and j-th column,* old_{image} *stands for the input image, beta (β) is the bias used to control the brightness of an image. In this study a beta value is maintained at zero to emphasize the importance of contrast analysis.*
Step 5:.	*Load the new images into the next block, containing the HCNN model's classification weight*

Algorithm 2 is used to change the contrast of images in Category II based on the IF values of images in Category I. Notice that the figure's intensity enhancement block needs the IF values of both the Category I and Category II images as inputs. Therefore, the proposed algorithm changes all Category II images to the IF values of Category I images that match them. These are called Cases III–VII in Table 5.2.

5.1.2. Experimental Results and Discussion

In this section, we present and discuss the classification results of Category II images after their IF values have been adjusted to match that of Category I images. Calculating the IF values for images in Categories I and II requires using all five criteria outlined in Table 5.1. The first criterion is to select the image with the smallest IF value from among all the candidates, while the second criterion chooses the image with the largest IF value. Additional criteria include averaging the IF values of the weed images, the IF values of the infected and healthy images, and the IF values of all images. The table shows that for images in Category I, the lowest and highest IF values are 0.569 and 2.23, respectively.

Similarly, all images have an average IF value of 1.175, where weed has an IF value of 0.717 and both infected and healthy have IF values of 1.017. Category II images can have an IF value between 0.133 and 1.462. The average IF value for the weed class is 0.519; for the infected and healthy classes it is 0.738, while for all images it is 0.705.

Table 5.1: Intensity factors values of images

Description	**Category I**	**Category II**
The minimum intensity factor value	0.569	0.133
The maximum intensity factor value	2.23	1.462
The average intensity factor value of weed images	0.717	0.519
The average intensity factor value of infected and healthy images	1.017	0.738
The average intensity factor value of all images	1.175	0.705

Table 5.2 summarizes seven cases obtained from Category II images. For a good comparison, we chose only 10 UAV images that show the balance between four classes out of 40 UAV images. Cropping 10 UAV images into 150x150 pixels yields 3380 Category II images.

Specifically, the Case I cropped images is divided into four groups: healthy, infected, weeds, and redundant. Images for Cases II–VII are automatically generated by the HCNN model, with Case I images serving as a baseline. As a result, only scores in Case I are accurate, while scores in Cases II through VII are estimates. The table shows that while the contrast of Category II images did not change during the experiments for cases I and II, it increased to 0.569, 2.23, 0.717, 1.017, and 1.175 when compared to the original Category I images. Furthermore, the results show that the classification outputs for the healthy and infected classes are 64.53% (2181) and 9.85% (333), respectively, corresponding to 17.1 and -57.82% PE compared to Case I images. After enhancing the Case III images to an IF value of 0.569, the classification results for the healthy and infected classes are 49.76% (1682) and 10.53% (356), respectively, corresponding to 36.07% and -68.72% PE. Increasing the IF value to 2.23 also improves the contrast of images in Case IV. In this case, the model classifies the healthy and infected groups as 35.59% (1203) and 6.86% (232), corresponding to 54.28% and -9.95% PE. Images in Case V also benefit from an improved IF of 0.717. Classifying individuals into healthy and infected groups yielded a 66.18% (2237) success rate and a 9.97% (337) success rate, or 14.98% and -59.72% PE, respectively. Classification results for the healthy and infected classes, after setting the IF value to 1.017 in Case VI, are 77.31% (2613) and 5.83% (197), respectively, corresponding to 0.683% and 6.643% PE. Case VII has an IF of 1.175 and a classification output of 74.92% (2566) for healthy images and 4.32% (146) for infected images, or 2.47 and 30.81% PE, respectively.

Table 5.2: Summary of the cases considered

Cases	**IF value**	**Classes**	**Score**	**Score (%)**	**PE (%)**
I. The output was generated after cropping the images, categorizing them into four classes and counting them.	-	Healthy	2631	86.35	-
		Infected	211	2.16	-
		Weed	40	0.91	-
		Redundant	498	10.58	-
II. The output was generated after passing the cropped images through the adopted model.	-	Healthy	2181	64.53	17.1
		Infected	333	9.85	-57.82
		Weed	130	3.85	-225
		Redundant	736	21.77	-47.79

III. The output was generated after enhancing the cropped images to the minimum intensity factor and passing them through the adopted model.	0.569	Healthy	1682	49.76	36.07
		Infected	356	10.53	-68.72
		Weed	230	6.81	-475
		Redundant	1112	32.90	-123.29
IV. The output was generated after enhancing the cropped images to the maximum intensity factor and passing them through the adopted model.	2.23	Healthy	1203	35.59	54.28
		Infected	232	6.86	-9.95
		Weed	203	6.01	-407.5
		Redundant	1742	51.54	-249.8
V. The output was generated after enhancing the cropped images to the average intensity factor of the weed class and passing them through the adopted model.	0.717	Healthy	2237	66.18	14.98
		Infected	337	9.97	-59.72
		Weed	127	3.76	-217.5
		Redundant	679	20.09	-36.35
VI. The output was generated after enhancing the cropped images to the average intensity factor of healthy and infected classes and passing them through the adopted model.	**1.017**	Healthy	**2613**	**77.31**	**0.68**
		Infected	**197**	**5.83**	**6.64**
		Weed	33	0.97	17.5
		Redundant	537	15.89	-7.83
VII. The output was generated after enhancing the cropped images to the average intensity factor of all images and passing them through the adopted model.	1.175	Healthy	2566	75.92	2.47
		Infected	146	4.32	30.81
		Weed	26	0.77	35
		Redundant	642	18.99	-28.92

Since this study aims to predict faw-infested maize images, Table 5.2 shows that the best results come from using the IF value of 1.017 with the contrast enhancement technique shown in Figure 5.1. Compared to the baseline score, the error rate for Case VI is the lowest, at 0.68%. In addition, Case VI gives good results because its IF value keeps image contrast from being too bright or dark, which helps the HCNN model reduce misclassification. On the other hand, Case III and IV images do not give good results because they make images darker or brighter, which hides the features that make the classes different. Moreover, Case V images do not work well because they are in the background and sometimes have shadows from the leaves of maize plants, so their contrast is close to that of Case III images. Finally, Case VII images are better than Cases III, IV, and V images because the average of all images IF values make the contrast better. However, compared

to Case VI, it does not work as well because when the IF values of background images like soil and weed are averaged with other images, the contrast goes down.

5.1.3. Summary of Discussion

This section suggests a way to improve images that uses different intensity factors to enhance the contrast of images taken in different environments so that they look like the ones used to train the models. The proposed technique for improving contrast cuts down on background noise, which makes it easier to classify. Since the contrast of an image depends on the camera, the environment, and the weather, any system that uses computer vision techniques should use a contrast enhancement algorithm. In this study, we took the values of 0.569, 2.23, 0.717, 1.017, and 1.175 from the corresponding Category I images to improve the contrast of Category II images taken on different farms and at other times of the year. The discrepancies created by varying these intensity factors were tested, and a healthy and infected image with an average intensity factor of 1.017 achieved high accuracy. The HCNN model was used to compare the results of classifying Category II images as healthy or infected. The healthy class got a score of 77.31% and the infected class got a score of 5.83%, corresponding to the lowest 0.68% and 6.64% PE, respectively.

5.2.**Autonomous System for Locating the Maize Plant Infected by Faw Using the UAV Images**

5.2.1.Introduction

Precision agriculture is the science of using high-tech sensors and analysis tools to improve crop yields and assist management decisions. It increases productivity, reduces operating costs, and ensures effective fertilizer and irrigation management [142]. For instance, farmers can monitor their plants' behavior at any time and from any location by employing three techniques: UAV, ML, and IoT [143]. Various methods have been proposed using these three techniques to automate the farming process and reduce farmers' costs and time.

For instance, Bhoi et al. proposed an IoT-assisted UAV-based rice pest detection model that uses the Imagga cloud to identify pests in rice during field production. Their method detected and identified the pests and sent the information to the owner for further action [143]. Moreover, Selvaraj et al. proposed a technique for classifying bananas in mixed-complex African landscapes using pixel-based classifications and ML models derived from multi-level satellite images (Sentinel 2, PlanetScope, and WorldView-2) and UAV

(MicaSense RedEdge) platforms. Their pixel-based classification from a random forest model using combined features of vegetation indices and principal component analysis achieved up to 97% overall accuracy [144]. Wu et al. proposed a method that employs two advanced deep learning algorithms: the Faster Region-based Convolutional Network (Faster RCNN) and You Only Look Once version 3. (YOLOv3). Model performance was compared using mean average precision, size, and processing speed. All models had similar precision (0.602–0.64), but the YOLO-based models were smaller and processed faster than the Faster R-CNN-adapted models. They used a UAV to collect many images from a pine tree canopy during an early stage of infection to create a training dataset [145].

Furthermore, Yadav et al. proposed a method for early detection of bacteriosis disease to reduce pesticide use and crop loss. Deep learning and an imaging method were used to develop CNN models for bacteriosis detection from peach leaf images. The proposed work compares the outcomes of imaging and CNN methods. The model architectures created using different DL algorithms performed the best, with an accuracy of 98.75% in identifying the corresponding peach leaf (bacterial and healthy) in 0.185 seconds per image [146].

Several systems have recently been developed that use UAVs, IoT devices, ML, and DL. For data collection, UAVs equipped with cameras and other IoT devices are used, while ML and DL algorithms are used to create models that can learn the behavior of data. In this chapter, we proposed a system that uses UAV and ML techniques to precisely predict faw-infested maize plants locate the position of infected maize plants and estimate the size of the infected area. The system performs three tasks: first, it crops UAV images of size 5472x3080 into 150x150 pixels and transplants the GPSc from UAV images into cropped images; second, it extracts the coordinates of the infested maize plant and counts similar GPSc that determines the size of the infected area on every UAV image; and third, it sends a report to the farmer indicating the infested plants and the size covered by infection on every UAV image.

The rest of this work is organized as follows. First, the proposed solution is presented in the following section, and the experimental results are presented and discussed in Section 5.2.3. Finally, in Section 5.2.4, we conclude the chapter.

