

Plant Disease Detection using Deep Learning

Cheng Jung Yin

UNIVERSITI TUNKU ABDUL RAHMAN

Plant Disease Detection using Deep Learning

Cheng Jung Yin


**A project report submitted in partial fulfilment of the
requirements for the award of Bachelor of Mechatronic
Engineering with Honours**

**Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman**

May 2024

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature :  _____

Name : Cheng Jung Yin _____

ID No. : 19UEB02674 _____

Date : 15 May 2024 _____

APPROVAL FOR SUBMISSION

I certify that this project report entitled “**PLANT DISEASE DETECTION USING DEEP LEARNING**” was prepared by **CHENG JUNG YIN** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Mechatronic Engineering with Honours at Universiti Tunku Abdul Rahman.

Approved by,

Signature

:



Supervisor

:

Tham Mau Luen

Date

:

15 May 2024

Signature

:



Co-Supervisor

:

Kwan Ban Hoe

Date

:

15 May 2024

The copyright of this report belongs to the author under the terms of the copyright Act 1987 as qualified by Intellectual Property Policy of Universiti Tunku Abdul Rahman. Due acknowledgement shall always be made of the use of any material contained in, or derived from, this report.

© 2024, Cheng Jung Yin. All right reserved.

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Ir.Ts. Dr. Tham Mau Luen and Dr. Kwan Ban Hoe for their invaluable advice, guidance, technical support and their enormous patience throughout the development of the research.

In addition, I would also like to express my gratitude to my loving parents and course mate who had helped and given me encouragement, motivation and support while I have been conducting this project.

ABSTRACT

The success of deep learning (DL) has greatly promoted the use of computer vision technology in smart agriculture. Many developments in this area are focusing on timely and accurate recognition of plant disease, with the goals of increasing crop productivity and fostering economic growth. This project aims to explore the potential of two popular DL architectures namely You Only Look Once (YOLO) and transformer for recognizing 70 distinct classes of plant leaf health conditions. Specifically, a total of six models namely Vision Transformer-B/16 (ViT-B/16), ViT-B/32, ViT-L/16, YOLOv8n-cls, YOLOv8s-cls and YOLOv8m-cls are implemented and compared. In the training stage where graphics processing unit (GPU) is utilized, ViT-B/32 yields the shortest training time, which is at least 80% faster than all YOLOv8-cls variants. However, when deploying these trained models on a central processing unit (CPU), YOLOv8 models consistently outperform ViT algorithms in terms of speed and accuracy. Experiment results indicate that YOLOv8n-cls attains the highest frames per second (FPS) of 31, whereas YOLOv8m-cls achieves a test accuracy of 97.7 %. Such findings suggest that YOLOv8 appears to be more promising for real-time object classification tasks..

TABLE OF CONTENTS

DECLARATION		i
APPROVAL FOR SUBMISSION		ii
ACKNOWLEDGEMENTS		iv
ABSTRACT		v
TABLE OF CONTENTS		vi
LIST OF TABLES		ix
LIST OF FIGURES		x
LIST OF SYMBOLS / ABBREVIATIONS		xi
CHAPTER		
1	INTRODUCTION	1
1.1	General Introduction	1
1.2	Importance of the Study	4
1.3	Problem Statement	5
1.4	Aim and Objectives	5
1.5	Scope and Limitation of the Study	6
1.6	Contribution of the Study	8
1.7	Outline of the Report	9
2	LITERATURE REVIEW	10
2.1	Introduction	10
2.2	Computer Vision and Machine Learning	10
2.3	Steps in Machine Learning Workflow	13
2.3.1	Data Collection (Image Acquisition)	13
2.3.2	Pre-processing of Dataset	15
2.3.3	Data Splitting	17
2.3.4	Model Selection	18
2.3.5	Training of Model	20
2.3.6	Model Evaluation and Finalization	21

2.4	Application of Shallow Learning on Plant Disease Detection	25
2.4.1	Support Vector Machine Classifier	25
2.4.2	Artificial Neural Network	29
2.5	Application of Deep Learning in Plant Disease Detection	32
2.5.1	Introduction to Convolutional Neural Network	32
2.5.2	CNN in Plant Disease Detection	35
2.5.3	YOLO in Plant Disease Detection	40
2.5.4	Research History on Application of YOLO	41
2.6	Introduction to Vision Transformer	45
2.6.1	ViT Architecture: Patch Embedding	46
2.6.2	ViT Architecture: Linear Embedding and Position Embedding	47
2.6.3	ViT Architecture: Transformer Encoder Layer	47
2.6.4	Transformer-Based Model in Plant Disease Detection	48
2.7	Summary	52
3	METHODOLOGY AND WORK PLAN	56
3.1	Introduction	56
3.2	Dataset	56
3.2.1	Data Augmentation	57
3.3	Application of Vision Transformer in Image Classification	58
3.3.1	Training Hyperparameter on Vision Transformer	58
3.3.2	ViT models Variants	59
3.4	Application of YOLO in Image Classification	63
3.4.1	Training hyperparameter of YOLOv8	63
3.4.2	YOLOv8 Model Variants	65
3.5	Programming Flow Chart for Real-Time Webcam Inference	66

3.6	Software Overview	67
3.7	Hardware Overview	67
3.8	Gantt Chart	68
3.9	Summary	70
4	RESULTS AND DISCUSSION	71
4.1	Introduction	71
4.2	Result from Vision Transformer for Image Classification	71
4.2.1	Results of ViT on Google Colab	72
4.3	Result of YOLOv8 in Image Classification	74
4.3.1	Result of YOLOv8 on Google Colab	74
4.4	Result for ViT and YOLOv8 on Local Hardware Device	77
4.4.1	Training and Testing Performance of ViT and YOLOv8	77
4.4.2	Real-Time Inference of ViT and YOLOv8	79
4.5	Limitations and Troubleshooting	81
4.6	Summary	82
5	CONCLUSIONS AND RECOMMENDATIONS	84
5.1	Conclusions	84
5.2	Recommendations for future work	85
	REFERENCES	86

LIST OF TABLES

Table 2.1.	Confusion Matrix for SIFT using SVM. (source: Mohan, 2016)	27
Table 2.2.	Experimental result of healthy and unhealthy dataset with MLP and RBF (source: Syafiqah Ishak et al. 2015)	31
Table 2.3.	ANN classification result (Kumari et al., 2019)	32
Table 2.4.	Description of training hyperparameter. (Source: Guo et al., 2022)	37
Table 2.5.	Test results of YOLOv5 and YOLOv5-CAcT (source: Dai & Fan, 2022)	44
Table 2.6.	Comparison of evaluation indicators between YOLOv5 and YOLOv7 (source: Soeb et al., 2023)	45
Table 2.7.	Performance of CST (Guo et al., 2022)	52
Table 2.8.	Performance Summary of Shallow Learning in Plant Disease Recognition	53
Table 2.9.	Performance Summary of CNN and YOLO in Plant Disease Recognition	54
Table 2.10.	Performance Summary of Transformer-Based Model in Plant Disease Recognition	54
Table 3.1.	Information and statistical data of training dataset images	57
Table 3.2.	Training hyperparameters of ViT on Google Colab	58
Table 3.3.	Training Hyperparameter of YOLOv8 in Google Colab	64
Table 4.1.	Model Performance on Testing Dataset	78
Table 4.2.	Performance of YOLOv8 and ViT on Training and Testing	78
Table 4.3.	Comparison of Model Size and Inference Speed between YOLOv8 and ViT	80

LIST OF FIGURES

Figure 1.1.	Accuracy of Developed Model in ImageNet (source: Paper With Code, 2023)	3
Figure 1.2	Performance of Transformer Model (source: Paper With code, 2023)	3
Figure 2.1.	Output of Pre-processed Image (source: Koerich, 2018)	16
Figure 2.2.	Performance of Evaluated Feature (source: Kusumo et al., 2018)	29
Figure 2.3.	RGB Image Processing in CNN (source: Saha, 2018)	33
Figure 2.4.	Convolutional Layers in CNN (source: Saha, 2018)	34
Figure 2.5.	Model Overview of Vision Transformer (source: Dosovitskiy et al., 2021)	46
Figure 3.1.	Model Architecture of ViT-B/16	61
Figure 3.2.	Model Architecture of ViT-B/32	62
Figure 3.3.	Model Architecture of ViT-L/32	63
Figure 4.1.	Performance of ViT-B/16	73
Figure 4.2.	Performance of ViT-B/32	73
Figure 4.3.	Performance of ViT-L/32	73
Figure 4.4.	YOLOv8n Training Performance	75
Figure 4.5.	YOLOv8s Training Performance	76
Figure 4.6.	YOLOv8m Training Performance	76
Figure 4.7.	YOLOv8m-cls Inference	80
Figure 4.8.	YOLOv8m-cls Inference	80
Figure 4.9.	ViT-L/16 inference	81
Figure 4.10.	ViT-L/16 inference	81

LIST OF SYMBOLS / ABBREVIATIONS

AI	Artificial Intelligence
SVM	Support Vector Machine
ANN	Artificial Neural Network
NB	Naïve Bayes
DL	Deep Learning
CNN	Convolutional Neural Network
NLP	Natural Language Processing
ViT	Vision Transformer
CV	Computer Vision
DT	Decision Tree
ML	Machine Learning
TPR	True Positive Rate
API	Application Programming Interfaces
SIFT	Scaled Invariant Feature Transform
RBF	Radial Basic Function
SURF	Speeded Up Robust Features
ORB	Oriented FAST and rotated BRIEF
HOG	Histogram of Oriented Gradients
MLP	Multi-Layer Perceptron
VGG16	Visual Geometry Group 16
YOLO	You Only Look Once
mAP	Mean Average Precision
IoU	Intersection over Union
AcTNN	Activation Compression
PANet	Path Aggregation Network
RA	Refinement Anchor
FL	Focal Loss
Smooth BCE	Smooth Binary Cross Entropy
BCELoss	Binary Cross-Entropy Loss
SGD	Stochastic Gradient Descent
MHA	Multi-Head Attention

CST	Convolutional Swim Transformer
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
W^Q	Queries
W^K	Keys
W^V	Values

CHAPTER 1

INTRODUCTION

1.1 General Introduction

Over the course of the previous decades, agriculture becomes one of the main economic activities for several countries for example China which produces 500 million tons of vegetable every year and this amount equivalent to half of the world's crops production (Edeh,S.C, 2023). Based on the statistic which is shared by Food and Agriculture Organization of the United Nations (2020), world production of primary crops was 9.2 billion tons, and this led to the increase of the sum of agriculture value. According to the statistic, the agriculture value increase by 68% from year 2000 to year 2018. Therefore, it is important to have the method to eliminate the crops and plants from the pests and disease so that can increase the food production and can also deal with the food demand issue.

To solve the problem, many vision and Artificial Intelligence (AI) based machine learning algorithm for classification and identification of real-time image of the plants and crops have been invented. The development of the deep learning system for plant illnesses has facilitated the timely identification and management of diseases, hence mitigating crop losses resulting from such ailments. Convolutional neural network (CNN) now is being utilised under novel applications within the sector of Machine Learning (ML) to detect and diagnose plant illnesses. This development is a direct result of advancements in Deep Learning (DL) technologies. The progress in DL technology facilitated the creation of a robust CNN model for purpose of detecting and classifying plant diseases. For past decade, the convolution neural network occupied a dominant position in machine learning and perform well in image classification.

Several research regarding to the machine learning has been carried out over years. Within their past research, researchers have been developing several popular methods for machine learning. Examples of machine learning algorithms often used in various fields include the SVM, ANN, and NB. and k-means clustering. DL algorithms have come into greater prominence recently due to the widespread availability of vast quantities of data and the capabilities

of advanced computational systems, also effective training methods. CNN architectures' robust feature learning capabilities have provided noticeable outcomes for identifying plant diseases. Then, customized CNN architectures have been proposed in addition to common designs like AlexNet, GoogleNet, VGG16, and ResNet with transfer learning techniques which can be applied in the plant disease detection (Poornima et al., 2022).

Although the CNN architecture is leading the machine learning in image classification, but in the recent year, Transformer become the new focus for the researcher to be explored in the machine learning based image classification. According to Chay Nandam (2023), the transformer at first is mainly focus on the natural language processing (NLP) task. After that, it come to the initial publication that introduces a Transformer encoder trained on the ImageNet dataset which is the Vision Transformer ViT by Dosovitskiy et al. (2020). The success of Transformers in computer vision is because it allows the management of vast data amount and perform well on tasks involving picture context interpretation. For example, due to its self-attention mechanism, transformers have been utilized to enhance object identification and image captioning by helping the model to better comprehend connections between items in an image.

The accuracy of the model generated on the ImageNet dataset is depicted in Figure 1.1. ImageNet is a huge database with more than 14 million pictures that was organised into 21,841 subcategories since 2010 (Devopedia, 2021). The developer will access the ImageNet to get the image set that can be used in their algorithm. The figure 1.1 listed out the performance of ML model in image classification under ImageNet database. The statistics show the improvement of the top-1 accuracy of the model in ImageNet. In statistic below, the model that achieves the highest top-1 Accuracy is the model named BASIC-L. The concept of top-1 accuracy is a criterion commonly employed to evaluate the efficacy of an image classification model. The model BASIC-L demonstrates a top-1 accuracy of 91.1% in the ImageNet database, thereby exhibiting superior performance compared to other models. BASIC-L is a model that has been proposed in the article “Symbolic Discovery of Optimization Algorithms” by Xiangning Chen, Chen Liang and their team. BASIC-L is the combination of

the CNN with the transformer. It proved that the transformer could help in boosting the performance of the CNN model.

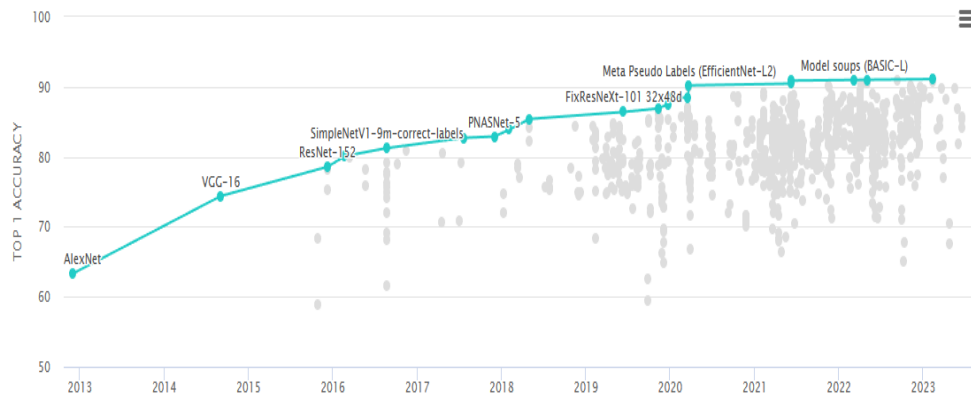


Figure 1.1. Accuracy of Developed Model in ImageNet (source: Paper With Code, 2023)

The figure 1.2 below shows the performance of the all of the transformer model in ImageNet classification and it is clear to see that the top-1 accuracy of the model was boosted to above 90% after few years of research and developing focus on the application of transformer in Computer Vision (CV) task.

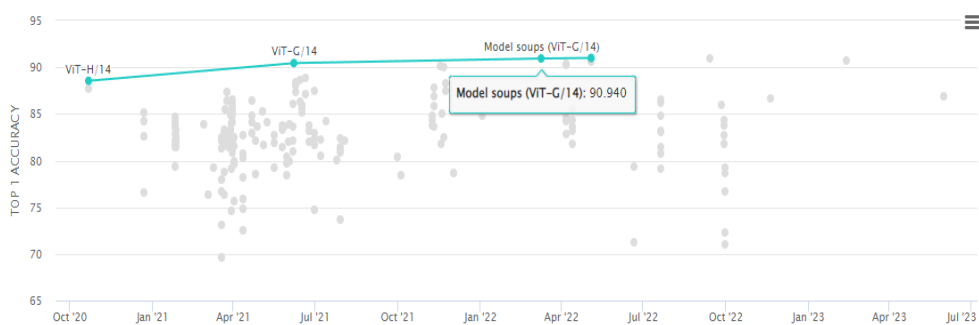


Figure 1.2 Performance of Transformer Model (source: Paper With code, 2023)

1.2 Importance of the Study

Plant diseases is always a significant threat to global food security which causing the economic and yield losses. With the rise of AI and 5G technology, the implementation of ML into the agricultural sector with lower cost is becoming possible. In the recent year, the performance of YOLOv8 and ViT is gaining the attention due to their unique ML architecture and effectiveness on the image analysis.

However, while both YOLOv8 and ViT demonstrated great success in various machine vision applications, their comparative performance on the tasks of plant disease detection from plant leaf images and the effectiveness of running real-time inference still remains huge potential to be explored. Therefore, this study is importance in measuring the accuracy and effectiveness of the proposed model. With the understanding of the model performance, we can find direction on how the development of the model can go in order to be implemented in real life applications.

Besides that, this project is important to showcase the potential of ML models on enhancing the agricultural practice. ML models are frequently considered the best option for jobs that need to be performed repeatedly and in difficult settings without sacrificing effectiveness. They are the best options for handling agricultural difficulties because of their capacity to process vast amounts of data, adjust to changing situations, and generate precise predictions in the face of adversity. With the involvement of ML model on agriculture practices, people are able to locate the best solutions on the plant diseases which can help to minimize the production loss and optimize the productivity.

This study can also show the generalization and adaptability of ML on the task of plant disease detection. This is because the self-learning and deep learning algorithm of ML model can provide the fast study speed and good memories on the studied knowledge which is comparable with human brain. This feature enables the generated model to be easily and effectively adjusted for different model parameters, learning datasets, and computational resources. This ability to scale helps the model to acquire knowledge and efficiently manage a wide range of plant diseases. The ability to share machine learning models worldwide through internet platforms is an essential aspect of creating these models, since it facilitates international collaboration and the spread of

information. The accessibility of improvements in plant disease detection enables rapid dissemination and assistance to agricultural communities worldwide. This expedites the development process and maximizes potential benefits.

1.3 Problem Statement

To develop a machine learning model for the purpose of detecting plant diseases. The evaluation of the ML model has always involved conducting a comparative analysis with another model. This analysis entails the examination of important metrics, such as precision, accuracy, F-1 score, and recall. Once the real-time plant disease detection is implemented, the model will possess the capability to differentiate between different sorts of diseases that impact plants. The primary objective of the project is to construct a real-time system for detecting plant diseases. This system will serve as a realistic demonstration of the machine learning model that has been developed. The system will be built to efficiently analyse the input video streams and instantly provide precise disease identification across different disease categories.

1.4 Aim and Objectives

This project aims to develop a ML algorithm that can be used to identify and classify the plant diseases from the leaf image. To approach this aim, two ML algorithms are proposed in this research which are Vision Transformer and YOLOv8. Another aim is to compare the performance of the proposed model on the task plant disease classification and detection and understanding the factors that will affect the performance of the model during the real-time inference with the webcam.

The objectives of this project are listed as below:

1. To develop a transformer-based and YOLO based plant disease detection.
2. To compare the performance of Vision Transformer and YOLO ML model.
3. To implement the real-time plant disease detection on webcam.

1.5 Scope and Limitation of the Study

The scope of the project is to develop YOLOv8 and ViT models on the task plant disease detection and recognition. Furthermore, I will carry out a thorough performance assessment of these models, namely by comparing their accuracy, efficiency, and resilience in the classification of plant diseases. In addition, the research aims to enhance the use of these models by putting plant disease detection on a live webcam stream. Finally, a crucial goal is to guarantee the feasibility and availability of the created ML models by producing a version that can run on a CPU, thus enabling general acceptance and use in various computing contexts.

In the research, one of the limitations is the limited resource. This project requires Graphics Processing Unit (GPU) for the model training. Model training, especially for deep learning algorithms like Vision Transformer and YOLOv8, necessitates the use of Graphics Processing Units (GPUs) and other specialized hardware due to the substantial computational power needed. However, as student, I lack access to GPU due to the high cost of the GPU hardware. This constraint has the potential to impact the speed and ability to scale up model construction and experimentation. Additionally, it may provide challenges in efficiently analyzing larger and more complex datasets. Thus, while the research aims to utilize state-of-the-art machine learning techniques for plant disease identification, the extent and scale of the study may be influenced by its reliance on GPU resources.

The second limitation in this study is faced during the training of ML models on online platform Google Collaboration. Although the platform provides the free GPU resource which is available for model training, but there are limited resources and also runtime for the free version. Google Collab only provides the connecting runtime for 12 hours free for GPU usage. Although this is enough to do training on smaller dataset but for it is not enough for the research purpose which is because with the limited training resources, it will affect the performance of the ML models on real time applications.

Moreover, in this study, the duration that is provided is only two semesters which is not more than 1 year. Therefore, one of the limitations is the dataset collection. Due to the time limitation, I can only collect the dataset online which the picture formats, environment background and leaf conditions. The

process of obtaining desired dataset is a time consuming and resource-intensive process, especially to eliminate the errors sometimes we need the expert domain knowledge during dataset arrangement and sorting.

While the ML algorithms developed demonstrate potential in controlled laboratory settings, there are numerous uncertainties regarding their performance in real-world scenarios. One major concern is the extent to which these models can apply their knowledge to new situations and datasets beyond their initial training. The training data lacked certain environmental characteristics that contribute significantly to the variability observed in real-world circumstances. These elements encompass variables such as illumination levels, variations in weather patterns, and the stages of plant development. In novel environments or when encountering previously unknown disease types, the algorithms may struggle to accurately identify and classify illnesses. Furthermore, machine learning models always face the difficulties posed by the dynamic nature of plant diseases, encompassing the emergence of novel pathogens and shifting disease patterns. Hence, the performance of the trained models will be limited on the real-time application.

