UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DE COMPUTAÇÃO

IGOR DE MOURA PHILIPPINI

Hydra: A Multi-Task Learning Approach to Fine-Grained Leaf-Level Agricultural Diagnostics

Recife

2025

IGOR DE MOURA PHILIPPINI

Hydra: A Multi-Task Learning Approach to Fine-Grained Leaf-Level Agricultural Diagnostics

The dissertation was submitted to the Graduate Program in Computer Science at the Informatics Center of the Federal University of Pernambuco as a partial requirement for obtaining a Master's degree in Computer Science.

Area of Concentration: Computational Intelligence

Advisor: Prof. Dr. Stefan Michael Blawid

Recife

2025

.Catalogação de Publicação na Fonte. UFPE - Biblioteca Central

Philippini, Igor de Moura.
Hydra: a Multi-Task learning approach to fine-grained leaf-level agricultural diagnostics / Igor de Moura Philippini. - Recife, 2025.
205f.: il.

Dissertação (Mestrado)- Universidade Federal de Pernambuco, Centro de Informática, Programa de Pós-Graduação em Ciência da Computação, 2025.
Orientação: Stefan Michael Blawid.

1. Aprendizagem de máquina; 2. VIsão computacional; 3. Assistente de fitopatologia; 4. Aprendizado profundo. I. Blawid, Stefan Michael. II. Título.

UFPE-Biblioteca Central

Igor de Moura Philippini

“Hydra: A Multi-Task Learning Approach to Fine-Grained Leaf-Level Agricultural Diagnostics”

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação. Área de Concentração: Inteligência Computacional.

Aprovado em: 28/07/2025.

BANCA EXAMINADORA

_____

Prof. Dr. Fernando Maciano de Paula Neto
Centro de Informática / UFPE

_____

Prof. Dr. Jayme Garcia Arnal Barbedo
EMBRAPA, Informática Agropecuária

_____

Prof. Dr. Stefan Michael Blawid
Centro de Informática / UFPE
(**orientador**)

“One thing I’ve learned: you can know anything, it’s all there, you just have to find it.”

— Neil Gaiman

AGRADECIMENTOS

Olhando para tudo o que vivi até agora, com certeza eu não estaria onde estou hoje sem todo o apoio da minha família, que fez tudo o que estava ao seu alcance, e muitas vezes até mesmo quando não estava, para me proporcionar tudo o que eu precisava para ter sucesso na vida.

Gostaria também de agradecer à minha namorada Rosa, que sempre esteve ao meu lado, me apoiando, motivando, e incentivamento a crescer não só como pessoa, mas também como profissional, sendo sempre muito compreensiva e carinhosa comigo.

Gostaria de agradecer em especial ao professor Stefan Blawid, que sempre foi paciente e demonstrou muito interesse no que estávamos construindo, mesmo quando estava fora de sua área de conhecimento, e sempre me incentivou a continuar seguindo e prosseguir com o nosso projeto, independente das adversidades e sempre da forma mais compreensível possível.

Também agradeço a todos os amigos, os quais são muitos para citar individualmente, mas vocês sabem bem quem são e que vivem no meu coração.

Por fim, mas não menos importante, um agradecimento especial a esta universidade, onde passei grande parte da minha vida e aprendi lições valiosas, podendo entrar em contato com os mais diversos tipos de pessoas em vários contextos distintos, que me ajudaram a entender diferentes pontos de vista e a ser mais empático com a trajetória única de cada um.

RESUMO

Pragas e doenças representam grandes desafios na agricultura, levando a perdas econômicas significativas. O uso incorreto de pesticidas, frequentemente decorrente de diagnósticos errados, agrava o problema, especialmente para agricultores familiares que não possuem acesso ao suporte especializado e acesso a informações em tempo hábil. Embora existam sistemas de visão computacional que auxiliam nesse problema com sua capacidade de detectar doenças em plantas a partir de imagens de folhas ou frutos, a maioria está limitada a esta única tarefa e não contempla o processo diagnóstico completo exigido em cenários práticos. Este trabalho propõe uma abordagem unificada baseada em Multi-Task Learning (MTL) para lidar com múltiplas tarefas relacionadas ao processo de diagnóstico de doenças em plantas a partir de uma única imagem de entrada. O modelo proposto é capaz de: (i) determinar se uma imagem contém uma folha, (ii) detectar se a folha está saudável ou doente, (iii) classificar a espécie da planta, (iv) identificar o provável agente patogênico, (v) detectar macro-sintomas visíveis associados à doença e (vi) classificar a doença específica da planta, quando presente. Para dar suporte à utilização prática, também desenvolvemos um sistema completo de diagnóstico em torno do modelo, que inclui detecção e segmentação automáticas das folhas, permitindo o processamento de todas as folhas presentes em uma imagem. O sistema é exposto por meio de uma API RESTful, que serve como interface central para inferências. Além disso, uma aplicação web intuitiva é construída sobre essa API, possibilitando que usuários finais—como agricultores e técnicos agrícolas—interajam facilmente com o modelo e testem suas funcionalidades através de uma interface visual acessível.

Palavras-chaves: Aprendizagem de máquina; Visão Computacional;
Detecção de doenças em plantas; Assistente de fitopatologia; Multi-Task Learning;

ABSTRACT

Pests and diseases pose major challenges in agriculture, leading to substantial economic losses. The misuse of pesticides, often stemming from incorrect diagnoses, exacerbates the problem, particularly for smallholder farmers who lack access to expert support and timely information. Although existing computer vision systems assist in detecting plant diseases from leaf or fruit images, most are restricted to single-task outputs and do not address the full diagnostic process required in practical scenarios.

This work proposes a unified approach based on Multi-Task Learning (MTL) to address multiple key diagnostic tasks from a single input image. The proposed model is capable of: (i) determining whether an image contains a leaf, (ii) detecting whether the leaf is healthy or sick, (iii) classifying the plant species, (iv) identifying the likely pathological agent, (v) detecting visible macro-symptoms associated with disease, and (vi) classifying the specific plant disease when present.

To support practical usage, we also develop a complete diagnostic system around the model that includes automatic leaf detection and segmentation for processing all leaves in an image. The system is exposed via a RESTful API, which serves as the core inference interface. Additionally, a user-friendly web application is built on top of this API, allowing end users—such as farmers and agricultural technicians—to easily interact with the model and test its capabilities through an accessible visual interface.

Keywords: Machine learning; Computer vision; Plant disease detection; Plant pathology assistant; Multi-Task Learning.

LIST OF FIGURES

LIST OF FRAMES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programming Interface
AS	Accuracy Score
ASIC	Application-Specific Integrated Circuit
CNN	Convolutional Neural Network
COCO	Common Objects in Context
CPU	Central Processing Unit
CV	Computer Vision
DL	Deep Learning
GPU	Graphics Processing Unit
Grad-CAM	Gradient-weighted Class Activation Mapping
GRU	Gated Recurrent Unit
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
IR	Infrared
JSS	Jaccard Similarity Score
LiDAR	Light Detection and Ranging
LLM	Large Language Model
LSTM	Long Short-Term Memory
ML	Machine Learning
MLP	Multi-Layer Perceptrons
MTL	Multi-Task Learning
NLP	Natural Language Processing
RAG	Retrieval-Augmented Generation
ResNet	Residual Network

RGB	Red-Green-Blue
RNN	Recurrent Neural Network
SAM	Segment Anything Model
SVM	Support Vector Machine
TPU	Tensor Processing Unit
ViT	Vision Transformer
XAI	Explainable Artificial Intelligence

CONTENTS

1 INTRODUCTION

1.1 MOTIVATION

Global food security is intrinsically linked to crop health, yet agricultural production faces persistent threats from pests and diseases. In Brazil, one of the world's largest food producers, these challenges are particularly acute, with estimated economic losses reaching 43% of annual production (IBGE, 2017). The nation's agricultural landscape is dominated by smallholders, who account for over 70% of domestic food production but often lack access to timely and accurate phytopathology expertise. This knowledge gap contributes to Brazil's standing as a leading global consumer of pesticides (RIGOTTO; VASCONCELOS; ROCHA, 2014), where the misuse of agrochemicals can lead to ineffective pest control, increased costs, and negative environmental externalities. With an average of only one agronomist for every 270 farmers (ASBRAER, 2014), the need for scalable, accessible, and reliable diagnostic tools is paramount.

The convergence of computer vision and machine learning offers a promising pathway to augment the diagnostic capabilities of agricultural specialists (BOULENT et al., 2019). By analyzing digital images of plant leaves, these technologies can rapidly identify symptoms and pathogens, enabling faster intervention and real-time disease assessments (BOCK et al., 2020). However, many existing systems exhibit significant limitations: they often address only a single classification task (e.g., distinguishing a healthy leaf from a diseased one), are trained on sanitised laboratory images that fail to represent real-world field conditions (MOHANTY; HUGHES; SALATHÉ, 2016), and operate as isolated tools rather than integrated components of a larger agricultural workflow.

This thesis posits that a more holistic approach is required to create a truly effective digital decision support system. We argue that plant disease diagnosis is not a singular classification problem but an inherently multi-faceted analytical process. Therefore, we leverage Multi-Task Learning (MTL), a paradigm where a single model learns to perform multiple related tasks simultaneously. This approach mimics the reasoning of a human expert—who might simultaneously assess health, identify symptoms, and infer a causal agent—and promotes the sharing of learned representations across tasks, improving generalization and reducing the risk of overfitting (RUDER, 2017; CRAWSHAW, 2020). Building upon this principle, from the work using a single-task model built by Barros et al.

(2021), this research develops and evaluates **Hydra**: an AI-powered system designed to accelerate the entire plant disease diagnostic pipeline. Hydra is conceptualized as a dual-purpose platform: a standalone web application for direct use and an API-first service for seamless integration into other agricultural software. By structuring the analysis into a multi-step process, Hydra aims to transform a diagnostic workflow that traditionally takes hours or days into a matter of seconds, thereby empowering specialists and enhancing their reach through remote monitoring capabilities (HAMPF et al., 2021).

1.2 PROBLEM STATEMENT

While the concept of image-based plant disease detection is not new, the transition from academic prototypes to robust, field-ready tools is hindered by several practical challenges. This thesis addresses the following specific problems:

1. **Fragmented and Inefficient Diagnosis:** Existing digital tools often rely on separate, single-purpose models for each diagnostic step. This approach is computationally inefficient, can produce contradictory results, and fails to model the interdependent nature of phytopathological reasoning, overlooking the regular diagnosis process of a human expert.

2. **Poor Generalization to Field Conditions:** Many models are over-specialized to a single crop or trained on idealized lab images. Their performance degrades significantly on in-field images with cluttered backgrounds, variable lighting, and diverse camera angles.

3. **Expertise as a Bottleneck:** The specialized knowledge of phytopathologists is a scarce resource. The lack of scalable, API-driven tools prevents this expertise from being effectively integrated into modern digital workflows, delaying critical treatment decisions.

4. **The "Black Box" Dilemma:** For a diagnostic tool to be trusted, its conclusions must be interpretable. Most deep learning models operate as opaque "black boxes," limiting their utility in high-stakes agricultural contexts where understanding the "why" is as crucial as the "what." This is particularly important for the field of

plant pathology, where the diagnosis of a disease is not only about identifying the disease, but also about understanding the causal agent and the severity of the disease.

1.3 OBJECTIVES

To address these challenges, this research pursues the following primary objectives:

1. **Design and Implement a Unified Multi-Task Architecture:** Develop a single MTL neural network capable of executing five core diagnostic tasks from a single 256×256 RGB image: (a) leaf recognition, (b) binary health assessment (healthy vs. diseased), (c) macro-symptom classification (e.g., chlorosis, rust), (d) causal agent identification (fungi, bacteria, virus, etc.), and (e) disease identification (e.g., powdery mildew, black rot, etc.).

2. **Develop a Robust, Field-Oriented Data Pipeline:** Assemble a large-scale, multi-crop dataset of over 60,000 images from existing datasets. Implement a standardized pre-processing pipeline, featuring background removal, to enhance model focus and robustness to visual noise typical of in-field images, alongside with a robust training pipeline to ensure the model is able to generalize to new crops and diseases, as well as being able to easily fine-tune for new datasets and tasks.

3. **Benchmark and Validate Against Diverse Architectures:** Conduct a comprehensive performance evaluation by benchmarking multiple modern CNN backbones (including ResNet50V2, MobileNetV2, EfficientNet, and Vision Transformers) and quantify the performance gains of the MTL model against a single-task baseline.

4. **Deliver an End-to-End Explainable System:** Deploy the model as a REST API and build an integrated platform around it, featuring a web-based client for ease of use by end-users, a REST API for integration with other systems, and Grad-CAM visualizations to provide visual explanations for model predictions.

1.4 CONTRIBUTIONS

This dissertation delivers the following contributions to the fields of computer vision and digital agriculture:

1. **A Novel Multi-Task Learning Framework for Plant Pathology:** We present a compact six-task CNN that shares over 98% of its parameters, demonstrating an efficient and effective method for modeling the diagnostic process. The model surpasses the baseline F1-score by up to 8 percentage points on challenging field-captured images.

2. **A Curated Public Dataset and Pre-processing Toolkit:** We release a cleaned, annotated, and aggregated dataset for multi-task plant disease classification, along with reproducible scripts for image pre-processing and model training, providing a valuable resource for future research.

3. **An Open-Source, Full-Stack Diagnostic Platform:** We deliver an operational system comprising an inference server available as a REST API, a web-based client for end-users, and a REST API for integration with other systems.

4. **Empirical Insights and Architectural Guidelines:** We provide a thorough comparative analysis of different neural network backbones for this domain. These findings offer practical guidelines for developing future machine learning models in agriculture.

2 THEORETICAL BACKGROUND

This chapter establishes the theoretical foundation necessary for understanding the research contributions presented in the proposed work. We examine the fundamental concepts and mathematical principles underlying modern machine learning approaches, focusing on artificial neural networks, computer vision techniques, and multi-task learning. The chapter also reviews key works that have shaped the development of the methodologies and experimental approaches discussed in later chapters. This theoretical background provides the conceptual basis for interpreting the novel contributions and experimental results that follow.

2.1 ARTIFICIAL INTELLIGENCE

Artificial Intelligence (AI) is a broad field that encompasses various subareas, including machine learning (ML), deep learning (DL), artificial neural networks (ANN), computer vision (CV), natural language processing (NLP), and robotics.

Its core principle is to create machines that can mimic human intelligence. The concept of intelligence is broad and can be defined in various ways, and there are many philosofical discussions on what encompasses intelligence and human behaviour, including ideas on consciousness, human emotion and how we react to it, and what we consider to be rational or irrational behaviour. As pointed out by Russell e Norvig (2009), should we be concerned with thinking or behavior when building such machines? Do we want to model humans, or work from an ideal standard?

For the purpose of this work, we will focus on the discussions that follow closely to the idea of a machine that can try to solve problems and achieve results in a more rational way, which can be broken down in two main approaches: the symbolic approach and the connectionist approach.

2.1.1 Symbolic Approach

The symbolic approach, which is also referred to as classical AI, is based on the idea of a machine that can reason about the world and solve problems using a set of rules

and symbols. Many of its core ideas were based in a set of rules to achieve a certain goal, be it by chaining a series of conditions based on the information from experts of a given field, or by using specific algorithms that are able to solve problems in a more general way.

Classic examples of some of the algorithms that are based on the symbolic approach are the search algorithms, such as the A* algorithm, or decision rules, such as the Minimax algorithm.

However, those methodologies either have a limited scope, or are not able to solve problems in a general way, or demand too much time and resources in order to build an adequate knowledge base out of expert knowledge.

2.1.2 Connectionist Approach

The connectionist approach, which is also referred to as modern AI, is based on the idea of a machine that can learn from experience based on examples, usually in a non-deterministic way. The idea is to learn the rules of a given problem by trial and error, and then use those learned rules to solve new problems.

Although it usually involves processes that are not deterministic, it has shown to be able to solve problems in a more general way, being able to learn from a large amount of examples, and being able to generalize to unseen problems. Examples of connectionist approaches are the subfields of machine learning, and more specifically artificial neural networks.

Figure 1 illustrates the main differences between the symbolic and connectionist approaches to AI.

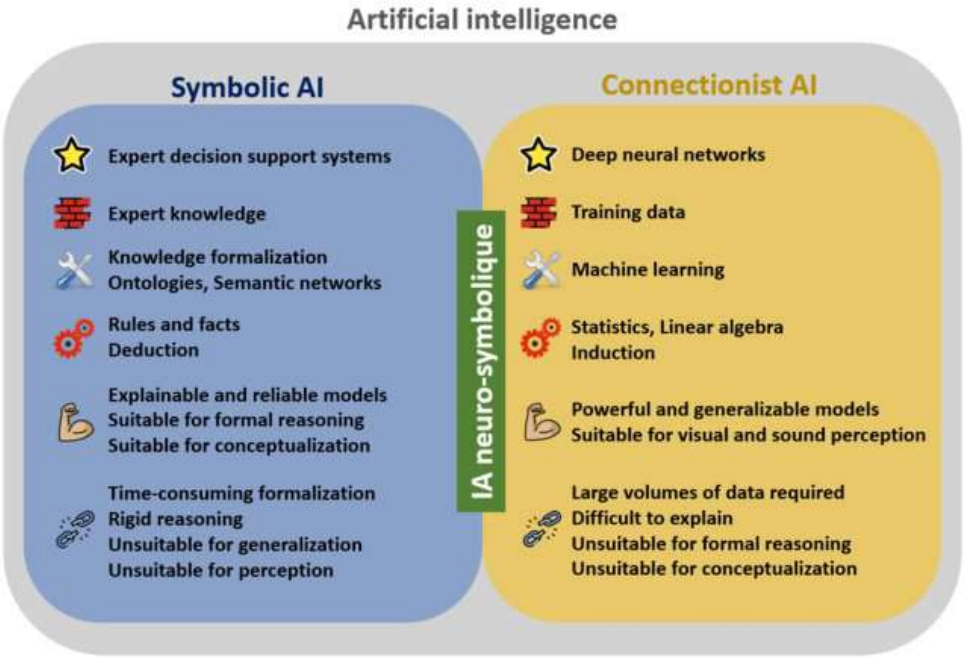


Figure 1 – Comparison between the symbolic and connectionist approaches to AI. Source: Alsaïdi (2023).

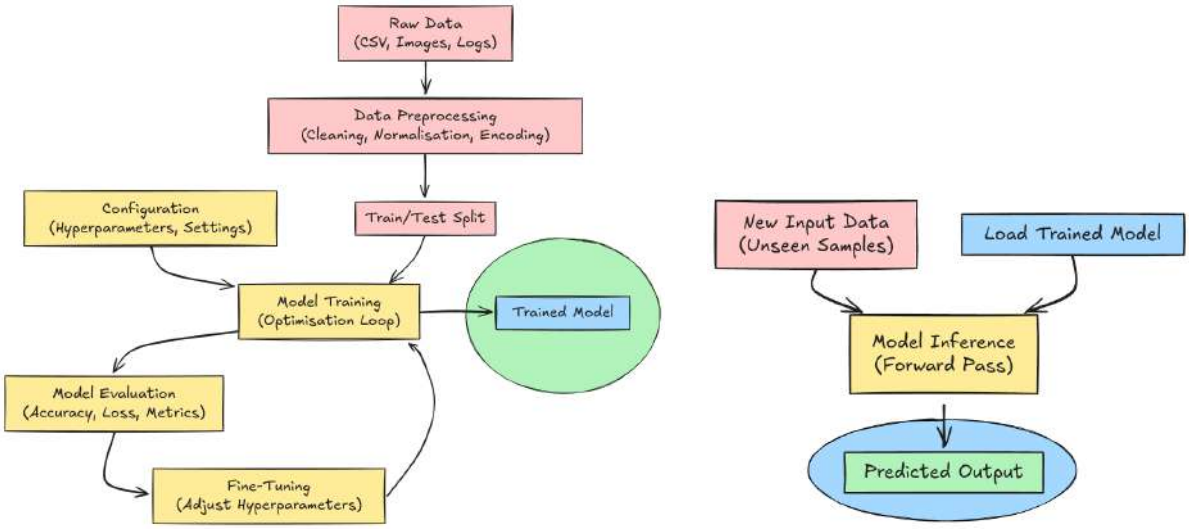
2.2 MACHINE LEARNING

Machine Learning (ML) is one of most discussed subareas of Artificial Intelligence (AI). ML is the subarea that tries to achieve a desired outcome without being given specific instructions on how to do so. Instead, predictions are obtained based on given examples and attempts to generalize from them by trial and error, as shown by Solomonoff (1957).

Using such techniques allows society to solve novel problems without spending much time pursuing mathematical proofs or complex formulas at the cost of having non-deterministic results due to the lack of interpretability of neural network-based models, often referred to as "black-box" models. The trade-off is deemed acceptable for most practical applications, ranging from recommendation systems to content creation, data forecasting, and speech and image recognition, the latter being the focus of this work.

The fundamental goal of machine learning is to create models that can generalize well to unseen data, meaning they perform accurately on new examples that were not part of the training process. This generalization capability is what distinguishes effective machine learning models from simple memorization systems (VAPNIK, 2013).

A typical machine learning workflow is divided into two main phases: training and



(a) Training Phase (b) Inference Phase

Figure 2 – Generic Machine Learning Workflow. The diagrams illustrate the standard pipeline commonly adopted in machine learning systems, separated into two distinct phases: training (a) and inference (b). Source: Author.

inference. During the training phase, the model learns from labeled examples by optimizing its parameters to minimize a loss function, resulting in a trained model. In the inference phase, this trained model is used to make predictions on new, unseen data. Figure 2 illustrates this standard pipeline, highlighting the distinction between the training and inference processes:

- **Training Phase:** The process begins with examples of data (e.g., CSVs, images, logs), which may be preprocessed through cleaning, normalization, and various other techniques to improve the quality of the data. This data is then split into training and testing subsets. During training, the model learns by minimising the difference of its predictions from the ground truth through iterative optimisation. A configuration module usually supplies hyperparameters such as learning rate and batch size. After initial training, model performance is evaluated using test data, and hyperparameters — for models that have them — may be adjusted in a feedback loop to refine the model. The output is a trained model ready for deployment. Another common practice is to use a validation set to tune the hyperparameters and select the best model.

- **Inference Phase:** Once the model is trained, it can be used to infer outcomes from previously unseen data. New input samples are passed through the trained

model in a forward-pass operation, producing predictions. This phase is typically deterministic and does not involve updates to the model's parameters.

2.2.1 Learning Paradigms

Machine learning approaches can be categorized into several fundamental paradigms based on the nature of the training data and the learning objectives. Understanding these paradigms is crucial for selecting appropriate methodologies for specific problems and interpreting their limitations and capabilities.

2.2.1.1 Supervised Learning

Supervised learning is the most widely used machine learning paradigm, where models are trained on datasets containing input-output pairs, also known as labeled data (BISHOP, 2006). The objective is to learn a mapping function $f : \mathcal{X} \rightarrow \mathcal{Y}$ from input space $\mathcal{X}$ to output space $\mathcal{Y}$ that can accurately predict outputs for new, unseen inputs.

Supervised learning problems are typically divided into two main categories:

Classification: The output variable is categorical, representing discrete classes or labels. For binary classification, $\mathcal{Y} = \{0, 1\}$ or $\mathcal{Y} = \{-1, +1\}$, while multi-class problems have $\mathcal{Y} = \{1, 2, ..., K\}$ where K is the number of classes. Examples include email spam detection, image recognition, and medical diagnosis.

Regression: The output variable is continuous, where $\mathcal{Y} \subseteq \mathbb{R}$. The goal is to predict numerical values such as house prices, stock market values, or temperature forecasting.

The performance of supervised learning algorithms is typically evaluated using metrics such as accuracy, precision, recall, and F1-score for classification tasks, and mean squared error (MSE), mean absolute error (MAE), and coefficient of determination (R^2) for regression tasks.

2.2.1.2 Unsupervised Learning

Unsupervised learning deals with datasets that contain only input data without corresponding target outputs (HASTIE; TIBSHIRANI; FRIEDMAN, 2009). The goal is to discover hidden patterns, structures, or representations within the data. Since there are

no explicit labels to guide the learning process, unsupervised methods must rely on intrinsic properties of the data.

Common unsupervised learning tasks include:

Clustering: Partitioning data into groups (clusters) such that data points within the same cluster are more similar to each other than to those in other clusters. Popular algorithms include k-means, hierarchical clustering, and Gaussian mixture models.

Dimensionality Reduction: Reducing the number of features while preserving important information. Techniques such as Principal Component Analysis (PCA), t-Distributed Stochastic Neighbor Embedding (t-SNE), and autoencoders are commonly used for visualization and feature extraction.

Density Estimation: Learning the probability distribution of the data, which can be used for anomaly detection, data generation, and statistical inference.

Association Rule Learning: Discovering relationships between different variables, commonly used in market basket analysis and recommendation systems.

2.2.1.3 Semi-Supervised Learning

Semi-supervised learning represents a hybrid approach that leverages both labeled and unlabeled data during training (CHAPELLE; SCHÖLKOPF; ZIEN, 2006). This paradigm is particularly valuable in scenarios where obtaining labeled data is expensive, time-consuming, or requires expert knowledge, while unlabeled data is readily available.

The fundamental assumption underlying semi-supervised learning is that the distribution of unlabeled data contains information about the underlying structure that can improve learning performance. Several key assumptions guide semi-supervised approaches:

Smoothness Assumption: If two points are close in the input space, their outputs should be similar.

Cluster Assumption: Data points in the same cluster are likely to belong to the same class.

Manifold Assumption: High-dimensional data lies on a lower-dimensional manifold, and both labeled and unlabeled data share this manifold structure.

Common semi-supervised learning techniques include self-training, co-training, graph-based methods, and semi-supervised variants of deep learning models. These

approaches have shown particular success in domains such as natural language processing, computer vision, and bioinformatics, where large amounts of unlabeled data are available but labeling requires specialized expertise.

2.2.2 Model Complexity and Generalization

A fundamental challenge in machine learning is achieving the right balance between model complexity and generalization performance. This balance is crucial for developing models that perform well on unseen data rather than merely memorizing the training examples.

2.2.2.1 Overfitting and Underfitting

Overfitting occurs when a model learns not only the underlying patterns in the training data but also the noise and random fluctuations specific to the training set (HAWKINS, 2004). An overfitted model exhibits excellent performance on training data but poor generalization to new, unseen data. This phenomenon is particularly common in complex models with many parameters relative to the amount of training data available.

Mathematically, overfitting can be understood through the bias-variance decomposition of the expected prediction error. For a model $\hat{f}$ trained on dataset $\mathcal{D}$, the expected error on a new point (x, y) can be decomposed as:

$$\mathbb{E}[(y - \hat{f}(x))^2] = \text{Bias}^2[\hat{f}(x)] + \text{Var}[\hat{f}(x)] + \sigma^2 \tag{2.1}$$

where σ^2 represents the irreducible error due to noise in the data.

Underfitting, conversely, occurs when a model is too simple to capture the underlying patterns in the data. Underfitted models exhibit poor performance on both training and test data, indicating high bias but low variance.

The relationship between model complexity and these phenomena is illustrated by the bias-variance tradeoff: as model complexity increases, bias typically decreases while variance increases. The optimal model complexity minimizes the total expected error.

2.2.2.2 Regularization Techniques

Regularization is a fundamental technique for controlling model complexity and preventing overfitting by adding constraints or penalties to the learning objective (TIKHONOV; ARSENIN, 1977). The general form of a regularized objective function is:

$$\mathcal{L}_{\text{reg}}(\theta) = \mathcal{L}_{\text{data}}(\theta) + \lambda \mathcal{R}(\theta) \tag{2.2}$$

where $\mathcal{L}_{\text{data}}(\theta)$ is the data-fitting term (e.g., mean squared error), $\mathcal{R}(\theta)$ is the regularization term, and λ is the regularization parameter that controls the strength of regularization.

Common regularization techniques include:

L1 Regularization (Lasso): Adds the sum of absolute values of parameters:

$$\mathcal{R}(\theta) = \|\theta\|_1 = \sum_{i=1}^{p} |\theta_i| \tag{2.3}$$

L1 regularization promotes sparsity by driving some parameters to exactly zero, effectively performing feature selection.

L2 Regularization (Ridge): Adds the sum of squared parameters:

$$\mathcal{R}(\theta) = \|\theta\|_2^2 = \sum_{i=1}^{p} \theta_i^2 \tag{2.4}$$

L2 regularization shrinks parameters toward zero but rarely makes them exactly zero, providing a smoother regularization effect.

Elastic Net: Combines L1 and L2 regularization:

$$\mathcal{R}(\theta) = \alpha \|\theta\|_1 + (1 - \alpha) \|\theta\|_2^2 \tag{2.5}$$

2.2.2.3 Cross-Validation

Cross-validation is a statistical technique used to assess model performance and select hyperparameters while making efficient use of available data (STONE, 1974). The most common approach is k-fold cross-validation, where the dataset is partitioned into k equally-sized folds. The model is trained on $k - 1$ folds and validated on the remaining fold, with this process repeated k times.

The cross-validation estimate of the expected error is:

$$\mathsf{CV}_{(k)} = \frac{1}{k} \sum_{i=1}^{k} \mathcal{L}(\mathcal{D}_i^{\mathsf{val}}, \hat{f}^{(-i)}) \tag{2.6}$$

where $\mathcal{D}_i^{\mathsf{val}}$ is the i-th validation fold and $\hat{f}^{(-i)}$ is the model trained on all folds except the i-th.

Cross-validation serves multiple purposes: (1) providing an unbiased estimate of model performance, (2) enabling model selection among different algorithms or architectures, and (3) facilitating hyperparameter tuning. Special cases include leave-one-out cross-validation (LOOCV) where k equals the number of data points, and stratified cross-validation, which maintains class proportions in each fold for classification problems.

2.2.3 Neurons and Artificial Neural Networks

The basis of any modern model able to learn and achieve results without being explicitly programmed to do so is related to the neuron found in the human brain. McCulloch e Pitts (1943) showed how a simple mathematical formula based on inputs, multiplied by arbitrary weights and then summed together before being compared through an activation function, can describe a human neuron. The model was called a Perceptron later on by Rosenblatt (1957), who also showed how multiple Perceptrons can work in tandem and in parallel.

