Machine Learning Classification Technique for Plant Disease Classification

by

SIRANJEV A/L G. MURALI
([redacted])

A report submitted in partial fulfillment of the requirements for the degree of
Bachelor of Engineering (Computer Engineering)

Faculty of Electronic Engineering Technology
UNIVERSITI MALAYSIA PERLIS

2023/2024

UNIVERSITI MALAYSIA PERLIS

DECLARATION OF THESIS

Author's Full Name	:	SIRANJEV A/L G. MURALI
Title	:	Machine Learning Classification Technique for Plant Disease Classification
Date of Birth	:	09 MARCH 2000
Academic Session	:	2020/2021

I hereby declare that this report becomes the property of Universiti Malaysia Perlis (UniMAP) and to be placed at the library of UniMAP. This report is classified as:

☐	**CONFIDENTIAL**	(Contains confidential information under the Official Secret Act 1997) *
☐	**RESTRICTED**	(Contains restricted information as specified by the organization where research was done) *
☑	**OPEN ACCESS**	I agree that my report to be published as online open access (Full Text)

I, the author, give permission to reproduce this report in whole or in part for the purpose of research or academic exchange only (except during the period of _____ years, if so, requested above)

Certified by:

SIGNATURE

SIGNATURE OF SUPERVISOR

NAME OF SUPERVISOR

[redacted]

(NEW IC NO. /PASSPORT NO.)

Assoc. Prof. Ts. Dr. Phak Len Eh Kan

Faculty of Electronic Engineering & Technology

Date: 12 July 2024

Date: 12 July 2024

NOTES : * If the thesis is CONFIDENTIAL or RESTRICTED, please attach with the letter from the organization with the period and reasons for confidentiality or restriction.

ACKNOWLEDGEMENT

I am eternally grateful to Assoc. Prof. Ts. Dr. Phak Len Eh Kan for his helpful advice and unflinching support throughout my Final Year Project. His knowledge in electronic engineering not only broadened my grasp of the subject, but it also inspired me to investigate the more sophisticated parts of the field. I consider myself fortunate to have had such important counsel from him.

I'd also like to thank the Faculty of Electronic Engineering & Technology (FKTEN) for fostering a stimulating learning atmosphere that aided my academic path. The faculty's resources and facilities were critical to the successful completion of my FYP. The demanding academic curriculum and numerous learning opportunities at FKTEN greatly influenced the development of my abilities and knowledge in the field of electronic engineering.

Finally, I want to express my heartfelt gratitude to my family and friends for their steadfast support during this journey. Their support and belief in me have been a constant source of inspiration.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ML	Machine Learning
AI	Artificial Intelligence
CNN	Convolutional Neural Network
KNN	k- Nearest Neighbour
VGG16	Visual Geometry Group 16-layer deep neural network
ResNet	Residual Network
SVM	Support Vector Machine
EPC	Emperor Penguin Colony algorithm
LSSVM	Least square support vector machine
YOLO	You Only Look Once
GLCM	Gray-Levelled Co-occurrence Matrix
EffNet	Convolutional Neural network with compound scaling
DenNet	Convolutional Neural network with dense block
LW17	Light Weight 17-layer convolutional neural network
RGB	Red, Green and Blue

LIST OF SYMBOLS

mAP	mean average precision
%	percentage

Teknik Klasifikasi Pembelajaran Mesin untuk Pengesanan Penyakit Tumbuhan

ABSTRAK

Peningkatan kelaziman penyakit tumbuhan menimbulkan bahaya yang besar kepada keselamatan makanan global, memerlukan proses yang cepat dan tepat untuk pengesanan dan rawatan awal. Pembelajaran mesin (ML) telah muncul sebagai cara yang menjanjikan untuk mengautomasikan pengesanan penyakit tumbuhan, menyediakan penyelesaian yang cepat dan tidak invasif. Kajian ini menyiasat pelbagai algoritma klasifikasi pembelajaran mesin untuk menemui dan mengkategorikan penyakit tumbuhan. Algoritma pembelajaran mesin boleh mengesan simptom penyakit dalam tumbuhan dalam masa nyata menggunakan pengecaman gambar dan analisis corak, membolehkan campur tangan awal. Projek ini bertujuan untuk membina dan menguji model pembelajaran mesin pada koleksi foto jagung yang dilabelkan sebagai Sihat, Hawar, Bintik Daun Kelabu dan Karat Biasa. Model CNN mencapai kadar ketepatan 94% selepas penyediaan data yang meluas, pembinaan model dan penilaian yang rapi.

Machine Learning Classification Techniques for Plant Disease Detection

ABSTRACT

The increasing prevalence of plant diseases poses a substantial danger to global food security, necessitating quick and precise processes for early detection and treatment. Machine learning (ML) has emerged as a promising way for automating plant disease detection, providing a quick and non-invasive solution. This study investigates various machine learning classification algorithms for discovering and categorising plant diseases. Machine learning algorithms may detect illness symptoms in plants in real time using picture recognition and pattern analysis, allowing for early intervention. This project aims to build and test machine learning models on a collection of maize photos labelled as Healthy, Blight, Grey Leaf Spot, and Common Rust. The CNN model attained a 94% accuracy rate after extensive data preparation, model building, and rigorous evaluation.

CHAPTER 1 : INTRODUCTION

1.1 General Overview and Background

In the current era of rapidly changing climate conditions and expanding global populations, the interaction of agriculture and technology has never been more important. Plant diseases are one of the major setbacks facing the agricultural sector, which is the foundation of the supply of food. These illnesses have the ability to completely destroy crops, resulting in significant financial losses and endangering the livelihoods of farmers everywhere. Effective plant disease management is crucial because, according to estimate from the Food and Agriculture Organisation of the United Nations, up to 40% of food crops are lost to pests and plant diseases every year. (About. (n.d.-c). Retrieved from https://www.fao.org/plant-health-2020/about/en/)

To address this issue, Machine Learning (ML) has become a pivotal tool. Its capacity to quickly and accurately handle as well as analyse enormous volumes of data offers a fresh method for classifying plant diseases. ML provides a methodical and objective substitute for traditional approaches, which can be labour-intensive, subjective, and frequently rely on the knowledge of pathologists. ML systems can quickly identify disease signals from plant imaging by utilising image recognition and pattern analysis. This allows for real-time diagnosis, which is crucial for immediate response.

The use of ML in the classification of plant diseases is not without its challenges, though. Significant problems include the wide diversity of plant species, the unpredictability in disease presentation, and the requirement for tremendous annotated

datasets for training. Furthermore, environmental considerations and computational limitations must be taken into account when implementing ML models in actual agricultural settings.

The purpose of this study is to investigate how ML classification methods might be used to overcome these obstacles. This project aims to contribute to the sustainable development of agriculture, assuring that food is plentiful and resilient in the face of environmental stresses, by building strong models that can properly detect plant diseases from picture data.

1.2 Problem Statement

Crop health is a major subject in agriculture since it has a direct impact on food supply, economic stability, and resource management. Plant diseases pose a significant hazard to human health, producing extensive damage that results in large yield losses and, as a result, financial strain on farmers and the agricultural economy. The intricacy of identifying plant illnesses stems from the variety of symptoms and the wide spectrum of probable afflictions, each of which necessitates specific knowledge and experience to correctly identify.

Farmers in rural areas frequently must rely on their own judgement to identify and manage plant diseases, which can lead to misdiagnosis and the use of ineffective treatments. Such practices not only fail to address the underlying ailment, but they may also contribute to the emergence of resistant pathogen strains, complicating future disease management efforts.

The difficulty gets worse by the fact that plant diseases can spread quickly and are impacted by a range of factors, including climatic conditions, plant physiology, and the presence of vectors. Early and correct diagnosis is thus critical for containing outbreaks and mitigating their consequences. However, due to the massive number of farms and the overwhelming diversity of plant species and diseases, the traditional technique to disease detection, which is mostly visual and subjective, is not scalable.

This study attempts to overcome these issues by constructing an automated and precise machine learning system for plant disease categorization. A system like this would be a decision-support tool for farmers and agronomists, allowing for quick response to disease outbreaks and educated management decisions. The proposed approach aims to overcome the limits of human diagnosis by utilising picture data and powerful machine learning algorithms to provide a consistent and objective assessment of plant health.

1.3 Study Objectives

The Main objective of this study is to discover a solution that is effective in controlled situations and robust enough to be used in agricultural settings, including those with limited access to expert knowledge and resources. By doing so, the initiative hopes to contribute to the long-term management of plant diseases, so protecting crop health, increasing the availability of food, and improving the livelihoods of farming people around the world.

In order to achieve this, the following objectives need to be fulfilled:

1. To develop a machine learning model that detect plant disease.
2. To perform data preprocessing, categorization, and training iterations to build a strong model that will effectively use of supervised machine learning approaches for accurate plant disease diagnosis.

1.4 Research Scope

This study is limited to the use of datasets for the classification of plant diseases through the use of supervised machine learning algorithms. A limited set of plant species and diseases mainly those with considerable nutritional and economic significance will be the subject of the investigation. Although the research will not address disease therapy, it will lay the groundwork for further studies into intervention techniques.

For this study, the dataset used is obtained from Kaggle website where it consists of picture of corn and maize in several categories which are Healthy, Blight, Gray Leaf Spot and Common Rust. This is further discussed in the following chapters.

1.5 Report Outline

The report is broken into five chapters. The Introduction is the first chapter, and it gives a broad overview and background for the study, as well as the problem statement, study objectives, and research scope. It also explains the report's structure.

Second chapter is about literature review. This chapter provides an overview of classification approaches and machine learning algorithms such as k-nearest neighbour (KNN), convolutional neural networks (CNN), support vector machines (SVM), and others. It contains a review of previously published work, a cutting-edge table, and a summary of the findings from the literature.

The third chapter is titled Methodology. This chapter describes the methods employed in the study, including the tools and software libraries used. It covers the entire machine learning model building process, from data collection to preprocessing, model selection, training, evaluation, and deployment.

The fourth chapter is Results. This chapter provides the findings from various machine learning models, including KNN, Random Forest, and CNN. It examines each model's performance as well as the final model's deployment process.

The fifth and last chapter is titled Conclusion. This chapter presents an overview of the project, summarises the important findings, and makes recommendations for future research based on the findings. The Appendices section contains extra materials like as the Gantt Chart and other tables and figures that supplement the main topic of the report.

CHAPTER 2 : LITERATURE REVIEW

2.1 Overview

This chapter contains a thorough review of the literature on plant disease detection methods. Keywords such as "machine learning," "deep learning," "classification," "disease detection," and "plant disease" were employed. Each paper's abstract was examined to determine its suitability and inclusion in this report. All papers that did not focus on ML algorithms were discarded. There are studies that refer to plant disease identification using ML approaches, but they use the Internet of Things (IOT), and other non-essential information. Non-English literature, technical reports, and review papers were also removed.

2.2 Classification

Classification is a process in machine learning that includes categorising data based on its properties into predetermined classes or categories. The purpose of classification is to create a model that can predict the class labels of fresh, previously unseen data based on the patterns and relationships acquired from the training data. Identifying and extracting important characteristics from the input data, selecting an effective classification technique, and training the model using labelled examples are all part of this process. Once trained, the model may be used to assign new instances to one of the specified classes, making it useful for tasks such as image recognition, spam detection, medical diagnosis, and others.

2.3 Machine Learning (ML)

Machine learning is a branch of artificial intelligence that focuses on creating algorithms and statistical models that allow computers to learn and make predictions or assessments without being explicitly programmed. It entails using a dataset to train a model to recognise patterns and make predictions or decisions based on new data. Machine learning classification algorithms are used to categorise data into different classes or groups based on specific traits or properties. Classification techniques can be used in the context of plant disease to identify and classify various forms of plant diseases based on visual symptoms, leaf colour, texture, and other features.

2.3.1 Algorithm Types

An algorithm is a step-by-step technique or collection of rules used to solve a specific problem or complete a specific activity. Algorithms can perform basic tasks like sorting a list of numbers or sophisticated procedures like training a neural network to recognise patterns in photos. Algorithm design and selection are critical factors in determining the efficiency, accuracy, and performance of computing systems. The various type of algorithm is explained in the next section.

2.3.1.1 k-Nearest Neighbour algorithm (KNN)

K-Nearest Neighbours (KNN) is a straightforward classification technique that categorises a data point depending on how its neighbours are classed. It works by computing the distance between the new data point and all other points in the dataset, and then assigning a class to the new data point based on the majority class of the KNN as

shown in figure 2.1. KNN can be used to categorise plants based on disease symptoms and traits in the context of plant disease classification. KNN can predict the sort of disease affecting a plant by assessing the similarity of its traits to those of other plants with known diseases. This method becomes particularly helpful when the plant's characteristics and disease signs are obvious and quantifiable.

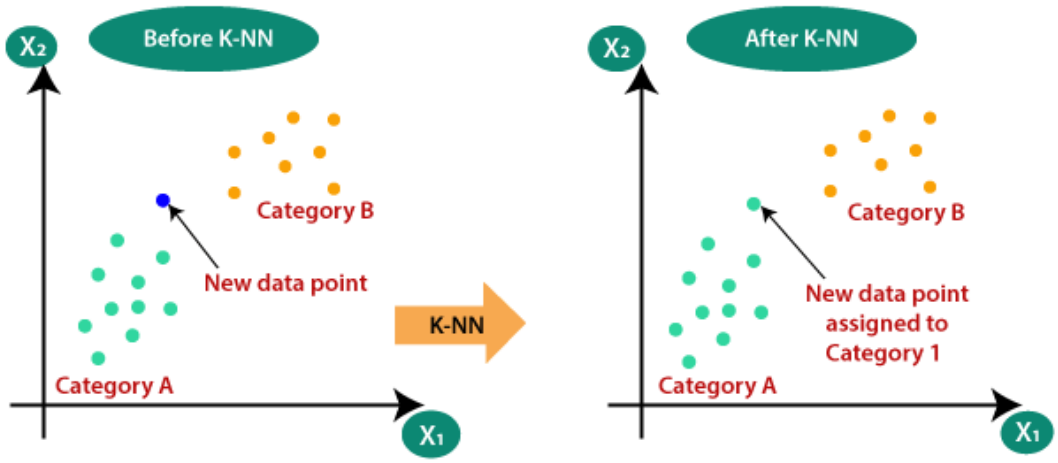

Figure 2.1 Mechanism of KNN
(Source: (KNN) – theory. (2020, March 28). Retrieved from http://www.datasciencelovers.com/machine-learning/k-nearest-neighbors-knn-theory/)

2.3.1.2 Convolution Neural Network (CNN)

CNNs are a sort of deep learning model that is specifically developed for processing and classifying visual data, such as photographs as shown in figure 2.2. CNNs employ a hierarchical pattern recognition approach in which they learn to recognise characteristics at several degrees of abstraction, ranging from edges and textures to complicated shapes and structures. CNNs may be trained to analyse photos of plant leaves and identify visual patterns associated with various diseases in the context of plant disease

classification. CNNs can automatically extract key features and classify plants depending on the presence of disease symptoms by learning from a huge collection of labelled photos. This method offers accurate and efficient diagnosis of plant diseases in agricultural settings, allowing for timely intervention and effective disease management.