1.6 Contribution of the Study

The main contribution of the study is it will emphasize the potential of state-of-the-art machine learning models on revolutionizing plant disease detection and recognition which has potential in advancing digital agriculture. The real-world uses of ML in agriculture of YOLOv8 and ViT are significantly impacted by their proven efficacy in identifying plant diseases. The research emphasizes how these models could be applied in practical settings to facilitate prompt and precise illness diagnosis, precision farming methods, and proactive disease control approaches.

Besides that, the study can also provide a deep insight into the performance and also the characteristic of YOLOv8 and ViT ML models. By comparing the models, the research can magnify their strength and limitations which can help other practitioners in informed model selection and deployment decisions.

Moreover, precision agriculture techniques, early and accurate disease diagnosis, and proactive disease management methods are made possible by the proven efficacy of YOLOv8 and ViT in plant disease detection. These findings have important applications on real-world agricultural applications. To promote reproducibility and facilitate future research in the field, the study also offers open-source implementations of the trained YOLOv8 and ViT models for plant disease identification. Through the unrestricted dissemination of these materials to the scientific community, the study fosters cooperation, exchange of information, and joint progress in the field of plant pathology and agricultural technology.

1.7 Outline of the Report

The outline of the report is structured into the following chapters.

Introduction:

- Present a succinct and precise summary of the AI model for license plate detection and recognition.
- Emphasize the significance of the study and states the problem that it addresses.
- Outline the goals and objectives of the project.
- Highlight the scope, limitations, and contribution of the study.

Literature Review:

- Analyse and categorize all the methodologies and advancements that are relevant to the project.
- Evaluate the current models.
- Identify the selected methodology to be employed in this project.

Methodology and Work Plan:

- Develop a comprehensive project timeline and identify key milestones.
- Analyse and outline the hardware and software components involved in this project.
- Describe the sequence of the design process.

Results and Discussion:

- Present the results of the research.
- Describe and explain the results.
- Show the results of the real-time inference and explain it.

Conclusion:

- Summarize all the main findings.
- Conclude the whole process in this research.
- Identify the limitations and restrictions in this research.
- Identify the future enhancements.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The field of ML has come a long way in the past few years, especially in the area of CV. The purpose of this literature review is to explain what ML and CV is and how they work together. Besides that, this chapter also listed out the review on what tasks and steps are needed to train ML models. The focus will also be on teaching both shallow learning and deep learning methods. In particular, past studies that used machine learning models to find and classify plant diseases will be looked at, with a focus on how deep learning and shallow learning methods compare. While YOLO and Transformer-based systems are unique and new in machine learning, it is important to learn not only about their structures and what they can do, but also about how to train and test models in a complex way. A model's performance depends on more than just its structure. It also depends on the quality of the training data, refining methods, and assessment metrics that are used.

This literature review also looks at a wider range of machine learning techniques than just YOLO and Transformer-based models. These include shallow learning, deep learning and Convolutional Neural Networks (CNNs). Finding out what each method does well and what it doesn't do well helps us understand their roles and how they can be used in finding and classifying plant diseases. By looking at things as a whole, we can find the best machine learning methods for dealing with the specific problems that come up when trying to develop YOLOv8 and ViT for the task plant disease detection and recognition.

2.2 Computer Vision and Machine Learning

CV is well known as research and development of the methods and techniques which enable computers to detect and interpret visual information from the real world. This entails techniques for gathering, handling, analysing, and comprehending digital pictures as well as the collection of information-producing data from the actual world. It seeks to automate tasks that human vision can achieve. Therefore, with the technology of computer vision,

computer can easily understand and recognize the videos, objects, words, and images just like our human eyes and brain. That means the computer must have abilities to understand the location of targeted object under a scenario or in an image. With the understanding, the computer must be able to categorize the objects perfectly since it has its own understanding on the object by learning and interpret the relationship and the context of the scene (Oliver et al., 2000).

According to Serokell (2000), to make the machine or a system to have abilities to recognize visual object from images or videos, it first has to be trained with grouped sample and this we name it as supervised learning. This theory is similar to our human when we need to recognize new object from real life, which human also require time to learn the recognizing factor of an object. For example, when human need to recognize a cat from real life or picture, mostly the brain will extract out the common feature of a cat such as whiskers, sharp and pointed ears and large, almond-shaped eyes. That information that our brain obtain will help the brain to give conclusion that the object is a cat. The human eye will act as sensor which similar to a computer vision system, there will be a method or device to obtain the image which is called input. Then, interpreting device has function that is similar to human brain which is understanding the image context and give the final result which categorize the image to the grouped categories.

ML is known as a branch of AI, encompasses the investigation on algorithms and statistical models. In the absence of explicit instructions, systems utilise it as a means to execute a task and instead depend on patterns and inference. Thus, it is applicable to the domains of pattern recognition, software engineering, and computer vision. Typically, computers autonomously engage in machine learning, relying on minimal intervention from software developers. This process involves leveraging data to inform decision-making and facilitating the utilisation of information in innovative ways across various industries. The methodology including three distinct types of categories which are supervised learning, semi-supervised learning, and unsupervised learning. (Mahesh, 2019)

The machine learning can help computer to do decision by applying supervised learning, unsupervised learning, or semi-supervised learning. For plant disease classification, most of the model is applying supervised learning

to solve the problem. It is known as a sub-category of machine learning. According to Mahesh (2019), the discipline of machine learning involves the use of labelled datasets for the purpose of training algorithms that are capable of effectively classifying input and anticipating outputs. The process of cross-validation involves the model adjusting its weights in response to new input data, and these adjustments endure until the model achieves a satisfactory alignment with the data.

Supervised learning encompasses two distinct issue types, namely classification and regression. Classification is a computational procedure used to accurately assign test data to specific categories. The process involves the identification of distinct entities present in the dataset and subsequently drawing inferences on the appropriate labelling or description of such entities. Linear classifiers, SVM, DT, k-nearest neighbour, and RF are among the often-employed classification approaches. Regression analysis is a statistical technique employed to ascertain the association between variables that are dependent and independent in nature. The regression techniques that are often employed include linear regression, logistic regression, and also polynomial regression. (Steven, 1998).

According to Mahesh (2019), supervised machine learning, a function is inferred using labelled training data, which is made up of a set of training samples. Normally, input dataset will be divided to training and testing subsets during supervise machine learning. The training dataset includes an output variable that serves as the target for prediction or classification. These algorithms will employ this information to create predictions or classifications on the test dataset after methodically extracting patterns from the training dataset.

In Mahesh's paper (2019), he discussed the utilisation of unlabelled data in the context of unsupervised learning. The algorithm discerns regularities within the dataset that facilitate the resolution of challenges related to clustering or association. In summary, unsupervised learning algorithms are tasked with autonomously discovering and presenting the intricate patterns and organisation inside the given dataset. Unsupervised learning algorithms will acquire various attributes from the data. Upon the introduction of new data, the system identifies the class of the data by leveraging the features that were previously learned. The

integration of supervised and unsupervised learning techniques gives rise to a semi-supervised learning algorithm. The presence of unlabelled data in data mining and machine learning applications might be advantageous when acquiring labelled data is a laborious and time-intensive task. In the context of supervised machine learning methods that are widely used, an algorithm undergoes training using a dataset that is "labelled", meaning that each record in the dataset includes the corresponding result data.

2.3 Steps in Machine Learning Workflow

The steps in Machine Learning can be divided into data collection, data preprocessing, data splitting, model selection, model training, model evaluation and model finalization. Machine learning is a set of carefully planned activities that help create and deploy effective models. Collection of relevant datasets from various sources sets the stage for analysis. Data preparation cleans, standardizes, and prepares data for model training. The dataset is divided into training, validation, and testing subsets for rigorous model evaluation. A suitable model architecture is chosen based on task difficulty and processing resources after data preparation. The chosen model is iteratively trained on the training dataset to capture patterns and correlations. Evaluation on the validation dataset assures model generality and directs refinement. Finally, the model is verified on the test dataset to assess its real-world performance impartially. This methodical methodology produces strong, accurate models for varied applications.

2.3.1 Data Collection (Image Acquisition)

The first step of developing an efficient ML model is data collection. Data collection is a process which the developer will gather specific amount of relevant data and categorize the data to create dataset for machine learning. The quality and quantity of the data (image, video, patterns, etc) collected for ML model is very important in optimize the performance of a model because collected data will affect the decision made by ML model during decision-making process (Gaudenz. B, 2022). This is because the good quality image can reduce the unnecessary information and noise which will cause the model to get false information during training process.

To enhance the performance of ML models, developer often rely on large datasets containing relevant and informative information. However, obtaining such datasets can be a complex and laborious process. In the context of image classification research, developer strive to find high-quality and relevant images online, as image quality significantly impacts the accuracy of ML model predictions (Gaudenz. B, 2022). To make the data collection process easier, developer can use approaches such as web scraping, data augmentation, and data synthesis. Web scraping is the process of obtaining relevant photos from multiple online sources, whereas data augmentation is the process of making variants of existing photographs using transformations such as rotations, flips, or zooms to enlarge the dataset.

According to Gaudenz (2022), to have great performance of computer-vision model, it must be trained with data that consist of thousand or even more images. There are a few characteristics of image need to be given attention to increase the accuracy of a computer vision system. First characteristic of the images in the dataset, must be of excellent quality. To put it another way, the image should be detailed enough for the AI model to recognise and find the target item. In most situations, AI algorithms on computer vision tasks do not yet approach human-level accuracy. As a result, if you can't recognise an item in a picture at first look, you can't expect your machine learning model to produce correct results.

Second, the picture data acquired must be diverse. The more diverse the training dataset, the more resilient the AI system and its performance in varied scenarios. The computer vision model will struggle to retain consistency in its predictions unless it has a robust collection of objects, situations, or even groups. Thirdly, quantity is a critical aspect. In general, your data collection should include many photographs - the more, the better! Training your models on a huge amount of precisely labelled data (supervised learning) can increase their odds of making accurate predictions. A decent data collection requires not only the quantity of photographs but also the density of target objects within the images.

After years of development, developers and researchers have created several large-scale online databases which consist of vast amount of structured and unstructured data. These databases are typically designed to serve specific

purpose and cater to various fields of study and industrial applications. The famous databases which have huge collection of images are ImageNet, PlantVillage, Leafsnap, Cifar-10, Cifar-100 and several others. Sharada P. Mohanty et al. (2016) used PlantVillage database to get 54, 306 of plant leaf images that was categorized to 34 classes for model training and validation in their research.

2.3.2 Pre-processing of Dataset

After collecting enough data, to build a highly effective machine learning model for image classification, it is essential to perform data pre-processing. This step significantly enhances the model's performance and ensures superior results. Data pre-processing is an important step to make sure the image data that we feed to the model has better quality which the pre-processing can eliminate useless information and noise in the images. This can help to boost the performance of the machine learning model. According to Delorme, P, J (2021), The images need to be minimal in size so that the number of features is not overfitting when fed to a Neural Network. As an example, if a coloured image is 600X800 in size, the Neural Network must manage $600*800*3 = 1,440,000$ parameters, which is huge. Any coloured image of 64X64 dimensions, on the other hand, requires just $64*64*3 = 12,288$ parameters, which is quite little and will be computationally fast.

In the research conducted by Alessandro L. Koerich (2018), the significance of image preprocessing in their topic was highlighted. The primary objective of the image preprocessing stage was to eliminate any unwanted structures from the images, such as leaf stems. To achieve this, the images were converted to greyscale, and Otsu's approach was employed to effectively separate the leaves from the background. Additionally, the top-hat approach was utilized to successfully remove the leaf petioles, further enhancing the quality of photos that has been processed. The output of the image pre-processing is showed in figure 2.1, which (a) showed the original image, (b) showed the greyscale image which set each pixel to a single intensity value then (c) showed the thresholding operation which had removed the background of the image. After thresholding, (d) is the product after top-hat operation that can help to remove the unwanted parts of the image and further refine the image. Lastly, (e)

showed the final image after bounding box which a bounding box was applied around the leaf region while removing any unnecessary or background features.

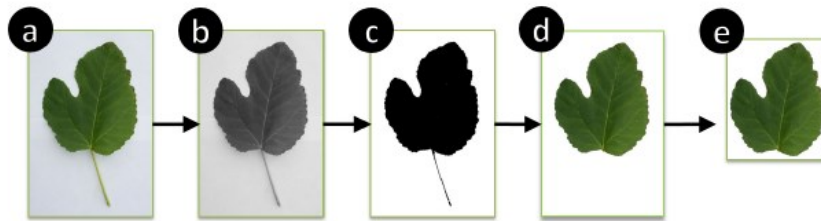


Figure 2.1. Output of Pre-processed Image (source: Koerich, 2018)

Besides that, there are many more data pre-processing methods such as randomly crop. This method can produce image that contain randomly parts of the original image. This has advantage for the model to have different features to be focused on during training process. Niventhitha et al. (2022) used image segmentation as in their image pre-processing. Image segmentation is a method which the image will be separated to form subgroup or segment and normally is being carry out through k-means clustering. With k-means clustering, it can form cluster by grouping similar between data items.

Rather than image segmentation, image scaling or image resizing is also one of the fundamental of image processing which most of the ML model will use this method to reduce the pixel of an image so that it can reduce the image computational complexity during image processing task. For most of the ML model, image scaling is their first choice during image pre-processing. This is because it can reduce the image resolution by change the value of the pixels in an image thus can reduce the information of an image and this can reduce the training period without affecting the model's accuracy.

For vision transformer, the model has different process of data processing which involve image scaling, normalization, and data augmentation. Data normalisation is an essential process that assures the data distribution for every parameter that is input. These speeds up convergence while training the network. Ali and Faraj (2014) propose that the process of normalising data involves the extraction of the mean value from each pixel, followed by the division of the obtained result by the standard deviation. The data in question

would exhibit a distribution that closely approximates a Gaussian curve with its mean centred at zero. In order to ensure that picture inputs consist of positive pixel counts, it is possible to opt for scaling the normalised data within the range of $[0,1]$ or $[0, 255]$. For transformer model, data augmentation is a way to add more information to a model by making new data points from existing data and this mostly involve in Natural Language Processing (NLP) task, which will increase the accuracy of the model by adding extra information into the existing data.

2.3.3 Data Splitting

After image data pre-processing, the next step is data splitting. Data splitting is a process that split the collected data to two or three subset which are training, testing and validation data (Gillis, A.S, 2022). To train a model, a training dataset will be generated and applied which allow the model to estimate by recognising the underlying patterns and interactions within the processed data. When generating training data from raw data, it is better to have greater representativeness of the data. This means that the extracted data should have a sufficient population for each data class. With this quality, it is also necessary to guarantee that the extracted data is impartial since biased data might generate an inaccurate model.

The validation dataset is used by the model when undergo the validating process. Validation is the method of evaluating the performance and generalisation capabilities of a trained model. It entails utilising a second dataset, known as the validation dataset, to assess the model's performance on data which cannot being observe as well as training of model is under progress. Validation dataset will be used on purpose to modify hyperparameters, evaluate model performance, also avoid overfitting. While, test data will be used after the model finishes training, validating and selection of optimum model. The forming of testing dataset needs to be careful because it may lead to overfitting and unreliable performance during testing progress. Generally, test data is the dataset that will be used to do the final evaluation on the selected model.

According to Nguyen et al. (2021), there is lots of influence of various training and testing data ratio. In his paper, he mentioned that the optimum ratio during train-test data split is 70/30. Once the ratio of the training data over 70

percent, the error will occur. This is due to the overfitting. When there is too much data is sent for ML model training, it will lead to generating noise and outlier which may change the prediction on the testing dataset. Besides that, too high ratio of training dataset means that testing dataset has lower ratio, this will cause the model to do lesser prediction on the testing data, or the variation of characteristics of the testing dataset is limited in smaller zone, thus developer cannot make sure the performance of the model when it comes to the real-world scenario although the model may perform well in the smaller testing dataset. In other words, if there is too less ratio of training dataset, this means that the model does not receive enough training and learning, and this may lead to the poor performance when it comes to make predictions, which is also known as underfitting.

2.3.4 Model Selection

In the workflow of implement ML, model selection is a step that cannot be ignored in creating robust and accurate prediction models. The process of choosing which method and model architecture is best suited for a specific job or dataset is known as model selection. It comprises contrasting multiple models, evaluating their effectiveness, and selecting the one that best resolves the current situation. This is because each models have their own complexity, fundamental presumption, and abilities. If a model which is very complicated may overfit the data thus be unable to generalize, whereas if a model that is overly basic may underfit the data and do badly in terms of prediction.

In a selection of model, there are a few factors that must be considered. Firstly, developer need to clear about the problem and issue need to be solved before proposing a suitable model for the issue. After problem identification, researcher can choose a collection of models that are relevant to the problem at hand. These models will range from simple approaches like decision trees or linear regression to more complex models. Shailendra Chauhan (2023) discussed when choosing a machine learning model, there are several various crucial factors to consider which can help to ensure that the chosen model is successful in resolving the fundamental issue and has the potential for remarkable performance (Chauhan, 2023).

The primary factor influencing model selection is the complexity of both the problem being addressed and the data being processed, necessitating an evaluation of the problem's intricacy. Simple models may be sufficient in certain cases, but when dealing with complex data linkages, more complicated models may be required. When establishing the optimal amount of model complexity, considering factors including the size of the dataset, the complexity of input characteristics, and the possibility of non-linear relationships. This assessment makes sure the model of choice appropriately captures the underlying patterns and delivers superior insights and forecasts. When choosing the model, it is also important to consider the importance of interpretability. As an example, Decision trees and linear regression models are able to give unambiguous conclusions about the correlations between input data and intended outcomes, making them simply interpretable. More complicated models, such as neural networks, may provide improved performance with the downside of poorer interpretability. Striking the right balance between interpretability and performance is essential, particularly in domains where understanding the model's decision-making process is crucial, such as healthcare, finance, or legal applications (Chauhan, 2023).

Besides that, the interpretability of a model is also one of the important factors during the model selection. Several models such as decision trees and linear regression will give detailed insights on the relationship between the input data with the results, causing the model more interpretable. Then, for the complex model for example neural network will provide better performance at the expense of diminished interpretability. This is because the complex model normally consists of many hidden layers and millions of parameters, and this will provide complex model better performance but trade-off their interpretability. Most applications of ML models will trade-off their interpretability to have better performance but in the case of legal decisions and finance, the decisions which are made by model often require transparency so that people will know the elements that influence the model's recommendation. But for NLP and image classification, performance is more prioritized than interpretability (Chauhan, 2023).

Lastly, always consider the resource constraints during model selection. This is because resource constraints include limited memory space, processor

speed or implementation time. Given the limited resources, it is difficult to make sure the chosen model can be effectively implemented and used efficiently. Some machine learning models will require larger computational resources during training or inference, causing them unsuitable to be implied under limited resources. Therefore, to achieve greatest results and successful implementation, it is crucial to achieve balancing between model performance and resource efficiency.

The selection of a suitable machine learning model is a pivotal stage in the development of a machine learning model, contingent upon the aforementioned elements. Software developers have the ability to choose an appropriate model that aligns with the specific requirements of the situation at hand. Algorithms can generally be classified into two main categories: shallow learning, which includes conventional machine learning models such as Support Vector Machine (SVM), Decision Tree (DT), and Artificial Neural Network (ANN), and deep learning. Shallow learning normally has lower cost but in certain tasks they have low performance, while the deep learning is inspired by the brain function which allow the developer to boost the performance of ML model with enough dataset for learning and exhibit powerful performance on the specific tasks and conditions for examples image classification, natural language processing and speech recognition.

2.3.5 Training of Model

Model Training is a process which we need to feed data to a parametrized machine learning algorithm to allow it to give the desired output which is making prediction and show high performance. Generally, it is the process of feeding the training data into the selected algorithm. Every machine learning model is dependent on the data it gets as input. Irrespective of the intricacy of the model, its performance is inherently linked to the calibre of the input data. The underlying idea that emphasises the significance of feature engineering in the training process is the fundamental notion that the quality of input directly influences the quality of output.

In a model training, there is one part that is very important which is parametrize the machine learning algorithm. Machine learning (ML) algorithms are encapsulated as coordinated sequences of code, consisting of a group of

instructions, and coordinated by a predetermined group of input parameters, commonly referred to as "hyperparameters." Developers can artistically fine-tune the learning trajectory of the algorithm using these malleable hyperparameters, carefully tailoring it to the specifics of the relevant dataset and the specific nuances of the usage context. In this orchestration, the documentation that accompanies each algorithm assumes a paramount role. Generally, in the domain of latest neural networks, the learning revolves around fine-tuning the weights associated with the activation functions within each layer, orchestrating a symphony of calculations to unveil intricate patterns in the data. This individuality in learning mechanisms underscores the diversity and adaptability inherent in machine learning, allowing each algorithm to flex its own set of trainable parameters to craft solutions tailored to specific challenges (Jing Wei et al. 2019)

2.3.6 Model Evaluation and Finalization

According to Japkowicz (2006), model evaluation is a process which allow the developer to understand the characteristic and completion of a ML model. Model evaluation process encompasses the application of various performance metrics, which can be carried out through two distinct approaches: offline and online assessments. The offline assessment is the evaluation method that is carried out based on the historical data or result from the model. In the approach, the model will be provided a fixed dataset, and the evaluation will not involve any interaction of real-time data. The most common matrices used for offline evaluation is accuracy, precision, recall, F1-score, and confusion matrix. The offline evaluation approach is primarily employed in scenarios involving classification metrics, particularly in supervised learning. In this method, the model's predictions are compared with the labelled dataset, and any discrepancies are recorded and calculated to determine the model's performance metrics.

In this study, which focusing the development of ML learning model to plant disease detection and recognition, the method of evaluation that will be focused on is the offline assessment on the supervised learning. Guo, Lan and Chen (2022) have their model evaluation by calculating the Top-1 accuracy, precision and F-1 score in their research.