2.2.3.1 Mathematical Foundation of Artificial Neurons

At its core, an artificial neuron is a computational unit that performs a weighted sum of its inputs followed by a non-linear transformation. Mathematically, for a neuron j in layer l, the computation can be expressed as:

$$z_j^{(l)} = \sum_{i=1}^{n} w_{ij}^{(l)} x_i^{(l-1)} + b_j^{(l)} \tag{2.7}$$

$$a_j^{(l)} = f(z_j^{(l)}) \tag{2.8}$$

where:

- $x_i^{(l-1)}$ represents the i-th input from the previous layer (or raw input for the first layer)
- $w_{ij}^{(l)}$ denotes the weight connecting input i to neuron j in layer l
- $b_j^{(l)}$ is the bias term for neuron j in layer l
- $z_j^{(l)}$ is the weighted sum (pre-activation value)
- $a_j^{(l)}$ is the final output (post-activation value)
- $f(\cdot)$ represents the activation function

2.2.3.2 The Learning Parameters of a Neural Network: Weights and Biases

Weights and biases are the fundamental learnable parameters in neural networks, serving distinct but complementary roles in the learning process.

Weights (w_{ij}): These parameters control the strength and direction of connections between neurons. Each weight represents the importance of a particular input to the neuron's output. Positive weights amplify the input signal, while negative weights inhibit it. The magnitude of the weight determines the strength of this influence. During training, weights are adjusted through gradient descent to minimize the loss function, effectively learning which input features are most relevant for the task at hand.

Biases (b_j): The bias term allows each neuron to shift its activation threshold, providing additional flexibility in the decision boundary. Without bias terms, the neuron's output would always be zero when all inputs are zero, severely limiting the model's expressiveness. Biases enable neurons to be active even when inputs are small or zero, allowing the network to learn more complex patterns and make the model more flexible in fitting the training data.

The initialization of weights and biases is crucial for successful training. Common initialization strategies include Xavier/Glorot initialization (GLOROT; BENGIO, 2010) and He initialization (HE et al., 2015), which help maintain proper gradient flow during back-propagation and prevent vanishing or exploding gradient problems.

2.2.3.3 Activation Functions

Activation functions are mathematical functions that determine the output of a neuron given its input. They introduce non-linearity into the network, enabling it to learn complex patterns and relationships that would be impossible with linear transformations alone. Without activation functions, a multi-layer neural network would be equivalent to a single linear transformation, regardless of the number of layers.

Common activation functions include:

Sigmoid Function:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.9}$$

The sigmoid function maps any real value to the range (0, 1), making it historically popular for binary classification tasks. However, it suffers from the vanishing gradient problem for very large or small input values, where gradients become extremely small, slowing down learning in deep networks.

Hyperbolic Tangent (tanh):

$$\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{2.10}$$

The tanh function maps inputs to the range (-1, 1) and is zero-centered, which can lead to faster convergence compared to sigmoid. However, it still suffers from vanishing gradients for extreme input values.

Rectified Linear Unit (ReLU):

$$\text{ReLU}(z) = \max(0, z) \tag{2.11}$$

ReLU has become the most widely used activation function in deep learning due to its computational efficiency and ability to mitigate the vanishing gradient problem. It outputs the input directly if positive, otherwise zero. Despite its simplicity, ReLU can suffer from the "dying ReLU" problem, where neurons can become permanently inactive.

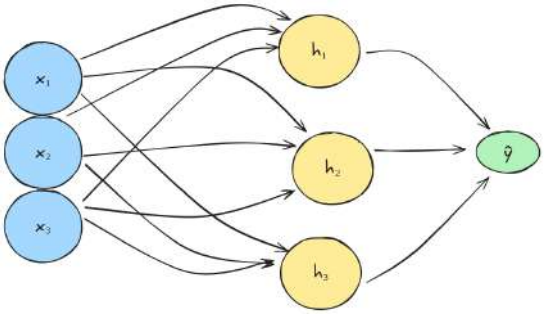
Leaky ReLU:

$$\text{Leaky ReLU}(z) = \max(\alpha z, z) \tag{2.12}$$

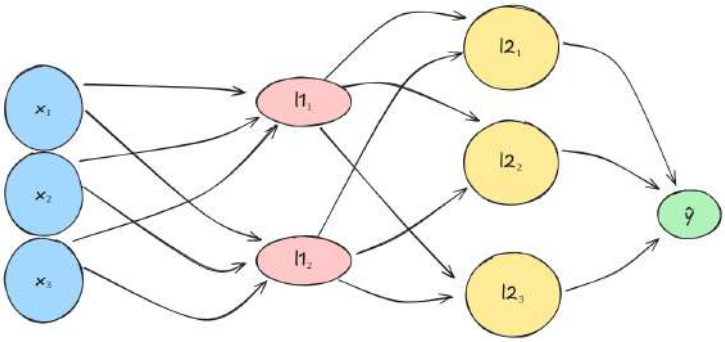
The Leaky ReLU function addresses the dying ReLU problem by allowing a small gradient when the input is negative, where α is a small positive constant (typically 0.01). This variant addresses the dying ReLU problem by allowing a small gradient when the input is negative.

2.2.3.4 Multi-layer Networks and Universal Approximation

Figure 3 illustrates the fundamental differences between basic neural network architectures. The single-layer network shown in (a) demonstrates the simplest form of artificial neural network, where inputs are directly connected to a single layer of hidden units through weighted connections, each with their own bias terms, which then produce the output after applying activation functions. This basic architecture, while limited in representational capacity, clearly shows how neural networks process information through the mathematical operations described in Sec. 2.2.3.2 e Sec. 2.2.3.3.



(a) Single-layer neural network



(b) Multi-layer neural network

Figure 3 – Comparison of Neural Network Architectures. In (a), a single-layer neural network is shown, consisting of three input features (x_1, x_2, x_3), a single fully-connected layer with three hidden units (h_1, h_2, h_3), and a single output $(\hat{y})$. Each input is connected to all hidden units, and all hidden units are connected to the output. This architecture represents a shallow model with limited representational capacity. In (b), a three-layer neural network is depicted. It takes the same three input features (x_1, x_2, x_3), followed by a first hidden layer with two units $(l1_1, l1_2)$, a second hidden layer with three units $(l2_1, l2_2, l2_3)$, and a final output layer with one unit $(\hat{y})$. Each layer is fully connected to the next, allowing for hierarchical feature transformations. This deeper architecture increases model expressiveness and is better suited for capturing complex data patterns. Source: Author.

However, a single-layer Perceptron network was not sophisticated enough to solve even some trivial problems, such as a boolean XOR. This limitation, famously highlighted by Minsky e Papert (1969), led to what became known as the "AI winter" in the 1970s.

Rosenblatt himself demonstrated that layers of Perceptrons built on top of one another are required to solve such problems.

The multi-layer architecture depicted in Figure 3(b) shows how deeper networks can capture more complex patterns through hierarchical feature learning, where each layer transforms the input in increasingly abstract ways. In a multi-layer network, the output of one layer becomes the input to the next layer, creating a composition of functions:

$$\mathbf{a}^{(L)} = f^{(L)}(\mathbf{W}^{(L)} f^{(L-1)}(\mathbf{W}^{(L-1)} \cdots f^{(1)}(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}) \cdots + \mathbf{b}^{(L-1)}) + \mathbf{b}^{(L)}) \tag{2.13}$$

where L is the number of layers, $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ are the weight matrix and bias vector for layer l, and $f^{(l)}$ is the activation function for layer l.

The theoretical foundation for multi-layer networks' power lies in the Universal Approximation Theorem (CYBENKO, 1989; HORNIK; STINCHCOMBE; WHITE, 1989), which states that a feedforward network with a single hidden layer containing a finite number of neurons can approximate any continuous function on a compact subset of $\mathbb{R}^n$ to arbitrary accuracy, provided the activation function is non-constant, bounded, and monotonically-increasing.

2.2.3.5 Network Connectivity and Information Flow

In fully-connected (dense) layers, each neuron in layer l receives input from every neuron in layer $l-1$. This creates a rich connectivity pattern where information can flow through multiple pathways. For a layer with n input neurons and m output neurons, this results in $n \times m$ weight parameters plus m bias parameters.

The forward pass through a multi-layer network involves sequential computation through each layer:

$$\mathbf{z}^{(1)} = \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)} \tag{2.14}$$

$$\mathbf{a}^{(1)} = f^{(1)}(\mathbf{z}^{(1)}) \tag{2.15}$$

$$\mathbf{z}^{(2)} = \mathbf{W}^{(2)}\mathbf{a}^{(1)} + \mathbf{b}^{(2)} \tag{2.16}$$

$$\mathbf{a}^{(2)} = f^{(2)}(\mathbf{z}^{(2)}) \tag{2.17}$$

$$\vdots \tag{2.18}$$

$$\mathbf{z}^{(L)} = \mathbf{W}^{(L)}\mathbf{a}^{(L-1)} + \mathbf{b}^{(L)} \tag{2.19}$$

$$\mathbf{a}^{(L)} = f^{(L)}(\mathbf{z}^{(L)}) \tag{2.20}$$

This hierarchical processing allows each layer to build upon the representations learned by previous layers, creating increasingly complex and abstract feature representations.

2.2.3.6 Backpropagation and Parameter Learning

Multi-layer perceptrons (MLP) networks became feasible only after the definition of backpropagation was formulated by Rumelhart, Hinton e Williams (1986), based on the work from Linnainmaa (1970). Backpropagation is an efficient algorithm for computing gradients of the loss function with respect to all network parameters using the chain rule of calculus.

The algorithm works by propagating error signals backward through the network. For a loss function $\mathcal{L}$, the gradient with respect to the weights in layer l is computed as:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(l)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{(l)}} \frac{\partial \mathbf{z}^{(l)}}{\partial \mathbf{W}^{(l)}} = \delta^{(l)} (\mathbf{a}^{(l-1)})^T \tag{2.21}$$

where $\delta^{(l)} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{(l)}}$ is the error term for layer l, computed recursively as:

$$\delta^{(l)} = ((\mathbf{W}^{(l+1)})^T \delta^{(l+1)}) \odot f'^{(l)}(\mathbf{z}^{(l)}) \tag{2.22}$$

where $\odot$ denotes element-wise multiplication and $f'^{(l)}$ is the derivative of the activation function in layer l.

Similarly, the gradient with respect to biases is:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(l)}} = \delta^{(l)} \tag{2.23}$$

Backpropagation allows MLPs to be trained based on labeled examples through trial and error in an autonomous way, establishing the modern concept of Artificial Neural Networks (ANNs). The efficiency of this algorithm, with computational complexity $O(W)$ where W is the total number of weights, made training deep networks practically feasible and laid the foundation for modern deep learning.

2.3 DEEP LEARNING

Deep Learning (DL) is a subfield of machine learning that leverages artificial neural networks with multiple hidden layers to automatically learn hierarchical representations from data (LECUN; BENGIO; HINTON, 2015). While the exact definition of "deep" remains subject to debate, it is generally accepted that networks with multiple layers (typically three or more hidden layers) constitute deep learning models (GOODFELLOW; BENGIO; COURVILLE, 2016).

The fundamental distinction between deep learning and traditional machine learning approaches lies in their approach to feature extraction and representation learning. Traditional machine learning methods rely heavily on manual feature engineering, where domain experts must identify and extract relevant features from raw data before applying learning algorithms. In contrast, deep learning models can automatically discover and learn hierarchical feature representations directly from raw data, reducing the need for extensive domain knowledge and manual preprocessing (BENGIO; COURVILLE; VINCENT, 2013).

2.3.1 Key Differences from Traditional Machine Learning

Deep learning distinguishes itself from conventional machine learning approaches across several critical dimensions:

2.3.1.1 Feature Engineering and Representation Learning

Traditional machine learning algorithms, such as Support Vector Machines (SVMs), Random Forests, and k-Nearest Neighbors, operate on handcrafted features extracted from raw data. This process requires significant domain expertise and often becomes the

bottleneck in developing effective machine learning systems. Deep learning models, particularly CNNs for image data, automatically learn hierarchical feature representations through multiple layers of non-linear transformations (LECUN et al., 1998).

For instance, in image classification tasks, the initial layers of a CNN typically learn to detect low-level features such as edges and textures, while deeper layers combine these features to recognize more complex patterns like shapes and objects. This hierarchical feature learning eliminates the need for manual feature design and often leads to superior performance on complex tasks.

2.3.1.2 Data Requirements and Scalability

Traditional machine learning algorithms often perform well with relatively small datasets and may plateau in performance as data volume increases. Deep learning models, conversely, typically require large amounts of training data to achieve optimal performance but continue to improve as more data becomes available (GOODFELLOW; BENGIO; COURVILLE, 2016). This characteristic makes deep learning particularly well-suited for applications where large datasets are available, such as computer vision and natural language processing.

The relationship between data size and model performance in deep learning follows a power law, where doubling the dataset size can lead to consistent improvements in model accuracy. This scalability advantage becomes more pronounced as the complexity of the task increases.

2.3.1.3 Computational Requirements

Deep learning models demand significantly more computational resources compared to traditional machine learning approaches. Training deep networks typically requires specialized hardware such as Graphics Processing Units (GPUs), Tensor Processing Units (TPUs), or other Application-Specific Integrated Circuits (ASICs) to handle the intensive matrix operations involved in forward and backward propagation (LECUN; BENGIO; HINTON, 2015). Traditional machine learning algorithms, in contrast, can often be trained efficiently on standard Central Processing Units (CPUs).

2.3.1.4 Model Interpretability and Explainable AI

Traditional machine learning models, such as decision trees and linear regression, offer inherent interpretability, allowing practitioners to understand the decision-making process. Deep learning models are often criticized as "black boxes" due to their complex architectures and numerous parameters, making it challenging to interpret how specific inputs lead to particular outputs (SAMEK; WIEGAND; MÜLLER, 2017). The field of Explainable Artificial Intelligence (XAI) has emerged to address this fundamental challenge, developing methods to make deep learning models more transparent and interpretable.

Motivation for Explainability: The need for explainability in AI systems stems from several critical requirements (GUNNING et al., 2019):

- **Trust and Accountability:** In high-stakes domains such as healthcare, finance, and criminal justice, stakeholders need to understand and validate AI decisions before acting on them.
- **Debugging and Improvement:** Understanding model behavior helps identify failure modes, biases, and opportunities for improvement.
- **Regulatory Compliance:** Regulations such as the European Union's General Data Protection Regulation (GDPR) include provisions for a "right to explanation" for automated decisions.
- **Scientific Discovery:** In scientific applications, understanding what models learn can lead to new insights and hypotheses.

Taxonomy of Explainability Methods: XAI techniques can be categorized along several dimensions (ADADI; BERRADA, 2018):

- **Intrinsic vs. Post-hoc:** Intrinsic methods build interpretability directly into the model architecture (e.g., attention mechanisms, sparse models), while post-hoc methods explain already-trained models.
- **Local vs. Global:** Local explanations describe model behavior for individual predictions, while global explanations characterize overall model behavior.

- **Model-specific vs. Model-agnostic:** Some methods are designed for specific architectures (e.g., Grad-CAM for CNNs), while others can be applied to any model (e.g., LIME, SHAP).

Key XAI Techniques for Deep Learning:

Gradient-based Methods: These techniques use gradients to identify which input features most influence the model's output. Gradient-weighted Class Activation Mapping (Grad-CAM) (SELVARAJU et al., 2017) produces visual explanations for CNN decisions by computing the gradient of the target class score with respect to feature maps, generating a localization map highlighting important regions in the input image.

Layer-wise Relevance Propagation (LRP): LRP (BACH et al., 2015) decomposes the prediction decision backward through the network layers, attributing relevance scores to each input feature based on its contribution to the final prediction.

Attention Mechanisms: Beyond their functional role in model architecture, attention weights provide inherent interpretability by showing which parts of the input the model focuses on when making predictions (VASWANI et al., 2017).

Saliency Maps: These visualizations highlight input regions that most strongly influence model predictions, computed through various methods including gradient-based approaches and perturbation-based techniques (SIMONYAN; VEDALDI; ZISSERMAN, 2013).

LIME and SHAP: Local Interpretable Model-agnostic Explanations (LIME) (RIBEIRO; SINGH; GUESTRIN, 2016) and SHapley Additive exPlanations (SHAP) (LUNDBERG; LEE, 2017) are model-agnostic frameworks that explain individual predictions by approximating the model locally with an interpretable model or by using game-theoretic approaches to attribute importance to features.

Challenges and Trade-offs: Despite significant progress, XAI faces ongoing challenges. There often exists a trade-off between model performance and interpretability, with simpler, more interpretable models sometimes achieving lower accuracy than complex black-box models. Additionally, different explanation methods can produce contradictory explanations for the same prediction, and evaluating the quality of explanations remains an open research question (DOSHI-VELEZ; KIM, 2017).

2.3.2 Training Deep Learning Models

Training deep learning models involves optimizing millions or billions of parameters through gradient-based optimization algorithms. The most fundamental component is the loss function, which quantifies the difference between predicted and actual outputs.

2.3.2.1 Loss Functions

The choice of loss function depends on the specific task:

Mean Squared Error (MSE): Used for regression tasks, MSE measures the average squared difference between predicted and actual values:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.24}$$

Cross-Entropy Loss: Commonly used for classification tasks, cross-entropy loss measures the dissimilarity between predicted probability distributions and true labels:

$$\mathcal{L}_{\text{CE}} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{C} y_{i,c}\log(\hat{y}_{i,c}) \tag{2.25}$$

where C is the number of classes, $y_{i,c}$ is the true label (1 if sample i belongs to class c, 0 otherwise), and $\hat{y}_{i,c}$ is the predicted probability for class c.

2.3.2.2 Optimization Algorithms

Modern deep learning relies on sophisticated optimization algorithms that extend beyond traditional gradient descent:

Adam Optimizer: Combines the advantages of AdaGrad and RMSprop, adapting learning rates for each parameter individually (KINGMA; BA, 2014):

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t \tag{2.26}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)g_t^2 \tag{2.27}$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{2.28}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{2.29}$$

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t}+\epsilon}\hat{m}_t \tag{2.30}$$

where g_t is the gradient at time t, m_t and v_t are exponential moving averages of gradients and squared gradients respectively, and β_1, β_2 are decay rates.

2.3.3 Data Representation and Encoding

Before data can be processed by machine learning models, it must be transformed into a numerical format that algorithms can interpret. This transformation process is fundamental to all machine learning pipelines, and different encoding strategies can significantly impact model performance.

2.3.3.1 One-Hot Encoding

One-hot encoding is a fundamental technique for representing categorical variables as binary vectors, widely used in both traditional machine learning and deep learning applications (GOODFELLOW; BENGIO; COURVILLE, 2016). This encoding scheme transforms a categorical variable with K possible values into a K-dimensional binary vector where exactly one element is 1 (hot) and all others are 0 (cold).

Formally, for a categorical variable x that can take one of K discrete values $\{v_1, v_2, ..., v_K\}$, the one-hot encoding function $\phi : \{v_1, ..., v_K\} \rightarrow \{0, 1\}^K$ maps each value to a unique binary vector:

$$\phi(v_i) = \mathbf{e}_i = [0, ..., 0, \underbrace{1}_{i\text{-th position}}, 0, ..., 0]^T \tag{2.31}$$

For example, consider a color attribute with three possible values: {red, green, blue}. The one-hot encoding would represent these as:

- red $\rightarrow [1, 0, 0]^T$
- green $\rightarrow [0, 1, 0]^T$
- blue $\rightarrow [0, 0, 1]^T$

Properties and Advantages: One-hot encoding possesses several important properties that make it suitable for machine learning applications. First, it eliminates any implicit ordering or magnitude relationships between categories, which is crucial when dealing with nominal variables where no natural ordering exists. Second, it creates

orthogonal representations where the dot product between any two different category vectors is zero, preventing the model from learning spurious relationships based on arbitrary numerical assignments.

Applications in Neural Networks: In the context of neural networks, one-hot encoding is particularly important for representing class labels in classification tasks. The final layer of a classification network typically uses a softmax activation function that outputs a probability distribution over classes, which can be directly compared to the one-hot encoded ground truth labels using cross-entropy loss (BISHOP, 2006).

Limitations: Despite its widespread use, one-hot encoding has notable limitations. For high-cardinality categorical variables (those with many unique values), one-hot encoding can lead to extremely high-dimensional sparse representations, increasing computational cost and memory requirements. Additionally, one-hot encoding does not capture any semantic similarity between categories—the encoding for "cat" is equally distant from "dog" as it is from "automobile," despite the former two being more semantically related. Modern approaches such as embedding layers in neural networks address these limitations by learning dense, continuous representations that can capture semantic relationships (BENGIO; COURVILLE; VINCENT, 2013).

2.3.4 Evaluation Metrics for Deep Learning Models

Proper evaluation of deep learning models requires careful selection of appropriate metrics based on the specific task and dataset characteristics.

2.3.4.1 Classification Metrics

Accuracy: The proportion of correctly classified instances:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.32}$$

Precision: The proportion of predicted positive instances that are actually positive:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.33}$$

Recall (Sensitivity): The proportion of actual positive instances that are correctly identified:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.34}$$

F1-Score: The harmonic mean of precision and recall, providing a balanced measure:

$$\text{F1} = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.35}$$

where TP, TN, FP, and FN represent true positives, true negatives, false positives, and false negatives, respectively.

Area Under the ROC Curve (AUC-ROC): Measures the model's ability to distinguish between classes across all classification thresholds. The ROC curve plots the true positive rate against the false positive rate, and the AUC provides a single scalar value summarizing performance across all thresholds.

2.3.4.2 Considerations for Metric Selection

The choice of evaluation metric depends heavily on the specific application domain and the relative costs of different types of errors. In medical diagnosis applications, recall might be prioritized to ensure that most actual disease cases are identified, even at the cost of increased false positives. Conversely, in spam detection, precision might be more important to avoid classifying legitimate emails as spam.

For multi-class classification problems, metrics can be computed using macro-averaging (computing metrics for each class and taking the unweighted mean) or micro-averaging (computing metrics globally by counting total true positives, false negatives, and false positives).

2.3.5 Deep Learning in Practice

The practical application of deep learning has been facilitated by several key developments:

Transfer Learning: Pre-trained models on large datasets (such as ImageNet) can be fine-tuned for specific tasks, significantly reducing training time and data requirements (PAN; YANG, 2009).

Regularization Techniques: Methods such as dropout, batch normalization, and weight decay help prevent overfitting and improve generalization (SRIVASTAVA et al., 2014).

Hardware Acceleration: The availability of specialized hardware and distributed computing frameworks has made training large-scale deep learning models feasible for a broader range of applications.

The success of deep learning across diverse domains—from computer vision and natural language processing to speech recognition and autonomous systems—has established it as a fundamental paradigm in modern artificial intelligence, particularly for tasks involving complex, high-dimensional data where traditional feature engineering approaches prove insufficient.

2.3.6 Zero-Shot and Few-Shot Learning

Zero-shot learning represents a paradigm where models recognize or classify instances from classes never seen during training, while few-shot learning allows models to adapt from very few examples per class (PALATUCCI et al., 2009; VINYALS et al., 2016). These approaches are particularly valuable when collecting labeled data for all classes is impractical. Recent vision-language models like CLIP (RADFORD et al., 2021) have demonstrated impressive zero-shot capabilities by learning joint embeddings of images and text from massive datasets.

While not directly employed in this thesis, these paradigms represent important developments in machine learning for scenarios with limited labeled data. For detailed technical information on zero-shot and few-shot learning methodologies, including problem formulations, meta-learning approaches, and applications, see Appendix C.

2.4 COMPUTER VISION

Computer Vision (CV) is a multidisciplinary subfield of artificial intelligence that enables machines to interpret, analyze, and understand visual information from the world. While early CV systems were inspired by the goal of emulating human vision (HUANG, 1996), the field has evolved to encompass capabilities that extend far beyond human visual perception, leveraging diverse sensing modalities and computational approaches to extract meaningful information from visual data.

Modern CV systems utilize a wide array of sensors and imaging modalities beyond traditional Red-Green-Blue (RGB) images. These include depth sensors (RGB-D cam-

eras), thermal and Infrared (IR) imaging, multispectral and hyperspectral sensors, Light Detection and Ranging (LiDAR) systems, and event-based cameras (BOCK et al., 2020). Each sensing modality captures different aspects of the electromagnetic spectrum or spatial information, enabling applications that surpass human visual capabilities. For instance, hyperspectral imaging can identify materials based on their spectral signatures across hundreds of wavelengths, while thermal imaging reveals temperature distributions invisible to human eyes.

The paradigm shift from merely mimicking human vision to developing machine vision systems with unique capabilities has opened new frontiers in applications such as autonomous navigation, medical imaging, remote sensing, and precision agriculture (JEZ et al., 2021). These systems can process information across multiple spectral bands simultaneously, detect rapid temporal changes imperceptible to humans, and analyze scenes with precision and consistency that exceed human performance.

In practice, CV encompasses a variety of tasks, each targeting different aspects of visual perception and interpretation. These include, but are not limited to, edge detection, image classification, object detection, semantic segmentation, and instance segmentation. Each task presents its own set of challenges and requires distinct algorithmic approaches and data representations. Figure 4 illustrates some of the most prominent tasks within the domain of computer vision.

This work focuses on two specific and highly relevant subfields of CV: image segmentation and image classification. These tasks have undergone significant methodological advancements in recent years, particularly with the advent of deep learning-based approaches (WU; SAHOO; HOI, 2020). While acknowledging the broader spectrum of sensing modalities available in modern CV, this thesis specifically concentrates on RGB imaging, which remains the most widely accessible and commonly used imaging modality in practical applications. In the following subsections, we elaborate on the theoretical foundations and practical implications of each task.

2.4.1 Image Segmentation

Image segmentation refers to the process of partitioning an image into semantically meaningful regions, allowing for the identification and delineation of objects and their boundaries within a scene. In the context of CV, "semantically meaningful" refers to

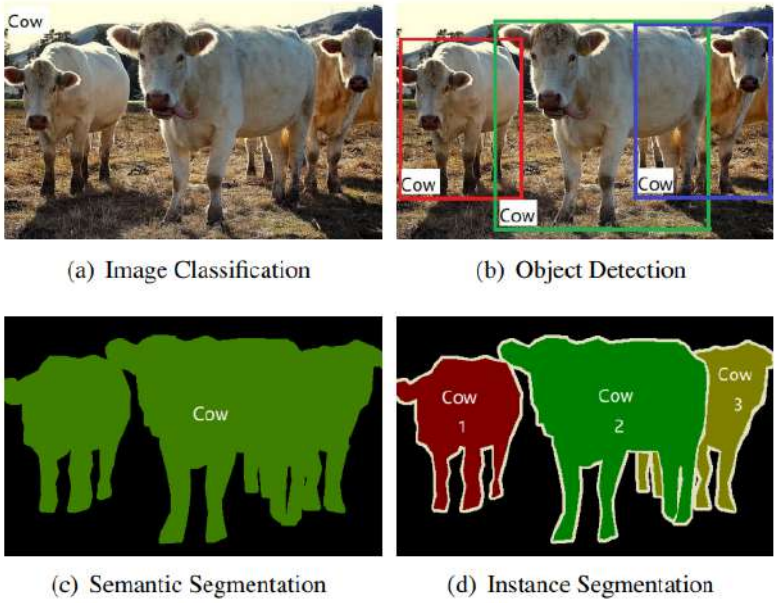


Figure 4 – Representative tasks in computer vision. (a) *Image classification* involves assigning a categorical label to an entire image. (b) *Object detection* extends classification by localizing each object instance using bounding boxes. (c) *Semantic segmentation* classifies each pixel in the image, typically without distinguishing between different instances of the same class. (d) *Instance segmentation* further refines semantic segmentation by identifying and separating individual instances of objects at the pixel level. Source: (WU; SAHOO; HOI, 2020).

regions that correspond to distinct objects, parts, or concepts that have interpretable significance within the visual scene. For example, in a natural scene, semantically meaningful regions might include individual trees, buildings, people, or sky areas—each representing conceptually coherent entities rather than arbitrary pixel groupings based solely on low-level visual features like color or texture.

At its core, the goal of segmentation is to assign a label to every pixel in an image such that pixels sharing the same label exhibit similar visual or semantic properties and belong to the same conceptual entity. This process transforms raw pixel data into structured, interpretable representations that facilitate higher-level scene understanding.

Two primary paradigms of segmentation are widely studied:

- **Semantic segmentation:** This technique categorizes each pixel in the image into a predefined class, without regard for the number of object instances. For example, all pixels corresponding to trees in a forest scene would be labeled identically, regardless of whether they belong to the same or different trees.

- **Instance segmentation:** While also assigning class labels at the pixel level, instance segmentation goes further by distinguishing between separate objects

of the same class. This allows the model to assign different labels to individual instances (e.g., two people in a crowd), thereby providing a finer level of granularity.

These segmentation tasks are crucial for applications requiring detailed scene understanding, such as autonomous navigation, medical image analysis, and agricultural monitoring.

2.4.2 Image Classification

Image classification is one of the most fundamental problems in CV. It involves assigning one or more labels to an entire image, indicating the presence of particular objects, concepts, or scene categories. The process typically assumes a supervised learning framework, where a model is trained on a dataset composed of labeled images and subsequently evaluated on its ability to correctly label previously unseen images.

The classification task can be approached in two main paradigms:

- **Single-label classification:** In this approach, each image is assigned exactly one label from a predefined set of mutually exclusive categories. This is suitable for scenarios where images contain a single dominant subject or where the task requires choosing the most representative category.
- **Multi-label classification:** This more complex approach allows images to be associated with multiple labels simultaneously, acknowledging that real-world images often contain multiple objects, concepts, or attributes. For instance, an image might be labeled as containing both "trees" and "buildings" and "sky," reflecting the multi-faceted nature of natural scenes.

Formally, given an input image, a single-label classification model outputs the most probable class from a predefined set of categories, while a multi-label classification model outputs a subset of labels that are deemed present in the image. Classical approaches relied on handcrafted features combined with statistical classifiers. However, modern techniques predominantly leverage CNNs, which have demonstrated remarkable performance in large-scale image classification benchmarks without the need for explicit feature extraction, such as ImageNet (RUSSAKOVSKY et al., 2015).

Image classification serves as the backbone for more complex CV tasks such as object detection, facial recognition, scene categorization, and content-based image retrieval. Its robustness and efficiency make it an essential component in both research and industry applications. The fundamental representations learned during classification tasks often transfer effectively to other CV problems, making it a cornerstone technique in the field.

2.5 CONVOLUTIONAL NEURAL NETWORKS

CNNs are a class of deep learning architectures specifically designed to process data with a grid-like topology, such as images. Unlike fully-connected networks that treat inputs as flat vectors, CNNs exploit the spatial structure of input data through localized receptive fields and shared weights. This design enables CNNs to capture spatial hierarchies and patterns (e.g., edges, textures, shapes) with fewer parameters, leading to better generalization and computational efficiency. Their architectural principles are inspired by the organization of the visual cortex in animals, where individual neurons respond to stimuli in localized regions of the visual field (LECUN et al., 1998). These properties make CNNs particularly effective for visual recognition tasks and are foundational in modern computer vision systems.