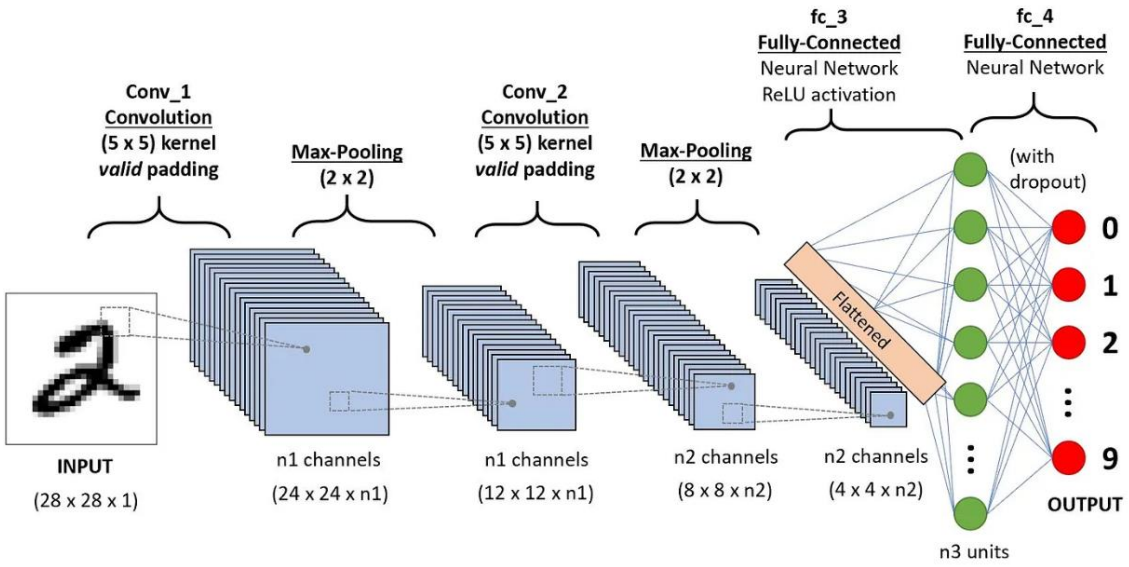

Figure 2.2 Architecture of CNN
(Source: Keita, Z. (2023, November 14). An Introduction to Convolutional Neural Networks (CNNs). Retrieved from https://www.datacamp.com/tutorial/introduction-to-convolutional-neural-networks-cnns)

2.3.1.3 Support Vector Machine (SVM)

SVM is a sophisticated classification algorithm that determines the best point for maximising the margin between classes shown in figure 2.3. Because of its ability to handle high-dimensional data and uncover complicated patterns, SVM can be especially useful in plant disease classification. It works by mapping input data such as plant leaf

colour and texture into a high-dimensional space where distinct disease classes can be separated with a clear border. This is accomplished through the use of kernel functions, which enable SVM to operate in a transformed feature space without explicitly computing the coordinates of the data in that space, allowing for accurate classification even when the relationship between the features and disease classes is not linear.

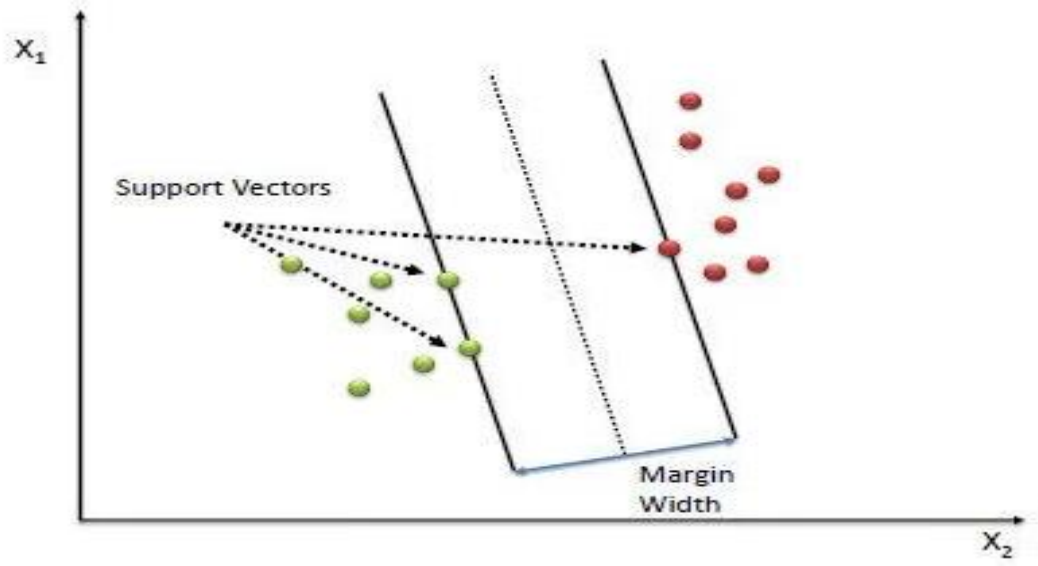

Figure 2.3 Working of SVM
(Source: Yadav, A. (2018, October 22). SUPPORT VECTOR MACHINES(SVM) - towards data science. Medium. Retrieved from https://towardsdatascience.com)

2.3.1.4 Least Squares Support Vector Machine (LSSVM)

The Least Squares Support Vector Machine (LSSVM) is a Support Vector Machine (SVM) variation that optimises using least squares cost function, resulting in a set of linear equations rather than a quadratic programming problem example shown in figure 2.4. This modification frequently results in faster computing speeds, making it appropriate for huge datasets. LSSVM keeps the essential notion of SVM, which is to discover the ideal hyperplane that divides classes with the greatest margin of separation,

but simplifies the solution method. LSSVM can manage the nonlinear and high-dimensional data that is typical when identifying illnesses based on a variety of plant traits, making it particularly efficient in the context of plant disease classification. LSSVM may find a framework that divides distinct plant diseases with a clear margin by translating the input space into a higher-dimensional space, allowing accurate classification even in complicated conditions.

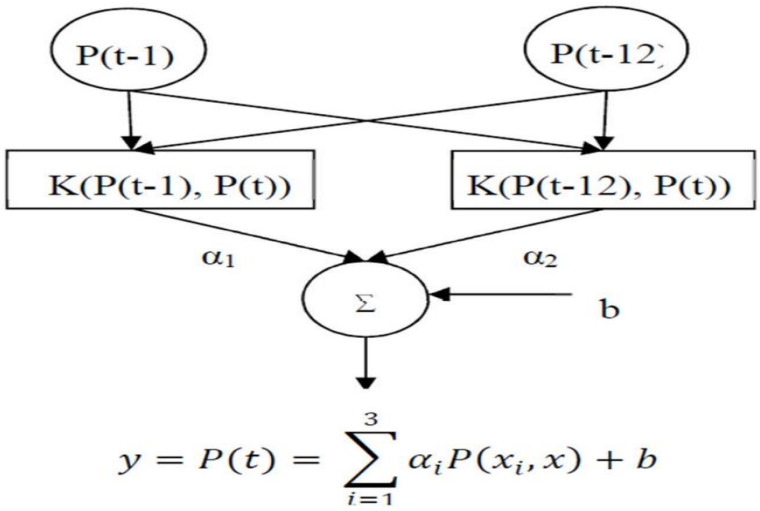

Figure 2.4 Structure of LSSVM
(Source: (LSSVM). (n.d.). Retrieved from https://www.researchgate.net/figure/Structure-of-least-square-support-vector-machine-LSSVM_fig4_331173889)

2.3.1.5 YOLO (You Only Look Once)

Yolov4, Yolov5 and Yolov7 are cutting-edge object identification models that recognise and classify items in images using a single-stage design. These models are intended to process photos quickly and accurately recognise many objects which are shown in figure 2.5. Yolov4, Yolov5 and Yolov7 can be used to detect and categorise sick regions on plant leaves or other plant parts in the context of plant disease categorization. These models can learn to identify and categorise different types of plant

illnesses based on visual signs by being trained on a dataset of photos containing healthy and diseased plants. This method enables the automated and rapid diagnosis of plant diseases in agricultural settings, allowing for timely intervention and disease management.

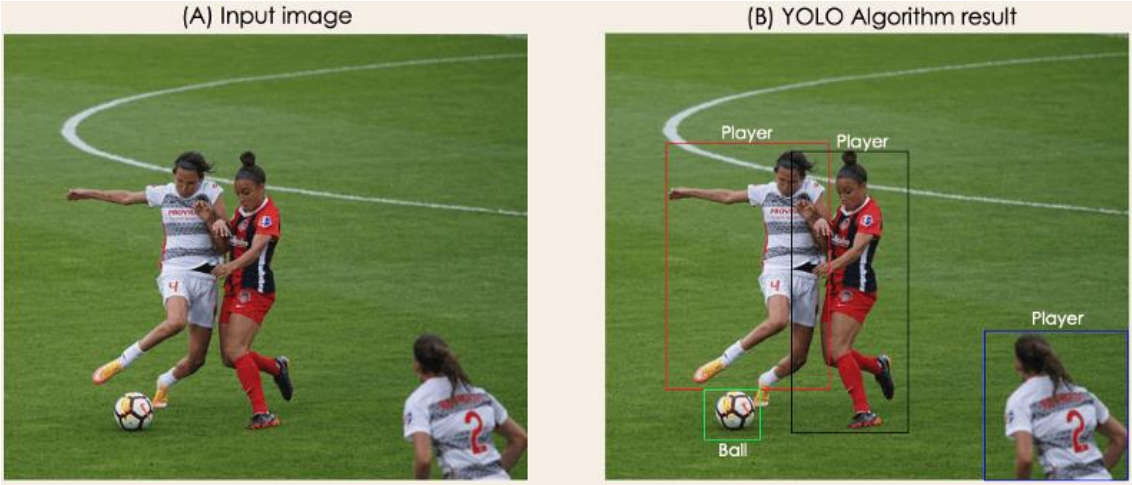

Figure 2.5 Example of YOLO output
(Source: Keita, Z. (2022, September 28). YOLO object detection explained. Retrieved from https://www.datacamp.com/blog/yolo-object-detection-explained)

2.3.1.6 Gray-Level Co-occurrence Matrix (GLCM)

The Gray-Level Co-occurrence Matrix (GLCM) is a statistical approach for analysing the spatial relationship between picture pixel values shown in figure 2.6. It assesses an image's texture by measuring how frequently different combinations of pixel values appear in relation to one another. GLCM can be used to extract textural information from photos of diseased plant leaves in the context of plant disease categorization. Texture variables like contrast, homogeneity, and entropy can then be employed as input features for classification models to differentiate between healthy and diseased plant samples. GLCM gives vital information for accurate categorization of plant

diseases based on visual characteristics by capturing the textural patterns associated with different illnesses.

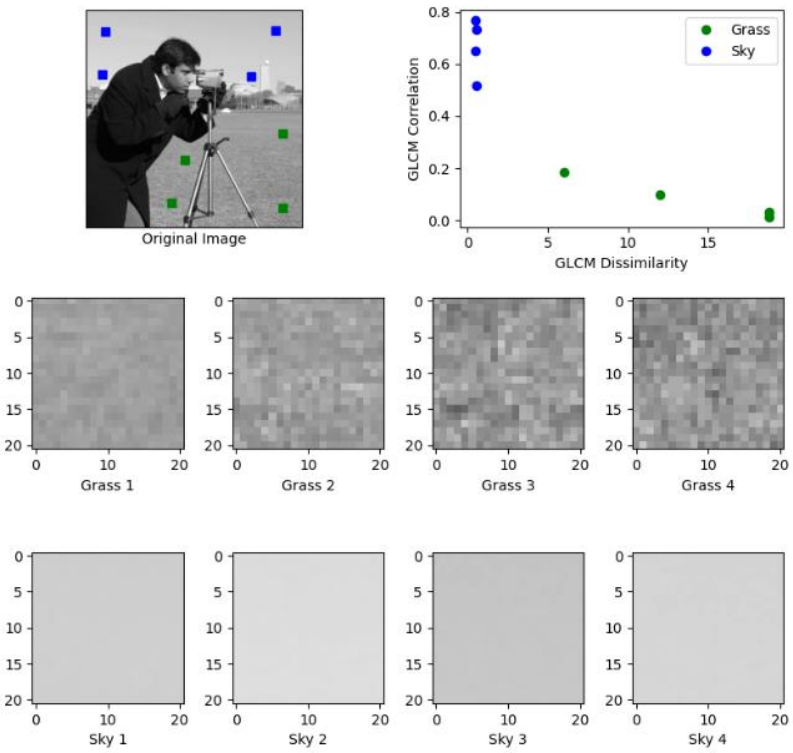

Figure 2.6 Mechanism of GLCM
(Source: Haralick, R. M., Shanmugam, K., & Dinstein, I. (1973). Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics, SMC-3*(6), 610–621. https://doi.org/10.1109/tsmc.1973.4309314)

2.3.1.7 Emperor Penguin Colony Algorithm (EPC)

The Emperor Penguin Colony Algorithm (EPC) is a metaheuristic optimisation method based on the behaviour of Antarctic emperor penguin colonies shown in figure 2.7. It solves complex optimisation issues by simulating penguin social behaviour such as foraging and huddling. In the context of plant disease classification techniques, EPC can be used to optimise machine learning model parameters such as feature selection,

hyperparameter tuning, and model optimisation. By leveraging the collective intelligence and cooperative behaviour of penguin colonies, EPC can improve the performance of plant disease classification models by efficiently searching for the best set of features and model configurations, resulting in improved disease classification accuracy and robustness.