The Top-1 accuracy refer to the convolutional accuracy that shows the top accuracy of the model prediction which the prediction outcome with the highest probability to be correct. The formula of the Top-1 accuracy as follow:

$$Top1 Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

Precision serves as a frequently employed metric for assessing model performance. It signifies the proportion of accurately predicted positive instances within a given sample (Guo et al., 2022). Higher precision means that the model is accurate in the positive predictions and there are fewer false positive. Precision is calculated as below:

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

Rather than precision, we also can have recall as one of the evaluation metric for a machine learning model. Recall, also known as sensitivity or True Positive Rate (TPR), quantifies the ability of the model to accurately detect true positive instances. A more advanced model demonstrates increased recall by efficiently collecting a substantial proportion of positive cases. The formula for calculating recall is as follows:

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

According to Guo et al. (2022), F1-score is also used in their research as a metric to evaluate the ML model performance. F1-score is also known as F1-measurement or F1-beta score. It is obtained by combines the different weighting of precision and recall. A higher F1-score indicates superior performance of the machine learning model. The formula for calculating F-1 score as follows.:

$$F1\ score = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (4)$$

Evaluation of ML model very important during the creation and implementation of machine learning algorithms. This is because we can make sure that the final models perform as well as they possibly can. By the comparing different model through their performance and determine which model obtaining an ideal level of performance which can give the better solution to the proposed assignment. Therefore, model evaluation provides standard matrices for developer to choose the model that is best at tackling the problem at hand thorough comparisons with other trained models. Besides that, the ML model evaluation can also determine the productionized models' dependability. This is done by seeking the performance of the model under different conditions during model evaluation and the purpose is to make sure the ML model achieve consistence performance under different condition. (Japkowicz, 2006).

To add to that, model evaluation is important because it can provide insights into the errors the model makes. Thus, model evaluation allow developer to conduct error analysis and the information that is obtained from the evaluation allows the developer to understand the limitations of their ML model. For example, model evaluation allows us to identify the model is more prone to false positive or false negative. By understanding the limitation or error in the model, developer can propose solution to improve the model in order to provide better user experience. The model improvement or troubleshoot can be done by hyperparameter tuning of the model or we named it as fine-tuning, while the model evaluation can help the developer to identify the model performance by assessing how different the hyperparameter will affect the model's effectiveness. On top of that, model evaluation allows the continuous improvement of the model as well as the model has been developed into the real-world application. This is because the requirement of the user will increase by the time, therefore model evaluation allows the model to always have best performance after every update.

After the process of model evaluation, the developer can prove to user that the model has been boosted to its best performance when the ML model is proposed to specific task or issue. Therefore, the next step after the model

evaluation is the model finalization. In the final step of a model construction, it is about model deploying and continuously monitoring on the developed model. According to Pruneski et al. (2022), the developed model can be shared via online repositories, fostering future collaboration and knowledge exchange. With the model deployment, the well-trained model can be put into practice in the real world. To put the developed model into real world, developer need to choose the deployment environment according to the needs of users. For example, developer can deploy the model through cloud services. Cloud services such as AWS, Azure, and Google Cloud, on the other hand, provide adaptable and flexible environments for the deployment of models. Besides that, developer can also build Application Programming Interfaces (API) to deploy their model. APIs allow input of real-world data, predictions, and results retrieval by allowing smooth connection between the model and other software applications or systems.

Pruneski et al. (2022) in his paper mentioned about after the deployment of the model, that is not the end of the model construction. This is because one of the common challenges that will be faced by every ML model is they will become out-date when the models were exposed to changes in real-world data. Therefore, we need continuous monitoring on the deployed model. It is a technique or process for keeping the ML learning model in the real-world settings. In the process, continuous model performance monitoring, tracking of features and hyperparameters used for retraining, and smooth management of the full model lifetime (Pruneski et al., 2022).

Model retraining is the main part of continuous monitoring of a ML model. Under real-world scenario, data that will be fed to the ML model will change over time and there will be a scheduled task after the model has been deployed (Pruneski et al., 2022). The changes of the fed data and scheduled task will deteriorate the performance of the model. Therefore, it is crucial to implement an automate system that is using an automated model pipeline during the finalization, so that ML engineer can do hyperparameter tuning to help remain the completion of developed model. In the realm of ML, it is of utmost importance to consistently assess the performance of the model and include fresh training data in order to maintain its efficacy. The task at hand necessitates the formation of a team responsible for overseeing the influx of data, inputting

it into the existing model, and occasionally developing a novel model that can consistently surpass or match the performance of previous models. This is done with the objective of enhancing the outcomes of the model or the relevant professional field.

2.4 Application of Shallow Learning on Plant Disease Detection

Ahead of the development of DL, researchers relied heavily on shallow learning approaches due to hardware restrictions that could not support computing requirements of DL models. Shallow learning approaches, such as linear regression and decision trees, performed well on basic tasks but struggled to catch complicated patterns in huge datasets. However, when advances in hardware technology, notably the introduction of GPUs, made it possible to handle deep learning's massive processing needs, academics began to investigate its possibilities. DL has subsequently transformed computer vision by allowing models to learn complicated patterns and representations from large volumes of data, resulting in important breakthroughs and advances in these areas.

2.4.1 Support Vector Machine Classifier

SVM have been widely employed in the fields of image identification and object detection due to its ability to perform pattern classification and nonlinear regression tasks. The SVM constructs a linear model that incorporates nonlinear class boundaries and support vectors in order to assess the predictive function. SVM utilizes linear models to find the best hyperplane for effectively separating data, while also maximizing the margin between the hyperplane and the nearest training data points. This method operates on the assumption that data can be perfectly separated using straight lines. Support vectors, which are the training points closest to the ideal separation hyperplane, are instrumental in defining and determining this hyperplane. SVM employs a nonlinear mapping technique to transform the input image to a higher-dimensional space of features. As a result, SVM functions as a linear classifier within the parameter space. However, it exhibits nonlinearity as a classifier by virtue of the nonlinear mapping that is applied to patterns of data within the feature space of higher dimensions (Mohan, Balasubramanian, & Palanivel, 2016).

Mohan et al. (2016) apply Haar-like features and AdaBoost classifier to classify the paddy plant image that has disease before determining the type of disease using SVM classification. But there is the disadvantage of shallow learning, which is it require more feature extraction or feature engineering and data preprocessing before being fed into the machine learning algorithm for training. This is because the shallow learning normally has limited representation, which lesser capacity to represent the complex patterns and relationship in data. Therefore, before feeding the image into the SVM for training and learning, it's essential to perform information extraction to eliminate extraneous noise that doesn't contribute to SVM classification.

In the research paper, the authors applied Scaled Invariant Feature Transform (SIFT) as the filter to capture abrupt changes in intensity within the input image, the approach involves filtering the images at various scales and patch sizes to extract relevant features. The features that have sudden changes of image intensities may represent the disease of the plant image, and with the SIFT filter, SVM only needs to learn from the extracted parts or features (Mohan, Balasubramanian, & Palanivel, 2016). SVM will optimize the separation boundary for grouping the paddy plant disease based on the difference from the input image. The critical properties from the input images will be extract by using SIFT and it will output seven-dimensional feature vector to SVM model when training SIFT. The seven-dimensional features are x and y coordinates, sub-level scale, picture feature size, edge flag, edge orientation, and response curvature throughout scale space (Mohan, Balasubramanian, & Palanivel, 2016). When the dimensional features are fed to SVM, the displacement between feature vector and the hyperplane of SVM are calculated. Then from the calculated distance, they obtained average distance, and it is used to identify the paddy plant disease by grouping the nearest value of average distance into groups.

From the research paper by Mohan (2016), the accuracy of the paddy plant disease detection achieves 91.10% by applying SIFT feature extraction with SVM model for paddy plant disease classification. The model has precision of 86.66%, recall of 86.66%, accuracy of 91.10% and F-1 score of 86.66%. Below table shows the confusion matrix of the model from Mohan:

Table 2.1. Confusion Matrix for SIFT using SVM. (source: Mohan, 2016)

Paddy Plant Diseases	TP	FN	FP	TN
Brown Spot	9	1	1	19
Leaf Blast	8	2	2	18
Bacterial Blight	9	1	1	19

The study conducted by K. Rajesh Babu in 2019 explores the application of Support Vector Machine (SVM) classification for identifying plant diseases. The research assesses the efficacy of the SVM model in classifying plant diseases by examining various selections of kernel parameters and soft margin parameters. Initially, the author compiled an input database of photos comprising several plant diseases such as *Alternaria alternative* (a fungal infection), Anthracnose, Bacterial Blight (caused by bacteria), *Cercospora Leaf Spot*, Bacterial leaf spot, frog eye leaf spot, sun burn disease, as well as photographs of healthy leaves. Babu utilised the SVM model to incorporate the database, opting not to employ feature selection. However, K-means was utilised for the purpose of picture segmentation, resulting in an accuracy rate of 90%. In the SVM model, a linear kernel was utilised, yielding an accuracy rate of 89%. The classification accuracy of the SVM model using the kernel of RBF and polynomial kernel achieved 88.8% and 90.2%, respectively.

According to Babu (2019), the effectiveness of SVM highly depends on selection of kernel and soft margin parameters, so he proposed the feature selection into SVM model with linear kernel, RBF kernel and polynomial kernel to investigate how well can feature extraction in improving SVM models. The author proposed "colour co-occurrence" technique is to extract features from photos depicting plant diseases. In particular, the conversion process involves transforming RGB images into the HIS (Hue, Saturation, Intensity) colour space representation. Within this particular colour space, a total of 14 distinct measurement feature measures are computed based on the co-occurrence matrix. After the adding of algorithm of proposed feature extraction, the performance of SVM model with linear kernel, RBF kernel and polynomial kernel classification accuracy increase to 95.63%, 94.23% and 95.87% respectively.

The increasing of the performance shows that SVM model is highly dependent on image processing especially segmentation and feature engineering.

The study conducted by Kusumo, Heryana, Mahendra, and Pardede (2018) focuses on assessing the effectiveness of shallow learning techniques in detecting corn-plant diseases. The researchers employ a dataset consisting of corn plant leaves and investigate the impact of various image processing features and feature extraction methods, including RGB, and SIFT. The research team is primarily dedicated to the advancement of plant disease identification by the use of established machine learning methodologies.

In research paper by Kusumo et al. (2018) explain how those feature extraction function. RGB is the feature extraction method that will extract the colour information and turn them into the values from 1 to 255 according to the intensity to allow pattern recognition. SIFT algorithm is designed to find and recognise distinctive key features within an image. It achieves this by identifying stable and recurring structures that exist across various scales. The process entails performing a convolution operation on the picture using Gaussian filters at various scales, resulting in the generation of a scale-space representation and also exhibit a localised orientation of the features. SUFT feature extraction is similar with SIFT which is it is also a method for detecting interest points in images by analysing distinctive points in an image by identifying areas where the intensity pattern changes significantly in both scale and orientation. HOG is the technique that will calculate and divide the image into small cells and computing the orientation of gradient and creating histograms to represent the local object and shapes in an image. Lastly, ORB extracts the features by firstly detects salient features inside an image by employing a modified version of the FAST algorithm, which is specifically designed to locate points that possess high distinctiveness.

The author prepares the dataset in 4 categories which are corn gray leaf spot with 513 images, corn common rust with 1192 images, corn nothern leaf blight with 985 images and healthy leaf with 1162 images. Kusumo et al. (2018) proposed their processing procedures undertaken, specifically the resizing of images from their initial dimensions of 256x256 to 64x64 pixels. Additionally, they highlight the extraction of multiple features, encompassing RGB, HOG (with 34,020 dimensions), SIFT (with 12,800 dimensions), SURF (with 12,800

dimensions), and ORB (with 8,000 dimensions). The analysis utilises various classifiers, including Gaussian Naive Bayes, SVM with linear and RBF kernels, and Random Forest with 1,000 trees. Kusumo et al. (2018) propose the classification algorithm with the combination of the proposed feature extraction methods and compare the accuracy and the best performance of each combination is taken. From the research, RGB feature extraction can give best assist to in improving the classification algorithm which as shown in the histogram below:

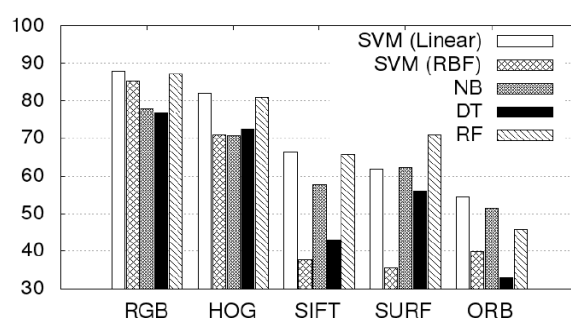


Figure 2.2. Performance of Evaluated Feature (source: Kusumo et al., 2018)

From the figure above, the performance of the classification metric reached the best performance under RGB feature extraction where RGB with SVM (linear kernel) reached accuracy of approximately 88%, RGB with SVM (RBF kernel) reached accuracy of approximately 85%, RGB with NB reached accuracy of approximately 78%, RGB with DT reached accuracy of approximately 76%, and RGB with RF reached accuracy or approximately 87%.

2.4.2 Artificial Neural Network

ANN is classified under shallow machine learning which is proven can be effectively utilised for the categorization of leaf diseases. ANNs commonly known as neural networks, are a class of machine learning models that draw inspiration from the physiological and functional characteristics of the human brain. Artificial Neural Networks (ANNs) incorporate interconnected nodes, also known as artificial neurons, that are organised in layers. A standard neural network setup usually consists of several layers, which include an initial input layer, one or more intermediary hidden layers, and a concluding output layer. In

a study conducted by Syafiqah Ishak et al. (2015), the authors presented a research paper titled "Leaf Disease Classification using Artificial Neural Network." In this publication, the authors established a feed-forward Neural Network and utilised back-propagation technique.

The Back-propagation algorithm well-known as commonly employed method for purpose of training of multi-layer perceptron (MLPs) and Radial RBF. MLP is a class of artificial neural networks that find utility in diverse fields. The algorithm proceeds through three main stages. Initially, it executes a feed-forward pass to process input patterns. Subsequently, it undertakes a back-propagation phase to trace and assess prediction errors in a reverse manner throughout the network. Finally, it iteratively adjusts weights and biases using error information to enhance the accuracy of predictions. A standard multi-layer perceptron is composed of input, output, and hidden layers, where each neuron in these levels is equipped with biases that resemble weights. The Back-propagation algorithm is based on the generalized delta rule and aims to minimize the total squared error through the use of gradient descent and can improve the network's efficiency (Syafiqah Ishak et al., 2015). Unlike MLP, RBF artificial neural networks have a simpler topology. An input layer, a hidden layer with Gaussian radial basis functions, and an output layer make up RBF networks. Function approximation and pattern recognition employ them, notably for nonlinear input-output interactions. RBF networks employ radial basis functions in the hidden layer to compare incoming data to established prototypes, capturing complicated data patterns.

The very first step is collecting the images, which 50 photos of healthy leaves and fifty images of unhealthy leaves are collected. After that, the images will undergo image processing consists of contrast enhancement, segmentation, and feature extraction. Then, the classification of healthy and unhealthy leaves will be completed with MLP and RBF, the performance will be compared. The table 2.2 below show the result of classification accuracy of MLP model with RBF model and the RBF shows better performance that the MLP. This shows that the ANN can also has ability to be train and get identify healthy and unhealthy leaves.

Table 2.2. Experimental result of healthy and unhealthy dataset with MLP and RBF (source: Syafiqah Ishak et al. 2015)

Training Samples	Testing Samples	Classification Efficiency (MLP)
90	10	99.15%
80	20	94.05%
30	70	90.3%
Training Samples	Testing Samples	Classification Efficiency (RBF)
90	10	98.85%
80	20	99.1%
30	70	99.2%

Kumari, Prasad, and Mounika (2019) identified and categorized healthy and diseased leaves using a neural network classifier in order to advance research on the detection of leaf diseases. The investigation begins with the acquisition or collection of photographs, which consists of two sets: twenty images representing diseased tomato leaves affected by Septoria leaf spot and leaf mold diseases, and twenty images depicting afflicted cotton leaves afflicted with diseases including bacterial leaf spot and target spot. The k-means clustering technique is subsequently implemented for image processing tasks including image segmentation and clustering. It may be beneficial to eliminate the discoloration portion of the diseased leaf. As illustrated in the figure below, the image is concentrated. The objective of performing image clustering is to eliminate the spotted regions of the leaf. Feature extraction will subsequently be executed on the segmented images. Kumari et al. (2019) suggests using extracted features to detect and classify leaf diseases.

The proposition put forth by Kumari et al. (2019) involves the application of extracted features in order to detect and classify leaf diseases. The classification process is executed utilizing an algorithm called ANN. The extracted features, including Contrast, Correlation, Energy, Homogeneity, Mean, Standard Deviation, and Variance, will comprise the input to the neural network. As a class vector, the target data for the neural network will be supplied. Following this, Kumari et al. (2019) implemented a back propagation neural network in order to classify the data. The operational results of the model are detailed in the table that follows.

Table 2.3. ANN classification result (Kumari et al., 2019)

Leaf disease	Bacterial Leaf Spot	Target Spot	Septoria Leaf Spot	Leaf Mold	Accuracy
Bacteria Leaf Spot	9	1	0	0	90%
Target Spot	2	8	0	0	80%
Septoria Leaf Spot	0	0	10	0	100%
Leaf Mold	0	0	0	10	100%
Average Accuracy					92.5%

2.5 Application of Deep Learning in Plant Disease Detection

Application of DL to recognize the plant diseases is a big step forward on the modern farming technology. Due to the growing problem of plant diseases and the pressing need for more ecologically friendly farming practices, there is an immediate need for rapid and precise techniques of identifying these illnesses. One potential method for satisfying this demand is DL, a subfield of ML that use numerous-layers networks to extract intricate patterns and features from data. Using massive volumes of picture data from both healthy and diseased plants, deep learning algorithms may learn to distinguish between various disease indicators and correctly categorise a plant's health status. The various applications of DL in the recognition of plant diseases are discussed in first section. It delves into the processes, challenges, and potential advantages of implementing these cutting-edge approaches in agricultural practices.

2.5.1 Introduction to Convolutional Neural Network

Jing Wei (2019) classifies Convolution Neural Network (CNN) as a DL method that integrates discrete convolution techniques for image processing with principles derived from artificial neural networks (ANNs). The approach employed in this study draws inspiration from the fundamental principles governing the functioning of simple and complex cells observed in the field of visual neuroscience. The utilisation of this technique enables the network to efficiently analyse images without requiring computationally intensive procedures such as feature extraction and data reconstruction, which are commonly performed in conventional image recognition systems.

Within CNNs, there is a considerable difference in the makeup of the neurons. These neurons have a three-dimensional structure that includes both the depth and length of the incoming data (height and width). In contrast to traditional ANNs, neurons in a particular CNN layer only link to a small portion of the layer above. For example, in a real-world situation, the input "volume" would include the following measurements: 64 64 3 (indicating height, width, and depth). This results in an output layer that is eventually 1 1 n in size, 'n' represents the possible class number.

Convolution Neural Network normally have three core layers, which are convolutional layer, pooling layer, also fully connected layer. Inside each layers, there are several hidden layers that bind together to form one type of layer. The convolutional layer is an important component of CNNs, and its key element is its learnable kernels (Shea & Nash, 2015).

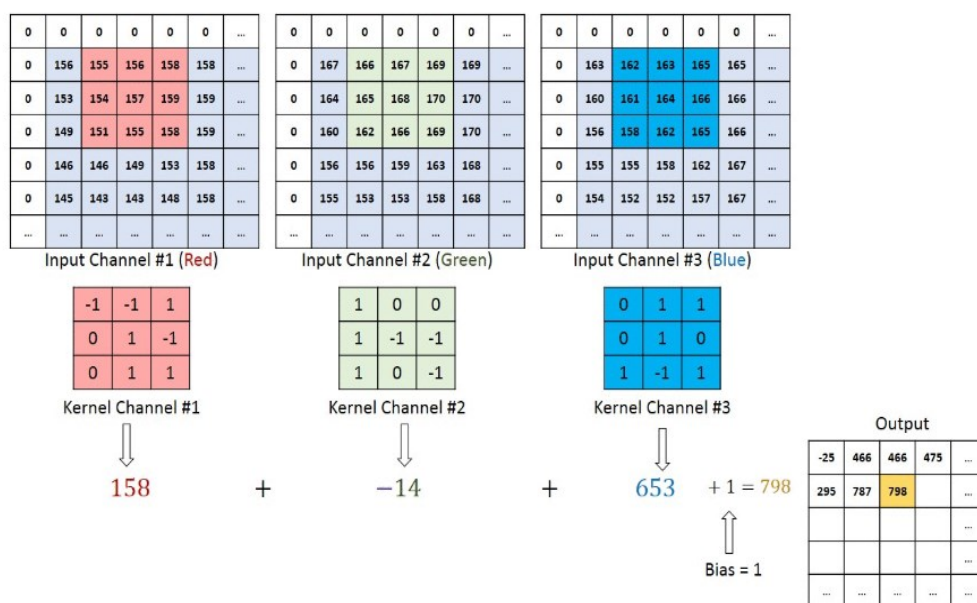


Figure 2.3. RGB Image Processing in CNN (source: Saha, 2018)

From the figure above, we can observe that normally an image will be converted to RGB image which only consist of three colour panes which are Red, Green Blue before it enter to the convolutional layer. In the figure above, an input image consists of $M \times N \times 1$ image matrix with $3 \times 3 \times 3$ of kernel or filter. The kernel will move horizontally by a until it completely fills the width. Then, using the same Stride Value, it goes down to the picture's beginning place

on the left and continues the procedure until it has scanned the entire image (Saha, 2018).