2.5.1 Mathematical Foundations of Convolutional Operations

The core operation in CNNs is the convolution, which applies a set of learnable filters (kernels) across the input to extract local features. For a 2D input $\mathbf{X} \in \mathbb{R}^{H \times W}$ and a filter $\mathbf{K} \in \mathbb{R}^{k_h \times k_w}$, the convolution operation is mathematically defined as:

$$(\mathbf{X} * \mathbf{K})_{i,j} = \sum_{m=0}^{k_h-1} \sum_{n=0}^{k_w-1} \mathbf{X}_{i+m,j+n} \cdot \mathbf{K}_{m,n} \tag{2.36}$$

where i and j are the spatial coordinates of the output feature map. For multi-channel inputs, such as RGB images with C channels, the convolution extends to:

$$(\mathbf{X} * \mathbf{K})_{i,j} = \sum_{c=0}^{C-1} \sum_{m=0}^{k_h-1} \sum_{n=0}^{k_w-1} \mathbf{X}_{c,i+m,j+n} \cdot \mathbf{K}_{c,m,n} + b \tag{2.37}$$

where b is the bias term. The key advantages of convolution include **parameter sharing** (the same filter is applied across all spatial locations), **translation equivariance** (shifting the input results in a correspondingly shifted output), and **sparse connectivity** (each output unit is connected only to a local region of the input).

2.5.2 Core Architectural Components

A typical CNN architecture is composed of several types of layers, each serving specific purposes in the feature extraction and classification pipeline:

2.5.2.1 Convolutional Layers

Convolutional layers apply multiple filters to extract various features from the input. Each filter produces a feature map that highlights specific patterns. The output dimensions of a convolutional layer depend on the input size $(H \times W)$, filter size $(k_h \times k_w)$, stride s, and padding p:

$$H_{\text{out}} = \left\lfloor \frac{H + 2p - k_h}{s} \right\rfloor + 1, \quad W_{\text{out}} = \left\lfloor \frac{W + 2p - k_w}{s} \right\rfloor + 1 \tag{2.38}$$

2.5.2.2 Pooling Layers

Pooling layers reduce the spatial dimensions of feature maps while retaining important information. The most common types are:

Max Pooling: Selects the maximum value within each pooling window:

$$\text{MaxPool}(\mathbf{X})_{i,j} = \max_{m,n \in \mathcal{N}_{i,j}} \mathbf{X}_{m,n} \tag{2.39}$$

Average Pooling: Computes the average value within each pooling window:

$$\text{AvgPool}(\mathbf{X})_{i,j} = \frac{1}{|\mathcal{N}_{i,j}|} \sum_{m,n \in \mathcal{N}_{i,j}} \mathbf{X}_{m,n} \tag{2.40}$$

where $\mathcal{N}_{i,j}$ represents the pooling neighborhood around position (i, j).

2.5.2.3 Normalization Techniques

Modern CNNs incorporate normalization techniques to stabilize training and improve convergence:

Batch Normalization: Normalizes inputs across the batch dimension, reducing internal covariate shift and enabling higher learning rates (IOFFE; SZEGEDY, 2015):

$$\text{BN}(\mathbf{x}) = \gamma \frac{\mathbf{x} - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta \tag{2.41}$$

where μ_B and σ_B^2 are the batch mean and variance, and γ and β are learnable parameters.

2.5.3 Architectural Innovations and Deep Networks

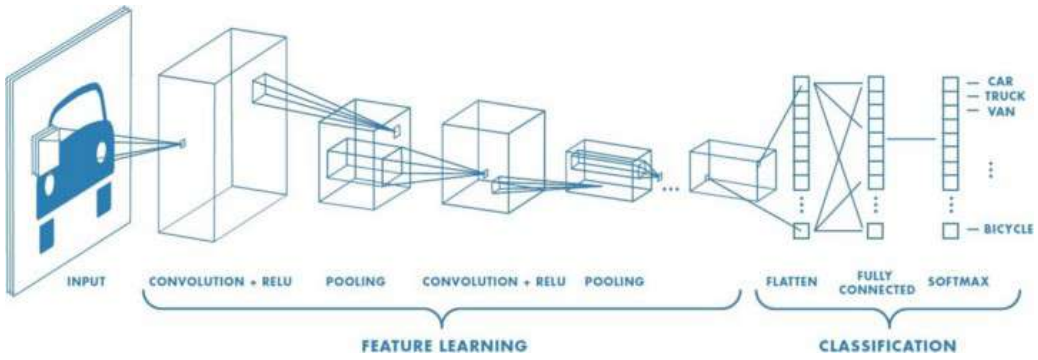


Figure 5 – Illustration of a typical CNN architecture, showing convolutional, pooling, and fully-connected layers. Source: (SAHA, 2018).

The hierarchical feature extraction mechanism inherent in CNNs allows them to outperform traditional machine learning models and standard multilayer perceptrons in image-related tasks. This ability to learn relevant patterns directly from raw pixel data without manual feature engineering has been instrumental in driving advancements across a wide range of computer vision applications.

2.5.3.1 Historical Evolution and Landmark Architectures

One of the earliest CNN architectures to achieve notable success was LeNet-5, introduced by LeCun et al. (1998). Designed to recognize handwritten digits from the MNIST dataset, LeNet-5 consisted of seven layers, including two convolutional layers,

two subsampling (pooling) layers, and three fully-connected layers. Despite its simplicity by today's standards, LeNet-5 laid the groundwork for many subsequent developments in deep learning.

A significant milestone in the evolution of CNNs came in 2012 with the introduction of AlexNet by Krizhevsky, Sutskever e Hinton (2017). AlexNet was a deeper and more computationally intensive architecture, composed of eight layers—five convolutional layers, followed by three fully-connected layers. It was the first deep CNN to achieve a substantial breakthrough in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), outperforming traditional approaches by a large margin and effectively popularizing deep learning in the computer vision community.

One of the key innovations behind AlexNet's success was its pioneering use of Graphics Processing Units (GPUs) to accelerate training. At the time, training deep neural networks on large-scale datasets like ImageNet was computationally prohibitive using only CPUs. AlexNet was among the first models to leverage GPU acceleration effectively, making it feasible to train a large network on over one million high-resolution images within a reasonable time frame. Moreover, due to memory limitations on individual GPUs, the authors employed a multi-GPU setup, partitioning the model across two NVIDIA GTX 580 GPUs. This allowed the training process to be parallelized, significantly reducing training time while enabling the use of a larger and deeper architecture than would otherwise have been possible. This approach also marked an important shift in the field, as it demonstrated the viability and necessity of GPU-based computing for training deep neural networks at scale, setting a precedent for nearly all modern deep learning research that followed.

AlexNet also introduced several key innovations including the use of ReLU activation functions instead of traditional sigmoid or tanh functions, dropout regularization to prevent overfitting (SRIVASTAVA et al., 2014), and data augmentation techniques to artificially expand the training dataset.

2.5.3.2 Residual Networks and Skip Connections

A fundamental challenge in training very deep networks is the vanishing gradient problem, where gradients become exponentially small as they propagate backward through many layers, making it difficult to train the early layers effectively. This limitation

was dramatically addressed with the introduction of Residual Networks (ResNets) by He et al. (2016a) in 2015.

ResNet architectures introduced the concept of **residual connections** or **skip connections**, which allow information to flow directly from earlier layers to later layers, bypassing intermediate transformations. The core idea is to reformulate the learning problem from learning a direct mapping $\mathcal{H}(\mathbf{x})$ to learning a residual mapping $\mathcal{F}(\mathbf{x}) = \mathcal{H}(\mathbf{x}) - \mathbf{x}$, with the final output being:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x} \tag{2.42}$$

where $\mathcal{F}(\mathbf{x}, \{W_i\})$ represents the residual function learned by the stacked layers, and the $+\mathbf{x}$ term is the identity shortcut connection.

The mathematical intuition behind residual learning is that it is easier to optimize the residual mapping $\mathcal{F}(\mathbf{x})$ than the original unreferenced mapping $\mathcal{H}(\mathbf{x})$. In the extreme case, if an identity mapping is optimal, it is easier to push the residual to zero than to fit an identity mapping by a stack of nonlinear layers.

The gradient flow through residual connections can be analyzed by considering the backward pass. For a residual block, the gradient of the loss with respect to the input is:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}} = \frac{\partial \mathcal{L}}{\partial \mathbf{y}} \left(1 + \frac{\partial \mathcal{F}}{\partial \mathbf{x}} \right) \tag{2.43}$$

The “1” term ensures that gradients can flow directly through the shortcut connection, preventing the vanishing gradient problem even in very deep networks. This innovation enabled the training of networks with hundreds or even thousands of layers, achieving state-of-the-art performance on the ImageNet dataset and securing first place in the ILSVRC 2015.

2.5.3.3 Advanced Architectural Patterns

Following the success of ResNet, several architectural innovations have further advanced the field:

Dense Connections: DenseNet (HUANG et al., 2017) extends the concept of skip connections by connecting each layer to every subsequent layer within a dense block, promoting feature reuse and reducing the number of parameters.

Inception Modules: The Inception architecture (SZEGEDY et al., 2015) introduced the concept of multi-scale feature extraction within a single layer by applying convolutions of different kernel sizes in parallel and concatenating the results.

Efficient Architectures: Modern architectures like EfficientNet (TAN; LE, 2019) and MobileNets (HOWARD et al., 2017) focus on achieving optimal trade-offs between accuracy and computational efficiency through techniques such as depthwise separable convolutions and neural architecture search.

2.5.4 Regularization and Training Techniques

Modern CNN training incorporates several regularization techniques to improve generalization:

Dropout: Randomly sets a fraction of input units to zero during training, preventing complex co-adaptations and reducing overfitting (SRIVASTAVA et al., 2014):

$$\mathbf{y} = \mathbf{r} \odot \mathbf{x} \tag{2.44}$$

where $\mathbf{r}$ is a random binary mask with probability p of being 1.

Data Augmentation: Artificially expands the training dataset through transformations such as rotation, scaling, cropping, and color jittering, improving model robustness and generalization.

Transfer Learning: Leverages pre-trained models on large datasets (such as ImageNet) as starting points for specific tasks, significantly reducing training time and data requirements (PAN; YANG, 2009).

These landmark architectures and innovations—from LeNet-5's foundational principles through AlexNet's breakthrough performance to ResNet's revolutionary skip connections—not only demonstrate the evolution of CNNs over time but also highlight their growing complexity, expressive power, and widespread applicability in modern visual recognition systems. The mathematical foundations and architectural innovations discussed provide the theoretical basis for understanding how CNNs achieve their remarkable performance in computer vision tasks.

2.6 VISION TRANSFORMERS

The application of Transformer architectures to computer vision represents a paradigm shift that has fundamentally challenged the dominance of convolutional neural networks in visual tasks. Originally developed for natural language processing by Vaswani et al. (2017), Transformers have demonstrated remarkable success when adapted to visual domains, leading to the emergence of Vision Transformers as a powerful alternative to traditional CNN-based approaches.

2.6.1 Encoder-Decoder Architectures and Attention

The encoder-decoder architecture represents a fundamental paradigm for sequence-to-sequence tasks, where an encoder compresses input into a context representation and a decoder generates output conditioned on this representation (SUTSKEVER; VINYALS; LE, 2014). The introduction of the attention mechanism by Bahdanau, Cho e Bengio (2014) significantly improved these architectures by allowing decoders to selectively focus on different input parts rather than relying on a single fixed-length context vector. This innovation laid the groundwork for modern Transformer architectures.

While this thesis employs CNNs with hard parameter sharing for multi-task learning rather than encoder-decoder architectures, understanding attention mechanisms provides important context for the development of Vision Transformers. For detailed information on encoder-decoder architectures, RNNs (LSTMs and GRUs), and attention mechanisms, see Appendix D.

2.6.2 Vision Transformers: Overview

Vision Transformers (ViT) represent the application of Transformer architectures—originally developed for natural language processing (VASWANI et al., 2017)—to computer vision tasks (DOSOVITSKIY et al., 2020). The key innovation lies in treating images as sequences of patches, enabling the use of self-attention mechanisms for visual recognition.

2.6.2.1 Core Architecture

ViT divides an input image into fixed-size patches, linearly embeds each patch, adds position embeddings, and processes the resulting sequence through a standard Transformer encoder. The self-attention mechanism allows each patch to attend to all other patches, providing a global receptive field from the first layer—a key difference from CNNs that build receptive fields gradually.

2.6.2.2 Inductive Biases and Data Requirements

Unlike CNNs, which incorporate strong inductive biases for visual data (translation equivariance, locality, hierarchical feature learning), ViT has minimal built-in assumptions about image structure. This reduced inductive bias requires substantially larger datasets to achieve good generalization but allows more flexible representation learning. When pretrained on large datasets like ImageNet-21k or JFT-300M, ViT can match or exceed CNN performance. However, performance degrades significantly when training from scratch on smaller datasets.

2.6.2.3 Application in This Thesis

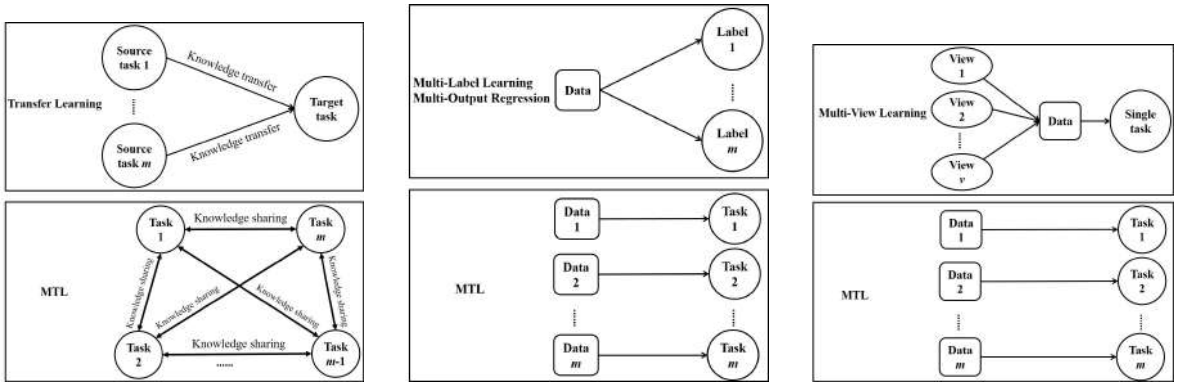
In this research, Vision Transformers (ViT-B/16 and ViT-B/32) were benchmarked alongside CNN architectures for the binary disease classification task. While ViT demonstrated competitive performance, ResNet-50V2 was ultimately selected as the backbone architecture due to its superior stability across varied data distributions, more efficient training on the available dataset size, and better balance between feature representation quality and computational requirements for deployment.

For detailed technical information on Vision Transformer architecture, self-attention mechanisms, patch-based processing, and comparative analysis with CNNs, see Appendix E.

2.7 MULTI-TASK LEARNING

Multi-task learning (MTL) is a machine learning paradigm in which multiple related tasks are learned jointly, with the goal of improving generalisation performance by exploiting shared structure among tasks (CARUANA, 1997). In contrast to single-task learning, where a model is trained for each task independently, MTL encourages the learning of common representations or parameters, enabling positive transfer of knowledge across tasks.

The core motivation for MTL is grounded in the observation that tasks often exhibit shared underlying patterns. For example, in domains with limited annotated data, MTL can act as a form of inductive transfer, regularising the learning process through the implicit use of auxiliary tasks. Theoretical work has shown that MTL reduces the risk of overfitting proportionally to the number of jointly learned tasks (BAXTER, 1997). This is particularly advantageous in data-scarce settings, where task-specific models might otherwise generalise poorly.



(a) MTL vs. transfer learning (b) MTL vs. multi-label learning/-multioutput regression (c) MTL vs. multi-view learning

Figure 6 – Illustrations of differences between MTL and other learning paradigms. (a) MTL vs. transfer learning: In transfer learning, knowledge is transferred from one or more source tasks to a specific target task, with the target task being the main focus. In contrast, MTL involves mutual knowledge sharing among all tasks, with no single task prioritized. (b) MTL vs. multi-label learning/multi-output regression: Multi-label learning and multi-output regression involve predicting multiple labels or outputs for the same data, whereas in MTL, each task may have its own data and objective, and tasks are learned jointly but not necessarily on the same dataset. (c) MTL vs. multi-view learning: Multi-view learning combines different sets of features (views) for a single task, while MTL addresses multiple tasks, each potentially with its own data and goal. These diagrams highlight how MTL is distinct in its approach to task relationships, data sharing, and learning objectives. Source: Adapted from (ZHANG; YANG, 2021).

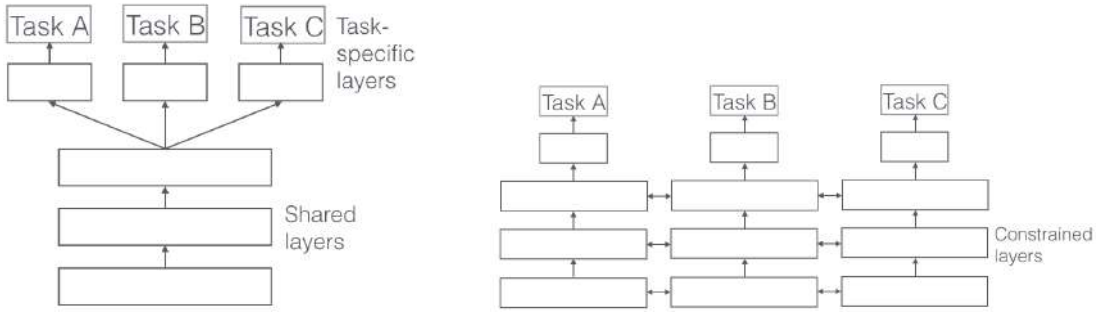
Figure 6 illustrates the main differences between MTL and related paradigms. Unlike transfer learning, where knowledge flows from source tasks to a specific target task,

MTL aims to improve all tasks jointly through mutual knowledge sharing. In contrast to multi-label learning and multi-output regression, where all tasks share the same data, MTL typically involves different data for each task. Finally, while multi-view learning combines multiple feature sets for a single task, MTL addresses multiple tasks, each potentially with its own data and objective.

2.7.1 Architectural Strategies and Taxonomy

MTL models are generally divided into two major design patterns: *feature-based* and *parameter-based* approaches (ZHANG; YANG, 2021). In feature-based methods, tasks share common representations, often in the form of learned embeddings or transformations of the input. Parameter-based approaches, by contrast, enforce constraints or regularization across task-specific model parameters.

A well-known architectural distinction is between **hard parameter sharing** and **soft parameter sharing** (CRAWSHAW, 2020). In the former, the initial layers of a network are shared across all tasks, followed by task-specific output heads. In the latter, each task has its own model, but parameter similarity is enforced through regularization. These strategies are visualised in Figure 7.



(a) MTL model with hard parameter sharing.

(b) MTL model with soft parameter sharing.

Figure 7 – Illustration of parameter sharing strategies in multi-task learning. In hard parameter sharing (a), the hidden layers are shared among all tasks, with separate task-specific output layers. This approach reduces the risk of overfitting by forcing the model to learn representations useful for all tasks. In soft parameter sharing (b), each task has its own model and parameters, but similarity between parameters is encouraged through regularization. This allows for flexibility in task-specific modeling while still promoting knowledge sharing.

Beyond these general designs, the MTL literature identifies five major methodological categories (ZHANG; YANG, 2021):

- **Feature learning approaches:** These focus on learning shared representations across tasks via either transformation or feature selection.
- **Low-rank approaches:** These assume that the task parameter matrix has low rank, capturing task relatedness via latent subspaces.
- **Task clustering approaches:** These group similar tasks into clusters and often combine shared and individual representations.
- **Task relation learning:** These methods attempt to learn quantitative task relationships, such as covariance matrices, directly from the data.
- **Decomposition approaches:** These decompose the task parameter matrix into multiple components (e.g., shared vs. task-specific).

2.7.2 Benefits and Flexibility

MTL offers several compelling advantages. First, shared representations act as an inductive bias, often yielding improved generalisation when tasks are sufficiently related. Second, MTL models are more modular: new tasks can be added by attaching additional heads to the shared body of the model. Third, MTL can be more parameter-efficient, reusing shared weights across tasks resulting in more efficient training and inference.

Additionally, by jointly training on more data points across tasks, MTL often learns more robust and transferable features. The survey in Zhang e Yang (2021) highlights how MTL has been adapted to various settings, including high-dimensional data and heterogeneous feature spaces. It also covers streaming or distributed learning environments.

2.7.3 Practical Considerations and Challenges

Despite its strengths, MTL introduces several nontrivial challenges:

- **Negative transfer:** When unrelated or conflicting tasks are trained jointly, performance can deteriorate. Designing mechanisms to detect and prevent negative transfer remains an open challenge.

- **Task balancing and interference:** Shared parameters can lead to conflicting gradient directions during training. Methods such as dynamic loss reweighting or gradient surgery are often employed to mitigate this issue.
- **Model versioning and organisational concerns:** In production systems, such as those described in Karpathy (2019), shared models may complicate coordination among teams managing different tasks, especially as architectures evolve or branch.

The question of what to share and how to share it is central to the design of MTL models. In feature-based approaches, this involves deciding whether to learn shared features directly or to select a subset from the original input space. In parameter-based models, this typically takes the form of structured regularization or Bayesian priors. Some advanced architectures combine these ideas—learning feature transformations and task relationships simultaneously through matrix factorisation, graph-based regularization, or probabilistic modeling.

2.7.4 Beyond the Standard Setting

Modern formulations of MTL extend beyond the classic setup. For instance, some approaches operate under *heterogeneous MTL*, where tasks differ not only in objectives but also in feature spaces, modalities, or data distributions. Others combine MTL with paradigms such as reinforcement learning, active learning, or semi-supervised learning, enabling broader applicability across domains with limited labeled data or streaming inputs.

Moreover, as highlighted in Zhang e Yang (2021), the scalability of MTL is increasingly important. Online and distributed MTL methods have been developed to handle large numbers of tasks and high-dimensional data, leveraging parallelisation and feature hashing to improve efficiency without compromising performance.

3 RELATED WORKS

This chapter surveys relevant literature that underpins the proposed use of multi-task learning for comprehensive leaf disease analysis. We categorize existing works based on the specific subtasks tackled by our proposed system: leaf detection, disease classification, symptom identification, agent recognition, and final disease classification.

3.1 SUPERVISED TRAINING OF A SIMPLE DIGITAL ASSISTANT FOR A FREE CROP CLINIC

Barros et al. (2021) proposed a mobile digital assistant aimed at aiding smallholder farmers in Pernambuco, Brazil, by providing disease diagnostics for plant leaves using a deep learning system. The assistant was built to support the operations of the local phytopathology clinic (CliFiPe) and enable scalable, low-cost expert support for family farmers. The system enables users to upload leaf images via a mobile app, receive preliminary classification results, and optionally consult with human experts. The application also fosters a collaborative community among farmers and phytopathologists.

The disease classification pipeline comprises three main stages: (1) image cropping to focus on the leaf region; (2) image segmentation into 4 or 6 patches depending on the aspect ratio; and (3) classification using a CNN based on ResNet50V2 pretrained weights. This segmentation step both augments the dataset and enhances the model's ability to localize symptoms. Segments with the highest confidence of symptom presence are highlighted and forwarded to experts for further diagnosis. Figures 8 and 9 show the system and pipeline architecture, respectively.

To train the CNN, the authors collected a novel field dataset of grape leaves from smallholder farms in Pernambuco. The images were labeled by experts as showing "Symptoms" or "No symptoms", yielding 3289 images (1302 symptomatic and 1987 asymptomatic). After segmentation, the training set included over 14,000 patches. Extensive hyperparameter tuning was performed, and training was repeated with various splits and class weightings to maximize recall while maintaining generalizability.

The final model achieved a recall of over 95%, a critical metric for reducing false negatives in this decision-support setting. The system also showed promising accuracy

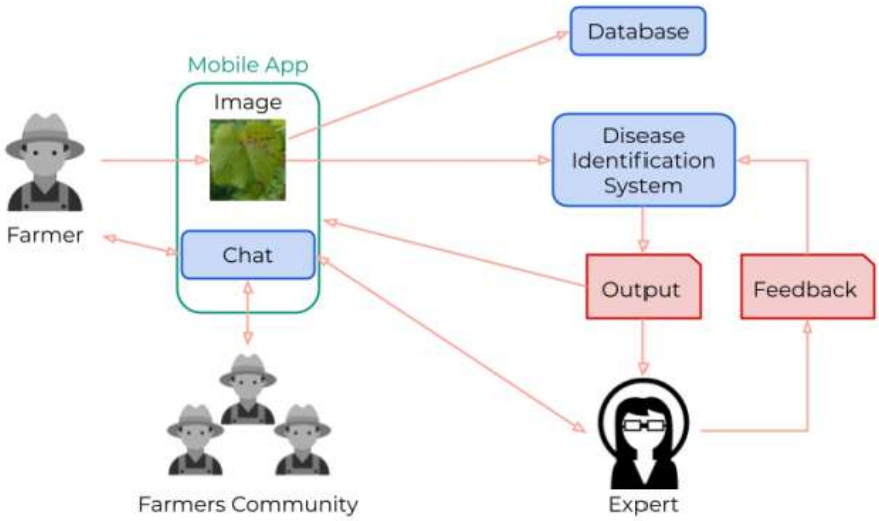


Figure 8 – General architecture of the proposed system by Barros et al. (2021), composed of a mobile app and a digital assistant for a crop clinic

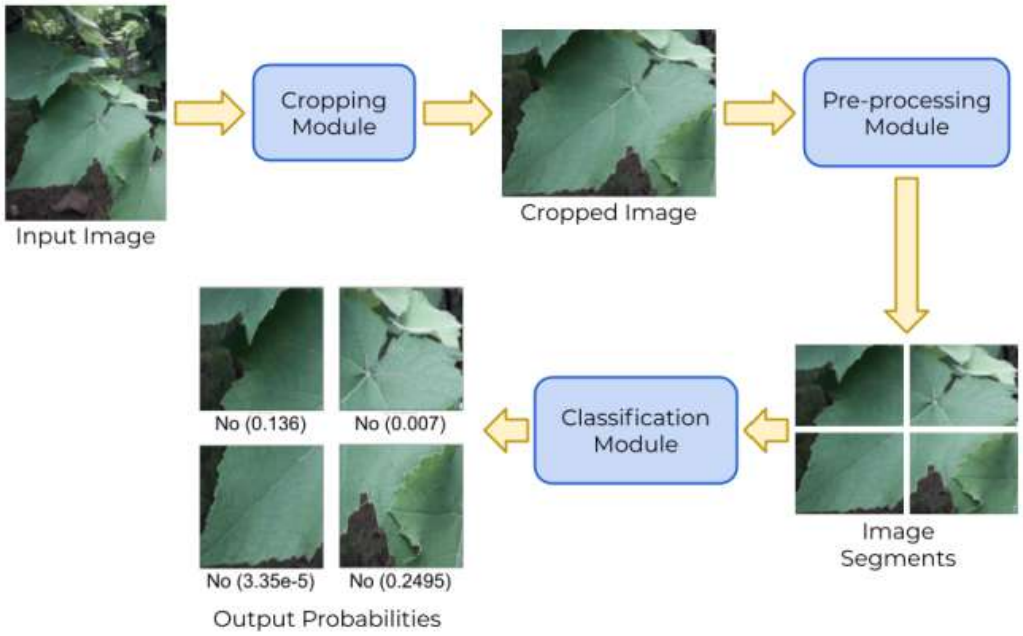


Figure 9 – Task pipeline proposed by (BARROS et al., 2021)

and F1-scores across several dataset configurations. Importantly, the dataset and assistant were designed with real-world variability in mind, including lighting conditions, framing, and image quality typical of non-expert users.

The current work builds upon this assistant platform but proposes a novel multi-task learning architecture that expands beyond binary classification. New task heads are added for leaf detection, symptom classification, disease agent identification, and final disease diagnosis. The training dataset is augmented with additional crops and conditions beyond grapes, and the preprocessing pipeline is extended to support the new tasks. These enhancements aim to bring the system closer to expert-level diagnosis and broader crop coverage.

3.2 USING DEEP LEARNING FOR IMAGE-BASED PLANT DISEASE DETECTION

Mohanty, Hughes e Salathé (2016) demonstrated the feasibility of using deep learning for automated plant disease classification through leaf images. The authors leveraged the PlantVillage dataset (HUGHES; SALATHE, 2015), which consists of 54,306 images across 14 crop species and 26 disease conditions (including healthy leaves), encompassing a total of 38 unique crop-disease combinations. This dataset was curated under controlled conditions to ensure uniformity in background, lighting, and image quality. Figure 10 shows examples of the various crop-disease combinations included in this dataset.

To evaluate performance, the authors trained two widely used CNN architectures: AlexNet and GoogLeNet. Both models were tested under various configurations: training from scratch versus transfer learning from ImageNet, and using different versions of the dataset—color, grayscale, and segmented leaves. Transfer learning consistently outperformed models trained from scratch, and GoogLeNet outperformed AlexNet across all metrics. The best performing configuration (GoogLeNet with transfer learning on the color dataset using an 80–20 train-test split) achieved an overall accuracy of 99.35% and a mean F1 score of 0.9934, highlighting the promise of CNN-based classification even with minimal preprocessing.

