الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي و البحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Département d'Électronique

Thesis of Final Graduation Project

For the obtention of the State Engineer Diploma in Electronics

Plant Leaves Disease Severity Estimation

Wissam ABID

Under the supervision of **Mme. Nesrine BOUADJENEK** MCB ENP

Publicly presented and defended on the 24th of June, 2025.

Jury members:

President	Mr. Adel	BELOUCHRANI	Prof.	ENP
Promoter	Mme. Nesrine	BOUADJENEK	MCB	ENP
Examiner	Mr. Mourad	ADNANE	Prof.	ENP

ENP 2025

الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي و البحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Département d'Électronique

Thesis of Final Graduation Project

For the obtention of the State Engineer Diploma in Electronics

Plant Leaves Disease Severity Estimation

Wissam ABID

Under the supervision of **Mme. Nesrine BOUADJENEK** MCB ENP

Publicly presented and defended on the 24th of June, 2025.

Jury members:

President	Mr. Adel	BELOUCHRANI	Prof.	ENP
Promoter	Mme. Nesrine	BOUADJENEK	MCB	ENP
Examiner	Mr. Mourad	ADNANE	Prof.	ENP

ENP 2025

الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي و البحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

Département d'Électronique

Mémoire de projet de fin d'étude

pour l'obtention du diplôme d'Ingénieur d'État en Électronique

Estimation de la sévérité des maladies des feuilles des plantes

Wissam ABID

Sous la supervision de **Mme. Nesrine BOUADJENEK** MCB ENP

Présenté et soutenue publiquement le 24 Juin 2025.

Membres de jury:

Président	Mr. Adel	BELOUCHRANI	Prof.	ENP
Promotrice	Mme. Nesrine	BOUADJENEK	MCB	ENP
Examinateur	Mr. Mourad	ADNANE	Prof.	ENP

ENP 2025

الملخص

تهدف الزراعة الذكية إلى تحسين مراقبة المحاصيل من خلال التحليل التلقائي والدقيق لصحة النبات. وتُعد تقدير شدة المرض مهمةً أساسية في هذا المجال، حيث تركز على تحديد مراحل تطور إصابات النبات. في هذا العمل، نقترح حلاً قائمًا على التعلم العميق باستخدام اثنتين من أحدث معماريات المحولات : Vision Transformer (ViT) و Swin Transformerتم تنفيذ هذه النماذج وتقييمها ودمجها في معمارية جديدة تستفيد من الانتباه العام لـ ViT والانتباه المحلي الهرمي لـ Swin لتصنيف دقيق لمستوى الشدة. تم تدريب النماذج على مجموعة بيانات Yellow Rust التي تتضمن ست مراحل من الشدة تتراوح من السليم إلى الحساس تمامًا. تم تقييم الأداء باستخدام مقاييس تصنيف معيارية، وأظهرت النتائج أن النموذج المدمج يتفوق على النماذج الفردية، مما يوفر حلاً فعالًا للتقدير الآلي لمراحل الشدة في الزراعة الدقيقة.

الكلمات المفتاحية: الزراعة الذكية، تقدير شدة المرض، آلية الانتباه، مشفر المحول، Vision Transformer (ViT)، Swin Transformer، انتباه متعدد الرؤوس، استخراج الميزات، الدمج.

Résumé

L'agriculture intelligente vise à améliorer la surveillance des cultures grâce à une analyse automatisée et précise de la santé des plantes. Une tâche cruciale dans ce domaine est l'estimation de la sévérité des maladies, qui consiste à identifier les stades de progression des infections des plantes. Dans ce travail, nous proposons une solution basée sur lapprentissage profond utilisant deux architectures de transformers : Vision Transformer (ViT) et Swin Transformer. Ces modèles sont implémentés, évalués, puis combinés dans une architecture innovante qui exploite lattention globale du ViT et l'attention locale hiérarchique du Swin pour une classification fine de la sévérité. Les modèles sont entraînés sur le jeu de données Rouille Jaune du blé, comprenant six stades de sévérité. Enfin, les résultats montrent que le modèle combiné surpasse les modèles individuels de référence, offrant une solution efficace pour l'estimation automatisée de la sévérité.

Mots-clés : Agriculture Intelligente, Estimation de la Sévérité des Maladies, Rouille Jaune, Mécanisme d'Attention, Encodeur Transformer, Vision Transformer (ViT), Swin Transformer, Auto-Attention Multi-tête, Extraction de Caractéristiques, Concaténation.

Abstract

Smart agriculture aims to improve crop monitoring through automated and accurate analysis of plant health. A critical task in this domain is disease severity estimation, which focuses on identifying the progression stages of plant infections. In this work, we propose a deep learning-based solution using two transformer architectures: Vision Transformer (ViT) and Swin Transformer. These models are implemented, evaluated, and combined into a novel architecture that leverages ViTs global attention and Swins hierarchical local attention for fine-grained severity classification. The models are trained on Wheat Yellow Rust dataset, which includes six severity stages. Finally, results show that the combined model outperforms individual baselines, providing an effective solution for automated severity estimation.

Keywords : Smart Agriculture, Disease Severity Estimation, Yellow Rust, Attention Mechanism, Transformer Encoder, Vision Transformer (ViT), Swin Transformer, Multi-head Self-Attention, Feature Extraction, Concatenation.

Acknowledgments

First and foremost, I would like to express my sincere gratitude to Dr. Nesrine BOUADJENEK, my supervisor, for her invaluable guidance, continuous support, and insightful advice throughout the course of this research. Her expertise and encouragement have been instrumental in shaping this thesis.

I extend my deepest appreciation to the members of the jury, particularly Pr. BELOUCHRANI Adel, the president of the jury, and Pr. ADNANE Mourad, the examiner, for accepting to evaluate my work and for their valuable feedback and constructive criticism. Their time and effort are greatly appreciated.

I would like to acknowledge all the teachers and students of the Electronics Engineering Department, whose contributions, discussions, and collaborative spirit have enriched my academic journey. Their support and engagement have been a source of motivation throughout this process.

A special mention goes to the ELN3 class, whose encouragement and unwavering support have made this journey more meaningful. Their presence has been a source of inspiration, and I am grateful for the kindness and positivity they have shared.

Dedication

"To my parents, who believed in me when I didn't believe in myself. Your sacrifices and endless love gave me the strength to push through every challenge. Your endless support and encouragement have been the foundation upon which this achievement stands.

To my best friend, who has been there through it all, my loyal companion and greatest support.

To my family, who celebrated every milestone and offered strength during challenging times.

To ELN department, who were there through all the ups and downs and shared in both the struggles and challenges.

To the VIC family and all its members, and especially the HR department and the committee I worked with, you made me believe that anything is possible when we support each other.

To all those who have helped in one way or another during this research, I extend my heartfelt thanks.

Thank you all."

Contents

List of Figures

List of Tables

List of Abbreviations

CNNs Conolutional Neural Networks

AI Artificial Intelligence

ML Machine Learning

NLP Natural Language Processing

RMSE Root Mean Squared Error

ViT Vision Transformer

CLS-token Classification token

MLP Multi Layer Perceptron

FFN Feed Forward Network

MHSA Multi Head Self Attention

LN Layer Normalization

Swin Shifted window

Swin ViT Shifted Window Vision Transformer

W-MSA Window-based Multi-head Self-attention

SW-MSA shifted Win- dow Multi-head Self-attention

GELU Gaussian Error Linear Unit

GAP Global Average Pooling

ReLU Rectified Linear Unit

R Resistant

MR Moderate Resistant

MRMS Moderate Resistant Moderate Susceptible

MS Moderate Susceptible

S Susceptible

General Introduction

Agriculture has always been the main source of livelihood for humanity. The agriculture sector is undergoing a major transformation with industry 4.0 and big data technologies [1].

In last decades, agriculture has witnessed a transformative shift with the integration of advanced technologies to enhance crop yield and ensure food security. One crucial aspect of this transformation is the utilization of computer vision with advanced machine learning techniques for the early classification and identification of plant diseases [2]. Moreover, plant disease severity estimation is a new challenging research issue in precision agriculture which helps to make effective disease management [3]. Better methods for detection and further increased accuracy in the estimation of the severity of the disease are demanded [4].

This work addresses the critical challenge of automated plant disease severity assessment in smart agriculture by developing a deep learning framework that combines two complementary transformer architectures. The proposed solution integrates Vision Transformer (ViT), which captures global image relationships through comprehensive attention mechanisms, with Swin Transformer, which excels at hierarchical local feature extraction. By merging these approaches, the hybrid model effectively balances global context awareness with detailed local pattern recognition, enabling precise classification of disease progression across six severity stages (Healthy, Resistant, Moderate Resistant, Moderate Resistant Moderate Susceptible, Moderate Susceptible, Susceptible). When evaluated on the Wheat Yellow Rust dataset, this combined architecture demonstrates superior performance compared to individual transformer models, offering a robust and automated approach for real-time crop health monitoring that can significantly enhance agricultural decision-making and early intervention strategies.

This thesis is divided into five chapters, structured as follows:

The first chapter introduces the plant leaf disease severity estimation system, outlining its key steps and reviewing the most recent work carried out in this field. This provides the foundational context for the study.

The second chapter introduces attention mechanisms and transformer models, which have revolutionized various fields, especially computer vision. We will explore the fundamental concepts of attention, followed by a comprehensive explanation of Vision Transformers ViT and its variants as well as the Swin Transformer and its variants. The chapter also discusses the motivation behind selecting these models for our study and highlights the benefits of combining different transformer architectures.

The third chapter presents the models used to estimate disease severity in plant leaves. These models include transformer-based architectures which are Vision Transformer (ViT) and Swin Transformer, each designed to extract different aspects of image features. The chapter focuses on their structure and performance independently.

The fourth chapter details the experimental setup used to evaluate the proposed combined model for severity estimation. It covers software and hardware environments, dataset specifics, preprocessing steps, evaluation metrics, and implementation details.

The fifth chapter presents the results obtained using the proposed models. We analyze their performance in detail and evaluate the effectiveness of our approach in addressing the problem. Based on the insights gained, we propose a novel combined model that integrates ViT and Swin Transformer architectures to leverage their complementary strengths. The features extracted from both branches are concatenated and passed through a classification head to predict severity levels between multiple class of diseases. This fusion aims to improve accuracy by effectively capturing both global and local features.

Finally, we will conclude by summarizing the findings, discussing the limitations of the proposed approach, and suggesting potential directions for future research to enhance the system's performance.

Chapter 1

State of the art on plant leaves disease severity estimation

1.1 Introduction

Plant diseases pose a significant threat to global agricultural productivity and food security, accounting for losses of 10 to 30% of the global harvest each year [5]. They manifest through diverse symptoms that can lead to reduced yields and compromised plant health. These diseases, caused by fungi, bacteria, and viruses, exhibit distinct pathological characteristics, making accurate identification and management essential for effective disease control.

This chapter presents a comprehensive overview of current knowledge and methodologies used to estimate the severity of plant leaf diseases. Covers the main categories of plant diseases, their symptoms, traditional and modern severity assessment techniques. Finnaly, we present state of the art advances in artificial intelligence for plant disease severity estimation.

1.2 Major Categories of plant leaf diseases

Plant leaf diseases can be classified based on their causal agents into two categories : [6]

- **Infectious diseases :** The infectious agents are called pathogens and can be grouped as follows: viruses, bacteria, fungi, nematodes, and parasitic seed plants. These pathogens can spread rapidly and severely affect plant health.
- **Non-infectious diseases :** The non-infectious agents are caused directly or indirectly by inappropriate physical, chemical or other abiotic environmental factors [7].

Infectious diseases are the most damaging and are responsible for the greatest losses in crop production worldwide.

1.2.1 Fungal Diseases

Fungal Diseases are an infectious disease. Pathogenic fungi can lead to significant ecological changes. It causes up to 30% of crop diseases. Notable outbreaks include Cryphonectria parasitica on chestnut trees, causing nearly total defoliation, and Phytophthora cinnamomi, which leads to root rot in many crops. These fungi can affect multiple plant species and involve other organisms like beetles and birds [8].

Figure 1.1: Black spot [9].

Figure 1.2: Yellow rust [9].

As shown in Figure 1.1 and Figure 1.2, both black spot and rust are fungal diseases that affect plant leaves, black spot causes dark lesions on the upper leaf surface, while rust forms rust-colored pustules underneath. These fungi thrive in moist conditions and spread through water, wind, and insects, leading to leaf discoloration and premature drop [9].

1.2.2 Bacterial Diseases

Plant diseases caused by bacterial pathogens are an infectious diseases that place major constraints on crop production and cause significant annual losses on a global scale. Bacterial leaf diseases are hard to control in agriculture. Some bacteria grow fast in good conditions, like Erwinia amylovora in fire blight. Others live on healthy-looking leaves or in the soil, like Pseudomonas syringae and Ralstonia solanacearum. Some hide inside plants where sprays cant reach, like Candidatus Liberibacter spp. in citrus. Insects can also spread these diseases. Bacteria can survive on old plants or tools and even on seeds without showing signs. To manage them, farmers need to use resistant plants, good timing, and a mix of treatments and farming practices [10].

Figure 1.3: Pepper bacterial diseases [11].

Pepper plants are also susceptible to bacterial diseases, which typically cause water-soaked spots that later turn dark and necrotic. As shown in Figure 1.3, these symptoms can lead to reduced photosynthesis and premature leaf drop, significantly affecting crop yield.

1.2.3 Viral Diseases

Viral plant diseases are caused by viruses and viroidstiny infectious agents that severely affect plant health. Once infected, plants are difficult to cure, making these diseases a major threat to global crop production. They spread through contact with infected plants, soil, seeds, pollen, insects, or vegetative reproduction. Common symptoms include malformations, necrosis, dwarfism, and discoloration. Notable viral diseases include tobacco mosaic, tomato spotted wilt, potato spindle tuber, cucumber mosaic, barley yellow dwarf, prunus necrotic ring spot, and citrus exocortisall of which significantly affect plant growth and crop yields [12].

Tomato plants are vulnerable to viral infections such as the mosaic virus, which causes mottled, light and dark green patterns on the leaves, often leading to leaf distortion and reduced growth. As illustrated in Figure 1.4, these symptoms are characteristic of

the tomato mosaic virus, which spreads through infected tools, hands, or plant-to-plant contact.

Figure 1.4: Tomato mozaic viral disease [11].

1.3 Plant disease severity

Plant disease severity refers to the quantitative measure of the extent of visible symptoms on plant tissue, typically expressed as the proportion or percentage of affected tissue. It is a crucial variable for understanding the impact of plant diseases and is widely used in research, disease management, and decision-making processes [17].

This metric holds significant practical importance for agricultural management:

- **Treatment Guidance:** Severity levels directly determine the type, timing, and dosage of control measures (e.g., fungicides, pesticides). Accurate assessment prevents under-application (risking crop loss) and over-application (increasing costs and environmental impact).
- **Yield Prediction:** Higher severity correlates strongly with reduced photosynthetic area, impaired plant growth, and diminished harvestable yield.
- **Epidemic Monitoring:** Tracking severity over time is vital for understanding disease progression, evaluating cultivar resistance, and assessing management strategy efficacy.

1.4 Severity assessement methods

Accurate visual assessment of plant disease severity plays an important role in agricultural production. It enables the evaluation of treatment efficacy, supports yield loss estimation, and helps monitor plant development. However, traditional methods are often affected by subjectivity and variability between raters. Disease severity is typically assessed using various types of visual scales, which can be broadly categorized into qualitative and quantitative methods.

Visual severity assessment typically uses four types of scales: nominal (descriptive), ordinal, interval (category), and ratio scales, each offering varying levels of precision and objectivity [18].

1.4.1 Qualitative Scales

- **Descriptive Scale:** A simple and subjective method that categorizes disease severity using terms like *healthy stage*, *early stage*, *middle stage*, and *end stage*, as shown in Figure 1.5. Its lack of quantitative definitions limits its accuracy [18].

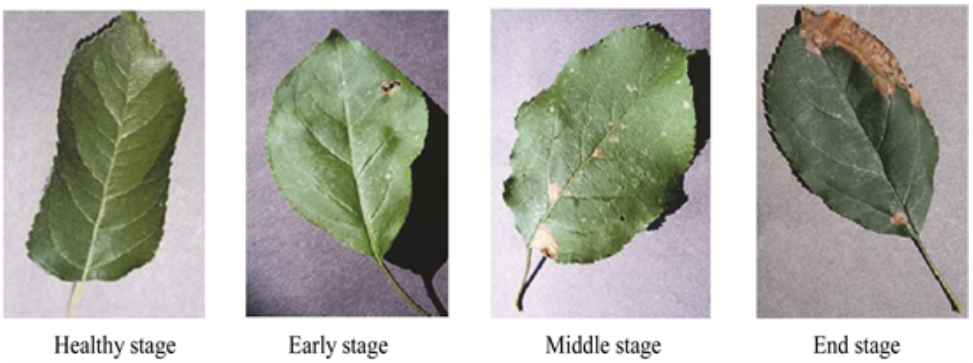

Figure 1.5: Leaf images of four severity stages (healthy, early, middle and end) of apple black rot disease [19].

- **Qualitative Ordinal Scale:** An enhanced version of the descriptive scale that uses ordered numerical values (e.g., from 0 to 5) to represent increasing severity levels. It is commonly used for diseases, especially viral ones, where symptoms are difficult to quantify precisely [18].

1.4.2 Quantitative Scales

Quantitative scales assign numerical values to assess disease severity more objectively.

- **Quantitative Ordinal Scale:** This scale categorizes symptoms by percentage ranges of affected areas. It includes equal interval scales, which may overestimate low severities, and unequal interval scales. [18].

  Table 1.1 presents the disease severity grading standard based on the proportion of disease spots observed on the leaf surface. This grading system ranges from Level 0, indicating a healthy leaf with no visible symptoms, to Level 5, where more than 50% of the leaf area is affected. As illustrated in Figure 1.6, each level corresponds to a distinct visual pattern of disease spread, which serves as a reference for consistent labeling during dataset annotation and model evaluation.

Table 1.1: Disease severity grading standard [21].

Disease Grade	Proportion of Disease Spots
Level 0	$p = 0\%$
Level 1	$0 < p \leq 5\%$
Level 2	$5\% < p \leq 10\%$
Level 3	$10\% < p \leq 25\%$
Level 4	$25\% < p \leq 50\%$
Level 5	$50\% < p \leq 100\%$

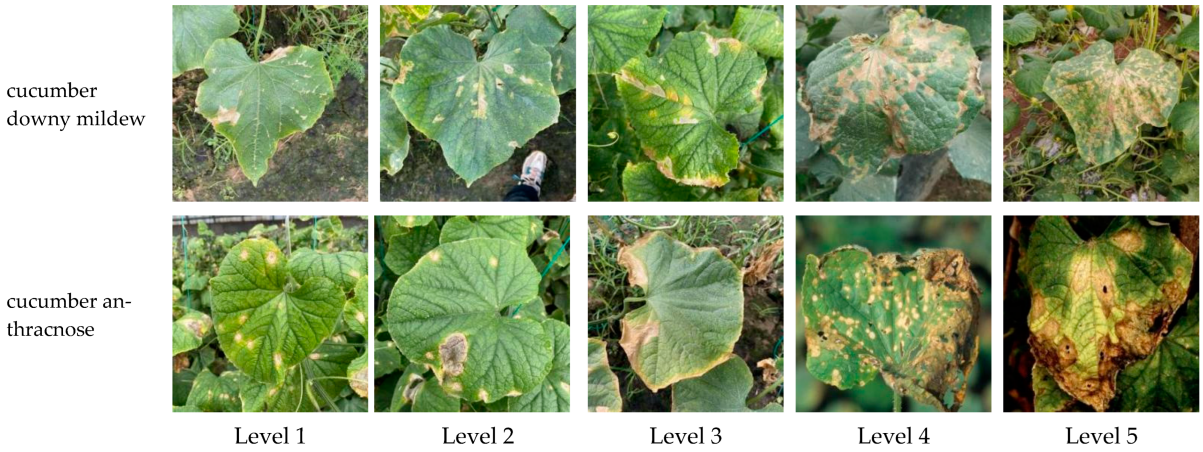

Figure 1.6: Example plot of two diseases severity ratings in cucumber [21].

- **Ratio Scale:** This method provides a direct percentage estimate (from 0 to 100%) of symptomatic organs, as illustrated in Figure 1.7. It offers high precision but requiring experienced raters [18].