5.2.2. Proposed Solution

This section explains the proposed method flow chart, evaluation parameters and algorithms. *Our solution is based on extension of the previous method presented in*

Chapter 4, and the green (or) shaded blocks indicate the extended solution. The study aims to design a system that uses UAV and ML techniques to predict faw-infested maize plants precisely, locate the position of infected maize plant, and estimate the size of the infected area. To achieve this, the HCNN model is employed to classify images into four distinct classes: healthy, infected, weed, and redundant. The healthy class comprises non-infected images, whereas the infected category encompasses images depicting faw-infested plants. On the other hand, the weed class identifies the various types of weeds, while the redundant class includes maize tassels, maize stems, and soil.

This study collected data using a quad copter drone, the Phantom 4 Pro v2, as described in section 3.3.6 between March 15 to 21, 2020. The captured images were uploaded to the cloud via a fourth-generation network, then downloaded and cropped into 150x150 pixels, as shown in Figure 5.2. Cropping removes GPSc from images; *Algorithm 3* is used to restore these metadata to cropped images. The cropped images are then run through the contrast enhancement block to remove background noise and reduce misclassification. The HCNN model divides these images into four categories: health, infected, weed, and redundant. Finally, the infected image lists are sent to a pathologist and a farmer. The pathologist list contains only infected images, whereas the farmer list contains the names of infected images, locations of infected images, the size of each infected UAV image, and the number of infected images on each UAV image. Pathologists' involvement is vital for validating the faw-caused patterns, identifying the disease stage, and recommending the pesticides to use. Moreover, to speed up the process, the pathologist must check two to three images to confirm the disease, as they receive only infected images.

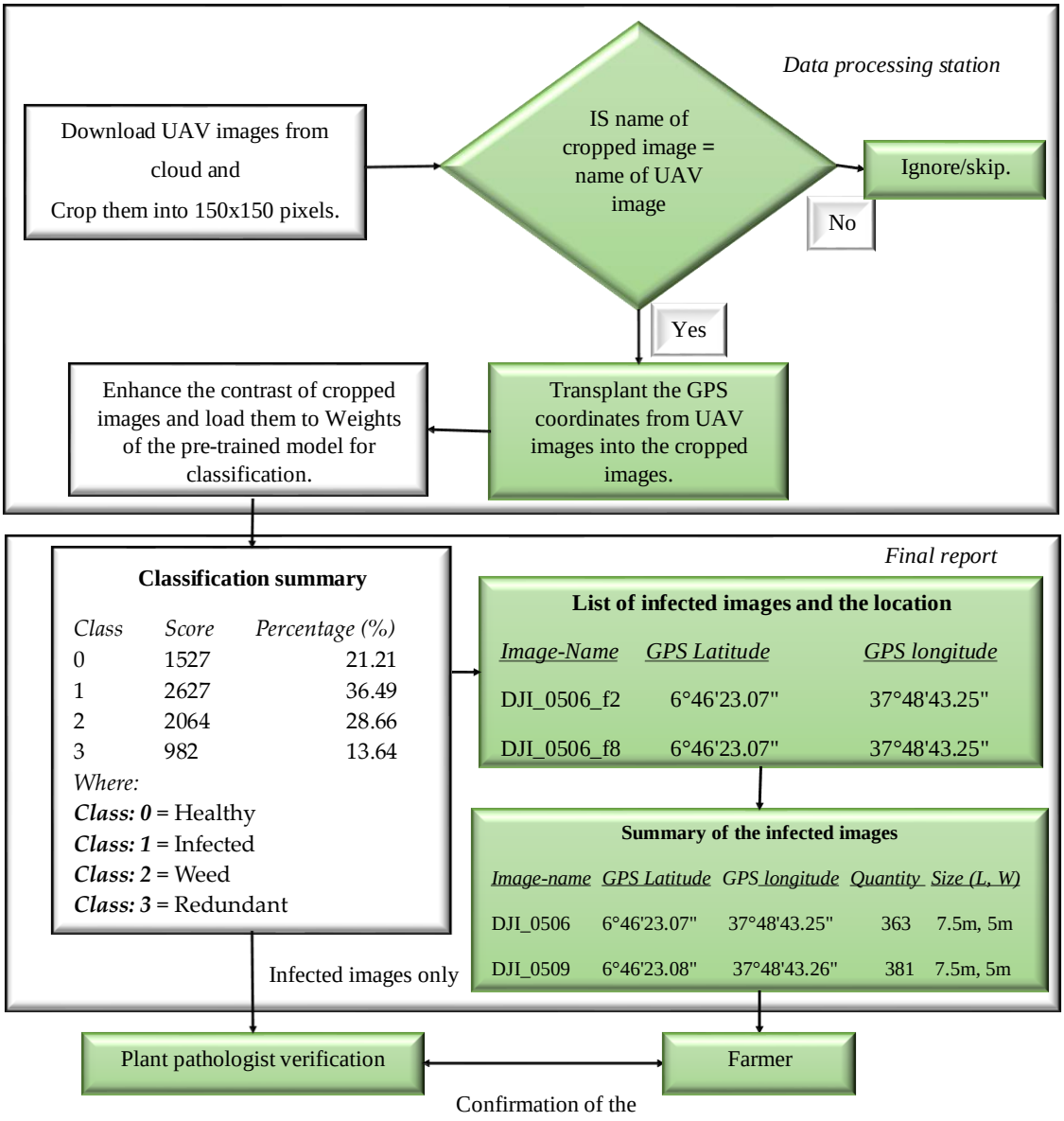


Figure 5.2: The proposed system workflow

5.2.3. Evaluation Parameters

The size of the UAV image, on the other hand, is calculated by multiplying the ground sample distance (GSD) by the width (I_w) and the length (I_l) of the image to get the actual width (W_{gnd}) and length (L_{gnd}) on the ground respectively. The GSD is the amount of surface area covered by a single image in flight. The size of the UAV image on the ground assists the farmer in identifying the surface area containing the diseases. Equations 20–22 explain how to calculate the image's GSD, the W_{gnd}, and the L_{gnd}. It should be noted that the smaller the GSD, the more precise the picture.

$$\boldsymbol{GSD}\ (\boldsymbol{m}) = \frac{\boldsymbol{S_w h_{gnd}}}{\boldsymbol{F_c I_w}} \quad (20)$$

$$\boldsymbol{W_{gnd}}\ (\boldsymbol{m}) = \boldsymbol{I_w GSD} \quad (21)$$

$$\boldsymbol{L_{gnd}}\ (\boldsymbol{m}) = \boldsymbol{I_l GSD} \quad (22)$$

The S_w is the width of the UAV sensor, while the F_c is the length of the UAV camera's focus. The h_{gnd} is the height used during image capture to represent the distance from the ground to the UAV.

5.2.4. Proposed Algorithms

This section describes the two proposed algorithms for transplanting GPSc into cropped images and counting infected cropped images. To transplant the GPSc from the UAV image into the cropped images, *Algorithm 3* is used. The system downloads the UAV images of size 5472x3080 pixels and crops them to 150x150 pixels. After cropping the system, transplant the GPSc onto the images by comparing the names of the UAV images and the cropped images, and repeat the process until all the images are GPSc-transplanted. In this experiment, one UAV image generates 720-cropped images. The images are loaded into the contrast enhancement block after the GPSc is transplanted.

Algorithm 3: Transplanting the GPS coordinates from UAV images into cropped images.

Step 1:	*Crop the downloaded UAV images to 150x150 pixels while retaining the source names in each image.*
Step 2:	*If the name of the cropped image is similar to the UAV image, transplant the coordinates from UAV image to cropped images.*
Step 3:	*Else ignore the image.*
Step 4:	*Load the images into the next block, for contrast enhancement and classification.*

Algorithm 4's goal, on the other hand, is to count the images with similar GPSc, allowing the farmer to assess the size of the problem on each UAV image. After receipt of the classification output, infected images are extracted from other images, and a list containing image names and the GPSc is generated. The algorithm detects and counts similar GPSc, then associates the total with UAV images. Finally, a report is generated with the UAV image name, GPSc, the number of infected images on each UAV image, and the UAV image size.

Algorithm 4: Counting the cropped infected images.

Step 1:	*Separate the infected class from the other classes.*
Step 2:	*Extract the images names and GPSc from the infected class.*
Step 3:	*Append the names linked with their GPSc on the new list.*
Step 4:	*Count the number of similar GPSc on the list and link the total to UAV images.*

5.2.5. Experimental Results and Discussion

This section investigates the experimental results. To evaluate the system's performance, 10 UAV images are cropped into 150x150 pixels, yielding 7200 images. Each UAV image produces 720 cropped images. The cropped images are classified by using the HCNN model which achieved an accuracy of 96.98%. According to the testing results, 21.21% of 1527 images are classified as healthy, 36.49% of 2627 images are infected, 28.66% of 2064 images are weed, and 13.64% of 982 images are redundant. The infected images are then counted using Algorithm 4 based on image name and coordinate similarity to assist the farmer in determining the magnitude of infection on each UAV image, as shown in Table 5.3.

Table 5.3 provides an overview of the farmer's report. The report includes a list of infected images with GPSc, the number of infected images on each UAV image, and the ground distance covered by each UAV image. The GPSc and ground distance covered by the UAV image help farmers apply pesticides to specific faw-infested areas more efficiently. Furthermore, the number of infected images on each UAV image directs the farmer to the areas that require more attention. The number of infected images on the DJI_0506 and DJI_0509 is 363 (50.42%) and 381 (52.92%), respectively, implying that farmers should focus their efforts on these areas because more than half of the images are infected. Moreover, the infection rate on DJI_0547, DJI_0551, DJI_0564, DJI_0567, and DJI_0579 images is 237 (32.92%), 276 (38.33%), 278 (38.61%), 272 (37.78%), and 275 (38.19%), respectively, which is considered moderate because less than half of the images are infected. On the other hand, the DJI_0532 and DJI_0576 images have lower infection rates of 170 (23.61%) and 171 (23.75%), respectively. It can also be seen that the DJI_0509 image has the highest percentage of infected rate of 52.92%, while the DJI_0532 image has the least percentage of infected rate of 23.61%. It should be noted that images were captured at a distance of 5m from the ground using a UAV equipped with an integrated camera with an 8.6mm standard F_c and a S_w of 12.83mm. As a result, using Equations 46-48, the GSD, W_{gnd}, and L_{gnd} in this study are 0.00136m, 7.5m, and 5m, respectively.