Despite these impressive results, the study identified major limitations when transferring the model to real-world settings. When tested on small curated datasets from online sources mimicking field conditions, the model's top-1 accuracy dropped signifi-

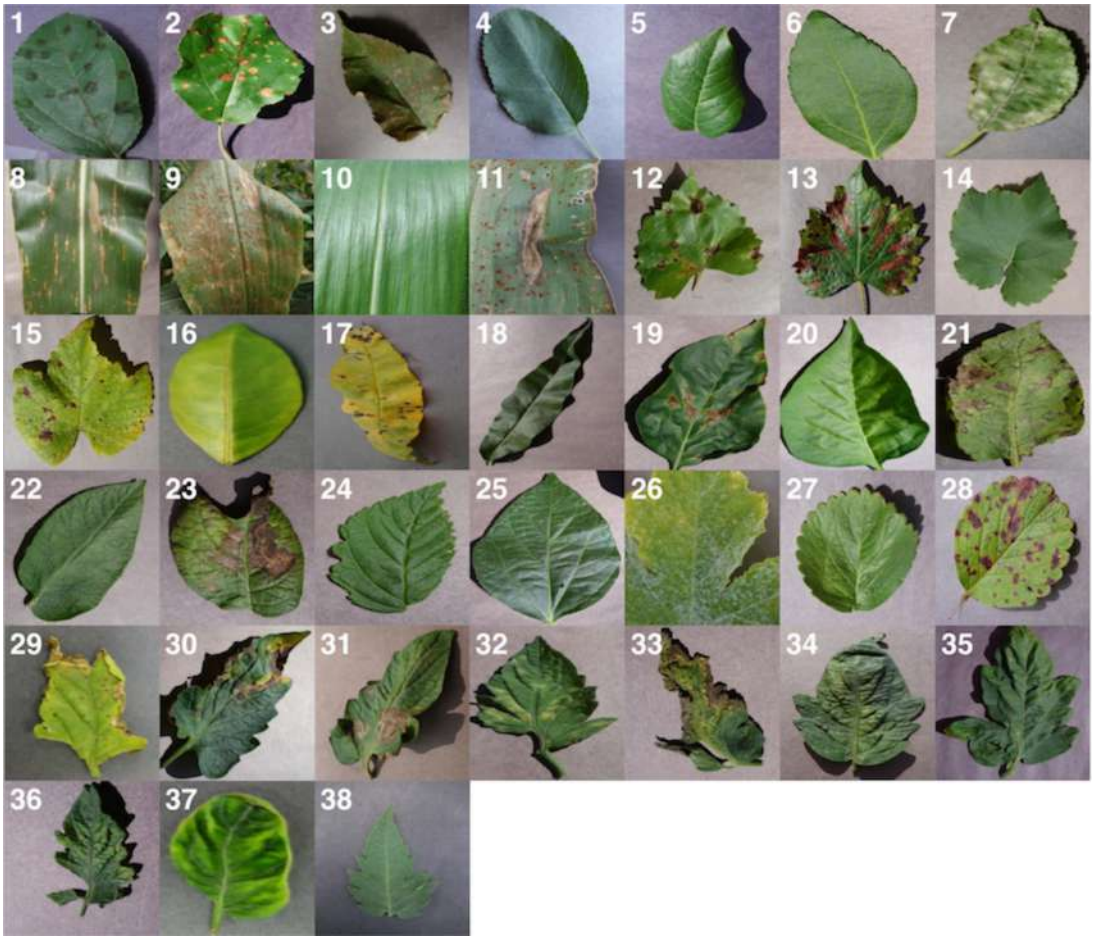


Figure 10 – Examples of leaf images from the PlantVillage dataset (HUGHES; SALATHE, 2015), showing various crop-disease combinations used in the study by Mohanty, Hughes e Salathé (2016)

cantly to around 31%, despite being well above random guessing. This suggests that models trained solely on PlantVillage do not generalize well to in-the-wild images due to the lack of visual variability in the training data (e.g., complex backgrounds, lighting inconsistencies, leaf overlap, or disease progression stages).

Additionally, the paper highlighted that their task formulation—joint classification of both crop species and disease status—may be more complex than necessary for practical deployment. As most farmers already know which crop they are growing, separating disease diagnosis from crop classification could simplify the problem and improve real-world performance.

Nevertheless, the work by Mohanty, Hughes e Salathé (2016) laid the groundwork for subsequent research in this area. It introduced an open-access, large-scale dataset and set the benchmark for disease recognition accuracy using CNNs. Their segmentation experiments also support the idea that removing background noise can improve model robustness—an insight reflected in our current work's preprocessing steps.

Our approach expands on these ideas by integrating multi-task learning, allowing the model to not only identify the presence of disease but also output secondary predictions such as symptom type and causal agent. Furthermore, unlike the PlantVillage dataset, our data pipeline includes images collected under varied real-world conditions, enhancing generalizability and addressing one of the core limitations identified in Mohanty, Hughes e Salathé (2016).

3.3 MULTI-LABEL LEARNING FOR CROP LEAF DISEASES RECOGNITION AND SEVERITY ESTIMATION

Ji et al. (2020) proposed a comprehensive multi-label learning framework for the automatic recognition of crop diseases and the estimation of disease severity. Their model, named BR-CNN, is based on the binary relevance (BR) strategy combined with deep CNNs, and it simultaneously classifies crop species, disease types, and severity levels.

The authors used the AI Challenger dataset, comprising 12,691 RGB images annotated with 20 distinct labels across 7 crop species, 10 disease types (including healthy), and 3 severity levels (normal, general, serious). Each image may contain multiple labels, such as (Potato, Late Blight, Serious). To address this, they applied a multi-label classification approach using CNN backbones including InceptionV3, ResNet50, DenseNet121, and NasNet.

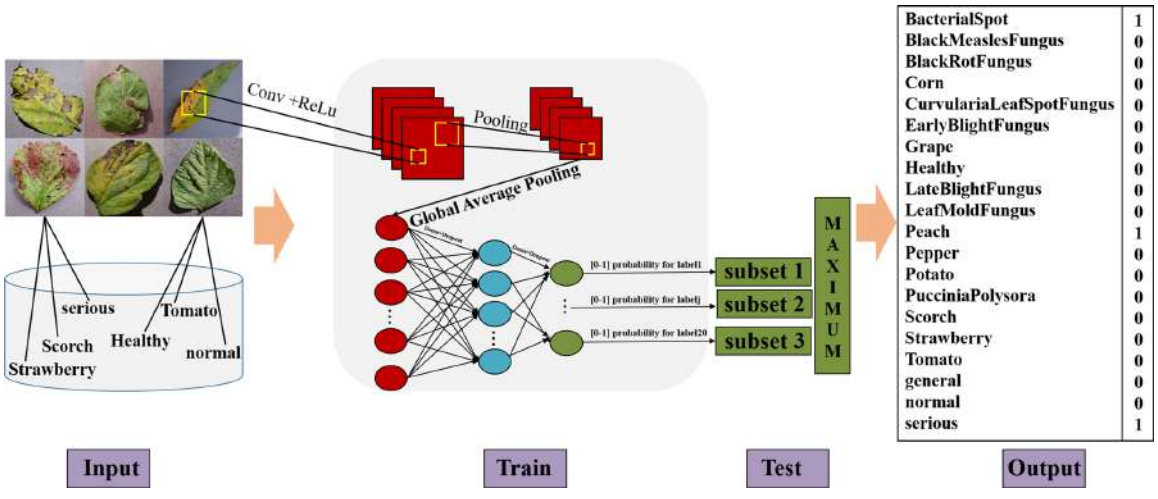


Figure 11 – Pipeline of the proposed BR-CNN for crop leaf diseases recognition and severity estimation. The sequential steps of the proposed BR-CNN for crop leaf diseases recognition and severity estimation are shown in this block diagram, adapted from Ji et al. (2020).

The block diagram in Figure 11 illustrates the sequential steps of the BR-CNN pipeline for crop leaf diseases recognition and severity estimation, as proposed by Ji et al. (2020).

The BR-CNN framework transforms the multi-label task into a set of binary classification problems—one for each label—and uses sigmoid activation at the output layer to compute label probabilities independently. Among the tested backbones, BR-CNN with ResNet50 achieved the best performance for severity estimation, with an accuracy score (AS) of 86.70% and a Jaccard similarity score (JSS) of 92.93%. For disease recognition, DenseNet121 yielded the best performance, with AS and JSS reaching 97.88% and 98.45%, respectively.

Their findings indicate that fine-grained severity estimation is more challenging than disease classification, particularly due to the subjectivity in severity labeling (e.g., distinguishing "general" from "serious" damage). Nonetheless, their BR-CNN models significantly outperformed single-label or label-powerset baselines in both accuracy and model efficiency. Moreover, lightweight models such as BR-CNN with NasNet showed promise for mobile deployment, aligning with trends in edge-based agricultural diagnostics.

Figure 12 demonstrates different classification frameworks based on multi-label learning algorithms and deep learning architectures, as discussed by Ji et al. (2020). The comparison includes:

- **LP-CNN**: The traditional single-label method for crop diseases recognition, as in Wang, Sun e Wang (2017), which joins all labels (crop species, disease types, and severity kinds) as a unified class and uses a softmax classifier for multi-class prediction.

- **MLP-CNN**: Proposed in Ji et al. (2020), MLP-CNN fuses the LP multi-label learning algorithm with deep CNNs. It transforms the multi-label problem into an ensemble of multi-class classification problems, where each component learner is a CNN trained on a subset of labels. Each subset returns the label with the maximum probability, using a softmax classifier. While this approach can better isolate overlapping attribute spaces, it adds complexity and can cause computational bottlenecks on large datasets.

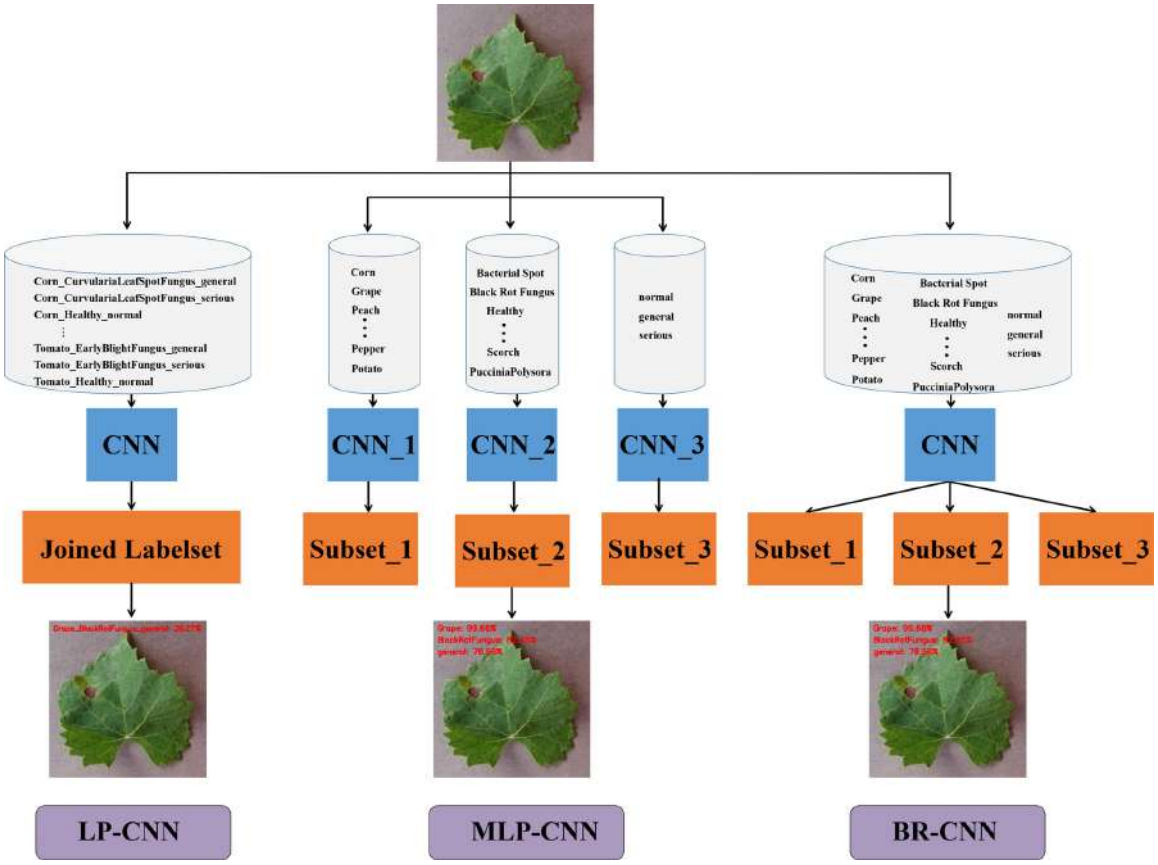


Figure 12 – Illustration and comparison of different classification frameworks for crop leaf diseases recognition and severity estimation. Figure adapted from Ji et al. (2020). The comparison includes LP-CNN (traditional single-label method, as in Wang, Sun e Wang (2017)), MLP-CNN (multi-label powerset CNN, proposed in Ji et al. (2020)), and BR-CNN (binary relevance CNN, also proposed in Ji et al. (2020)).

- **BR-CNN**: Also proposed in Ji et al. (2020), BR-CNN transforms the multi-label task into a set of binary classification problems (one per label), using sigmoid activations for independent probability estimation.

Our proposed system builds upon similar motivations but integrates the disease classification and severity estimation into a unified MTL model rather than separate binary classifiers. This design encourages feature sharing across tasks, reduces parameter redundancy, and streamlines deployment. Furthermore, while Ji et al. (2020)'s model treated each task independently, our approach enables joint optimization of disease presence, symptom type, pathogen agent, and disease class, enhancing the model's consistency and scalability.

3.4 FACTORS INFLUENCING THE USE OF DEEP LEARNING FOR PLANT DISEASE RECOGNITION

Barbedo (2018) presented a comprehensive study investigating the primary factors that affect the design and performance of deep learning models applied to plant disease recognition. While deep learning has shown promising results in plant pathology, the study highlights that these results often come from experiments performed under highly controlled conditions that do not reflect real agricultural settings. As such, the author emphasizes the importance of critically examining model assumptions, dataset construction, and deployment feasibility.

The experiments were conducted using a subset of the Digipathos dataset, containing images of nine corn leaf diseases, all caused by fungi. The study compared CNN performance across different preprocessing strategies: original full images, background-removed images, and subdivided images focused on symptomatic regions. To increase the dataset size and test the CNN's performance with more localized information, the original samples were divided into smaller images containing individual lesions or localized symptom regions (Figure 13). Notably, the subdivided dataset achieved the highest accuracy (87%), compared to 76% for the original and 79% for the background-removed images. This suggests that localizing symptoms improves the model's capacity to extract relevant features.

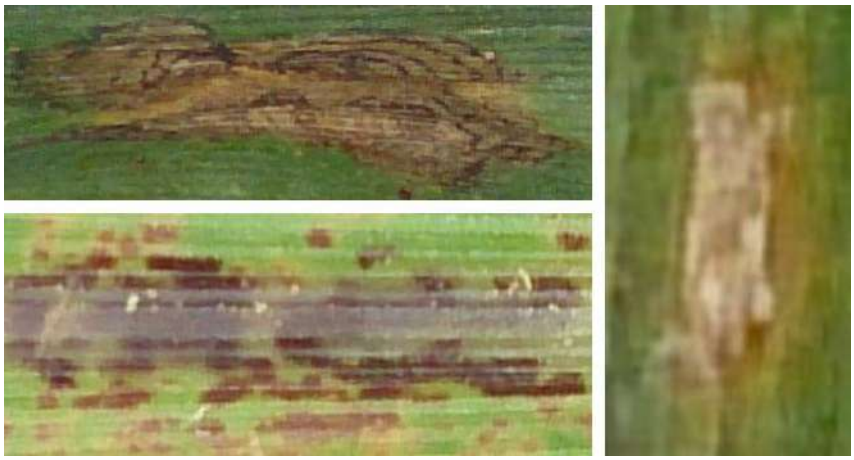


Figure 13 – Examples of subdivided images from Barbedo (2018), showing individual symptomatic regions. Top left: anthracnose; bottom left: physoderma brown spot; right: tropical rust.

Based on these results, Barbedo (2018) identified nine key factors influencing CNN-based plant disease recognition, as detailed in Table 1. The first five factors are related to

the dataset construction and how representative it is of the real-world problem, whereas the last four are intrinsically connected to how the model should be trained to take into account the challenges present in the problem of symptom and disease classification. Our current work addresses several of these intrinsic challenges through its multi-task learning architecture and preprocessing pipeline.

Factor	**Impact**
Limited annotated datasets	Datasets do not have enough samples for deep neural networks to properly learn the classes. Annotation errors may damage the learning process.
Symptom representation	Datasets do not adequately represent the symptom variety found in practice, weakening the robustness of the trained model.
Covariate shift	Training and testing a model using the same dataset often leads to unrealistic performance assessment, as the model will likely fail when applied to other datasets.
Image background	Image background may contain elements that may disturb the training process, especially if those elements are present in multiple samples.
Image capture conditions	Images can be captured in a wide variety of conditions. In order to be representative, a dataset has to contemplate all possibilities, which is currently unfeasible.
Symptom segmentation	Images with many spurious elements often cause difficulties for the model. Taking more localized regions of the leaf may prevent some problems.
Symptom variations	Symptoms produced by a disease may present a wide range of characteristics. It is difficult to build datasets capable of representing symptom diversity properly.
Simultaneous disorders	It is difficult to detect multiple simultaneous disorders when images are analysed as a whole. Adopting localized symptom regions may mitigate this problem.
Disorders with similar symptoms	Some disorders produce visually similar symptoms. In cases like this, simple RGB images may not be enough for proper recognition, even with well-trained models.

Frame 1 – Factors impacting the performance of CNNs for plant disease recognition. Table adapted from (BARBEDO, 2018).

These insights are directly relevant to the current work, which adopts a symptom-focused, multi-task learning architecture. While our approach does not implement image subdivision as suggested (BARBEDO, 2018), it not only faced the afore mentioned challenges but also addresses some of them through alternative strategies. Our preprocessing pipeline handles background noise through careful image curation and augmentation techniques during training, and proper image segmentation during inference. The multi-task architecture explicitly models symptom variations and simultaneous

disorders by outputting separate predictions for disease presence, symptom types, and causal agents, being augmented during its training to overcome the lack of proper labels on the original datasets. This fine-grained approach helps distinguish between visually similar disorders by leveraging complementary information across different diagnostic subtasks. Additionally, our dataset construction process incorporates varied capture conditions to enhance model robustness in real-world agricultural settings.

Ultimately, Barbedo (2018)'s work serves as both a critique of overly optimistic claims in the literature and a roadmap for constructing practical, scalable disease detection systems. His emphasis on real-world variability, dataset rigor, and interpretability is reflected in our model's multi-output structure and dataset curation process.

3.5 REAL-TIME GRAPE LEAF DISEASE DETECTION USING AN IMPROVED CNN

Xie et al. (2020) proposed a real-time detection system for grape leaf diseases using an enhanced convolutional neural network architecture named Faster DR-IACNN. The model was designed to address challenges in detecting small, dense, and diverse lesion patterns on grape leaves, particularly under varied environmental conditions and complex backgrounds. The authors introduced several architectural improvements to increase detection accuracy and speed while enabling real-time inference.

To support the model, the authors built the Grape Leaf Disease Dataset (GLDD), composed of 4,449 original images representing four common diseases: Black rot, Black measles, Leaf blight, and Mites of grape. After data augmentation—including changes in brightness, contrast, sharpness, and rotations—the dataset was expanded to 62,286 images.

The proposed architecture modifies Faster R-CNN by integrating an enhanced feature extraction backbone called INSE-ResNet, which combines ResNet34 with Inception-v1, Inception-ResNet-v2 modules, and SE-blocks. These additions enable multiscale feature learning and enhance the model's ability to detect small and irregular disease spots. A novel double Region Proposal Network (double-RPN) module was also introduced to improve localization precision across scales.

Figure 14 illustrates the overall framework of the Faster DR-IACNN model, which integrates a pre-network for extracting disease image features, a Region Proposal Network for locating diseased spots, and fully connected layers for final classification

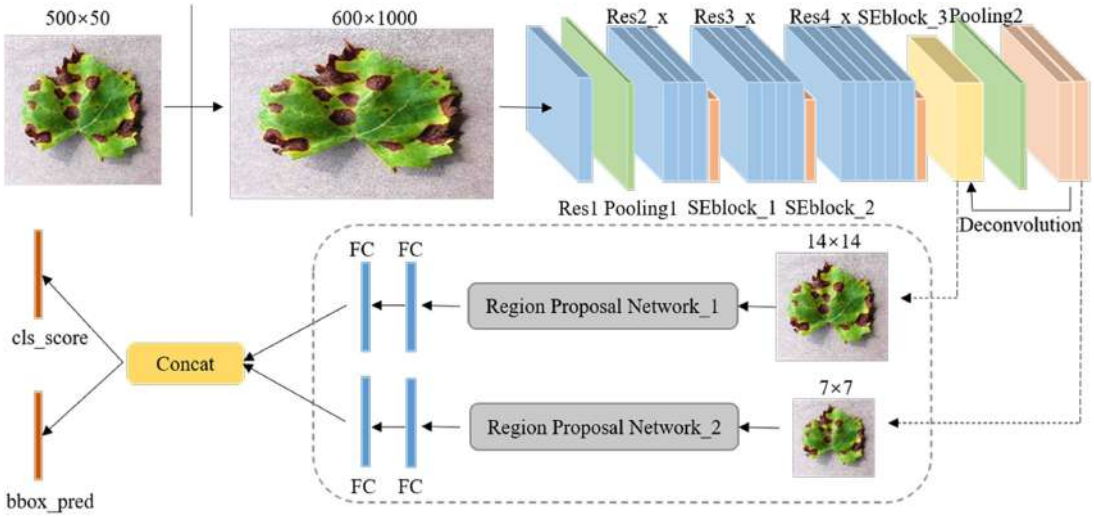


Figure 14 – Overall structure of the Faster DR-IACNN model proposed by (XIE et al., 2020). The architecture consists of three main components: (1) INSE-ResNet backbone for feature extraction with residual structures, inception modules, and SE-blocks; (2) Region Proposal Network (RPN) for object localization; and (3) fully connected layers for classification and regression.

and regression. The INSE-ResNet backbone combines residual structures with inception modules and SE-blocks to widen the receptive field and obtain multiscale features, while the double-RPN enhances localization precision across different scales.

Experimental results showed that Faster DR-IACNN achieved a mean Average Precision (mAP) of 81.1% on GLDD, outperforming several classical models such as SSD, R-FCN, and Faster R-CNN with standard backbones. Additionally, it maintained a real-time inference speed of 15.01 frames per second (FPS), satisfying practical demands in vineyard environments.

Figure 15 demonstrates the detection capabilities of the Faster DR-IACNN model on grape leaves with various diseases. The results show that the model can detect not only multiple diseased spots of the same disease type in one leaf (Figures A-D) but also multiple spots of different diseases on a single leaf simultaneously (Figure E). Most detection boxes achieve confidence scores greater than 0.99, demonstrating high detection precision and accurate localization. This robust performance across diverse disease patterns and mixed infections highlights the model's strong generalization capabilities.

While the model focused on object detection using bounding boxes, its modular backbone and disease-specific design offer relevant insights for our work. Our proposed model complements this detection-focused approach by tackling disease diagnosis through multi-task classification. Instead of localizing lesions, we aim to assign semantic

labels for leaf presence, disease status, symptom types, causal agent, and disease class. Nevertheless, Xie et al. (2020)'s methodological contributions—particularly in model augmentation and multiscale feature extraction—inform the architectural decisions in our work, especially in designing shared layers for multiple diagnostic tasks.

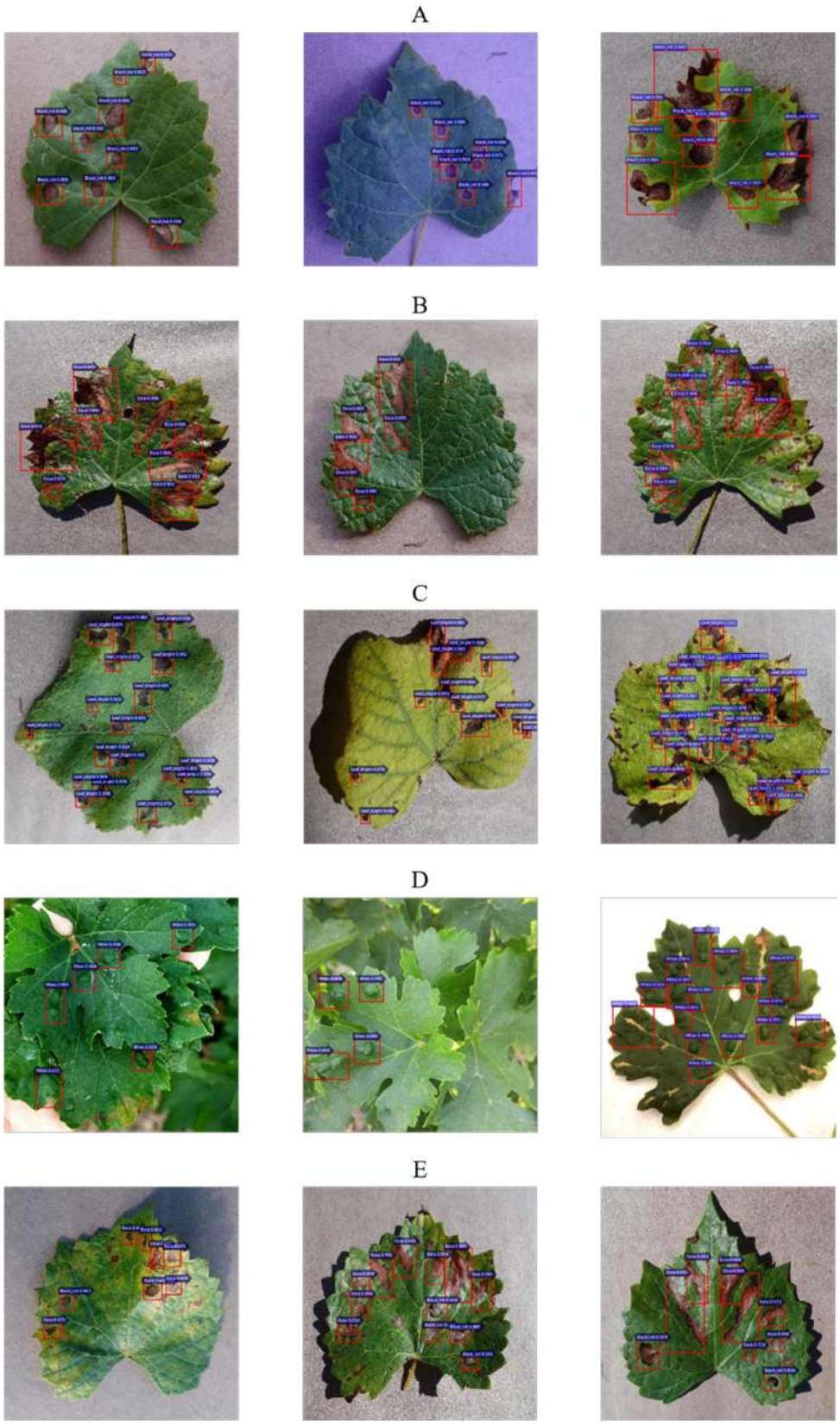


Figure 15 – Grape leaf diseased spots detection results by Xie et al. (2020). (A) Multiple Black rot spots in one leaf. (B) Multiple Black measles spots in one leaf. (C) Multiple Leaf blight spots in one leaf. (D) Multiple Mites of grape spots in one leaf. (E) Diversified diseased spots in one leaf with multiple disease types detected simultaneously.

3.6 DEEP LEARNING FOR FRUIT DETECTION IN CITRUS ORCHARDS

Neto et al. (2019) proposed a computer vision system based on deep convolutional neural networks to detect and count green oranges directly from digital images captured in citrus orchards. Their goal was to support yield estimation efforts in a way that is more scalable, automated, and economically feasible than traditional labor-intensive methods.

The study employed the YOLOv3 architecture for object detection, using an enhanced version of the model implemented by AlexeyAB, which included several optimizations such as improved detection accuracy, reduced training time, and efficient memory allocation. The network architecture was composed of 106 layers, including convolutional, shortcut, upsample, and detection layers, distributed across three detection scales (13×13, 26×26, and 52×52 grid cells).

The dataset used for training consisted of 2,035 manually annotated images cropped to 416×416 pixels. These were collected from various citrus fields and captured using heterogeneous devices (e.g., smartphones and digital cameras) under uncontrolled lighting conditions to better reflect real-world variability. Of these, 1,832 images were used for training, 203 for validation, and a separate set of 1,030 images was reserved for testing. All images were annotated manually to mark visible fruit regions, accounting for challenges such as shadow occlusion, overlapping leaves, and various fruit maturation stages.

During training, 490,000 iterations were performed. The model's performance was evaluated using standard metrics such as precision, recall, F1-score, and mean Intersection over Union (mIoU). The final model achieved a precision of 0.95, recall of 0.82, F1-score of 0.88, and mIoU of 78.2% on the test set. Additional experiments using sliding windows over full-tree images revealed that the system was highly sensitive to input resolution and benefited from window sizes approximating the input dimensions of the network.

Figure 16 demonstrates the detection capabilities of the YOLOv3-based system on citrus trees with multiple green fruits. The results show that the model can effectively detect fruits even under challenging conditions such as shadow occlusion, overlapping leaves, and various fruit maturation stages. The bounding boxes indicate high-confidence detections, with the system achieving excellent precision while main-

Figure 16 – Detection results of green fruits using a YOLOv3-based model in citrus orchards. The image shows the detection of multiple green oranges with bounding boxes, demonstrating the model's ability to identify fruits under natural field conditions with varying occlusion and lighting. The system successfully detected 208 fruits with no false positives and 21 false negatives in this example, showcasing the effectiveness of YOLO-based architectures for agricultural object detection tasks. Figure adapted from Neto et al. (2019).

taining reasonable recall despite the complex visual environment typical of agricultural settings.

This work highlights the feasibility and effectiveness of using YOLO-based models for fruit detection tasks under natural field conditions, providing an important precedent for integrating deep learning in agricultural monitoring. Although it focuses on fruit detection rather than disease classification, the system's ability to handle image variability and occlusion informs our current work's approach to handling field-acquired images of plant leaves. Specifically, their methodology of training with diverse, annotated real-world images aligns with our strategy for creating a robust, generalizable multi-task model.

3.7 GRAPE CLUSTER DETECTION AND SEGMENTATION FOR PREPROCESSING

Santos et al. (2020) proposed a comprehensive methodology for grape cluster detection and segmentation using deep neural networks, with a particular focus on enhancing downstream applications like yield estimation and automated harvesting. Their work introduced the Embrapa Wine Grape Instance Segmentation Dataset (WGISD), a public dataset with over 4,000 annotated grape clusters, including both bounding boxes and instance segmentation masks.

To address the inherent challenges of grape imagery—such as occlusions, variable lighting, and irregular fruit morphology—they evaluated two prominent architectures: YOLO (v2 and v3) (REDMON; FARHADI, 2017; REDMON; FARHADI, 2018) and Mask R-CNN (HE et al., 2017). Their findings showed that while YOLO offered faster inference, Mask R-CNN achieved superior detection and segmentation performance, reaching an F1-score of 0.91 for instance segmentation at 30% intersection over union (IoU) and maintaining competitive performance even at higher thresholds. These results underscore the efficacy of instance segmentation in precisely localizing clusters, which is particularly beneficial for applications requiring fine-grained fruit delineation. Figure 17 illustrates the comparative detection results across different grape varieties, demonstrating Mask R-CNN's superior precision in the segmentation of clusters of grapes compared to YOLO variants.