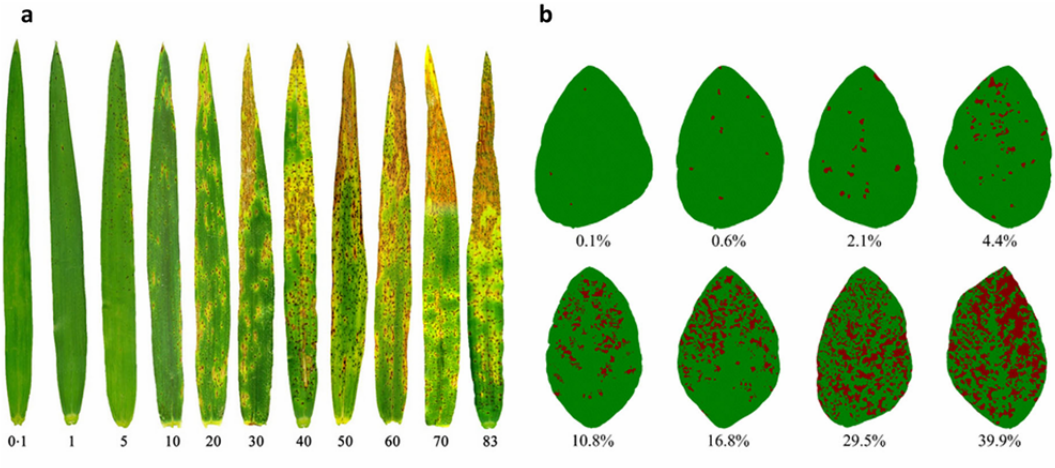

Figure 1.7: Severity estimation using ratio scale for : a. spot blotch severity on wheat leaves. b. Frogeye leaf spot on soybean [17].

1.4.3 Advanced methods

Advancements in plant disease diagnostics have introduced automated and high-precision methods that address the limitations of traditional techniques. The most notable innovations include machine learning-based approaches. Convolutional Neural Networks (CNNs) and Transformers based models are particularly effective in automatically learning features from images.

1.4.3.1 Deep learning models

CNN and Transformers based models have revolutionized plant disease severity estimation by enabling automated classification of leaf diseases from images. These models analyze visual symptoms with high precision, often surpassing traditional methods in accuracy [20].

1.4.4 Limitations of traditional Approaches

These traditional methods face inherent constraints:

- **Subjectivity:** Significant rating variability due to assessor experience, perception, and environmental conditions
- **Resource Intensity:** Time-consuming processes requiring skilled personnel, impractical for large-scale monitoring
- **Symptom Variability:** Symptoms vary widely depending on plant species, disease stage, and environmental conditions.
- **Scalability Issues:** Logistically challenging and costly to implement consistently across diverse agricultural landscapes

1.5 Wheat Yellow rust

Wheat yellow rust, or stripe rust, is a fungal disease caused by *Puccinia striiformis f. sp. tritici*, primarily affecting wheat leaves, though it can also infect stems and spikes, leading to severe grain yield and quality loss [13].
This disease was chosen as the focus of our study due to its widespread presence and significant threat to wheat production globally.

As shown in Figure 1.8 and Figure 1.9, Yellow rust symptoms begin with chlorotic streaks on wheat leaves, which completely colonize the leaves and consume the nutrients synthesized by the host plant. In severe cases, the infection spreads to spikes and stems, leading to significant yield losses.

Figure 1.8: Wheat leaf with Yellow Rust infection [14].

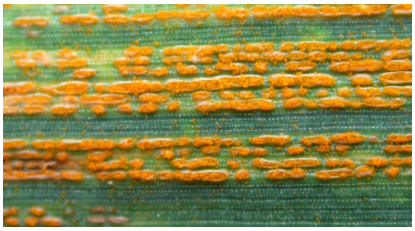

Figure 1.9: Zoomed in: Wheat leaf with Yellow Rust infection [15].

Wheat yellow rust is a major threat to wheat production worldwide due to its ability to spread rapidly and adapt to new environments. The disease can cause yield losses of up to 70% under favorable conditions [16], making early detection and accurate severity estimation essential for effective management. Monitoring yellow rust progression helps optimize control strategies, reduce chemical usage, and support breeding programs for resistant varieties.

1.6 State of the art

Accurate estimation of plant disease severity is essential for effective crop management. This section reviews state-of-the-art approaches to assess disease severity, highlighting key models, datasets, and performance metrics to provide a comparative analysis of current research trends.

Hayit et al., [23] proposed a **deep convolutional neural network-based model, Yellow-Rust-Xception**, to classify the severity of **yellow rust disease in wheat**. Using 5,421 wheat leaf images, the model categorizes five levelsno disease, resistant (R), moderately resistant (MR), moderately susceptible (MS), and susceptible (S)and achieved 91% accuracy.

Wang et al., [25] proposed a **Sliding Segmentation Algorithm (SSA)** to enhance limited training data for soybean bacterial blight. They compared multiple deep learning modelsVGG16 (93.21%), Vision Transformer (94.85%), EfficientNet (95.47%), Swin Transformer V2 (98.12%), and ResNet-50 (96.73%)with the Swin Transformer achieving 99.64% accuracy on 15,600 images across five classes.

Yang et al., [26] proposed a **novel deep learning-based framework, LDI-NET**, integrating CNNs and transformers to simultaneously identify plant type, disease, and severity. The architecture consists of three main modules: **feature tokenizer, token encoder, and multi-label decoder**. The **feature tokenizer module** leverages both **CNNs and transformers**, ensuring the extraction of both **local and global contextual information** from plant leaf images. The **token encoder module** enhances contextual relationships among extracted tokens, improving disease severity identification. The study was conducted using the **AI Challenger 2018 dataset**, comprising **31,718 training images** and achieved an overall accuracy of 87.40%.

Kundu et al., [27] proposed a **deep learning-based framework, MaizeNet** for

automatic disease severity prediction and crop loss estimation in maize. By segmenting diseased regions using K-Means clustering and classifying severity on a 19 scale, the model achieved 98.50% accuracy on 2,996 images, with TLB and Rust measured at 57.48% and 82.13%, respectively.

Parikh et al., [28] proposed a **two-step classification framework** for detecting and estimating the severity of **Grey Mildew disease** in cotton leaves from unconstrained images. The first step segments **leaves from a cluttered background** using statistical color and texture features processed through a **K-Nearest Neighbor (KNN)** classifier. The second step classifies **healthy and diseased regions** based on **hue and luminance** features in the HSV color space. Severity estimation is calculated by measuring the proportion of diseased pixels relative to the leaf area. The model achieved 82.5% accuracy on 190 images.

Wang et al., [29] proposed a **deep learning-based approach** for automatic severity estimation of **apple black rot**. Using transfer learning on the PlantVillage dataset, they tested VGG16, VGG19, Inception-V3, and ResNet50achieving 90.4%, 89.1%, 85.7%, and 80.0% accuracy, respectively; a shallow CNN (8 convolutional layers) reached 79.3%.

Patil et al., [30] proposed an **image processing-based method** to estimate the severity of **brown spot disease** in sugarcane. By segmenting the leaf area with simple thresholding and the lesion area via Triangle thresholding. The severity is calculated as the ratio of diseased area to total leaf area. The method, tested on 90 samples, achieved 98.60% accuracy.

The table 1.2 summarizes several state-of-the-art approaches proposed in the literature for plant disease detection and severity estimation. It highlights the different models, datasets, and evaluation metrics used, providing a clear comparison of their performance and methodological choices.

Table 1.2: Global state-of-the-art summary of plant disease severity estimation methods, including accuracy of all tested models.

Authors	Plant Type	Number of Classes	Dataset	Number of Images	Model Used	Accuracy	Year
[23]	Wheat	6 Healthy,R MR, MRMS MS, S	Yellow-Rust 19	15000	Yellow-Rust-Xception	91%	2021
[24]	Wheat	6 Healthy,R MR, MRMS MS, S	Yellow-Rust 16	5,242	C-DenseNet (DenseNet + CBAM)	97.99%	2020
[25]	Soybean	5 (Normal, Early, Middle, Late, Background)	private	15,600	VGG16 Vision Transformer EfficientNet Swin Transformer V2 ResNet-50 Swin Transformer	93.21% 94.85% 95.47% 98.12% 96.73% 99.64%	2023
[26]	Apple, Cherry, Citrus, Corn, Grape, Peach, Pepper, Potato, Pumpkin, Soybean, Strawberry, Tomato.	3 (Healthy, General, Severe)	AI Challenger 2018	34534	LDI-NET (CNN+Transformer)	87.40%	2024
[27]	Maize	3 (Mild, Moderate, Severe)	private	2,996	MaizeNet (CNN)	98.50%	2022
[28]	Cotton	3 (Stage 1, Stage 2, Stage 3)	private	190	KNN (HSV + Texture Features)	82.5%	2016
[29]	Apple (Black Rot)	4 (Healthy, Early, Middle, End)	PlantVillage	2086	VGG16 VGG19 Inception-V3 ResNet50 Shallow CNN 8	90.4% 89.1% 85.7% 80.0% 79.3%	2017
[30]	Sugarcane	5	private	90	Triangle Thresholding	98.60%	2011
[31]	Cucumber	Continuous scale (percentage)	private	2976	DM-BiSeNet (BiSeNet V2 + MobileNetV3 + Depthwise Separable Convolutions)	$R^2 = 0.9407$ RMSE=1.068	2024
[32]	Leafminer damage	Continuous (percentage)	private	4,782	DeepLab-Leafminer (Edge-aware module Canny loss)	92.38%	2025

1.7 Conclusion

The estimation of plant leaf disease severity is a vital component of precision agriculture, offering the potential to significantly enhance crop management and reduce yield losses. This chapter has explored the nature and impact of various plant leaf diseases, as well as the evolution of disease severity diagnostic techniques from traditional visual inspections to advanced deep learning-based approaches. Recent progress, particularly with the integration of Convolutional Neural Networks and Transformer-based architectures, has enabled more accurate, scalable, and automated assessment of disease severity across a variety of plant species. As reviewed, state-of-the-art models demonstrate promising performance in diverse agricultural contexts, benefiting from large annotated datasets and improved computational methods.

In the next chapter, we will focus on attention mechanisms and Transformer-based models. These models, first used in natural language processing, have recently been applied with success to image analysis tasks. We will explain how they work and why they are well-suited for plant disease severity estimation.

Chapter 2

Transformers and attention mechanism

2.1 Introduction

In recent years, transformers have revolutionized the field of deep learning, especially in NLP (natural language processing) and, more recently, in computer vision. At the heart of these models lies the attention mechanism, a powerful concept that enables models to dynamically focus on the most relevant parts of the input data. This revolutionary approach represents a fundamental departure from traditional sequential architectures, as transformers move away from recurrence and focus on self-attention mechanisms to process data in parallel. Unlike RNNs (Reccurent Neural Networks) that process sequences step-by-step and struggle with vanishing gradients over long distances, transformers can capture long-range dependencies without relying on recurrence, allowing them to directly model relationships between any two positions in a sequence regardless of their distance [33].

Transformers have proven their exceptional performance across numerous fields: natural language processing (powering models like GPT and BERT for translation, text generation, and understanding), computer vision (Vision Transformers for image classification and object detection), speech recognition, code generation and programming assistance, multimodal tasks (combining text, images, and audio), etc. [34].

This chapter introduces the foundations of attention, then explores two of the most influential transformer architectures in the field of vision: the Vision Transformer (ViT) and the Swin (Shifted Window) Transformer. We present their architectures, discuss their key variants, and highlight how they adapt the transformer paradigm to effectively process image data. Also, we justify our choice of these two architectures for our application, as they represent complementary approaches to visual processing, ViT with global attention, and Swin with hierarchical local attention.

2.2 Attention mechanism

Attention mechanism emerged as a fundamental invention in artificial intelligence (IA) and machine learning (ML) that redefined the potential of deep learning models.

Attention mechanism is a technique used in deep learning models that allows the model to choose focus on particular regions of the input data. When dealing with lengthy data sequences, such as in computer vision or natural language processing (NLP) jobs, attention mechanism is particularly beneficial. Attention mechanism allows the model to pay different levels of attention to distinct bits of data instead of processing all inputs in the same way. It allows models to concentrate on particular segments of an image that are more informative for disease severity classification.

By assigning different weights or attention scores to various parts of an image, these mechanisms enable the model to prioritize disease stage-related features and disregard irrelevant or misleading information.

The influential paper "Attention Is All You Need" [35], published by Vaswani et al.

(2017), marked a significant advancement in classification tasks across machine learning applications. Prior to this innovation, classification models primarily relied on convolutional neural networks or recurrent architectures with limited ability to weigh the importance of different input features. The attention mechanism addressed this limitation by enabling models to assign varying levels of significance to different parts of the input data during classification processes.

By implementing self-attention, classification models gained the ability to create context-aware representations of input features, leading to more accurate discrimination between classes and improved performance on complex categorization problems. The subsequent adoption of Transformer-based architectures in classification systems demonstrated substantial improvements over previous methods, particularly when handling long sequences or when interpretability of classification decisions was required.

2.3 Motivation for models selection

2.3.1 Choice of the models

In recent years, numerous Transformer-based architectures have been proposed for computer vision tasks. Among them, we selected Vision Transformer (ViT) and Swin Transformer as the core components of our model due to their complementary characteristics and strong performance.

ViT is the first pure Transformer model successfully applied to image classification, introducing a patch-based tokenization approach that enables global context modeling through self-attention. However, ViT does not naturally take into account important image properties, such as spatial locality. This can make it less effective when working with smaller datasets.

Swin Transformer, on the other hand, addresses this limitation by introducing a hierarchical structure with shifted windows, allowing it to capture both local and global dependencies efficiently. Its design also makes it more scalable and suitable for dense prediction tasks.

2.4 Vision Transformer (ViT)

The Vision Transformer (ViT), introduced by Dosovitskiy et al [36], signifies a paradigm shift in the domain of computer vision by leveraging self-attention mechanismsoriginally devised for natural language processingto vision tasks. Its capability to capture long-range dependencies through self-attention makes it particularly suited for complex image understanding tasks. The core innovation of ViT lies in its treatment of images as sequences of patches rather than utilizing the spatial inductive biases inherent to convolutional operations. This section delineates the principal components and sequential operations of ViT.

2.4.1 ViT applications

Vision Transformers have demonstrated remarkable versatility across diverse computer vision tasks, extending beyond traditional image classification to complex multimodal applications [37] :

- **Image Classification**: Categorizing images into predefined classes. ViTs outperform CNNs on very large datasets.
- **Image Captioning**: Generating descriptive text captions for images by learning general visual representations.
- **Image Segmentation**: Partitioning images into semantic regions.
- **Anomaly Detection**: Identifying unusual patterns in images using reconstruction-based approaches with patch embedding.
- **Action Recognition**: Classifying human actions in videos by extracting spatiotemporal tokens through transformer layers.

2.5 ViT architecture

The Vision Transformer (ViT) architecture, as illustrated in Figure 2.1, begins by dividing an input image into fixed-size patches, which are flattened and mapped into a high-dimensional embedding space via a learnable linear projection. Positional encodings are then added to these embeddings, and a dedicated class token is appended to capture global context. The resulting token sequence is processed by a series of transformer encoder layers, as shown in Figure 2.2, each comprising multi-head self-attention and feed-forward sub-networks, to iteratively refine the representation. Finally, the output corresponding to the class token is passed through a classification head to produce the final prediction.

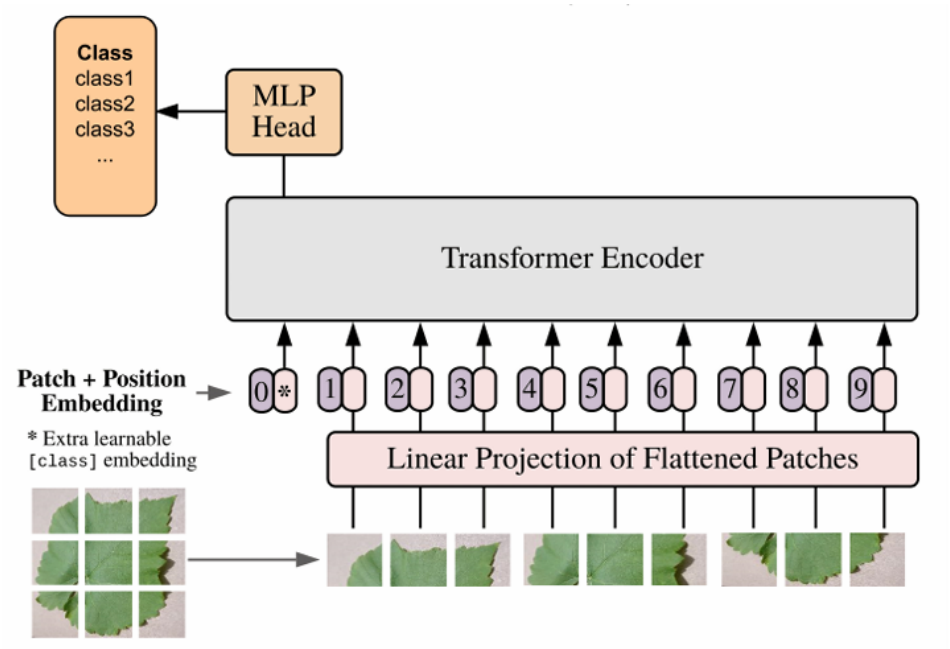

Figure 2.1: Vision Transformer architecture [36].

2.5.1 Patch Extraction and Linear Embedding

An input image of dimensions $H \times W$ with C channels is divided into a grid of non-overlapping patches, each of size $P \times P$. This operation yields a total of N patches :

$$N = \frac{H \times W}{P^2} \tag{2.1}$$

Each patch is subsequently flattened into a one-dimensional vector of length P^2C. A learned linear projection, represented by the weight matrix $\mathbf{W}_{\text{proj}} \in \mathbb{R}^{(P^2C)\times D}$, transforms each flattened patch into a D-dimensional embedding, as follows:

$$\mathbf{x}_p^{(i)} = \mathbf{W}_{\text{proj}} \cdot \text{vec}(\mathbf{P}^{(i)}), \tag{2.2}$$

where $\mathbf{P}^{(i)}$ denotes the i^{th} patch. This process preserves local structural information while converting the spatial image into a sequential format amenable to transformer processing.

2.5.2 Incorporation of Positional Encodings and Class Token Augmentation

Given that the patch extraction process disregards the original spatial order, it is necessary to reintroduce positional information. This is achieved by adding learnable positional encodings $\mathbf{E}_{\text{pos}} \in \mathbb{R}^{(N+1)\times D}$ to the patch embeddings. Additionally, a special class token $\mathbf{x}_{\text{class}} \in \mathbb{R}^D$ is prepended to the sequence, resulting in:

$$\mathbf{z}_0 = \left[\mathbf{x}_{\text{class}};\, \mathbf{x}_p^{(1)} + \mathbf{E}_{\text{pos}}^{(1)};\, \ldots;\, \mathbf{x}_p^{(N)} + \mathbf{E}_{\text{pos}}^{(N)}\right]. \tag{2.3}$$

This class token is designed to aggregate information across the entire image, serving as the basis for subsequent classification tasks.

2.5.3 Transformer Encoder Blocks

The enriched sequence $\mathbf{z}_0$ is then input to a stack of L transformer encoder blocks, as shown in Figure 2.2 :

Each encoder block is comprised of two primary sub-layers:

- **Multi-Head Self-Attention (MHSA):** For each token, queries, keys, and values are computed via learned linear mappings.

  Given an input sequence $\mathbf{z} \in \mathbb{R}^{N\times D}$, where N is the number of tokens and D is the input feature dimension, we compute the queries ($\mathbf{q}$), keys ($\mathbf{k}$), and values ($\mathbf{v}$) using a single learned linear projection:

  $$[\mathbf{q}, \mathbf{k}, \mathbf{v}] = \mathbf{z}\mathbf{U}_{qkv}, \mathbf{U}_{qkv} \in \mathbb{R}^{D\times 3D_h} \tag{2.4}$$

  Each token's query interacts with all keys through a dot product to compute the attention weights:

$$A = \text{softmax}\left(\frac{\mathbf{q}\mathbf{k}^\top}{\sqrt{D_h}}\right), \quad A \in \mathbb{R}^{N\times N} \tag{2.5}$$

These attention weights are then used to compute a weighted sum of the values:

$$\text{SA}(\mathbf{z}) = A\mathbf{v} \tag{2.6}$$

To improve model capacity, multi-head self-attention (MHSA) runs k self-attention computations (called heads) in parallel. Each head has its own projection matrices. The outputs of all heads are concatenated and linearly projected:

$$\text{MHSA}(\mathbf{z}) = [\text{SA}_1(\mathbf{z}); \cdots ; \text{SA}_k(\mathbf{z})]\, \mathbf{U}_{msa}, \quad \mathbf{U}_{msa} \in \mathbb{R}^{k\cdot D_h \times D} \tag{2.7}$$

This design enables the model to jointly attend to information from different representation subspaces at different positions.

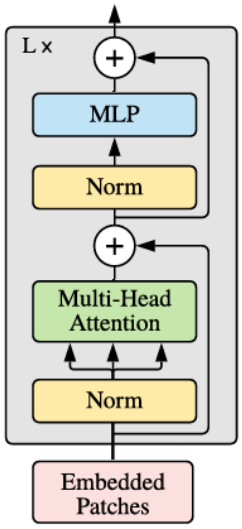

Figure 2.2: Transformer Encoder [36].

- **Feed-Forward Neural Network (FFN):** The FFN is applied independently to each token and is identical across all positions. It consists of a two-layer multilayer perceptron (MLP) with a non-linear activation in between.

  Given an input $\mathbf{x} \in \mathbb{R}^D$ (the output of the MHSA sub-layer), the FFN transforms it as follows:

$$\text{FFN}(\mathbf{x}) = \mathbf{W}_2\, \phi(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2 \tag{2.8}$$

  where:

  - $\mathbf{W}_1 \in \mathbb{R}^{D\times D_{ff}}$, $\mathbf{W}_2 \in \mathbb{R}^{D_{ff}\times D}$ are learnable weight matrices,
  - $\mathbf{b}_1 \in \mathbb{R}^{D_{ff}}$, $\mathbf{b}_2 \in \mathbb{R}^D$ are bias terms,
  - $\phi(\cdot)$ is a non-linear activation function, GELU (Gaussian Error Linear Unit).