Table 5.3: Final report received by the farmer

UAV Image name	**GPSc**		**Number of infected images on every UAV image**		**Size (m)**	
	Latitude	*Longitude*	*Quantity*	*Percentage (%)*	W_{gnd}	L_{dgn}
DJI_0506	6°46'23.07"	37°48'43.25"	363	50.42	7.5	5
DJI_0509	6°46'23.08"	37°48'43.26"	381	52.92		
DJI_0532	6°46'22.61"	37°48'43.38"	170	23.61		
DJI_0537	6°46'22.64"	37°48'43.00"	204	28.33		
DJI_0547	6°46'23.03"	37°48'43.06"	237	32.92		
DJI_0551	6°46'23.20"	37°48'43.09"	276	38.33		
DJI_0564	6°46'23.57"	37°48'43.18"	278	38.61		
DJI_0567	6°46'23.42"	37°48'43.15"	272	37.78		
DJI_0576	6°46'23.34"	37°48'42.99"	171	23.75		
DJI_0579	6°46'23.02"	37°48'43.26"	275	38.19		

5.2.6. Conclusion

In this chapter, we propose an autonomous system based on the HCNN model provided in the preceding chapter. Two separate methods are used to give the system autonomy. First, a contrast-enhancing method is utilized, which allows the contrast of images collected in various farms or in the same farms but at different times of the year to resemble the contrast of images used to train the HCNN model. The discrepancies caused by modifying these intensity factors were investigated, and a healthy and diseased image with an average intensity factor of 1.017 demonstrated great accuracy.

On the other hand, the second method is used to count the size of each infection on each UAV image and transplant the GPSc from UAV images into cropped images. The proposed system uses UAV and ML techniques to predict faw-infested maize plants and sends a report to the farmer that includes the GPSc of the infested images, the number of infested images on each UAV image, and the ground size covered by each UAV image. The system crops 10 UAV images, yielding 7200 images that are then classified into four categories: healthy, infected, weed, and redundant using the HCNN model. The infected images are then extracted and counted based on GPSc similarity to determine the amount of infection on each UAV image. Identifying the number of infected images on each UAV image allows the farmer to concentrate more attention on infected areas. Moreover, because each UAV image has a different ground size, the farmer can use pesticides based on the size of the infected area. In this chapter, the report received by the farmer shows that DJI_0509 has the highest infection rate of 52.92%, while DJI_0532 has the lowest infection rate of 23.61%. The combination of methods one and two allows the system to identify images autonomously with minimal FP and FN.

In the following chapter, an overview of the general discussion, limitations of the study, directions for future research, and the overall conclusion of the thesis are presented.

Farian S. Ishengoma

This page has been purposefully left blank.

Chapter 6

Conclusion and Recommendations

This chapter summarizes the thesis, highlights the research's contribution, and suggests some future directions based on the current work.

6.1. **Summary**

When using algorithms for computer vision and machine learning to find diseases in maize plants, image analysis algorithms are improving, which lets techniques in various agricultural applications move forward. In this thesis, we made a system that uses UAV technology and machine learning to find the patterns that faw makes on maize leaves and tell the farmer what's wrong. The thesis aimed to investigate and talk about ways to find faw-infected maize leaves, automatically shows how nasty the infection is, and pinpoint their location on every UAV image. Because of this goal, the following questions came up.

1st-question. *What ML methods are used to detect the pattern caused by faw on maize leaves easily?*

The machine learning models built on CNN and trained using the methods described in Chapter 3 were able to detect the patterns accurately and effectively on maize leaves that faw caused. In addition, before cropping the images into 150x150 pixels, applying the Shi-Tomasi corner detection method on the UAV leaves helped the models generalize well, and this helped some of the models achieve accuracy, specificity, and F_1-scores of 100%.

2nd-question. *What technologies can be used to create a system capable of detecting faw patterns on maize fields and reporting them to farmers?*

The system was able to receive UAV images via the cloud, crop them to 150x150, and then use the ML algorithms to locate the infested maize images by utilizing the UAV technology, ML techniques, and cloud computing as described in Chapter 4. In addition, a hybrid CNN model was developed to enable the system to classify UAV images automatically. This model reduced the number of training parameters and decreased the computation time required. In addition, the UAV images were separated into four categories: healthy, infected, weed, and redundant. This helped cut down on the background noise and improve the classification accuracy, precision, and sensitivity.

3rd-question. *How can drone technologies be combined with IoT devices and machine learning techniques to provide an autonomous reporting system that displays infested plants, their location and the size of each UAV image covered on the ground?*

In Chapter 4, we saw how a UAV fitted with IoT sensors was used to collect data. Finally, in Chapter 5, we saw how ML algorithms were used to automatically classify images and send a report to the farmer. To achieve autonomous reporting, the system employed three algorithms: one to transplant the coordinates onto the cropped images, another to match the contrast of the captured images to the original images used to train the model, and a third to determine the area covered by each UAV image and the severity of the infection within it.

6.2. **Limitation**

Several factors constrain the methodology employed and the conclusions drawn from the analyses of the results. In the first place, hardware with more powerful processors, like GPUs, is needed for image processing. The dataset size and processing time increased when a single UAV image was cropped to 150x150 pixels, yielding 720 images.

Second, a fast speed 50-250Mbps network, like the fifth generation (5G), is required to upload and download the UAV image in the cloud with minimal latency. Delays can occur due to the use of third- or fourth-generation networks.

Third, the mobile application is the only choice for real-time image processing; however, due to the size of the UAV image after cropping, it generates many images that are too much for a smartphone to handle.

6.3. **Contribution of the Research**

The thesis' main contribution is the identification of a few patterns on maize leaves in the field using UAV and ML techniques. This major contribution is supplemented by empirical studies, which yielded a few empirical contributions summarized below.

Contribution 1. Using the Shi-Tomasi corner detection method to detect the corners of the patterns caused by faw on maize leaves. The method enabled the CNN-based algorithms that are VGG16, VGG19, MobileNetV2, and InceptionV3 to improve the detection of the patterns. Moreover, the method improved the accuracy of MobileNetV2 and InceptionV3 to reach a steady state within five iterations/epochs.

Contribution 2. Development of the hybrid CNN model helped to reduce the trainable parameters that reduced the computation time as it was necessary because we were using a

personal computer with a CPU. Moreover, to reduce misclassification the UAV images were classified into four groups to cover every part of the image, including healthy, infected, weed, and redundant.

Contribution 3. Development of the algorithm that changes the contrast of the captured images in other farms or the same farms in different seasons to resemble the images used to train the HCNN to reduce misclassification.

Contribution 4. Development of the two algorithms that are transplanting the coordinates of the UAV images into the cropped images, counting the infected images on every UAV image, and calculating the size of the infected region.

6.4. **Direction of Future Work**

During this thesis are still some issues that must be addressed. Therefore, the following sections discuss some potential future research directions.

6.4.1. Develop a System that Reduces Delay.

The delay was the main challenge in the developed system. The following can improve the

- Design and Development of a drone equipped with 5G technology that will instantly send the captured images to the edge servers for computation, reducing the delay during uploading and downloading images on the server.
- Design and develop a system that uses UAV and the DL models based on YOLO and integrate it with 5G, enabling the user in a remote area to detect diseases in real-time.

6.4.2. Develop a Model for the Detection of Several Diseases on Maize.

Classifying highly faw-infected crops like cotton, corn, millet, peanut, rice, sorghum, soybean, sugarcane, and wheat might help farmers increase productivity while recognizing infections in the early stages for future treatment.

6.5. **Conclusion**

This research uses AI algorithms and UAV techniques to present a novel CV method for identifying and classifying faw-caused patterns on maize leaves. The method was developed as part of this study. On the other hand, the AI algorithms based on CNN can classify faw-infested plants, locate the position, and indicate the size of infection on every UAV image. This was accomplished by using a UAV outfitted with IoT sensors, which sped up the data collection process in the farms.

Farian S. Ishengoma

This page has been purposefully left blank.

References

[1] OECD-FAO, *Agricultural Outlook 2021-2030*. Paris: OECD Publishing, 2021. doi: https://doi.org/10.1787/19428846-en.

[2] OECD-FAO, *Agriculture in Sub-Saharan Africa. Prospects and challenges for the next decade*. Paris: OECD Publishing, 2016. doi: 10.1787/agr_outlook-2016-en.

[3] FAO, "PART I Chapter 2 Agriculture in Sub-Saharan Africa: Prospects and challenges for the next decade," *Agricultural Outlook 2016-2025*, vol. 181, no. November 2016, pp. 59–93, 2016, doi: 10.1787/agr_outlook-2016-en.

[4] R, Day, P. Abrahams, M. Bateman, T. Beale, V. Clottey, M. Cock, Y. Colmenarez, N. Corniani, R. Early, J. Godwin, J. Gomez, G.P. Moreno, and T.S. Murphy, "Fall Armyworm Fall Armyworm: Impacts and Implications for Africa," vol. 2016, no. October 2017, doi: 10.1564/v28.

[5] S. Devi, "Fall armyworm threatens food security in southern Africa," *Lancet*, vol. 391, no. 10122, p. 727, 2018, doi: 10.1016/S0140-6736(18)30431-8.

[6] R. Day, P. Abrahams, M. Bateman, T. Beale., V. Clottey., M. Cock., Y. Colmenarez., N. Corniani., R. Early, J. Godwin, J. Gomez, G.P. Moreno, T.S. Murphy, B. Oppong-Mensah, N. Phiri, C. Pratt, S. Silvestri, and A. Witt, "Fall armyworm: Impacts and implications for Africa," *Outlooks on Pest Management*, vol. 28, no. 5, pp. 196–201, 2017, doi: 10.1564/v28_oct_02.

[7] A.P. Abrahams, M. Bateman, T. Beale, V. Clottey, M. Cock, Y. Colmenarez, N. Corniani, R. Day, R. Early, J. Godwin, J. Gomez, G.P. Moreno, T.S. Murphy, B. Oppong-mensah, N. Phiri, C. Pratt, G. Richards, S. Silvestri, and A. Witt, "Fall Armyworm: Impacts and Implications for Africa:Evidence Note (2).," *Cabi*, no. September, p. 144, 2017, doi: https://doi.org/10.1564/v28_oct_02.

[8] S. De Buck, "Maize in Africa," *International Plant Biotechnology Outreach*, vol. 6, no. 12, pp. 1–51, 2017, doi: 10.1088/0305-4470/31/34/016.

[9] CABI, "The Life Cycle of Fall Armyworm - The Plantwise Blog," *Cabi*, 2017. Availabele https://blog.plantwise.org/2017/07/17/the-life-cycle-of-fall-armyworm/ Accessed: February 8, 2023.