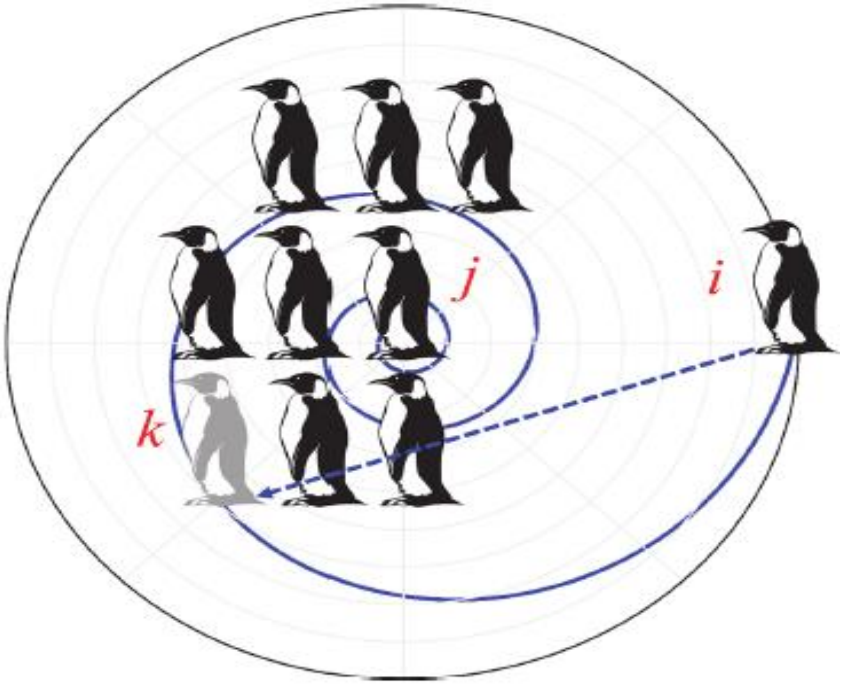

Figure 2.7 Mechanism of EPC
(Source: Wahdan, H. G., Abdelslam, H. E., Abou-El-Enien, T. H. M., & Kassem, S. (2020). Two-Modified Emperor Penguins Colony Optimization Algorithms. Revue D'intelligence Artificielle, 34(2), 151–160. https://doi.org/10.18280/ria.340205)

2.3.1.8 ResNet50

ResNet50 is a 50-layer deep convolutional neural network design noted refer to figure2.8 for using residual connections to overcome the vanishing gradient problem during training. ResNet50 can be used to handle and analyse complicated visual data in plant disease classification, learning to identify detailed patterns and features indicative

of various plant diseases. Its deep structure and residual connections enable it to learn with high accuracy from a large amount of picture data, making it a useful tool for recognising plant illnesses from photographs, which is critical for monitoring crop health and assuring agricultural productivity.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 2.8 Architecture of ResNet50
(Source: Kaushik, A. (2020b, July 21). Understanding ResNet50 architecture. Retrieved from https://iq.opengenus.org/resnet50-architecture/)

2.3.1.9 EfficientNetV2

EfficientNetV2 is an advanced neural network design that improves on its predecessor, EfficientNet shown in figure 2.9, by increasing training speed and optimising parameter efficiency. This is accomplished by employing progressive learning approaches and model scaling. EfficientNetV2 can be particularly effective in the field of plant disease classification due to its capacity to handle a wide variety of image resolutions and its improved efficiency, which allows for faster training times without sacrificing accuracy. EfficientNetV2 can learn to recognise subtle and complicated patterns associated with numerous plant illnesses by training on a varied set of plant

photos, making it a powerful tool for effectively classifying and diagnosing plant health issues.

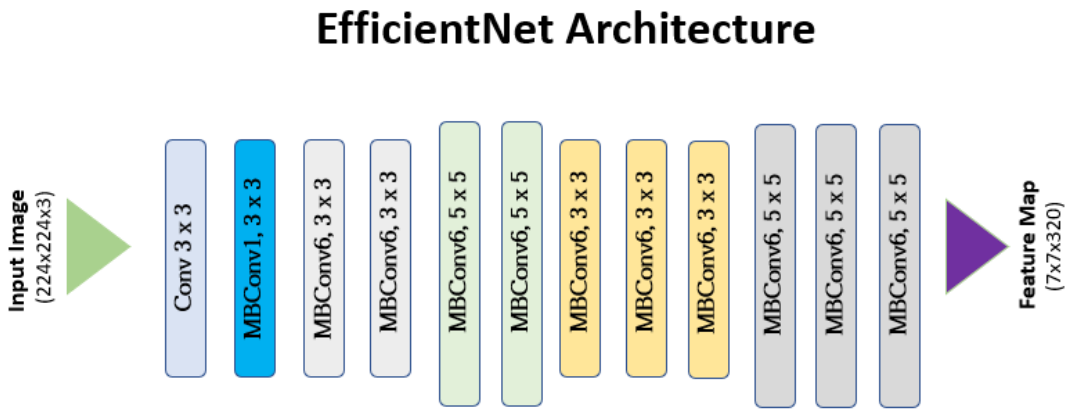

Figure 2.9 Architecture of EfficientNet
(Source: Wisdomml. (2023b, March 31). EfficientNet and its Performance Comparison with Other Transfer Learning Networks - Wisdom ML. Retrieved from https://wisdomml.in/efficientnet-and-its-performance-comparison-with-other-transfer-learning-networks/)

2.3.1.10 Light Weight 17-layer (LW17)

LW17 is a lightweight convolutional neural network (CNN) architecture designed for efficient image processing, especially in circumstances with limited computational resources. LW17's streamlined structure enables for the speedy and precise analysis of plant imaging, allowing for the detection and categorization of numerous plant diseases with minimal processing overhead. Its capacity to maintain high accuracy while remaining resource-efficient makes it well-suited for real-time applications and deployment on mobile or edge devices in agricultural situations, allowing for on-the-spot plant health detection.

2.3.1.11 Visual Geometry Group 16-layer neural network (VGG16)

VGG16 is a deep convolutional neural network (CNN) architecture noted for its simplicity and depth, distinguished by its 16 layers and extensive usage of 3x3 convolutional filters shown in figure 2.10. Because of its great performance on benchmark datasets, it has been widely used for image recognition tasks. VGG16 can be fine-tuned on specific datasets to identify and categorise diseases from photos of plant leaves in plant disease classification. Its deep architecture enables it to learn complex features from raw pixels, making it particularly successful at capturing the nuanced visual patterns indicative of diverse plant health conditions, so providing a robust framework for accurate disease identification in agriculture.

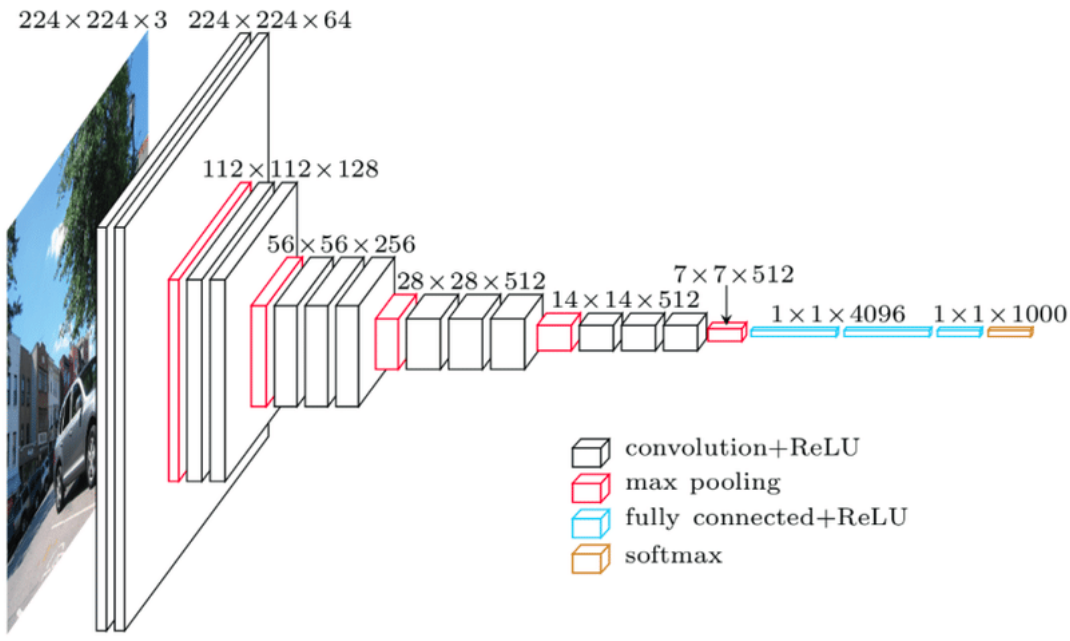

Figure 2.10 Architecture of VGG16
(Source: Understanding VGG16: Concepts, architecture, and performance. (2023, May 22). Retrieved from https://datagen.tech/guides/computer-vision/vgg16/#VGG16_Architecture)

2.3.1.12 DenseNet121

DenseNet121 is a feed-forward convolutional neural network design in which each layer is connected to every other layer the architecture for this can be seen in figure 2.11. DenseNet121's unique property of maximising information flow between layers makes it particularly efficient for plant disease classification. This is accomplished by reusing features via its dense connections, which can result in higher accuracy with fewer parameters when compared to alternative architectures. DenseNet121's efficiency and accuracy make it particularly well-suited for analysing complicated picture data to identify and categorise plant diseases, as it can collect a full range of features critical for recognising varied disease patterns on plant leaves.

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv; 3 × 3 conv] × 24	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 48	[1 × 1 conv; 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv; 3 × 3 conv] × 16	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Figure 2.11 Architecture of Densenet121
(Source: Ahmed, A. (2021, August 26). Architecture of DenseNet-121. Retrieved from https://iq.opengenus.org/architecture-of-densenet121/)

2.3.1.13 RGB and grayscale photos

The RGB colour model (Red, Green, Blue) is widely used to depict colour images, with each pixel represented by a mix of red, green, and blue colour channels. Grayscale photographs, on the other hand, are represented using a single channel, with pixel values indicating the intensity of light ranging from black to white. RGB photographs, in the context of plant disease classification approaches, contain detailed colour information that can be used to identify visual indications of disease, such as discoloration, lesions, and patterns on plant leaves. Grayscale photos, on the other hand, simplify image representation by focusing just on light intensity, which might be useful for extracting textural features and patterns linked with plant diseases.

2.3.1.14 K-means Clustering

K-means clustering is an unsupervised machine learning algorithm that divides a dataset into a set number of clusters. It operates by allocating data points to the nearest cluster centre iteratively and then updating the cluster centres depending on the mean of the given points. K-means clustering can be used to identify separate groups of plant samples based on visual criteria such as colour, texture, and form in the context of plant disease classification. K-means can uncover patterns and groups that may correspond to distinct disease kinds or severity levels by clustering comparable plant samples together. This can help with the early study and understanding of a dataset's diversity of plant diseases, providing useful insights for subsequent categorization approaches and disease control tactics.

2.4 Previous Works

This section is a careful and a thorough review of existing research on a certain issue. It synthesises and evaluates the findings of various studies to provide a clear picture of the field's present state of knowledge. In the topic of "machine classification techniques for plant disease," a review paper would investigate and summarise the numerous methodologies, algorithms, and technologies utilised for categorising and diagnosing plant diseases through machine learning and computer vision. It would also explore the methodologies' strengths, limits, and future directions, giving significant insights for agricultural and plant science researchers.

2.4.1 An Approach to Identify and Classify Agricultural Crop Diseases Using Machine Learning and Deep Learning Techniques (Ahmed et al., 2023)

This article, compares several machine learning and deep learning algorithms for recognising and classifying plant diseases from photos. On a plant disease dataset, the paper analyses the accuracy of multiple approaches, including KNN, SVM, CNN, and GLCM texture features. The Inception v3 transfer learning model scored the highest accuracy of 97%, and CNN models beat classical ML models in general display in table 2.1. The document provides a current and relevant examination of machine learning approaches for agricultural crop disease diagnosis, providing useful insights into the performance of several algorithms in this domain.

Table 2.1 Performance Analysis of algorithm used(Ahmed et al., 2023)

Algorithm	Accuracy	Precision	Recall
Inception v3	97%	95.6%	95.14%
VGG 16	95%	93.35%	94.55%
ResNet 50	94%	90.5%	90.15%
Support Vector Machine	98.2%	91.6%	94.17%
Random Forest	89.3%	88.1%	93.12%
CNN	87%	86.2%	92.35%

2.4.2 Apple and Tomato Leaves Disease Detection using Emperor Penguins Optimizer based CNN (Suguna et al., 2023)

This publication proposes a viable system for detecting plant diseases using cutting-edge deep learning algorithms. To detect and categorise damaged leaves in apple and tomato plants the results of apple is shown in table 2.2 while the result of tomato is shown in table 2.3, the researchers used a YOLOv4 object detection network and a convolutional neural network (CNN) optimised with an Emperor Penguin Colony algorithm. Pre-processing the photos, detecting the leaves with YOLOv4, classifying the leaves with CNN, and optimising the CNN learning rate with the EPC algorithm are all part of the methodology. The proposed method is tested on publicly available datasets and found to be more accurate than other baseline methods. The research contributes to the field of agricultural technology by providing significant insights into the implementation of advanced deep learning techniques for plant disease diagnosis.