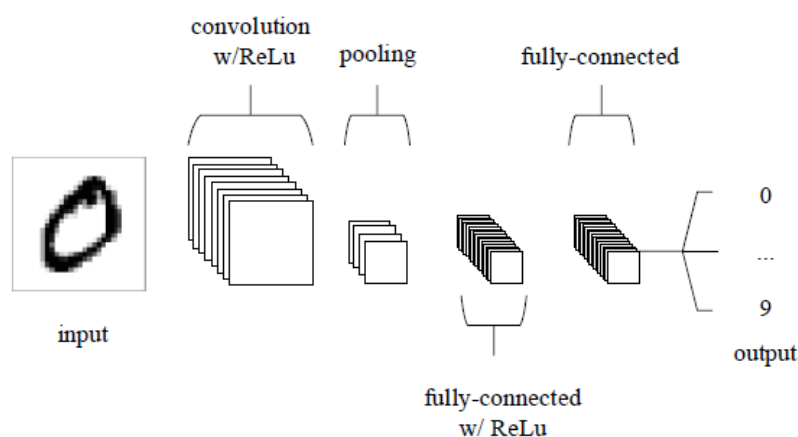


Figure 2.4. Convolutional Layers in CNN (source: Saha, 2018)

As shown in the figure, the pooling layer follows the convolutional layer. Pooling layers are implemented with the aim of reducing the dimension of images, thereby minimising the number of variables and computational complexity. In addition to employing activation maps, these layers frequently utilise max pooling with 2×2 kernels and a stride of 2. A 25% reduction is applied to the dimensions of the activation map, while the depth volume remains unchanged. There are two typical ways to max-pooling: one use 2×2 stride-2 filters that cover the whole spatial dimension associated with the input image data, and the other employs overlapping pooling with a 2×2 stride and a 3×3 kernel. Larger kernel sizes are often avoided since they degrade model performance (Shea & Nash, 2015).

While for the last layer is named as fully connected layer. This specific layer is the most critical component in CNN topologies. It is made up of neurons that only make connections with neurons in the two neighbouring layers. various neurons are critical in the transmission and translation of information across various surrounding layers, allowing the network to extract detailed characteristics and patterns. These direct connections let the layer to capture intricate linkages and dependencies within the input data, which helps the network generate more accurate predictions (Shea & Nash, 2015). In other

hands, those hidden and multiple layers allow the deep learning to have better machine learning performance and allow us to deal with more complicated task.

2.5.2 CNN in Plant Disease Detection

In the research that is conducted by Mohanty et al. (2016), they dig into a comparative comparison of two well-known deep learning algorithms, namely AlexNet and GoogleNet. Both of these algorithms are classified as CNNs, which is another name for the category of neural networks. The research makes use of a significant dataset that was obtained from the PlantVillage dataset, which is accessible to the general public. This collection has an astounding 54,306 photos that are separated into 38 different categories. The sheer volume and diversity of this dataset highlight the considerable complexity inherent in deep learning tasks, setting them apart from the more manageable challenges addressed by shallow learning approaches.

According to Mohanty et al. (2016), the research starts with the preprocessing of data to resize the images to the size of 256 x 256 pixels. They divide the experiment into different version to get the best performance among different settings. The experiment involves a variety of essential criteria. Firstly, this study investigates and compare the results from two DL systems, specifically AlexNet and GoogleNet. Additionally, this study examines several training methodologies, such as transfer learning and training from the ground up. Additionally, the analysis considers the different sorts of datasets, such as those containing color images, grayscale images, and images specifically focused on leaf segmentation. The experiment examines various test-train split scenarios, encompassing different distributions of training and testing sets, namely 80-20%, 60-40%, 50-50%, 40-60%, and 20-80%. The research was focusing on the getting the hyperparameter setup that will have the best results in the proposed tasks. (Mohanty et al, 2016)

AlexNet is categorized under CNN because it is built under the theory of CNN. The model structure of AlexNet is built up of a total of five convolutional layers and three totally connected layers. and is activated by softmax activation function or in other word last layer is softmax. The network comprises several critical layers: the initial two convolutional layers, denoted as conv1 and conv2, are immediately followed by normalization and pooling

layers. The neural network is finalised by a crucial fully connected layer known as fc8. This layer produces 38 output nodes, each of which corresponds to a distinct class within the dataset. These outputs subsequently serve as input to a softmax layer, whose role is to perform exponential normalization (Mohanty et al, 2016).

While The GoogleNet design is characterised by its increased depth and width, consisting of 22 convolution layers. Despite this expansion, it is noteworthy that GoogleNet manages to retain a relatively lower parameter count of 5 million, in contrast to AlexNet's substantially higher parameter count of 60 million. One notable characteristic of GoogleNet is its implementation of the "network-in-network" architecture using inception modules. The modules under consideration incorporate parallel combinations of 1x1, 3x3, and 5x5 convolutions together with max-pooling layers, enabling the model to effectively capture diverse input concurrently. To enhance computational efficiency, the use of 1x1 convolutions is implemented both before to and subsequent to the bigger convolutions. Additionally, the merging of outputs from these parallel layers is achieved using filter concatenation layers (Mohanty et al, 2016).

In the result of research, the GoogleNet is performing better than the AlexNet. This is because GoogleNet has more convolutional layer compared to AlexNet. Then the transfer learning is always having better performance than the training from scratch because of Transfer learning is a prominent machine learning approach that involves the adaptation of a pre-trained model from one task to another activity that is similar, though distinct. This methodology proves to be especially advantageous in situations when there is a scarcity of annotated data accessible for the intended objective, since it enables the model to use the extensive information acquired during the reference work. Then the dataset which has color based can provide better performance to the model because the model can also learn from the color and relate the information. The optimal setup among the proposed parameters which is demonstrating the highest performance, involves utilizing the color-based database in conjunction with GoogleNet for transfer learning, and employing an 80-20 train-test split which achieved accuracy of 99.34% (Mohanty et al, 2016).

Zaki et al. in 2020, employed the MobileNetV2 CNN model for the task of recognition and classification of tomato plant illness for their research. According to Zaki et al. (2020), MobileNet is deep learning framework specifically developed to cater to the computing constraints of mobile devices. Google later produced a refined version of the technology, referred to as MobileNet V2, which featured slight modifications compared to the MobileNet V1. The fundamental component of the network, known as separable convolution, stays unchanged. In an independent investigation, the utilisation of MobileNet version 2, which had been pre-trained on ImageNet datasets, was employed to extract characteristics from photos of fruits. The study posited that the parameters of the model were decreased from 4.24 million to a mere 3.47 million, resulting in enhanced accuracy. The advantages of MobileNet V2 compared to other CNN models is it can be easily fit into mobile or embedded system use.

Table 2.4. Description of training hyperparameter. (Source: Guo et al., 2022)

Parameter	Description
Epochs	The total number of training iterations using the entire dataset from the training set.
Batch Size	Number of images in a batch
Optimizer	Minimizing the loss function
Learning Rate	Amount of gradient error which will be apply to update current weights.
Loss Function	Difference between the true and predicted value.

The Table 2.4.2.1 shows the description of training hyperparameter in a deep learning model. In the research of MobileNet V2, Zaki et al. (2020) collected PlantVillage dataset that consists of 4671 images which consists of a total of 1590 photos depicting healthy tomato leaves, 952 images representing leaf mold, and 1756 images showcasing late blight, and 373 mosaic virus images with size of 224 x 224 pixels for training and testing. Then, the team applied training from scratch on MobileV2 and research on the performance of MobileNet V2 under different setup of hyperparameter and choose the best among them. Zaki et al. (2020) did a thorough investigation on the performance

of MobileNet, employing a range of optimisation techniques such as Adagrad, Adam, SGD, RMSprop, and Nadam. In addition, the researchers conducted an evaluation of the model's performance using several learning rates, namely 0.00001, 0.0001, and 0.001. The study also examined the effects of several test-train split ratios, including ratios of 9:1, 4:1, 7:3, and 3:2. Moreover, the investigation examined the impact of varying batch sizes (48, 32, and 16) on the performance of the model. After all comparison and matching, the team obtain the optimum accuracy of 95.94% when MobileNet V2 is trained with Adagrad optimization algorithm, with a batch size of 16, a learning rate of 0.001, and a 4:1 data split between training and testing.

The study conducted by Menon, V. et al. (2021) investigated the efficacy of CNN models, which are VGG16, Inception V3, and Xception, in tasks of plant disease detection. The researchers evaluated the performance of these models using two separate datasets. One dataset was made of laboratory photographs that were utilized for the purpose of training, which is collected from PlantVillage, whereas the other dataset consisted of images that were captured under natural lighting settings and included environmental backgrounds, which is collected from PlantDoc. Visual Geometry Group 16 (VGG16) is a deep learning CNN architecture. It is distinguished by its composition of 16 weight layers, comprising of 13 convolutional layers and 3 fully linked layers. The VGG16 model is widely recognized for its simplicity and efficacy in the domain of picture classification. It has served as a fundamental model in the field of computer vision, frequently employed as a pre-trained framework for transfer learning across many applications.

While the Inception V3 architecture is recognized because it has extreme good performance in challenges related to image identification. The network incorporates an innovative "Inception module" which utilizes numerous concurrent convolutional processes with varying kernel sizes. In details, Inception V3 is a deep neural network architecture that has exceptional efficacy in image classification tasks due to its extensive depth of 48 layers and a vast number of parameters. Lastly, Xception, also named as "Extreme Inception", a CNN structure renowned for its utilization of depth wise separable convolutions. These convolutions are designed to decrease computational complexity while maintaining a high level of expressive capability. The purpose of its design was

to optimize the efficiency of CNNs through the substitution of different and multiple convolutional layers with depthwise separable convolutions. These convolutions are composed of a depth-wise convolution then a pointwise convolution (Menon, V. et al., 2021).

In the study by Menon et al. (2021) discovered that their convolutional neural network (CNN) models exhibited superior performance compared to alternative models when they were trained using photos from the PlantVillage database. The VGG16 model demonstrated superior performance, attaining a notable training accuracy of 92% and an equivalent validation accuracy of 92%. The Xception model had a strong performance, achieving an 88% training accuracy and an 88% validation accuracy. The Inception V3 model demonstrated satisfactory performance on the PlantVillage database, with an 85% accuracy during training and an 87% accuracy during validation. But the performance of the CNN models is bad with the PlantDocs database, and according to Menon, V. et al. (2021), PlantVillage dataset demonstrates enhanced accuracy as a result of its controlled experimental conditions; nonetheless, its applicability for real-time field implementation is limited. This potential is particularly evident in the enhancement of picture resolution, which can lead to enhanced accuracy in disease detection.

Rather than that, in the research paper by Saxena, O. et al. (2021). The research introduces a method on CNN models that will achieve accuracy of above 95% on the task identifying and classifying early blight and late blight disease on potato leaves. The research started by collecting the images from Kaggle, where the dataset consists of 900 images that was divided into 3 groups which are healthy potato leaf, early blight, and late blight. Before the training of the CNN models, Saxena, O. et al. (2021) proposed the image preprocessing on the dataset because the amount of the training dataset is too less to allow the model to have accuracy more than 95%. The team applied image resizing that convert the images to size 256 x 256 x 3 pixels, then color adjustment using RGB channel, orientation, and image augmentation. The image preprocessing can remove the noise from the dataset. After the preprocessing of image data, segmentation and feature extraction on database, the result of the CNN models is able to achieve better performance where AlexNet reached accuracy of 98.51%, while GoogleNet reached accuracy of 99.10%.

2.5.3 YOLO in Plant Disease Detection

The CNN model named as YOLO is initially and mainly designed for the purpose of real-time identification and computer vision applications. What distinguishes it is its capacity to rapidly identify several objects in a single iteration of the neural network. This is achieved by partitioning an image into a grid and generating predictions for bounding boxes and class probabilities within each individual grid cell. The YOLO algorithm has garnered significant attention and widespread acclaim owing to its exceptional characteristics, such as its notable efficiency, precision, and speed, rendering it highly applicable across a diverse array of domains. The scholarly article entitled "Object detection utilizing YOLO: obstacles, architectural advancements, datasets, and applications" produced by Diwan, T. et al. (2021), explores the architectural framework of YOLO.

The architectural architecture of YOLO is influenced by GoogLeNet and is subsequently implemented for the purpose of object detection, utilizing the VOC Pascal Dataset from 2007 and 2012. In contrast to the utilization of inception modules in GoogLeNet, YOLO employs a combination of (1×1) and (3×3) convolutional filters. The initial convolutional layer in the YOLO model utilizes a filter with dimensions of (7×7) . YOLO's architecture comprises two fully connected layers and twenty-four convolutional layers in total. Following four convolutional layers in succession, the architecture executes a max-pooling operation. The model incorporates two crucial components: a (1×1) convolution operation and global average pooling. Diwan et al. (2021) assert that YOLO possesses inherent characteristics that make it exceptionally well-suited for tasks involving real-time object detection and can be modified to function in applications other than GPUs.

Generally, the YOLOv8 is a model that is developed under ultralytics. The development of YOLOv8 is referred to YOLOv8 and several updates and improvement is applied to YOLOv5 in order to create YOLOv8. YOLOv8 consist of three main parts in its structure which are Backbone, Neck, and Head. The backbone of YOLOv8 is named as CSPDarknet-53 which is the core section that can extract the image features. It used 53 convolutional layers and applied cross-stage partial (CSP) connections to increase the speed of data transfer. The

layers and specific structures build up a lightweight and powerful filter that can extract more details from complex image.

After YOLO was introduced, many researchers give significant attention contribute to optimize the completion of ML model's functionalities to make it suitable for real-life task. In the research paper "Crop Disease Detection Using YOLO" by Morbeka, Parihar and Jadhav (2020) proposed research regarding to the comparison of YOLOv3 with the ANN models. Dataset which is used for model testing and training comprises 25,044 images distributed across nine distinct plant classes YOLOv3, the third iteration of the YOLO, object recognition model which has a number of significant enhancements. The system employs a logistic classifier for efficient computation of item probabilities in a multi-label classification setting. According to YOLOv3 model incorporates residual skip connections, up sampling techniques, and operates on three distinct scales in order to facilitate object detection. Detection kernels are subsequently applied to feature maps that have been derived from three distinct locations within the network. This process yields detections that possess diverse dimensions and classes. The architecture of YOLOv3 incorporates a feature extractor called Darknet-53 and a multi-scaled detector that is capable of processing feature vectors from various scales. The ultimate result comprises of detections at various scales, denoted as tuples that indicate the dimensions and probabilities of different classes According to Morbeka, Parihar and Jadhav (2020), they proposed the research to investigate and compare the performance between ANN with SUFT, FOANN with SURF OANN with SUFT and the YOLOv3. In the research, YOLOv3 obtained the greatest performance with mean accuracy of 99.50%, recall of 0.88 and precision of 0.933.

2.5.4 Research History on Application of YOLO

In the year 2021, Shill and Rahman, they propose an experiment to do comparison between YOLOv3 and YOLOv4. According to Shill and Rahman (2021), YOLOv4 is an updated version of YOLOv3. The main difference of YOLOv4 compared to YOLOv3 is it has a backbone network which consist of CSPDarknet53 architecture. The CSPDarknet53, which serves as the backbone network in YOLOv4, plays a crucial role in extracting features for object detection. The technology serves a dual function, as it is capable of capturing

complex patterns, edges, textures, and structures present in an image, while also understanding both detailed and overarching contextual information. The design of CSPDarknet53 aims to mitigate challenges commonly seen in deep networks, such as the vanishing gradient problem. Significantly, the concept of a "cross-stage hierarchy" is introduced, wherein feature maps are divided at distinct stages to undergo individual processing before their outputs are combined. This facilitates the transmission of information and the dissemination of gradients, hence fostering efficient comprehension of objects in various settings. The outcome is an enhanced feature representation that effectively handles fluctuations in object scale, orientation, and surroundings, hence leading to an improvement in the accuracy of object detection.

In YOLOv3 and YOLOv4, image preprocessing is more complicated than image categorization. Resizing the images to 416x416 yields $52 \times 52 \times 3 \times (4+1+\text{number of classes})$ and $13 \times 13 \times 3 \times (5+\text{number of classes})$ network outputs for both models. Creating a model-training format using data labels is critical. Random flipping, cropping, translating, Mosaic data augmentation, and DropBlock regularisation were used to diversify the training dataset and reduce class imbalance. The training phase was subsequently carried out on a single GPU using Google Colab. During the training, they set batch size to 64, with 0.001 learning rate, and also momentum value of 0.9, and the decay rate is 0.005. In the course of the training procedure, a total of 9 anchors and 105 filters were employed. The selection of these values was based on a formula, specifically 3 multiplied by the sum of 5 and the number of classes (C). In this particular case, the number of classes was 30. The training process consisted of a total of 60,000 batches, each executed with precise step sizes. The evaluation on the performance of the model is focused on the Mean Average Precision (mAP) metric, which considers the Intersection over Union (IoU) with a threshold of 0.5. The evaluation process involves doing 10,000 iterations to determine the ideal parameters. To gauge the model's performance, mAP was employed to assess the accuracy of object predictions by determining the IoU threshold between the ground truth and the anchor for each detection. The formula of mAP as shown in below:

$$mAP = \left(\frac{1}{N}\right) \sum_{r=0}^1 (p(r)) \quad (5)$$

The performance of the YOLOv4 is better than YOLOv3 which is proven in the result of the research by Shill and Rahman (2021) with mAP is 55.45 for YOLOv4 while 53.08 for YOLOv3.

The research produced by Dai and Fan (2022) presents the introduction of a unique network architecture known as YOLOv5-CAcT. The study incorporates a comparison evaluation with the original model, YOLOv5, to gauge the system performance. YOLOv5 is an advanced target detection method that falls within the One-Stage category. YOLOv5 presents numerous advantages in comparison to its previous iterations, notably a substantially reduced weight file, rendering it well-suited for real-time detection on embedded devices. The YOLOv5 framework consists of three essential components, namely the Neck Network, the Backbone Network, and the Detect Network. These components collaborate to extract meaningful information, minimize computational requirements, and enable effective training.

While for the YOLOv5-CAcT which are proposed by Dai and Fan (2022) based on YOLOv5 and they have different features, for example, YOLOv5 use CSPDarknet53 as its backbone but modified YOLOv5 add Activation Compression (AcTNN) on top of CSPDarknet53 to decrease the number of parameters without effect its performance, while for the neck of the architecture, original YOLOv5 applied Path Aggregation Network (PANet) while modified YOLOv5 added Refinement Anchor (RA) to improve the accuracy of object detecting by refining anchor boxes. Then the YOLOv5-CAcT is also apply Focal Loss (FL) and Smooth Binary Cross Entropy (Smooth BCE) to help reduce the imbalance between positive and negative sample. The dataset used is taken from PlantVillage and consists of 52,589 images with 59 diseases from 10 different crops species with size 512 x 512 pixels. (Dai & Fan, 2022)

Before training of the models, both models are set to their own hyperparameter that will provide optimum performance. YOLOv5 utilizes BCELoss (Binary Cross-Entropy Loss) in conjunction with SGD (Stochastic Gradient Descent) optimization. The model is trained using a batch size of 128

with size of 384 pixels. Initial learning rate is set at 0.0032, which is then adjusted to 0.12. The momentum parameter is assigned a value of 0.843, and the weight decay is configured to be 0.00036. A preheating parameter of 5 is utilized. In contrast, the YOLOv5-CAcT model incorporates AcTNN, model pruning, also knowledge distillation techniques. It utilizes the same loss function and optimizer as the original YOLOv5 model, but with a batch size of 64 and a bigger input image size of 512 pixels (Dai & Fan, 2022). The remaining hyperparameters are kept consistent with the pre-training phase. From the result of the research, YOLOv5-CAcT has better performance compared to YOLO V5. The below table shows the results of proposed YOLO models:

Table 2.5. Test results of YOLOv5 and YOLOv5-CAcT (source: Dai & Fan, 2022)

Models	Precision (%)	Recall (%)	F1-score (%)	Accuracy (%)
YOLOv5	87.2	92.6	89.8	94.3
YOLOv5-CAcT	90.7	92.3	91.5	95.6

In the research paper titled "Tea leaf disease detection and identification based on YOLOv7 (YOLO-T)" by Soeb et al. (2023), they compared the performance of YOLOv5 and YOLOv7. They prepared a dataset that consisted of 4000 of tea leaf images and all capture from the tea field with 640 x 640 pixels size with camera. The images are divided to 5 categories consist of leaves infected by red spiders, tea mosquito bugs, black rot, brown blight and leaf rust. According to Soeb et al. (2023), YOLOv7 is a progressive advancement in the realm of model structures, building upon the foundations laid by YOLOv4, Scaled YOLOv4, and YOLO-R. The architecture of YOLOv7 incorporates an enlarged ELAN (E-ELAN) backbone that utilises several techniques like as expansion, shuffling, merge cardinality, and group convolution. These approaches are employed to augment the learning capacity of the model while ensuring the preservation of gradient flow. Additionally, the incorporation of compound model scaling is employed to withstand the fundamental characteristics of the model. Furthermore, YOLOv7 integrates "bag-of-freebies" (BoF) methodologies in order to enhance performance while keeping training expenses at a minimum. The result as shown as below:

Table 2.6. Comparison of evaluation indicators between YOLOv5 and YOLOv7 (source: Soeb et al., 2023)

Model	Precision (%)	Recall (%)	F1-score	Accuracy (%)
YOLOv5	95.4	96.4	0.958	96.1
YOLOv7	96.7	96.4	0.965	97.3

In the research proposed by Susa et al. (2022), they proposed an investigation to figure out how YOLOv3 perform in the task detection of cotton and leaf classification. The research introduced a crucial step for YOLO which is image annotation. Image annotation refers to the procedure of incorporating labels, metadata, or visual indicators onto photographs in order to elucidate and ascertain the items, regions, or features present within them. Various annotation approaches commonly employed in computer vision tasks include bounding boxes, segmentation masks, key point markers, and text labels. The inclusion of annotations is crucial in the training of YOLO models, as it facilitates the recognition and comprehension of picture content. After the training of the model with dataset that consists of 400 images with healthy and disease leave, the proposed YOLOv3 obtain 95.09% of accuracy. Subsequently, the performance of the model was assessed by utilising a video consisting of eight images, with two images representing each class. This movie had a total duration of 16 seconds and had 480 individual image frames. The system effectively identified and categorised objects inside the video frames, attaining notable accuracy rates of 98% and 99%. Furthermore, the utilisation of a camera for live stream detection exhibited exceptional performance, as where accuracy rates that varied between 74% and 100%. This observation demonstrates the efficacy of the model in accurately identifying and categorising items inside dynamic, real-world situations.