[10] N. C. Akeme, C. Ngosong, A.S. Sumbele, A. Aslan, S. A. Tening, Y.C. Krah, M.B. Kamanga, A. Denih, and J.O. Nambangia, "Different controlling methods of fall armyworm (Spodoptera frugiperda) in maize farms of small-scale producers in Cameroon," in *IOP Conference Series: Earth and Environmental Science*, IOP Publishing Ltd, Nov. 2021. doi: 10.1088/1755-1315/911/1/012053.

[11] T. Kasinathan and S. R. Uyyala, "Detection of fall armyworm (spodoptera frugiperda) in field crops based on mask R-CNN," *Signal Image Video Process*, 2023, doi: 10.1007/s11760-023-02485-3.

[12] D. Hanyurwimfura, E. Nizeyimana, F. Ndikumana, D. Mukanyiligira, A. Bakar Diwani, and F. Mukamanzi, "Monitoring System to Strive against Fall Armyworm in Crops Case Study: Maize in Rwanda," *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pp. 66–71, 2018, doi: 10.1109/SmartWorld.2018.00046.

[13] D. G. Sena, F. A. C. Pinto, D. M. Queiroz, and P. A. Viana, "Fall armyworm damaged maize plant identification using digital images," *Biosyst Eng*, vol. 85, no. 4, pp. 449–454, 2003, doi: 10.1016/S1537-5110(03)00098-9.

[14] L. Klerkx, E. Jakku, and P. Labarthe, "A review of social science on digital agriculture, smart farming and agriculture 4.0: New contributions and a future research agenda," *NJAS - Wageningen Journal of Life Sciences*, vol. 90–91, p. 100315, 2019, doi: 10.1016/j.njas.2019.100315.

[15] R. P. Sishodia, R. L. Ray, and S. K. Singh, "Applications of remote sensing in precision agriculture: A review," *Remote Sens (Basel)*, vol. 12, no. 19, pp. 1–31, 2020, doi: 10.3390/rs12193136.

[16] D. C. Tsouros, S. Bibi, and P. G. Sarigiannidis, "A review on UAV-based applications for precision agriculture," *Information (Switzerland)*, vol. 10, no. 11, 2019, doi: 10.3390/info10110349.

[17] G. Modica, G. De Luca, G. Messina, and S. Praticò, "Comparison and assessment of different object-based classifications using machine learning algorithms and UAVs multispectral imagery: a case study in a citrus orchard and an onion crop," *Eur J Remote Sens*, vol. 54, no. 1, pp. 431–460, 2021, doi: 10.1080/22797254.2021.1951623.

[18] H. Tian, T. Wang, Y. Liu, X. Qiao, and Y. Li, "Computer vision technology in agricultural automation, A review," *Information Processing in Agriculture*, vol. 7, no. 1, pp. 1–19, 2020, doi: 10.1016/j.inpa.2019.09.006.

[19] V. Kakani, V. H. Nguyen, B. P. Kumar, H. Kim, and V. R. Pasupuleti, "A critical review on computer vision and artificial intelligence in food industry," *J Agric Food Res*, vol. 2, no. February, p. 100033, 2020, doi: 10.1016/j.jafr.2020.100033.

[20] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep Learning for Computer Vision: A Brief Review," *Comput Intell Neurosci*, vol. 2018, 2018, doi: 10.1155/2018/7068349.

[21] J. F. S. Gomes and F. R. Leta, "Applications of computer vision techniques in the agriculture and food industry: A review," *European Food Research and Technology*, vol. 235, no. 6, pp. 989–1000, 2012, doi: 10.1007/s00217-012-1844-2.

[22] A. Dhillon and G. K. Verma, "Convolutional neural network: a review of models, methodologies and applications to object detection," *Progress in Artificial Intelligence*, vol. 9, no. 2, pp. 85–112, 2020, doi: 10.1007/s13748-019-00203-0.

[23] H. J. Yoo, "Deep Convolution Neural Networks in Computer Vision: A Review," *IEIE Transactions on Smart Processing and Computing*, vol. 4, no. 1, pp. 35–43, 2015, doi: 10.5573/ieiespc.2015.4.1.035.

[24] F. Emmert-Streib, Z. Yang, H. Feng, S. Tripathi, and M. Dehmer, "An Introductory Review of Deep Learning for Prediction Models with Big Data," *Front Artif Intell*, vol. 3, no. February, pp. 1–23, 2020, doi: 10.3389/frai.2020.00004.

[25] L, Alzubaidi., J, Zhang., J.A, Humaidi., A, Al-Dujaili., Y, Duan., O, Al-Shamma., J, Santamaría., A.M, Fadhel., M, Al-Amidie., and L, Farhan., *Review of deep learning: concepts, CNN architectures, challenges, applications, future directions*, vol. 8, no. 1. Springer International Publishing, 2021. doi: 10.1186/s40537-021-00444-8.

[26] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8689 LNCS, no. PART 1, pp. 818–833, 2014, doi: 10.1007/978-3-319-10590-1_53.

[27] M. P. Rico-Fernández, R. Rios-Cabrera, M. Castelán, H. I. Guerrero-Reyes, and A. Juarez-Maldonado, "A contextualized approach for segmentation of foliage in different crop species," *Comput Electron Agric*, vol. 156, no. November 2018, pp. 378–386, 2019, doi: 10.1016/j.compag.2018.11.033.

[28] R. Pérez-Zavala, M. Torres-Torriti, F. A. Cheein, and G. Troni, "A pattern recognition strategy for visual grape bunch detection in vineyards," *Comput Electron Agric*, vol. 151, no. August, pp. 136–149, 2018, doi: 10.1016/j.compag.2018.05.019.

[29] Y. Sun, J. Gao, K. Wang, Z. Shen, and L. Chen, "Utilization of Machine Vision to Monitor the Dynamic Responses of Rice Leaf Morphology and Colour to Nitrogen, Phosphorus, and Potassium Deficiencies," *Journal of Spectroscopy*, vol. 2018, pp. 16–18, 2018, doi: 10.1155/2018/1469314.

[30] F. Fahmi, D. Trianda, U. Andayani, and B. Siregar, "Image processing analysis of geospatial uav orthophotos for palm oil plantation monitoring," *J Phys Conf Ser*, vol. 978, no. 1, 2018, doi: 10.1088/1742-6596/978/1/012064.

[31] H. Liu and J. S. Chahl, "A multispectral machine vision system for invertebrate detection on green leaves," *Comput Electron Agric*, vol. 150, no. April, pp. 279–288, 2018, doi: 10.1016/j.compag.2018.05.002.

[32] Y. Zhong, J. Gao, Q. Lei, and Y. Zhou, "A vision-based counting and recognition system for flying insects in intelligent agriculture," *Sensors (Switzerland)*, vol. 18, no. 5, 2018, doi: 10.3390/s18051489.

[33] M. Toseef and M. J. Khan, "An intelligent mobile application for diagnosis of crop diseases in Pakistan using fuzzy inference system," *Comput Electron Agric*, vol. 153, no. July, pp. 1–11, 2018, doi: 10.1016/j.compag.2018.07.034.

[34] S. Sabzi, Y. Abbaspour-Gilandeh, and G. García-Mateos, "A fast and accurate expert system for weed identification in potato crops using metaheuristic algorithms," *Comput Ind*, vol. 98, pp. 80–89, 2018, doi: 10.1016/j.compind.2018.03.001.

[35] C. L. Chang and K. M. Lin, "Smart agricultural machine with a computer vision-based weeding and variable-rate irrigation scheme," *Robotics*, vol. 7, no. 3, 2018, doi: 10.3390/robotics7030038.

[36] H. Tian, T. Wang, Y. Liu, X. Qiao, and Y. Li, "Computer vision technology in agricultural automation —A review," *Information Processing in Agriculture*, vol. 7, no. 1, pp. 1–19, 2020, doi: 10.1016/j.inpa.2019.09.006.

[37] I. B. Yonah, S. K. Mourice, S. D. Tumbo, B. P. Mbilinyi, and J. Dempewolf, "Unmanned aerial vehicle-based remote sensing in monitoring smallholder, heterogeneous crop fields in Tanzania," *Int J Remote Sens*, vol. 39, no. 15–16, pp. 5453–5471, 2018, doi: 10.1080/01431161.2018.1455241.

[38] P. J. Wójtowicz M., Wójtowicz A., "Application of remote sensing methods in agriculture," pp. 31–50, 2016.

[39] L. F. P. Oliveira, A. P. Moreira, and M. F. Silva, "Advances in agriculture robotics: A state-of-the-art review and challenges ahead," *Robotics*, vol. 10, no. 2, pp. 1–31, 2021, doi: 10.3390/robotics10020052.

[40] S. Zheng, Z. Wang, and C. J. Wachenheim, "Technology adoption among farmers in Jilin Province, China: The case of aerial pesticide application," *China Agricultural Economic Review*, vol. 11, no. 1, pp. 206–216, 2019, doi: 10.1108/CAER-11-2017-0216.

[41] L, Han, G, Yang, H, Dai, B, Xu, H, Yang, H, Feng, Z, Li, and X, Yang, "Modeling maize above-ground biomass based on machine learning approaches using UAV remote-sensing data," *Plant Methods*, vol. 15, no. 1, pp. 1–19, 2019, doi: 10.1186/s13007-019-0394-z.

[42] Y. Niu, L. Zhang, H. Zhang, W. Han, and X. Peng, "Estimating above-ground biomass of maize using features derived from UAV-based RGB imagery," *Remote Sens (Basel)*, vol. 11, no. 11, pp. 1–21, 2019, doi: 10.3390/rs11111261.

[43] C. H. W. de Souza, R. A. C. Lamparelli, J. V. Rocha, and P. S. G. Magalhães, "Mapping skips in sugarcane fields using object-based analysis of unmanned aerial vehicle (UAV) images," *Comput Electron Agric*, vol. 143, no. August, pp. 49–56, 2017, doi: 10.1016/j.compag.2017.10.006.

[44] X. Shi, W. Han, T. Zhao, and J. Tang, "Decision Support System for Variable Rate Irrigation Based on UAV Multispectral Remote Sensing," *Sensors (Switzerland)*, vol. 19, no. 13, 2019.