Table 2.2 Performance Analysis of Proposed Method on Apple(Suguna et al., 2023)

Method	Parameter Evaluation			
	Accuracy (%)	Precision (%)	Recall (%)	F-measure (%)
CNN	71.52	81.54	82.21	87.11
EPCO-CNN	82.01	81.99	91.16	88.24
Yolo4-PSO-CNN	85.16	81.04	85.17	83.08
Yolo4-BAT-CNN	88.89	79.12	80.92	85.27
Yolo4-WOA-CNN	94.38	95.43	96.46	96.34
Yolo4-EPCO-CNN	**96.90**	**97.84**	**98.20**	**98.67**

Table 2.3 Performance Analysis of Proposed Method on Tomato(Suguna et al., 2023)

Method	Parameter Evaluation			
	Accuracy (%)	Precision (%)	Recall (%)	F-measure (%)
CNN	86.22	79.48	85.78	88.73
EPCO-CNN	87.45	82.73	88.36	88.46
Yolo4-PSO-CNN	79.86	82.07	90.06	91.45
Yolo4-BAT-CNN	84.64	83.63	91.67	85.33
Yolo4-WOA-CNN	92.09	93.43	94.98	95.63
Yolo4-EPCO-CNN	**96.59**	**96.45**	**97.44**	**98.34**

2.4.3 Crop Recommendation using Machine Learning and Plant Disease Identification using CNN and Transfer-Learning Approach (Tiwari et al., 2022)

The paper describes a dual-purpose agriculture system that uses machine learning to recommend crops and convolutional neural networks (CNN) to identify plant diseases. The study compares three CNN architectures for disease identification (VGG16, ResNet50, and EfficientNetV2), with EfficientNetV2 outperforming the others with a test

accuracy of 96.06% display in table 2.4. The models are deployed in an easy-to-use web application built on the Flask/Docker/MySQL architecture shown in figure 2.12. Despite their positive results, the authors emphasise the need for future improvements, such as incorporating advanced transformer models, increasing dataset diversity, and improving the web application's frontend and load balancing capabilities.

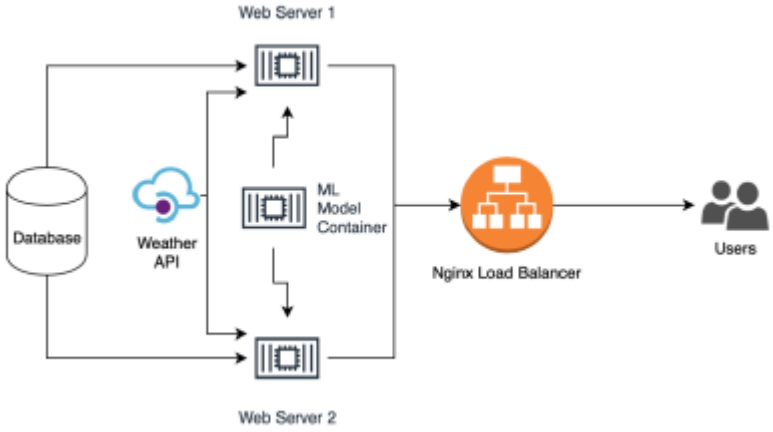

Figure 2.12 System architecture(Tiwari et al., 2022)

Table 2.4 Plant Disease Classification Accuracy to Various Architectures (Tiwari et al., 2022)

Architecture	Training Acc.	Validation Acc	Test Accuracy
VGG16	92.18	91.33	91.78
ResNet50	96.02	95.41	95.53
EfficientNetV2	96.06	95.53	95.83

2.4.4 Detection of rice plant disease from RGB and grayscale images using an LW17 deep learning model (Rathore et al., 2023)

The article offers steps to build LW17 model for detecting rice plant illnesses from RGB and grayscale photos. The study makes use of a dataset from the UCI Machine Learning Repository that has been modified to improve the model's consistency. The

LW17 model is painstakingly optimized across multiple parameters and achieves a noteworthy accuracy of 93.75%, outperforming other pre-trained CNN models while preserving a more simplified architecture.

2.4.5 Rice Leaf Disease Recognition using Local Threshold Based Segmentation and Deep CNN(Islam et al., 2021)

The article describes a complete methodology for detecting rice leaf illnesses that employs local threshold-based segmentation and deep convolutional neural networks (CNNs). To classify rice diseases from leaf photos, three CNN architectures (VGG16, ResNet50, and DenseNet121) are thoroughly assessed on three datasets, including one gathered by the authors. The procedure begins with image preparation (background removal and scaling), followed by disease region segmentation using local thresholding. Following that, the segmented images are used to train and evaluate the CNN models. Experiment findings on the three datasets show that DenseNet121 achieves the greatest classification performance, and segmentation improves accuracy greatly when compared to using whole images.

2.4.6 Infectious diseases of Rice plants classified using a deep learning-powered Least Squares Support Vector Machine Model(Goluguri et al., 2022)

A deep learning-powered least squares support vector machine (LSSVM) model is used in this study to classify rice plant illnesses. Advanced picture pre-processing techniques such as histogram equalisation and Kuwahara filtering are used in the

methodology, which is then followed by spatial fuzzy C-means segmentation. A multi-model LeNet5 CNN is used for feature extraction, and an LSSVM is used for classification, which is optimised using a weighted sparrow search optimisation technique. On two public datasets, the proposed LSSVM-WSSO model achieves above 98% accuracy, precision, F1-measure, and recall for several rice illnesses. The methods and findings of the study are well-documented, ensuring reproducibility and confirming the model's superiority over competing categorization methodologies.

2.4.7 Machine Learning for Plant Leaf Disease Detection and Classification – A Review(Adhiparasakthi Engineering College et al., n.d.)

The paper discusses various machine learning algorithms for identifying and detecting bacterial, fungal, and viral illnesses in plant leaves. Fuzzy C-means clustering, K-means clustering, and principal component analysis are among the unsupervised techniques addressed. K-nearest neighbours, artificial neural networks, probabilistic neural networks, support vector machines, radial basis function networks, random forests, and decision trees are among the supervised techniques covered. Future research will concentrate on detecting diseases in mulberry plant leaves using convolutional neural networks and improving identification rates utilising hybrid algorithms.

2.4.8 Performance Analysis of Real-Time Object Detection algorithm for a Multi-Class Plant Disease Detection and Classification Using Deep Learning (Shah et al., 2023)

The research presents a real-time plant disease detection system based on the YOLOv5 and YOLOv7 object detection algorithms. YOLOv5 architecture is shown in figure 2.13 while the YOLOv7 architecture is shown in figure 2.14. The Plantdoc dataset was used to train the system, which contains photos of 28 different plant kinds, both healthy and diseased. To improve the dataset, the photos were pre-processed utilising data augmentation techniques such as random scaling, cropping, and flipping. Using the expanded dataset, the models were trained across 55 epochs. The models' training times differed greatly, with YOLOv7 requiring around 2 hours and 33 minutes and YOLOv5 lasting approximately 1 hour and 3 minutes. YOLOv5 surpassed YOLOv7 in terms of mean average precision (mAP), with a 62% mAP versus 42% for YOLOv7.mAP levels in YOLOv7 were more variable across epochs than in YOLOv5.

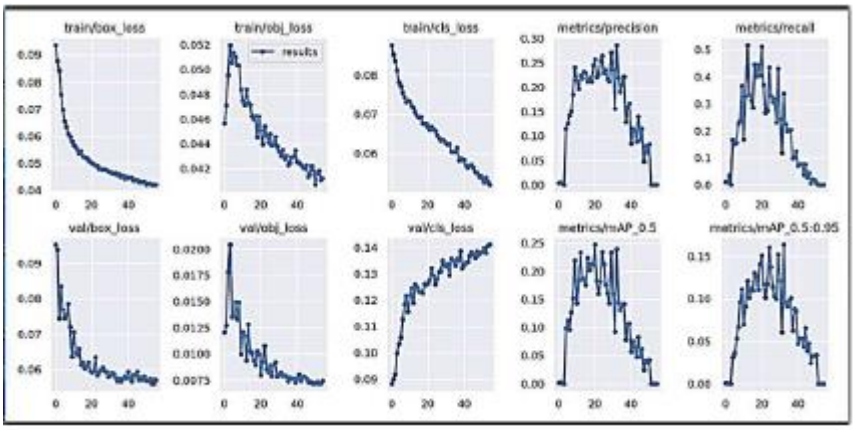

Figure 2.13 Yolov5 Architecture (Shah et al., 2023)

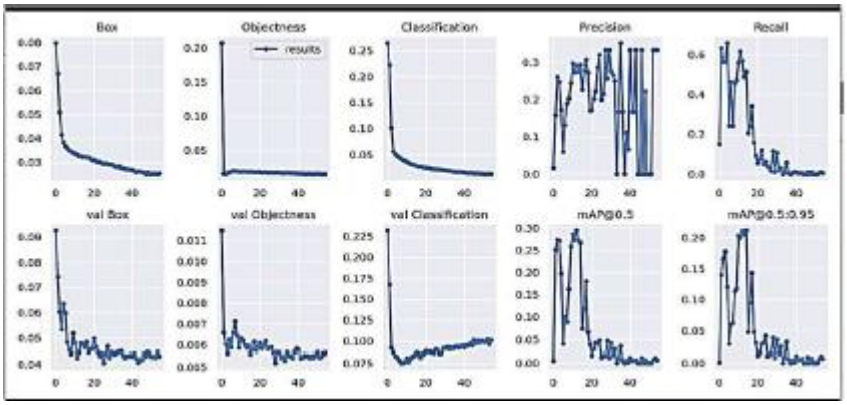

Figure 2.14 Yolov7 Architecture (Shah et al., 2023)

2.5 State of The Art Table (SOTA)

SOTA refers to advanced machine learning-based categorization algorithms used to diagnose plant diseases. The subject matter covers the most advanced and effective solutions accessible at any given time, which are always improving as new algorithms, feature extraction approaches, and dataset developments contribute to the field.

No	Research Article	Method Used	Results	Limitation
1	An Approach to Identify and Classify Agricultural Crop Diseases Using Machine Learning and Deep Learning Techniques (Ahmed et al., 2023)	- KNN - SVM - CNN - GLCM	Inception v3 transfer learning model scored the highest accuracy of 97%, and CNN models beat classical ML models.	Not Stated

2	Apple and Tomato Leaves Disease Detection using Emperor Penguins Algorithm based CNN (Suguna et al., 2023)	- YOLOv4 - CNN - EPC	EPC algorithm speed up the learning rate of CNN.	Tested on public dataset which can be alter anytime. Limit to only using CNN.
3	Crop Recommendation using Machine Learning and Plant Disease Identification using CNN and Transfer-Learning Approach(Tiwari et al., 2022)	- CNN - VGG16 - ResNet50 - EffecientNetV2	EffecientNetV2 outperforms the other architecture.	Poor web application's frontend and load balancing capabilities. Need incorporating advanced transformer models and increasing dataset diversity.
4	Detection of rice plant disease from RGB and grayscale images using an LW17 deep learning model (Rathore et al., 2023)	- LW17(CNN) - RGB and grayscale photos	Outperforming other pre-trained CNN models with a simplified architecture.	LW17 very hard to optimize and time consuming.
5	Rice Leaf Disease Recognition using Local Threshold Based Segmentation and Deep CNN (Islam et al., 2021)	- CNN - VGG16 - ResNet50 - DenseNet121	DenseNet121 Outperform other pre-trained CNN models	Can be improved with more data.
6	Infectious diseases of Rice plants classified using a deep learning-powered Least Squares Support Vector Machine Model(Goluguri et al., 2022)	- LSSVM	Achieves above 98% accuracy and precision.	Test on benchmark data only.

7	Machine Learning for Plant Leaf Disease Detection and Classification – A Review (Adhiparasakthi Engineering College et al., n.d.)	- K-means clustering	Able to detect bacterial, fungal, and viral illnesses in plant leaves.	Only use one type of machine learning.
8	Performance Analysis of Real-Time Object Detection algorithm for a Multi-Class Plant Disease Detection and Classification Using Deep Learning (Shah et al., 2023)	- YOLOv5 - YOLOv7	YOLOv7 outperforms YOLOv5 in various aspects but not consistent.	Only use limited dataset. The algorithm is only single stage object detector.

2.6 Summary

To summarize, the above SOTA table validates that to build a good machine model for plant diseases classification include a combination of many machines learning algorithm or only one learning algorithm with good optimization depending on the criteria or task to be achieved by that model. The following chapter, will look at how to build a plant disease classification model utilising SOTA insights derived from previous works. The approaches used include proximity-based methods such as KNN, ensemble learning with Random Forest, and deep learning via CNN. The following chapter will go into greater detail about the algorithms employed.

CHAPTER 3 : METHODOLOGY

3.1 Introduction

This chapter provides a brief explanation of the research methodology used to accomplish the results presented in ML classification techniques for plant disease detection. Before getting into the process flow, it is critical to recognise the tools and libraries that were crucial in the building of the machine learning model. The project was produced using Google Colab, a platform that provides a consistent environment for developing and executing Python code that is both accessible and powerful for ML tasks. Furthermore, the dataset was obtained from Kaggle, a platform that stores datasets and facilitates data science competitions. A variety of specialised software and libraries were also used, including NumPy for numerical operations, Pandas for data manipulation, Seaborn and Matplotlib for data visualisation, TensorFlow and Keras for neural network model building and training, Scikit-learn for model evaluation and selection, scikit-image for image processing, and Python as the primary programming language. These tools were used to manage data, create models, evaluate performance, and deploy the final models. The process flow supported by figure 3.12 is taken to create the machine model for plant disease detection: Data collection and preprocessing, data preparation, model selection, model training, model evaluation, model deployment, and continual improvement are all part of the modelling process.

3.2 Tools and Software Libraries

The project's tools and libraries are briefly described below.