2.6 Introduction to Vision Transformer

The Transformer model was initially introduced for natural language processing tasks, where it achieved remarkable success. Encouraged by its prowess in language-related tasks, researchers have sought to extend the Transformer's capabilities to image classification tasks. Their objective has been to

demonstrate that Transformers can excel in tasks beyond NLP, including image classification and object detection. In the year 2021, the first transformer named “Vision Transformer” (ViT) model that can be used for image classification is proposed by Dosovitskiy et al. (2021). The main innovation of ViT in order to fit transformer model into image classification task is the ability to handle image data as sequence without increasing computing complexity. In the paper, author mentioned that the team proved the performance of transformer by comparing ViT with ResNet, while ResNet is a very well performed CNN model in image classification. The model architecture overview is shown in Figure 2.5 below.

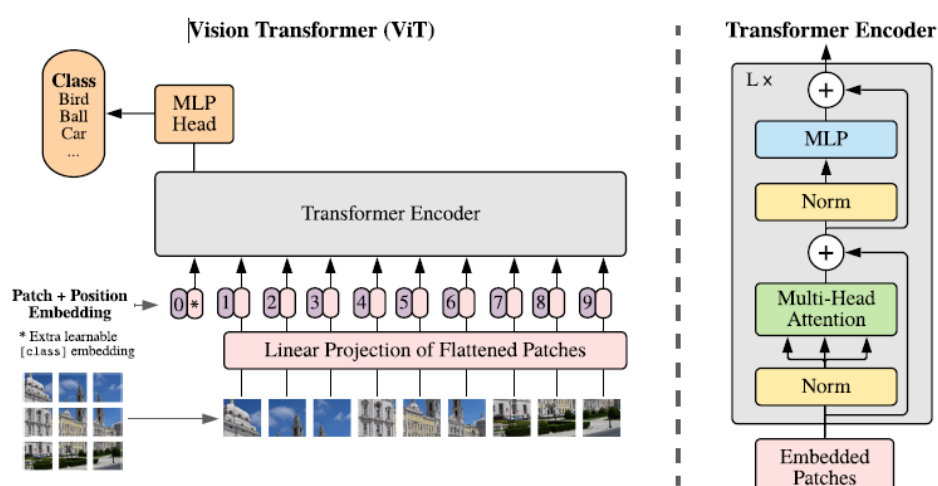


Figure 2.5. Model Overview of Vision Transformer (source: Dosovitskiy et al., 2021)

2.6.1 ViT Architecture: Patch Embedding

According to Dosovitskiy et al. (2021), the images will firstly proceed with the layer of patch embedding, where it will split the image to equally sized patches, then the patches will be flattened. The patches were set to the size of $16 \times 16 \times 3$ within the whole image. The first patch is derived directly from upper left corner of the input image, while the final patch is obtained from the lower right corner. In this manner, the patches have the capability to be organised in a linear configuration, which is specifically delineated as the num-patches.

2.6.2 ViT Architecture: Linear Embedding and Position Embedding

Linear embeddings are constructed from every patch in the picture, and these embeddings are known as "Patch Embeddings." As a consequence, each patch is represented by a vector of dimensions 1 by 768 (16 x 16 x 3). Following the division of the original image into 196 smaller image patches, each measuring 14 by 14 pixels, the aforementioned alteration can be implemented. Subsequently, a linear projection layer is employed to process the aforementioned patches and generate the matrix representing the embedding of the image. This subsequent phase occurs subsequent to the preceding stage. The patch embeddings, which have dimensions of 196 by 768 pixels, function as representations for the diverse patches that constitute the entirety of the image. Once the identification of a patch creation as a layer was made, the Patch Encoder layer was constructed. In addition, the projected vector incorporates a learnable position embedding. The learnable class token is used to generate as a global representation of the input image of the input image, which can allow the model to understand the whole image.

2.6.3 ViT Architecture: Transformer Encoder Layer

After the position embedding and patch application processes are completed, the image patches are transferred to transformer encoder layer. Three primary components comprise a transformer encoder layer: layer normalisation, multi-head attention, and a multi-layer perceptron. The Transformer Encoder Layer is an integral component of the image patch processing pipeline within a Vision Transformer (ViT). At the outset, the input will be passed through the Multi-Head Attention block after being filtered through a Layer Norm within that layer. The input is subsequently supplied to the MLP Block. In addition to two linear layers, the MLP block includes a GELU nonlinearity. To derive the final output from a solitary layer of the Transformer Encoder, the initial inputs and outputs from the Multilayer Perceptron (MLP) block are merged once more. The employment of multi-head self-attention to capture interdependencies among the patches, while the employment of layer normalisation and residual connections serves to stabilise and enhance the learning process. The stacking of Encoder Layers in ViT facilitates the capturing of hierarchical characteristics

and dependencies among patches, hence serving as the foundation for picture classification and other computer vision tasks.

2.6.4 Transformer-Based Model in Plant Disease Detection

In the research paper titled by Boukabouya et al. (2022). The research team fine tune the hyperparameter of a ViT model to optimize its performance toward the function of the plant disease classification. Dataset from the PlantVillage is taken for model training and it consists of 18,345 of tomato leave images that consist of 10 different disease samples.

From Boukabouya et al. (2022), the input is initially divided to several patches, each measuring 16 by 16 pixels and including RGB channels. The patches are arranged linearly, covering the entirety of the image. Subsequently, linear embeddings are calculated for every patch, resulting in vectors of size 1×768 ($16 \times 16 \times 3$). The embeddings in question integrate positional information. The input embeddings are taken as input by the Transformer Encoder, which has 12 similar layers. Each layer in the architecture consists of a Multi-Head Attention block, Layer Normalisation, and an MLP block with two linear layers and GELU nonlinearity. Furthermore, the Patch Encoder layer applies a linear transformation to each patch, incorporating a position embedding that may be adjusted through learning. In the end, the extracted characteristics are inputted into a Multi-Layer Perceptron in order to classify images. Lastly, the research obtained the ViT employs a patch size of 16, a projection dimension of 128, 4 attention heads, 16 layers, 512 MLP units, a learning rate of 0.0005, Adam optimizer, and Gelu activation function has the optimum performance which is up to accuracy of 99.7%.

In their study publication, Thakur et al. (2022) presented the PlantXViT model as a solution for the task of detecting and recognising plant diseases. The PlantXViT model represents an innovative hybrid methodology that combines the functionalities of Vision Transformers (ViT) and Convolutional Neural Networks (CNN) in order to accurately detect and classify plant diseases. The model is built by combining two convolutional blocks from the VGG16 architecture, an Inception block, and the ViT architecture. The proposed model accepts input pictures with dimensions of $224 \times 224 \times 3$. It incorporates Convolution blocks derived from VGG16 and Inception v7,

together with components from the Vision Transformer (ViT) architecture. These ViT components consist of MHA and MLP modules that employ linear projections. The VGG16 Convolutional blocks are employed as the initial stage for processing the input picture. Subsequently, an Inception-like multi-level feature extraction block is utilised to improve the learning of local features. The feature map is transformed into patches, which are then subjected to linear projection to produce feature vectors. These feature vectors are subsequently processed by four transformer blocks. The classification result is generated by using a global pooling layer with fully linked layer with softmax activation.

From the research, several datasets are used to do training and testing on the PlantXViT model such as PlantVillage dataset that has total of 54,305 images under 38 different classes, Embrapa dataset that has total of 46,379 images under 93 classes, Apple dataset, Maize dataset and Rice dataset. The model gets the best performance when it is trained with PlantVillage dataset with patch size of 5 and Nadam optimizer. With the setup of PlantXViT model achieved impressive performance metrics, including a low loss of 0.04, high accuracy at 98.86%, excellent precision, recall, and F1 score at 98.90%, 98.81%, and 98.85%, respectively, along with a remarkable AUC of 99.92% and a kappa score of 0.99.

Yu, Xie, and Huang (2023) introduced a more advanced model called the Inception Convolutional Vision Transformer (ICVT) based on the ViT model. ICVT comprises several stages, including soft split embedding, depth-wise convolutional transformer block, and inception transformer block, making it a more complex model than both CNNs and ViTs. The utilisation of soft split token embedding is a method employed to effectively collect localised information derived from neighbouring pixels and patches within a transformer block. The process involves dividing the input tokens into smaller sub tokens and subsequently representing each sub token's local environment through vector embedding. The inclusion of the transformer block enables the acquisition of associations between distinct segments of the input, regardless of their immediate proximity. While the depth-wise convolution transformer block is a variant of the transformer block that use depth-wise convolutions in lieu of conventional convolutions. Depth-wise convolutions refer to a specific sort of convolution operation wherein a singular filter is exclusively applied to each

individual input channel. This characteristic renders them more effective in comparison to conventional convolutions, and additionally enables them to acquire localised information from neighbouring pixels and patches.

The research used four different datasets to investigate the performance of the ICVT in the plant disease detection and classification. The first dataset, referred to as PlantVillage, consisted of a total of 55,448 photos that were categorised into 38 distinct plant disease classes. These classes encompassed a wide range of plant species, amounting to a total of 14 different types of plants. The second dataset, known as the ibean leaf image dataset, has photos that have been classified into three distinct categories. However, the specific quantity of photographs within each category has not been explicitly stated. The third dataset, referred to as AI2018, consisted of a total of 31,718 training photos and 4,540 validation images. These images were categorised into eleven plant categories, which were further broken into 61 subcategories based on illness, degree, and species. The fourth dataset, referred to as PlantDoc, comprised a total of 2598 photos. These images were selected to represent 13 distinct plant categories, each associated with 27 different species affected by various diseases. The ICVT model obtain the mean accuracy of 99.94%, average precision of 0.9989, average recall of 0.9988, and average F-1 score of 0.9989.

Guo, Lan, and Chen (2022) introduced a more advance and effective model than ViT, which is Convolutional Swin Transformer (CST) that can help in image classification and plant disease detection. Before training, Guo et al. (2022) prepared datasets that are provided with encompass various plant diseases. The first dataset, known as the Cucumber Plant Diseases Dataset, contains a total of 679 photographs. These images depict both healthy cucumber leaves and leaves affected by rust. The second dataset, named the Banana Leaf Disease Images dataset, comprises 1,288 images. These images showcase healthy banana leaves, as well as leaves infected with 'Xanthomonas' and 'Sigatoka'. The third dataset, referred to as the Potato Disease Leaf Dataset, consists of 4,062 images. The leaves depicted in these images are categorized as follows: healthy leaves, foliage impacted by early blight, and leaves impacted by late blight. In conclusion, the Plant Village dataset comprises a subset comprising 4,021 images. These images are distributed among ten distinct categories, each representing a different plant disease.

According to Guo et al. (2022), CST is the incorporation of transformer and convolutional blocks that can enhance the efficacy of plant disease detection. The backbone of the CST is using the original Swin Transformer architecture which are organised in pairs along with patch merging layers in each step. The CST model has 4 stages where each stage consists of patch merging and a Swin Transformer block. The model first extracts local features from the image and subsequently use self-attention to collect long-range dependencies among these features. The patch merging layer is an essential component within the design as it performs the critical function of unifying feature maps from the preceding level through the process of averaging. The self-attention mechanism plays a crucial role in facilitating the model's ability to identify connections between various regions of the image. In the CST model, this mechanism is influenced by the window-based self-attention process observed in the Swin Transformer. In order to enhance the acquisition of complex attributes, the CST model employs residual learning. The proposed methodology leverages existing features by including the output of the patch merging layer with the output of the Swin Transformer block. Additionally, in order to mitigate the issue of overfitting, the model utilises label smoothing cross-entropy as its loss function. The implementation of label smoothing serves as a preventive measure against the potential issue of the model overfitting to the training data. Lastly, by incorporating label smoothing, the model is encouraged to acquire more generalised features that transcend the specific characteristics of the training.

According to Guo et al. (2022), the self-attention processes that is proposed in the CST will process the entire information from the image to obtain three learnable matrices where are Queries (W^Q), Keys (W^K) and Values (W^V). The sequence is first started with the multiplication of the matrices to get $Q = IW^Q$, $K = TW^K$, $V = IW^V$, where I is the input sequence. Then the attention score can be calculated with the following equation:

$$Attention = softmax\left(\frac{QK^T}{\sqrt{d_q}}\right) \quad (6)$$

The training procedure of the model employed a configuration consisting of predetermined parameters. The training method consisted of 150 epochs, where each epoch encompassed a full iteration through the whole training dataset. During each epoch, batches of 20 images were simultaneously processed. The utilisation of the AdamW optimizer was implemented to minimise the loss function of the model, while a cyclic learning rate scheduler was utilised to dynamically adapt the learning rate throughout the training process. The upper bound for the learning rate was established at 0.0001, whilst the optimal learning rate of 0.000001 was identified using the CyclicLR scheduler's parameters. The employed loss function for training purposes was label smoothing cross entropy, which functioned as a metric to measure the discrepancy between the expected and actual values in the model's predictions. The below table is the result of the performance of CST:

Table 2.7. Performance of CST (Guo et al., 2022)

Model	Accuracy	Precision
CST-small	0.937	0.938
CST-base	0.942	0.942
CST-large	0.924	0.924

From the table above, The CST architecture comprises three distinct variants: CST-small, CST-base, and CST-large. CST-small is characterized by 96 channels in its initial stage, accompanied by 2 Swin Transformer blocks in each stage, and a patch size of 4x4. In contrast, CST-base features 96 channels in its first stage, incorporates 6 Swin Transformer blocks within each stage, and maintains a patch size of 4x4. Lastly, CST-large exhibits a configuration with 128 channels in its initial stage, integrates a substantial 18 Swin Transformer blocks per stage, and continues to employ a patch size of 4x4.

2.7 Summary

The table below concludes the performance of the related research on shallow learning model in the field of plant leaves disease detection and also classification. Although shallow ML approaches are recognised for their user-friendly nature and reduced computational demands, they may not presently

constitute the central area of research in the field of ML. However, shallow learning approaches continue to hold significance and worth in particular situations, such as when there are limitations on processing resources or when the importance of interpretability and transparency cannot be compromised.

Table 2.8. Performance Summary of Shallow Learning in Plant Disease Recognition

References	Backbone	Image Classes	Accuracy (%)
Mohan et al. (2016)	SIFT + SVM	3	91.1
Babu (2019)	SVM (linear kernel)	8	95.63
Babu (2019)	SVM (RBF kernel)	8	94.23
Babu (2019)	SVM (polynomial kernel)	8	95.87
Syafiqah Ishak et al. (2015)	ANN (MLPS)	2	99.15
Syafiqah Ishak et al. (2015)	ANN (RBF)	2	99.2
Kumari et al. (2019)	Neural Network (ANN)	2	92.5
Kusumo et al. (2018)	RGB with SVM (linear)	4	88
Kusumo et al. (2018)	RGB with SVM (RBF)	4	85
Kusumo et al. (2018)	RGB with NB	4	78
Kusumo et al. (2018)	RGB with DT	4	76
Kusumo et al. (2018)	RGB with RF	4	87

Table 2.9 below summarize the performance of the CNN and also YOLO from the past research on the topic of plant disease classification and detection. Compared to shallow learning, deep learning has its own advantages which can obtain better accuracy in the task. DL models particularly CNN and YOLO, excel in the self-extracting features and patterns from the image data, which allow the models achieved superior accuracy. Rather than that, with the advance technology, ML models nowadays is able to be trained with the support of greater computer resource with higher complexity which means that DL models also excel in learning from larger dataset which can directly boost their performance.

Table 2.9. Performance Summary of CNN and YOLO in Plant Disease Recognition

Reference	Backbone	Image classes	Accuracy (%)
Mohanty et al. (2016)	AlexNet	38	98.1
Mohanty et al. (2016)	GoogleNet	38	99.54
Saxena et al.(2021)	AlexNet	3	98.51
Saxena et al.(2021)	GoogleNet	3	99.1
Zaki et al. (2020)	MobileNet V2	4	95.94
Menon et al. (2020)	VGG 16	38	92
Menon et al. (2020)	Xception	38	88
Morbeka et al.(2020)	YOLO V3	24	99.55
Shill & Rahman (2021)	YOLO V3	30	53.08 (mAP)
Shill & Rahman (2021)	YOLO V4	30	55.45 (mAP)
Dai & Fan (2022)	YOLO V5	59	94.3
Dai & Fan (2022)	YOLO V5CACT	59	95.6
Soeb et al. (2023)	YOLO V5	3	96.1
Soeb et al. (2023)	YOLO V7	3	97.3

The table 2.10 below summarizes the performance of Transformer-Based models from the reviewed past research. Transformer-Based models has the performance that is similar to CNN and YOLO models from the comparison of table 2.10 and 2.9. This signifies a noteworthy accomplishment in the advancement of Transformer-Based models, which were initially introduced in the field of NLP. The efficacy and adaptability of Transformer-Based architectures, which were initially developed for NLP, are now being applied to the field of image processing, demonstrating their versatility. This transition signifies a substantial turning point in the progression of deep learning methodologies, creating fresh opportunities for advancement and implementation in diverse sectors outside their initial domains.

Table 2.10. Performance Summary of Transformer-Based Model in Plant Disease Recognition

Reference	Backbone	Image classes	Accuracy (%)
Boukabouya et al. (2022)	VIT	10	99.7

Yu et al. (2023)	ICVT	101	99.94
Guo et al. (2022)	CST-Small	17	93.7
Guo et al. (2022)	CST- Base	17	94.2
Guo et al. (2022)	CST-Large	17	92.4
Thakur et al. (2022)	PlantXViT	131	98.86

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

In response to the critical need for enhanced food security, this project leverages advancements in artificial intelligence to combat the insidious threat of plant diseases, safeguarding agricultural prosperity. To achieve this objective, two state-of-the-art models, YOLOv8 and ViT, will be utilized as specialized diagnostic tools. YOLOv8, characterized by its swiftness and vigilance, will efficiently scan leaf surfaces, identifying potential anomalies with hawk-like precision. Through comprehensive performance evaluations, the model demonstrating superior proficiency in both speed and accuracy will be designated the champion. Subsequently, this champion model will be disseminated across online and offline platforms, democratizing access to this potent disease-fighting arsenal. Empowered by this technology, even geographically remote farmers, unconstrained by internet connectivity, will be equipped to instantly diagnose their crops, enabling prompt and effective intervention strategies. Throughout the project, the photo will be collected online from Kaggle while the chili plant leaves picture will be collected individually from the planted chili. The photo will be prepared with two categories which are diseased leaves and healthy leaves.

3.2 Dataset

The dataset utilized in this study was sourced from Kaggle. The dataset obtained for analysis comprises a collection of plant leaf photos belonging to various classes and exhibiting different types of illnesses. Each image in the dataset has a resolution of 256x256 pixels. The training and testing dataset are separated into different folder paths. To reduce the training and testing time taken, the dataset is reduced in size by removing extra classes out from the prepared dataset.

The table below lists the classes and amount of the images that are contained in the dataset. The data is taken from the dataset named as New Plant Disease Dataset, which is taken from Kaggle, the dataset is divided into 8

classes to minimize the computation time and resource used during the training of the models.

Table 3.1. Information and statistical data of training dataset images

Class	Name	Number
00	Apple Scab	2016
01	Apple Rust	1760
02	Corn Common Rust	1907
03	Potato Early Blight	1939
04	Potato Healthy	1825
05	Tomato Early Blight	1920
06	Tomato Healthy	1926
07	Tomato Yellow Leaf Curl Virus	1961

After that the dataset is generated, it will be import into Google Drive, so that coding in Google Colab can access the image dataset. The setup of the used Cloud device in the Google Colab is Python 3 with Tesla, T4 GPU with 16GB of memory and 2,560 CUDA cores.

The second stage of the model training will be done under the support of device GPU RTX 3080. The dataset that will be use for the second stage model training is extended New Plant Disease dataset which contain 70 classes of plant leaves condition categories. In the second stage of model training, I prepared the dataset that is containing 121,975 images from 25 types of plants. This dataset is a combination of the dataset that can found online such as PlantVillage and PlantDoc dataset and also self-collected image dataset.

3.2.1 Data Augmentation

One of the issues during the data collection is some of the dataset size is too small for models training which is around few hundreds. Therefore, it is important to have dataset augmentation which can help to enhance the size of the dataset for better training and testing performance. From the data augmentation, we can generate the new training samples extra from the original pictures from several transformation such as image resizing, brightness changing, image rotation or orientation changing, increasing noise and also

exposure of the images. The one of the precautions during the image augmentation is the changing of the image contrast and also colour is not recommended in this specific task which is because it is important to have original colour of leaf in order to recognize its original condition.

3.3 Application of Vision Transformer in Image Classification

In the application of the vision transformer to the task of plant disease detection and classification, the path of the training and testing dataset is located and stored into a directory. After that, all of the images will be resized to 224x224 pixels which is the standard input size that can help to maintain the compatibility and consistency, also prevent the loss of information. Then, the training and testing data loader is created. Data loader is important in a vision transformer model because it can allow the larger dataset can proceed with data batching smoothly during the processing of image data. Then, a PyTorch function ‘torch.randn’ is used to generate a random tensor filled with numbers drawn from a standard deviation distribution. From the function, batch size is defined to 32, with 3 colour channels and image height and width of 224 pixels.