  This structure expands the feature dimension from D to $4D$, applies the activation, and projects it back to dimension D.

To facilitate training, residual connections and layer normalization are applied around both the MHSA and FFN sub-layers. For an input $\mathbf{x}$, a generic sub-layer is wrapped as:

$$\text{Output} = \text{LayerNorm}(\mathbf{x} + \text{SubLayer}(\mathbf{x})) \tag{2.9}$$

This design helps stabilize training, preserves information through residual paths, and improves gradient flow.

Together, these components enable the encoder blocks to iteratively refine the extracted features, capturing both local and global patterns in the image.

2.5.4 Final Classification

After processing through the transformer encoder blocks, the output corresponding to the class tokennow enriched with aggregated contextual informationis extracted. This output is subsequently fed into a classification head, typically implemented as a multilayer perceptron (MLP), to map the learned representation to the final output space:

$$\hat{y} = \text{MLP}(\mathbf{z}_{\text{class}}). \tag{2.10}$$

2.6 ViT variants

The original Vision Transformer (ViT) framework has since inspired the development of several variants designed to accommodate varying computational budgets and application requirements. In particular, models such as ViT-Tiny, ViT-Small, ViT-Base, and ViT-Large offer distinct parameterizations that directly influence model capacity, computational complexity, and overall performance.

Parameterization and Model Complexity

ViT variants are primarily differentiated by several hyperparameters, including:

- **Depth (L):** The number of transformer encoder layers, with deeper models generally providing enhanced representational capabilities.
- **Embedding Dimension (D):** The dimensionality of the token embeddings. A higher dimensionality permits a richer representation of image patches.
- **Number of Self-Attention Heads:** This parameter dictates the number of parallel attention operations, which allow the model to capture diverse relationships among tokens.
- **MLP Dimension:** The size of the feed-forward sub-network within each transformer block, often set as a multiple of the embedding dimension.

A judicious selection of these parameters is crucial when deploying vision transformers in resource-constrained environments or when higher performance is demanded by complex visual tasks.

Table 2.1: Details of Vision Transformer model variants [36].

Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Tiny	12	192	768	3	5.7M
ViT-Small	12	384	1536	6	22M
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

The table 2.1 presents the configurations of various Vision Transformer (ViT) model variants, including ViT-Tiny, ViT-Small, ViT-Base, and ViT-Large. It outlines key parameters such as the number of layers (L), model dimension (D), self-attention heads, and total parameter count, which range from 5-10 million for ViT-Tiny to 307 million for ViT-Large. These configurations are critical in determining computational efficiency and model capacity, thereby guiding the selection of an optimal variant based on resource constraints and task-specific performance requirements in visual processing applications.

2.7 Swin (Shifted Window) Transformer

The Swin Transformer [38] introduces an innovative architectural framework for visual representation learning, effectively integrating the expressive power of Transformer models with the computational efficiency and locality biases essential for computer vision applications. Unlike earlier approaches that compute global self-attention over all image tokens, the Swin Transformer introduces a hierarchical architecture that partitions an image into local windows and subsequently aggregates features across multiple scales, as illustrated in Figure 2.3. This design not only reduces computational complexity but also enables the effective modeling of visual entities at different resolutions [38].

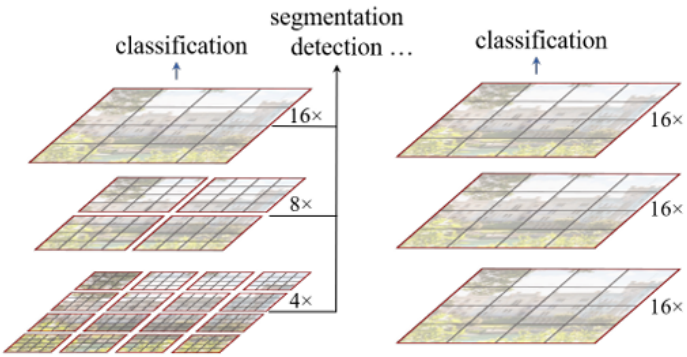

Figure 2.3: a. Swin Transformer. b. Vision Transformer [38].

2.7.1 Applications of Swin Transformer

The Swin Transformer has demonstrated strong performance across various domains due to its hierarchical architecture and efficient attention mechanism. Its applications span both traditional computer vision tasks and extend into other areas such as medical imaging and natural language processing [39]:

- **Image Classification:** T It uses its hierarchical structure for feature extraction at multiple scale and these features help in classification of image.
- **Object Detection:** It detects fine details present in image this helps in understanding its global context. This can be used to detect various objects present in a image.
- **Image Segmentation:**Feature extraction at multiple scale helps in segmenting image into different distinct regions.
- **Medical Imaging:** Swin Transformers precise feature extraction aids in identifying anomalies in medical scans, contributing to improved diagnostic performance in healthcare applications.
- **Natural Language Processing (NLP):** Although primarily designed for computer vision, Swin Transformer can be adapted for NLP tasks by modifying its architecture to process sequential textual data.

2.8 Swin architecture

As illustrated in Figure 2.4, the Swin Transformer follows a hierarchical feature representation approach, where the input image undergoes successive transformations through patch partitioning, embedding, and multi-head self-attention (MHSA) in shifted windows.

Each stage is designed to efficiently balance local feature extraction and global context aggregation while reducing the spatial resolution of the feature map and increasing the depth of representations.

The Swin Transformer's hierarchical design allocates distinct stages to different vision applications: initial stages (Stage 12) focus on local feature extraction (e.g., object detection, instance segmentation) using high-resolution feature maps, while later stages (Stage 34) capture global semantic context, making them ideal for image classification and scene understanding

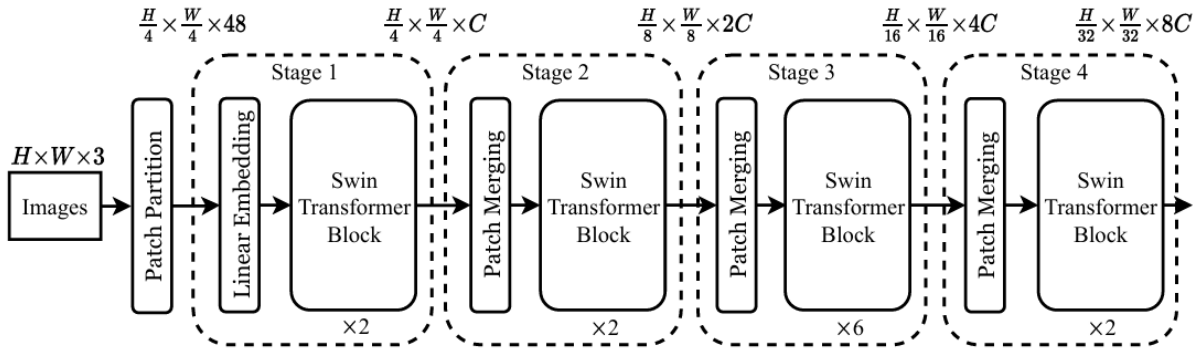

Figure 2.4: Swin architecture [38].

The process starts with an input image of dimensions $H \times W \times 3$, where H and W denote the height and width, respectively, and the three channels correspond to RGB

color information. This image is the raw data that is progressively transformed into a compact, high-level representation.

First, the input image is passed through a Patch Partition, to split it into a fixed-size non-overlapping patches.

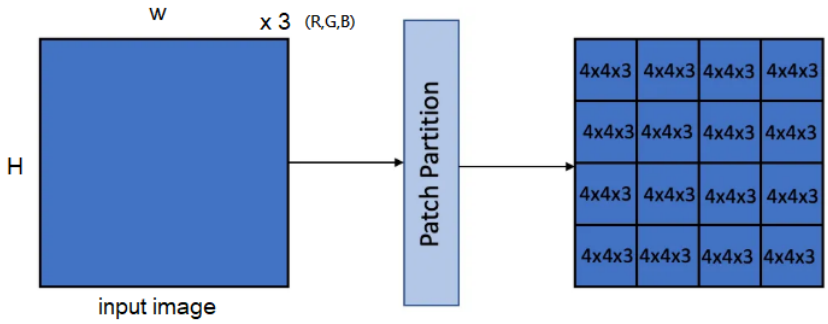

Figure 2.5: Patch Partition [40].

A patch size of 4 x 4 is used and the Patch Partition gives us H/4 x W/4 patches. Each patch has a channel dimension of 4 x 4 x 3 = 48 pixels and is treated as a token whose feature vector is derived from raw pixel values.

2.8.1 Stage 1: Patch Partitioning and Embedding

- **Linear embedding layer:** Each patch is projected into a feature vector of size C using a linear layer. This creates a new representation suitable for the transformer.
- The resulting feature map is then passed through a **Swin Transformer block**, which keeps the same shape for both input and output.
- **Swin Transformer blocks** (see Figure 3.16) applies attention mechanisms:
  - **Window-based multi-head self-attention (W-MSA).**
  - **Shifted window multi-head self-attention (SW-MSA).**
- These blocks allow the model to capture relationships between patches within local windows and across neighboring windows.

  → Final output shape of this stage: $\frac{H}{4} \times \frac{W}{4} \times C$

2.8.2 Stage 2: Patch Merging and Feature Transformation

→ Input shape of this stage: $\frac{H}{4} \times \frac{W}{4} \times C$

1. The first **patch merging layer** concatenates the features of 2 x 2 neighboring patches and applies a linear layer, reducing the number of tokens by 4 x (effectively downsampling by a factor of 2).
2. This stage introduces a higher-dimension representation (**2C** instead of **C**) while maintaining spatial hierarchy.

3. Successive **Swin Transformer blocks** 3.16 operate on these merged patches, refining feature extraction.

→ Output shape of this stage: $\frac{H}{8} \times \frac{W}{8} \times 2C$

2.8.3 Stage 3 and Stage 4: Progressive Feature Refinement

1. Each subsequent stage further reduces the resolution by applying additional patch merging layers, progressively increasing the depth of feature representations and a successive Swin Transformer blocks.

2. The final output contains a compact yet semantically rich representation suitable for various computer vision tasks like image classification and dense prediction.

Table 2.2: Input and Output Shapes of Swin Transformer Stages.

Stage	Input Shape	Output Shape
Stage 1	$H \times W \times 3$	$\frac{H}{4} \times \frac{W}{4} \times C$
Stage 2	$\frac{H}{4} \times \frac{W}{4} \times C$	$\frac{H}{8} \times \frac{W}{8} \times 2C$
Stage 3	$\frac{H}{8} \times \frac{W}{8} \times 2C$	$\frac{H}{16} \times \frac{W}{16} \times 4C$
Stage 4	$\frac{H}{16} \times \frac{W}{16} \times 4C$	$\frac{H}{32} \times \frac{W}{32} \times 8C$

As shown in Table 2.2, the Swin Transformer employs a hierarchical architecture where each stage progressively reduces spatial resolution while expanding channel depth. The network transforms an input image ($H \times W \times 3$) into a high-dimensional feature representation ($\frac{H}{32} \times \frac{W}{32} \times 8C$) through successive downsampling operations.

2.8.4 Inside Two Successive Swin Transformer Blocks

Each Swin Transformer block (Figure 3.16) consists of multiple layers that sequentially transform feature representations while maintaining efficiency through window-based self-attention.

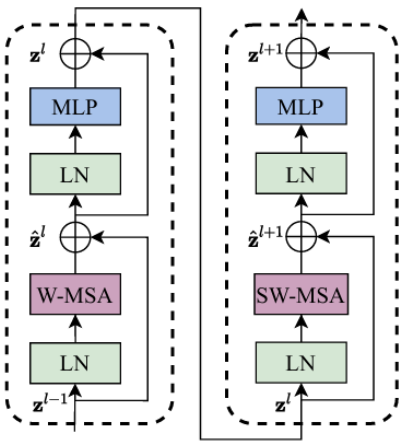

Figure 2.6: Two successive Swin Transformer Blocks with regular and shifted windowing configurations [38].

With the shifted window partitioning approach, consecutive Swin Transformer blocks are computed as :

$$\hat{z}^l = \text{W-MSA}\left(\text{LN}\left(z^{l-1}\right)\right) + z^{l-1}, \tag{2.11}$$

$$z^l = \text{MLP}\left(\text{LN}\left(\hat{z}^l\right)\right) + \hat{z}^l, \tag{2.12}$$

$$\hat{z}^{l+1} = \text{SW-MSA}\left(\text{LN}\left(z^l\right)\right) + z^l, \tag{2.13}$$

$$z^{l+1} = \text{MLP}\left(\text{LN}\left(\hat{z}^{l+1}\right)\right) + \hat{z}^{l+1}, \tag{2.14}$$

Where z^l and $\hat{z}^l$ denote the outputs of the MLP and self-attention sub-layers at block l, respectively. W-MSA and SW-MSA refer to window-based multi-head self-attention with regular and shifted partitioning, respectively, as illustrated in Figure 3.16. Specifically, Equations (2.11) and (2.12) describe the operations in the first block (W-MSA), while Equations (2.13) and (2.14) correspond to the second block (SW-MSA).

1. **First Swin Transformer Block: Window-Based Multi-Head Self-Attention (W-MSA) :**

   - **Layer Normalization (LN)**: Before computing self-attention, each token embedding undergoes LayerNorm (LN) to stabilize training and normalize feature distributions.
   - **Multi-Head Self-Attention (W-MSA)**: Instead of computing self-attention globally, attention is applied within local non-overlapping windows of size $M \times M$, as shown in Figure 2.7, and calculated using the expression 2.15:

$$\text{Attention}(Q_i, K_j, V_j) = \text{Softmax}_j\left(\frac{Q_i K_j^\top}{\sqrt{d}} + B_{i,j}\right) V_j \tag{2.15}$$

     – Where :
     – **Q :** Query matrix: the input being processed.
     – **K :** Key matrix: used to match against the query.
     – **V :** Value matrix: holds the information to be aggregated.
     – **D :** Scaling factor (usually $\sqrt{d_k}$) to prevent large dot product values.
     – **softmax :** Converts similarity scores into attention weights.

     — **The relative position bias $B_{i,j}$:** encodes the spatial relationship between tokens i and j within a local window. Instead of using absolute positional encodings, the model learns a set of biases based on the relative positions (e.g., token i is 2 steps to the right of token j), which allows for better translation-invariance and locality in vision tasks. These biases are added directly to the attention logits before the softmax, influencing how much attention is paid based on relative positions.

     This operation allows local contextual feature extraction while reducing computational complexity.

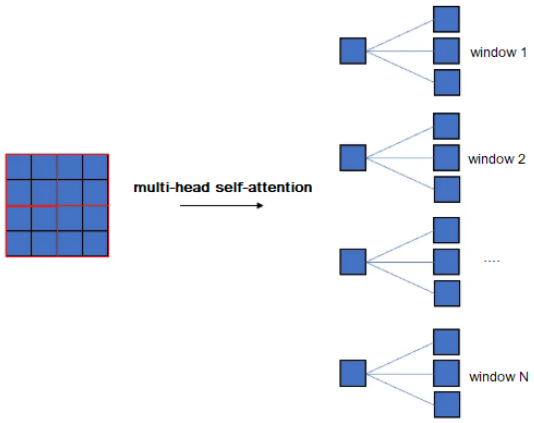

Figure 2.7: W-MSA in Swin Transformer [40].

- **Residual Connection**: The attended tokens are added back to the original input, preventing gradient vanishing.
  This results in the intermediate representation $\hat{z}^l$, as defined in Equation (2.11).
- **MLP (Multi-Layer Perceptron) block**: After the attention mechanism, a two-layer MLP is applied to each token independently. This helps the model learn more complex feature transformations. The MLP consists of:
  - A linear layer with weight matrix W_1.
  - A non-linear activation function: GELU (Gaussian Error Linear Unit).
  - Another linear layer with weight matrix W_2.

  This can be written as:
  $$x' = \text{GELU}(W_1 x) W_2$$
  Here:
  - x is the input token embedding
  - W_1 projects x to a higher-dimensional space (increasing capacity)
  - **GELU activation:** The Gaussian Error Linear Unit (GELU) is a smooth, non-linear activation function. It is defined as:
    $$\text{GELU}(x) = x \cdot \Phi(x), \tag{2.16}$$
    where $\Phi(x)$ is the standard Gaussian CDF[1].
  - W_2 projects back to the original dimension

  This process allows the model to refine the features learned by the attention mechanism.
- **Second Layer Normalization (LN)**: Another normalization step ensures stable feature propagation.

[1]The cumulative distribution function (CDF) gives the probability that a standard normal variable is less than or equal to x. It is defined as: $\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2} dt$.
In a Transformer, the CDF is used in the GELU activation function to provide a more probabilistic and smooth behavior to neuron activation.

- **Residual Connection**: The processed features are added back to the input of the MLP, improving network stability.
  This yields the output z^l, as shown in Equation (2.12).

2. **Second Swin Transformer Block: Shifted Window Multi-Head Self-Attention (SW-MSA) :**

   - This block follows the same structure as the first Swin Transformer block, with the key difference being the use of **(SW-MSA)** instead of W-MSA.
   - The shifted windows allow cross-window connections, enabling better modeling of global dependencies without significantly increasing computational complexity.

2.8.5 Cyclic Shift

To enhance cross-window information exchange while maintaining computational efficiency, the Swin Transformer introduces a cyclic shift strategy prior to the self-attention operation. Specifically, in the Shifted Window Multi-Head Self-Attention (SW-MSA) module, the input feature map is shifted by a fixed number of pixels (typically half the window size) along the spatial dimensions before window partitioning. This shift brings tokens from neighboring windows into the same local window, allowing information exchange across regions (see Figure 2.8).

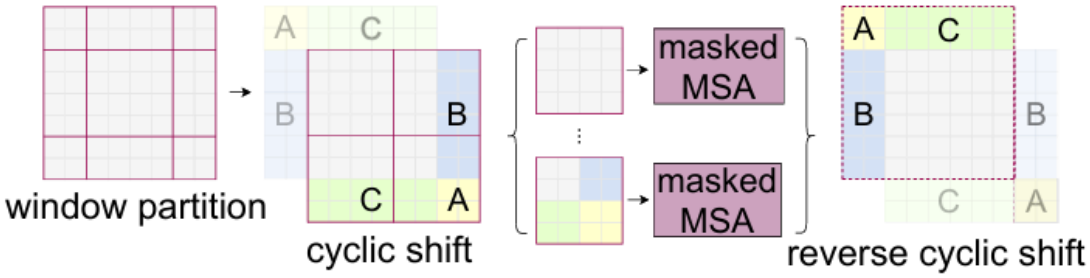

Figure 2.8: Illustration of efficient batch computation approach for self-attention in shifted window partitioning [38].

However, this shift also introduces tokens from different original windows into the same attention window, which may result in unintended connections. To prevent such cross-window attention, a **binary attention mask** is applied during the computation of self-attention. This mask assigns zero to allowed token pairs and large negative values (e.g., $-\infty$) to disallowed ones, effectively blocking attention between unrelated tokens. It ensures each token only attends to others from the same window prior to shifting, preserving local structure while enabling hierarchical modeling.

After attention is computed, a reverse cyclic shift restores the original spatial alignment of tokens.

2.9 Swin Variants

The Swin Transformer is designed with multiple variants to balance computational cost and performance for various vision tasks. The main variantsSwin-Tiny, Swin-Small, Swin-Base, and Swin-Largediffer primarily in terms of network depth, embedding dimensions, and the number of attention heads. These configuration differences allow practitioners to select a variant that best suits the available computational resources and specific task requirements.
Table 2.3 summarizes the common configurations for these variants where depth and number of heads are given per stage.

Table 2.3: Details of Swin Transformer model variants [38].

Variant	Layers	Embedding Dim (D)	Attention Heads	Params
Swin-Tiny	[2, 2, 6, 2]	96	[3, 6, 12, 24]	29M
Swin-Small	[2, 2, 18, 2]	96	[3, 6, 12, 24]	50M
Swin-Base	[2, 2, 18, 2]	128	[4, 8, 16, 32]	88M
Swin-Large	[2, 2, 18, 2]	192	[6, 12, 24, 48]	197M

- Where :
- **Depth (per stage)**: Number of Transformer blocks in each of the four stages of the network.
- **Embedding Dim (D)**: The dimensionality of the token embeddings at the first stage. This increases by a factor of 2 at each subsequent stage.
- **Attention Heads**: Number of self-attention heads used in each stage.
- **Parameters**: Approximate total number of trainable parameters in the model.

2.10 Conclusion

Transformers, through their attention-driven design, have opened new possibilities for vision tasks by enabling models to reason globally or locally depending on their architecture. The Vision Transformer offers a pure attention-based approach with strong representational power, while the Swin Transformer brings a more scalable, hierarchical structure tailored to visual patterns.

Our choice to focus on ViT and Swin stems from their proven effectiveness and complementary characteristics. Furthermore, selecting ViT-Small and Swin-Base variants allows us to balance computational cost with model accuracy, making them well-suited for our objective of disease severity estimation. A solid understanding of these architectures lays the foundation for the design and development of our proposed solution.