Figure 3.1 Google Colab

Google Colab is a platform for developing and executing Python code in a consistent environment. It is especially effective for ML applications due to its accessibility and powerful processing capabilities.

Figure 3.2 Kaggle website

Kaggle is a website that hosts datasets and facilitates data science competitions, giving users access to a diverse range of datasets and tools for data analysis and ML projects.

Figure 3.3 Python

The principal programming language used in the project, noted for its simplicity, readability, and versatility, is extensively utilised in a variety of fields such as data analysis, ML, web development, and others.

Figure 3.4 NumPy

NumPy is a foundational Python module for numerical operations that includes support for huge, multidimensional arrays and matrices, as well as a set of mathematical functions for array manipulation.

Figure 3.5 Pandas

Pandas is a Python data manipulation and analysis toolkit that provides data structures and functions for working with structured data, including data frames, series, and time series data.

Figure 3.6 Matplotlib and Seaborn

Seaborn and Matplotlib are Python libraries for data visualisation. Matplotlib is a comprehensive toolkit for building static, interactive, and animated visualisations, whereas Seaborn is a high-level interface for making visually appealing and useful statistical visuals built on Matplotlib.

Figure 3.7 Scikit-learn

Scikit-learn is a versatile Python machine learning framework that provides easy and efficient tools for data mining and analysis. It includes several methods for classification, regression, clustering, dimensionality reduction, and model selection.

Figure 3.8 Scikit-image

Scikit-image is a Python-based library of image processing methods that includes tools for image alteration, segmentation, feature extraction, and restoration.

Figure 3.9 TensorFlow and Keras

TensorFlow is an open-source machine learning framework created by Google to design and train machine learning models, particularly deep learning models. Keras is a high-level neural network API written in Python that runs on top of TensorFlow, making it easier to create and train deep learning models.

Figure 3.10 Streamlit logo

Streamlit is an open-source Python toolkit that allows you to easily construct and distribute beautiful, unique web apps for machine learning and data research. Streamlit was quite useful in this project, at which created a web-based application for diagnosing maize illnesses using machine classification algorithms. It enabled the ability to swiftly create an interactive and user-friendly interface for my machine learning model, allowing users to upload maize plant photos, run the disease detection algorithm, and seamlessly examine the results. Streamlit's simplicity and robust features helped speedy development and deployment, making it an excellent choice for demonstrating the project's potential and effectively engaging users.

Figure 3.11 Ngrock logo

Ngrok is a robust tool for creating secure tunnels to localhost, allowing you to connect a local server to the internet. Ngrok played an important role in this project, which involved developing a web-based tool for detecting maize illnesses using machine classification algorithms. This allowed for real-time demonstrations and feedback, expedited development workflows, and insured that my application could be easily accessed and tested from anywhere with security.

3.3 Machine Learning Model Development

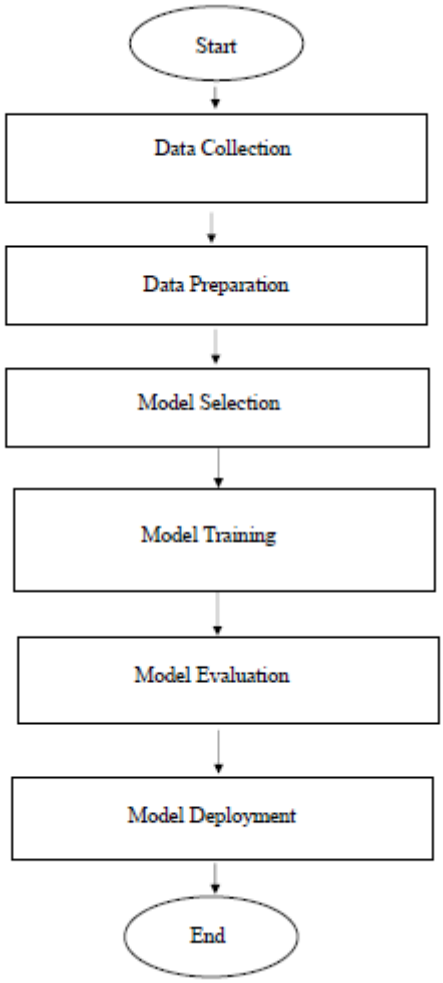

Figure 3.12 Process Flow of Building a Machine Model

Data Collection

The Kaggle website was used as a beneficial platform for research on ML classification techniques for plant disease, with a specific focus on the Corn or Maize Leaf Disease Dataset. This collection is drawn from the popular Plant Doc and Plant Village datasets and contains photos labelled with distinct categories showed in figure 3.13.

The categories include:

- Category 0: Common Rust, with 1306 images
- Category 1: Gray Leaf Spot, with 574 images
- Category 2: Blight, with 1146 images
- Category 3: Healthy, with 1162 images

These categories provide a thorough picture of several diseases as well as the overall health of maize or maize plants. The diversified and well-labelled dataset allows the ability to create a strong model for plant disease detection using effective machine learning classification techniques. The user-friendly interface of Kaggle, as well as the availability of such high-quality datasets, have tremendously aided research process, allowing for a methodical investigation of the subtle links between picture data and disease categorization in the context of plant pathology.

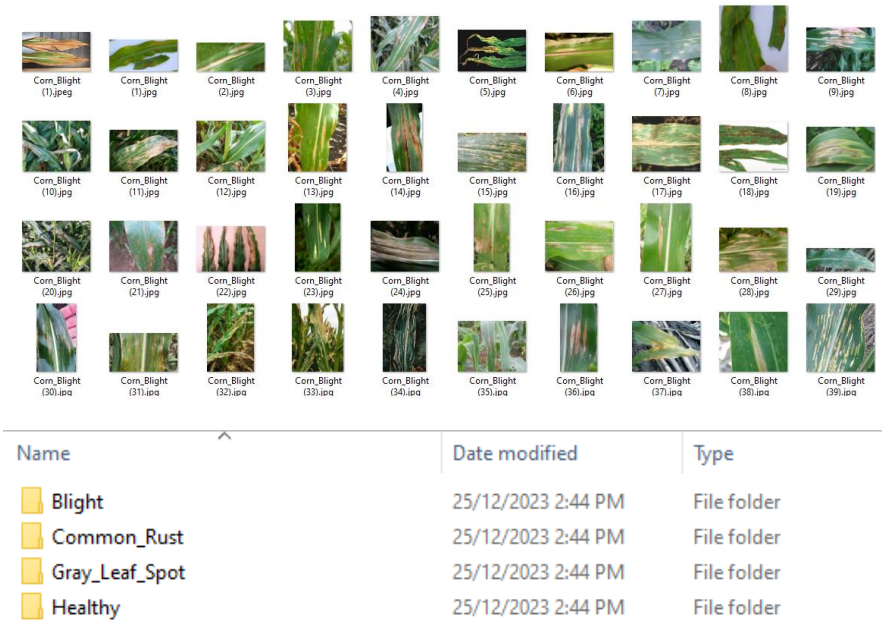

Figure 3.13 The dataset class and the images in each class
(Source: Corn or maize leaf disease dataset. (2020, November 11). Retrieved from https://www.kaggle.com/datasets/smaranjitghose/corn-or-maize-leaf-disease-dataset/data)

Data Preprocessing

Data preprocessing emerges as a significant component of the research on Machine Learning Classification Techniques for Plant Disease, which makes use of the Kaggle website's Corn or Maize Leaf Disease Dataset. This dataset includes photographs classified as Common Rust, Grey Leaf Spot, Blight, and Healthy, each with a different number of images—1306 for Common Rust, 574 for Grey Leaf Spot, 1146 for Blight, and 1162 for Healthy. An effective preprocessing procedure was carried out before to digging into the fundamental categorization techniques shown in figure 3.14. The dataset was divided into three major categories: training, testing, and validation. Furthermore, all photos in the dataset were reduced to (264,264) pixels, which was an important step in standardising input dimensions for the machine learning model.

Moreover, a batch size of 32 was chosen to allow for quick processing and training of the model on subsets of the data prove with the help of figure 3.15. This thorough preprocessing not only ensures the dataset's integrity but also optimises it for subsequent machine learning model training.

```
import numpy as np
import pandas as pd
import seaborn as sns
import tensorflow as tf
from tensorflow.keras import models,layers
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from skimage.transform import resize
```

```
IMAGE_SIZE = (264,264)
BATCH_SIZE = 32
```

Figure 3.14 Importing the necessary libraries and resizing the image

```
def get_dataset(ds,train_split=0.8,val_split=0.1,test_split=0.1,shuffle=True,shuffle_size=10000):
    ds_size = len(ds)

    if shuffle:
        ds = ds.shuffle(shuffle_size,seed=8)

    train_size = int(train_split * ds_size)
    val_size = int(val_split * ds_size)

    train_ds = ds.take(train_size)

    val_ds = ds.skip(train_size).take(val_size)
    test_ds = ds.skip(train_size).skip(val_size)

    return train_ds,val_ds,test_ds
```

Figure 3.15 Splitting the data categories training, testing and validation

Model Selection

An important stage in this study on Machine Learning Classification Techniques for Plant Disease Using the Kaggle Corn or Maize Leaf Disease Dataset is the selection of relevant models to ensure reliable disease identification.

For proximity-based classification, the K-Nearest Neighbours (KNN) method with a k-value of 5 was used showed in figure 3.16. In addition, the Random Forest technique, which is characterised by an ensemble of decision trees, was combined with 100 trees to improve resilience showed in figure 3.17. A Convolutional Neural Network (CNN) was created to capture intricate spatial hierarchies within visual data supported by figure 3.18. Convolutional layers for feature extraction, max-pooling layers for dimensionality reduction, and fully linked layers for classification composed the CNN architecture.

Deep learning was used to identify complicated patterns and relationships in photos. Each algorithm was trained on the pre-processed dataset to ensure that it could correctly classify plant diseases based on visual cues. This combination of KNN, Random Forest, and CNN approaches demonstrates a comprehensive strategy for achieving robust plant disease categorization using both classic and deep learning techniques.

```
# Extract labels from the training dataset
train_labels = np.concatenate([y.numpy() for _, y in train_ds], axis=0)

# K-Nearest Neighbors (KNN) Model
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(train_features, train_labels)
```

```
KNeighborsClassifier
KNeighborsClassifier()
```

Figure 3.16 K-Nearest Neighbour initialization

```
# Random Forest
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

# Convert the labels in train_ds to a NumPy array
train_labels = np.concatenate([y.numpy() for _, y in train_ds], axis=0)

# Fit the Random Forest model
rf_model.fit(train_features, train_labels)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

Figure 3.17 Random Forest initialization

```
n_classes = 4
input_shape = (BATCH_SIZE,264,264,3)
model = models.Sequential([
    resize_and_rescale,
    layers.Conv2D(32,(3,3),activation ='relu',input_shape = input_shape),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64,kernel_size = (3,3),activation ='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64,kernel_size = (3,3),activation ='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64,kernel_size = (3,3),activation ='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64,kernel_size = (3,3),activation ='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64,kernel_size = (3,3),activation ='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Flatten(),
    layers.Dense(64,activation='relu'),
    layers.Dense(n_classes,activation='softmax'),
])
model.build(input_shape=input_shape)
```

Figure 3.18 CNN initialization

Model Training

Each algorithm was fine-tuned during the model training phase to successfully learn from the pre-processed information and produce accurate predictions. The K-Nearest Neighbours (KNN) model was trained with a k-value of 5 by fitting the extracted features and labels from the training dataset. The KNN algorithm classified new instances in the feature space based on the majority class of their k-nearest neighbours.

Concurrently, the Random Forest model (100 decision trees) was trained using the extracted characteristics and labels. Random Forest's ensemble nature enabled the model to generalise well and provide solid predictions by combining the outputs of several decision trees.

The Convolutional Neural Network (CNN), on the other hand, was trained using the TensorFlow and Keras frameworks. The design of the CNN, which consists of convolutional, pooling, and dense layers, was optimised to learn hierarchical features from picture input. The Adam optimizer, sparse categorical cross entropy loss, and accuracy as the metric were used to build the model. The training dataset was used for 15 epochs of training, with performance on the validation set monitored to prevent overfitting and ensure generalisation proved with figure 3.19.

These training procedures attempted to iteratively alter the parameters of each algorithm in order to reduce classification error and improve their ability to effectively identify and distinguish between various plant disease.

```
[ ]:
model.compile(
optimizer='adam',
loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
metrics=['accuracy']
)
```

MODEL FITTING

```
[ ]:
history = model.fit(
train_ds,
epochs=15,
batch_size=BATCH_SIZE,
verbose=1,
validation_data=val_ds)
```

Figure 3.19 CNN Training to fit the model

Model Evaluation

Following model training, a detailed evaluation was performed to analyse each algorithm's performance and generalisation capabilities. To evaluate the K-Nearest

Neighbours (KNN) model, labels on the test dataset were predicted using the trained model and compared to the ground truth labels. Metrics including accuracy, precision, recall, and F1-score were most likely used to assess the model's overall performance and ability to accurately categorise occurrences across disease classifications shown in figure 3.20.

```
# Convert validation dataset to NumPy arrays
val_true_labels = np.concatenate([y.numpy() for _, y in val_ds], axis=0)

# KNN predictions
knn_predictions = knn_model.predict(val_features)

# Calculate accuracy and display the classification report
print("KNN Accuracy on Validation Data:", accuracy_score(val_true_labels, knn_predictions))
print("KNN Classification Report on Validation Data:")
print(classification_report(val_true_labels, knn_predictions))
```

Figure 3.20 Evaluation of KNN

Similarly, the Random Forest model was evaluated on the test set, with its ensemble of decision trees making predictions and comparing the findings to the true labels. The evaluation metrics revealed information about the model's correctness and efficacy in handling various cases in the test dataset proven with figure 3.21.