3.3.1 Training Hyperparameter on Vision Transformer

The vision transformer model that is used in the coding is ViT-Base. The ViT-Base model consists of 12 layers of transformer encoder layer and each layer consist of 768 embedding dimension, MLP size of 3072 and also 12 Multi-Head Attention. After that, the model of ViT architecture is turned to a pretrained model by applying pretrained ViT_B_16_Weights.IMAGENET1K from the pytorch. Table below shows the hyperparameter of the ViT model.

Table 3.2. Training hyperparameters of ViT on Google Colab

Parameter	Value
Epochs	15
Batch Size	32
Optimizer	Adam
Dropout	0.1
Learning rate	0.0001

Evaluation method will follow the description in Chapter **Error! Reference source not found.** Model Evaluation and Finalization.

3.3.2 ViT models Variants

In the development of a ViT model on image classification, the focus is to optimize the performance of the model. Therefore, it is important to train the different variants of the ViT model to investigate the performance of the model on our specific tasks. The model size denoted as “Base”, “Large” and “Huge” based on the different model size and parameters. Each different model sizes can also be divided into different input patch size in each ViT models, which are 16 x 16 and 32 x 32 patch size. From the model design, it is observed that the transformer sequence length exhibits an inverse proportionality to the square of the patch size. Consequently, computational resources escalate with diminishing patch sizes. In summary, the optimal-performing model is chosen from the following options: ViT-B/16, denoting the "Base" variant with a 16 x 16 input patch size; ViT-B/32, representing the "Base" variant with a 32 x 32 input patch size; ViT-L/16, signifying the "Large" variant with a 16 x 16 input patch size; ViT-L/32, indicating the "Large" variant with a 32 x 32 input patch size; and ViT-H/16, which corresponds to the "Huge" variant with a 16 x 16 input patch size.

3.3.2.1 Vision Transformer-B/16

Vision Transformer “Base” variant consists of two main type which is model with input patch size 16 x 16 pixels and 32 x 32 pixels input patch size. This represents one image will be divided into patches with each patch consist of 16 x 16 pixels or 32 x 32 pixels. One of the focus of the research is to observe the performance of the ViT models with different input patch sizes.

Generally, the smaller input patch sizes will help the models to capture finer details because the patches are focusing on smaller area of the image. This may help to improve the accuracy of the models but it will occupy larger computation resource in order to train the model. With the input patch size of 32 x 32 pixels, the models was able to capture the broader context of the image,

potentially improve the global understanding between the features. Rather than that, the larger input patch size will also reduce the computation resource and make the model to be faster and efficient.

Figure 3.3.2.1.1 below shows the architecture of ViT-Base with input patch size of 16 x 16 pixels. Figure belows summarizes that the ViT-Base model will have consist of total parameter of 87,461,384 with 6,162 trainable parameters and 87,455,232 of non-trainable parameters. The trainable parameters can help the models to understand the task from the training dataset while the non-trainable parameters is the fixed parameters for example the parameters from the pretrained components and also the positional information which can encode the relative positions of patches within the image, helping the model to understand spatial relationships.

Figure 3.3.2.1.1 below also summarizes the the input shape of the images from the dataset which is [1, 3, 224, 224] which represent 1 image, 3 color channels (RGB) and resolution of 224 x 224 with output shape of [1, 8] representing 1 vector of 8 elements, each representing a predicted probability for one of the 8 distinct classes. Within the model architecture of ViT-Base, the model consists of 12 stacks of identical Encoder Blocks which each containing multi-head self attention that allows the model to understand the relationships between patches in the image. MLP block is also one of the main component in the Ecoder blocks which can adds non-linearity to the model's predictions. In each encoder layer, the input shape of the ViT model is set to be [1,197,768] in the figure below. This represents that the image will pass through 12 layers of encoder layers with every images will be divided into 197 pathes as well as the input image is set to be size of 224 x 224 pixels. Lastly, the figure also summarizes the estimated total size of the whole model which is around 333.91 MB.

Layer (type (var_name))	Input Shape	Output Shape	Param #	Trainable
===== VisionTransformer (VisionTransformer)	[1, 3, 224, 224]	[1, 8]	768	Partial
└Conv2d (conv_proj)	[1, 3, 224, 224]	[1, 768, 14, 14]	(598,592)	False
└Encoder (encoder)	[1, 197, 768]	[1, 197, 768]	151,296	False
└Dropout (dropout)	[1, 197, 768]	[1, 197, 768]	--	False
└Sequential (layers)	[1, 197, 768]	[1, 197, 768]	--	False
└EncoderBlock (encoder_layer_0)	[1, 197, 768]	[1, 197, 768]	(7,087,872)	False
└EncoderBlock (encoder_layer_1)	[1, 197, 768]	[1, 197, 768]	(7,087,872)	False
└EncoderBlock (encoder_layer_2)	[1, 197, 768]	[1, 197, 768]	(7,087,872)	False
└EncoderBlock (encoder_layer_3)	[1, 197, 768]	[1, 197, 768]	(7,087,872)	False
└EncoderBlock (encoder_layer_4)	[1, 197, 768]	[1, 197, 768]	(7,087,872)	False
└EncoderBlock (encoder_layer_5)	[1, 197, 768]	[1, 197, 768]	(7,087,872)	False
└EncoderBlock (encoder_layer_6)	[1, 197, 768]	[1, 197, 768]	(7,087,872)	False
└EncoderBlock (encoder_layer_7)	[1, 197, 768]	[1, 197, 768]	(7,087,872)	False
└EncoderBlock (encoder_layer_8)	[1, 197, 768]	[1, 197, 768]	(7,087,872)	False
└EncoderBlock (encoder_layer_9)	[1, 197, 768]	[1, 197, 768]	(7,087,872)	False
└EncoderBlock (encoder_layer_10)	[1, 197, 768]	[1, 197, 768]	(7,087,872)	False
└EncoderBlock (encoder_layer_11)	[1, 197, 768]	[1, 197, 768]	(7,087,872)	False
└LayerNorm (ln)	[1, 197, 768]	[1, 197, 768]	(1,536)	False
└Sequential (heads)	[1, 768]	[1, 8]	--	True
└Linear (0)	[1, 768]	[1, 8]	6,152	True
===== Total params: 85,804,808 Trainable params: 6,152 Non-trainable params: 85,798,656 Total mult-adds (M): 172.47 ===== Input size (MB): 0.68 Forward/backward pass size (MB): 104.09 Params size (MB): 229.22 Estimated Total Size (MB): 333.91 =====				

Figure 3.1. Model Architecture of ViT-B/16

3.3.2.2 Vision Transformer-B/32

Figure 3.2 below shows the model architecture of the model ViT-B/32. The figure shows that the model consists total of 87,461,384 parameters and 6,152 parameters are trainable and 87,455,232 parameters are non-trainable. The model architecture with input shape of each image in RGB three color channels and is set to be 224 x 224 pixels. Then the output shape is [1, 8] representing 1 vector of 8 elements, each representing a predicted probability for one of the 8 distinct classes. Within the model architecture of ViT-Base, the model consists of 12 stacks of identical Encoder Blocks which each containing multi-head self attention and also MLP block within the model connection.

In each encoder layer, the input shape of the ViT model is set to be [1,50,768] in the figure below. This represents that the image will pass through encoder layers with every images will divide into 50 pathes as well as the input image is set to be size of 224 x 224 pixels. Lastly, the figure also summarizes the estimated total size of the whole model which is around 263.31 MB.

From the figure 3.1 and figure 3.2, model ViT-B/16 and ViT-B/32 has same model architecture, but the model ViT-B/32 require the image to be divided into lesser patches within the block for training and learning. This will result in the smaller model size of the model ViT-B/32.

```

=====
Layer (type (var_name))                               Input Shape      Output Shape     Param #         Trainable
=====
VisionTransformer (VisionTransformer)                 [1, 3, 224, 224] [1, 8]           768             Partial
├─Conv2d (conv_proc)                                  [1, 3, 224, 224] [1, 768, 7, 7]  (2,360,064)     False
├─Encoder (encoder)                                    [1, 50, 768]     [1, 50, 768]    38,400          False
│   └─Dropout (dropout)                                [1, 50, 768]     [1, 50, 768]    --              --
│       └─Sequential (layers)
│           └─EncoderBlock (encoder_layer_0)           [1, 50, 768]     [1, 50, 768]    (7,087,872)     False
│               └─EncoderBlock (encoder_layer_1)       [1, 50, 768]     [1, 50, 768]    (7,087,872)     False
│                   └─EncoderBlock (encoder_layer_2)   [1, 50, 768]     [1, 50, 768]    (7,087,872)     False
│                       └─EncoderBlock (encoder_layer_3) [1, 50, 768]     [1, 50, 768]    (7,087,872)     False
│                           └─EncoderBlock (encoder_layer_4) [1, 50, 768]     [1, 50, 768]    (7,087,872)     False
│                               └─EncoderBlock (encoder_layer_5) [1, 50, 768]     [1, 50, 768]    (7,087,872)     False
│                                   └─EncoderBlock (encoder_layer_6) [1, 50, 768]     [1, 50, 768]    (7,087,872)     False
│                                       └─EncoderBlock (encoder_layer_7) [1, 50, 768]     [1, 50, 768]    (7,087,872)     False
│                                           └─EncoderBlock (encoder_layer_8) [1, 50, 768]     [1, 50, 768]    (7,087,872)     False
│                                               └─EncoderBlock (encoder_layer_9) [1, 50, 768]     [1, 50, 768]    (7,087,872)     False
│                                                   └─EncoderBlock (encoder_layer_10) [1, 50, 768]     [1, 50, 768]    (7,087,872)     False
│                                                       └─EncoderBlock (encoder_layer_11) [1, 50, 768]     [1, 50, 768]    (7,087,872)     False
├─LayerNorm (ln)                                       [1, 50, 768]     [1, 50, 768]    (1,536)         False
├─Sequential (heads)                                    [1, 768]          [1, 8]           --              True
│   └─Linear (0)                                        [1, 768]          [1, 8]           6,152           True
=====
Total params: 87,461,384
Trainable params: 6,152
Non-trainable params: 87,455,232
Total multi-adds (M): 172.36
=====
Input size (MB): 0.60
Forward/backward pass size (MB): 26.41
Params size (MB): 236.30
Estimated Total Size (MB): 263.31
=====

```

Figure 3.2. Model Architecture of ViT-B/32

3.3.2.3 Vision Transformer-L/32

Figure 3.3 below shows the model architecture of ViT “Large” with input patch size 32 pixels. The figure shows that the model consists total of 305,518,600 parameters and 8,200 parameters are trainable and 305,510,400 parameters are non-trainable. The model architecture with input shape of each image in RGB three color channels and is set to be 224 x 224 pixels. Then the output shape is [1, 8] representing 1 vector of 8 elements, each representing a predicted probability for one of the 8 distinct classes. Within the model architecture of ViT-Base, the model consists of 24 stacks of identical Encoder Blocks which each containing multi-head self attention and also MLP block within the model connection.

In each encoder layer, the input shape of the ViT model is set to be [1,50,1024] in the figure below. This represents that the image will pass through encoder layers with every images will divide into 50 pathes as well as the input image is set to be size of 224 x 224 pixels. Lastly, the figure also summarizes the estimated total size of the whole model which is around 889.05 MB.

```

=====
Layer (type (var_name))      Input Shape      Output Shape     Param #         Trainable
=====
VisionTransformer (VisionTransformer)
├──Conv2d (conv_proj)        [1, 3, 224, 224] [1, 8]           1,024           Partial
├──Encoder (encoder)         [1, 3, 224, 224] [1, 1024, 7, 7] (3,146,752)    False
├──Dropout (dropout)         [1, 50, 1024]     [1, 50, 1024]   51,200         False
├──Sequential (layers)      [1, 50, 1024]     [1, 50, 1024]   --             --
├──EncoderBlock (encoder_layer_0) [1, 50, 1024]     [1, 50, 1024]   (12,596,224)  False
├──EncoderBlock (encoder_layer_1) [1, 50, 1024]     [1, 50, 1024]   (12,596,224)  False
├──EncoderBlock (encoder_layer_2) [1, 50, 1024]     [1, 50, 1024]   (12,596,224)  False
├──EncoderBlock (encoder_layer_3) [1, 50, 1024]     [1, 50, 1024]   (12,596,224)  False
├──EncoderBlock (encoder_layer_4) [1, 50, 1024]     [1, 50, 1024]   (12,596,224)  False
├──EncoderBlock (encoder_layer_5) [1, 50, 1024]     [1, 50, 1024]   (12,596,224)  False
├──EncoderBlock (encoder_layer_6) [1, 50, 1024]     [1, 50, 1024]   (12,596,224)  False
├──EncoderBlock (encoder_layer_7) [1, 50, 1024]     [1, 50, 1024]   (12,596,224)  False
├──EncoderBlock (encoder_layer_8) [1, 50, 1024]     [1, 50, 1024]   (12,596,224)  False
├──EncoderBlock (encoder_layer_9) [1, 50, 1024]     [1, 50, 1024]   (12,596,224)  False
├──EncoderBlock (encoder_layer_10) [1, 50, 1024]     [1, 50, 1024]   (12,596,224)  False
├──EncoderBlock (encoder_layer_11) [1, 50, 1024]     [1, 50, 1024]   (12,596,224)  False
├──EncoderBlock (encoder_layer_12) [1, 50, 1024]     [1, 50, 1024]   (12,596,224)  False
├──EncoderBlock (encoder_layer_13) [1, 50, 1024]     [1, 50, 1024]   (12,596,224)  False
├──EncoderBlock (encoder_layer_14) [1, 50, 1024]     [1, 50, 1024]   (12,596,224)  False
├──EncoderBlock (encoder_layer_15) [1, 50, 1024]     [1, 50, 1024]   (12,596,224)  False
├──EncoderBlock (encoder_layer_16) [1, 50, 1024]     [1, 50, 1024]   (12,596,224)  False
├──EncoderBlock (encoder_layer_17) [1, 50, 1024]     [1, 50, 1024]   (12,596,224)  False
├──EncoderBlock (encoder_layer_18) [1, 50, 1024]     [1, 50, 1024]   (12,596,224)  False
├──EncoderBlock (encoder_layer_19) [1, 50, 1024]     [1, 50, 1024]   (12,596,224)  False
├──EncoderBlock (encoder_layer_20) [1, 50, 1024]     [1, 50, 1024]   (12,596,224)  False
├──EncoderBlock (encoder_layer_21) [1, 50, 1024]     [1, 50, 1024]   (12,596,224)  False
├──EncoderBlock (encoder_layer_22) [1, 50, 1024]     [1, 50, 1024]   (12,596,224)  False
├──EncoderBlock (encoder_layer_23) [1, 50, 1024]     [1, 50, 1024]   (12,596,224)  False
├──LayerNorm (ln)           [1, 50, 1024]     [1, 50, 1024]   (2,048)        False
├──Sequential (heads)       [1, 1024]          [1, 8]           --             True
├──Linear (0)                [1, 1024]          [1, 8]           8,200          True
=====
Total params: 305,518,600
Trainable params: 8,200
Non-trainable params: 305,510,400
Total multi-adds (M): 355.75
=====
Input size (MB): 0.60
Forward/backward pass size (MB): 69.62
Params size (MB): 818.82
Estimated Total Size (MB): 889.05
=====

```

Figure 3.3. Model Architecture of ViT-L/32

3.4 Application of YOLO in Image Classification

In the application of YOLOv8 model in the task of plant disease classification, firstly the path of the training, testing and validation dataset is stored and located in a directory. Then the YOLOv8 model is installed with pip from ultralytics. After that, prepare the data.yaml file that contain the information of the training, testing and validation dataset directory locations. The training of YOLOv8 can be started by export the data.yaml with YOLOv8. Different from application of detection with YOLOv8, for classification we need to specify the task to classify in order to train the model for classification. The training of YOLOv8 will be focus on the pre-trained YOLOv8 model for classification from the ImageNet. The one of the differences of the YOLOv8 from ViT is the hyperparameter setup. During the YOLOv8 model training, the hyperparameter is defaulted and optimized for their pre-trained model. Changing of the hyperparameter setup is capable but not necessary.

3.4.1 Training hyperparameter of YOLOv8

Since the hyperparameter is optimized and default in the pre-trained model, the only setting that is changeable is the training epochs. In this research, the training epochs are set to be 15 epochs during the training on the Google Colab

but approach of early stopping is applicable during the model training as well as the model training reach its optimum accuracy. Below table shows the hyperparameter setup for YOLOv8 for all of the model branches.

Table 3.3. Training Hyperparameter of YOLOv8 in Google Colab

Parameter	Value
Epochs	15
Batch Size	16
Optimizer	SGD
Decay	0.05
Learning rate	0.001
Momentum	0.9

Evaluation method will follow the description in Chapter 2.3.6 Model Evaluation and Finalization. From the table above, there are some different settings compared to the setup of ViT from Table 3.3.

It is important to fine tune YOLOv8 model to achieve optimize performance. One of the hyperparameter that is different with ViT is decay (weight decay). This parameter is served to be a regularization approach to prevent the overfitting during the model training. The optimizer modifies the weights depending on the computed gradients throughout every iteration of YOLOv8 training. Weight decay adds a penalty term that somewhat reduces the weights' distance from zero. This enables the model to acquire more generalizable properties and lessens its dependence on certain weight values.

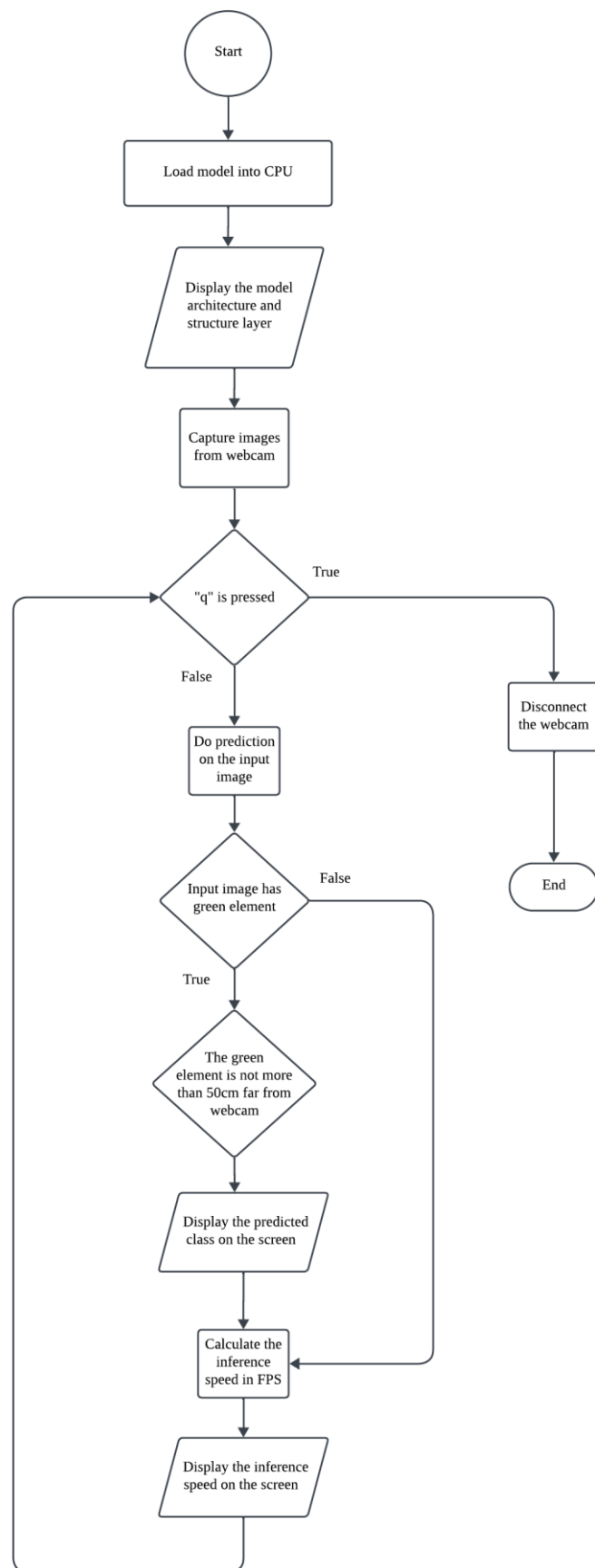
While momentum is a key hyperparameter for YOLOv8 training, it's a general optimization technique applicable beyond the Vision Transformer architecture. This parameter aids as one of the optimizers which can helps to navigate the complex loss landscape more effectively. Momentum takes the direction (gradient) from the previous update into consideration and adds a little forward velocity to each update. By doing so, the optimizer may be able to obtain the lowest loss more quickly and obtain updates in a consistent manner. It comes in very useful while navigating steep inclines or brief gullies.

3.4.2 YOLOv8 Model Variants

Similar to ViT, developer of YOLOv8 also designed pre-trained YOLOv8 model in different variants. The design purpose is to optimize the performance of the model in the different condition such as device setup and environment. These versions differ primarily between YOLOv8 variants lies in their backbone network. The size or the number of layers in the backbone of YOLOv8 allows the model to have variants on the model compatibility and also performance. The model size denoted as “Nano”, “Small”, “Medium”, “Large” and “X” which arranged in ascending order. Those model variants in YOLOv8 prioritize either speed or accuracy, and with the trade-off between two, we need to find the most suitable choice on our research task need.

In this research, I chose YOLOv8n-cl, YOLOv8s-cl, YOLOv8-m as the comparable models to ViT. The smallest model is YOLOv8n-cl which has the fastest speed but limited performance on its accuracy. This is because the smaller the model size, the least complexity of the backbone structure which may affect the model performance. YOLOv8n-cl is a specialized version of the YOLOv8 model designed specifically for image classification tasks. This model consists of 2.7 million of parameter with 3.1MB of model size, which is the smallest model that will be trained in this research. YOLOv8s-cl is one of the branches under the YOLOv8, which is larger than the YOLOv8-nano. The model consists of 6.4M parameters with 10.4MB of model size. In our research, I also employed YOLOv8m-cl, the largest model variant within the YOLOv8 family used in this study. It boasts 17 million parameters and a total model size of 31.9 MB, indicating its increased complexity compared to smaller YOLOv8 models.