Beyond detection, their work implemented a robust annotation tool based on interactive image segmentation using attributed relational graphs, streamlining the labor-intensive process of generating high-quality instance masks. Additionally, they proposed a 3D spatial registration mechanism using structure-from-motion to track and match grape clusters across video frames, enabling accurate fruit counting while mitigating double-counting due to occlusions. To evaluate the model's generalization capabilities, they tested Mask R-CNN on novel scenarios without any parameter tuning, as shown in Figure 18. The results demonstrated robust performance across different camera poses, developmental stages, and environmental conditions, indicating the model's strong adaptability to real-world agricultural settings.

The insights and tools developed by Santos et al. (2020) are directly relevant to our preprocessing pipeline. Their segmentation approach provides a foundation for isolating informative regions of interest (ROIs) in complex grapevine images, which can

Figure 17 – Object detection results comparison between Mask R-CNN, YOLOv2, and YOLOv3 on grape cluster detection, adapted from (SANTOS et al., 2020). The figure demonstrates the varying performance of each architecture in identifying, localizing, and segmenting grape clusters under real-world vineyard conditions, with Mask R-CNN showing superior precision in cluster localization compared to YOLO variants.

be fed into our multi-task classification model. By leveraging their high-fidelity masks or applying similar segmentation strategies, our system can focus on disease-prone areas, reducing noise from background elements and improving classification robustness under real-world vineyard conditions.

This segmentation methodology is particularly valuable for real-life agricultural scenarios where farmers often capture images containing multiple leaves or plant parts in a single photograph. The instance segmentation capabilities demonstrated by Santos et al. (2020) enable our system to automatically detect and separate individual leaves within such composite images, effectively splitting a single input containing multiple leaves into separate, focused images for individual analysis. This preprocessing step is crucial for maintaining classification accuracy, as it allows our multi-task model to

Figure 18 – Mask R-CNN generalization on novel scenarios without parameter tuning, adapted from (SANTOS et al., 2020). The figure shows detection results on images with different camera poses (a-b), varying leaf textures (c), and different developmental stages (d), demonstrating the model's robust generalization capabilities across diverse real-world conditions. Images are Creative Commons-licensed from public sources.

process each leaf independently, avoiding the confusion that might arise from analyzing multiple leaves simultaneously. By applying similar segmentation techniques to our leaf detection task, we can ensure that each leaf receives focused attention during disease classification, symptom identification, and agent recognition, thereby improving the overall diagnostic precision of our system in real-world field conditions.

Furthermore, this approach directly addresses several critical factors identified by Barbedo (2018) that impact CNN performance in plant disease recognition. Specifically, the segmentation methodology complements Barbedo (2018)'s findings by effectively mitigating the "Image background" factor, which he identified as a major challenge where background elements can disturb the training process. By isolating individual leaves through instance segmentation, we eliminate background noise and spurious elements that could interfere with disease classification. Additionally, this preprocessing strategy partially helps to overcome the "Simultaneous disorders" challenge. Even though it does not solve the issue where a single leaf may contain multiple disorders, it alleviates the issue by ensuring that each leaf is analyzed independently, reducing the complexity of detecting multiple disorders when images contain overlapping or adjacent leaves. The segmentation-based approach thus provides a practical solution to some of the intrinsic challenges that Barbedo (2018) identified as limiting factors in real-world plant disease recognition systems.

3.8 STATE OF THE ART IN PLANT DISEASE DETECTION SYSTEMS

Despite significant advances in deep learning and computer vision for plant disease detection, the field has yet to converge on an optimal, generalizable solution that addresses the diverse needs of agricultural practitioners worldwide. Multiple comprehensive reviews of the literature reveal a persistent pattern: most publicly available models and research efforts focus on specific crop types or particular diseases, limiting their applicability to broader agricultural contexts (ANNABEL; ANNAPOORANI; DEEPALAKSHMI, 2019; SARKAR et al., 2023).

Annabel, Annapoorani e Deepalakshmi (2019) surveyed various machine learning techniques for plant leaf disease detection and classification, highlighting the variety of approaches employed across different crops and pathogens. Their review emphasizes that while numerous classification techniques—including Support Vector Machines, Artificial Neural Networks, and various feature extraction methods—have shown promising results on specific datasets, these solutions typically lack the cross-crop and cross-disease generalization necessary for practical deployment at scale. The requirement for extensive domain knowledge and manual feature engineering further limits the accessibility and scalability of many proposed systems.

More recent surveys confirm this trend continues even with the adoption of deep learning approaches. Sujatha et al. (2021) compared the performance of traditional machine learning methods (Support Vector Machine, Random Forest, Stochastic Gradient Descent) against deep learning architectures (Inception-v3, VGG-16, VGG-19) for citrus plant disease detection. Their results demonstrated that deep learning methods consistently outperform traditional machine learning approaches, with VGG-16 achieving the highest classification accuracy of 89.5%. However, their evaluation was limited to citrus diseases, and the study did not assess model generalization to other crop species or disease types.

Sarkar et al. (2023) conducted a comprehensive review covering publications from 2010 to 2022, analyzing both machine learning and deep learning approaches for leaf disease detection across multiple crops. Their analysis revealed that while models such as CNN, VGG, and ResNet demonstrate strong performance on specific datasets, significant challenges remain in creating systems that generalize across different imaging conditions, crop varieties, and pathogen types. The authors identified several persistent issues: the predominance of controlled laboratory conditions in dataset creation, limited cross-dataset validation, and the scarcity of models capable of handling multiple crop species within a unified framework.

Recent specialized approaches continue this pattern of crop-specific solutions. Vallabhajosyula, Sistla e Kolli (2024) proposed a novel hierarchical framework combining improved Vision Transformer and ResNet9 architectures, achieving strong performance on the PlantVillage and Extended PlantVillage datasets across 13 to 51 disease classes. Similarly, Kulkarni e Shastri (2024) developed a CNN-based system specifically for rice leaf disease detection, achieving 95% accuracy on rice-specific disease classification. While these works represent important advances in model architecture and performance optimization, they exemplify the field's tendency toward crop-specific solutions rather than generalizable frameworks.

The gap between academic research and practical deployment is perhaps most evident when examining commercial solutions. Plantix (PEAT GmbH, 2025), developed by PEAT GmbH, represents one of the most widely deployed agricultural disease detection applications, with over 100 million crop-related queries processed globally (HAMPF et al., 2021). The application claims to diagnose diseases across multiple crop species and provide treatment recommendations through automated image analysis. Despite

its widespread adoption and reported success in field conditions, the underlying model architecture, training methodology, and validation procedures remain proprietary and unavailable for peer review or independent verification. This lack of transparency makes it difficult for the research community to assess the system's true capabilities, limitations, and generalization performance across diverse agricultural contexts.

This situation highlights a critical gap in the current landscape: while proprietary commercial solutions claim broad applicability across crops and diseases, they lack the scientific transparency necessary for validation and improvement by the broader research community. Conversely, publicly available academic models—though peer-reviewed and reproducible—typically focus on narrow problem domains that limit their practical utility for farmers and agricultural extension services who must deal with diverse crops and pathogens.

3.9 COMPARATIVE ANALYSIS: SINGLE-TASK VS MULTI-TASK APPROACHES

This chapter presented different models and approaches to address plant disease recognition and related agricultural computer vision challenges. Although they share similar objectives—automated diagnosis and monitoring of plant health—the models differ significantly in their task formulation, architectural design, output predictions, and real-world applicability. This section provides a comparative analysis of the reviewed works and positions our proposed multi-task learning approach within the existing literature.

The surveyed approaches can be broadly categorized into four main paradigms: (1) **single-task classification** focusing on binary or multi-class disease identification; (2) **multi-label classification** treating each attribute (crop, disease, severity) as independent binary problems; (3) **object detection and segmentation** for localizing diseased regions or plant organs; and (4) **multi-task learning** (our proposed approach) that jointly optimizes multiple related diagnostic subtasks.

Single-task approaches, exemplified by Barros et al. (2021) and Mohanty, Hughes e Salathé (2016), demonstrate strong performance on their specific objectives but are limited in scope. Barros et al. (2021)'s binary classification (symptoms vs. no symptoms) achieves high recall but provides minimal diagnostic information for expert consultation. Similarly, Mohanty, Hughes e Salathé (2016)'s joint crop-disease classification, while

achieving 99.35% accuracy on PlantVillage, fails to generalize to real-world conditions due to the controlled nature of the training data and the overly complex joint formulation.

Multi-label approaches, particularly Ji et al. (2020)'s BR-CNN framework, address multiple attributes simultaneously but treat each prediction independently using separate binary classifiers. While this approach achieved strong performance (97.88% accuracy for disease recognition), it suffers from parameter redundancy and lacks the feature sharing benefits that could improve consistency across related predictions. The binary relevance strategy also ignores potential correlations between tasks, such as the relationship between symptom types and causal agents.

Object detection and segmentation approaches, represented by Xie et al. (2020), Neto et al. (2019), and Santos et al. (2020), excel at spatial localization but are primarily designed for region identification rather than comprehensive diagnosis. Xie et al. (2020)'s Faster DR-IACNN achieves 81.1% mean average precision (mAP) for grape disease detection but requires bounding box annotations and provides limited semantic understanding beyond localization. These approaches are valuable for preprocessing but insufficient for complete diagnostic workflows.

In contrast, our proposed multi-task learning approach addresses several limitations identified in the literature. Unlike single-task models that provide minimal diagnostic information, our framework outputs predictions for leaf detection, disease classification, symptom identification, agent recognition, and final disease diagnosis within a unified architecture. This design enables feature sharing across tasks, reducing parameter redundancy compared to multi-label approaches while maintaining task-specific specialization through dedicated output heads.

Table 2 presents a detailed comparison of the reviewed approaches across key dimensions. Our multi-task approach stands out in several aspects: (1) **comprehensive output** providing multiple diagnostic predictions rather than single classifications; (2) **unified architecture** enabling joint optimization and feature sharing; (3) **real-world focus** incorporating field-collected data with natural variability; and (4) **expert-level diagnosis** attempting to replicate the multi-step reasoning process used by plant pathologists.

The multi-task formulation also addresses several challenges identified by Barbedo (2018). By explicitly modeling symptom variations through dedicated output heads, our approach can better distinguish between visually similar disorders. The joint optimization

of disease presence, symptom types, and causal agents provides complementary information that improves diagnostic consistency.

Moreover, the unified architecture substantially reduces computational overhead and overall model size relative to maintaining multiple independent models. This lower computational cost facilitates deployment in resource-constrained environments, such as mobile devices and, potentially, embedded systems, thereby enhancing the practical applicability of the approach in real-world agricultural scenarios.

However, our approach also introduces new challenges, particularly in dataset construction and label consistency. Unlike single-task models that require simple class labels, multi-task learning demands comprehensive annotations across all diagnostic subtasks. This requirement is addressed through our data augmentation strategy and expert annotation protocol, but it remains a significant consideration for scalability.

In summary, while existing approaches have made valuable contributions to agricultural computer vision, they are primarily limited to specific subtasks or suffer from architectural limitations that reduce their practical applicability. Our multi-task learning framework represents a natural evolution toward more comprehensive, efficient, and expert-level diagnostic systems that can better serve the complex needs of real-world plant disease management.

Work	Task Type	Architecture	Output Predictions	Dataset Characteristics	Real-World Focus
(BARROS et al., 2021)	Single-task	ResNet50V2	Binary (symptoms vs. no symptoms)	Field-collected grape leaves, 3,289 images	High - designed for farmer use
(MOHANTY; HUGHES; SALATHÉ, 2016)	Single-task	GoogLeNet	Joint crop-disease classification (38 classes)	PlantVillage controlled dataset, 54,306 images	Low - controlled conditions only
(JI et al., 2020)	Multi-label	BR-CNN (ResNet50/ DenseNet121)	Independent predictions: crop, disease, severity	AI Challenger dataset, 12,691 images	Medium - mixed conditions
(BARBEDO, 2018)	Single-task	CNN	Disease classification (9 corn diseases)	Digipathos subset, subdivided symptomatic regions	High - real-world variability focus
(XIE et al., 2020)	Object Detection	Faster R-CNN + INSE-ResNet	Bounding boxes + disease class	GLDD grape dataset, 4,449 original images	Medium - varied conditions
(NETO et al., 2019)	Object Detection	YOLOv3	Fruit detection and counting	Citrus orchard images, 2,035 annotated images	High - uncontrolled field conditions
(SANTOS et al., 2020)	Instance Segmentation	Mask R-CNN	Cluster detection + segmentation masks	WGISD dataset, 4,000+ annotated clusters	High - real vineyard conditions
Our Approach	**Multi-task**	**Shared CNN + Task-specific heads**	**Leaf detection, disease status, symptoms, agent, diagnosis**	**Multi-crop field dataset with augmented annotations**	**High - comprehensive real-world pipeline**

Frame 2 – Comparative analysis of plant disease recognition approaches presented in this chapter. Our multi-task learning approach provides comprehensive diagnostic output through a unified architecture designed for real-world agricultural applications.

4 PROPOSED APPROACH AND IMPLEMENTATION

This chapter presents the practical aspects of the system designed to diagnose plant diseases from leaf images using a MTL approach. While previous chapters outlined the theoretical foundations and related works, the focus here is on how those ideas were translated into an operational pipeline: from data preparation to model deployment.

The system was developed to be modular, scalable, and extensible, aiming to replicate part of the diagnostic process carried out by specialists. It integrates multiple components, including a web application for image capture, a pre-processing stage to standardize input data, a neural network model trained to perform several tasks simultaneously, and a backend system to deliver predictions in a user-friendly and timely manner.

This chapter details each stage of the implementation: the creation and organization of the dataset, the design of the MTL model architecture, training and evaluation strategies, and how inference was integrated into a deployable pipeline. In addition, we include an honest account of the challenges faced, including failed attempts and lessons learned, which guided the decisions made throughout the project.

By unpacking the design choices and technical procedures, this chapter is meant to provide a comprehensive view of the system's development lifecycle, laying the groundwork for interpreting the results discussed in the following sections.

4.1 SYSTEM ARCHITECTURE OVERVIEW

The proposed system implements a distributed microservices architecture designed to provide scalable, modular, and extensible plant disease diagnosis capabilities through a RESTful API interface. Rather than being constrained to a specific client application, this design enables integration with diverse front-end systems, research platforms, and agricultural tools through standardized API endpoints.

The architecture is built around the principle of separation of concerns, where each service specializes in a specific aspect of the disease detection pipeline. This approach facilitates independent development, deployment, and scaling of system components while maintaining clear interfaces between services. The system can accommodate both

real-time interactive applications and batch processing workflows, making it suitable for various agricultural and research scenarios.

4.1.1 Core Microservices Architecture

The system comprises three primary microservices that work in concert to deliver comprehensive plant disease analysis:

Disease Analysis Server serves as the main API gateway and orchestration layer, coordinating the entire diagnostic workflow. It exposes the primary `/analyze` endpoint that accepts plant images and optional parameters, manages the complete processing pipeline, and aggregates results from downstream services. This service also handles request logging, maintains analysis history in a database, and provides health monitoring across all system components.

Segmentation Service specializes in advanced computer vision techniques for isolating individual leaves from plant images. It supports multiple state-of-the-art segmentation models including Segment Anything Model (SAM), SAM2, FastSAM, YOLOv8 segmentation, and LangSAM for text-guided segmentation. This service enables flexible segmentation strategies ranging from general-purpose foundation models to real-time optimized approaches, allowing researchers to select the most appropriate method for their specific use case.

Inference Server hosts the multi-task learning model and provides specialized machine learning inference capabilities. It manages model loading, GPU utilization, and executes the core disease detection algorithms. The server supports both standard inference pipelines and optimized processing paths for known plant varieties, enabling efficient prediction generation across multiple classification tasks simultaneously.

4.1.2 Processing Workflow and Service Interactions

Figure 19 illustrates the high-level processing workflow implemented by the system. The inference pipeline begins when a user submits a plant image through the API gateway. The system then makes an intelligent decision about whether to apply leaf segmentation based on the image characteristics and user preferences.

The process begins with a user uploading a plant image, which may optionally

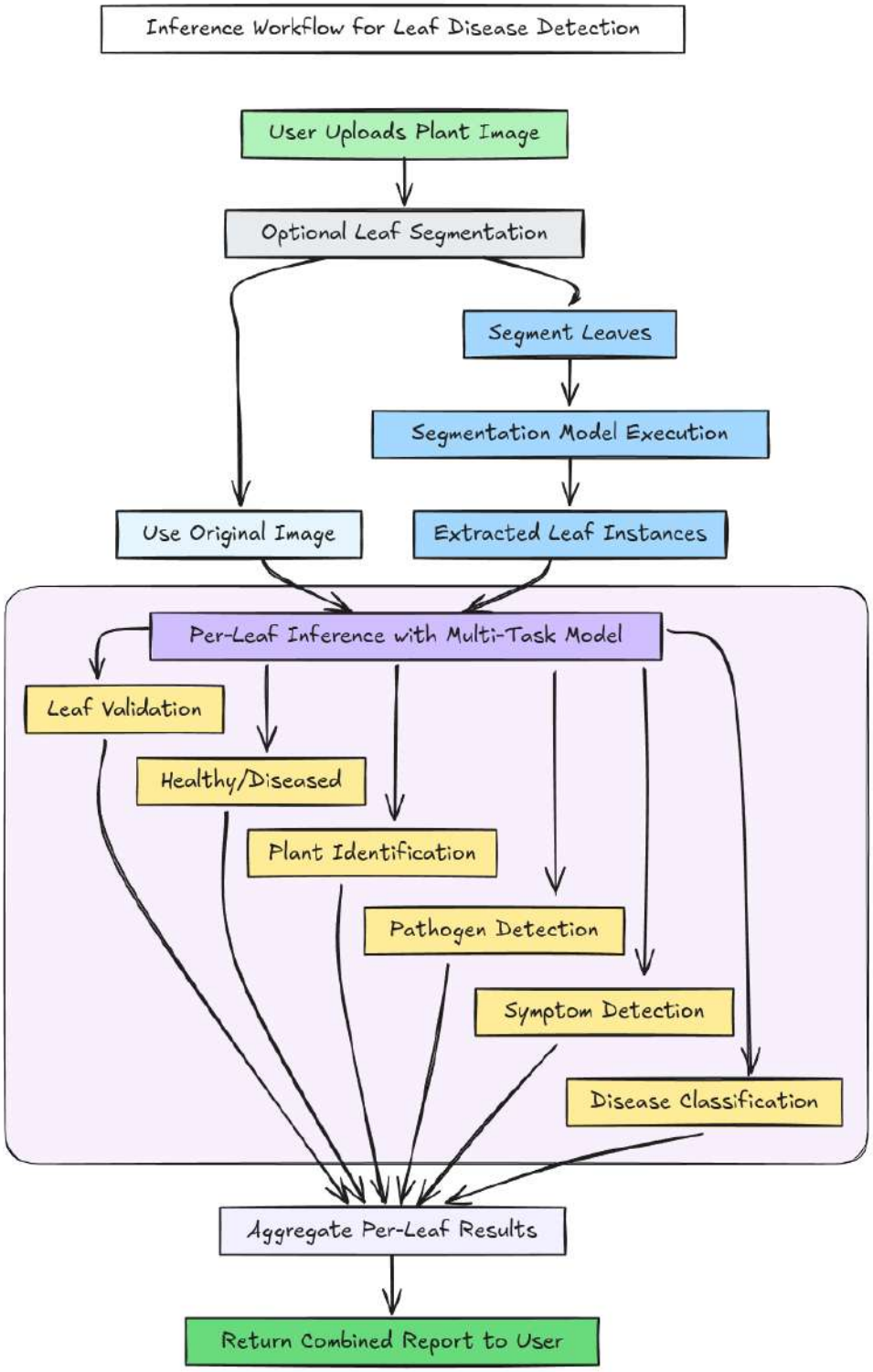


Figure 19 – High-level inference workflow showing the processing pipeline from image input to diagnostic report generation.

undergo leaf segmentation to isolate individual leaves. Each leaf (or the entire image, if segmentation is skipped) is then analyzed using a multi-task deep learning model that performs several tasks in parallel: binary classification (healthy vs. diseased), plant species identification, pathogen detection, disease classification, symptom detection, and leaf validation. The outputs from all tasks and leaves are then aggregated to produce a comprehensive diagnostic report, which is returned to the user.

When segmentation is enabled, the image is processed by the segmentation service to identify and extract individual leaf instances. This step is particularly valuable for complex images containing multiple leaves or when precise leaf-level analysis is required. The segmentation service applies area-based filtering and quality assessment to ensure that only relevant leaf structures are forwarded for disease analysis.

Each identified leaf (or the original image if segmentation is bypassed) undergoes inference through the multi-task learning model. This model simultaneously performs six different classification tasks: binary health assessment (healthy vs. diseased), plant species identification, pathogen detection, specific disease classification, symptom detection, and leaf validation. This parallel execution approach maximizes the information extracted from each image while maintaining computational efficiency.

The detailed interactions between system components are shown in Figure 20. This sequence diagram demonstrates how a web-based client application communicates with the disease detection system through the API gateway, highlighting the asynchronous processing capabilities and structured response handling. The interaction flow shows how the web application sends uploaded images to the Disease Analysis API, which coordinates with the segmentation service (if enabled) and the inference server to process each detected leaf through the multi-task learning model before returning an aggregated diagnostic report to the front-end.

The architecture adheres to several key design principles that ensure its suitability for both research and practical applications. **Modularity** is achieved through clear service boundaries, enabling independent development and testing of each component. **Scalability** is addressed through the microservices pattern, allowing individual services to be scaled and adjusted based on demand patterns and computational requirements.

Extensibility is built into the system through standardized interfaces using a RESTful API design and plugin-like architectures. New segmentation models can be integrated into the segmentation service without affecting other components, and additional clas-

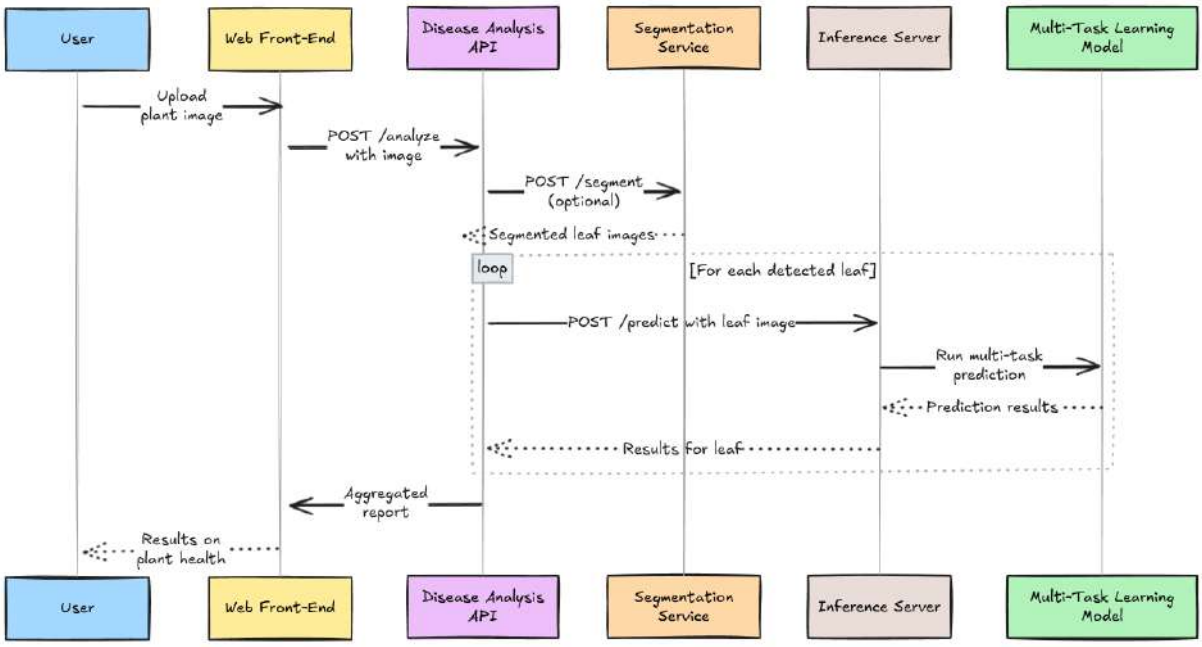


Figure 20 – Sequence diagram showing the interaction flow between user, web front-end, and backend microservices for plant disease detection.

sification tasks can be added to the multi-task model through the inference server's flexible endpoint structure. Moreover, new steps can be added to the pipeline without modifying existing components, allowing for easy expansion of the system's capabilities.

The API-first design philosophy ensures that the system can be integrated seamlessly with existing and new agricultural information systems, research platforms, and custom applications. Standard HTTP/REST interfaces enable easy integration regardless of the client technology, while structured JSON responses provide consistent data formats for downstream processing.

To demonstrate the practical implementation of this architecture, a sample web front-end application that utilizes the described API is available at <https://oracle.cultivai.com.br/> (Cultivai, 2024a), providing users with an intuitive interface for plant disease diagnosis. Additionally, the complete API documentation, including endpoint specifications, parameter descriptions, and example requests, can be accessed at <https://predict.cultivai.com.br/docs> (Cultivai, 2024b), enabling developers and researchers to integrate the system into their own applications and research workflows, allowing for users that are both not technical and not familiar with the system to use it, as well as more advanced users to use it in a more customized way.

4.2 DATASET CONSTRUCTION AND LABELING

The development of robust multi-task learning models for plant disease classification requires training datasets that are both comprehensive in scale and rich in semantic annotations. While existing plant pathology datasets provide substantial image collections, they typically lack the detailed pathological metadata necessary for fine-grained symptom analysis and multi-task learning objectives. This section presents a novel methodology that combines Retrieval-Augmented Generation (RAG) techniques with vision-capable Large Language Models (LLMs) to systematically augment heterogeneous plant disease datasets with comprehensive, literature-grounded annotations.

The approach addresses a fundamental challenge in specialized domain datasets: the labor-intensive nature of expert annotation and the inconsistency of metadata quality across different data sources. By leveraging authoritative plant pathology literature as a knowledge base and employing multi-modal LLM analysis for image-specific validation, this methodology generates scientifically grounded annotations while maintaining scalability for large dataset processing.

4.2.1 Dataset Summary and Characteristics

To support the proposed multi-task learning pipeline, a diverse and semantically rich dataset collection was constructed by integrating multiple open-source datasets. Each dataset contributes unique characteristics in terms of plant species, disease representation, annotation depth, and image acquisition conditions. This subsection details the composition and content of each source dataset, including information on the number of samples, annotated labels, and the presence or absence of higher-order metadata such as symptoms, pathogen names, and scientific disease nomenclature.

4.2.1.1 PlantVillage Dataset

The PlantVillage dataset (HUGHES; SALATHE, 2015) serves as the primary corpus, comprising 54,305 images across 14 plant species and 38 disease classes. This dataset is particularly known for its standardized imaging setup—each image typically depicts a single, isolated leaf placed on a homogeneous background under consis-

tent lighting. As previously discussed in Section 3.2 and illustrated in Figure 10, this setup enhances visual clarity and model training efficacy, but it lacks real-world variability. Metadata in PlantVillage is embedded in the folder structure following the pattern `Plant_Species___Disease_Condition`, which supports straightforward extraction of plant and disease labels. However, deeper annotations such as symptomatology, pathogen names, or disease progression stages are not included.

4.2.1.2 MangoLeafBD Dataset

The MangoLeafBD dataset (AHMED et al., 2023) contains 4,000 annotated images of mango leaves captured under more natural conditions, including partial occlusions, complex backgrounds, and varying lighting, as demonstrated in Figure 21. The dataset comprises both healthy and diseased samples and includes images categorized into five disease types (Anthracnose, Bacterial Canker, Gall Midge, Powdery Mildew, and Sooty Mold). While basic disease names are available, this dataset also lacks scientific disease identifiers, symptom profiles, and causal pathogen annotations. Its inclusion significantly enhances the domain diversity and robustness of the training pipeline for tropical disease manifestations.

Figure 21 – Sample images from the MangoLeafBD dataset by Ahmed et al. (2023) showing natural imaging conditions with partial occlusions, complex backgrounds, and varying lighting that enhance training robustness.

4.2.1.3 Cassava Leaf Disease Dataset

This dataset (MWEBAZE et al., 2021), released as part of a Kaggle competition by the Makerere University AI Lab, comprises 21,367 field-acquired images of cassava leaves,

each annotated with one of five well-defined classes: Cassava Bacterial Blight (CBB), Cassava Brown Streak Disease (CBSD), Cassava Green Mottle (CGM), Cassava Mosaic Disease (CMD), and Healthy. The class labels are provided via structured CSV files, ensuring reliable mapping between images and pathological states. However, despite its large scale and comprehensive labeling, this dataset was ultimately excluded from the final training pipeline due to several critical limitations. The images lack controlled imaging conditions, with many samples containing multiple plant specimens within a single frame, making it difficult to isolate individual leaf pathology. Additionally, a significant portion of the dataset exhibits inconsistent aspect ratios that result in stretched or distorted images, along with significant blurriness, potentially introducing artifacts that could negatively impact model training. These limitations are illustrated in Figure 22. The highly variable field conditions, while representative of real-world scenarios, introduce excessive noise and complexity that would require substantial preprocessing efforts to achieve consistency with other datasets in the collection.

Figure 22 – Representative images from the Cassava Leaf Disease dataset by Mwebaze et al. (2021) highlighting principal limitations: (**left**) an image containing multiple plant specimens within a single frame and exhibiting aspect ratio distortion; (**right**) an image characterized by significant blurriness. These factors were instrumental in the decision to exclude this dataset from the final training pipeline due to concerns regarding data quality and consistency.

4.2.1.4 PlantDoc Dataset

The PlantDoc dataset (SINGH et al., 2020) includes 2,598 annotated images spanning 13 plant species and 17 disease categories. While this dataset initially appeared promising due to its taxonomic diversity and realistic imaging conditions, it was ultimately excluded from the final training pipeline due to several fundamental quality issues. The

images are sourced from internet collections rather than controlled acquisition protocols, resulting in significant heterogeneity in image quality and composition. Many images contain watermarks overlaid on plant specimens, particularly on fruit images, which would interfere with feature extraction and model training. The dataset exhibits extreme variability in background conditions, ranging from highly controlled white backgrounds to complex natural environments with excessive textual elements that could serve as confounding factors. Additionally, a substantial portion of the dataset consists of low-resolution or blurry images that fail to meet minimum quality thresholds for reliable pathological analysis. These quality variations are demonstrated in Figure 23. The inconsistent imaging conditions, combined with the presence of non-leaf specimens and quality artifacts, made this dataset unsuitable for integration with the more controlled datasets selected for the final training pipeline.