In the next chapter, we delve deeper into the architecture of the selected transformer variants. This detailed analysis will shed light on their internal components and design principles, providing a solid foundation for their integration into our proposed methodology for plant leaf disease severity estimation.

Chapter 3

Proposed Model Approach for Estimating Plant Leaf Disease Severity

3.1 Introduction

Precise estimation of plant disease severity plays a crucial role in modern agriculture, enabling informed decision-making for disease control and maximizing crop yields. With increasing pressure to reduce pesticide use and enhance food security, accurate and automated methods for assessing leaf disease severity have become indispensable tools for researchers and farmers alike.

This chapter presents a comprehensive approach to plant leaf disease severity estimation using two distinct model architectures: Vision Transformer Small (ViT-Small), Swin Transformer Base (Swin-Base). Our approach addresses the inherent challenges in disease severity estimation, including the subtle visual differences between severity levels, varying lighting conditions, and the need for fine-grained feature extraction from leaf images.

The proposed methodology transforms the complex task of disease severity assessment into a multi-class classification problem, enabling automated and consistent evaluation of plant health status.

3.2 Motivation for variants choice

We considered two Transformer-based architectures: ViT-Small and Swin-Base, to evaluate the trade-offs between model complexity, accuracy, and computational cost.

- **ViT-Small**: [around 22 million parameters]
  A lightweight version of the Vision Transformer, ViT-Small requires fewer parameters and less computation. It is well-suited for medium-scale datasets and limited resources, while still benefiting from global self-attention.
- **Swin-Base**: [around 88 million parameters]
  Swin Transformer introduces a hierarchical structure with shifted windows, which allows for both local and global feature extraction. Swin-Base strikes a good balance between accuracy and computational cost, offering better performance than Swin-Tiny on various vision benchmarks.

3.3 ViT-small

The Vision Transformer (ViT) architecture treats images as sequences of patches, similar to tokens in natural language processing. The ViT-Small variant employs a smaller number of parameters while retaining the model's ability to capture global dependencies through self-attention mechanisms.

The main parameters of the ViT-Small model are summarized in Table 3.1. The model takes input images of size 224×224 pixels and divides them into patches of size 16×16, resulting in 196 patches per image. Each patch is linearly projected into a 384-dimensional embedding. The transformer backbone consists of 12 layers, each with 6 attention heads. The feed-forward (MLP) part of each layer has a hidden dimension of

1536. A dropout rate of 0.1 is applied during training to reduce overfitting. Overall, the model contains approximately 22 million parameters. The final prediction is made using a linear classification head that outputs the severity class.

Table 3.1: Key Parameters of the ViT-Small Model

Parameter	**Value**
Patch Size	16×16
Input Image Size	224×224
Embedding Dimension	384
Number of Transformer Layers	12
Number of Attention Heads	6
MLP Hidden Dimension	1536
Dropout Rate	0.1
Number of Parameters	22 million
Classification Head	Linear

ViT architecture, as illustrated in Figure 3.1, consists of several interconnected components that transform raw leaf images into disease severity classifications. The following provides a detailed step-by-step explanation of each architectural component:

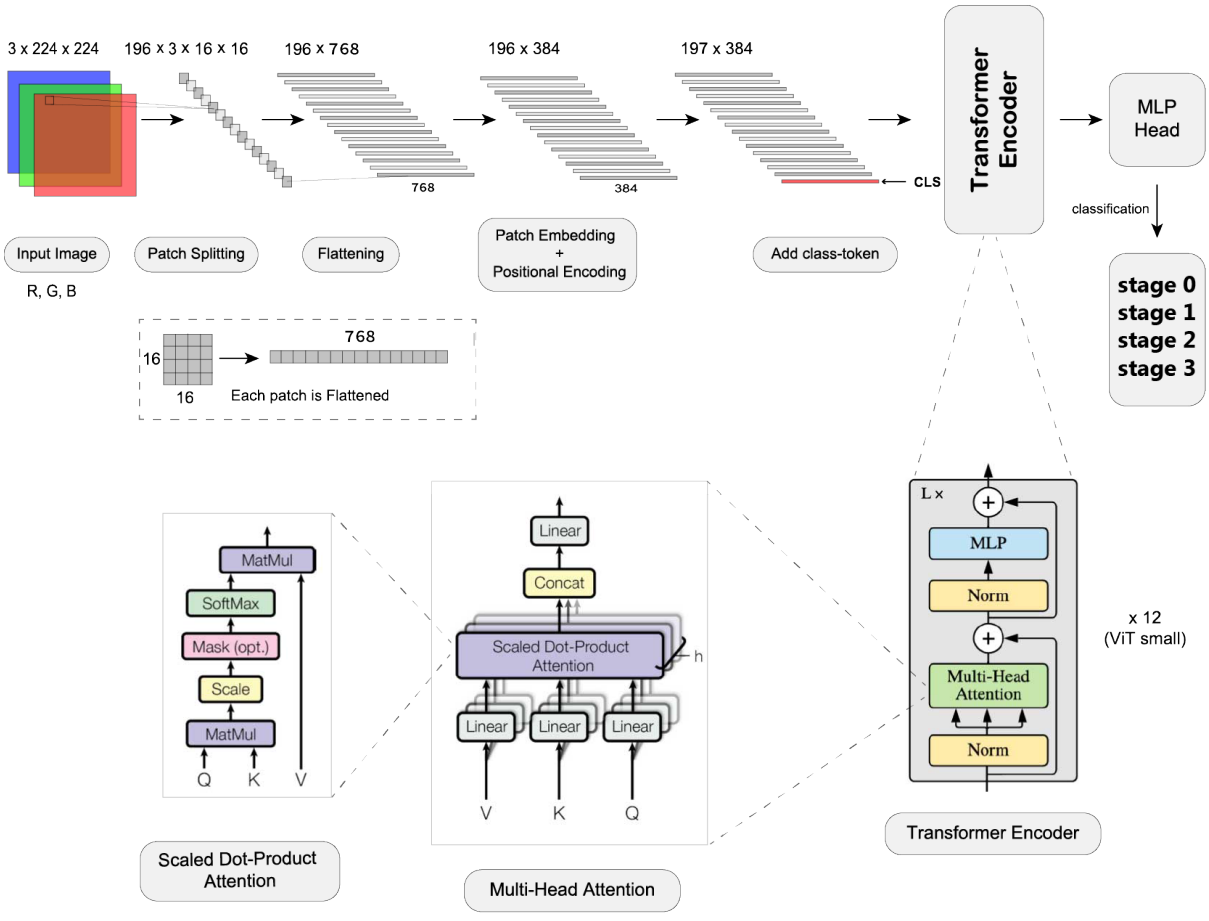

Figure 3.1: Architecture of the ViT-Small model.

3.3.1 Step 1: Input Image Processing

- **Input Dimensions**: The architecture begins with an RGB input image of dimensions $224 \times 224 \times 3$, where the three channels represent Red (R), Green (G), and Blue (B) color components.
- **Image Representation**: The input image contains plant leaf samples with varying degrees of disease severity, captured under different lighting conditions and orientations.
- **Preprocessing**: Standard normalization is applied to ensure pixel values are within the appropriate range for neural network processing.

3.3.2 Step 2: Patch Splitting and Tokenization

- **Patch Extraction**: The 224×224 input image is systematically divided into non-overlapping square patches of size 16×16 pixels.
- **Patch Count**: This division results in $\frac{224}{16} \times \frac{224}{16} = 14 \times 14 = 196$ individual patches. [see Figure 3.2]

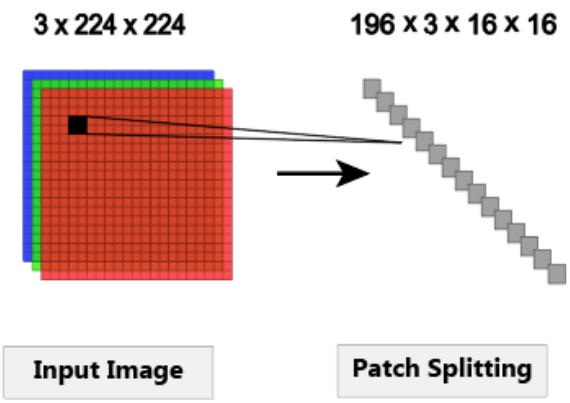

Figure 3.2: ViT-base : Patch splitting.

3.3.3 Step 3: Flattening

- **Patch Flattening**: Individual patches are flattened from $16 \times 16 \times 3 = 768$ dimensional vectors, as shown in the detailed patch flattening illustration. [see Figure 3.3]

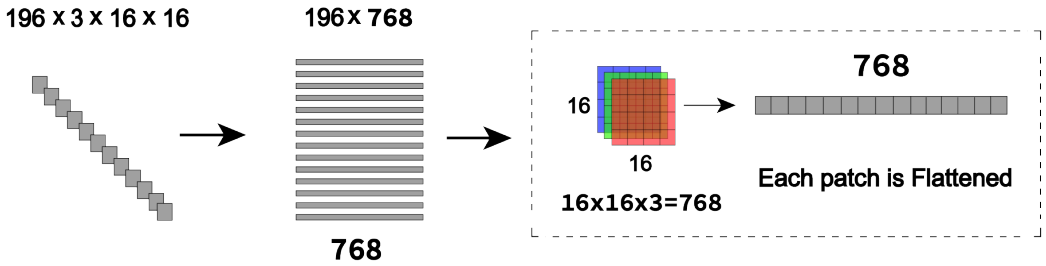

Figure 3.3: ViT-base : Patch flattening.

3.3.4 Step 4: Linear Embedding and Dimensionality Transformation

- **Embedding Layer**: Each 768-dimensional flattened patch undergoes linear transformation through a learnable embedding matrix.
- **Output Dimensions**: The embedding process transforms each patch into a 384-dimensional feature vector, resulting in 196×384 patch embeddings, as illustrated in Figure 3.4.
- The embedding layer learns to map raw pixel values into a more meaningful feature space suitable for transformer processing.

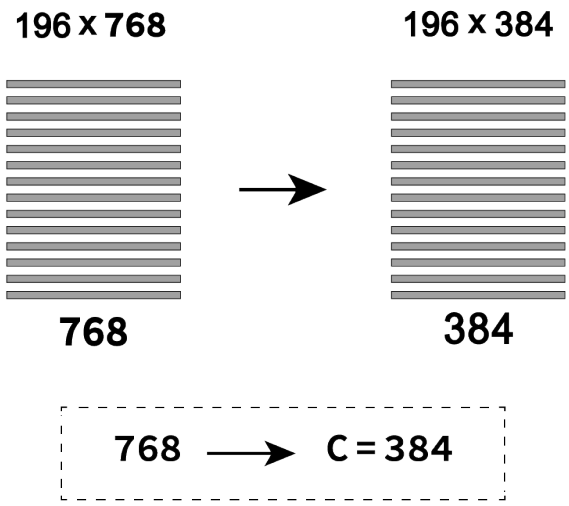

Figure 3.4: ViT-base : Patch Embedding.

3.3.5 Step 5: Positional Encoding Integration

- **Positional Information**: Since transformers are inherently permutation-invariant, positional encodings are added to preserve spatial relationships between patches.
- Positional encodings are element-wise added to patch embeddings, maintaining the 384-dimensional representation.
- This step ensures the model can distinguish between patches from different spatial locations within the leaf image.

3.3.6 Step 6: Class Token Insertion

- **CLS Token**: A special classification token `[CLS]` is prepended to the sequence of patch embeddings.
- The CLS token serves as a global representation that aggregates information from all patch tokens through self-attention mechanisms.
- The input sequence expands from 196 patch tokens to 197 tokens (including the CLS token), as shown in Figure 3.5.

- The CLS token is initialized as a learnable parameter that evolves during training.

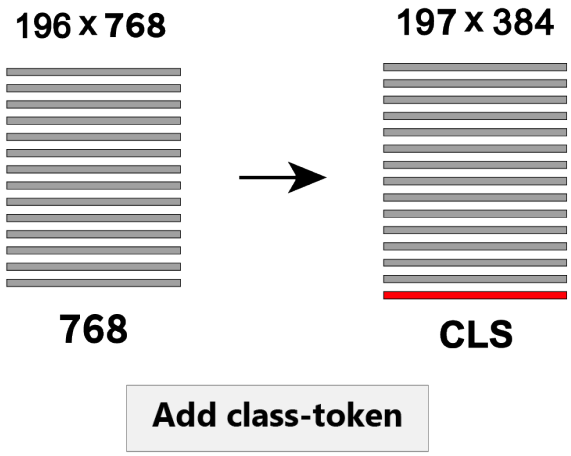

Figure 3.5: ViT-base : Adding class token [CLS].

3.3.7 Step 7: Transformer Encoder Processing

The core transformer encoder processes the token sequence through multiple identical layers:

3.3.7.1 Multi-Head Self-Attention Mechanism

- **Query, Key, Value Generation**: Each input token is linearly transformed to generate Query (Q), Key (K), and Value (V) matrices, as shown in Figure 3.6.

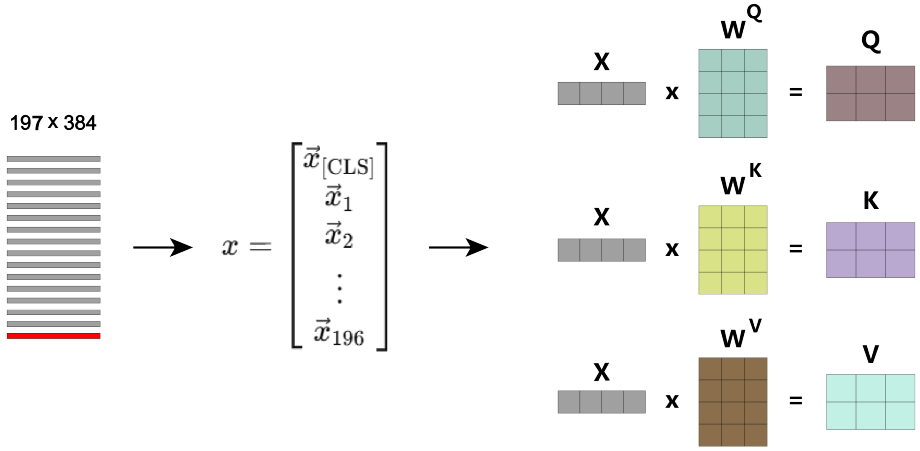

Figure 3.6: Query, Key and Value matrices computation.

- **Attention Computation**: The scaled dot-product attention is computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.1}$$

where d_k represents the key dimension.
- The attention mechanism computes a weighted sum of the values V based on the similarity between queries Q and keys K. As shown in Figure 3.7, the dot

products of Q and K are first calculated, optionally masked, then scaled by $\frac{1}{\sqrt{d_k}}$, and passed through a SoftMax to obtain the attention weights. These weights are then multiplied by V to produce the final attention output.

- **Multi-Head Processing**: The attention mechanism is applied in parallel across multiple heads ($h = 6$ for ViT-Small), allowing the model to attend to different representation subspaces.
- **Head Concatenation**: Outputs from all attention heads are concatenated and linearly projected back to the original dimension, as illustrated in Figure 3.8.

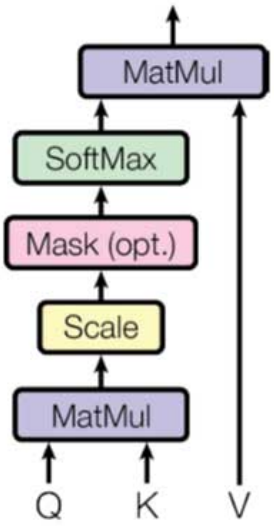

Figure 3.7: Scaled Dot-Product Attention [35].

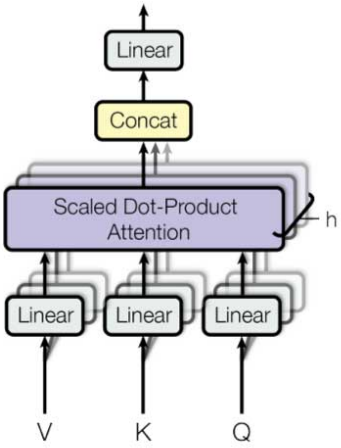

Figure 3.8: Multi-Head Attention [35].

3.3.7.2 Feed-Forward Network (MLP)

- **Two-Layer Structure**: The MLP consists of two linear transformations with a GELU activation function in between.
- **Dimension Expansion**: The first layer expands the feature dimension from 384 to 1536 (x 4 expansion ratio).
- **Non-Linear Activation**: GELU (Gaussian Error Linear Unit) activation introduces non-linearity: $\text{GELU}(x) = x \cdot \Phi(x)$, where $\Phi(x)$ is the cumulative distribution function of the standard normal distribution.
- **Dimension Reduction**: The second layer projects back to the original 384 dimensions.

3.3.7.3 Residual Connections and Normalization

- **Skip Connections**: Residual connections are applied around both the attention and MLP blocks, facilitating gradient flow during training.

- **Layer Normalization**: Pre-normalization is applied before each sub-layer, stabilizing training dynamics.
- **Mathematical Representation**: The overall transformer block can be expressed as:

$$z'_l = \text{MSA}(\text{LN}(z_{l-1})) + z_{l-1} \tag{3.2}$$
$$z_l = \text{MLP}(\text{LN}(z'_l)) + z'_l \tag{3.3}$$

where z_l represents the output of the l-th transformer layer.

3.3.8 Step 8: Feature Extraction and Aggregation

- After processing through all transformer layers, as shown in Figure 3.9, only the CLS token representation is extracted for classification.
- The final CLS token contains aggregated information from all patch tokens through self-attention interactions and effectively summarizes the entire leaf image's disease-relevant characteristics.
- **Feature Dimension**: The extracted feature maintains the 384-dimensional representation established throughout the transformer layers.

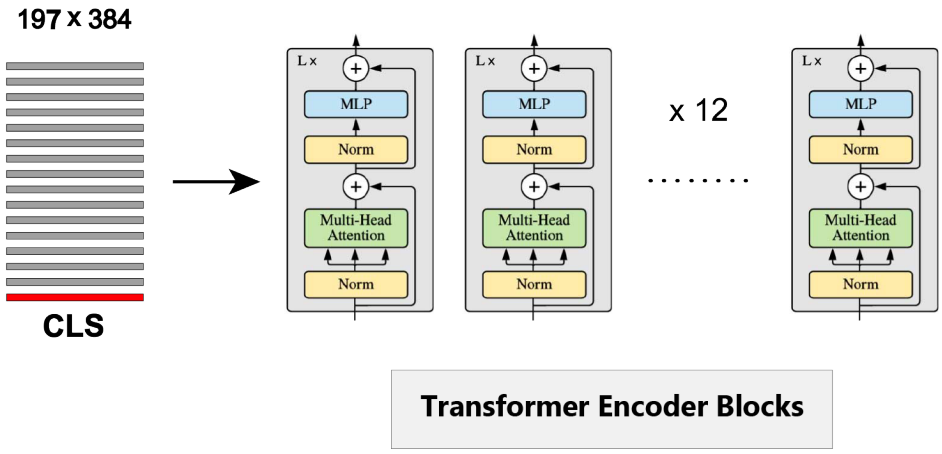

Figure 3.9: ViT-small : Transformers processing.

3.3.9 Step 9: Classification Head and Output Generation

- **Layer Normalization**: A final layer normalization is applied to the CLS token representation before classification.
- **Linear Classification**: A single linear layer maps the 384-dimensional CLS token to the number of disease severity classes.
- **Output Classes**: The model produces logits for each severity level.
- **Probability Distribution**: Softmax activation converts logits to probability distributions across severity classes, as illustrated in Figure 3.10.

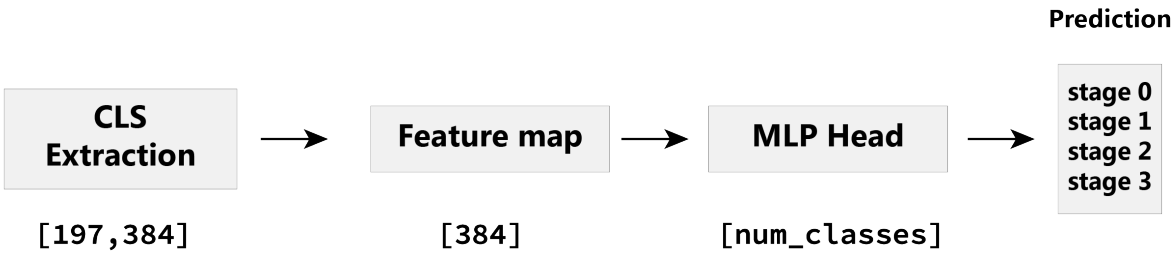

Figure 3.10: ViT-small : Final classification.

3.4 Swin-base

The Swin Transformer introduces a hierarchical architecture with local self-attention computed within non-overlapping windows, enabling linear computational complexity with respect to image size. The Swin-Base variant balances performance and model complexity by using a deeper architecture with increased embedding dimensions compared to smaller versions.