3.5 Programming Flow Chart for Real-Time Webcam Inference



3.6 Software Overview

Google Colab, sometimes known as Google Colaboratory, is a robust cloud-based platform that enables users to write, execute, and exchange Python code with one another which is available on the Google Drive platform. With free GPU and TPU resources available, it provides a suitable environment for Python writing, especially for machine learning and data analytic jobs. Without any setup or installation required, users may create and use Jupyter notebooks straight in their web browsers with Google Colab. The trial training of ViT and YOLOv8 is carried out on the platform of Google Colab.

Google Colab has facilitated the accessibility of GPUs for free, like the Tesla T4 16 GB, which is good for machine learning development. To use Google Colab's services, users only need to register for an account. Importantly, Colab's support for a number of well-known machine-learning libraries such as PyTorch, TensorFlow, OpenCV, and Keras. However, there are limitations on how you can use Google Colab's computer resources. With a free Google Colab account, the Google GPUs are only used for 12 hours every day. Therefore, on this platform, just to examine the functionality of YOLOv8 and also ViT.

Other than Google Colab, one of the software platforms used is Virtual Studio Code (VS Code). Since the development of the models are conducted with Python language, therefore VS Code is one of the most suitable platforms for the model development and implementation. Besides that, it is also robust and free code editor available for Linux, Mac, and Windows platforms. With its many themes and plugins, VS Code may also be easily customized to meet your unique coding requirements and also allow the features such as debugging and code completion.

3.7 Hardware Overview

The research will continue with the upgrade of the hardware. The new session of model training will be conducted with local computer with external GPU connected to the system, in order to boost the training performance of the models and also provide better environment for the model inference and real-time application with lesser limitation. The research then been continued with Intel NUC 10th Gen Core i7 Mini PC with external GPU RTX3080. The mini-PC comes with a soldered-down 10th Gen Intel Core i7 processor which offers

6 cores and 12 threads for processing demands. The processor uses a 14-nanometer manufacturing process and runs at speeds between 1.1GHz and 4.7GHz. It is designed to generate minimal heat with a thermal design power of 25watts. The mini-PC also consists of expandable RAM up to 64GB for improved performance. The external GPU, Aorus RTX 3080 Gaming Box is connected to the mini-PC to enhance the performance of the models training and inference. The GPU can support real-time ray tracking at high resolutions and frame rates with its 10 GB of GDDR6X memory.

An additional crucial piece of hardware used in this investigation is a USB camera with a 1920x1080p maximum resolution and a 30 frames per second (FPS) frame rate. Throughout the process of inference as well as model deployment, this high-resolution camera is an essential tool. The study's overall success is greatly enhanced by its ability to record intricate details at a smooth frame rate, which guarantees precise and efficient performance in a variety of settings.

3.8 Gantt Chart

Task	Duration (Week)	W	W	W	W	W	W	W	W	W	W	W	W	W	W
		1	2	3	4	5	6	7	8	9	0	1	2	3	4
Final Year Project 1															
FYP Briefing	1														
Planning Project	2														
Study on related research	10														
Image data rearrangement	2														

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

Chapter 4 provides a comprehensive analysis of the real-time inference outcomes and the findings gained from training models in two separate contexts. During the early phase of the project, Google Colab was used to train the models ViT-B/16, ViT-B/32, ViT-L/16, YOLOv8n, YOLOv8s, and YOLOv8m. The results derived from this training serve as an initial standard for the following step, which involves more education on a specific instrument. The initial phase utilised a smaller dataset to assess the feasibility and effectiveness of the proposed models in achieving their specific machine learning goals. The data collected during this step will feed the second round of training and help identify any necessary adjustments to improve the model's performance.

In the second stage of training, the project will be focused on the training on the larger self-customised dataset on the local CPU with the support of GPU. In this stage, the training and also testing performance of the models will be recorded and evaluated. After that, the well-trained models will be installed on the local device to be use on the real-time inference. During the real time inference, the model performance will be recorded and evaluated. Discussion on every record result will be generated.

4.2 Result from Vision Transformer for Image Classification

As mentioned on the previous chapter, the training of ViT will be divided into two sections which one is trial training on the platform of Google Colab. Trial training is intended to minimize the amount of time invested by evaluating the training procedure, estimating the time needed, and assessing the model's performance using smaller data sets at first. After that, the second training will aim to maximize the performance and also the ability of the models by boosting the sizes of the training dataset by increase the number of classes to investigate how is the model's performance when it is trained with numerous of different classes training datasets. The second stage of training will be held with the

support of the hardware devices such as mini-PC and GPU and also external webcam that are mentioned in Chapter 3.6.

4.2.1 Results of ViT on Google Colab

The training of ViT model in Google Colab started with the training of model ViT-B /16. The training of the model took the time for around 2 hours and 15 minutes for 15 epochs and the accuracy of the trained model reached 98.8%. While the model ViT-B/32 took the time for around 1 hour to finish the model training with 15 epochs in Google Colab, which can reach the testing accuracy of 99.6%. After that, the training of ViT model is continue with the training of ViT-L/32, where the pre-trained model ViT-L/16 and ViT-H/16 has too large parameter and model size which occupy too much memory and is unable to be trained and evaluated with Google Colab. The model ViT-L/32 took half an hour to finish the training and the accuracy of the model is 98.2%.

The evaluation of various trained models revealed ViT-B/32 as the champion, achieving an impressive 99.6% testing accuracy. However, its path to further improvement seems obstructed by its reliance on exceptionally small image patches. This granular approach necessitates an exponential increase in model parameters, making further gains increasingly resource intensive. Conversely, while ViT-L/32 boasts a larger model capacity, its immense parameter count presents a different challenge. Effectively utilizing this potential hinge on access to a correspondingly vast dataset, a requirement that currently impedes its performance compared to its smaller counterpart. In essence, both models face unique roadblocks to achieving even greater accuracy: ViT-B/32 grapples with scalability due to its finely grained approach, while ViT-L/32 is bottlenecked by the need for a much larger dataset to fully leverage its parameter-rich architecture.

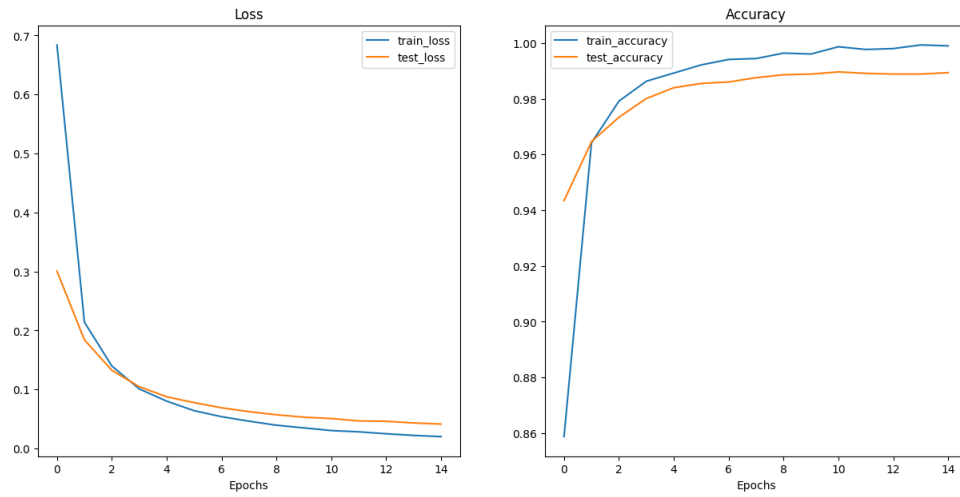


Figure 4.1. Performance of ViT-B/16

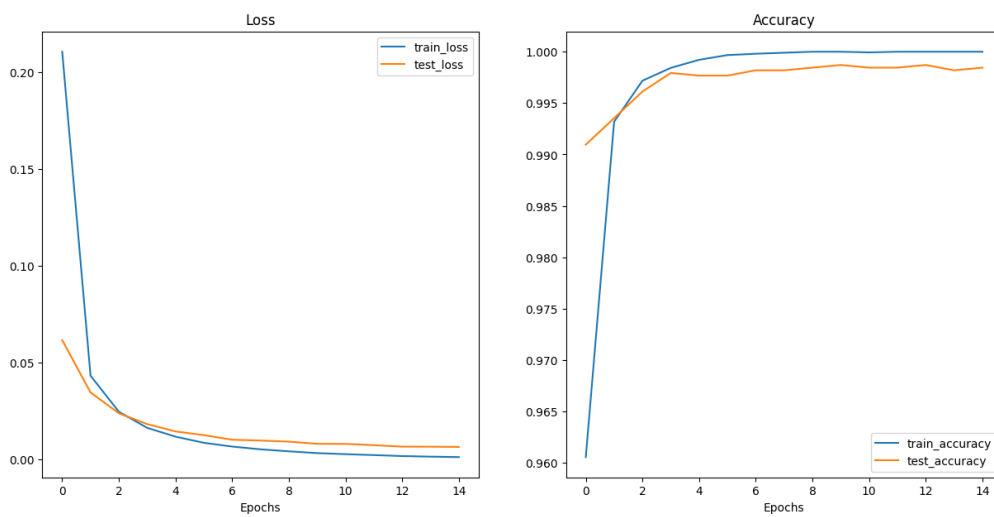


Figure 4.2. Performance of ViT-B/32

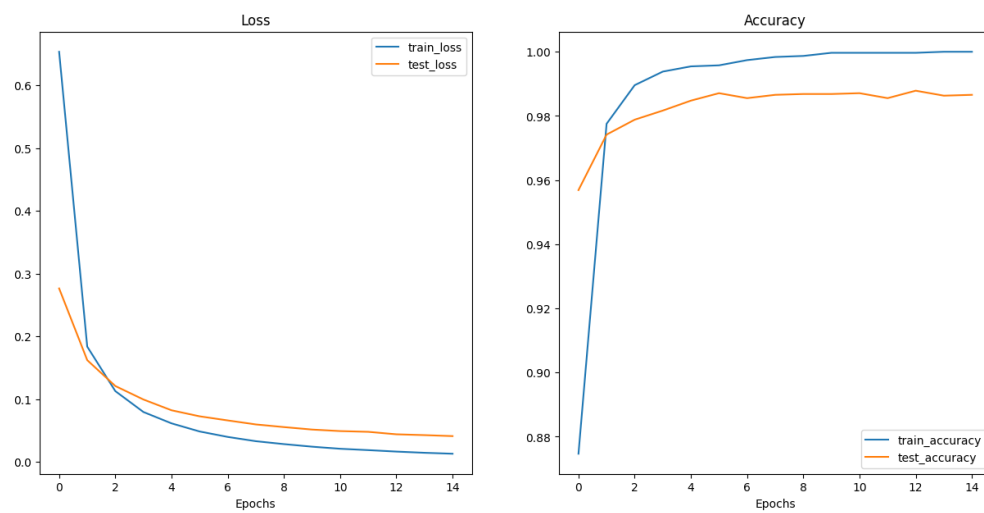


Figure 4.3. Performance of ViT-L/32

4.3 Result of YOLOv8 in Image Classification

Similar with the training process of ViT, I also conducted a trail training for YOLOv8 with two section which one is trial training on Google Colab and the second section is training on local device. With the trial training of YOLOv8 models on Google Colab, we can minimize the amount of time invested by evaluating the training procedure, estimating the time needed, and assessing the model's performance using smaller data sets at first. After that, the second training will aim to maximize the performance and also the ability of the models by boosting the sizes of the training dataset by increase the number of classes to investigate how is the model's performance when it is trained with numerous of different classes training datasets. The second training will aim to maximize the performance and also the ability of the models by boosting the sizes of the training dataset by increase the number of classes to investigate how is the model's performance when it is trained with numerous of different classes training datasets.

4.3.1 Result of YOLOv8 on Google Colab

In this section, YOLOv8n, YOLOv8s and YOLOv8m are chose during the training on Google Colab. The model size of YOLOv8 models is small, the training of the models takes lesser time if compared to ViT. For YOLOv8n, the training time taken is 25 minutes for 15 epochs. While the training of the model YOLOv8s took the time for around 38 minutes for 15 epochs training in Google Colab. After that, the training is continued with the model YOLOv8m in Google Colab which took the time for 50 minutes for 15 epochs training. The result of training and testing accuracy of YOLOv8 models are likewise if it is compared to ViT models.

The training loss, depicted in the subsequent figures, is an essential metric that provides insight into the model's performance on the training dataset. Ideally, it ought to diminish as the model acquires greater proficiency in discerning the patterns within the dataset. Validation loss, conversely, is a metric that demonstrates the model's performance on untrained data. Overfitting, which occurs when a model becomes overly specialized to the training data and performs inadequately on new data, must be avoided. Top-1 accuracy pertains to the percentage of images in which the model accurately and with the highest

degree of confidence classifies the most prominent object category. Top-5 accuracy refers to the percentage of photos in which the right object class is among the top 5 categories predicted with the highest confidence.

From the figures 4.4, 4.5, 4.6, the graphs summarized the accuracy of YOLOv8 models on training and testing dataset. The graphs shown demonstrate positive trends, with both the training and validation loss consistently decreasing during the training period. This indicates that the models' performance is improving, and it has the capacity to make accurate predictions beyond the training data. We can summarize that the performance of the YOLOv8s and YOLOv8m have better performance than YOLOv8s on the custom dataset which their top-1 accuracy is 99.0% and 99.6% respectively while YOLOv8s has slightly lower top-1 accuracy which is 98.7%. The top-5 accuracy of the models is the same which reached the accuracy of 100% after 15 epochs of training.

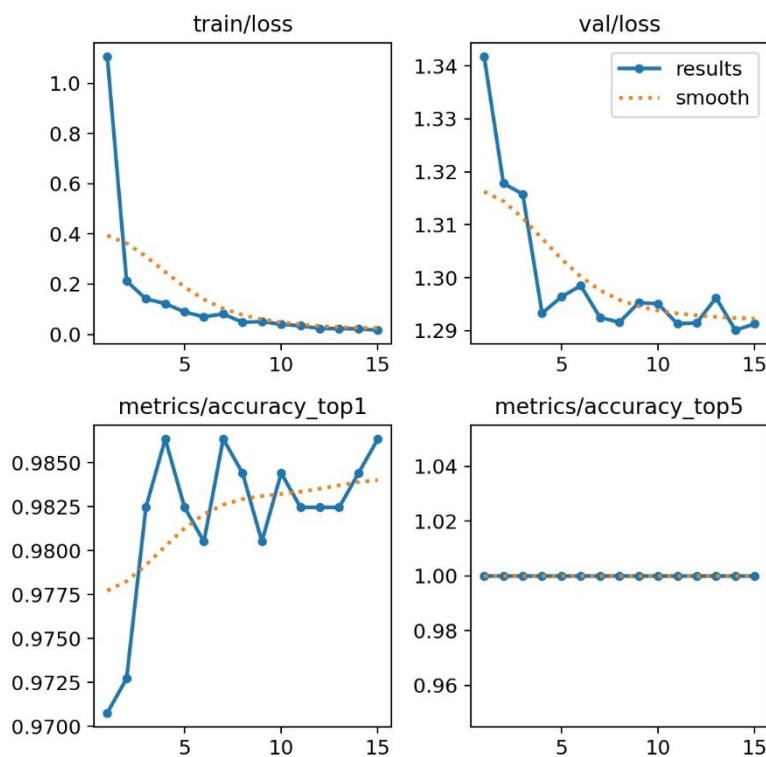


Figure 4.4. YOLOv8n Training Performance

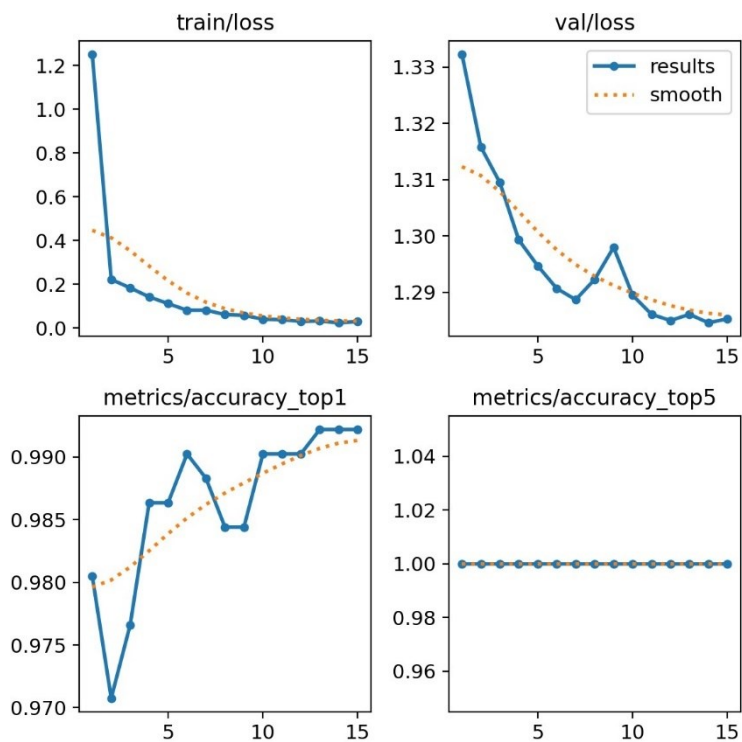


Figure 4.5. YOLOv8s Training Performance

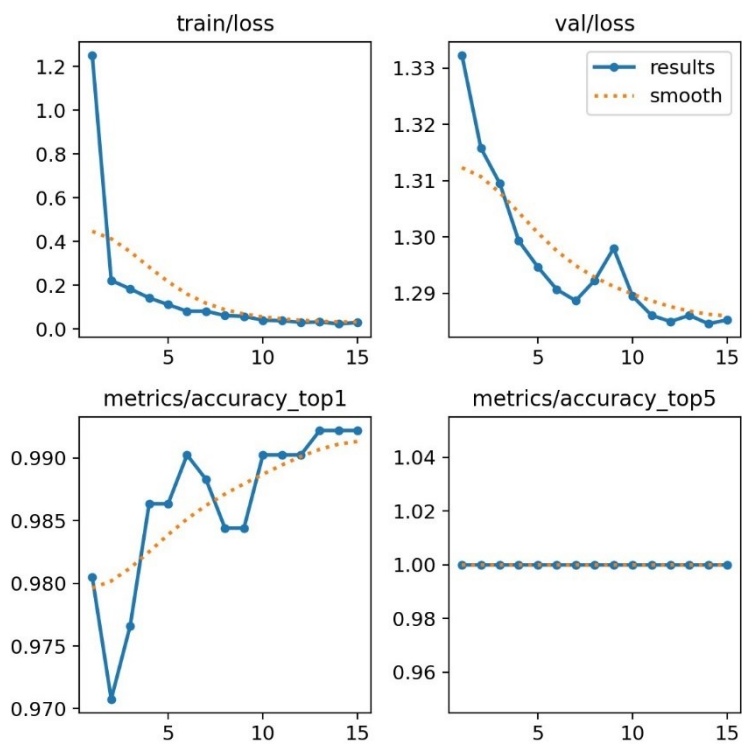


Figure 4.6. YOLOv8m Training Performance

4.4 Result for ViT and YOLOv8 on Local Hardware Device

During the second phase of model training, it is crucial to utilize both local GPU and CPU resources, as explained in Chapter 3.7 Hardware Overview. This hardware configuration allows for accelerated training of ViT and YOLOv8 models on large datasets in a considerably shorter time. Using a local GPU speeds up computational operations, making iterations faster and improving model performance. As a result, researchers and practitioners may effectively manage larger datasets and quickly make changes to enhance models, ultimately resulting in better results for different computer vision applications. The incorporation of nearby GPU resources represents a notable progress in deep learning techniques, enabling professionals to address intricate problems with enhanced efficiency and efficacy.

4.4.1 Training and Testing Performance of ViT and YOLOv8

Tables 4.1 and 4.2 below provide detailed information on the training and testing performance of the ViT and YOLOv8 designs on the local hardware infrastructure. The tables present comprehensive data on the duration of training and the accuracy of assessment. From the table, ViT-B/16 took 12 hours and 50 minutes to train on the specific dataset mentioned in Chapter 3.2, which consisted of 70 different categories. On the other hand, ViT-B/32 finished its training in a significantly shorter period of time, specifically 8 hours and 58 minutes. Notably, ViT-L/16, despite its improved capabilities, required a longer training duration of 36 hours and 18 minutes. This duration indicates the highest training time among the many study models assessed. Comprehensive performance metrics provide significant information about the computing needs and effectiveness of various models. This helps academics and practitioners make informed choices about model selection and optimization tactics.

The training durations of YOLOv8 models on the custom dataset demonstrate their superior efficiency in model completion when compared to ViT architectures. YOLOv8 models, on average, exhibited faster training times. More precisely, the training for YOLOv8n-cla was completed in 16 hours and 35 minutes, while YOLOv8m-cla ended shortly after in 17 hours and 10 minutes. The YOLOv8s-cla, the most compact version, necessitated a slightly longer training period of 17 hours and 30 minutes. The training times highlight the

efficiency of YOLOv8 architectures in handling huge datasets well, providing competitive performance while minimizing computing burden.

By evaluating the training and testing accuracies of various models, one can obtain valuable insights into their effectiveness in handling the given dataset. Among the ViT designs, ViT-L/16 achieves the best training accuracy, reaching 99.7%. Both ViT-B/16 and ViT-B/32 have a 99.0% accuracy rate. Despite this, ViT-L/16 continues to hold its dominance with an impressive accuracy rate of 93.8% in testing. On the other hand, the YOLOv8 models show comparable performance. Among them, YOLOv8s-cls achieves the highest testing accuracy of 97.6%, closely followed by YOLOv8m-cls at 97.7%. It should be noted that while YOLOv8n-cls and YOLOv8s-cls achieve somewhat lower training accuracies compared to ViT-L/16, they also achieve training accuracies of 99.7% and 99.6% respectively. This showcases their capacity to perceive intricate patterns within the dataset. Overall, whereas ViT-L/16 shows outstanding accuracy during the training process, YOLOv8s-cls has higher performance on the testing dataset. From the table 4.1, YOLOv8 models achieve better accuracy on the testing dataset which they have higher top-1 and top-5 accuracy than ViT models.