Figure 23 – Representative images from the PlantDoc dataset by Singh et al. (2020) illustrating quality variations: (**left**) a high-quality leaf image suitable for analysis; (**center**) an image with prominent watermarks over the specimen; (**right**) an image containing excessive background text. These issues contributed to the exclusion of PlantDoc from the final training pipeline.

4.2.1.5 Plant Disease Recognition Dataset

This dataset (RAHMAN, 2021) contains 1,530 images organized into three symptom-based categories: "Healthy", "Powdery", and "Rust". The dataset is pre-divided into train, test, and validation sets. It focuses on symptom manifestation rather than taxonomic classification, with unknown plant types and disease specifics. While the images are not captured under controlled conditions, the leaf subjects are consistently central and clearly in focus in all pictures, providing good visual clarity for symptom analysis, as shown in Figure 24. The dataset's value lies in its symptom-centric approach, though it lacks detailed metadata regarding plant species identification or scientific disease

nomenclature.

Figure 24 – Representative images from the Plant Disease Recognition dataset: (**left**) healthy leaf, (**center**) leaf with powdery symptoms, and (**right**) leaf with rust symptoms. Each image demonstrates the symptom-based categorization approach with consistently centered and clearly focused leaf specimens.

4.2.1.6 Apple Disease Dataset (D-KAP)

The D-KAP dataset (SHARMA; PADHA; BASHIR, 2022) contains 6,000 high-resolution images of Kashmiri apple plants categorized into four classes (healthy, scab, rust, and alternaria blotch). While the images are captured under real-world conditions rather than strictly controlled laboratory settings, they exhibit characteristics that approach controlled conditions. Each image consistently features a single leaf specimen positioned centrally within the frame, with backgrounds predominantly consisting of granite surfaces or carpet textures, as illustrated in Figure 25. This quasi-controlled approach maintains visual consistency while introducing minimal environmental variation, making the dataset particularly suitable for robust model training. The dataset's focus on a single crop species provides specialized representation of apple pathology that complements the broader taxonomic coverage of other datasets.

4.2.1.7 Plant Pathology Challenge 2020 Dataset

The Plant Pathology Challenge 2020 dataset (THAPA et al., 2020) comprises 3,651 high-quality images of apple foliar diseases captured under real-world conditions with variable illumination, angles, surfaces, and background noise. The dataset was specifically designed for the Plant Pathology Challenge held as part of the Fine-Grained Visual Categorization (FGVC) workshop at CVPR 2020. While the images are captured in uncontrolled environments, they maintain excellent usability characteristics with the primary leaf consistently positioned in the center of each image frame. The in-the-wild

Figure 25 – Representative images from the D-KAP apple disease dataset (SHARMA; PADHA; BASHIR, 2022): (**left**) a single apple leaf specimen on a granite background, and (**right**) a single apple leaf specimen on a carpet background. Both images demonstrate the dataset's quasi-controlled imaging conditions with consistent single-leaf presentation and minimal environmental variation.

backgrounds provide natural environmental context without introducing excessive noise, and importantly, the images maintain their original aspect ratios without distortion or stretching artifacts, as demonstrated in Figure 26. The images are expert-annotated and focus on three primary categories: apple scab, cedar apple rust, and healthy leaves. The dataset is structured using CSV files for metadata organization, similar to the Cassava dataset. Its emphasis on real-world image acquisition conditions combined with consistent compositional quality and expert-quality annotations makes it particularly valuable for developing robust disease classification models that can perform effectively in practical agricultural settings.

4.2.1.8 DiaMOS Plant Dataset

The DiaMOS Plant dataset (FENU; MALLOCI, 2021) contains 3,505 field-collected images of pear fruit and leaves, encompassing four disease classes including healthy samples. While this dataset was initially considered for its comprehensive annotation approach and practical agricultural relevance, it was ultimately excluded from the final training pipeline due to several integration challenges. The dataset lacks controlled imaging conditions, with many images containing multiple leaf specimens within a single

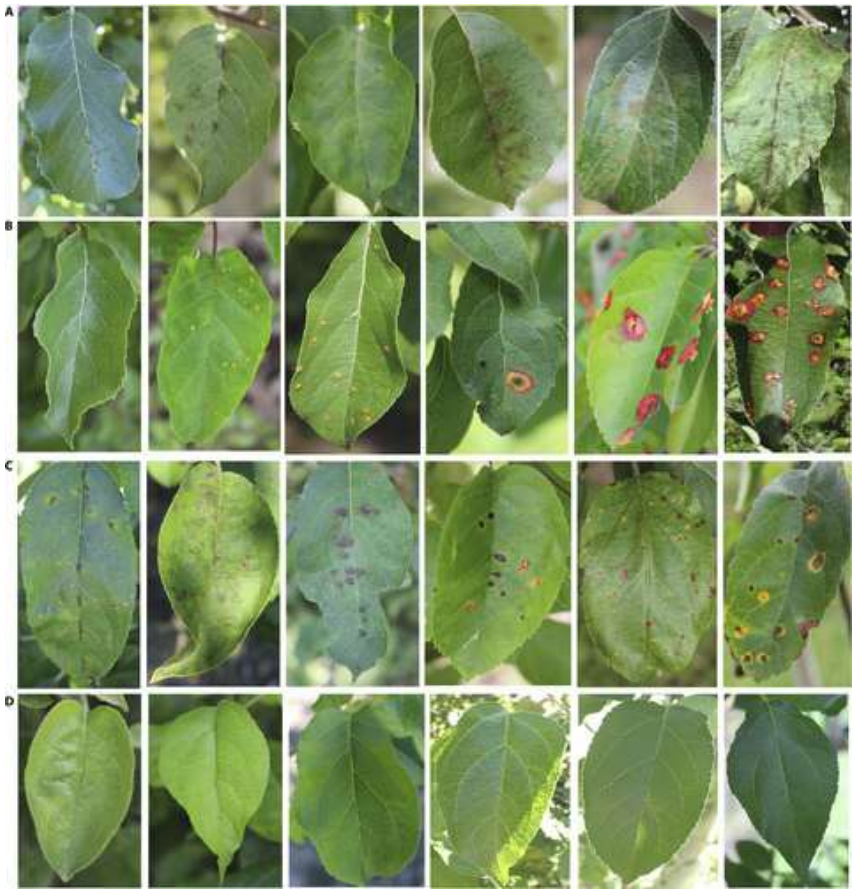


Figure 26 – Sample images from the Plant Pathology Challenge 2020 dataset by Thapa et al. (2020) demonstrating centered leaf positioning with natural in-the-wild backgrounds and expert-quality annotations for apple foliar diseases.

frame, making it difficult to establish consistent pathological analysis protocols. Although the dataset provides YOLO-format bounding box annotations, these annotations typically target only a single leaf within each image, leaving other visible leaves unlabeled and potentially creating training inconsistencies. The time constraints of the project made it impractical to develop specialized processing pipelines to extract and utilize the partial annotations effectively. Additionally, a significant portion of the dataset focuses on fruit pathology rather than leaf-based diseases, which diverges from the primary focus of this research on foliar disease classification. The combination of complex annotation requirements, multi-specimen imaging, and the substantial fruit-focused content made this dataset unsuitable for integration within the established processing pipeline, as illustrated in Figure 27.

Figure 27 – Representative images from the DiaMOS Plant dataset by Fenu e Malloci (2021): (**left**) an image focusing on fruit rather than leaves, and (**right**) an image containing multiple leaves within a single frame. These characteristics contributed to the dataset's exclusion due to challenges in maintaining consistent analysis protocols.

4.2.1.9 PDDB (Digipathos) Dataset

The PDDB dataset (BARBEDO et al., 2018), also known as Digipathos, is a comprehensive plant disease image collection developed by Embrapa, comprising 2,326 original images that span 171 diseases and other disorders across 21 plant species. While the dataset initially appeared attractive due to its controlled imaging conditions and broad taxonomic coverage, it was ultimately excluded from the final training pipeline for several practical reasons.

A central issue is the dataset's heavy reliance on augmentation: the original images are systematically subdivided and cropped, resulting in an expanded version (XDB) with 46,104 images distributed across 93 "Cropped" folders. However, this expansion does not address the underlying scarcity of original data per class. Quantitative analysis of the folder structure highlights a pronounced class imbalance and data skew in both the original and augmented sets, with the following summary statistics describing the number of files per folder:

- **Original (non-cropped) data:** 2,339 files in 177 folders. The number of files per folder ranges from 1 to 88, with a mean of 13.21, a median of 6, and a standard deviation of 16.70. The 25th percentile (Q1) is 2.0, and 24 folders (13.6%) have two or fewer images. This means that most disease categories have very few original images, making them unsuitable for robust model training.

- **Cropped (XDB) data:** 46,104 files in 93 folders. The number of files per folder ranges from 1 to 3,791, with a mean of 495.74, a median of 150, and a standard deviation of 775.58. The 25th percentile (Q1) is 42.0, and 23 folders (24.7%) have 42 or fewer images. While the augmentation increases the number of images per class, the distribution remains highly skewed, with a small number of categories dominating the dataset and many still underrepresented.

For reference, when considering both the original and cropped data together, there are 270 folders and 48,443 files in total, with the number of files per folder having a mean of 179.42, a median of 11.5, and a standard deviation of 508.60. The 25th percentile (Q1) is 3.0, and 58 folders (21.5%) have three or fewer images.

Further compounding these issues, a substantial fraction of the images are of fruit pathology rather than leaf specimens, which diverges from the primary focus of this research. The authors themselves note that the dataset's expansion is achieved mainly through automatic cropping and zooming of a limited set of base images, rather than through the acquisition of new, diverse samples. Given the project's time constraints and the considerable effort required to extract and process only the relevant leaf-based images from this highly augmented and imbalanced dataset, the potential benefits did not justify the required preprocessing investment. The combination of limited original leaf imagery per category, severe data skew, and augmentation-heavy methodology rendered the PDDB dataset unsuitable for integration into the established training pipeline, as illustrated in Figure 28.

Table 1 summarizes the key characteristics and evaluation outcomes for all datasets considered in this study, providing a comprehensive comparison of their suitability for the proposed multi-task learning pipeline.

Table 1 – Comparison of Evaluated Plant Disease Datasets

Dataset	# Images	Plant Species	Disease Classes	Controlled Images	Structured Metadata	Used in Training
PlantVillage (HUGHES; SALATHE, 2015)	54,305	14	38	✓	Folder names	✓
MangoLeafBD (AHMED et al., 2023)	4,000	1 (Mango)	5	✓	Folder names	✓
PlantDisease (RAHMAN, 2021)	1,530	Unknown	3	✗	Folder names	✓
Apple (D-KAP) (SHARMA; PADHA; BASHIR, 2022)	6,000	1 (Apple)	4	Quasi-controlled	Folder names	✓
Plant Pathology 2020 (THAPA et al., 2020)	3,651	1 (Apple)	3	✗	CSV	✓
Excluded Datasets						
Cassava (MWEBAZE et al., 2021)	21,367	1 (Cassava)	4 + healthy	✗	CSV	✗
PlantDoc (SINGH et al., 2020)	2,598	13	17	Mixed	Folder names	✗
DiaMOS Plant (FENU; MALLOCI, 2021)	3,505	1 (Pear)	4	✗	CSV + YOLO	✗
PDDB (Digipathos) (BARBEDO et al., 2018)	46,513	21	171	✓	Folder names	✗

The systematic evaluation and selection process resulted in a curated dataset collection comprising 69,486 samples from five high-quality sources that met the project's

Figure 28 – Representative images from the PDDB (Digipathos) dataset: (**left**) an example of a fruit specimen rather than a leaf, (**center**) a cropped version of a leaf image, and (**right**) another cropped version of a leaf. The prevalence of fruit images and the heavy reliance on systematic cropping of limited base leaf images illustrate the dataset's augmentation-heavy methodology and its limitations for this research.

technical and methodological requirements. While the excluded datasets represented over 73,000 additional samples, their integration would have introduced significant preprocessing challenges, annotation inconsistencies, and quality issues that would have compromised the overall training pipeline effectiveness. The selected datasets maintain varying degrees of metadata richness, with basic plant and disease labels consistently present across all sources. However, critical information necessary for multi-task modeling—such as symptom descriptions, pathogen classification, and scientific disease naming—remains largely absent even in the selected datasets. To address this limitation, subsequent stages of the pipeline employ RAG techniques to systematically enrich these datasets with comprehensive pathological annotations, as discussed in the following section.

4.2.2 Additional Datasets for Binary Leaf Classification

Beyond the primary datasets integrated into the main multi-task learning pipeline, several additional datasets were incorporated to support a specialized binary classification task designed to determine whether an image contains a leaf or not. This foundational task serves as a preprocessing step for the broader plant disease classification pipeline, ensuring that only relevant botanical imagery is processed by downstream disease detection models. While these datasets do not participate in the overall multi-task learning architecture, they are essential for training a robust leaf detection model

that filters input data before disease classification.

The binary leaf classifier combines multiple datasets to create a balanced training set with both positive and negative examples. This approach ensures that the model can effectively distinguish between botanical and non-botanical imagery across diverse visual contexts.

4.2.2.1 Positive Examples (Leaf Images)

The positive training examples are primarily sourced from the PlantVillage dataset (HUGHES; SALATHE, 2015), where all 54,305 images serve as positive examples since they exclusively contain plant leaves with various disease conditions and healthy states. This comprehensive coverage ensures that the binary classifier is exposed to the full range of leaf appearances, disease manifestations, and imaging conditions present in the downstream classification pipeline.

Additionally, the Flowers102 dataset (NILSBACK; ZISSERMAN, 2008) was incorporated to increase leaf sample diversity. This dataset contains 8,189 images of flowers from 102 different categories, providing complementary botanical imagery that enhances the model's ability to recognize diverse plant structures and leaf configurations not represented in the disease-focused datasets.

4.2.2.2 Negative Examples (Non-Leaf Images)

The negative training examples are carefully selected from datasets that provide clear contrast to botanical imagery while maintaining visual complexity comparable to leaf images. The StanfordCars dataset (KRAUSE et al., 2013) contributes vehicle images that serve as unambiguous negative examples, ensuring that the classifier can distinguish between manufactured objects and natural plant structures.

The ImageNette dataset (HOWARD; GUGGER, 2020), a subset of ImageNet (DENG et al., 2009) containing everyday objects, provides additional negative examples while excluding plant-related classes. This dataset contributes images of tench, English springer, cassette player, chain saw, church, French horn, garbage truck, gas pump, golf ball, and parachute, offering diverse visual contexts that help the model generalize beyond simple object categories.

The Caltech256 dataset (GRIFFIN et al., 2007) serves as a mixed dataset containing both leaf and non-leaf samples. Due to its heterogeneous nature, this dataset requires manual labeling to separate positive and negative examples, contributing to the overall balance of the training set while providing additional visual diversity in both categories.

This specialized binary classification approach ensures that the primary disease classification models receive only relevant botanical inputs, improving both computational efficiency and classification accuracy by filtering out irrelevant imagery at the preprocessing stage.

4.2.3 Source Dataset Integration

Following the systematic evaluation described in the previous section, the final training pipeline integrates five carefully selected datasets. These datasets collectively provide robust coverage of plant disease manifestations while maintaining compatibility with the established processing methodology. The PlantVillage dataset (HUGHES; SALATHE, 2015) serves as the primary source with 54,305 images spanning 14 plant species and 38 disease classes, providing broad taxonomic coverage with consistent imaging conditions. The MangoLeafBD dataset (AHMED et al., 2023) contributes 4,000 specialized images focusing on tropical disease manifestations under controlled conditions. The Plant Disease Recognition dataset (RAHMAN, 2021) adds 1,530 symptom-focused images that enhance the model's capacity for fine-grained symptom analysis. The Apple Disease (D-KAP) dataset (SHARMA; PADHA; BASHIR, 2022) provides 6,000 high-quality images with quasi-controlled conditions, while the Plant Pathology Challenge 2020 dataset (THAPA et al., 2020) contributes 3,651 expert-annotated apple disease images captured under real-world conditions, adding specialized focus on apple foliar diseases including scab and cedar apple rust.

A comprehensive dataset creation pipeline was developed to transform these heterogeneous datasets into a unified, semantically rich training corpus suitable for multi-task learning. This pipeline addresses the fundamental limitations identified during dataset evaluation. Figure 29 presents an overview of this complete pipeline, illustrating how the integrated source datasets undergo sequential processing through four major transformation stages. Each stage addresses specific limitations identified during the initial analysis, culminating in a unified dataset that supports both traditional disease classifi-

cation and fine-grained symptom analysis tasks. The following subsections detail each component of this pipeline, explaining the methodologies, technical implementations, and design rationales that enable systematic knowledge augmentation at scale.

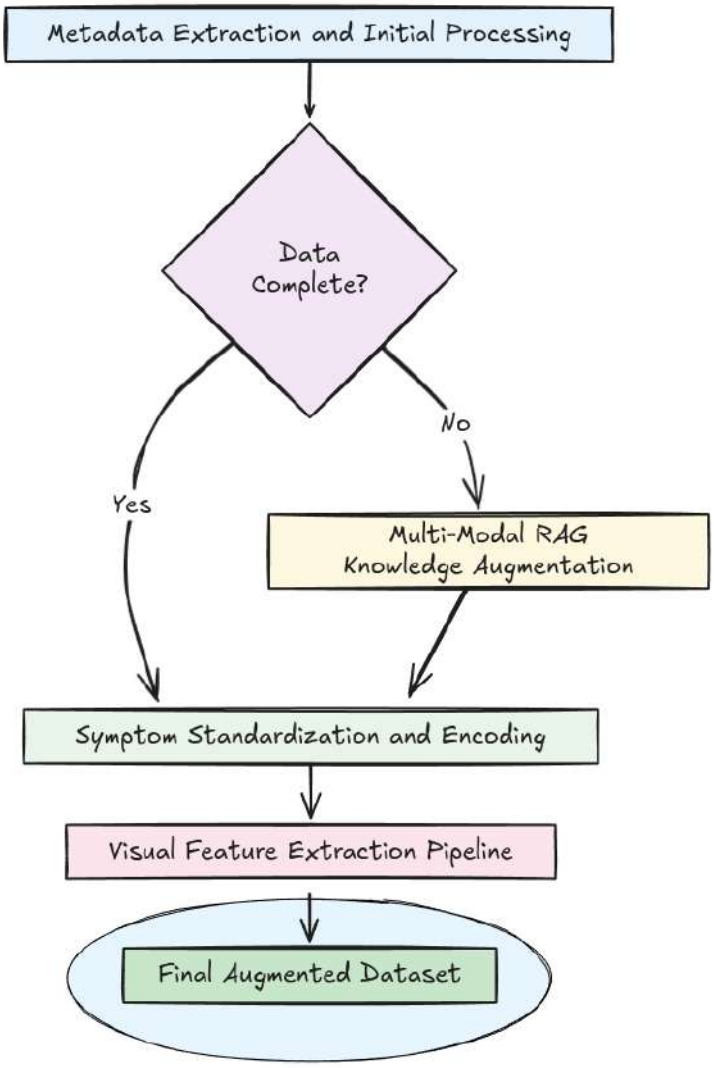


Figure 29 – Complete overview of the dataset creation pipeline (author's own work) showing the four major processing stages that transform basic taxonomic annotations into comprehensive, multi-modal training data for multi-task learning applications.

Each source dataset presented distinct organizational structures and metadata schemas, necessitating a systematic integration approach. The PlantVillage dataset, MangoLeafBD dataset, Plant Disease Recognition dataset, and Apple Disease (D-KAP) dataset employed hierarchical folder naming conventions following patterns such as `Plant_Species___Disease_Condition`. This structure enables direct extraction of plant and disease identifiers through systematic parsing of directory structures. The Plant Pathology Challenge 2020 dataset utilized CSV-based metadata organization, requiring structured file parsing to link image identifiers to taxonomic and pathological

classifications. This diversity in organizational approaches required the development of flexible parsing routines capable of handling different metadata schemas while maintaining consistency in the resulting unified dataset structure.

Figure 30 illustrates the initial metadata extraction process that accommodates these diverse organizational patterns. This preprocessing stage establishes a unified metadata schema while preserving dataset provenance, enabling systematic assessment of annotation completeness across sources. The analysis revealed that while basic plant and disease classifications were consistently available, critical pathological information including scientific disease nomenclature, pathogen classifications, and symptom profiles were largely absent across all datasets.

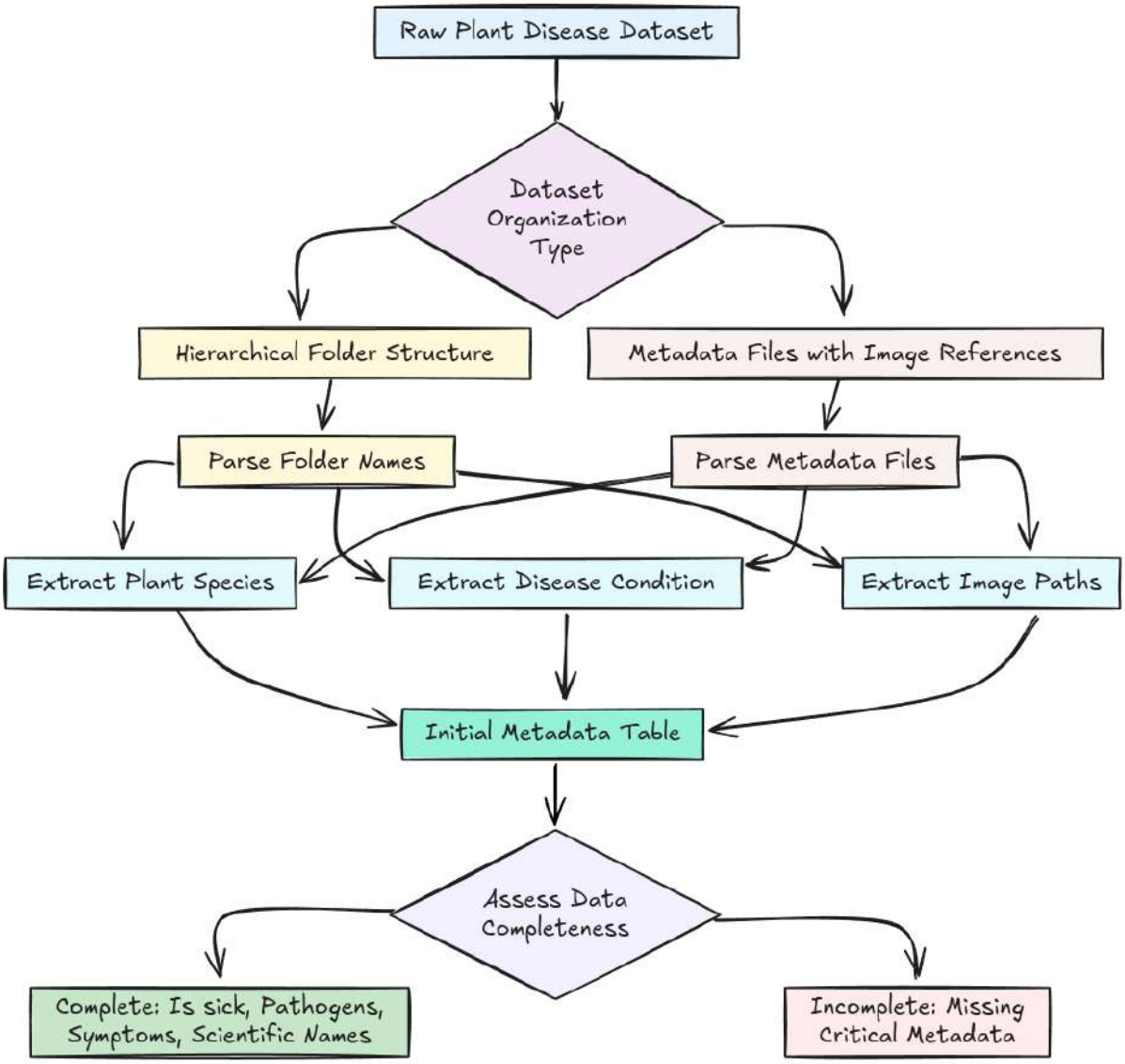


Figure 30 – Metadata extraction pipeline (author's own work) illustrating the systematic processing of diverse dataset organizational structures into a unified schema for assessment of annotation completeness.

4.2.4 Multi-Modal RAG-Enhanced Knowledge Augmentation

To address the systematic lack of comprehensive pathological annotations, we developed a novel multi-modal knowledge augmentation pipeline that combines domain-specific literature retrieval with vision-capable LLM analysis. This approach represents a significant advancement over traditional crowdsourcing or expert annotation methods, providing both scalability and scientific rigor through literature grounding.

The methodology employs a two-stage RAG architecture that first retrieves relevant pathological knowledge from authoritative plant pathology references, then leverages multi-modal LLM capabilities to validate and refine this knowledge against visual evidence from the target images. The knowledge base was constructed from comprehensive plant pathology handbooks, including Westcott's Plant Disease Handbook (HORST, 2013) and the Handbook of Plant Disease Identification and Management (AGLAVE, 2018). These references provide broad coverage of pathological conditions across agricultural and horticultural species.

Figure 31 demonstrates the complete multi-modal RAG augmentation workflow. For each plant-disease combination identified during metadata extraction, the system constructs targeted literature search queries encompassing symptom descriptions, pathogen classifications, and diagnostic characteristics. The retrieval component employs semantic search techniques to identify relevant passages from the indexed literature, creating contextual knowledge that serves as the foundation for subsequent LLM analysis.

The integration of multi-modal capabilities distinguishes this approach from purely text-based knowledge augmentation methods. By combining retrieved literature context with visual analysis of the target leaf images, the system performs cross-validation between established pathological knowledge and observable symptoms. This dual-modal approach ensures that generated annotations reflect both scientific accuracy and image-specific manifestations, addressing the common problem of generic descriptions that do not correspond to visible evidence.

The structured output generation employs GPT-4o (HURST et al., 2024) with carefully engineered prompts that combine retrieved literature context with visual analysis instructions. The system prompt establishes the role of an expert plant pathologist with access to comprehensive literature, while user prompts integrate specific retrieved knowledge

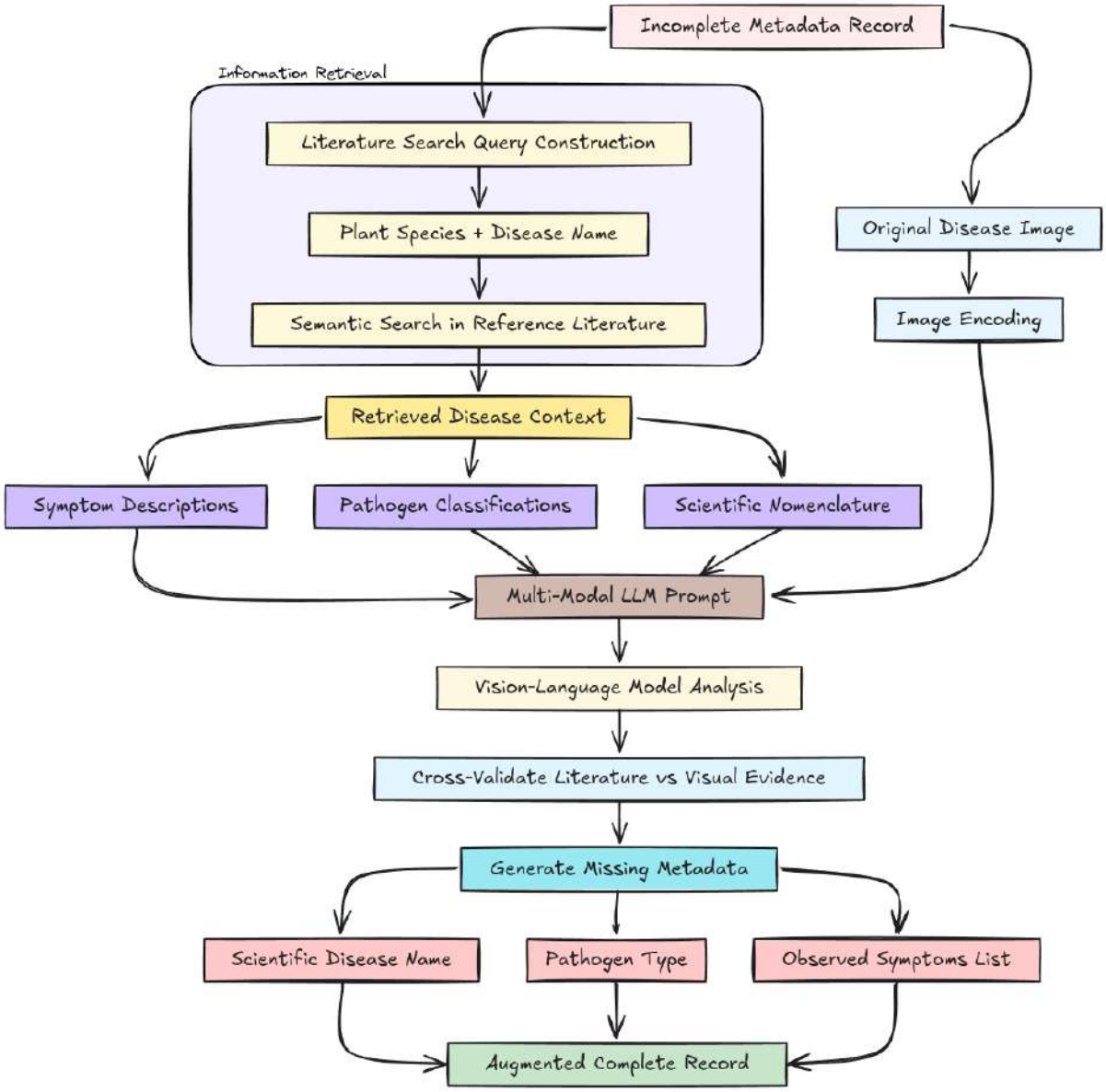


Figure 31 – Multi-modal RAG pipeline combining literature retrieval with vision-capable LLM analysis for systematic generation of comprehensive plant disease annotations.

with image analysis tasks. The use of structured output formats ensures consistent data generation across the entire dataset, facilitating subsequent processing and integration steps.