```
# Evaluate Random Forest model
rf_predictions = rf_model.predict(val_features)

# Convert the labels in val_ds to a NumPy array
val_labels = np.concatenate([y.numpy() for _, y in val_ds], axis=0)

print("\nRandom Forest Accuracy on Validation Data:", accuracy_score(val_labels, rf_predictions))
print("Random Forest Classification Report on Validation Data:")
print(classification_report(val_labels, rf_predictions))
```

Figure 3.21 Evaluation of Random Forest

The Convolutional Neural Network (CNN) was evaluated by making predictions on the test set and analysing its performance metrics. The CNN's capacity to capture

subtle spatial patterns and characteristics inside images was assessed using metrics such as accuracy and perhaps confusion matrices to analyse class-wise performance.

Comparisons between these three algorithms were most likely made using their respective assessment measures, which provided a comprehensive understanding of their strengths and drawbacks in the context of plant disease classification. The purpose was to find the best effective algorithm for the given task and dataset, taking into account accuracy, computing efficiency, and interpretability. The insights provided during the model evaluation phase influenced model selection and potential areas for further optimisation.

Model Deployment

Figure 3.22 Streamlit used to build app.py file

Following a thorough study and selection of the best-performing model for plant disease classification, the deployment phase entailed developing a web-based application with Streamlit and Ngrok. Streamlit was used to create an interactive and

user-friendly interface that allows users to input photographs of maize plants and receive illness diagnosis instantly using the trained machine learning model shown in figure 3.22. The application displayed the model's predictions, performance indicators, and input data visualisations, offering users an intuitive interaction.

Figure 3.23 Ngrock used to deploy app.py to local host

Ngrok played an important role in the deployment process by providing secure tunnels to localhost, allowing the locally hosted Streamlit application to be accessed from anywhere on the internet proven with figure 3.23. This enabled real-time demos, remote testing, and user response without having a public server. By combining Streamlit's ease of constructing interactive web apps with Ngrok's secure tunnelling capabilities, the deployment phase guaranteed that the application was both easy to use and broadly available, allowing for successful testing and participation. This configuration also provided a solid foundation for demonstrating the project's capabilities and fine-tuning the model depending on user interactions.

3.4 Summary

In this chapter, presented the study methods used for ML classification algorithms in plant disease detection. The method included data collection and preprocessing, model selection, training, and evaluation. The Kaggle website proved to be an invaluable resource, particularly for the Corn or Maize Leaf Disease Dataset, which was derived from the Plant Doc and Plant Village databases.

The data collecting phase includes obtaining photos labelled with several categories, such as Common Rust, Grey Leaf Spot, Blight, and Healthy, to provide a thorough overview of maize plant diseases. Data preprocessing was critical, including classification, image resizing, and batch size selection, to ensure dataset integrity and optimise it for future machine learning model training.

Model selection employed proximity-based (KNN), ensemble (Random Forest), and deep learning (CNN) methods. Each algorithm was trained on the pre-processed dataset to achieve reliable plant disease classification.

The model training step refined algorithms to learn from pre-processed data. The KNN and Random Forest models were trained with specified parameters, while the CNN was trained utilising TensorFlow and Keras frameworks, with performance monitored to prevent overfitting.

Following model training, a rigorous model evaluation was performed to assess accuracy, precision, recall, and F1-score for KNN and Random Forest models. The CNN's performance was assessed using measures such as accuracy and confusion matrices. Comparisons of the three algorithms revealed their strengths and flaws, which influenced model selection and identified possible areas for optimisation.

The model deployment phase entailed developing a web-based application with Streamlit and Ngrok to make the chosen machine learning model available to users.

In the following chapter, we will look at the findings of the model evaluation phase, offering a thorough analysis of each algorithm's performance and implications for plant disease classification.

CHAPTER 4 : RESULTS

4.1 Introduction

This chapter presents the findings and discussions from the study on ML classification methods for plant disease detection. Investigate the usefulness of proximity-based (KNN), ensemble (Random Forest), and deep learning (CNN) methods in diagnosing plant diseases using a thorough model selection, training, and evaluation process. By thoroughly examining parameters such as accuracy, precision, recall, and F1-score, one can comprehend the performance intricacies of each algorithm, providing insights into its strengths and places for improvement.

4.2 Experimental Results

4.2.1 KNN results

```
KNN Accuracy on Validation Data: 0.2980769230769231
KNN Classification Report on Validation Data:
              precision    recall  f1-score   support

           0       0.31      0.41      0.35       118
           1       0.31      0.40      0.35       117
           2       0.09      0.05      0.06        62
           3       0.33      0.22      0.26       119

    accuracy                           0.30       416
   macro avg       0.26      0.27      0.26       416
weighted avg       0.28      0.30      0.28       416
```

Figure 4.1 KNN results

The (KNN) technique on the validation data had an accuracy of roughly 29.81%. The classification report provides further information on each class's precision, recall, and F1-score. For class 0, which represents one type of sickness, the precision is 31%, recall is 41%, and F1-score is 35%. Similarly, in class 1, the precision is 31%, recall is 40%, and F1-score is 35%. Class 2 has lower precision (9%), recall (5%), and F1-score (6%), showing difficulty in accurately identifying this category. Class 3 has a precision of 33%, recall of 22%, and an F1-score of 26 percent. The total weighted average precision, recall, and F1-score are 28%, 30%, and 28%, respectively, demonstrating the model's performance across all classes. The macro average statistics provide an average assessment across all classes, with precision, recall, and F1-scores averaging around 26%. These results support by figure 4.1 show the various degrees of effectiveness in disease classification produced by the KNN algorithm.

4.2.2 Random Forest Results

```
Random Forest Accuracy on Validation Data: 0.28365384615384615
Random Forest Classification Report on Validation Data:
              precision    recall  f1-score   support

           0       0.26      0.23      0.24       118
           1       0.29      0.40      0.34       117
           2       0.21      0.13      0.16        62
           3       0.33      0.30      0.31       119

    accuracy                           0.28       416
   macro avg       0.27      0.27      0.26       416
weighted avg       0.28      0.28      0.28       416
```

Figure 4.2 Random Forest Results

The Random Forest method performed poor on the validation data, with an accuracy of about 28.37%. The categorization report shows the precision, recall, and F1-score for each disease class. For class 0, the precision is 26%, the recall is 23%, and the F1-score is 24%. Similarly, in class 1, precision is 29%, recall is 40%, and the F1-score is 34%. Class 2 shows a precision of 21%, recall of 13%, and F1-score of 16%, but class 3 shows a precision of 33%, recall of 30%, and F1-score of 31%. The macro average precision, recall, and F1-score are around 27%, showing mediocre overall performance across all classes. The weighted average precision, recall, and F1-score are all around 28%, indicating the model's performance despite the class imbalance depict in figure 4.2.

4.2.3 CNN Results

```
Epoch 15/15
105/105 [==============================] - 470s 4s/step - loss: 0.1556 - accuracy: 0.9440 - val_loss: 0.1852 - val_accuracy: 0.9351
```