[45] E. R. Hunt and C. S. T. Daughtry, "What good are unmanned aircraft systems for agricultural remote sensing and precision agriculture?" *Int J Remote Sens*, vol. 39, no. 15–16, pp. 5345–5376, 2018, doi: 10.1080/01431161.2017.1410300.

[46] C. Djellali, M. Adda, and M. T. Moutacalli, "A data-driven deep learning model to pattern recognition for medical diagnosis, by using model aggregation and model selection," in *Procedia Computer Science*, Elsevier B.V., 2020, pp. 387–395. doi: 10.1016/j.procs.2020.10.052.

[47] J. Serey, M. Alfaro, G. Fuertes, M. Vargas, C. Durán, R. Ternero, R. Rivera, and J. Sabattin, “Pattern Recognition and Deep Learning Technologies, Enablers of Industry 4.0, and Their Role in Engineering Research,” *Symmetry*, vol. 15, no. 2. MDPI, Feb. 01, 2023. doi: 10.3390/sym15020535.

[48] G. A. Fink, *Markov models for pattern recognition: from theory to applications*. Springer-Verlag Berlin Heidelberg, Dortmund, Germany, 2008.

[49] M. Jordan, J. Kleinberg, B. Schölkopf, and C. M. Bishop, "Pattern Recognition and Machine Learning," Cambridge, U.K.: Microsoft Research Ltd, 2008.

[50] H. Singh, *Practical Machine Learning, and Image Processing*. Apress, 2019. doi: 10.1007/978-1-4842-4149-3.

[51] S. Lai, I. Zayas, and Y. Pomeraz, “Application of Pattern Recognition Techniques in the Anlaysis of Cereal Grains”.

[52] H. Singh, *Practical Machine Learning, and Image Processing*. 2019. doi: 10.1007/978-1-4842-4149-3.

[53] J. Chaki and N. Dey, "A Beginner’s Guide to Image Preprocessing Techniques," 1st ed., Boca Raton: CRC Press, Nov. 16, 2018. [Online]. Available: https://doi.org/10.1201/9780429441134. [Accessed: 20 January 2023].

[54] P.L. Osco, J. Marcato Junior, P.A. Marques Ramos, A.L. de Castro Jorge, N.S, Fatholahi, J. de Andrade Silva, T.E. Matsubara, H. Pistori, N.W. Gonçalves, and J. Li, “A review on deep learning in UAV remote sensing,” *International Journal of Applied Earth Observation and Geoinformation*, vol. 102, 2021, doi: 10.1016/j.jag.2021.102456.

[55] T. Kattenborn, J. Leitloff, F. Schiefer, and S. Hinz, “Review on Convolutional Neural Networks (CNN) in vegetation remote sensing,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 173, no. December 2020, pp. 24–49, 2021, doi: 10.1016/j.isprsjprs.2020.12.010.

[56] M. Thoma, “A Survey of Semantic Segmentation,” pp. 1–16, 2016, http://arxiv.org/abs/1602.06541.

[57] X. Wu, D. Sahoo, and S. C. H. Hoi, “Recent advances in deep learning for object detection,” *Neurocomputing*, vol. 396, pp. 39–64, 2020, doi: 10.1016/j.neucom.2020.01.085.

[58] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs,” *IEEE Trans Pattern Anal Mach Intell*, vol. 40, no. 4, pp. 834–848, 2018, doi: 10.1109/TPAMI.2017.2699184.

[59] V. Sharma and R. N. Mir, “A comprehensive and systematic look up into deep learning-based object detection techniques: A review,” *Comput Sci Rev*, vol. 38, p. 100301, 2020, doi: 10.1016/j.cosrev.2020.100301.

[60] R. Xu, Y. Tao, Z. Lu, and Y. Zhong, "Attention-mechanism-containing neural networks for high-resolution remote sensing image classification," *Remote Sens (Basel)*, vol. 10, no. 10, pp. 1–29, 2018, doi: 10.3390/rs10101602.

[61] I. S. and G. E. H. Alex Krizhevsky, "ImageNet Classification with Deep Convolutional Neural Networks Alex," *Adv Neural Inf Process Syst*, pp. 1097–1105, 2012.

[62] K. Weiss, T. M. Khoshgoftaar, and D. D. Wang, *A survey of transfer learning*, vol. 3, no. 1. Springer International Publishing, 2016. doi: 10.1186/s40537-016-0043-6.

[63] H, Wu, T. Wiesner-Hanks, L.E. Stewart, C. DeChant, N. Kaczmar, A.M. Gore, J.R. Nelson, and H, Lipson, "Autonomous Detection of Plant Disease Symptoms Directly from Aerial Imagery," *Plant Phenome Journal*, vol. 2, no. 1, pp. 1–9, 2019, doi: 10.2135/tppj2019.03.0006.

[64] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *Journal of Machine Learning Research*, vol. 10, pp. 1633–1685, 2009.

[65] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects," *IEEE Trans Neural Netw Learn Syst*, pp. 1–21, 2021, doi: 10.1109/tnnls.2021.3084827.

[66] D. Choi, C. J. Shallue, Z. Nado, J. Lee, C. J. Maddison, and G. E. Dahl, "On Empirical Comparisons of Optimizers for Deep Learning," no. 1, 2019.

[67] S. Ruder, "An overview of gradient descent optimization algorithms," pp. 1–14, 2016, http://arxiv.org/abs/1609.04747.

[68] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation Functions: Comparison of trends in Practice and Research for Deep Learning," pp. 1–20, 2018.

[69] R. Sun, "Optimization for deep learning: theory and algorithms," pp. 1–60, 2019, http://arxiv.org/abs/1912.08957.

[70] T. Kattenborn, J. Leitloff, F. Schiefer, and S. Hinz, "Review on Convolutional Neural Networks (CNN) in vegetation remote sensing," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 173, no. March, pp. 24–49, 2021, doi: 10.1016/j.isprsjprs.2020.12.010.

[71] S. M. Anwar, M. Majid, A. Qayyum, M. Awais, M. Alnowami, and M. K. Khan, "Medical Image Analysis using Convolutional Neural Networks: A Review," *J Med Syst*, vol. 42, no. 11, 2018, doi: 10.1007/s10916-018-1088-1.

[72] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, *A survey of the recent architectures of deep convolutional neural networks*, vol. 53, no. 8. Springer Netherlands, 2020. doi: 10.1007/s10462-020-09825-6.

[73] D.M. Zeiler, M, Ranzato, R. Monga, M. Mao, K. Yang, V.Q. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, and E.G. Hinton, "On rectified linear units for speech recognition," *New York*, pp. 3517–3521, 2013.

[74] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2015 Inter, pp. 1026–1034, 2015, doi: 10.1109/ICCV.2015.123.

[75] X. Glorot, A. Bordes, and Y. Bengio, "Deep Sparse Rectifier Neural Networks Xavier," in *Appearing in Proceedings of the 14th International Con- ference on Artificial Intelligence and Statistics (AISTATS)*, Fort Lauderdale, FL, USA: JMLR: W&CP 15, 2011, pp. 315–323.

[76] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines Vinod," in *Proceedings of the 27th International Confer- ence on Machine Learning*, Haifa, Israel, 2010, pp. 807–814.

[77] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8689 LNCS, no. PART 1, pp. 818–833, 2014, doi: 10.1007/978-3-319-10590-1_53.

[78] S. R. Christian Szegedy, Wei Liu, Yangqing Jia1, Pierre Sermanet and A. R. Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, "Going Deeper with Convolutions Christian," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2015, pp. 1–9.

[79] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, vol. 1, no. 9. MIT Press, 2016.

[80] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, vol. 28, 2013.

[81] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical Evaluation of Rectified Activations in Convolutional Network," 2015.

[82] R. M. Neal, "Connectionist learning of belief networks," *Artif Intell*, vol. 56, no. 1, pp. 71–113, 1992, doi: 10.1016/0004-3702(92)90065-6.

[83] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *Journal of Machine Learning Research*, vol. 9, pp. 249–256, 2010.

[84] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *IEEE Trans Pattern Anal Mach Intell*, vol. 39, no. 12, pp. 2481–2495, 2017, doi: 10.1109/TPAMI.2016.2644615.

[85] M. Lin, Q. Chen, and S. Yan, "Network in network," *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, pp. 1–10, 2014.

[86] A. Ajit, K. Acharya, and A. Samanta, "A Review of Convolutional Neural Networks," *International Conference on Emerging Trends in Information Technology and*

Engineering, ic-ETITE 2020, no. October 2020, doi: 10.1109/ic-ETITE47903.2020.049.

[87] X. Yang, N. Bao, W. Li, S. Liu, Y. Fu, and Y. Mao, "Soil nutrient estimation and mapping in farmland based on uav imaging spectrometry," *Sensors*, vol. 21, no. 11, 2021, doi: 10.3390/s21113919.

[88] L. Ma, Y. Liu, X. Zhang, Y. Ye, G. Yin, and B. A. Johnson, "Deep learning in remote sensing applications: A meta-analysis and review," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 152, no. March, pp. 166–177, 2019, doi: 10.1016/j.isprsjprs.2019.04.015.

[89] E. Cengil and A. Cinar, "Multiple classification of flower images using transfer learning," *2019 International Conference on Artificial Intelligence and Data Processing Symposium, IDAP 2019*, 2019, doi: 10.1109/IDAP.2019.8875953.

[90] Z. K. Huang, C. Q. He, Z. N. Wang, J. M. Xi, H. Wang, and L. Y. Hou, "Cinnamomum Camphora Classification Based on Leaf Image Using Transfer Learning," *Proceedings of 2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference, IAEAC 2019*, no. Iaeac, pp. 1426–1429, 2019, doi: 10.1109/IAEAC47372.2019.8997791.

[91] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–14, 2015.

[92] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 2818–2826, 2016, doi: 10.1109/CVPR.2016.308.

[93] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 770–778, 2016, doi: 10.1109/CVPR.2016.90.

[94] M. S. Ebrahimi and H. K. Abadi, "Study of Residual Networks for Image Recognition," *Intelligent Computing - Proceedings of the 2021 Computing Conference*, pp. 754–763, 2021, doi: 10.1007/978-3-030-80126-7_53.

[95] G.A. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *ArXiv*, 2017.

[96] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1800–1807.*, 2016, pp. 1251–1258.

[97] D. Rahmawati, "Major pests and diseases of maize and availability of control technology," 2020, doi: 10.1088/1755-1315/484/1/012105.