A critical aspect of this methodology is the emphasis on generating only missing metadata rather than replacing existing annotations. This conservative approach preserves valuable human-generated annotations while systematically filling gaps in pathological information. The system prioritizes visual evidence when conflicts arise between retrieved literature and observable symptoms, ensuring that annotations remain grounded in actual image content rather than generic disease descriptions.

However, not all datasets permit comprehensive augmentation across all metadata

categories. For example, the Plant Disease Recognition Dataset (RAHMAN, 2021) provides only symptom-based classifications ("Healthy", "Powdery", "Rust") without plant species identification or disease-specific context. In such cases, where the system cannot confidently infer missing information—such as plant taxonomy, pathogen names, or scientific disease nomenclature—these metadata fields are deliberately left empty rather than populated with potentially erroneous information. This conservative approach ensures data integrity by avoiding the introduction of speculative annotations that could compromise model training. During subsequent training phases, these missing values are appropriately handled through per-task filtering mechanisms, allowing each multi-task learning objective to utilize only the annotations that are available and reliable for that specific task.

4.2.5 Symptom Standardization and Semantic Harmonization

The integration of multiple datasets and literature sources necessitates systematic standardization of symptom terminology to enable effective multi-task learning. Plant pathology literature and different datasets employ varying descriptive vocabularies for similar pathological manifestations, creating challenges for model training and cross-dataset generalization.

Figure 32 illustrates the comprehensive symptom standardization process implemented to address this challenge. The methodology begins with the extraction of all unique symptom descriptions from both literature-retrieved knowledge and existing dataset annotations. These raw descriptions undergo systematic mapping to a standardized vocabulary that maintains biological accuracy while reducing terminological variation.

The standardization mapping addresses common variations in symptom descriptions while preserving specificity necessary for accurate classification. For example, descriptions such as "bronzed leaf surfaces" and "bronze coloration" are unified under the standardized term "bronzing", while maintaining distinction from related but different symptoms like "chlorosis" or "necrosis." This process required careful consideration of plant pathological terminology to ensure that biologically distinct symptoms remained separable while reducing unnecessary vocabulary fragmentation.

Following standardization, symptom annotations undergo one-hot encoding to cre-

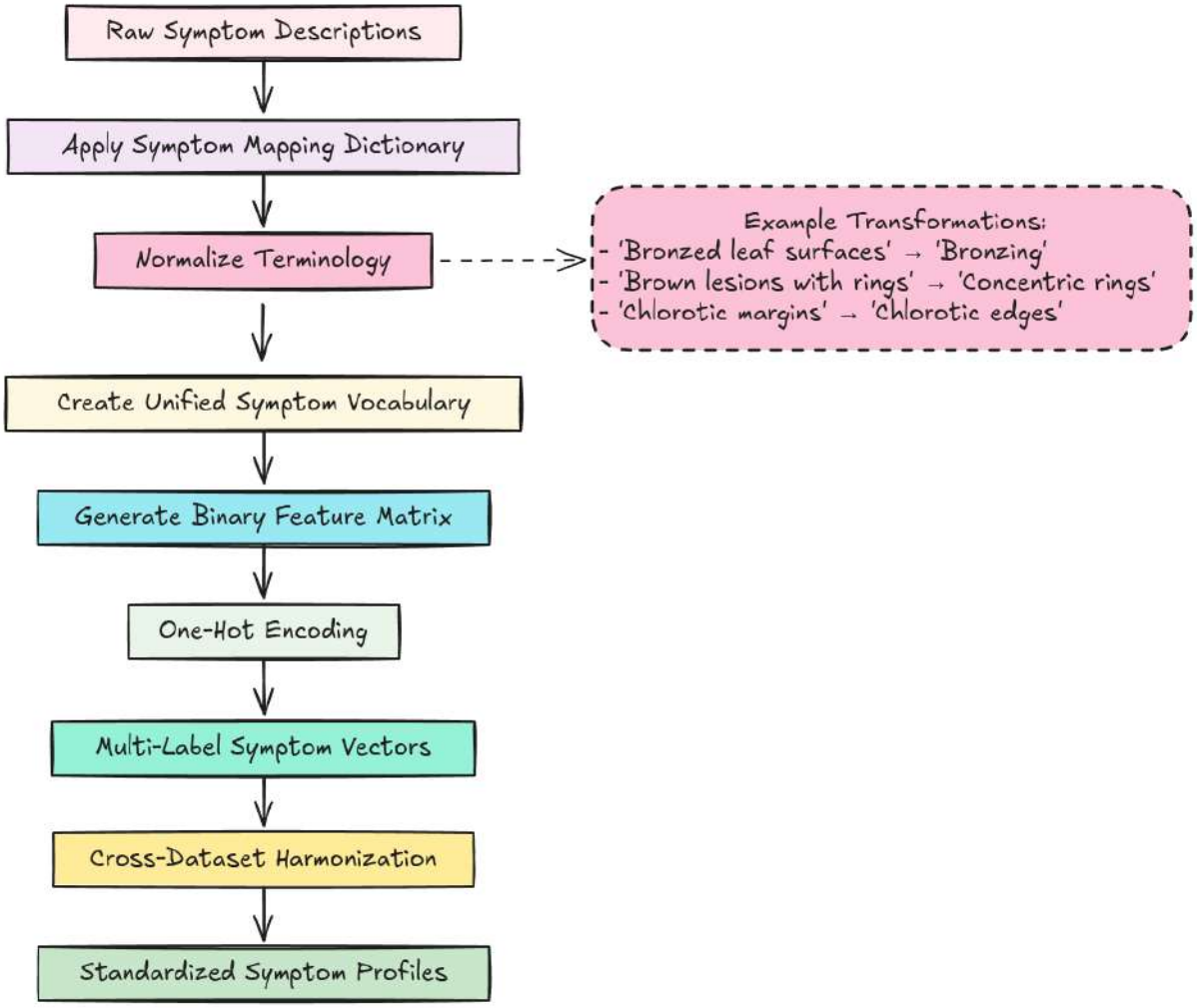


Figure 32 – Systematic standardization of symptom terminology across datasets and literature sources, followed by one-hot encoding for multi-label classification tasks.

ate binary feature vectors suitable for multi-label classification tasks. This encoding strategy enables the multi-task learning model to simultaneously predict multiple symptom manifestations, supporting both fine-grained symptom analysis and hierarchical disease classification objectives. The resulting symptom profiles provide rich semantic annotations that complement traditional taxonomic disease classifications.

4.2.6 Visual Feature Integration

In an effort to reduce computational costs during model training, an approach was implemented to pre-extract high-dimensional latent vectors from all images using a pre-trained convolutional neural network. The motivation behind this strategy was to avoid repeatedly processing each image through the backbone network (ResNet50) for every task and epoch during multi-task learning, thereby streamlining the training pipeline.

The intention and rationale for this approach, as well as the challenges encountered, will be discussed in the following chapter on model training.

In addition to feature extraction, a dedicated background removal pipeline was also developed to isolate the leaf regions in each image prior to embedding generation. This preprocessing step aimed to reduce the influence of irrelevant background information and enhance the focus on disease-relevant visual features. The technical details, implementation, and impact of the background removal process are described in later sections.

Figure 33 illustrates the visual feature extraction pipeline. The process employed a ResNet50 architecture pre-trained on ImageNet, with the top classification layers removed to access the intermediate feature representations. Images were standardized by resizing to 224×224 pixels and applying ImageNet normalization to ensure compatibility with the pre-trained weights.

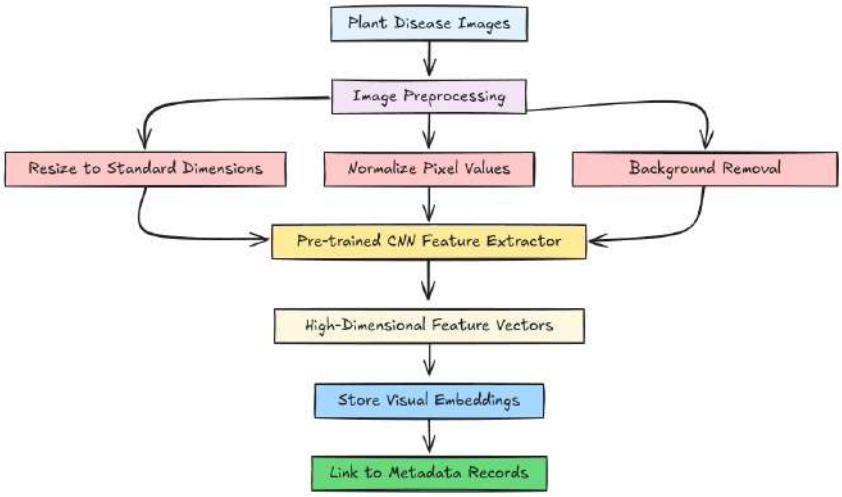


Figure 33 – Visual feature extraction process using pre-trained ResNet50 to generate high-dimensional embeddings for integration with semantic annotations.

Global average pooling was applied to the spatial dimensions of the feature maps, resulting in 2048-dimensional embedding vectors that were intended to capture high-level visual patterns relevant to disease classification. These embeddings were stored alongside the semantic annotations, with the goal of enabling the multi-task learning model to leverage both explicit pathological knowledge and implicit visual features, while significantly reducing the computational burden during training.

Batch processing strategies were used to efficiently extract features across the entire dataset. However, as will be detailed in the subsequent chapter, this batched approach ultimately proved unsuccessful due to specific limitations and issues encountered during

model training. The lessons learned from this attempt informed the final design of the training pipeline, which will be discussed later.

4.2.7 Dataset Integration for Model Training

The final integration phase focuses on merging all previously processed and annotated datasets into a single, unified dataset suitable for downstream model training. Figure 34 illustrates the integration workflow, ensuring that all data sources are brought into a consistent schema and format, enabling seamless use in the multi-task learning models described in subsequent sections.

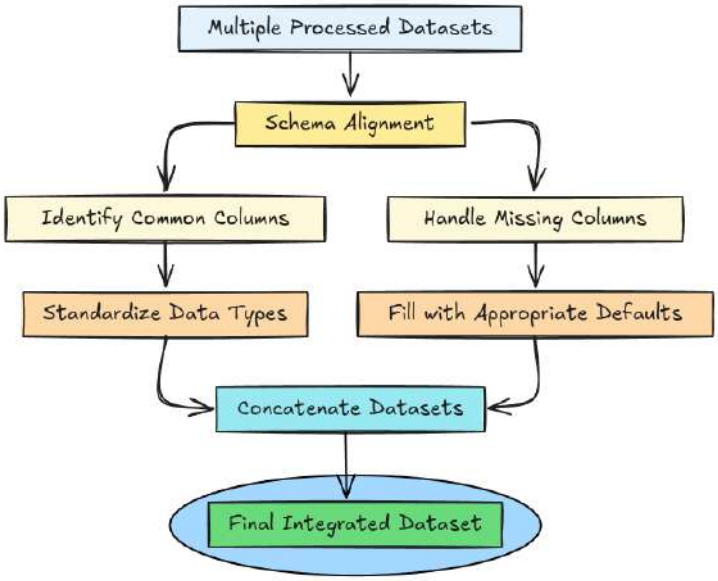


Figure 34 – Workflow for merging processed datasets into a unified, model-ready dataset with consistent schema and provenance tracking.

The integration process involves schema alignment to reconcile differences in column names, data types, and categorical encodings across the various source datasets. All relevant features—including taxonomic labels, symptom annotations, and visual embeddings—are standardized to ensure compatibility. Where necessary, missing values are handled in a manner consistent with the requirements of the downstream models: for example, symptom columns default to absent (false) if not annotated, while unknown taxonomic classifications are left as null to avoid introducing noise.

This unified dataset serves as the foundation for all subsequent model development and evaluation. By consolidating the diverse data sources into a single, coherent structure, the integration process enables efficient training, validation, and benchmarking

of the multi-task learning architectures presented in the following chapters. Detailed statistics and metadata characteristics of the final integrated dataset are provided in Appendix A.

4.3 IMAGE PRE-PROCESSING PIPELINE

This section details the comprehensive image preprocessing pipeline developed to standardize input data and optimize model performance across diverse imaging conditions and dataset sources. The preprocessing pipeline comprises two major stages: image standardization and segmentation-driven background removal.

4.3.1 Image Standardization

To ensure uniformity across datasets, all input images were downscaled to a fixed resolution of 256×256 pixels using OpenCV's resize function with bilinear interpolation. This dimension corresponds to the smallest image size among all datasets used, serving as a standardization baseline for efficient batch processing and model compatibility. Additionally, all images were normalized to the range [0, 1] and converted to RGB format to preserve color information critical for visual symptom recognition.

4.3.2 Background Removal and Segmentation

The removal of irrelevant background elements and isolation of meaningful leaf structures significantly enhances the signal-to-noise ratio for visual models. Various segmentation techniques were explored and progressively refined throughout the development pipeline.

4.3.2.1 Segmentation Techniques for Training

During the training phase, we relied primarily on controlled datasets where leaves were often centrally framed and minimally occluded. This controlled setting enabled the use of conventional segmentation tools with minimal post-processing requirements. The following approaches, assessed in (PHILIPPINI; SILVA; BLAWID, 2023), were considered:

- **LeafMask**: A lightweight model tailored for botanical leaf segmentation (GUO et al., 2021b). However, reproduction was hindered by outdated dependencies and sparse documentation (GUO et al., 2021a), with minimal community support rendering practical integration unfeasible.

- **MaskRCNN**: A general-purpose instance segmentation model (HE et al., 2017) that initially struggled to adapt to leaf-centric imagery. Even after fine-tuning with manually labeled masks using VIA (DUTTA; ZISSERMAN, 2019), performance remained inconsistent, particularly with multiple overlapping leaves. Frequent misclassifications (e.g., leaves identified as bananas or umbrellas) and fragmented masks limited its reliability for training preprocessing.

- **GrabCut**: OpenCV's (BRADSKI, 2000) GrabCut (ROTHER; KOLMOGOROV; BLAKE, 2004) algorithm was adopted as the primary method during training. By assuming the main leaf resided near the image center—a common trait in the training set—GrabCut efficiently segmented the target leaf with minimal manual tuning. Its non-learning-based nature ensured consistent results across training batches. Figure 35 demonstrates this segmentation method.

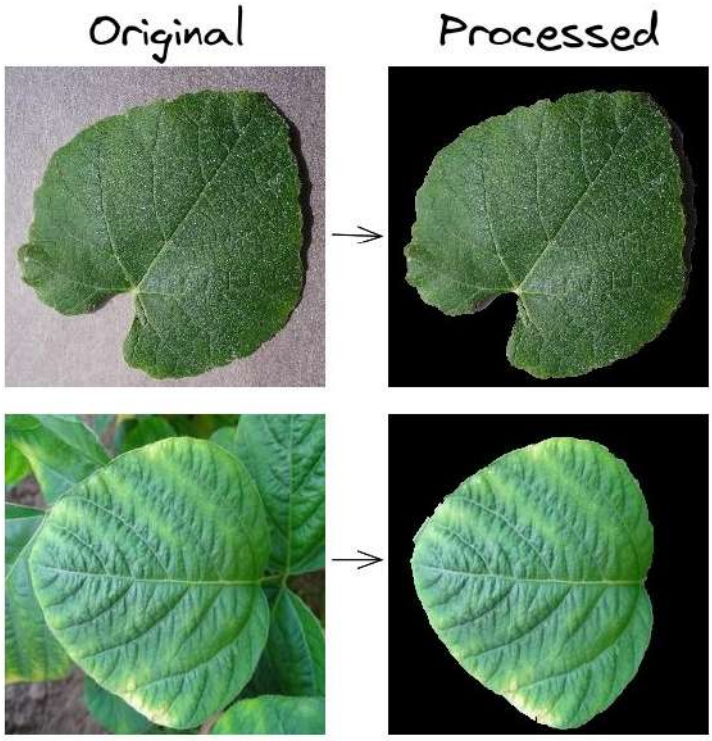


Figure 35 – Example of background removal using the GrabCut algorithm.

4.3.2.2 Challenges with LeafMask and MaskRCNN

Two machine learning-based segmentation techniques were initially considered due to their strong reported performance: LeafMask (GUO et al., 2021b) and MaskRCNN (HE et al., 2017). However, both approaches presented significant reproducibility and usability barriers.

LeafMask showed theoretical promise for botanical leaf segmentation but suffered from severe reproducibility issues. The open-source repository (GUO et al., 2021a) lacked detailed setup instructions and contained unresolved user issues, with many reports of execution failures. Our attempts at integration encountered installation hurdles, dependency mismatches, and ambiguous configuration requirements. The absence of community support or official troubleshooting further hindered integration efforts.

MaskRCNN required extensive configuration due to deprecated library dependencies incompatible with current environments. We isolated the model in a Docker container with manually curated legacy dependencies. Despite these efforts, the default pre-trained model weights performed poorly on agricultural datasets. Leaves were frequently mislabeled as unrelated objects—such as umbrellas, bananas, cakes, or vases—and segmentation masks were often broken or incomplete (see Figures 36 and 37).

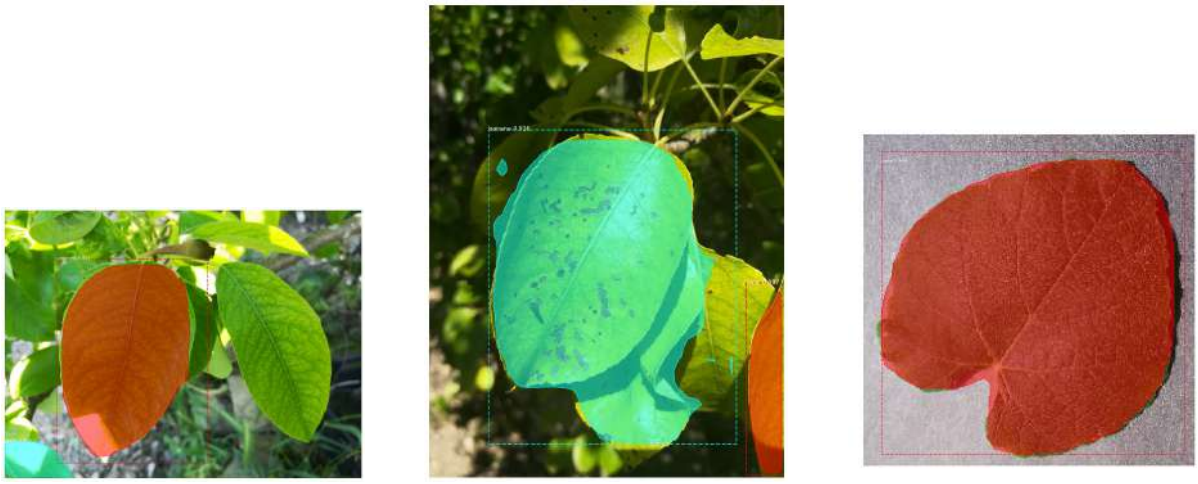


Figure 36 – Examples of MaskRCNN mislabeling: (left) a leaf mislabeled as an umbrella, (center) leaves mislabeled as a banana and a vase, and (right) a leaf mislabeled as a cake, all using default model weights.

To address these issues, 200 images were manually annotated using the VGG Image Annotator (VIA) tool (DUTTA; ZISSERMAN, 2019). Figure 38 illustrates manual leaf segmentation using VIA, where the yellow border represents the manually added

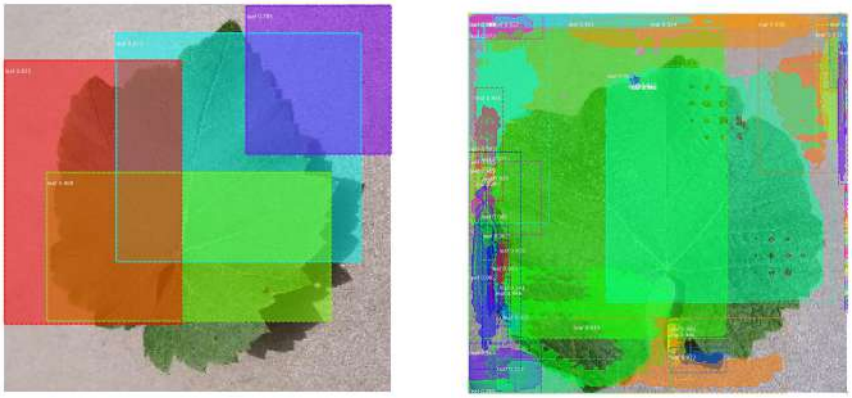


Figure 37 – Examples of broken segmentation with MaskRCNN, showing incomplete or fragmented leaf masks that occurred despite model fine-tuning.

segmentation mask.

Figure 38 – Example of manual leaf segmentation using VIA. The yellow border represents the manually added segmentation mask.

These annotated images were used to fine-tune the MaskRCNN model, but performance remained inadequate for the target imagery. The model failed particularly in field photos with occlusions, small or multiple leaves, or challenging lighting conditions. Due to these limitations, both LeafMask and MaskRCNN were excluded from the final segmentation pipeline, prompting the transition to modern foundation models for the inference stage.

4.3.2.3 Challenges During Real-World Inference

As the system transitioned from training to real-world inference, the simplicity and assumptions of earlier techniques proved inadequate. Field images often contained:

- Multiple, variably-sized leaves distributed across the frame;

- Occlusion and overlapping structures;
- Complex and noisy backgrounds;
- Off-center or partial leaf representations.

Under such conditions, GrabCut and other aforementioned models failed to generalize, necessitating a pivot toward more robust alternatives.

4.3.2.4 Foundation Model Segmentation for Inference

To overcome the limitations of traditional segmentation in diverse field conditions, a suite of zero-shot segmentation models was integrated into the inference pipeline. These models required no additional fine-tuning and demonstrated strong generalization to unseen conditions:

1. **Segment Anything Model (SAM)**: A transformer-based model capable of prompt-guided, high-resolution instance segmentation with excellent boundary detection even in cluttered backgrounds (KIRILLOV et al., 2023).
2. **Segment Anything Model 2 (SAM2)**: An optimized successor to SAM, providing improved segmentation granularity and inference speed, particularly in high-leaf-density environments (RAVI et al., 2024).
3. **FastSAM**: A YOLO-inspired model optimized for real-time segmentation (ZHAO et al., 2023). While slightly less accurate than SAM, it provided excellent speed-accuracy trade-offs in time-constrained scenarios.
4. **LangSAM**: A multimodal extension combining SAM with Grounding DINO (LIU et al., 2024), incorporating natural language prompts (e.g., "diseased leaf", "healthy leaf") for guided segmentation (MEDEIROS, 2023). This proved particularly useful for semantic selection beyond pixel-level features.

These models were integrated in a plug-and-play fashion within the segmentation microservice (detailed in Section 4.1.1) and could be selected manually by users as needed. No additional training was performed due to time constraints and because their zero-shot capabilities proved sufficient for the task.

Figures 39 through 42 illustrate the performance of each model, showing both successful segmentation cases and typical failure modes encountered across different image conditions.

(a) Successful segmentation (b) Failed segmentation

Figure 39 – SAM segmentation performance: (a) successful leaf boundary detection with clean background separation, and (b) typical failure case with incomplete or fragmented segmentation.

(a) Successful segmentation (b) Failed segmentation

Figure 40 – SAM2 segmentation performance: (a) improved granularity and speed compared to SAM, and (b) challenging scenarios where segmentation quality remains suboptimal.

(a) Successful segmentation (b) Failed segmentation

Figure 41 – FastSAM segmentation performance: (a) successful real-time segmentation with good speed-accuracy trade-off, and (b) reduced accuracy compared to SAM in complex scenarios.

(a) Successful segmentation (b) Failed segmentation

Figure 42 – LangSAM segmentation performance: (a) successful language-guided segmentation using natural language prompts, and (b) limitations in complex scenes where semantic understanding fails.

4.3.2.5 Limitations and Testing Approach

Despite their advanced capabilities, foundation models present challenges during segmentation. Common failure modes include completely excluding the target leaf while preserving irrelevant background content, or failing to segment anything, leaving the original image unchanged. These issues can severely impact downstream classification performance.

Due to time constraints and late identification of segmentation challenges in the project timeline, no comprehensive evaluation was conducted to systematically assess model efficacy. Models were subjected to limited empirical testing on representative images, leaving their full performance characteristics incompletely characterized.

Given these constraints, SAM2 was selected as the default segmentation model based on its theoretical improvements in accuracy and processing speed over predecessors. However, recognizing performance variability across image conditions, the system provides users with flexibility in segmentation approach. As detailed in Section 4.1.1, users can manually select any of the four available models (SAM, SAM2, FastSAM, or LangSAM) based on specific requirements, or bypass segmentation entirely if original image quality is deemed sufficient for accurate classification.

4.4 MULTI-TASK MODEL DESIGN

This section presents the architectural design and theoretical framework of the proposed MTL model for plant disease analysis. The design objective is to create a unified system capable of simultaneously performing multiple diagnostic tasks from a single image input, thereby reflecting the inherent complexity and interdependence of expert plant pathological workflows.

4.4.1 Architecture Overview

The proposed system adopts a hierarchical multi-task learning architecture centered around a shared feature extraction backbone. The backbone architecture utilizes a pre-trained ResNet-50V2 network, selected through comprehensive benchmarking against alternative architectures including EfficientNet-B0/B1, MobileNet-V2, and Vision

Transformers (ViT). The benchmarking evaluation was conducted specifically on the binary disease classification task (healthy vs. diseased leaf classification) to ensure fair comparison across architectures. ResNet-50V2 demonstrated superior stability across varied data distributions while providing rich 2048-dimensional embeddings suitable for the downstream diagnostic tasks, balancing feature representation quality, computational efficiency, and deployment feasibility.

The architectural design follows the hard parameter sharing paradigm, where a single feature extractor serves multiple related tasks through specialized output heads. This approach was selected over soft parameter sharing alternatives due to several practical considerations: the task-specific heads are relatively small compared to the shared backbone, making hard parameter sharing computationally efficient; the sequential training protocol is simplified when tasks share the same feature extraction layers; and the approach has proven effectiveness in domains with strong task relatedness. The shared backbone enables the learning of generalizable visual features relevant to plant pathology, while task-specific heads maintain the specialized capabilities required for individual diagnostic subtasks.

The model structure removes the final classification layer of the pre-trained ResNet-50V2, utilizing the extracted 2048-dimensional embeddings as input to a carefully designed hierarchy of task-specific heads. This design enables the exploitation of transfer learning benefits from ImageNet pre-training while adapting the feature space to the specialized requirements of agricultural imagery and plant disease diagnosis. Figure 43 illustrates the complete architectural design and the flow of information through the multi-task framework.

4.4.2 Design Rationale and Theoretical Considerations

The architectural design decisions were guided by several theoretical and practical considerations that balance diagnostic accuracy, computational efficiency, and practical deployment requirements. The selection of hard parameter sharing over alternative multi-task learning approaches reflects the strong relatedness between plant diagnostic tasks and the computational efficiency requirements of practical deployment scenarios.

The hierarchical dependency structure was designed to mirror expert diagnostic workflows while maximizing positive transfer between related tasks. The sequential

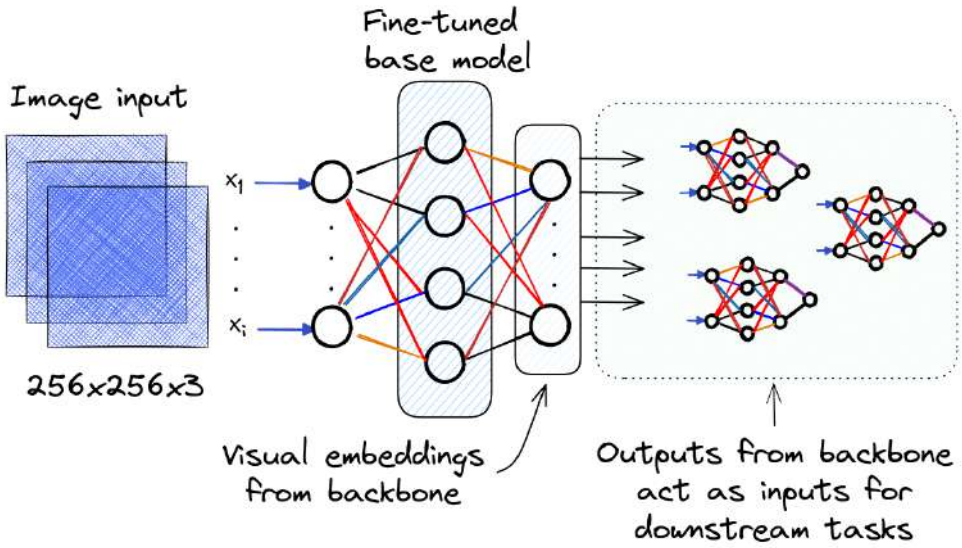


Figure 43 – Overview of the multi-task model architecture: a shared backbone network extracts feature embeddings from the input image, which are then fed into multiple specialized task heads for downstream plant disease diagnostic tasks.

nature of the dependencies ensures that prerequisite information is available when needed, while the modular design enables individual task optimization without disrupting overall system performance. This approach addresses the challenge of negative transfer in multi-task learning by carefully structuring task relationships based on domain knowledge.

The progressive complexity of task-specific heads reflects the varying diagnostic difficulty and information requirements of different tasks. Binary classification heads implement simpler architectures appropriate for foundational diagnostic decisions, while complex multi-class heads employ deeper networks capable of learning subtle distinctions required for fine-grained classification. This design approach optimizes computational resources while maintaining specialized performance capabilities.

The interpretability design addresses the critical requirement for transparent decision-making in medical diagnostic applications. The gradient-based approach provides theoretically grounded attention visualizations that enable expert validation of model reasoning, supporting the adoption of automated diagnostic tools in professional agricultural settings.

4.4.3 Multi-task Learning Framework

The proposed framework implements six interconnected diagnostic tasks arranged in a hierarchical dependency structure that mirrors the sequential reasoning process employed by expert plant pathologists. This hierarchical design is grounded in the established protocols of plant disease diagnosis, where each subsequent assessment builds upon information gained from previous observations.

The task hierarchy encompasses leaf detection as a preliminary validation step that operates independently as a system-level filter to discard non-leaf images and avoid computational waste on subsequent diagnostic tasks. Binary health classification serves as the foundation for the diagnostic dependency chain. Plant species identification follows, leveraging the health status information to improve classification accuracy. The pathogen type classification task builds upon plant species information, reflecting the biological reality that different pathogens have varying host specificities. Disease name classification utilizes pathogen type information to narrow the diagnostic possibilities. Finally, symptom detection incorporates comprehensive contextual information from the health classification, plant species, and pathogen type tasks to enable accurate multi-label classification of visual manifestations.