The main parameters of the Swin-Base model are summarized in Table 3.2. The model operates an input image of size $224 \times 224 \times 3$ which is first divided into non-overlapping patches of size 4×4, resulting in a feature map of shape $56 \times 56 \times 48$. Each 48-dimensional patch embedding is then projected to 128 dimensions using a linear layer, yielding a final output of shape $56 \times 56 \times 128$. The architecture is organized into four stages with a total of 24 transformer blocks distributed as (2, 2, 18, 2) per stage. The embedding dimension is doubled at each stage: 128, 256, 512, and 1024 respectively. Each block uses 4 or more attention heads depending on the stage. A dropout rate of 0.1 is applied to regularize the model. Overall, the Swin-Base model has approximately 88 million parameters. The final classification is performed by a linear head using the features from the last stage.

Table 3.2: Key Parameters of the Swin-Base Model.

Parameter	**Value**
Patch Size	4×4
Input Image Size	224×224
Initial Embedding Dimension	128
Number of Stages	4
Blocks per Stage	[2, 2, 18, 2]
Embedding Dimensions	[128, 256, 512, 1024]
Number of Attention Heads	[4, 8, 16, 32]
Number of Parameters	88 million
Classification Head	Linear

Swin Transformer architecture, as depicted in Figure 3.11, includes shifted window attention and hierarchical feature extraction, making it suitable for dense and fine-grained

classification tasks such as estimating disease severity in leaf images. The following provides a detailed step-by-step explanation of each architectural component:

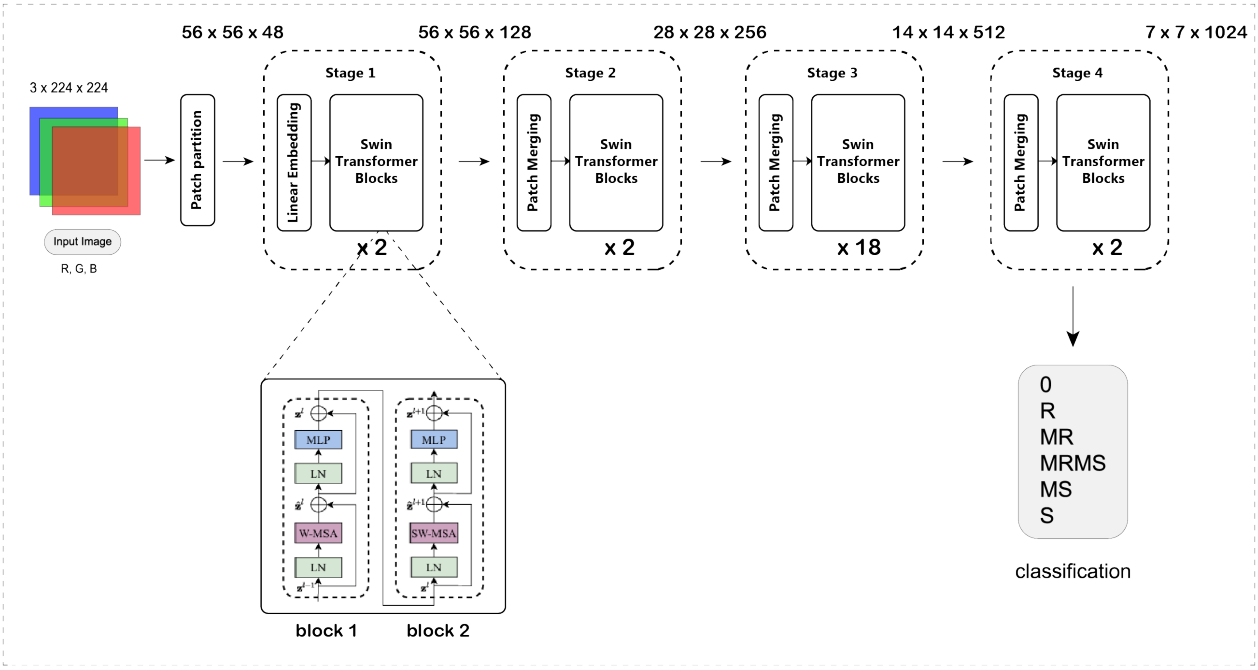

Figure 3.11: Architecture of the Swin-Base model.

3.4.1 Step 1: Input Image Processing and Patch Partition

- **Input Dimensions**: The architecture processes RGB input images of dimensions $224 \times 224 \times 3$, representing plant leaf samples with various disease severity levels.
- **Patch Partition**: Unlike ViT's larger patches, Swin Transformer divides the input image into smaller, non-overlapping patches of size 4×4 pixels, as illutrated in Figure 3.12.
- **Initial Patch Count**: This results in $\frac{224}{4} \times \frac{224}{4} = 56 \times 56 = 3136$ individual patches of size $4 \times 4 \times 3 = 48$.

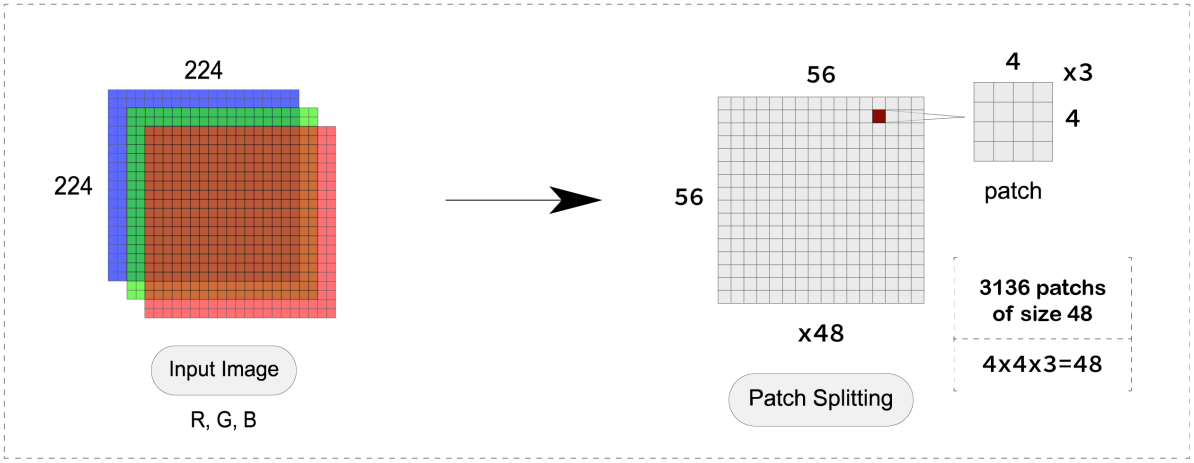

Figure 3.12: Step 1 : Patch Partition.

3.4.2 Step 2: Linear Embedding and Initial Feature Mapping

- **Patch Flattening**: Each $4 \times 4 \times 3 = 48$-dimensional patch is flattened to create the initial token representation, as shown in Figure 3.13.

Figure 3.13: Step 2 : a. Flatteninig.

- **Embedding Transformation**: Each 48-dimensional flattened patch undergoes linear projection to the initial feature dimension $C = 128$.
- **Feature Map Dimensions**: The resulting feature map has dimensions $56 \times 56 \times 128$, preserving spatial structure while creating learnable feature representations, as shown in Figure 3.14.

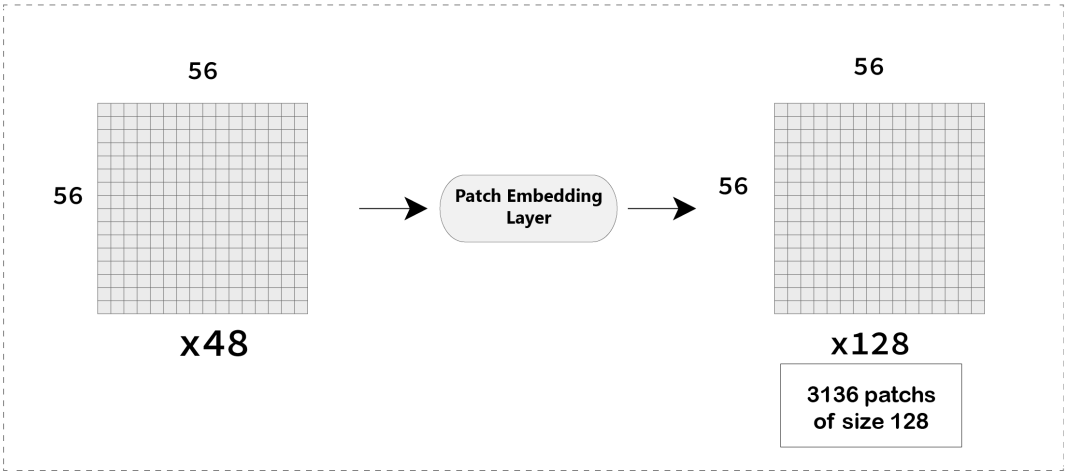

Figure 3.14: Step 2 : b. Linear Embedding.

3.4.3 Step 3: Stage 1 - Initial Feature Processing

a. Window Partitioning and Local Attention

- **Spatial Resolution**: Maintains 56×56 spatial resolution with 128-dimensional features per location.
- **Window Configuration**: The feature map is partitioned into non-overlapping windows of size $M \times M = 7 \times 7$.
- **Window Count**: Results in $\frac{56}{7} \times \frac{56}{7} = 8 \times 8 = 64$ windows, each containing $7 \times 7 = 49$ tokens, as illustrated in Figure 3.15.

- **Local Self-Attention**: Multi-head self-attention is computed within each 7×7 window independently.

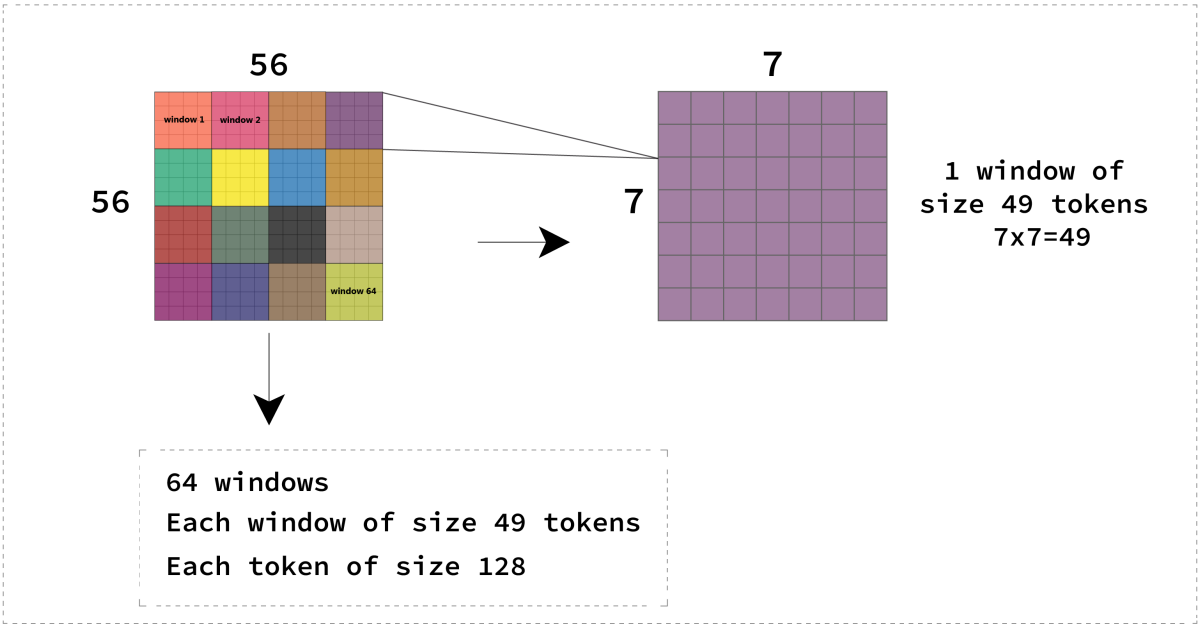

Figure 3.15: Step 3 : Windows partition.

b. Swin Transformer Blocks

- **Block Structure**: Stage 1 contains 2 consecutive Swin Transformer blocks, as shown in Figure 3.16.
- **W-MSA Block**: The first block employs Window-based Multi-head Self-Attention (W-MSA) with regular window partitioning.
- **SW-MSA Block**: The second block uses Shifted Window Multi-head Self-Attention (SW-MSA) with window partitions shifted by $(\lfloor \frac{M}{2} \rfloor, \lfloor \frac{M}{2} \rfloor) = (3, 3)$ pixels.
- **Attention Coputation**: Let the flattened input features in a window be denoted by $X \in \mathbb{R}^{M^2 \times C}$, where $C = 96$ is the embedding dimension at the first stage of Swin Base.
  - The input X is linearly projected into queries Q, keys K, and values V using learned weight matrices:

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V$$

    where $W_Q, W_K, W_V \in \mathbb{R}^{C \times d}$, and $d = \frac{C}{h}$ is the dimension per head, with $h = 3$ heads in the first stage (so $d = 32$).
  - For each head, self-attention is calculated independently within each window. The result is:

$$\text{Attention}(Q, K, V) = \text{SoftMax}\left(\frac{QK^T}{\sqrt{d}} + B\right)V \tag{3.4}$$

where $B \in \mathbb{R}^{M^2 \times M^2}$ is a learnable relative position bias that accounts for spatial relationships within each window.

- The outputs from all heads are concatenated and projected back to the original embedding dimension C via a final linear layer.

- Each block follows this pattern:
  1. **Normalize** the input data.
  2. **Apply attention** (either W-MSA or SW-MSA) to find important relationships.
  3. **Add** the result back to the original input (residual connection).
  4. **Normalize** again.
  5. **Apply MLP** (a small neural network) for further processing.
  6. **Add** this result back to the input from step 3.

- **Mathematical Formulation**: Each Swin block can be expressed as:

$$\hat{z}^l = \text{W-MSA}(\text{LN}(z^{l-1})) + z^{l-1} \tag{3.5}$$
$$z^l = \text{MLP}(\text{LN}(\hat{z}^l)) + \hat{z}^l \tag{3.6}$$
$$\hat{z}^{l+1} = \text{SW-MSA}(\text{LN}(z^l)) + z^l \tag{3.7}$$
$$z^{l+1} = \text{MLP}(\text{LN}(\hat{z}^{l+1})) + \hat{z}^{l+1} \tag{3.8}$$

where:

- LN$(\cdot)$ denotes Layer Normalization
- W-MSA$(\cdot)$ denotes Window-based Multi-head Self-Attention
- SW-MSA$(\cdot)$ denotes Shifted Window Multi-head Self-Attention
- MLP$(\cdot)$ denotes Multi-Layer Perceptron (feed-forward network)
- z^l represents the feature map at layer l
- $\hat{z}^l$ represents intermediate feature map after attention

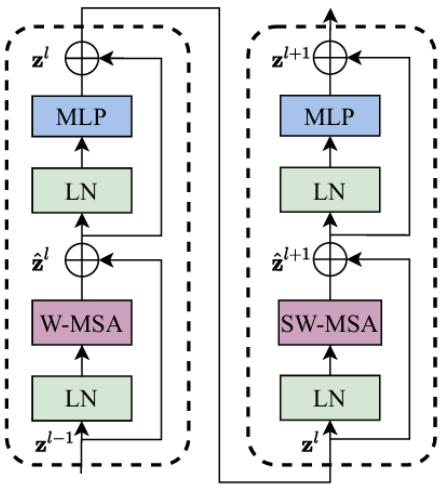

Figure 3.16: Two Successive Swin Transformer Blocks [38].

3.4.4 Step 4: Patch Merging and Stage 2 Processing

a. Patch Merging Operation

- **Spatial Downsampling**: Adjacent 2×2 patches are merged to reduce spatial resolution by a factor of 2.
- **Dimension Changes**: Resolution decreases from 56×56 to 28×28, while feature dimension increases from 128 to 256.

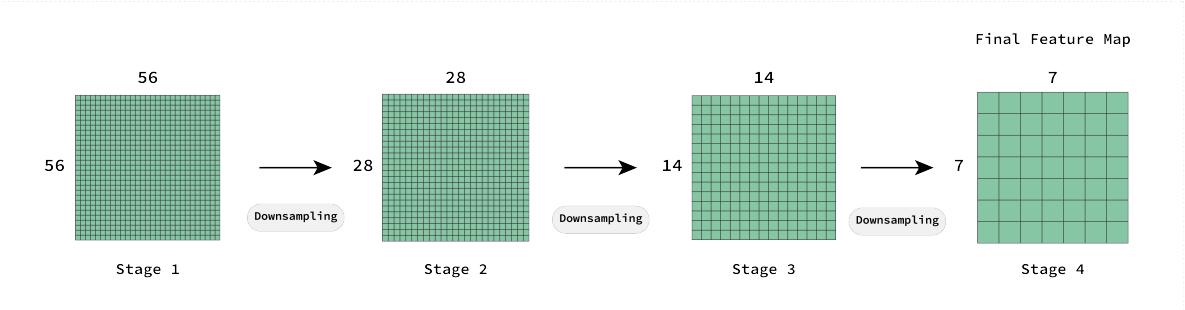

Figure 3.17: Step 4 : Patch Merging.

b. Stage 2 Swin Transformer Processing

- **Spatial Configuration**: Operates on 28×28 feature maps with 256-dimensional features.
- **Window Partitioning**: Creates $\frac{28}{7} \times \frac{28}{7} = 4 \times 4 = 16$ windows of size 7×7.
- **Block Count**: Contains 2 Swin Transformer blocks following the W-MSA and SW-MSA pattern.
- **Feature Enhancement**: Captures medium-range spatial dependencies and disease pattern relationships.

3.4.5 Step 5: Stage 3 - Deep Feature Extraction

a. Second Patch Merging

- **Resolution Reduction**: Further downsampling from 28×28 to 14×14 spatial resolution.
- **Feature Expansion**: Feature dimension doubles from 256 to 512 through the merging and projection process.

b. Intensive Processing Stage

- **Block Count**: Stage 3 contains 18 Swin Transformer blocks, making it the most computationally intensive stage.
- **Window Configuration**: Operates with $\frac{14}{7} \times \frac{14}{7} = 2 \times 2 = 4$ windows of size 7×7.

- **Deep Feature Learning**: The 18 blocks enable learning of complex, high-level disease pattern representations.
- **Critical Feature Extraction**: In this stage,the reduced spatial resolution and increased channel depth enable the extraction of high-level semantic features, effectively capturing global indicators of disease severity.

3.4.6 Step 6: Stage 4 - Final Feature Refinement

a. Final Patch Merging

- **Ultimate Downsampling**: Spatial resolution reduces from 14×14 to 7×7.
- **Maximum Feature Dimension**: Feature dimension reaches its maximum of 1024 dimensions.
- **Global Context**: The small spatial resolution combined with high feature dimension captures global disease severity indicators.

b. Final Processing Blocks

- **Block Configuration**: Stage 4 contains 2 Swin Transformer blocks for final feature refinement.
- **Single Window**: The entire 7×7 feature map is processed as a single window, enabling global attention computation.
- **High-Level Abstraction**: Focuses on overall disease severity patterns and global leaf health indicators.

3.4.7 Classification

As illustrated in Figure 3.18, before classification, a **Global Average Pooling**[1] operation is applied to the output from the last transformer stage using equation 3.9. This reduces the spatial dimensions, summarizing each feature map into a single value, resulting in a fixed-size vector of shape `[Batch, 1024]`.

$$\text{GAP}(x) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} x_{i,j} \tag{3.9}$$

Then, the final linear layer outputs 6 raw scores (logits), one for each disease severity class. These scores are converted into probabilities using the Softmax function, which ensures all outputs are between 0 and 1 and sum to 1.

The Softmax function is defined as:

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{6} e^{z_j}} \tag{3.10}$$

[1]GAP reduces a tensor of shape $[H, W, C]$ into a vector of shape $[C]$ by averaging all spatial positions over the height and width dimensions.

where z_i is the score for class i.

The class with the highest probability is selected as the predicted disease severity.

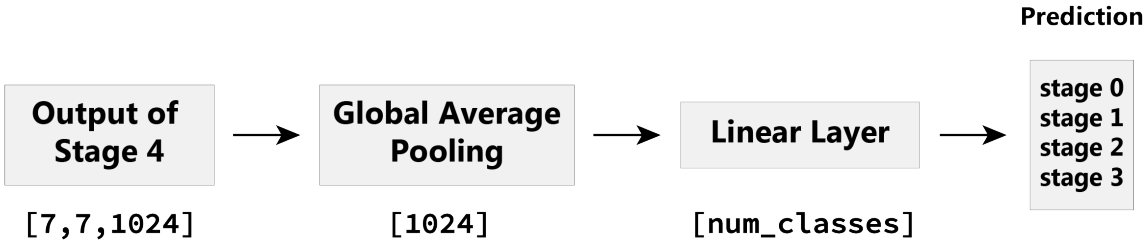

Figure 3.18: Swin base : Final classification.

3.4.8 Model Architectural Benefits

- **Hierarchical Design**: The progressive patch merging stages enable multi-scale feature learning, capturing both local details and global disease patterns.
- **Efficient Attention**: Shifted window self-attention balances computational efficiency and context modeling.
- Self-attention usually costs $O(N^2)$ because each patch attends to all others. Swin splits the image into M small windows of size $k \times k$, reducing computation to $O(M \times k^2)$. Shifting windows between layers allows efficient information flow with low cost.
- **Strong Representation**: High-dimensional embeddings and deep blocks allow modeling subtle severity differences.
- **Spatial Awareness**: The model preserves the spatial layout of features, which helps in accurately identifying the location and spread of plant diseases.