[98] R. Ahila Priyadharshini, S. Arivazhagan, M. Arun, and A. Mirnalini, "Maize leaf disease classification using deep convolutional neural networks," *Neural Comput Appl*, vol. 31, no. 12, pp. 8887–8895, 2019, doi: 10.1007/s00521-019-04228-3.

[99] P. Bhatt, S. Sarangi, A. Shivhare, D. Singh, and S. Pappula, "Identification of diseases in corn leaves using convolutional neural networks and boosting," *ICPRAM 2019 - Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods*, no. Icpram, pp. 894–899, 2019, doi: 10.5220/0007687608940899.

[100] S. Yuheng and Y. Hao, "Image Segmentation Algorithms Overview," *ArXiv*, vol. 1, 2017.

[101] H. Zhang, Z. Hu, W. Qin, M. Xu, and M. Wang, "Adversarial co-distillation learning for image recognition," *Pattern Recognit*, vol. 111, 2021, doi: 10.1016/j.patcog.2020.107659.

[102] A. Waheed, M. Goyal, D. Gupta, A. Khanna, A. E. Hassanien, and H. M. Pandey, "An optimized dense convolutional neural network model for disease recognition and classification in corn leaf," *Comput Electron Agric*, vol. 175, no. January, p. 105456, 2020, doi: 10.1016/j.compag.2020.105456.

[103] H. Das Kshyanaprava Panda Panigrahi, A. K. Sahoo, and S. C. Moharana, "Maize Leaf Disease Detection and Classification Using Machine Learning Algorithms," *Progress in Computing, Analytics and Networking, Advances in Intelligent Systems and Computing*, no. January, pp. 659–669, 2020, doi: 10.1007/978-981-15-2414-1_66.

[104] X. Zhang, Y. Qiao, F. Meng, C. Fan, and M. Zhang, "Identification of maize leaf diseases using improved deep convolutional neural networks," *IEEE Access*, vol. 6, pp. 30370–30377, 2018, doi: 10.1109/ACCESS.2018.2844405.

[105] Z. Lin, S. Mu, A. Shi, C. Pang, and X. Sun, "a N Ovel M Ethod of M Aize L Eaf D Isease I Mage I Dentification B Ased on a M Ultichannel C Onvolutional," pp. 1–29, 2018.

[106] E. Alehegn, "Maize Leaf Diseases Recognition and Classification Based on Imaging and Machine Learning Techniques," in *International Journal of Innovative Research in Computer and Communication Engineering*, 2017, pp. 1–11.

[107] S. Kai, Z. Liu, H. Su, and C. Guo, "A research of maize disease image recognition of corn based on BP networks," *Proceedings - 3rd International Conference on Measuring Technology and Mechatronics Automation, ICMTMA 2011*, vol. 1, no. 2009921090, pp. 246–249, 2011, doi: 10.1109/ICMTMA.2011.66.

[108] M. Weiss, F. Jacob, and G. Duveiller, "Remote sensing for agricultural applications: A meta-review," *Remote Sens Environ*, vol. 236, no. August 2019, p. 111402, 2020, doi: 10.1016/j.rse.2019.111402.

[109] H. Wu, T. Wiesner-Hanks, L.E. Stewart, C. DeChant, N. Kaczmar, A.M. Gore, J.R. Nelson, and H. Lipson, "Autonomous Detection of Plant Disease Symptoms Directly from

Aerial Imagery," *Plant Phenome Journal*, vol. 2, no. 1, pp. 1–9, 2019, doi: 10.2135/tppj2019.03.0006.

[110] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018, doi: 10.1109/CVPR.2018.00474.

[111] K. Song, X. Sun, and J. Ji, "Corn leaf disease recognition based on support vector machine method," *Nongye Gongcheng Xuebao/Transactions of the Chinese Society of Agricultural Engineering*, vol. 23, no. 1, pp. 155–157, 2007.

[112] E. L. Stewart *et al.*, "Quantitative Phenotyping of Northern Leaf Blight in UAV Images Using Deep Learning," *Remote Sens (Basel)*, vol. 11, no. 19, pp. 1–10, 2019, doi: 10.3390/rs11192209.

[113] J. Shi and C. Tomasi, "Good Features to Track (Shi-Tomasi)," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 593–600, 1994.

[114] K. Y. Kok and P. Rajendran, "Validation of Harris Detector and Eigen Features Detector," *IOP Conf Ser Mater Sci Eng*, vol. 370, no. 1, 2018, doi: 10.1088/1757-899X/370/1/012013.

[115] J. Howse and J. Minichino, "Learning OpenCV 4 Computer Vision with Python 3," 3rd ed., Birmingham, UK: Packt Publishing Ltd., Feb. 2020. [Online]. Available: https://www.packtpub.com/product/learning-opencv-4-computer-vision-with-python-3-third-edition/9781789531619. [Accessed: September 16, 2022].

[116] Dr. M. Gevorgyan, A. Mamikonyan, and M. Beyeler, *OpenCV 4 with Python Blueprints*, 2nd Editio. Birmingham: Packt Publishing Ltd, 2020.

[117] H. Yang and I. Patras, "Mirror, mirror on the wall, tell me, is the error small?" *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June, pp. 4685–4693, 2015, doi: 10.1109/CVPR.2015.7299100.

[118] S. Xie and Z. Tu, "Holistically-nested edge detection," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2015 Inter, pp. 1395–1403, 2015, doi: 10.1109/ICCV.2015.164.

[119] J. Salamon and J. P. Bello, "Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification," *IEEE Signal Process Lett*, vol. 24, no. 3, pp. 279–283, 2017, doi: 10.1109/LSP.2017.2657381.

[120] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2015 Inter, pp. 2650–2658, 2015, doi: 10.1109/ICCV.2015.304.

[121] P. Barmpoutis, T. Stathaki, and V. Kamperidou, "Monitoring of Trees' Health Condition Using a UAV Equipped with Low-cost Digital Camera," *ICASSP, IEEE*

International Conference on Acoustics, Speech and Signal Processing - Proceedings, vol. 2019-May, pp. 8291–8295, 2019, doi: 10.1109/ICASSP.2019.8683128.

[122] D. M. Hawkins, "The Problem of Overfitting," *J Chem Inf Comput Sci*, vol. 44, no. 1, pp. 1–12, 2004, doi: 10.1021/ci0342472.

[123] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights Imaging*, vol. 9, no. 4, pp. 611–629, 2018, doi: 10.1007/s13244-018-0639-9.

[124] I. Stancin and A. Jovic, "An overview and comparison of free Python libraries for data mining and big data analysis," *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2019 - Proceedings*, pp. 977–982, 2019, doi: 10.23919/MIPRO.2019.8757088.

[125] W. Zhu, N. Zeng, and N. Wang, "Sensitivity, specificity, accuracy, associated confidence interval and ROC analysis with practical SAS® implementations.," *Northeast SAS Users Group 2010: Health Care and Life Sciences*, pp. 1–9, 2010.

[126] A. Kosmopoulos, I. Partalas, E. Gaussier, G. Paliouras, and I. Androutsopoulos, "Evaluation measures for hierarchical classification: A unified view and novel approaches," *Data Min Knowl Discov*, vol. 29, no. 3, pp. 820–865, 2015, doi: 10.1007/s10618-014-0382-x.

[127] M. Hai, Y. Zhang, and Y. Zhang, "A Performance Evaluation of Classification Algorithms for Big Data," *Procedia Comput Sci*, vol. 122, pp. 1100–1107, 2017, doi: 10.1016/j.procs.2017.11.479.

[128] W.S. Chen, S.S. Shivakumar, S. Dcunha, J. Das, E. Okon, C. Qu, J.C. Taylor, and V. Kumar, "Counting Apples and Oranges with Deep Learning: A Data-Driven Approach," *IEEE Robot Autom Lett*, vol. 2, no. 2, pp. 781–788, 2017, doi: 10.1109/LRA.2017.2651944.

[129] S. Taghavi Namin, M. Esmaeilzadeh, M. Najafi, T. B. Brown, and J. O. Borevitz, "Deep phenotyping: Deep learning for temporal phenotype/genotype classification," *Plant Methods*, vol. 14, no. 1, pp. 1–14, 2018, doi: 10.1186/s13007-018-0333-4.

[130] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *J Big Data*, vol. 6, no. 1, 2019, doi: 10.1186/s40537-019-0197-0.

[131] X. Chen, B. Xu, and H. Lu, "Effects of parallel structure and serial structure on convolutional neural networks," *J Phys Conf Ser*, vol. 1792, no. 1, 2021, doi: 10.1088/1742-6596/1792/1/012074.

[132] H. Tian, T. Wang, Y. Liu, X. Qiao, and Y. Li, "Computer vision technology in agricultural automation —A review," *Information Processing in Agriculture*, vol. 7, no. 1, pp. 1–19, 2020, doi: 10.1016/j.inpa.2019.09.006.

[133] M. Abdoli, H. Sarikhani, M. Ghanbari, and P. Brault, “Gaussian mixture model-based contrast enhancement,” *IET Image Process*, vol. 9, no. 7, pp. 569–577, 2015, doi: 10.1049/iet-ipr.2014.0583.

[134] H. Lidong, Z. Wei, W. Jun, and S. Zebin, “Combination of contrast limited adaptive histogram equalisation and discrete wavelet transform for image enhancement,” *IET Image Process*, vol. 9, no. 10, pp. 908–915, 2015, doi: 10.1049/iet-ipr.2015.0150.

[135] K. Gu, G. Zhai, X. Yang, W. Zhang, and C. W. Chen, “Automatic Contrast Enhancement Technology with Saliency Preservation,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 9, pp. 1480–1494, 2015, doi: 10.1109/TCSVT.2014.2372392.

[136] X. Sun, L. Chen, and D. H. K. Tsang, “Energy-efficient cooperative sensing scheduling for heterogeneous channel access in Cognitive Radio,” *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, pp. 145–150, 2012, doi: 10.1109/INFCOMW.2012.6193477.

[137] M. Shakeri, M. H. Dezfoulian, H. Khotanlou, A. H. Barati, and Y. Masoumi, “Image contrast enhancement using fuzzy clustering with adaptive cluster parameter and sub-histogram equalization,” *Digital Signal Processing: A Review Journal*, vol. 62, pp. 224–237, 2017, doi: 10.1016/j.dsp.2016.10.013.