```
scores = model.evaluate(test_ds)
```

```
14/14 [==============================] - 42s 1s/step - loss: 0.2311 - accuracy: 0.9196
```

Figure 4.3 CNN results

The CNN performed well after training support by figure 4.3, with an accuracy of roughly 94.40% on training data and 93.51% on validation data. The loss dropped gradually over the 15 epochs of training to 0.1556 on the training set and 0.1852 on the validation set. This shows that the model efficiently learned the features from the training data and generalised well to previously unseen validation data. During the evaluation phase, the CNN performed well, with an accuracy of 91.96% and a loss of 0.2311. These findings indicate that the CNN model successfully learned to classify plant diseases with high accuracy, indicating its use in this application.

4.3 Analysis of Results

4.3.1 Analysis Of KNN Results

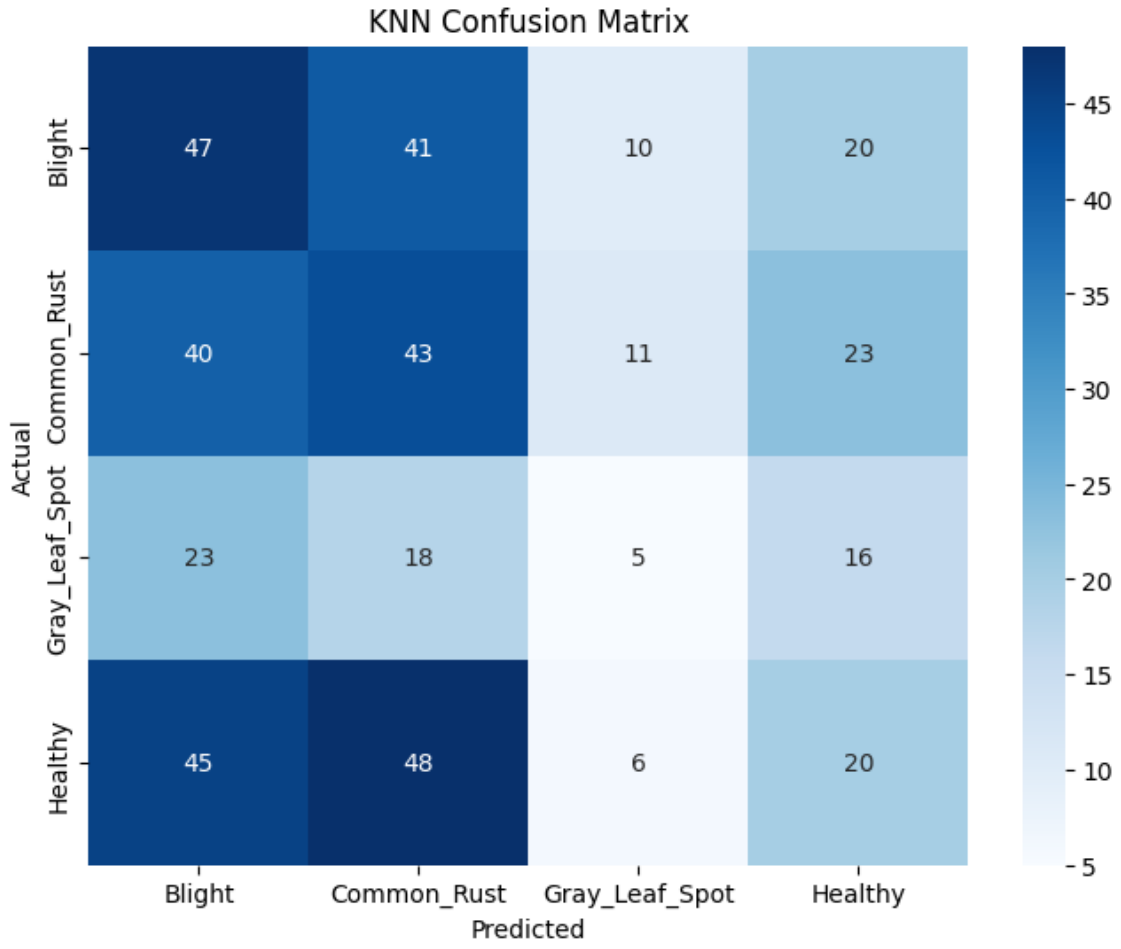

Figure 4.4 KNN Confusion matrix

The presented confusion matrix via figure 4.4 depicts the performance of the (KNN) algorithm in classifying plant diseases. The rows indicate the actual classes (Healthy, Grey Leaf Spot, Common Rust, and Blight), whilst the columns show the expected classes. Each cell in the matrix represents the number of cases classified accordingly. For example, in the cell corresponding to Healthy (actual) and Healthy (predicted), 47 cases are accurately classified as Healthy. In the cell corresponding to

Grey Leaf Spot (actual) and Healthy (predicted), 40 cases were misclassified as Healthy. Analysing this matrix reveals information about the algorithm's performance, highlighting areas of accurate classification and misclassification for each disease category.

4.3.2 Analysis Of Random Forest Results

The Random Forest confusion matrix provides a detailed look into the algorithm's classification performance across multiple disease categories. The rows of the matrix indicate the actual classes (Healthy, Grey Leaf Spot, Common Rust, and Blight), and the columns represent the anticipated classes. Each cell in the matrix represents the number of cases classified accordingly. For example, in the cell corresponding to Healthy (actual) and Grey Leaf Spot (predicted), 27 cases have been misclassified as Grey Leaf Spot. Similarly, the cell at the intersection of Blight (actual) and Common Rust (predicted) has 20 occurrences that were misclassified as Common Rust. Consider the above matrix to identify patterns of accurate categorization and regions where the algorithm does not succeed proven by the figure 4.5 below.

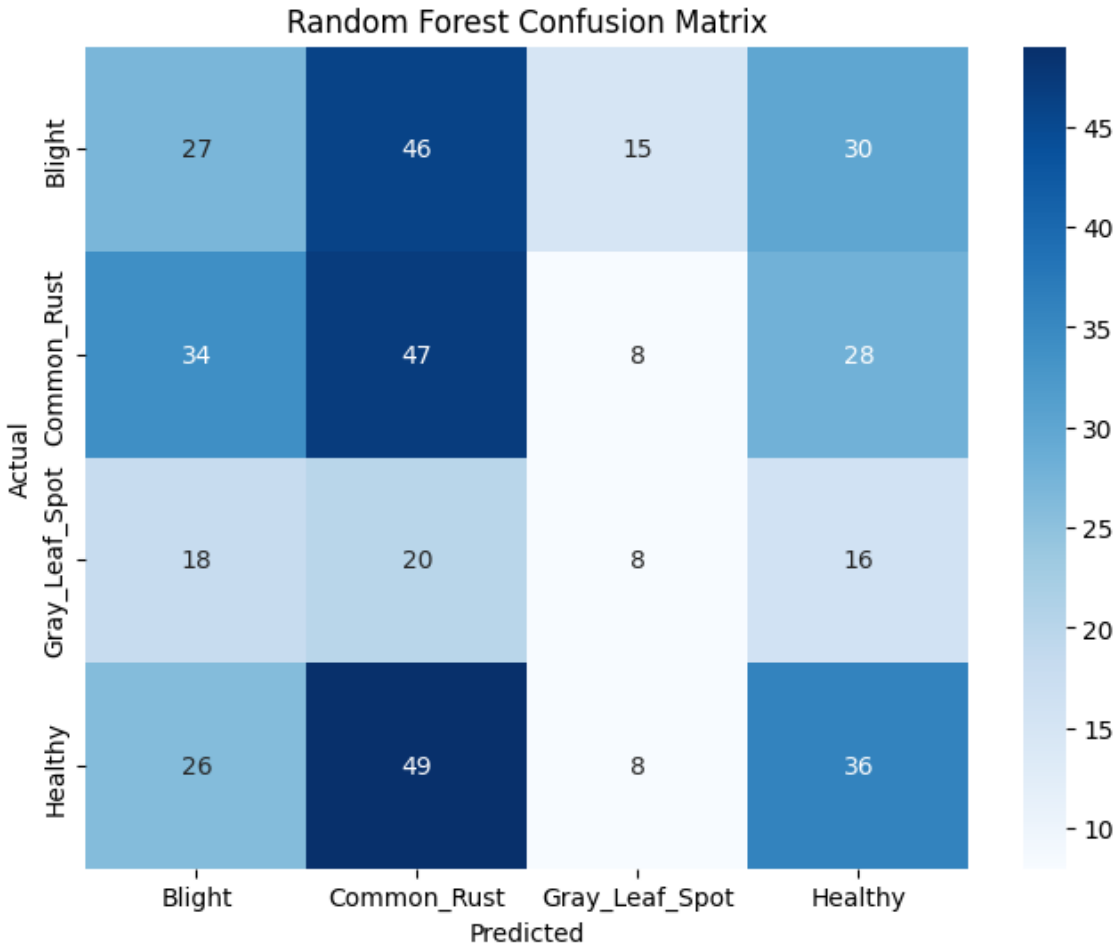

Figure 4.5 Random Forest Confusion matrix

4.3.3 Analysis Of CNN Results

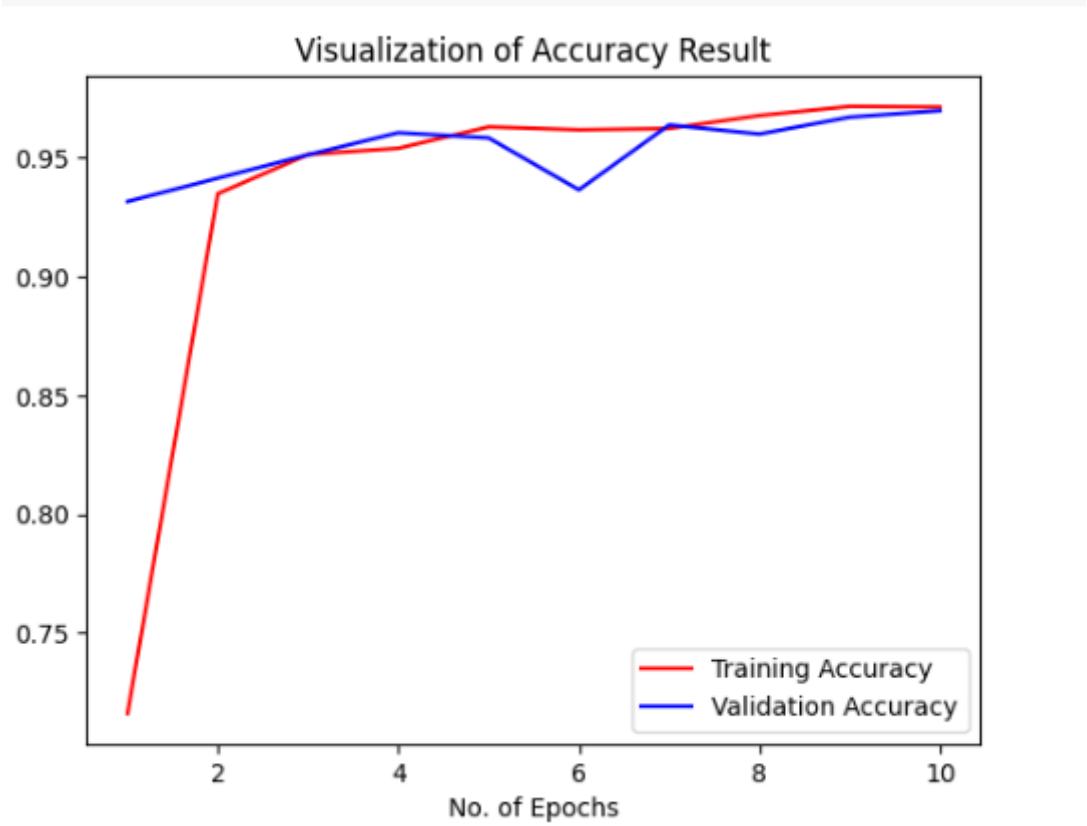

Figure 4.6 Training and Validation Graph loss of CNN

The graph in Figure 4.6 depicts the training and validation accuracies of a neural network model spanning multiple epochs. The x-axis shows the number of epochs, and the y-axis on the right represents the accuracy values. During training, the training accuracy steadily rises from 0.75 to 0.95, suggesting that the model is learning the data's properties. Similarly, the validation accuracy steadily increases, reaching around 0.90. However, after around 8 epochs, both training and validation accuracies peak. This stabilisation indicates that the model has absorbed the training data well and is generalising properly to the validation data with little overfitting. These findings emphasise the need of monitoring both training and validation measures in order to achieve a fair trade-off between model complexity and generalisation performance.

The "Actual" column depicts the instances' true classes, and the "Predicted" column indicates the classes anticipated by the model. Each forecast is accompanied by a "Confidence Score," which represents the model's level of certainty in the prediction. In this situation, the model consistently displays excellent accuracy and confidence. It accurately predicts cases of "Healthy," "Blight," and "Common Rust" with 100% confidence, demonstrating a thorough comprehension of these classes' features. Overall, these results indicate that the model is quite good at accurately identifying cases, which gives confidence in its predictions shown in the figure 4.7 below.

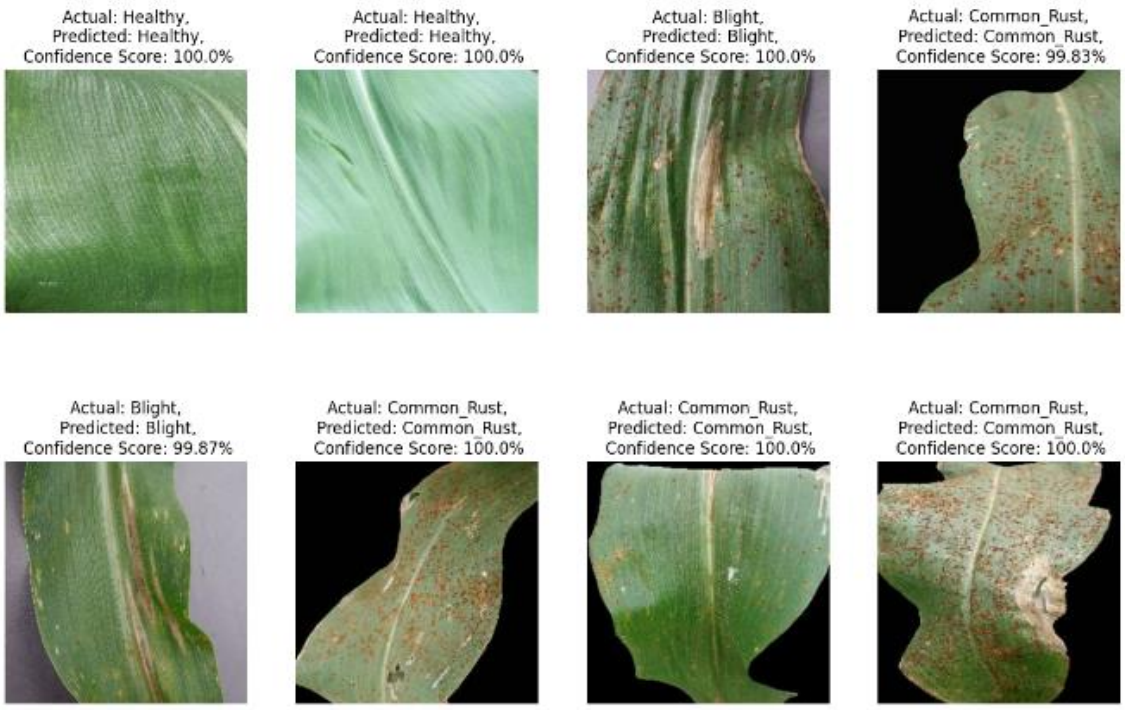

Figure 4.7 Confidence Score of CNN model

4.4 Model Deployment

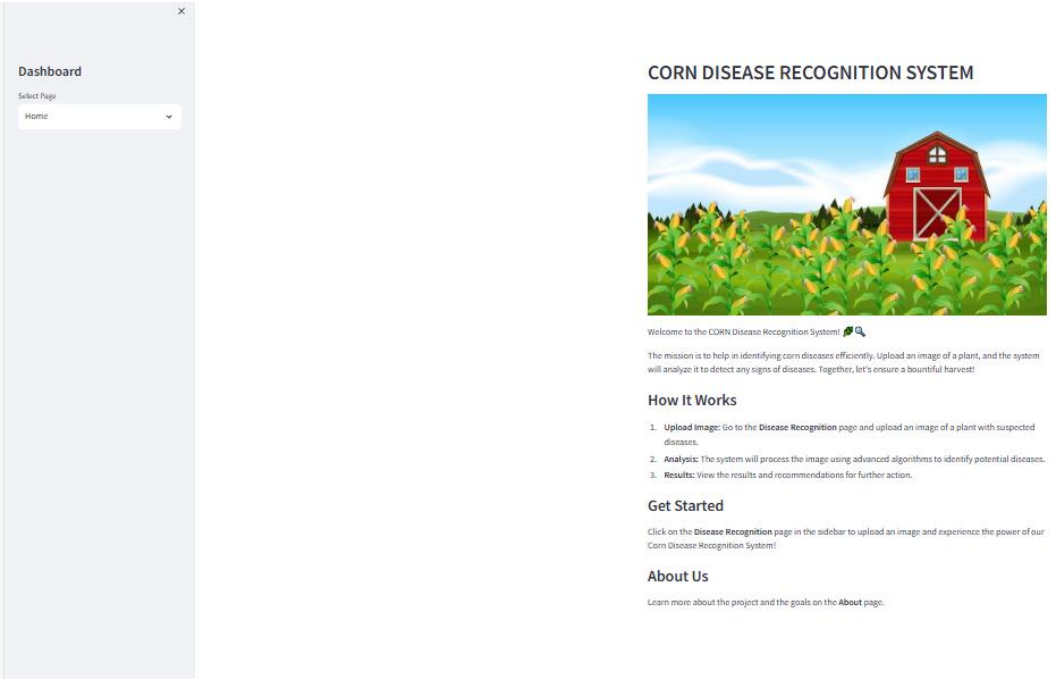

Figure 4.8 Dashboard of model launch on local host

After creating the CNN model, Streamlit and Ngrok were used to deploy it to a local web host shown in figure 4.8, resulting in the Corn Disease Recognition System, an interactive web application. This approach is intended to help users identify maize diseases efficiently. The application's site welcomes users and describes its purpose of assisting in the identification of maize illnesses through the analysis of provided plant photos. Users can go to the Disease Recognition page and upload photographs of maize plants suspected of having illnesses. The system analyses these photos using powerful algorithms and delivers a diagnosis as well as recommendations for further steps.

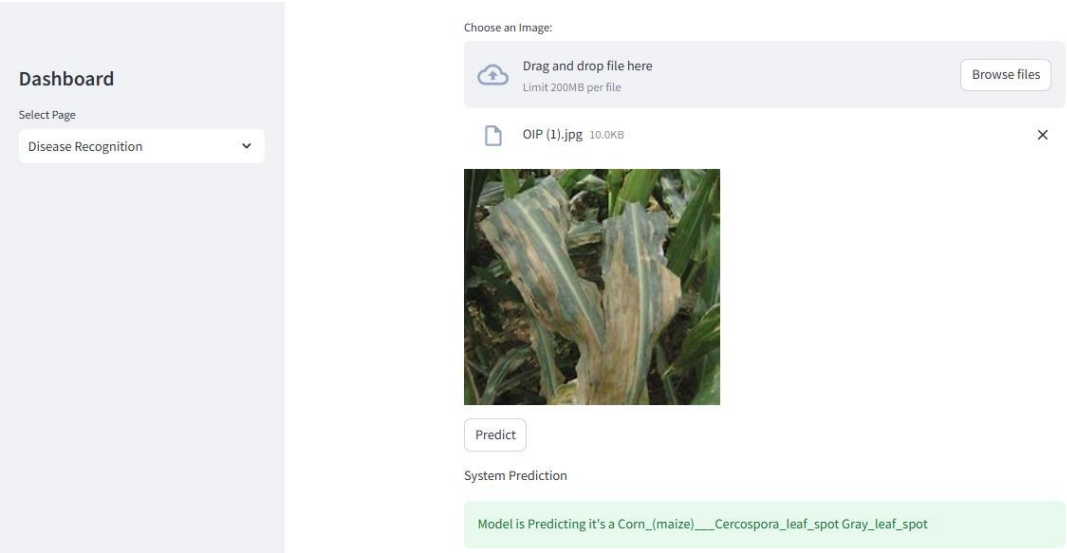

Figure 4.9 Results of model in web application

The design features a simple dashboard with options for accessing other pages, such as Home and Disease Recognition. On the Disease Recognition page, users can submit their image files by dragging and dropping them or browsing them, with a file size restriction of 200MB. Once an image has been uploaded, users can commence the analysis by clicking the Predict button. The programme then processes the image and predicts the illness, such as "Corn (maize) Gray Leaf Spot," giving customers quick and precise results shown via figure 4.9.

4.5 Summary

Based on the results, CNN outperforms Random Forest and KNN for plant disease diagnosis. With its high accuracy, robustness in categorising diverse disease classifications, and capacity to learn intricate patterns from picture data, the CNN model beats both the Random Forest and KNN method. Its continuous performance and high confidence scores in producing accurate predictions demonstrate its potential as a

dependable method for plant disease categorization. To make this capable tool more accessible, the model was deployed on a local web host with Streamlit and Ngrok. This deployment enables users to interact with the Corn Disease Recognition System via a user-friendly web application. Users can upload photographs of maize plants suspected of being diseased, and the system will analyse the images using the CNN model to deliver precise diagnoses.

CHAPTER 5 : CONCLUSION

5.1 Project Summary

The project successfully built a machine learning model for detecting plant diseases with the accuracy of 94%, accomplishing its two main goals. It requires painstaking data collecting and preparation, followed by the creation of a Convolutional Neural Network (CNN) specifically designed for picture categorization. Rigorous hyperparameter tuning and model comparisons with other approaches, such as Random Forest and K-Nearest Neighbours (KNN), revealed the CNN's higher accuracy and dependability. The final model was implemented as a user-friendly web application that enables rapid disease identification from uploaded photos. This comprehensive strategy efficiently used supervised machine learning techniques to accurately diagnose plant diseases.

5.2 Recommendations

Based on the project's results, many possibilities for future improvements should be considered. For starters, increasing the dataset to include a broader range of plant species and diseases can improve the model's accuracy and generalizability. Furthermore, advanced machine learning techniques such as transfer learning and ensemble methods for example EfficientNetV2 may improve performance. Creating a smartphone application for real-time illness detection in the field will significantly improve farmers'

convenience and accessibility. Finally, adopting a system for continual model update with fresh data will keep the model current with changing illness patterns, ensuring its efficacy throughout time.

REFERENCES

Adhiparasakthi Engineering College, Institute of Electrical and Electronics Engineers. Madras Section, & Institute of Electrical and Electronics Engineers. (n.d.). *Proceedings of the 2019 IEEE International Conference on Communication and Signal Processing (ICCSP): 4th - 6th April 2018, Melmaruvathur, India.*

Ahmed, I., Habib, G., & Yadav, P. K. (2023). An Approach to Identify and Classify Agricultural Crop Diseases Using Machine Learning and Deep Learning Techniques. *2023 International Conference on Emerging Smart Computing and Informatics, ESCI 2023*. https://doi.org/10.1109/ESCI56872.2023.10099552

Goluguri, N. R., Suganya Devi, K., & Prathima, C. H. (2022). Infectious diseases of Rice plants classified using a deep learning-powered Least Squares Support Vector Machine Model. *Indian Journal of Computer Science and Engineering*, *13*(5), 1640–1659. https://doi.org/10.21817/indjcse/2022/v13i5/221305186

Islam, A., Islam, R., Haque, S. M. R., Islam, S. M. M., & Khan, M. A. I. (2021). Rice Leaf Disease Recognition using Local Threshold Based Segmentation and Deep CNN. *International Journal of Intelligent Systems and Applications*, *13*(5), 35–45. https://doi.org/10.5815/ijisa.2021.05.04

Rathore, Y. K., Janghel, R. R., Swarup, C., Pandey, S. K., Kumar, A., Singh, K. U., & Singh, T. (2023). Detection of rice plant disease from RGB and grayscale images using an LW17 deep learning model. *Electronic Research Archive*, *31*(5), 2813–2833. https://doi.org/10.3934/ERA.2023142

Shah, N. B., Gupta, A., & Kumar, A. (2023). Performance Analysis of Real-Time Object Detection algorithm for a Multi-Class Plant Disease Detection and Classification Using Deep Learning. *International Conference on Emerging Trends in Engineering and Technology, ICETET, 2023-April*. https://doi.org/10.1109/ICETET-SIP58143.2023.10151484

Suguna, R., Vinoth Kumar, C. N. S., Deepa, S., & Arunkumar, M. S. (2023). Apple and Tomato Leaves Disease Detection using Emperor Penguins Optimizer based CNN. *2023 9th International Conference on Advanced Computing and Communication Systems, ICACCS 2023*, 1803–1808. https://doi.org/10.1109/ICACCS57279.2023.10112941

Tiwari, S., Kumar, S., Tyagi, S., & Poonia, M. (2022). Crop Recommendation using Machine Learning and Plant Disease Identification using CNN and Transfer-Learning Approach. *2022 IEEE Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation, IATMSI 2022*. https://doi.org/10.1109/IATMSI56455.2022.10119276

APPENDIX A
FYP1 GANTT CHART

Activities	Week 1 &2	Week 3&4	Week 5&6	Week7 (mid Sem break)	Week 8&9	Week 10&11	Week 12&13	Week 14 &15	Week 16 &17
Select FYP Tittle and wait for selection									
Write Chapter 1 of FYP1									
Create Gantt Chart									
Progress report for Chapter 1									
Find relevant literature review based on FYP topic									
Write literature review									
Select research methodology									
Write the Methodology									
Submission of Progress report 2 (Chapter 1,2,3)									
Finalize FYP 1 report & prep slides									
Full FYP1 submission									
Viva Presentation									
Final FYP 1 submission									

APPENDIX B
FYP2 GANTT CHART

Activities	Week 1 &2	Week 3&4	Week 5&6	Week7 (mid Sem break)	Week 8&9	Week 10&11	Week 12&13	Week 14 &15	Week 16 &17
FYP2 modify report based on feedback from FYP1									
Write draft of Chapter 4 FYP 2									
Create Gantt Chart for FYP									
Progress report 1									
Get results of FYP project									
Write Chapter 4									
Make sure all FYP objectives are accomplished									
Write the Conclusion									
Submission of Progress report 2 (Chapter 1,2,3,4,5)									
Finalize FYP 2 report & prep slides									
Full FYP 2 submission									
Viva Presentation									
Final FYP 2 submission									

APPENDIX C
CODING

Trained_plant_disease.ipynb file