3.5 Conclusion

This chapter provided an in-depth description of two prominent vision transformer architectures: ViT-small and Swin-base. We began by introducing the detailed architecture of ViT-small, highlighting its patch embedding process, multi-head self-attention mechanism, and position embedding strategy.

After that, we present the Swin-base architecture, which introduces a hierarchical structure and shifted window-based self-attention. These innovations enable Swin-base to efficiently model both local and global dependencies while reducing computational complexity. The use of non-overlapping windows and their strategic shifting between layers allows the model to achieve better scalability and representation power, particularly for high-resolution vision tasks.

By understanding the inner workings and design philosophies of these two architectures, we establish a solid foundation for their comparative analysis and practical application. The insights gained here will be instrumental in the next chapter, where we present

the experimental setup used to evaluate and compare their performance across plant leaf disease severity estimation task.

Chapter 4

Experimental Framework

4.1 Introduction

This chapter defines the experimental framework for evaluating yellow rust severity on wheat leaves. It details the software tools, dataset properties, preprocessing methods, and evaluation metrics. The framework is built for reproducibility, robustness, and controlled testing, ensuring reliable model performance analysis. This setup forms the basis for presenting and discussing the results.

4.2 Experimental Setup : Software

4.2.1 Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python is extensively used in web development, data science, machine learning, artificial intelligence, automation, and scientific computing. It encourages modularity and code reuse through its support for modules and packages. It is freely available on all major platforms in both source and binary forms [41].

4.2.2 Visual Studio Code

Visual Studio Code is a cross-platform source code editor that supports hundreds of languages, IntelliSense code completion, debugging, and more. It is an open-source project that combines web, native, and language-specific technologies. It is free and available on any platform - Linux, macOS, and Windows [42].

4.2.3 PyTorch

PyTorch is an open-source machine learning library developed by Meta. It provides flexible tools for building and training deep learning models, with dynamic computation graphs and strong GPU acceleration. Widely used in both research and production, PyTorch supports a wide range of neural network architectures and is compatible with major platforms [43].

4.2.4 timm

timm (PyTorch Image Models) is an open-source library that compiles a diverse set of state-of-the-art computer vision models, layers, utilities, optimizers, schedulers, data-

loaders, augmentations and also training/validating scripts with ability to reproduce ImageNet training results [44].

4.3 Experimental Setup : Hardware

The training process was carried out on a high-performance computing setup featuring an NVIDIA RTX 4050 GPU with 32GB of VRAM. This hardware configuration provided the necessary computational resources to efficiently handle large-scale image datasets and the complexity of the models used. The ample memory and processing power enabled smooth and accelerated training, contributing to effective model convergence.

4.4 Dataset

The Yellow Rust 19 dataset [45] is used in this study to assess the severity of plant leaf diseases. It is a collection of 15 000 images with 6 severity stages. This dataset captures the visual symptoms of yellow rust a fungal infection caused by Puccinia striiformiswhich typically appears as yellow, powdery patches or stripes on the leaves and sheaths of wheat plants.

4.4.1 Dataset classes

As illustrated in Figure 4.1, the dataset categorizes images into 6 distinct classes based on the extent of rust infection observed on plant leaves :

- 0 Healthy
- R Resistant
- MR Moderately Resistant
- MRMS Moderately Resistant to Moderately Susceptible
- MS Moderately Susceptible
- S Susceptible

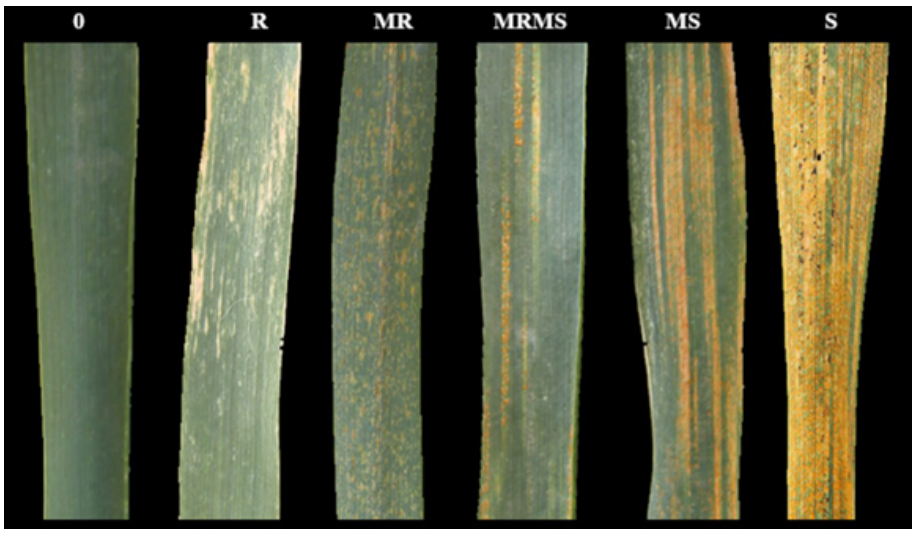

Figure 4.1: Severity levels images from the dataset [23].

Table 4.1: Severity levels of infection in the Yellow Rust 19 dataset.

Classes	Description	Number of Images
0	No signs of infection; healthy plant leaves.	2500
R	Minor signs of infection; plant exhibits resistance.	2500
MR	Small to medium signs of infection; partial resistance.	2500
MRMS	Transition between resistant and susceptible.	2500
MS	Medium signs of infection; moderate vulnerability.	2500
S	Major signs of infection; high susceptibility.	2500
Total		**15000**

Table 4.1 summarizes the distribution of image samples across six severity classes in the Yellow Rust 19 dataset, ranging from healthy leaves (class 0) to highly susceptible ones (class S), with each class containing an equal number of 2500 images.

4.4.2 Dataset division

To ensure robust experimental integrity, the dataset is partitioned into three subsets:

- **Training set (80%)**: Supports the model in learning characteristic features and patterns associated with yellow rust severity.
- **Validation set (10%)**: Aids in tuning hyperparameters and mitigates overfitting by offering an intermediate check during the training process.
- **Test set (10%)**: Provides an unbiased evaluation of the model's performance on unseen samples, ensuring reliability and generalizability.

Table 4.2 presents the structured distribution of images across the three subsets.

Table 4.2: Dataset split for Yellow Rust 19.

Subset	Percentage	Number of Images	Number per Class
Training Set	80%	12 000	2 000
Validation Set	10%	1 500	250
Test Set	10%	1 500	250

4.4.3 Data Preprocessing

The dataset preprocessing pipeline differs between the training phase and the validation/testing phases, with data augmentation techniques, as illustrated in Figure 4.2, applied only to the training data to enhance model generalization while keeping validation and testing conditions consistent.

On-the-fly augmentation dynamically applies stochastic transformations at runtime during training, leveraging GPU acceleration to increase data diversity efficiently without additional storage or preprocessing overhead.

4.4.3.1 Resizing

All images, regardless of their original dimensions, are resized to a fixed resolution of IMAGE_SIZE × IMAGE_SIZE pixels. This standardization ensures consistent input dimensions for the transformer model, which is essential for batch processing.

4.4.3.2 Training Data Augmentation

The following data augmentation techniques, as shown in Figure 4.2, were exclusively applied to the training dataset to artificially expand the diversity of training samples:

a. Random Horizontal Flip

This transformation randomly flips images horizontally with a 50% probability [as shown in Figure 4.2.c].

b. Random Vertical Flip

This transformation randomly flips images vertically with a 50% probability [as shown in Figure 4.2.d].

c. Random Rotation

Images are randomly rotated by an angle of up to 15 degrees [as shown in Figure 4.2.e].

d. Color Jitter

This transformation randomly adjusts three image properties [as shown in Figure 4.2.f] :

- Brightness: Random adjustment within a range of ±20%.
- Contrast: Random adjustment within a range of ±20%.

- Saturation: Random adjustment within a range of ±20%.

Color jitter helps the model become less sensitive to variations in lighting conditions and color distributions, increasing its ability to generalize across different environments and image capture conditions.

4.4.3.3 Normalization

Applied to both training and validation/test datasets:

b. Normalization

Images are normalized using the mean [0.485, 0.456, 0.406] and standard deviation [0.229, 0.224, 0.225] across the RGB channels. These values correspond to the statistics of the ImageNet dataset and are commonly used as a standard normalization practice. Normalization helps stabilize and accelerate the training process by ensuring that input features have similar scales.

4.4.3.4 Validation and Test Preprocessing

For validation and test datasets, only the essential preprocessing steps are applied without any data augmentation:

- Resizing to IMAGE_SIZE × IMAGE_SIZE.
- Normalization.

This simpler pipeline ensures that the model's performance is evaluated on clean, consistent data that represents the actual deployment conditions.

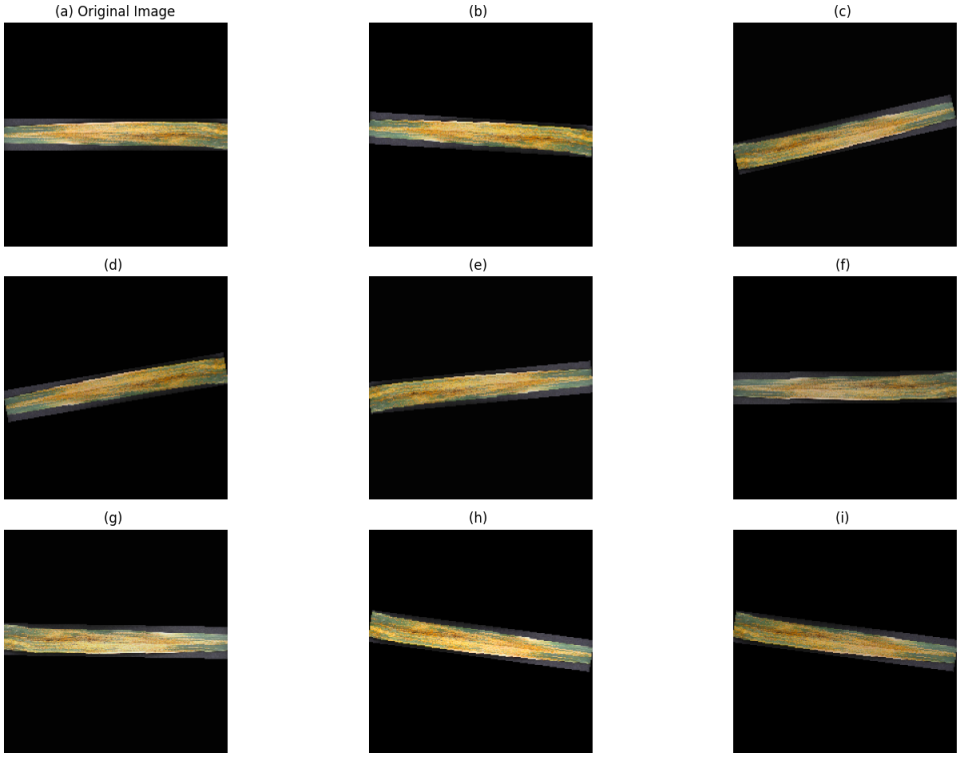

Figure 4.2: Data augmentation examples on yellow rust images: (a) original; (b) resized (224x224); (c) horizontal flip; (d) vertical flip; (e) ± 15° rotation; (f) color jitter (±20%).

4.5 Evaluation metrics

Evaluation metrics constitute quantitative measures implemented to assess the predictive capabilities of classification algorithms in machine learning. These metrics facilitate objective quantification of model performance through mathematical formulations that compare predicted outcomes against ground truth values. Many measures have been introduced in research, each addressing specific aspects of an algorithms performance. Consequently, for every machine learning problem, researchers need a suitable set of measures for performance assessment.

In this study, several common metrics were collected to obtain crucial information on the effectiveness of the algorithms used to estimate the disease severity. These metrics include precision, recall, F1-score, accuracy and the confusion matrix.

4.5.1 Confusion matrix

The confusion matrix represents a fundamental analytical tool in the evaluation of classification models, providing a comprehensive tabular representation of prediction outcomes compared against actual class labels. This matrix, as shown in Figure 4.3, constitutes the foundational structure from which numerous performance metrics are derived, offering multidimensional insights into model performance characteristics.

In the binary classification context, the confusion matrix is structured as a 2Œ2 contingency table that categorizes predictions into four distinct outcome categories: True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN).

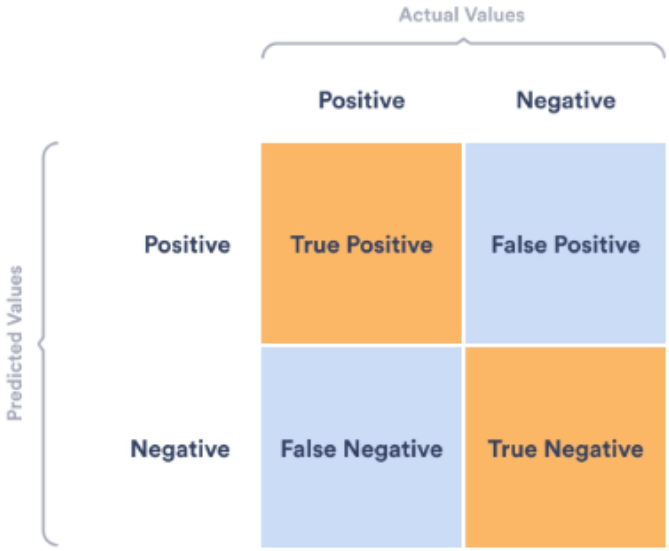

Figure 4.3: Confusion matrix [46].

- **True Positives (TP)**: Instances correctly predicted as positive when the actual class is positive.
- **False Negatives (FN)**: Instances incorrectly predicted as negative when the actual class is positive (Type II errors).

- **False Positives (FP)**: Instances incorrectly predicted as positive when the actual class is negative (Type I errors).
- **True Negatives (TN)**: Instances correctly predicted as negative when the actual class is negative.

4.5.2 Accuracy

Probably the most common and first way to evaluate an algorithms classification performance. It measures the percentage of successfully predicted data relative to the total number of observations in the dataset. It is mathematically expressed as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

4.5.3 Precision

Precision quantifies the proportion of correctly identified positive instances among all instances predicted as positive. This metric is particularly relevant in contexts where false positive predictions incur significant consequences. Its mathematical formulation is:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.2}$$

4.5.4 Recall (Sensitivity)

Recall, also termed sensitivity or true positive rate, measures the proportion of actual positive instances correctly identified by the model. Its mathematical representation is:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.3}$$

4.5.5 F1-score

The F1-score represents the harmonic mean of precision and recall, providing a balanced assessment when these metrics exhibit trade-off relationships. It is calculated as:

$$F1 - score = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.4}$$

4.6 Implementation Details

We implemented and trained all the models under the same conditions to ensure fair comparative analysis. The training was performed using the PyTorch framework as detailed in Table 4.6.

- Input images standardized to 224×224 pixels, with pixel values normalized using `Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])`.
- We adopted the AdamW optimizer with a base learning rate of 1×10^{-4}, weight decay 1×10^{-3} for L2 regularization, and 0.1 dropout probability to mitigate overfitting in

dense layers. Training proceeded for 60 epochs with a batch size of 32, optimized for memory efficiency on an NVIDIA RTX 4050 GPU (32GB VRAM) while maintaining gradient stability.

- A cosine annealing scheduler modulated the learning rate across training iterations, complemented by early stopping with a patience of 10 epochs based on validation accuracy. The CrossEntropyLoss function was used to optimize the model for 6-class severity estimation.

Table 4.3: Training Configuration.

Parameter	**Value**
Batch Size	32
Number of Epochs	60
Learning Rate	1×10^{-4}
Weight-decay	1×10^{-3}
Loss Function	CrossEntropyLoss
Optimizer	AdamW
Learning Rate Scheduler	CosineAnnealingLR
Early Stopping	max val-acc (patience = 10)
kernel	NVIDIA RTX 4070, 32GB VRAM

These hyperparameters were finalized through iterative ablation studies prioritizing generalization over training convergence speed.

4.7 Conclusion

This chapter detailed the entire experimental setup, including the software and hardware tools used, the dataset description, and the evaluation metrics applied to assess the models performance. These elements provide a solid foundation to ensure the rigor and reproducibility of the experiments.

In the following chapter, we will provide a detailed discussion and comprehensive presentation of the results obtained from the experimental evaluation of our proposed approach. This includes analysis, comparisons, and interpretations that highlight the effectiveness and relevance of our model in the context of the addressed problem.

Chapter 5

Experimental results

5.1 Introduction

This chapter presents the experimental results of our study on plant leaf disease severity estimation. Each model's performance is assessed using widely accepted classification metrics, including accuracy, precision, recall, F1-score, and the confusion matrix. These metrics provide a detailed understanding of the models strengths and weaknesses in handling the task. Building upon these, the proposed combined model was introduced, leveraging the strengths of both architectures through effective feature fusion and classification strategies. Furthermore, a comparative analysis is conducted to highlight the differences in performance, effectiveness, and computational efficiency among all tested architectures. In addition, the Combined Model is evaluated against state-of-the-art methods to assess its competitiveness in plant disease severity estimation.

This analysis provides insights into the strengths and limitations of the proposed method and serves as a foundation for further research and improvements in automated plant disease assessment.

5.2 ViT-small Model

In this section, we present the evaluation results of the ViT-Small model applied to the task of plant leaf disease severity classification. This model is based on the Vision Transformer (ViT) architecture, adapted for small-scale training with reduced parameters. The performance is assessed using standard classification metrics, model complexity measures, and confusion matrix analysis.

5.2.1 Model complexity

As shown in Table 5.1, the model has 21.66 million parameters and a relatively small footprint of 82.66 MB, making it suitable for deployment on devices with limited resources. The training time of 1.6 hours and inference time of 10 ms per image show that the model is efficient and scalable.

Table 5.1: Model Complexity and Training Details for ViT-Small model.

Metric	**Value**	**Unit**
Training Time	1.6	hours
Model Size	82.66	MB
Number of Parameters	21.66	million
Inference Time	100	mS

5.2.2 Classification Report

The ViT-Small model demonstrates **moderate performance** with a test accuracy of **61.33%** and a test loss of **0.8978**. The classification report is presented in Table 5.2.

Table 5.2: Classification Report for the ViT-small model.

Class	Precision	Recall	F1-score	Support
0	0.64	0.78	0.70	250
R	0.56	0.58	0.57	250
MR	0.53	0.37	0.44	250
MRMS	0.59	0.58	0.59	250
MS	0.56	0.59	0.58	250
S	0.76	0.78	0.77	250
Test Accuracy		61.33%		
Test Loss		0.8978		

– The model achieves its best performance on class **S (Susceptible)**, with high precision (0.76), recall (0.78), and F1-score (0.77), indicating that it correctly identifies this class in most cases. Class **0 (Healthy)** also performs relatively well, particularly in recall (0.78), though its lower precision (0.64) suggests some confusion with other classes.

– In contrast, the model struggles significantly with classes **MR(Moderate Resistant)** and **MS(Moderate Susceptible)**. Class **MR** has a low recall (0.37), meaning many true MR samples are misclassified. Its precision (0.53) is also suboptimal, reflecting frequent false positives. The confusion matrix (Figure 5.1) confirms that MR is often misclassified as class **0** or **R(Resistant)**, which highlights a clear limitation of the model in distinguishing this class.

– Class **MRMS(Moderate Resistant Moderate Susceptible)** shows average results, with precision (0.59) and recall (0.58) indicating moderate performance. Class **R** also achieves consistent but unimpressive scores, with all metrics in the 0.560.58 range, reflecting limited class separability.

5.2.3 The Confusion Matrix

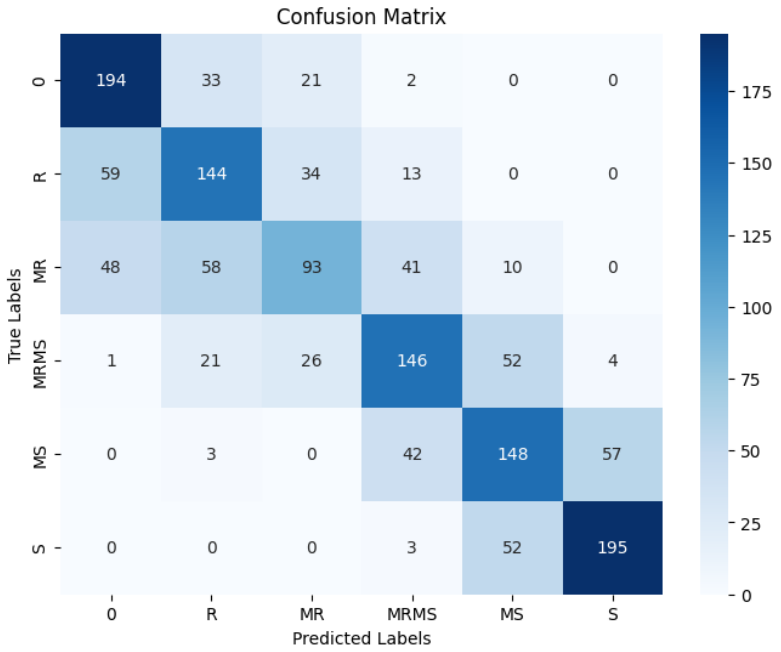

Figure 5.1: Confusion Matrix for ViT-small.