[138] K. Neupane and F. Baysal-Gurel, “Automatic identification and monitoring of plant diseases using unmanned aerial vehicles: A review,” *Remote Sens (Basel)*, vol. 13, no. 19, 2021, doi: 10.3390/rs13193841.

[139] Z. Huang, Z. Wang, J. Zhang, Q. Li, and Y. Shi, “Image enhancement with the preservation of brightness and structures by employing contrast limited dynamic quadri-histogram equalization,” *Optik (Stuttg)*, vol. 226, p. 165877, 2021, doi: 10.1016/j.ijleo.2020.165877.

[140] W. Wang, X. Yuan, Z. Chen, X. Wu, and Z. Gao, “Weak-Light Image Enhancement Method Based on Adaptive Local Gamma Transform and Color Compensation,” *J Sens*, vol. 2021, 2021, doi: 10.1155/2021/5563698.

[141] F. S. Ishengoma, I. A. Rai, and S. R. Ngoga, “Hybrid convolution neural network model for a quicker detection of infested maize plants with fall armyworms using UAV-based images,” *Ecol Inform*, vol. 67, no. July 2021, p. 101502, 2022, doi: 10.1016/j.ecoinf.2021.101502.

[142] P. Singh, C.P. Pandey, P.G. Petropoulos, A. Pavlides, K.P. Srivastava, N. Koutsias, K.A.K. Deng, and Y. Bao, *Hyperspectral remote sensing in precision agriculture: Present status, challenges, and future trends*. LTD, 2020. doi: 10.1016/B978-0-08-102894-0.00009-7.

[143] K.S. Bhoi, K.K. Jena, K.S. Panda, V.H. Long, R. Kumar, P. Subbulakshmi, and B.H. Jebreen, “An Internet of Things assisted Unmanned Aerial Vehicle based artificial intelligence model for rice pest detection,” *Microprocess Microsyst*, vol. 80, no. November 2020, p. 103607, 2021, doi: 10.1016/j.micpro.2020.103607.

[144] M. Gomez Selvaraj, A. Vergara, F. Montenegro, H. Alonso Ruiz, N. Safari, D. Raymaekers, W. Ocimati, J. Ntamwira, L. Tits, B.A. Omondi, and G. Blomme, "Detection of banana plants and their major diseases through aerial images and machine learning methods: A case study in DR Congo and Republic of Benin," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 169, no. April, pp. 110–124, 2020, doi: 10.1016/j.isprsjprs.2020.08.025.

[145] B. Wu, A. Liang, H. Zhang, T. Zhu, Z. Zou, D. Yang, W. Tang, J. Li, and J. Su, "Application of conventional UAV-based high-throughput object detection to the early diagnosis of pine wilt disease by deep learning," *For Ecol Manage*, vol. 486, no. February, p. 118986, 2021, doi: 10.1016/j.foreco.2021.118986.

[146] S. Yadav, N. Sengar, A. Singh, A. Singh, and M. K. Dutta, "Identification of disease using deep learning and evaluation of bacteriosis in peach leaf," *Ecol Inform*, vol. 61, no. February, p. 101247, 2021, doi: 10.1016/j.ecoinf.2021.101247.

[147] T. Dozat, "Incorporating Nesterov Momentum into Adam," *ICLR Workshop*, no. 1, pp. 2013–2016, 2016.

Farian S. Ishengoma

This page has been purposefully left blank.

Appendix I

Publication

1. **F. S. Ishengoma,** I. A. Rai, and R. N. Said, "Identification of maize leaves infected by fall armyworms using UAV-based imagery and convolutional neural networks," Comput Electron Agric, vol. 184, no. October 2020, p. 106124, 2021, doi: 10.1016/j.compag.2021.106124.
2. **F. S. Ishengoma,** I. A. Rai, and S. R. Ngoga, "Hybrid convolution neural network model for a quicker detection of infested maize plants with fall armyworms using UAV-based images," Ecol Inform, vol. 67, no. July 2021, p. 101502, 2022, doi: 10.1016/j.ecoinf.2021.101502.
3. **F. S. Ishengoma**, I. A. Rai, and I. Gatare "Contrast Enhancement of UAV-Based Maize Plant Images for Automatic Detection of Fall Armyworm," in Data Science and Algorithms in Systems, P. and P. Z. Silhavy Radek and Silhavy, Ed., Cham: Springer International Publishing, 2022, pp. 100–107, doi:10.1007/978-3-031-21438-7_8.
4. **F. S. Ishengoma**, I. A. Rai, and I. Gatare "Autonomous System for Locating the Maize Plant Infected by Fall Armyworm Using the UAV images" in Artificial Intelligence Application in Networks and Systems, R. Silhavy and P. Silhavy (Eds.), Cham: Springer International Publishing 2023, pp. 1–8, doi:10.1007/978-3-031-35314-7_10.

Farian S. Ishengoma

This page has been purposefully left blank.

Appendix II

Optimizers

a) Gradient Descent

Training neural networks using gradient descent is an effective strategy since it relies on making incremental changes at the local level to eventually reach a global consensus [67]. After each epoch, a small update is made to the weights. The gradient descent technique starts with a given set of coefficients, calculates the cost of those coefficients, and then looks for a cost value that is less than the current value [68]. When the local minimum has been attained, the cycle starts over again. Whenever a process reaches a local minimum, it halts. Gradient descent is computationally expensive when the data size is huge [67]. Despite its success with convex functions, it is unable to determine how far up the gradient to advance for a non-convex function. The Gradient descent is the way to minimize an objective function $\nabla F(\theta)$ parameterized by a model's parameters $\theta \in \mathbb{R}^d$ by updating the parameters in the opposite direction of the gradient of the objective function $\nabla F(\theta)$ with respect to the parameters as indicated in Equation 1.

$$\theta = \theta - \eta . \nabla F(\theta) \qquad 1$$

b) Stochastic Gradient Descent

The randomness that the method is designed to exploit is referred to as stochastic, and the name "stochastic" refers to that randomness. Instead of utilizing the complete dataset, the stochastic gradient descent (SGD) algorithm uses a random selection process to choose data batches for each iteration [52], [67]. It adjusts the parameters for each batch separately as $training\ samples\ (x^i) and\ label\ (y^i)$ as indicated in Equation 2.

$$\theta = \theta - \eta . \nabla \mathrm{F}(\theta; x^i; y^i) \qquad 2$$

Each cycle then involves randomly shuffling the data to get closer to the minimum value. The algorithm's path is noisier than that of the gradient descent algorithm since it uses subsets of the dataset at each iteration rather than the whole thing [52], [67]. To find the local minimums, SGD uses more iterations. When compared to the gradient descent optimizer, the overall calculation time grows with the number of iterations, but the computation cost shrinks. SGD is preferable to the batch gradient descent approach when

dealing with large datasets and limited processing resources. When the learning rate is lowered gradually, SGD converges in the same way as batch gradient descent, with high certainty reaching either a local or global minimum for non-convex and convex optimization [67].

c) Stochastic Gradient descent with momentum

The weights are adapted as the stochastic gradient descent with momentum algorithm sways back and forth between gradient directions [67]. Conversely, the process can be sped up by including a small proportion of the prior update into the current update. To save computing time, the stochastic gradient descent using a momentum algorithm eliminates the need for many iterations to find the minimal value [52]. Momentum is a technique that allows one to speed up in the right direction while simultaneously reducing erratic movement. The method relies on incorporating a proportion of the previous time step's update vector v_{t-1} to the current update vector as indicated in Equations 3 and 4. The momentum decay term γ is usually set to 0.9 or similar value.

$$v_t = \gamma v_{t-1} + \eta \nabla \mathrm{F}(\theta_t) \quad (3)$$

$$\theta = \theta - v_t \quad (4)$$

d) Mini-Batch Gradient Descent

Mini-batch gradient descent algorithms minimize training iterations by calculating the loss function with a portion of the dataset (n training samples) rather than the full dataset [67]. Because not all the training data needs to be imported into memory, the approach is faster than both stochastic gradient descent and batch gradient descent algorithms. Mini-batch gradient descent also has a rougher cost function than batch gradient descent but a smoother cost function than stochastic gradient descent [52], [67]. Mini-batch gradient descent strikes the best balance between speed and accuracy and is thus the method of choice. Getting the necessary precision calls for tuning a mini-batch-size hyper parameter [67]. Mini batch gradient descent is demonstrated numerically in Equation 5. Mini-batch sizes typically fall in the range of 50 to 256, though this might be higher or lower depending on the use case.

$$\theta = \theta - \eta \nabla F\left(\theta; x^{i:i+n}; y^{i:i+n}\right); \quad 5$$

e) Adagrad

Adagrad is a gradient-based optimization technique that adjusts its learning rate based on the frequency with which a parameter is updated, performing greater updates for less-frequently-used parameters. As a result, it is ideally suited for processing sparse data [67]. The adaptive gradient descent algorithm adjusts the learning rate with each iteration. Training-time parameter changes account for most of the variance in learning velocity. The greater the number of adjustable parameters, the less sensitive the learning rate will be to those adjustments [52], [67]. These improvements are especially useful for sparse and dense features found in real-world datasets. Adagrad uses Equations 6 - 8 to modify the weights. The Adagrad does away with the necessity for manual tweaks to the learning rate. Compared to other gradient descent methods, it converges more quickly and reliably. The learning rate is aggressively and monotonically decreased by the Adagrad optimizer, which is a drawback. This is because the squared gradients in the denominator keep adding up, making that fraction larger [69][67].

The update for every parameter θ_i at each time step *t* then becomes:

$$\theta_{t+1,i} = \theta_{t,i} - \eta . p_{t,i} \nabla_{\theta i} F\left(\theta_{t,i}\right) \quad 6$$

In its update rule, Adagrad modifies the general learning rate η at each time step *t* for every parameter θ_i based on the past gradients that have been computed for θ_i:

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} . \nabla_{\theta i} F\left(\theta_{t,i}\right) \quad 7$$

In Equation 7 the $G_t \in \mathbb{R}^{dxd}$ represents the diagonal matrix where each diagonal element *i, i* is the sum of the squares of the gradients with respect to θ_i up to time step *t,* while ϵ is a smoothing term that avoids division by zero. Without the square root operation, the algorithm performs much worse.