```
from google.colab import drive
drive.mount('/content/drive')
# Install necessary libraries
!pip install seaborn
import tensorflow as tf
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import json
from sklearn.metrics import confusion_matrix, classification_report
# Define the path to the dataset
train_dir = '/content/drive/MyDrive/Plant_disease_detection/train'
valid_dir = '/content/drive/MyDrive/Plant_disease_detection/valid'
```

```
# Load training and validation datasets
training_set = tf.keras.utils.image_dataset_from_directory(
    train_dir,
    labels="inferred",
    label_mode="categorical",
    class_names=None,
    color_mode="rgb",
    batch_size=32,
    image_size=(128, 128),
    shuffle=True,
    seed=None,
    validation_split=None,
    subset=None,
    interpolation="bilinear",
    follow_links=False,
    crop_to_aspect_ratio=False
)
```

```
validation_set = tf.keras.utils.image_dataset_from_directory(
    valid_dir,
    labels="inferred",
    label_mode="categorical",
    class_names=None,
    color_mode="rgb",
    batch_size=32,
    image_size=(128, 128),
    shuffle=True,
    seed=None,
    validation_split=None,
    subset=None,
    interpolation="bilinear",
    follow_links=False,
    crop_to_aspect_ratio=False
)
```

```
# Normalize pixel values to be between 0 and 1
normalization_layer = tf.keras.layers.Rescaling(1./255)
normalized_training_set = training_set.map(lambda x, y:
(normalization_layer(x), y))
normalized_validation_set = validation_set.map(lambda x, y:
(normalization_layer(x), y))
# Print the number of classes
num_classes = len(training_set.class_names)
print("Number of classes:", num_classes)
# Define the CNN model with the correct number of output units
cnn = tf.keras.models.Sequential()
```

```
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3,
padding='same', activation='relu', input_shape=[128, 128, 3]))
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3,
activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

cnn.add(tf.keras.layers.Conv2D(filters=64, kernel_size=3,
padding='same', activation='relu'))
cnn.add(tf.keras.layers.Conv2D(filters=64, kernel_size=3,
activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

cnn.add(tf.keras.layers.Conv2D(filters=128, kernel_size=3,
padding='same', activation='relu'))
cnn.add(tf.keras.layers.Conv2D(filters=128, kernel_size=3,
activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

cnn.add(tf.keras.layers.Conv2D(filters=256, kernel_size=3,
padding='same', activation='relu'))
cnn.add(tf.keras.layers.Conv2D(filters=256, kernel_size=3,
activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

cnn.add(tf.keras.layers.Conv2D(filters=512, kernel_size=3,
padding='same', activation='relu'))
cnn.add(tf.keras.layers.Conv2D(filters=512, kernel_size=3,
activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

cnn.add(tf.keras.layers.Dropout(0.25))

cnn.add(tf.keras.layers.Flatten())

cnn.add(tf.keras.layers.Dense(units=1500, activation='relu'))

cnn.add(tf.keras.layers.Dropout(0.4))  # To avoid overfitting

# Output Layer with the correct number of units
cnn.add(tf.keras.layers.Dense(units=num_classes, activation='softmax'))

cnn.compile(optimizer=tf.keras.optimizers.legacy.Adam(learning_rate=0.0
001), loss='categorical_crossentropy', metrics=['accuracy'])
```

```
cnn.summary()
# Train the model
training_history = cnn.fit(x=normalized_training_set,
validation_data=normalized_validation_set, epochs=10)
# Training set accuracy
train_loss, train_acc = cnn.evaluate(normalized_training_set)
print('Training accuracy:', train_acc)
# Validation set accuracy
val_loss, val_acc = cnn.evaluate(normalized_validation_set)
print('Validation accuracy:', val_acc)
# Save the model
cnn.save('/content/drive/MyDrive/saved
models/trained_plant_disease_model.keras')
# Record training history in json
with open('/content/drive/MyDrive/saved models/training_hist.json',
'w') as f:
    json.dump(training_history.history, f)
print(training_history.history.keys())
```

```
# Plot training and validation accuracy
epochs = [i for i in range(1, 11)]
plt.plot(epochs, training_history.history['accuracy'], color='red',
label='Training Accuracy')
plt.plot(epochs, training_history.history['val_accuracy'],
color='blue', label='Validation Accuracy')
plt.xlabel('No. of Epochs')
plt.title('Visualization of Accuracy Result')
plt.legend()
plt.show()
# Get class names
class_names = validation_set.class_names
# Load the test set
test_set = tf.keras.utils.image_dataset_from_directory(
    valid_dir,
    labels="inferred",
    label_mode="categorical",
    class_names=None,
    color_mode="rgb",
    batch_size=1,
    image_size=(128, 128),
    shuffle=False,
    seed=None,
    validation_split=None,
    subset=None,
    interpolation="bilinear",
    follow_links=False,
    crop_to_aspect_ratio=False
)
```

```
# Normalize the test set
normalized_test_set = test_set.map(lambda x, y:
(normalization_layer(x), y))
# Make predictions
y_pred = cnn.predict(normalized_test_set)
predicted_categories = tf.argmax(y_pred, axis=1)
true_categories = tf.concat([y for x, y in normalized_test_set],
axis=0)
Y_true = tf.argmax(true_categories, axis=1)
# Precision, Recall, F-score
print(classification_report(Y_true, predicted_categories,
target_names=class_names))

# Confusion matrix
cm = confusion_matrix(Y_true, predicted_categories)

plt.figure(figsize=(40, 40))
sns.heatmap(cm, annot=True, annot_kws={"size": 10})

plt.xlabel('Predicted Class', fontsize=20)
plt.ylabel('Actual Class', fontsize=20)
plt.title('Plant Disease Prediction Confusion Matrix', fontsize=25)
plt.show()
```

Test_plant_disease.ipynb

```
import numpy as np
import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
import cv2
from google.colab import drive
drive.mount('/content/drive')
# Path to your model and data in Google Drive
model_path = '/content/drive/MyDrive/saved
models/trained_plant_disease_model.keras'
image_path =
'/content/drive/MyDrive/Plant_disease_detection/test/CornCommonRust1.JP
G'
# Load the validation dataset
validation_set = tf.keras.utils.image_dataset_from_directory(
    '/content/drive/MyDrive/Plant_disease_detection/valid',
    labels="inferred",
    label_mode="categorical",
    class_names=None,
    color_mode="rgb",
    batch_size=32,
    image_size=(128, 128),
    shuffle=True,
    seed=None,
    validation_split=None,
    subset=None,
    interpolation="bilinear",
    follow_links=False,
    crop_to_aspect_ratio=False
)

class_name = validation_set.class_names
print(class_name)
```

```
# Load the model
cnn = tf.keras.models.load_model(model_path)
# Test Image Visualization
# Reading an image in default mode
img = cv2.imread(image_path)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) # Converting BGR to RGB
# Displaying the image
plt.imshow(img)
plt.title('Test Image')
plt.xticks([])
plt.yticks([])
plt.show()
# Preparing the image for prediction
image = tf.keras.preprocessing.image.load_img(image_path,
target_size=(128,128))
input_arr = tf.keras.preprocessing.image.img_to_array(image)
input_arr = np.array([input_arr])  # Convert single image to a batch.
predictions = cnn.predict(input_arr)
print(predictions)
# Find the index of the highest confidence
result_index = np.argmax(predictions)
# Displaying the disease prediction
model_prediction = class_name[result_index]
plt.imshow(img)
plt.title(f"Disease Name: {model_prediction}")
plt.xticks([])
plt.yticks([])
plt.show()
```

Deploy steamlit app on Google Colab.ipynb

```
from google.colab import drive
drive.mount('/content/drive')
# Step 1: Install the necessary packages
!pip install pyngrok

!pip install -r
"/content/drive/MyDrive/Plant_disease_detection/requirements.txt"
import os
from threading import Thread
from pyngrok import ngrok
# Add your ngrok token here
ngrok.set_auth_token('2hlLLfmUqXvbe7YXIvYaOrNX5Gb_4cSbHMvYBrNPsqRuY2xKK
')
def run_streamlit():
    # Change the port if 8501 is already in use or if you prefer
another port
    os.system('streamlit run
/content/drive/MyDrive/Plant_disease_detection/app.py --server.port
8501')
# Start a thread to run the Streamlit app
thread = Thread(target=run_streamlit)
thread.start()
# Open a tunnel to the streamlit port 8501
public_url = ngrok.connect(addr='8501', proto='http', bind_tls=True)
print('Your Streamlit app is live at:', public_url)
```