The confusion matrix in Figure 5.1 highlights several key points. Notably:

- A significant number of **MR** instances are misclassified as class **0** (48 samples) and **R** (58 samples).
- Class **MRMS** has confusion with **MS** (52 samples) and **MR** (26 samples), indicating potential feature overlap.
- Class **S** is well separated from the others, with 195 correctly classified samples and very few confusions.

5.2.4 Discussion and Analysis

The ViT-Small model achieved varying performance across the different classes, which can be explained by analyzing both the characteristics of the data and the model architecture.

Strengths: The model performs best on class **S**, with high precision (0.76) and recall (0.78). This strong performance can be attributed to the clear and pronounced visual symptoms associated with susceptible leaves, such as widespread necrosis and visible lesions. These global patterns are effectively captured by the self-attention mechanism of the Vision Transformer, which excels at modeling long-range dependencies and large-scale spatial structures. Similarly, class **0 (Healthy)** shows good recall (0.78), likely due to the distinct absence of disease symptoms, which provides a strong contrast compared to infected classes.

Weaknesses: In contrast, the model struggles with classes **MR** and **MRMS**. These classes tend to exhibit subtle or mixed visual features that are often difficult to distinguish from other severity levels. The confusion matrix shows that MR is frequently misclassified as **0 (Healthy)** and **R**, indicating that the model confuses mild symptoms with either no symptoms or more resistant appearances. This is attributed to the ViT-Small models limited ability to capture fine-grained local features, as its patch-based input representation often overlooks subtle yet important small-scale details.

Architectural limitations: The ViT-Small model, with its reduced depth and number of parameters, is efficient but less capable of modeling complex or subtle visual cues. While it captures global structures well, it may overlook local variations critical for distinguishing intermediate severity levels. Additionally, Vision Transformers generally benefit from large training datasets. Since our dataset contains ambiguous labels, the model's ability to learn robust discriminative features is further limited.

Interpretation: These results suggest that the ViT-Small model is effective in classifying clearly defined classes but lacks the resolution and representational power required to differentiate between overlapping or less distinct categories. Improving the performance on these classes may require either a larger dataset or a hybrid architecture that integrates both local and global feature extractors, such as combining ViT with another transformer-based model such as Swin.

5.3 Swin-base Model

In this section, we analyze the performance of the Swin Base model on the task of plant leaf disease severity classification. This model utilizes a hierarchical Transformer architecture with shifted windows, as detailed in chapter three, enabling strong performance across image recognition tasks.

5.3.1 Model complexity

As shown in Table 5.3, the Swin Base model has a significantly higher parameter count than ViT-Small, totaling 86.75 million parameters and a model size of 330.92 MB. Its training time is slightly higher (1.8 hours), and inference time remains efficient with 10 ms per image.

Table 5.3: Model Complexity and Evaluation Details for the Swin Base model.

Metric	**Value**	**Unit**
Training Time	1.8	hours
Model Size	330.92	MB
Number of Parameters	86.749	million
Inference Time	100	mS

5.3.2 Classification Report

The Swin Base model achieves a test accuracy of **88.07%**, substantially outperforming ViT-Small, with a test loss of **0.9643**. The detailed classification metrics are provided in Table 5.4.

Table 5.4: Classification Report for the Swin Base model.

Class	Precision	Recall	F1-score	Support
0	0.94	0.98	0.96	250
R	0.92	0.95	0.93	250
MR	0.91	0.86	0.88	250
MRMS	0.86	0.83	0.84	250
MS	0.79	0.76	0.77	250
S	0.86	0.91	0.88	250
Test Accuracy	88.07%			
Test Loss	0.9643			

- The model performs best on class **0**, achieving an exceptional recall of 0.98 and an F1-score of 0.96, indicating that it can reliably identify healthy samples.
- Classes **R**, **MR**, and **S** also exhibit strong results, each exceeding 0.85 in precision and recall, confirming robust separability and low confusion.
- Although slightly weaker, classes **MS** and **MRMS** still demonstrate solid performance with F1-scores above 0.77. This reflects the models capacity to distinguish fine-grained severity levels more effectively than ViT-Small.

5.3.3 The Confusion Matrix

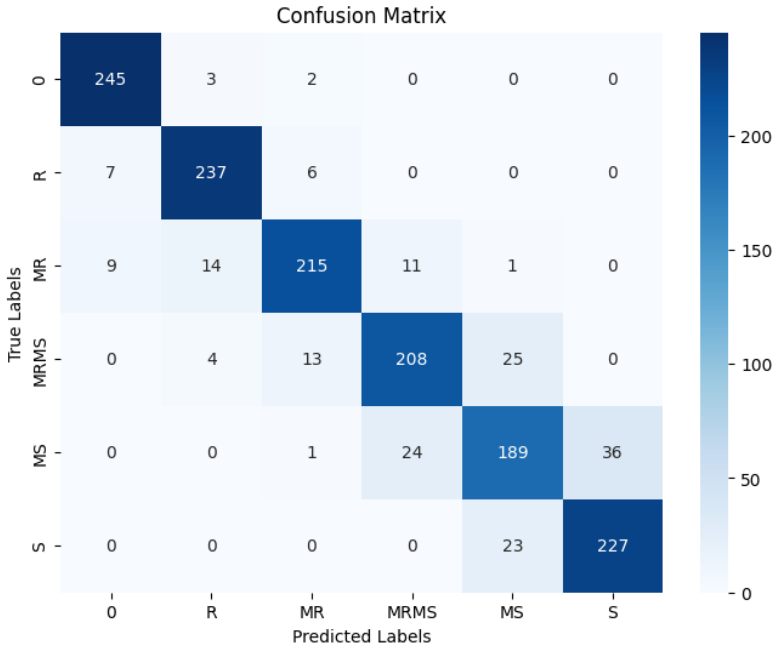

Figure 5.2: Confusion Matrix for the Swin Base model

The confusion matrix in Figure 5.2 reveals that:

- Class **0** is very well predicted with only a handful of misclassifications (5 samples misclassified).
- Class **MR** shows some confusion, notably with **MRMS** (11 instances) and **R** (14 instances), but still maintains high recall (0.86).
- Class **MRMS** is occasionally confused with **MS** (25 cases) and **MR** (13 cases), indicating moderate overlap.
- Class **MS** is misclassified as **MRMS** (24 cases) and **S** (36 cases), suggesting a need for better spatial resolution in those classes.
- Class **S** is clearly separated with 227 correct predictions and few confusions, primarily with **MS**.

5.3.4 Discussion and Analysis

The Swin Base model achieved consistently strong performance across all classes, with notable improvements compared to ViT-Small. These results can be interpreted by considering both the data characteristics and the strengths of the Swin architecture.

Strengths: The model performs best on class **0 (Healthy)**, with an outstanding recall of 0.98 and F1-score of 0.96. This is due to the distinct and uniform appearance of healthy leaves, which contrasts sharply with any form of disease. The hierarchical structure of Swin and its ability to process information at multiple scales allow it to accurately identify these clean patterns. Similarly, the model excels on classes **R (Resistant)** and **S**

(Susceptible), both showing high precision and recall above 0.85. The visual symptoms in these classes are typically well-defined (e.g., strong discoloration or lesion patterns), which the shifted window attention captures effectively by balancing local and global context.

Weaknesses: The model struggles more with intermediate severity classes such as **MR**, **MRMS**, and **MS**. These categories often present overlapping or ambiguous visual features, leading to confusion between them. For example, MR is frequently misclassified as MRMS and R, while MS is confused with MRMS and S. These misclassifications suggest that, although Swin captures multi-scale features, subtle visual differences between intermediate severity levels are still challenging to distinguish, possibly due to shared texture patterns or disease progression stages.

Architectural limitations: While the Swin Base model improves over ViT-Small by introducing locality through shifted windows and hierarchical representation, it may still have limitations in detecting very fine-grained variations, especially in densely textured regions. Additionally, the presence of visual similarity across MR, MRMS, and MS can create decision boundaries that are difficult to learn.

Interpretation: These results indicate that the Swin Base model is highly capable of identifying classes with distinct and consistent visual traits, leveraging its architectural advantages in multi-scale feature extraction. However, its performance on overlapping or borderline categories is constrained by the inherent visual complexity of those classes. Addressing this could involve combining it with a Vision Transformer (ViT) to jointly exploit Swin's hierarchical locality and ViT's strong global attention capabilities. Such a hybrid model may offer improved discrimination between closely related severity levels by capturing both fine-grained and long-range dependencies more effectively.

5.4 Enhancing model accuracy : Combined model

The proposed combined model architecture integrates two complementary transformer-based vision models to leverage their distinct feature extraction capabilities. As illustrated in Figure 5.3, the architecture consists of parallel feature extraction using Vision Transformer (ViT-small) and Swin Transformer (Swin-base), followed by feature concatenation and classification.

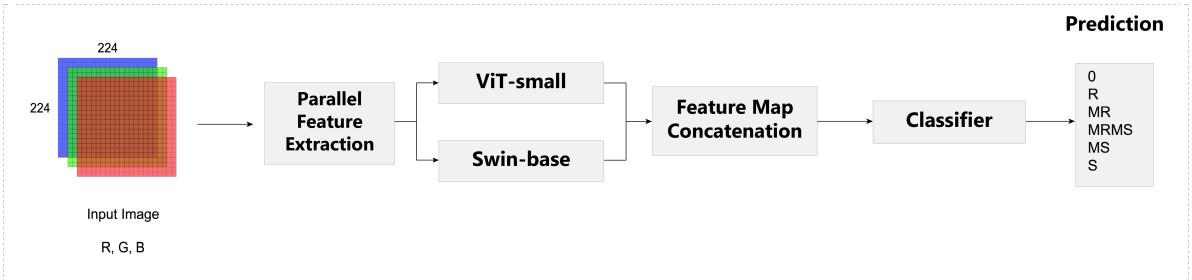

Figure 5.3: Architecture of the proposed combined model.

The input RGB image (224 x 224) undergoes parallel feature extraction through both networks. ViT-small captures global relationships through self-attention mechanisms, producing 384-dimensional features, while Swin-base employs hierarchical windowed attention to extract local-to-global representations, yielding 1024 x 7 x 7 dimensional features. These features are then flattened and concatenated to form a comprehensive feature vector that combines both global and hierarchical spatial information. Finally, a two-layer MLP classifier with ReLU activation and dropout regularization maps the combined features to the final predictions across 6 classes (0, R, MR, MRMS, MS, S).

5.4.1 Architecture Overview

The input RGB image with dimensions 224×224 is processed simultaneously by both transformer networks:

- **ViT-small**: Extracts global contextual features through self-attention mechanisms, producing a 384-dimensional feature vector that captures long-range dependencies across the entire image.
- **Swin-base**: Employs hierarchical windowed self-attention to extract multi-scale features, generating $1024 \times 7 \times 7$ dimensional features that preserve both local and global spatial information.

5.4.2 Feature Fusion

1. **Feature Extraction**: Both models extract features without their original classification heads (num_classes=0).
2. **Feature Flattening**: Swin transformer features are flattened from $(1024, 7, 7)$ to $(1024 \times 7 \times 7) = 50176$ to ensure compatibility with ViT features.
3. **Feature Concatenation**: The ViT features (384-dim) and flattened Swin features (50176-dim) are concatenated to form a comprehensive feature vector of dimension 50560.

5.4.3 Classification

A two-layer MLP classifier with the following structure:

$$h_1 = \text{ReLU}(W_1 \cdot f_{combined} + b_1) \tag{5.1}$$
$$h_2 = \text{Dropout}(h_1, p = 0.1) \tag{5.2}$$
$$y = W_2 \cdot h_2 + b_2 \tag{5.3}$$

where:

- $f_{combined} \in \mathbb{R}^{50560}$ is the concatenated feature vector
- $W_1 \in \mathbb{R}^{512 \times 50560}$ and $b_1 \in \mathbb{R}^{512}$ are the weights and bias of the first linear layer
- $h_1 \in \mathbb{R}^{512}$ is the output of the first hidden layer after ReLU(Rectified Linear Unit) activation

- $h_2 \in \mathbb{R}^{512}$ is the output after applying dropout regularization
- $W_2 \in \mathbb{R}^{6\times512}$ and $b_2 \in \mathbb{R}^6$ are the weights and bias of the second linear layer
- $y \in \mathbb{R}^6$ represents the final class logits for the 6 target classes

5.5 Proposed combined model results

In this section, we analyze the performance of the proposed combined model on the task of plant leaf disease severity classification. The architecture combines the strengths of ViT Small and Swin Base, enabling it to effectively capture both global and local features. This integration allows the model to deliver improved classification accuracy across all severity levels, demonstrating its effectiveness in handling complex visual patterns associated with plant diseases.

5.5.1 Model complexity

As shown in Table 5.5, the proposed combined model has substantially higher computational requirements compared to the Swin Base model, with 134.299 million parameters and a model size of 512.31 MB. Despite the increased complexity, the model maintains efficient inference time at 10 ms per image, while requiring 3 hours for training.

Table 5.5: Model Complexity and Evaluation Details for the Combined model.

Metric	**Value**	**Unit**
Training Time	3	hours
Model Size	512.31	MB
Number of Parameters	134.299	million
Inference Time	100	mS

5.5.2 Classification Report

The proposed combined model achieves exceptional performance with a test accuracy of **94.67%**, representing a substantial 33.34% improvement over ViT base model (61.33%) and 6.6 % improvement over the Swin Base model (88.07%). The test loss is significantly reduced to **0.5324**, demonstrating improved model confidence and reduced prediction uncertainty. The detailed classification metrics are provided in Table 5.6.

Table 5.6: Classification Report for the Combined model.

Class	Precision	Recall	F1-score	Support
0	0.99	1.00	1.00	250
R	0.99	1.00	0.99	250
MR	0.98	0.97	0.98	250
MRMS	0.91	0.94	0.92	250
MS	0.89	0.84	0.86	250
S	0.91	0.93	0.92	250
Test Accuracy	94.67%			
Test Loss	0.5324			

- The model achieves perfect performance on class **0** (healthy samples), with exceptional precision (0.99), perfect recall (1.00), and perfect F1-score (1.00), indicating flawless identification of healthy plant samples.
- Class **R** demonstrates near-perfect performance with 0.99 precision, perfect recall (1.00), and 0.99 F1-score, showing excellent separation of severely diseased samples.
- Class **MR** exhibits outstanding performance with 0.98 precision, 0.97 recall, and 0.98 F1-score, representing a significant improvement over previous models.
- Classes **MRMS** and **S** show strong and balanced performance, both achieving F1-scores of 0.92, indicating robust classification of intermediate severity levels.
- Although class **MS** shows the lowest performance, it still achieves solid results with an F1-score of 0.86, representing substantial improvement over the Swin Base model's 0.77.

5.5.3 The Confusion Matrix

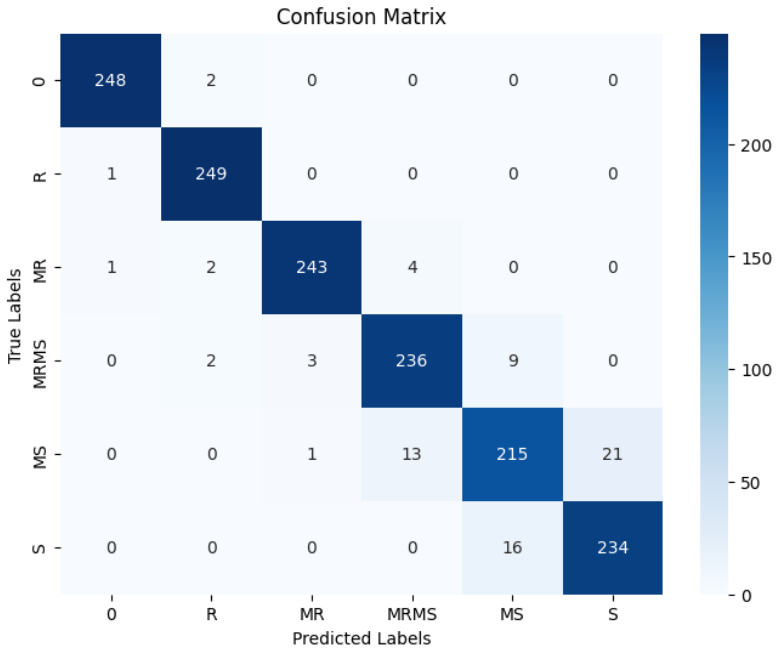

Figure 5.4: Confusion Matrix for the Combined model.

The confusion matrix in Figure 5.4 reveals remarkable classification precision with minimal misclassifications:

- Class **0** achieves near perfect classification with 248 samples correctly identified and only 2 misclassifications, demonstrating the model's exceptional ability to distinguish healthy samples.
- Class **MR** shows excellent performance with 243 correct predictions out of 250, with minimal confusion primarily with **MRMS** (4 instances) and **R** (2 instances), indicating precise boundary detection.
- Class **MRMS** demonstrates strong classification with 236 correct predictions, showing limited confusion with **MS** (9 instances) and **MR** (3 instances), representing significant improvement in distinguishing intermediate severity levels.
- Class **MS** exhibits the most challenging classification pattern with 215 correct predictions, primarily confused with **S** (21 instances) and **MRMS** (13 instances), though still maintaining acceptable performance levels.
- Class **R** achieves near perfect classification with 249 samples correctly identified, demonstrating exceptional capability in recognizing severe disease manifestations.
- Class **S** shows strong performance with 234 correct predictions out of 250, with minimal confusion limited to **MS** (16 instances), indicating clear differentiation of severe disease states.

5.5.4 Interpretation

The results demonstrate that the proposed combined model significantly enhances classification performance by effectively capturing both global and local contextual information. This dual-stream architecture mitigates the limitations observed in the Swin-Base model, particularly in classifying intermediate severity levels such as MS and MRMS. The improved F1-scores across all classes, especially in challenging cases, suggest that fusing ViT and Swin features enables a richer and more discriminative representation.

5.6 Strengths and Limitations of the Proposed Model

5.6.1 Strengths

- **Superior Accuracy and Reduced Loss:**
  - The proposed combined model achieves an impressive test accuracy of 94.67% with a low test loss (0.5324).
  - This performance is a notable improvement compared to the individual ViT-Small and Swin-Base models.
  - Such results indicate that the combined approach optimally captures disease severity characteristics in the yellow rust dataset.
- **Complementary Feature Fusion:**
  - By integrating the global attention capabilities of ViT-small with the local and hierarchical feature extraction of Swin-base, the model effectively captures both broad contextual information and fine-grained spatial details.
  - This dual approach leads to enhanced feature representation and enables near-perfect discrimination for classes that are distinct (e.g., the healthy class 0 and the severely diseased class S).
- **Robust Class Performance:**
  - The classification report shows that classes such as 0 (healthy) and R (Resistant) are almost perfectly classified with almost perfect precision and recall demonstrating the models strong ability to accurately identify and distinguish clearly defined cases.
- **Effective Handling of Complex Visual Patterns:**
  - The fusion of features from both transformer architectures allows the model to better handle the inherent ambiguities present in plant leaf images.
  - The combined representation provides a comprehensive view, enhancing the models robustness in distinguishing subtle variations across disease severity stages.

5.6.2 Limitations

- **High Computational Complexity:**
  - The proposed combined model has 134.299 million parameters and a model size of 512.31 MB.
  - Such complexity entails a higher computational cost during training (which takes around 3 hours).
  - This could limit deployment in resource-constrained or real-time environments
- **Residual Ambiguity in Intermediate Classes:**
  - Although overall performance is strong, the confusion matrix reveals that classes with subtle differences, and particularly the intermediate categories (MS and MRMS), still suffer from some misclassification.
  - There remains some ambiguity in distinguishing these borderline classes, which may be due to overlapping visual features.
- **Potential Scalability Issues:**
  - The concatenation of the high-dimensional feature vectors (with the Swin feature map flattened to over 50,000 dimensions and then combined with the 384-dimensional ViT features) increases the memory footprint.
  - This complexity might pose challenges for scalability or deployment on devices with limited memory or processing power.
- **Training Overhead:**
  - The increased model complexity means a longer training time and possibly greater susceptibility to overfitting if not properly regularized.

In summary, the proposed combined model is remarkably effective, leveraging the complementary strengths of global and local attention mechanisms to achieve high accuracy and robust performance for plant disease severity estimation. However, its benefits come at the cost of increased computational demands and some challenges in resolving classification boundaries among visually similar intermediate classes.

5.7 Models comparison

The table 5.7 represents the comparison between three models : ViT-Small, Swin-Base, and the proposed combined model, in terms of classification test accuracy, number of parameters, and model size.

Table 5.7: Models accuracy comparison.

Model	Accuracy (%)	Num. of paras (M)	Model size (MB)
ViT-small	61.33	21.66	82.66
Swin-base	88.07	86.749	330.92
Combined Model	**94.67**	134.299	512.31

Table 5.7 and Figure 5.5 clearly highlight the superior performance of the proposed combined model compared to ViT-small and Swin-base models.