As G_t contains the sum of the squares of the past gradients with respect to all parameters θ along its diagonal, therefore an element-wise matrix-vector multiplication between G_t and p_t as shown in Equation 8:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot \nabla_\theta F(\theta_t) \quad 8$$

f) RMSProp

RMS prop is also an improvement to the AdaGrad optimizer because it slows down the rate at which the learning rate keeps going down. The algorithm's main goal is to speed up the process of optimizing by reducing the number of function evaluations needed to get to the local minimum [67], [69]. As shown in Equations 9 and 10, the algorithm figures out the moving average of squared gradients for each weight and divides each gradient by the square root of the mean square. The problem with RMS Prop is that the learning rate must be set by hand, and the value it suggests doesn't always work.

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1\nabla_\theta F(\theta_t^2) \quad (9)$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}}\nabla_\theta F(\theta_t) \quad (10)$$

RMSprop as well divides the learning rate by an exponentially decaying average of squared gradients. The γ is suggested to be set to 0:9, while a default value for the learning rate η is 0.001.

g) AdaDelta

The AdaDelta optimizer is stronger than the AdaGrad optimizer. It is based on adaptive learning and is meant to fix problems that AdaGrad and RMS prop optimizer have. AdaDelta uses two state variables to store the leaky average of the second moment gradient and the leaky average of the second moment of change of model parameters [67][69]. Adadelta only sums up gradients from a window of size w, rather than summing up all squared gradients from the past. The sum of gradients is defined as a decaying average of all squared gradients from the past, which saves space in comparison to storing w squared gradients from the past [67]. The running average $E[p^2]_t$ at time step *t* then depends only on the previous average and the current gradient as shown in Equations 11 - 21:

$$p_t = \nabla_\theta F(\theta_t) \quad (11)$$

$$E[p^2]_t = \gamma E[p^2]_{t-1} + (1-\gamma)p_t^2 \quad (12)$$

The momentum term γ is set to 0.9. For clarity $\Delta\theta_t$ can be written as in Equations 13 and 14.

$$\Delta\theta_t = -\eta . p_{t,i} \quad (13)$$

$$\theta_{t+1} = \theta_t + \Delta\theta_t \quad (14)$$

The parameter update vector of Adagrad that was derived previously thus takes the form of Equation 15:

$$\Delta\theta_t = -\frac{\eta}{\sqrt{G_t + \epsilon}} \odot p_t \quad (15)$$

The diagonal matrix G_t is replaced with the decaying average over past squared gradients $E[p^2]_t$ as shown in Equation 16.

$$\Delta\theta_t = -\frac{\eta}{\sqrt{E[p^2]_t + \epsilon}} p_t \quad (16)$$

In Equation 17 the denominator is replaced with the root mean squared (RMS) error criterion of the gradient.

$$\Delta\theta_t = -\frac{\eta}{RMS[p]_t} p_t \quad (17)$$

It was pointed out that the units in this update, as well as in SGD, Momentum, or Adagrad, do not match, which means that the update should have the same hypothetical units as the parameter [67]. So, another average of squared parameter updates with an exponentially decreasing rate was set up as shown in Equation 18.

$$E[\Delta\theta^2]_t = \gamma E[\Delta\theta^2]_{t-1} + (1-\gamma)\Delta\theta_t^2 \quad (18)$$

Equation 19 gives us the root mean squared error of parameter updates.

$$RMS[\Delta\theta]_t = \sqrt{E[\Delta\theta^2]_t + \epsilon} \quad (19)$$

Since $RMS[\Delta\theta]_t$ is unknown, [69] approximate it with the RMS of parameter updates until the previous time step. Replacing the learning rate η in the previous update rule with $RMS[\Delta\theta]_{t-1}$ finally gives the Adadelta update rule as shown in Equations 20 and 21:

$$\Delta\theta_t = -\frac{RMS[\Delta\theta]_{t-1}}{RMS[p]_t} p_t \quad (20)$$

$$\theta_{t+1} = \theta_t + \Delta\theta_t \quad (21)$$

h) Adam

The adaptive moment estimation (Adam) optimizer is an extension of stochastic gradient descent that changes the weights of the network as it learns. It performs periodic manual adjustments to the network's learning rate for each weight in the network [67],

[69]. Adam takes after both the Adagrad and RMS prop algorithms in terms of his features. Adam uses the second instant of gradients to adjust learning rates, while RMS Prop just uses the first. Many people choose to use the Adam optimizer because of its many benefits. It is recommended as the go-to optimization method and used as a standard by many deep learning research papers [66][69]. Furthermore, the approach requires less tuning than any other optimization technique, has a faster running time, uses less memory, and has less storage requirements. Adam's weakness is that it prioritizes speed of computing over data points, as is the case with algorithms like stochastic gradient descent [66], [67]. Thus, algorithms like SGD provide better generalization but at the cost of computational slowness.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) p_t \quad (22)$$

$$v_t = \beta_2 \, v_{t-1} + (1 - \beta_2) p_t^2 \quad (23)$$

m_t and $v_t$are estimates of the first moment (the mean) and the second moment (the uncentered variance) of the gradients respectively, hence the name of the method. As m_t and v_t are initialized as vectors of 0's, the authors of Adam observe that they are biased towards zero, especially during the initial time steps, and especially when the decay rates are small (i.e., β_1 and β_2 are close to 1). To account for these distortions, they calculate first and second moment estimates that are bias corrected, as shown in Equation 24 - 25.

$$\widehat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad (24)$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \quad (25)$$

The Adam update rule is given by Equation 26. The authors propose default values of 0.9 for β_1, 0.999 for β_2, and 10^{-8} for ϵ.

$$\theta_{t+1} = \theta_t + \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \widehat{m}_t \quad (26)$$

i) AdaMax

The Adam optimization algorithm is built upon by the AdaMax algorithm. It goes beyond what the Gradient Descent Optimization algorithm can do [67]. In the Adam update rule as shown in Equations 27 and 28, the v_t factor scales the gradient inversely

proportionally to the l_2 norm of the previous gradients (via the v_{t-1} term) and current gradient $|p_t|^2$:

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)|p_t|^2 \quad 27$$

We can generalize this update to the l_g standard as shown in Equation 21.

$$v_t = \beta_2^g v_{t-1} + \left(1-\beta_2^g\right)|p_t|^g \quad 28$$

Large g value norms are generally numerically unstable, which is why l_1 and l_2 norms are most used in practice. However, l_∞ on the other hand, has generally stable behavior. As a result, the authors propose AdaMax [67] and demonstrate that v_t with l_∞ to the more stable value shown in Equations 29 and 30. To avoid confusion with Adam, u_t is used to stand for the infinity norm-constrained v_t:

$$v_t = \beta_2^\infty v_{t-1} + (1-\beta_2^\infty)|p_t|^\infty \quad 29$$

$$v_t = max(\beta_2 . v_{t-1}, |p_t|) \quad 30$$

When Equation 31 is substituted for the Adam update equation, the AdaMax update rule is obtained by replacing $\sqrt{\hat{v}_t} + \epsilon$ with u_t:

$$\theta_{t+1} = \theta_t - \frac{\eta}{u_t}\hat{m}_t \quad 31$$

Because u_t relies on the max operation, it is less likely to bias towards zero than m_t and v_t in Adam, so there is no need of computing a bias correction u_t. Again, good default values are η = 0.002, β_1 = 0.9, and β_2 = 0.999.

j) Nadam

Nadam (Nesterov-accelerated Adaptive Moment Estimation) combines Adam and Nesterov accelerated gradient (NAG) Adam's momentum term, m_t needs to be modified so that NAG may be incorporated [67][66][147].

$$m_t = \gamma m_{t-1} + \eta p_t \quad 32$$

$$\theta_{t+1} = \theta_t - m_t \quad 33$$

$$\theta_{t+1} = \theta_t - (\gamma m_{t-1} + \eta p_t) \quad 34$$

It can be shown from Equations 32- 37 that gaining momentum consists of taking a single step in the direction of the preceding momentum vector and a single step in the direction of the current gradient. By recalibrating the parameters with the momentum step before

computing the gradient, NAG enables us to take a more accurate step in the gradient direction. By adjusting the gradient p_t, the equation of NAG can be obtained:

$$\mathrm{p_t} = \nabla \mathrm{F}(\theta_\mathrm{t} - \gamma m_{t-1}) \quad 35$$

$$\mathrm{m_t} = \gamma \mathrm{m_{t-1}} + \eta \mathrm{p_t} \quad 36$$

$$\theta_{t+1} = \theta_t - m_t \quad 37$$

The following are some of the alterations to NAG that Dozat [147] suggests: The parameters can be updated independently of the gradient, p_t, saving the momentum step for use only once. Θ_{t+1}, we now use the look-ahead momentum vector to update the current parameters:

$$p_t = \nabla F(\theta_t) \quad 38$$

$$m_t = \gamma m_{t-1} + \eta p_t \quad 39$$

$$\theta_{t+1} = \theta_t - (\gamma m_t + \eta p_t) \quad 40$$

Replacement of the old momentum vector m_t with the new momentum vector m_{t-1} in Equation 34. Similarly, the new momentum vector can substitute for the old one to give Adam some Nesterov momentum. The rule for updating Adam is as follows in Equation 41 - 42.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) p_t \quad 41$$

$$\widehat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad 42$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\widehat{v}_t} + \epsilon} \widehat{m}_t \quad 43$$

Expanding equation 43 with the definitions of m_t and $\widehat{m}_t$ in turn gives us Equation 44:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\widehat{v}_t} + \epsilon} \left(\frac{\beta_1 m_{t-1}}{1 - \beta_1^t} + \frac{(1 - \beta_1) p_t}{1 - \beta_1^t} \right) \quad 44$$

It is worth noting that $\frac{\beta_1 m_{t-1}}{1-\beta_1^t}$ is the bias-corrected estimate of the momentum vector of the previous time step. It can be replaced with $\widehat{m}_{t-1}$:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\widehat{v}_t} + \epsilon} \left(\beta_1 \widehat{m}_{t-1} + \frac{(1 - \beta_1) p_t}{1 - \beta_1^t} \right) \quad 45$$

Comparing Equation 45 to the modified momentum term in Equation 34, relative to how the Nesterov momentum was added in Equation 40, we can now add it by replacing the previous momentum vector $\widehat{m}_t$ with the current momentum vector $\widehat{m}_{t-1}$. This yields the Nadam update rule in Equation 46:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\widehat{v}_t} + \epsilon}\left(\beta_1 \widehat{m}_t + \frac{(1-\beta_1)p_t}{1-\beta_1^t}\right) \quad 46$$