Table 4.1. Model Performance on Testing Dataset

Algorithm	Top-1 Accuracy	Top-5 Accuracy
ViT-B/16	78.21%	97.6%
ViT-B/32	81.94%	98.2%
ViT-L/16	85.5%	99.0%
YOLOv8n-cls	95.1%	99.7%
YOLOv8s-cls	95.7%	99.6%
YOLOv8m-cls	95.8%	99.6%

Table 4.2. Performance of YOLOv8 and ViT on Training and Testing

Algorithm	Train Period	Train Accuracy	Test Accuracy
ViT-B/16	12hrs 50mins	99.0%	92.0%
ViT-B/32	8hrs 58mins	99.0%	91.0%
ViT-L/16	36hrs 18mins	99.7%	93.8%
YOLOv8n-cls	16hr 35mins	99.7%	96.0%

YOLOv8s-cls	17hrs 30mins	99.6%	97.6%
YOLOv8m-cls	17hrs 10mins	99.5%	97.7%

4.4.2 Real-Time Inference of ViT and YOLOv8

The performance comparison of the YOLOv8 and ViT models during real-time inference after models training is presented in Table 4.3.2.1. After the training process, both models can be downloaded and saved locally in the '.pt' file type. After the download procedure is finished, the size of each model is shown and documented for future reference. The process of real-time inference is simple: the CPU retrieves the model by supplying the model directory. Afterwards, through OpenCV, the code establishes connectivity with the webcam in order to acquire photos. After taking a picture, the code builds a link between the taken image and the loaded model. Subsequently, the model carries out predictions on the image, and the resultant predictions are exhibited on the screen utilizing OpenCV. The process will continue frame by frame with the images captured with the webcam until the program is stopped.

As shown in Table 4.3.2.1 below, Varied levels of efficacy are observed among the models that were examined. Using the CPU, the ViT-B/16 model obtains 15 FPS inference performance, while GPU support enables 29 FPS. The model's file size is 333.91MB. In a similar fashion, the ViT-B/32 variant exhibits a processing speed of 5FPS on the CPU and 27 FPS on the GPU despite its 263.31MB smaller model size. The inference performance of the ViT-L/16 model is 2 FPS on the CPU and 30 FPS with GPU assistance, despite its larger size of 1084.58MB. Moreover, YOLOv8n-cls, YOLOv8s-cls, and YOLOv8m-cls are variants of the YOLOv8 model that are incorporated into our research. Among the various variations, YOLOv8n-cls possesses the least model size which is 3.1MB, while YOLOv8s-cls along with YOLOv8m-cls each have a larger size of 10.4MB and 31.9MB, respectively. But the average model size of YOLOv8 models still smaller than ViT modes. With combination of GPU and a CPU efficiency of 31 FPS, YOLOv8n-cls exhibits exceptional inference performance, attaining 30 FPS. Inference rates on the CPU are marginally slower for YOLOv8s-cls and YOLOv8m-cls, at 27 FPS and 19 FPS,

respectively. These models with the support of GPU still remain competitive which obtained inference speed at 29 and 32 FPS.

Table 4.3. Comparison of Model Size and Inference Speed between YOLOv8 and ViT

Algorithm	Model Size	Inference Speed (CPU)	Inference Speed (GPU)
ViT-B/16	333.91MB	15 FPS	29 FPS
ViT-B/32	263.31MB	5 FPS	27 FPS
ViT-L/16	1084.58MB	2 FPS	30 FPS
YOLOv8n-cls	3.1MB	31 FPS	30 FPS
YOLOv8s-cls	10.4MB	27 FPS	29 FPS
YOLOv8m-cls	31.9MB	19 FPS	32 FPS

The monitor screen exhibits the outcomes captured through the webcam feed, as illustrated in Figures 4.3.2.1 and 4.3.2.2. After conducting a thorough assessment of multiple YOLOv8 model variants, it has been concluded that the YOLOv8m-cls model demonstrates the highest level of performance. Because of this, this model was chosen for the purpose of inference. Significantly, the anticipated outcomes are conspicuously exhibited in the upper-left corner of every figure. The top five most confident predictions generated by the YOLOv8 models are displayed with the predicted class followed by its confidence in an up-left corner on the display from webcam. Notably, the forecasts generated by the model correspond precisely with the images that were observed. Both the healthy grape leaf in Figure 4.3.2.1 and the healthy blueberry leaf in Figure 4.3.2.2 were accurately classified by the model.

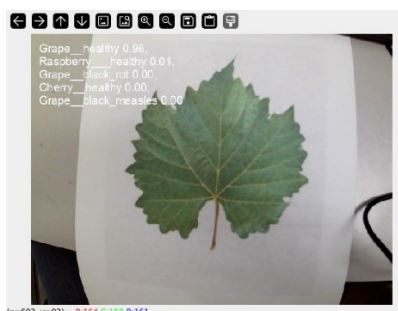


Figure 4.7. YOLOv8m-cls Inference



Figure 4.8. YOLOv8m-cls Inference

Among ViT models, ViT-L/16 achieved the best testing accuracy, so it is chosen for the inference on the webcam. From the figure 4.3.2.3 and figure 4.3.2.4 below is the results from the ViT-L/16 on the real time inference. However, the model has bad performance on the inference speed with CPU, which is only 2FPS and this is impossible for the model to run real time inference test and show the result smoothly. Therefore, the inference of this model is supported with GPU. From the figures below, the model exhibited remarkable prognostic capabilities by precisely distinguishing between a healthy blueberry leaf and a leaf affected by apple rot.

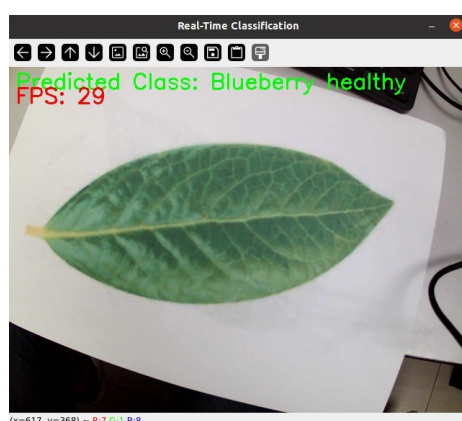


Figure 4.9. ViT-L/16 inference



Figure 4.10. ViT-L/16 inference

4.5 Limitations and Troubleshooting

During the research, there are several limitations and the drawbacks that is observed throughout the research. To maximize the performance and the results of the proposed research, several solutions and enhancements have been proposed to solve the drawbacks and problems during the research. Recognizing the importance of maximizing the performance of the proposed ML models, I devised a comprehensive set of solutions aimed to mitigate the identified limitations. These solutions encompassed a spectrum of strategies, including algorithms refinements and hardware enhancements.

From the result that is obtained from this chapter, there are several limitations and drawbacks during the development and testing of the models. For example, there is always false detection during the real-time inference. False detection is when the ML models do the wrong prediction. This seldom happen

when the models are used to predict an image, but this is a major problem when come to the real time inference on the webcam.

One of the factors that causing the false detection is lack of training. When come to the real time webcam inference, there is too many uncertainties that will affect the model performance for example the hardware which will affect the quality of the input images and the smoothness of the prediction. For ViT models, they have one common weakness which they require larger computing resources to run prediction on real-time webcam, which means it is almost impossible to be run under CPU without support of GPU. This make YOLOv8 models better which it can be run with the CPU with good performance. In the other words, ViT has larger demand on the computing resource compared to YOLOv8.

Rather than that, YOLOv8 and ViT models required large training dataset to achieve better performance, and the quality of the training dataset and also variety of the fed data is also determine factor for the model performance. This makes the data collection and sorting process become a very time and resources consuming process. Although the process of a ML development is not simple, but they still can bring valuable benefits and convenience to human life.

4.6 Summary

In short, from this chapter results prove that the YLOv8 achieved better performance if compared with ViT. YOLOv8 models, specifically YOLOv8m has better testing accuracy compared to ViT models, achieving testing accuracy of 97.7%. Moreover, YOLOv8m model also exhibit better performance when the model is deployed for the real time inference, maintaining its speed even when deployed on a CPU.

The biggest reason that causing the performance difference between the models is model size and the model complexity. From the result, we can conclude that the model complexity is directly proportional to the model size. Which means that with a larger model size, the model structure is more complex. This will affect the calculation time during the real time prediction which will cause lagging and low FPS during the inference. Besides that, the complexity of the high complexity of model will cause the model to be more sensitivity to the hyperparameter setup and also quality of the training source. This is because

the complex models often have larger number of hyperparameters and slightly changes and modify may cause different outcome. Thus, this may be the reason of the performance of ViT models is slightly lesser than YOLOv8.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

In this study, the objectives have been achieved. The comparison between YOLOv8 models with ViT models is ended with YOLOv8m-cls achieved the best performance among all of the selected models. All of the developed models are able to complete the task of plant disease recognition but different performance. Lastly, the YOLOv8 models all can run smoothly for the real-time inference with the webcam on CPU which achieved the objective of the project.

The results from the Chapter 4 proved that YOLOv8 models and ViT models has similar training performance which is around 99% which their accuracy difference can be neglected. This proves the high potential of the ML models in learning process and can learning the information from image data clearly. Rather than that, the models also achieved high testing accuracy which is above 90%. This proved that the chose model is well-trained and has the ability to make the predictions on the testing dataset accurately. Among the models, YOLOv8m-cls achieved the highest testing accuracy. In the second part of the evaluation, YOLOv8m-cls showed its lightweight characteristics and can be run real-time inference with webcam smoothly with satisfying performance under CPU if compared to other models.

Although YOLOv8 models shows its superior in the proposed task with limited computing and graphic resource, but the performance of ViT still showing the model has great potential in image classification and recognition. ViT is a transformer-based model and is modified from the NLP task ML model which make it more complex than most of the CNN models include YOLOv8. Therefore, ViT model often required more computing power to support the prediction which can be clearly proved as the ViT can run smoothly with the support of GPU with 28FPS and can has similar performance with YOLOv8 in real-time inference.

In conclusion, this study demonstrates that YOLOv8 models exhibit superior performance compared to alternative models in terms of precision,

learning velocity, and instantaneous deduction, specifically under conditions of limited computational resources. Despite exhibiting promise, ViT models require substantial computational resources and meticulously structured datasets to attain comparable performance. By capitalising on hardware advancements and increasing data accessibility, ViT models may potentially enhance their competitiveness. At present, YOLOv8 models represent the most optimal choice for real-world applications requiring efficient and rapid image classification.

5.2 Recommendations for future work

Application of ML and AI into the agricultural is still long to go, but from this project, we can see the progress of the implementation of AI into human's life. Nevertheless, there are several recommendations for the future work and improvement. Firstly, from the research, the most important factors throughout the development are the quality of training dataset. For ML model, training dataset is their learning material, therefore it is important to improve the quality and also quantity of dataset. For example, the future work for this project is to include wider variety of plant leaf images under different environment to help increase the diversity of the images for the models to have better learning and focus on leaf and other important criteria in the image.

Besides that, to improve the performance of ViT on CPU which to make the model runs smoothly under CPU, the proposed future modification is trying to build the model from scratch. In this project, the model taken for both ViT and YOLOv8 is pretrained with ImageNet, therefore the model will be more complex and heavier. Therefore, to help improve the prediction speed during the real-time inference, one of the methods is building the model from scratch and let it learns on the prepared dataset to reduce the unusable parameters inside the models.

Lastly, during the real-time inference the results shows lots of false detection and this situation is very resource inefficient. The recommendations on this is to try apply more filter on the system to filter the useless information from environment and try to fix the webcam on a more stable fixture so that the information that fed to the model during the real-time inference.

REFERENCES

ACMC, E. S. C. (2023, January 7). Countries with the highest agricultural output 2023: Top 12. Bschorly. Retrieved from: <https://bscholarly.com/countries-with-the-highest-agricultural-output/>.

Arvindpdmn. (2021, April 7). ImageNet. Devopedia. Retrieved from: <https://devopedia.org/imagenet>.

Boesch, G. (2022, January 17). A guide to data collection for Computer Vision in 2022. viso.ai. Retrieved from: <https://viso.ai/computer-vision/data-collection/>.

Chauhan, S. (2023, June 1). Model selection for Machine Learning. Live Training, Prepare for Interviews, and Get Hired. Retrieved from: <https://www.scholarhat.com/tutorial/machinelearning/model-selection-for-machinelearning#:~:text=Model%20selection%20in%20machine%20learning%20is%20the%20process%20of%20selecting,data%20%26%20produces%20the%20best%20results>.

Food and Agriculture Organization of the United Nations. (2020). STATISTICAL YEARBOOK. STATISTICAL YEARBOOK WORLD FOOD AND AGRICULTURE 2020. Retrieved from: <https://www.fao.org/3/cb1329en/online/cb1329en.html#>.

Machine Learning in computer vision. Full Scale. (2019, May 8). Retrieved from: <https://fullscale.io/blog/machine-learning-computer-vision/>.

Papers With Code. (2023). Image Classification on ImageNet. The latest in Machine Learning. Retrieved from: <https://paperswithcode.com/sota/image-classification-on-imagenet>.

Saha, S. (2018, December 15). A comprehensive guide to Convolutional Neural Networks - the eli5 way. Saturn Cloud Blog. Retrieved from: <https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/>.

Delorme, pierre joseph. (2021, May 18). *Image preprocessing*. Medium. Retrieved from: <https://medium.com/unpackai/image-preprocessing-6654d1bb4daa>.

Peshawa J. Muhammad Ali, Rezhna H. Faraj; “Data Normalization and Standardization: A Technical Report”, Machine Learning Technical Reports, 2014, 1(1), pp 1-6.

https://docs.google.com/document/d/1x0A1nUz1WWtMCZb5oVzF0SVMY7a_58KQulqQVT8LaVA/edit#.

Babu, R. (2019, February). *Plant disease identification and classification using image processing*. Plant Disease Identification and Classification using Image

Processing.

https://www.researchgate.net/publication/337023535_Plant_Disease_Identification_and_Classification_using_Image_Processing

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021, June 3). *An image is worth 16x16 words: Transformers for image recognition at scale.* arXiv.org. <https://doi.org/10.48550/arXiv.2010.11929>

Guo, Y., Lan, Y., & Chen, X. (2022). CST: Convolutional Swin Transformer for detecting the degree and types of plant diseases. *Computers and Electronics in Agriculture*, 202, 107407. <https://doi.org/10.1016/j.compag.2022.107407>

Jagan, K., Balasubramanian, M., & Palanivel, S. (2016). Detection and recognition of diseases from Paddy Plant Leaf Images. *International Journal of Computer Applications*, 144(12), 34–41. <https://doi.org/10.5120/ijca2016910505>

Oliver, N., Rosario, B., & Pentland, A. (2000). A Bayesian computer vision system for modeling human interactions. *Lecture Notes in Computer Science*, 255–272. https://doi.org/10.1007/3-540-49256-9_16

O'Shea, K., & Nash, R. (2015, December 2). *An introduction to Convolutional Neural Networks.* arXiv.org. <https://doi.org/10.48550/arXiv.1511.08458>

Saxena, O., Agrawal, S., & Silakari, S. (2021). Disease detection in plant leaves using deep learning models: Alexnet and googlenet. *2021 IEEE International Conference on Technology, Research, and Innovation for Betterment of Society (TRIBES)*. <https://doi.org/10.1109/tribes52498.2021.9751620>

Susa, J. A., Nombrefia, W. C., Abustan, A. S., Macalisang, J., & Maaliw, R. R. (2022). Deep learning technique detection for cotton and leaf classification using the Yolo algorithm. *2022 International Conference on Smart Information Systems and Technologies (SIST)*. <https://doi.org/10.1109/sist54437.2022.9945757>

T, N., Vijayalakshmi, P., Jaya, J., & S, S. (2022). A review on coconut tree and plant disease detection using various deep learning and convolutional neural network models. *2022 International Conference on Smart and Sustainable Technologies in Energy and Power Sectors (SSTEPS)*. <https://doi.org/10.1109/ssteps57475.2022.00042>

Zaki, S. Z., Asyraf Zulkifley, M., Mohd Stofa, M., Kamari, N. A., & Ayuni Mohamed, N. (2020). Classification of tomato leaf diseases using MobileNet V2. *IAES International Journal of Artificial Intelligence (IJ-AI)*, 9(2), 290. <https://doi.org/10.11591/ijai.v9.i2.pp290-296>

Batta, M. (2019, January). *Abstract of machine learning ALG, IJSR, call for papers, online journal.* International Journal of Science and Research (IJSR). <http://dx.doi.org/10.21275/ART20203995>

- Diwan, T., Anirudh, G., & Tembhrne, J. V. (2022). Object detection using yolo: Challenges, architectural successors, datasets and applications. *Multimedia Tools and Applications*, 82(6), 9243–9275. <https://doi.org/10.1007/s11042-022-13644-y>
- Ishak, S., Fazalul Rahiman, M. H., Mohd Kanafiah, S. N., & Saad, H. (2015). Leaf disease classification using Artificial Neural Network. *Jurnal Teknologi*, 77(17). <https://doi.org/10.11113/jt.v77.6463>
- Kumari, Ch. U., Jeevan Prasad, S., & Mounika, G. (2019). Leaf disease detection: Feature extraction with K-means clustering and classification with ann. *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*. <https://doi.org/10.1109/iccmc.2019.8819750>
- Kusumo, B. S., Heryana, A., Mahendra, O., & Pardede, H. F. (2018). Machine learning-based for automatic detection of corn-plant diseases using image processing. *2018 International Conference on Computer, Control, Informatics and Its Applications (IC3INA)*. <https://doi.org/10.1109/ic3ina.2018.8629507>
- Nguyen, Q. H., Ly, H.-B., Ho, L. S., Al-Ansari, N., Le, H. V., Tran, V. Q., Prakash, I., & Pham, B. T. (2021). Influence of data splitting on performance of machine learning models in prediction of shear strength of Soil. *Mathematical Problems in Engineering*, 2021, 1–15. <https://doi.org/10.1155/2021/4832864>
- Wei, J., Chu, X., Sun, X., Xu, K., Deng, H., Chen, J., Wei, Z., & Lei, M. (2019). Machine learning in materials science. *InfoMat*, 1(3), 338–358. <https://doi.org/10.1002/inf2.12028>
- Yu, S., Xie, L., & Huang, Q. (2023). Inception Convolutional Vision Transformers for Plant Disease Identification. *Internet of Things*, 21, 100650. <https://doi.org/10.1016/j.iot.2022.100650>
- Araujo, V., Britto, A. S., Brun, A. L., Koerich, A. L., & Palate, R. (2017). Multiple classifier system for plant leaf recognition. *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. <https://doi.org/10.1109/smc.2017.8122891>
- Boukabouya, R. A., Moussaoui, A., & Berrimi, M. (2022). Vision transformer-based models for plant disease detection and diagnosis. *2022 5th International Symposium on Informatics and Its Applications (ISIA)*. <https://doi.org/10.1109/isia55826.2022.9993508>
- Dai, G., & Fan, J. (2022). An industrial-grade solution for crop disease image detection tasks. *Frontiers in Plant Science*, 13. <https://doi.org/10.3389/fpls.2022.921057>
- K R, C. L., B, P., G, S., J, N. J., T, G., & Hashim, M. (2023). Yolo for detecting plant diseases. *2023 Third International Conference on Artificial Intelligence and Smart Energy (ICAIS)*. <https://doi.org/10.1109/icaais56108.2023.10073875>

Menon, V., Ashwin, V., & Deepa, R. K. (2021). Plant disease detection using CNN and transfer learning. *2021 International Conference on Communication, Control and Information Sciences (ICCISc)*. <https://doi.org/10.1109/iccisc52257.2021.9484957>

Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Using deep learning for image-based plant disease detection. *Frontiers in Plant Science*, *7*. <https://doi.org/10.3389/fpls.2016.01419>

Pruneski, J. A., Williams, R. J., Nwachukwu, B. U., Ramkumar, P. N., Kiapour, A. M., Martin, R. K., Karlsson, J., & Pareek, A. (2022). The development and deployment of Machine Learning Models. *Knee Surgery, Sports Traumatology, Arthroscopy*, *30*(12), 3917–3923. <https://doi.org/10.1007/s00167-022-07155-4>

Shah, R. S. (2010). *Support Vector Machines for classification and regression*. Library and Archives Canada = Bibliothèque et Archives Canada.

Shill, A., & Rahman, Md. A. (2021). Plant disease detection based on Yolov3 and Yolov4. *2021 International Conference on Automation, Control and Mechatronics for Industry 4.0 (ACMI)*. <https://doi.org/10.1109/acmi53878.2021.9528179>

Soeb, Md. J., Jubayer, Md. F., Tarin, T. A., Al Mamun, M. R., Ruhad, F. M., Parven, A., Mubarak, N. M., Karri, S. L., & Meftaul, I. Md. (2023). Tea leaf disease detection and identification based on Yolov7 (YOLO-T). *Scientific Reports*, *13*(1). <https://doi.org/10.1038/s41598-023-33270-4>

Thakur, P. S., Khanna, P., Sheorey, T., & Ojha, A. (2022, July 16). *Explainable vision transformer enabled convolutional neural network for Plant Disease Identification: Plantxvit*. arXiv.org. <https://doi.org/10.48550/arXiv.2207.07919>

Japkowicz, N. (2006). Why Question Machine Learning Evaluation Methods? <https://cdn.aaai.org/Workshops/2006/WS-06-06/WS06-06-003.pdf>