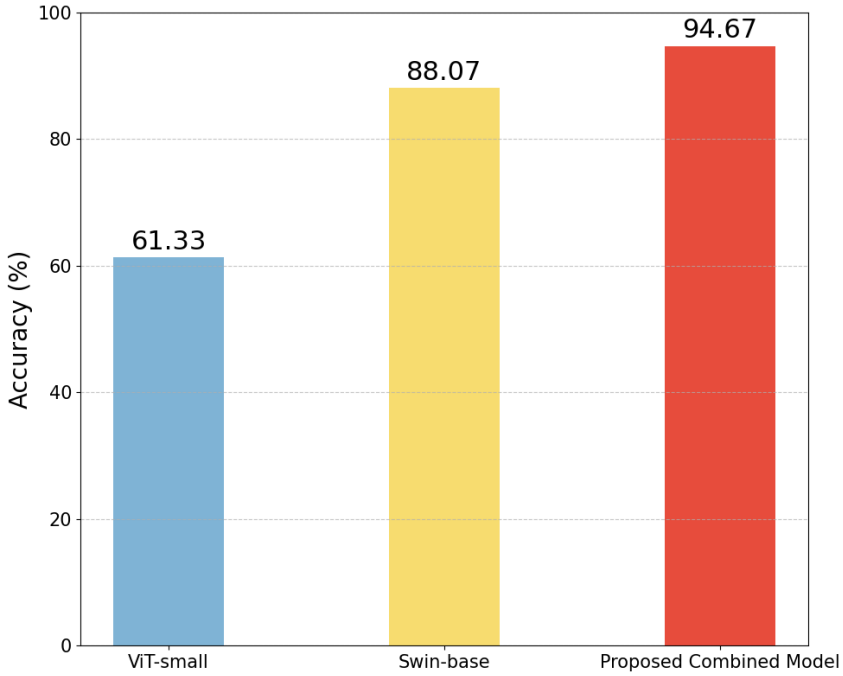

Figure 5.5: Models accuracy comparison.

The ViT-Small model, relying on pure global self-attention, effectively captures long-range dependencies but exhibits limitations in discerning fine-grained local features due to its uniform patch processing. In contrast, the Swin-Base architecture utilizes a hierarchical shifted-window mechanism, efficiently extracting multi-scale local features with inherent translation invariance and reduced computational complexity, though potentially at the cost of some global context. Consequently, a combined ensemble model integrating both architectures consistently achieves superior accuracy, as shown in Figure 5.5 compared to either model alone. This enhancement arises from their complementary strengths: ViT-Small provides robust global structural understanding, while Swin-Base excels at modeling local textures and details, leading to diverse error profiles. Their fusion mitigates individual weaknesses, creates more comprehensive representations, and significantly boosts robustness.

Comparison using F1-score

To go deeper in our comparison, we use the **F1-score** metric, because that provides a balanced measure between *precision* and *recall*, which is especially important in multi-class classification tasks. Unlike accuracy, which can be misleading when some classes are easier to predict than others, the F1-score better reflects the model's ability to correctly identify each class without favoring the dominant ones.

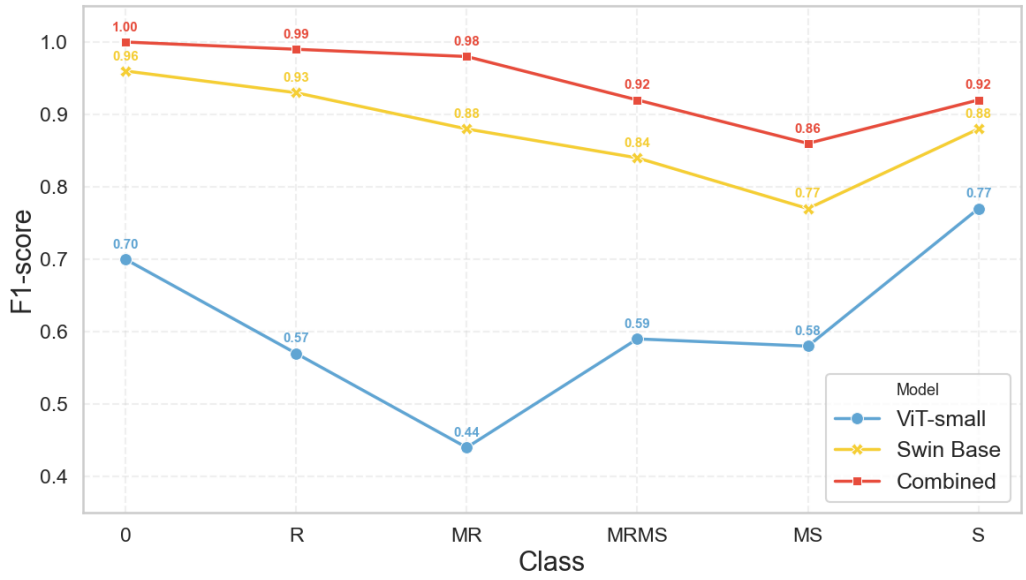

Figure 5.6: F1-score per class for each model.

Figure 5.6 illustrates the F1-scores obtained by the three models **ViT-small**, **Swin Base**, and the **Combined model** for each of the six plant leaf disease severity classes: `0 (Healthy)`, `R (Resistant)`, `MR`, `MRMS`, `MS`, and `S (Susceptible)`.

The Combined model consistently outperforms the others, achieving near-perfect F1-scores across all classes. The Swin Base model also performs well, particularly in classes `0` and `R`, but shows a slight drop in class `MS`. In contrast, the ViT-small model exhibits more variability, with its lowest performance in class `MR` (F1-score of 0.44), indicating difficulties in recognizing certain intermediate severity levels.

This comparison highlights the robustness and generalization capability of the proposed combined architecture in handling nuanced disease severity classification.

5.8 Comparison with State-of-the-Art Models

Table 5.8 presents the performance comparison of our proposed combined model with existing state-of-the-art approaches for Wheat Yellow Rust disease severity estimation.

Table 5.8: Performance Comparison with State-of-the-Art Models.

Model	Architecture	Dataset	Accuracy (%)
Proposed Model	ViT-Small + Swin-Base	Yellow Rust 19	**94.67**
Yellow-Rust-Xception [23]	CNN-based (Xception)	Yellow Rust 19	91.00
C-DenseNet [24]	DenseNet + CBAM	WSRgrading	97.99

5.8.1 Comparison with Yellow-Rust-Xception Model

Our proposed combined model significantly outperforms the Yellow-Rust-Xception model by Hayit et al. [23] across multiple aspects, as shown in Table 5.9.

Our model achieves 3.67% higher accuracy while eliminating the need for manual preprocessing steps. The transformer-based architecture captures both global and local features more effectively than traditional CNN approaches.

Table 5.9: Detailed Comparison with Yellow-Rust-Xception Model.

Aspect	Our Model	Yellow-Rust-Xception
Accuracy (%)	**94.67**	91.00
Preprocessing	Not required	Manual segmentation
Architecture	Transformer-based	CNN-based
Implementation	End-to-end	Multi-step process

- A major advantage of our transformer-based approach is its ability to **process raw field images directly**, without requiring the complex preprocessing steps that other models depend on.
- In contrast, models like the **Yellow-Rust-Xception** demand a **four-stage preprocessing workflow**, which includes:
  - **Thresholding** to isolate relevant image regions,
  - **Morphological transformations** to refine structures and shapes,
  - **Masking** to emphasize diseased areas,
  - And **format conversion**, all of which require **technical expertise** and **specialized software**.
- Our model avoids these requirements entirely by supporting **end-to-end processing of original images**, while still achieving **superior accuracy (94.67% compared to 91%)**, showing its effectiveness even without preprocessing.
- The **elimination of preprocessing** has a transformative impact on **practical deployment**:
  - Farmers can simply **capture images with standard smartphones**,

  - And **receive immediate assessments of disease severity** without needing any technical skills or spending time on manual image preparation.

- This ability to directly handle raw images makes the model:
  - **More user-friendly**, especially for non-technical users in the agricultural sector,
  - **Faster to deploy in real-time settings**,
  - And **less prone to errors** that may arise from multi-step preprocessing chains.

- Overall, these qualities make our approach **far more suitable for real-world agricultural use**, where **simplicity, speed, and accessibility** are essential.

5.8.2 Comparison with C-DenseNet Model

Although C-DenseNet [24] reports higher accuracy (97.99%), our model demonstrates superior practical value due to dataset characteristics shown in Table 5.10.

Table 5.10: Dataset Comparison.

Characteristic	**Our Model**	**C-DenseNet**
Dataset size	15,000 images	5,242 images
Data source	Multi-device field images	Single-location images
Class distribution	Balanced	Imbalanced

Our model achieves 94.67% accuracy on a dataset three times larger and more diverse than C-DenseNet's dataset. This demonstrates better generalization capability for real-world agricultural applications.

5.8.3 Summary

Our proposed combined model offers the best combination of accuracy and practical applicability:

- **Higher accuracy**: 3.67% improvement over Yellow-Rust-Xception
- **Automated processing**: No manual preprocessing required.
- **Robust performance**: Effective on diverse, large-scale datasets.
- **Modern architecture**: Transformer-based design for better scalability.

These advantages make our model the most suitable solution for practical deployment in smart agriculture systems.

5.9 Conclusion

In conclusion, this chapter demonstrates that combining the strengths of the ViT-small and Swin-base models leads to a significant improvement in plant leaf disease severity estimation. While ViT-small alone had moderate performance and Swin-base delivered robust metrics, their integration raised the overall accuracy to 94.67% and reduced the loss. This hybrid approach successfully leverages a transformer's ability to capture global features alongside localized attention, enabling it to clearly distinguish even subtle differences in disease severity.

Moreover, when compared to state-of-the-art models, our approach proves its value in practical applications. For instance, it outperforms the CNN-based Yellow-Rust-Xception model by achieving a 3.67% higher accuracyall while eliminating the need for complex manual preprocessing steps. Although the C-DenseNet model reports a slightly higher accuracy of 97.99%, it was evaluated on a significantly smaller and less diverse dataset. Hence, our models performance on a larger and more varied dataset makes it better suited for real-world agricultural scenarios where simplicity, speed, and reliability are paramount.

General conclusion

General conclusion

In this work, we developed and evaluated an automated, deep learning based solution for plant leaf disease severity estimation, which is a critical task for advancing precision agriculture and safeguarding crop yields. We began with an in depth review of traditional diagnostic methods and their limitations, which motivated us to explore transformer architectures as innovative alternatives. By leveraging the global attention capacity of the Vision Transformer (ViT) and the hierarchical, locality-aware strengths of the Swin Transformer, we developed a novel combined model that effectively captures both broad contextual cues and fine-grained spatial details.

Our experimental framework was rigorously designed using the Yellow Rust 19 dataset, which provided a comprehensive set of images categorized into six distinct severity levels. We ensured reproducibility and robustness by implementing a standardized preprocessing pipeline with data augmentation techniques applicable during training. Through extensive analysis, we observed that while the individual models : ViT-small and Swin-base offered valuable insights into global and local feature extraction respectively, each had its own limitations. Specifically, ViT-small struggled with distinguishing borderline severity classes, whereas Swin-base delivered robust performance owing to its efficient window-based self-attention mechanism.

To overcome these challenges, we devised a combined model that fused the global contextual features from ViT-Small with the detailed, hierarchical representations obtained from Swin-Base. This integration significantly improved our overall performance, as our combined approach achieved a test accuracy of 94.67% and a notably low test loss. Moreover, our method eliminates the need for complex, manual preprocessing steps, simplifying deployment in real-world agricultural settings and providing immediate, reliable assessments of plant disease severity using standard imaging devices.

Our work contributes a transformative perspective on plant disease severity estimation. It shows that transformer models can improve how we estimate the severity of plant diseases. By capturing both global and local image features, these models can help build better, more reliable systems for automatically monitoring crops.

Perspectives & Future works

While our current work demonstrates promising results, several avenues remain to be explored to enhance the models performance, usability, and real-world applicability. Below, we outline key directions for future research and development.

- **Making the Model Lighter:** Although our model is effective, it is computationally heavy.
- **Using More Diverse Data:** Our model was tested on the Yellow Rust 19 dataset. Expanding to other crops, diseases, and environmental conditions would improve its robustness and generalizability.
- **Testing in the Field:** Field trials are essential to validate the model under real conditions. Deploying it on mobile or edge devices and collecting feedback from farmers will help improve the system.
- **Making the Model More Transparent:** To gain user trust, future versions should include explainability tools like attention maps or visual explanations to show how the model makes decisions.
- **Improving Feature Fusion:** We currently combine features from ViT and Swin using simple concatenation. Smarter fusion strategies like attention-based methods could better capture complementary information.

Bibliography

[1] Dilan Onat Alaku, brahim Türkolu. Smart Agriculture, Precision Agriculture, Digital Twins in Agriculture: Similarities and Differences. In 2024 Innovations in Intelligent Systems and Applications Conference (ASYU).

[2] Nitin Lokhande, Vijaya Thool, Pratap Vikhe. Comparative analysis of different plant leaf disease classification and detection using CNN. In 2024 International Conference on Recent Innovation in Smart and Sustainable Technology (ICRISST).

[3] Mohamed Rayane Lakehal, Hassiba Nemmour, Mohamed Lamine Bouibed, Yakout Fetmouche, Melissa Harchaoui, Youcef Chibani. CNN Ensembles for Pear Leaf Disease Severity Estimation. In 2023 30th IEEE International Conference on Electronics, Circuits and Systems (ICECS).

[4] K. R. Prasanna Kumar, M. Gunasekar, K. Logeswaran, P Dhanya Sree, P Malathi, V Vignesh. Fig Leaf Disease Prediction and Severity Estimation Using Deep Learning. In 2024 International Conference on Computing and Intelligent Reality Technologies (ICCIRT).

[5] Sachin Gupta, Baby Summuna and Moni Gupta. Plant Diseases: A Potential Threat to Global Food Security. 2015.

[6] Shurtle , M.C.; Pelczar, M.J.; Kelman, A.; Pelczar, R.M. Plant disease. In Plant Pathology; Encyclopedia Britannica: Chicago, IL, USA, 2020; Available online: [https://www.britannica.com/science/plant-disease].

[7] Ning Zhang, Guijun Yang, Yuchun Pan, Xiaodong Yang, Liping Chen and Chunjiang Zhao. A Review of Advanced Technologies and Development for Hyperspectral-Based Plant Disease Detection in the Past Three Decades.

[8] Archana Jain, Surendra Sarsaiya, Qin Wua , Yuanfu Lu, and Jingshan Shi. A review of plant leaf fungal diseases and its environment speciation. 2019

[9] *How to Identify and Control Common Plant Fungal Diseases.* [https://www.gardentech.com/blog/pest-id-and-prevention/keep-your-garden-free-from-fungal-disease].

[10] Sundin, G. W., Castiblanco, L. F., Yuan, X., Zeng, Q., and Yang, C.-H. *Bacterial disease management: challenges, experience, innovation and future prospects.* 2022

[11] PlantVillage Dataset. [https://www.kaggle.com/datasets/abdallahalidev/plantvillage-dataset].

[12] Cherlinka, V. (2025, February 19). Crop Diseases: Types, Control, and Prevention. EOS Data Analytics.[https://eos.com/blog/crop-diseases/]

[13] Laura Bouvet, Sarah Holdgate, Lucy James, Jane Thomas, Ian J. Mackay, James Cockram. The evolving battle between yellow rust and wheat: implications for global food security.

[14] Yellow rust: understanding, preventing and controlling this wheat disease. [https://uk.blog.sencrop.com/yellow-rust-understanding-preventing-and-controlling-this-wheat-disease/].

[15] Céréales : Risque de rouille jaune sur blé. [https://www.agri-mag.com/2017/06/19/cereales-risque-de-rouille-jaune-sur-ble/].

[16] X.M. Chen. Epidemiology and control of stripe rust [Puccinia striiformis f. sp. tritici] on wheat.

[17] Clive H. Bock, Jayme G. A. Barbedo, Emerson M. Del Ponte, David Bohnenkamp, Anne-Katrin Mahlein. From visual estimates to fully automated sensor-based measurements of plant disease severity: status and challenges for improving accuracy.

[18] Shi, T., Liu, Y., Zheng, X., Hu, K., Huang, H., Liu, H., & Huang, H. (2021). Recent advances in plant disease severity assessment using convolutional neural networks.

[19] Demba Faye, Idy Diop, Nalla Mbaye, Doudou Dione, Marius Mintu Diedhiou. Plant Disease Severity Assessment Based on Machine Learning and Deep Learning: A Survey. 2023

[20] Shi, T., Liu, Y., Zheng, X., Hu, K., Huang, H., Liu, H., Huang, H. (2023). Recent advances in plant disease severity assessment using convolutional neural networks. https://doi.org/10.1038/s41598-023-29230-7

[21] Hui Yao,Chunshan Wang, Lijie Zhang, Jiuxi Li, Bo Liu, Fangfang Liang. A Cucumber Leaf Disease Severity Grading Method in Natural Environment Based on the Fusion of TRNet and U-Net.

[22] Chuvieco, E., De Santis, A., Riaño, D., Halligan, K. . Simulation Approaches for Burn Severity Estimation Using Remotely Sensed Images. 2007.

[23] Hayit, T., Erbay, H., Varçn, F., Hayit, F., & Akci, N. (2021). Determination of the severity level of yellow rust disease in wheat by using convolutional neural networks. *Journal of Plant Pathology*. https://doi.org/10.1007/s42161-021-00886-2

[24] Mi et al., "Wheat Stripe Rust Grading by Deep Learning With Attention Mechanism and Images From Mobile Devices," 2020.

[25] Wang, X., Pan, T., Qu, J., Sun, Y., Miao, L., Zhao, Z., Li, Y., Zhang, Z., Zhao, H., Hu, Z., Xin, D., Chen, Q., & Zhu, R. (2023). Diagnosis of soybean bacterial blight progress stage based on deep learning in the context of data-deficient. *Computers and Electronics in Agriculture, 212*, 108170. https://doi.org/10.1016/j.compag.2023.108170

[26] Yang, B., Li, M., Li, F., Wang, Y., Liang, Q., Zhao, R., Li, C., & Wang, J. (2024). A novel plant type, leaf disease and severity identification framework using CNN and transformer with multi-label method. *Scientific Reports, 14*, 11664. https://doi.org/10.1038/s41598-024-62452-x

[27] Kundu, N., Rani, G., Dhaka, V.S., Gupta, K., Nayaka, S.C., Vocaturo, E., & Zumpano, E. (2022). Disease detection, severity prediction, and crop loss estimation in maize crop using deep learning. *Artificial Intelligence in Agriculture, 6*, 276-291. https://doi.org/10.1016/j.aiia.2022.11.002

[28] Parikh, A., Raval, M. S., Parmar, C., & Chaudhary, S. (2016). Disease detection and severity estimation in cotton plant from unconstrained images. *Proceedings of the IEEE DSAA 2016*, 81. https://doi.org/10.1109/DSAA.2016.81

[29] Wang, G., Sun, Y., & Wang, J. (2017). Automatic image-based plant disease severity estimation using deep learning. *Computational Intelligence and Neuroscience, 2017*, 2917536. https://doi.org/10.1155/2017/2917536

[30] Patil, S. B., & Bodhe, S. K. (2011). Leaf disease severity measurement using image processing. *International Journal of Engineering and Technology, 3(5)*, 297-301.

[31] Kaiyu Li, Yuzhaobi Song, Xinyi Zhu, Lingxian Zhang. A severity estimation method for lightweight cucumber leaf disease based on DM-BiSeNet. 2024.

[32] Ye, Z., Liu, Y., Ye, F., Li, H., Luo, J., Guo, J., Feng, Z., Hong, C., Li, L., Liu, S., Yang, B., Liu, W., & Yao, Q. (2025). Automatic diagnosis of agromyzid leafminer damage levels using leaf images captured by AR glasses.

[33] Sequence Models Compared: RNNs, LSTMs, GRUs, and Transformers. [`https://aiml.com/compare-the-different-sequence-models-rnn-lstm-gru-and-transformers/`].

[34] NLP Rise with Transformer Models | A Comprehensive Analysis of T5, BERT, and GPT. [`https://www.unite.ai/nlp-rise-with-transformer-models-a-comprehensive-analysis-of-t5-bert-and-gpt/`].

[35] Vaswani et al., "Attention Is All You Need," 2017.

[36] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021.

[37] Vision Transformer: What It Is & How It Works [2024 Guide]. [`https://www.v7labs.com/blog/vision-transformer-guide`].

[38] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows*.

[39] Swin Transformer. [`https://www.geeksforgeeks.org/computer-vision/swin-transformer/`].

[40] Explanation Swin Transformer[`https://chautuankien.medium.com/explanation-swin-transformer-93e7a3140877/`]

[41] *What is Python? Executive Summary* [https://www.python.org/doc/essays/].

[42] *What is Visual Studio Code?* [https://code.visualstudio.com/].

[43] *What Is PyTorch? Definition, Uses and Tools.* [https://builtin.com/machine-learning/pytorch]

[44] *timm: PyTorch Image Models.* https://timm.fast.ai/].

[45] *Yellow Rust 19 Dataset.* https://www.kaggle.com/datasets/tolgahayit/yellowrust19-yellow-rust-disease-in-wheat].

[46] *What is a confusion matrix ?*

[https://plat.ai/blog/confusion-matrix-in-machine-learning/].