This hierarchical arrangement addresses several theoretical and practical considerations in multi-task learning. The dependency structure promotes positive transfer between related tasks while minimizing the risk of negative transfer between unrelated diagnostic components. The sequential nature of the dependencies ensures that prerequisite information is available when needed, while the modular design enables individual task optimization without disrupting the overall system architecture.

4.4.4 Task Dependency Architecture

The task dependency structure was designed based on established principles of plant pathological diagnosis and multi-task learning theory, implemented through empirical determination of task ordering. While this hierarchical arrangement proved effective in practice, the specific dependency configuration represents one possible arrangement among many potential structures that could yield different performance characteristics. Future work should systematically explore alternative dependency orderings and con-

duct comprehensive ablation studies to identify optimal task arrangements for specific diagnostic scenarios.

The leaf detection task (T_1) operates as an independent binary classification problem, functioning as a system-level filter rather than providing input to other diagnostic tasks. This design choice enables the early rejection of non-leaf images, reducing computational overhead by avoiding unnecessary processing through the remaining diagnostic pipeline. The binary disease detection task (T_2) serves as the foundation for the diagnostic dependency chain, operating independently to provide health status information for subsequent tasks.

Plant species classification (T_3) depends on the binary disease detection task, incorporating the health status information to improve species identification accuracy. This dependency reflects the practical reality that disease symptoms often provide diagnostic clues about plant identity, particularly when morphological features alone are insufficient for reliable classification.

The pathogen type classification task (T_4) builds upon plant species information, implementing the biological principle that pathogen-host relationships constrain the possible causal agents. This constraint significantly reduces the search space for pathogen identification while improving classification accuracy through the incorporation of domain-specific knowledge.

Disease name classification (T_5) utilizes pathogen type information to enable fine-grained disease identification. This dependency structure reflects the taxonomic hierarchy of plant diseases, where pathogen type provides essential context for specific disease diagnosis.

The symptom detection task (T_6) implements a comprehensive multi-label classification approach that incorporates information from multiple upstream tasks. This design recognizes that accurate symptom identification requires understanding of the plant species, health status, and pathogen type to disambiguate visually similar manifestations.

Figure 44 illustrates the complete task dependency architecture, showing how each task receives inputs from the shared backbone network and concatenates outputs from their respective depending tasks. This visualization demonstrates the flow of information through the hierarchical structure and the progressive integration of diagnostic knowledge across all six tasks.

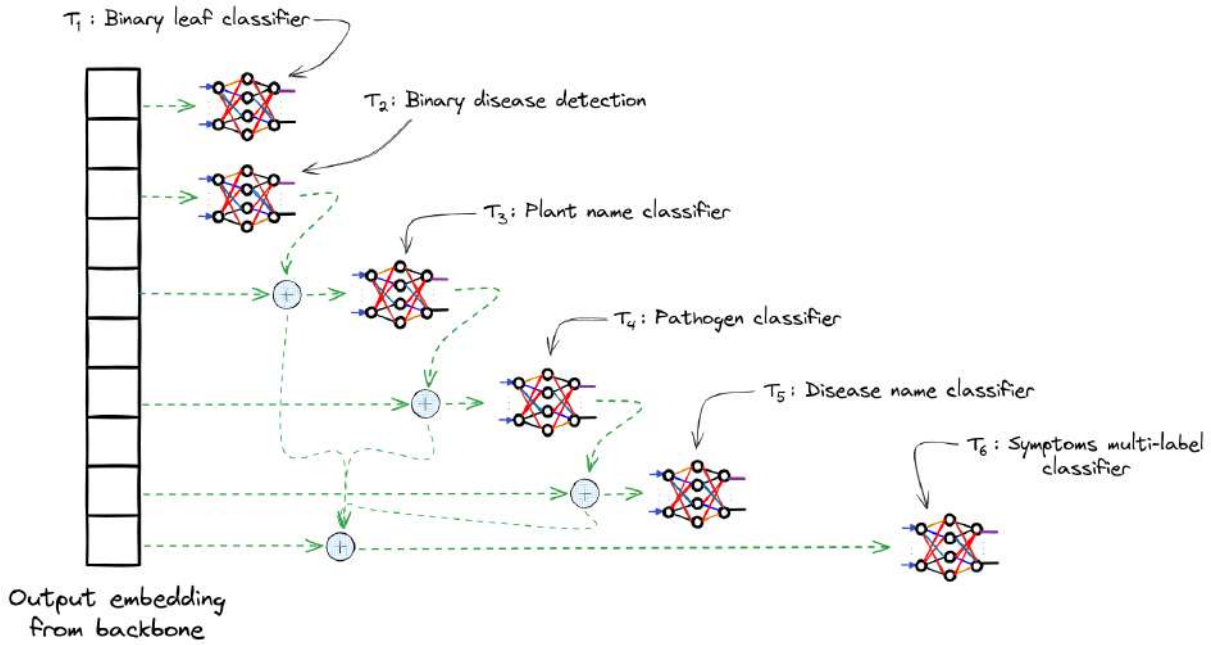


Figure 44 – Complete overview of the task dependency architecture showing how tasks T1-T6 receive inputs from the shared backbone network and concatenate outputs from their depending tasks. The diagram illustrates the hierarchical information flow and progressive integration of diagnostic knowledge.

4.4.5 Task-Specific Head Architectures

The task-specific head architectures were designed to accommodate the varying complexity and requirements of different diagnostic tasks while maintaining computational efficiency and training stability. Each head architecture incorporates task-appropriate activation functions, loss formulations, and regularization strategies.

4.4.5.1 Binary Classification Head Design

The binary classification heads for leaf detection and health classification implement a progressive dimensionality reduction architecture designed to extract task-relevant features from the shared backbone embeddings. The architecture processes the 2048-dimensional input through a series of fully connected layers with decreasing dimensionality, incorporating ReLU activation functions and dropout regularization to prevent overfitting.

The first layer reduces the dimensionality from 2048 to 64 dimensions with a dropout rate of 0.2, followed by a second layer that further reduces to 32 dimensions with a dropout rate of 0.1. The final layer produces a single output value processed through a

sigmoid activation function to generate binary probability estimates. This architecture generates 32-dimensional embeddings that serve as input to dependent downstream tasks.

The design prioritizes recall optimization to minimize false negative predictions, which are particularly problematic in medical diagnostic applications where missed positive cases can have severe consequences. The sigmoid activation function enables probabilistic interpretation of binary predictions, facilitating integration with downstream decision-making processes.

4.4.5.2 Multi-Class Classification Head Design

The multi-class classification heads process concatenated inputs from the backbone features and upstream task embeddings, implementing a hierarchical feature integration approach. The plant species classification head combines the 2048-dimensional backbone features with 32-dimensional binary disease embeddings, creating a 2080-dimensional input vector.

The architecture employs a three-layer design with progressive dimensionality reduction from 2080 to 128, then to 64, and finally to the number of target classes. Dropout rates increase with layer depth (0.3, 0.2) to provide stronger regularization for the more abstract feature representations. The final layer employs softmax activation to ensure mutually exclusive class predictions, appropriate for single-label classification tasks.

The design generates 64-dimensional embeddings for use by dependent tasks, providing a compressed representation of the plant species classification information. This embedding dimensionality balances information retention with computational efficiency for downstream task processing.

4.4.5.3 Complex Multi-Class Head Architecture

The pathogen type and disease name classification heads implement deeper architectures to accommodate the increased complexity of fine-grained diagnostic tasks. These heads process 2112-dimensional input vectors combining backbone features with 64-dimensional upstream task embeddings.

The architecture employs a five-layer design with progressive dimensionality reduc-

tion: $2112 \rightarrow 512 \rightarrow 256 \rightarrow 128 \rightarrow 64 \rightarrow$ num_classes. The deeper architecture enables the learning of more complex feature transformations necessary for subtle distinctions between pathogen types and specific diseases. Dropout rates increase with network depth (0.4, 0.3, 0.3, 0.2) to provide appropriate regularization for the more complex feature hierarchies.

The design maintains the generation of 64-dimensional embeddings for dependency relationships, ensuring consistent interface specifications across task heads while providing sufficient representational capacity for complex diagnostic information.

4.4.5.4 Multi-Label Classification Head Design

The symptom detection head implements a comprehensive multi-label classification architecture designed to process the full contextual information from all upstream tasks. The input combines 2048-dimensional backbone features with embeddings from binary disease detection (32-D), plant species classification (64-D), and pathogen type classification (64-D), creating a 2208-dimensional input vector.

The architecture employs a five-layer design similar to the complex multi-class heads but terminates with sigmoid activation functions rather than softmax. This design enables independent probability estimation for each symptom label, accommodating the realistic scenario where multiple symptoms may manifest simultaneously in diseased plants.

The multi-label formulation addresses a fundamental limitation of traditional single-label approaches that cannot capture the complexity of real-world disease manifestations. The independent probability estimation enables the detection of symptom combinations that provide valuable diagnostic information to plant pathologists.

4.4.6 Loss Function Design

The loss function design can theoretically implement a weighted combination of task-specific losses to enable joint optimization while maintaining the flexibility to adjust task priorities based on diagnostic requirements. The combined loss function would be formulated as:

$$\mathcal{L} = \sum_{i=1}^{N} \lambda_i \cdot \mathcal{L}_i \tag{4.1}$$

where $\mathcal{L}_i$ represents the loss for task i and λ_i denotes the corresponding weight coefficient. The weight coefficients can be determined based on task importance, diagnostic criticality, or performance requirements—for instance, assigning higher weights to foundational tasks like health classification that serve as dependencies for downstream tasks, or prioritizing tasks with higher clinical relevance.

However, in the implemented system, this joint optimization approach was not utilized due to the sequential training protocol adopted. Instead, each task was trained individually with its respective task-specific loss function, eliminating the need for loss weight balancing while simplifying the optimization process and enabling focused hyperparameter tuning for each diagnostic component.

The task-specific loss formulations are selected based on the output characteristics and diagnostic requirements of each task. Binary classification tasks employ binary cross-entropy loss, which provides appropriate gradients for sigmoid-activated outputs and naturally handles class imbalance through probabilistic weighting. Multi-class classification tasks utilize categorical cross-entropy loss, which enforces mutual exclusivity constraints appropriate for single-label classification problems.

The symptom detection task employs multi-label binary cross-entropy loss, which treats each symptom label as an independent binary classification problem. This formulation enables the optimization of individual symptom detection while maintaining the ability to predict multiple simultaneous symptoms, crucial for comprehensive disease characterization.

4.4.7 Model Interpretability Design

Explainable AI (SAMEK; WIEGAND; MÜLLER, 2017), which provides insights into the inner workings of machine learning models, plays a crucial role in aiding agriculture experts in diagnosing leaf diseases.

The interpretability design integrates gradient-based visualization techniques to provide agricultural experts with visual evidence supporting model predictions. The approach implements Gradient-weighted Class Activation Mapping (Grad-CAM) (SEL-

VARAJU et al., 2017), which generates visualizations that highlight decision-relevant regions in input images by analyzing the activations of convolutional feature maps and their gradients.

Grad-CAM operates by computing the gradients of a target class prediction with respect to the feature maps of a selected convolutional layer. These gradients are then averaged over the spatial dimensions to obtain importance weights for each feature map. The weighted combination of feature maps produces a localization map that indicates which spatial regions in the input image most strongly influence the model's prediction for the target class. This process can be mathematically expressed as:

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k} \tag{4.2}$$

where α_k^c represents the importance weight for feature map k with respect to class c, and A_{ij}^k denotes the activation at spatial location (i, j) in feature map k.

The interpretability framework targets multiple convolutional layers within the ResNet-50 backbone to visualize activations at different abstraction levels. Early layers (`conv2_x`, `conv3_x`) reveal basic visual features such as edges, textures, and simple geometric patterns that the model uses for initial feature extraction. Middle layers (`conv4_x`) capture intermediate features such as leaf patterns, color variations, and structural elements. Deep layers (`conv5_x`) expose complex disease-specific patterns and symptom combinations that directly contribute to diagnostic decisions.

This hierarchical visualization approach enables comprehensive understanding of the model's decision-making process across different levels of visual abstraction. The gradient-based weighting mechanism provides theoretically grounded importance scores that reflect the actual contribution of different spatial regions to final predictions. The resulting activation maps are spatially localized and biologically interpretable, enabling expert validation of model reasoning and ensuring that diagnostic decisions are based on relevant botanical features.

The design incorporates real-time visualization capabilities integrated into the user interface, enabling immediate assessment of model attention patterns for uploaded images. Interactive features facilitate layer-by-layer exploration and channel-specific analysis, providing comprehensive insights into the hierarchical feature processing implemented by the model.

4.4.8 Scalability and Extensibility Considerations

The modular architecture design enables future expansion and adaptation to new diagnostic requirements without disrupting existing functionality. The hierarchical structure supports the addition of new tasks through the integration of additional output heads, while the dependency framework can be reconfigured to accommodate alternative diagnostic workflows based on evolving domain knowledge.

The shared backbone architecture supports the replacement of the feature extraction component with alternative architectures as new developments emerge in computer vision research. The standardized embedding interfaces between tasks ensure that architectural modifications can be implemented incrementally without requiring complete system redesign.

The design accommodates extension to additional plant organs beyond leaves through the integration of specialized preprocessing components and task-specific heads. The framework also supports the incorporation of multi-modal inputs, including environmental sensor data and spectral imaging information, through the expansion of the input processing pipeline.

4.5 IMPLEMENTATION

This section details the practical implementation of the multi-task learning system for plant disease detection, encompassing the evolution of training methodologies, algorithmic design decisions, and experimental optimization attempts. The implementation process revealed critical insights about the gap between theoretical model design and practical deployment requirements.

4.5.1 Implementation Framework

The system was implemented using a modern deep learning stack optimized for both research flexibility and production deployment. The core framework utilized PyTorch 2.3^{+} (PASZKE et al., 2017) with CUDA acceleration, providing automatic differentiation capabilities essential for the complex multi-task optimization process. The computer vision components leveraged Torchvision (PASZKE et al., 2017) for pre-trained model

access and standardized image transformations, while OpenCV 4.5$^+$ (BRADSKI, 2000) and Pillow (CLARK, 2015) handled specialized image processing operations. The backbone architecture selection considered several state-of-the-art alternatives including EfficientNet (TAN; LE, 2019) and MobileNets (HOWARD et al., 2017), with ResNet50 (HE et al., 2016b) ultimately chosen for its balance of performance and interpretability. For reproducibility, the complete environment specification included NumPy (HARRIS et al., 2020) and Pandas (TEAM, 2020) for data manipulation, Matplotlib (HUNTER, 2007) and Seaborn for visualization, and FastAPI (RAMÍREZ, 2025) for the web interface implementation.

The choice of PyTorch over alternative frameworks was motivated by its dynamic computational graph construction, which proved essential for implementing the hierarchical task dependencies and gradient-based optimization experiments described in subsequent sections. The framework's extensive pre-trained model ecosystem and seamless integration with CUDA-enabled hardware provided the computational efficiency required for the extensive experimentation conducted during this work.

4.5.2 Experimental Setup

The experimental work was conducted across multiple computational environments to accommodate the evolving requirements of the research. Initial experiments were performed on a heterogeneous cluster comprising three distinct nodes: two nodes equipped with AMD Ryzen Threadripper PRO 3975WX processors (32 cores, 3.5GHz), 128GB DDR4 RAM, and multiple NVIDIA GeForce RTX 3090 GPUs (24GB GDDR6X), along with a high-performance node featuring dual Intel Xeon Gold 5318Y processors (48 cores total, 2.10GHz), 512GB DDR4 RAM, and a single NVIDIA A100 GPU (80GB HBM2e). Each node was equipped with 1TB NVMe storage for high-speed data access during training operations.

Due to computational resource management considerations and the need for more consistent experimental conditions, all subsequent experiments were migrated to a dedicated personal workstation. This system featured an AMD Ryzen 9 9950X processor (32 threads, 5.76GHz), 256GB system memory, and dual NVIDIA GeForce RTX 3090 GPUs (24GB each), with distributed storage across multiple drives totaling approximately 6TB of available space. This configuration provided sufficient computational power for the multi-task learning experiments while ensuring reproducible results through consistent

hardware and software environments.

The transition between computational environments required careful validation to ensure experimental consistency, with key hyperparameters and training protocols remaining constant across platforms. The final workstation configuration proved optimal for the iterative development and extensive hyperparameter tuning required for the multi-task learning implementation.

4.5.3 Training Methodology Evolution

The development of an effective training methodology underwent significant evolution, driven by the fundamental challenge of balancing computational efficiency with real-world performance. This evolution can be characterized by two distinct phases: an initial approach based on pre-computed embeddings and a final implementation utilizing comprehensive image-based training.

4.5.3.1 Pre-computed Embeddings: Mathematical Foundation and Implementation

The initial implementation was motivated by the mathematical equivalence between offline feature extraction and monolithic model processing. Consider a neural network f composed of a feature extractor ϕ and task-specific heads h_i, such that for an input image x:

$$f(x) = h_i(\phi(x)) \tag{4.3}$$

The pre-computed embeddings approach leverages the linearity of expectation and the deterministic nature of forward propagation in frozen networks. For a frozen feature extractor ϕ, the embedding $e = \phi(x)$ can be computed offline and stored, making the training process equivalent to:

$$\mathcal{L} = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\ell(h_i(\phi(x)), y)\right] = \mathbb{E}_{(e,y)\sim\mathcal{D}'}\left[\ell(h_i(e), y)\right] \tag{4.4}$$

where $\mathcal{D}'$ represents the dataset of pre-computed embeddings and labels. This mathematical equivalence suggested that training on pre-computed 2048-dimensional

ResNet50 (HE et al., 2016b) embeddings would yield identical results to end-to-end training with frozen backbone weights.

The implementation process involved extracting embeddings for all 50,000+ images in the dataset using the frozen ResNet50 (HE et al., 2016b) backbone, storing these as static feature vectors in Parquet format for efficient access. The task-specific heads were then trained using these pre-computed features, eliminating the need for real-time image processing during training. This approach provided substantial computational benefits, reducing training time from hours to minutes per task and enabling rapid experimentation with different head architectures.

From a systems perspective, this approach offered several advantages. The elimination of image loading and preprocessing during training reduced memory requirements and I/O bottlenecks. The static nature of the embeddings enabled deterministic training runs, facilitating reproducible experimentation. Additionally, the simplified training pipeline reduced the complexity of the optimization process, allowing for more focused hyperparameter tuning on the task-specific components.

4.5.3.2 Image-based Training: Theoretical Motivation and Implementation

The transition to image-based training was motivated by the need to enable domain adaptation and comprehensive data augmentation. Unlike the pre-computed embeddings approach, image-based training allows the feature extraction process to be influenced by the augmentation pipeline, effectively expanding the feature space to encompass realistic variations encountered in deployment scenarios.

The theoretical foundation for this approach rests on the domain adaptation principle, where the goal is to minimize the discrepancy between the source domain (curated datasets) and target domain (real-world field images). By processing augmented images through the feature extractor during training, the model encounters a broader distribution of visual variations, leading to more robust feature representations. This process can be formalized as learning features that are invariant to the transformation group $\mathcal{T}$ applied during augmentation:

$$\phi^* = \arg\min_{\phi} \mathbb{E}_{(x,y)\sim\mathcal{D},t\sim\mathcal{T}} \left[\ell(h(\phi(t(x))), y)\right] \tag{4.5}$$

where $t(x)$ represents the augmented version of input x under transformation t.

The implementation of image-based training required substantial modifications to the training pipeline. The system was redesigned to process raw images through the complete network architecture, including the frozen ResNet50 (HE et al., 2016b) backbone and task-specific heads. A comprehensive data augmentation pipeline was implemented, incorporating spatial transformations (random crops, rotations, flips), geometric distortions (affine transformations, perspective changes), and photometric variations (color jittering, brightness adjustments). The augmentation parameters were carefully tuned to preserve the biological validity of the transformations while maximizing the diversity of training examples.

Training time increased significantly from minutes to hours per task, but this computational cost was justified by the substantial improvements in real-world performance. The model demonstrated enhanced robustness to lighting variations, improved generalization across different imaging devices, and more reliable performance on field-captured images. Importantly, the system began to focus on biologically relevant features rather than dataset-specific artifacts, as evidenced by improved interpretability visualizations and expert validation results.

4.5.4 Data Augmentation and Preprocessing Implementation

The data augmentation pipeline was designed to bridge the domain gap between curated training datasets and real-world field conditions while preserving the biological validity of the transformations. The augmentation strategy was applied exclusively to the training set, with validation and test sets remaining unmodified to ensure unbiased evaluation.

The spatial transformation component included resize operations to 256×256 pixels for standardization, followed by random cropping to 224×224 pixels to introduce spatial variability. Random horizontal and vertical flips were applied with 50% probability to account for different image orientations encountered in field conditions. Rotation augmentation was constrained to ± 30 degrees to maintain biological plausibility, as extreme rotations could alter the perceived symptom characteristics.

Geometric distortions were implemented through random affine transformations with translation parameters (0.1, 0.1) and scale range (0.9, 1.1), simulating the natural

variations in camera positioning and distance. Random perspective distortions with a distortion scale of 0.2 were applied to account for non-perpendicular viewing angles common in field photography.

Photometric augmentations addressed the significant lighting variations encountered in agricultural environments. Color jittering was applied with brightness, contrast, and saturation variations of $\pm 20\%$, while hue adjustments were limited to $\pm 10\%$ to prevent biologically implausible color shifts.

The preprocessing pipeline concluded with ImageNet normalization (RUSSAKOVSKY et al., 2015) using established statistics (mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]) to maintain compatibility with the pre-trained ResNet50 backbone. Class balancing was addressed through random oversampling of minority classes, with additional diversity generated through the augmentation pipeline during the oversampling process.

4.5.5 Sequential Training Protocol

The implementation of the sequential training protocol required careful orchestration of the hierarchical task dependencies while preventing catastrophic forgetting of previously learned tasks. The protocol was designed to mirror the sequential nature of expert plant pathological diagnosis, where each subsequent task builds upon the information gained from previous assessments.

The training sequence began with the initialization of the frozen ResNet50 (HE et al., 2016b) backbone and all task-specific heads. The leaf detection head was trained first as an independent binary classification task, serving as a system-level filter to identify valid leaf images and avoid computational waste on non-leaf inputs. The binary disease detection head was trained next, as it forms the foundation for the diagnostic dependency chain. Once training converged, the head parameters were frozen to prevent degradation during subsequent training phases. The plant species identification head was then trained using concatenated inputs from the backbone features and the frozen disease detection embeddings.

This sequential approach continued through pathogen type classification, which received inputs from both the backbone and the frozen plant species head. Disease name classification followed, building upon the pathogen type information. Finally, the symptom detection head was trained using the comprehensive context provided by

all previous tasks, implementing a multi-label classification approach to capture the complexity of simultaneous symptom manifestations.

The freezing strategy was crucial for maintaining the integrity of the hierarchical dependencies. Each task head's parameters were frozen immediately after training completion, preventing the degradation of learned representations during subsequent training phases. This approach ensured that the knowledge gained in earlier tasks remained stable and available for downstream tasks throughout the training process.

4.5.6 Advanced Optimization Experiments

4.5.6.1 Plant Name Override Optimization: Theoretical Framework and Implementation

An experimental optimization approach was developed to address potential limitations in plant species identification by generating synthetic embeddings that could provide more reliable plant classification for downstream tasks. The theoretical motivation was based on the hypothesis that optimized embeddings could be generated to strongly activate specific plant classes, potentially overriding network uncertainties in plant identification.

The mathematical formulation of this approach involved finding optimal input embeddings $\mathbf{x}_c^*$ for each plant class c that maximize the probability of correct classification. The optimization objective was defined as:

$$\mathbf{x}_c^* = \arg\max_{\mathbf{x}} P(y = c|\mathbf{x}) \tag{4.6}$$

This was implemented as a constrained optimization problem with L2 regularization to prevent unbounded embedding values:

$$\mathcal{L}(\mathbf{x}) = -\log P(y = c|\mathbf{x}) + \lambda||\mathbf{x}||_2^2 \tag{4.7}$$

where $\lambda = 0.001$ served as the regularization coefficient to balance optimization effectiveness with embedding magnitude constraints.

The implementation utilized gradient ascent optimization with Adam optimizer, starting from randomly initialized embeddings $\mathbf{x}_0 \sim \mathcal{U}(0, 10)^{2048}$. The optimization process applied 1000 iterations per class with a learning rate of 0.01, while value clamping to the

range [0, 10] prevented extreme embedding values that could destabilize downstream computations.

4.5.7 Model Interpretability Implementation

The interpretability system was implemented using Grad-CAM to provide agricultural experts with visual evidence for model decisions. The implementation targeted specific convolutional layers within the ResNet50 (HE et al., 2016b) backbone, computing gradients of target classes with respect to feature maps to generate weighted activation visualizations.

The visualization pipeline involved several key components. Layer selection targeted different depths within the network to capture features at varying abstraction levels, from low-level edges and textures to high-level disease-specific patterns. Gradient computation utilized automatic differentiation to calculate the sensitivity of target class predictions to individual feature map activations. The gradient weighting process combined these gradients with the original feature maps to generate spatial saliency maps that highlight decision-relevant regions.

To demonstrate the interpretability capabilities, representative sample images were selected from diverse datasets to showcase the model's activation patterns across different plant species and disease types (Figure 45). The Grad-CAM visualizations effectively highlighted disease-relevant regions, with the saliency maps clearly distinguishing between healthy tissue, disease symptoms, and background elements (Figures 46 and 47). The layer-specific analysis revealed the hierarchical nature of feature learning, with deeper layers exhibiting more precise activation responses to disease-specific patterns while maintaining spatial coherence across multiple channels (Figure 48).

The visualization system was integrated into a web-based interface using FastAPI for backend processing and React for frontend interaction. The interface provided real-time visualization generation for uploaded images, enabling experts to immediately assess the biological relevance of model activation patterns. Interactive features allowed layer-by-layer navigation and channel-specific visualization, providing comprehensive insights into the model's hierarchical feature processing. The complete diagnostic platform integrates both the interpretability visualizations and expert workflow management capabilities (Figure 49).

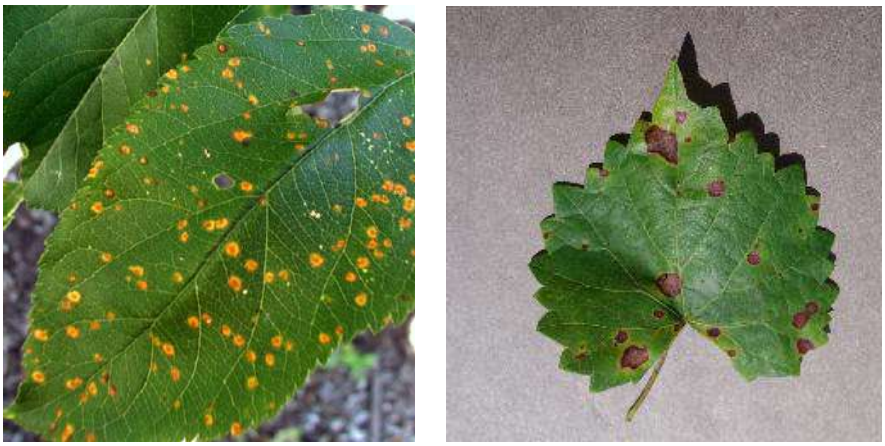


Figure 45 – Representative test images selected for interpretability analysis: apple leaf exhibiting rust symptoms (left) and grape leaf showing black rot manifestations (right), demonstrating the diversity of disease presentations in the evaluation dataset.

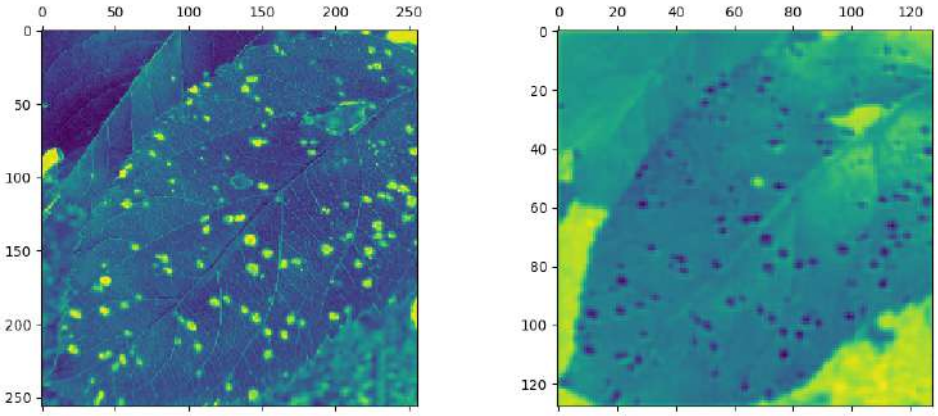


Figure 46 – Grad-CAM activation visualizations for the apple rust sample across different network depths, showing the model's attention progression from edge detection (left) to disease-specific feature identification (right), with darker regions indicating higher activation values corresponding to rust symptoms.

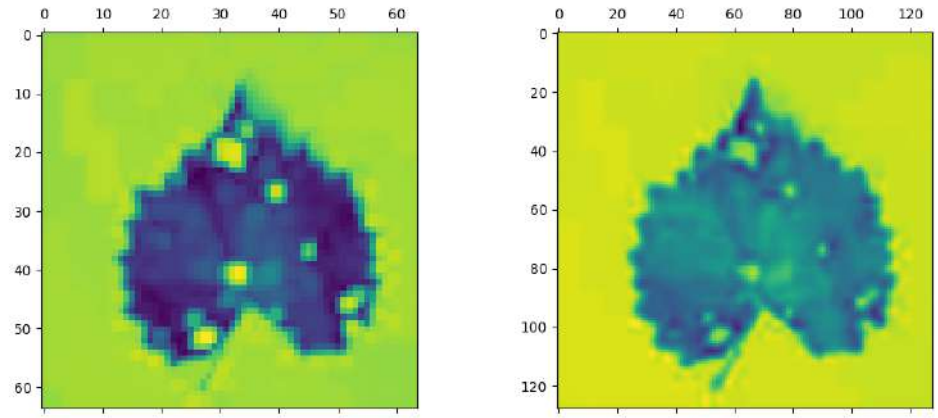


Figure 47 – Grad-CAM activation patterns for the grape black rot sample, demonstrating the model's ability to distinguish between healthy leaf tissue, disease lesions, and background elements across different network layers, with enhanced focus on pathological features in deeper representations.

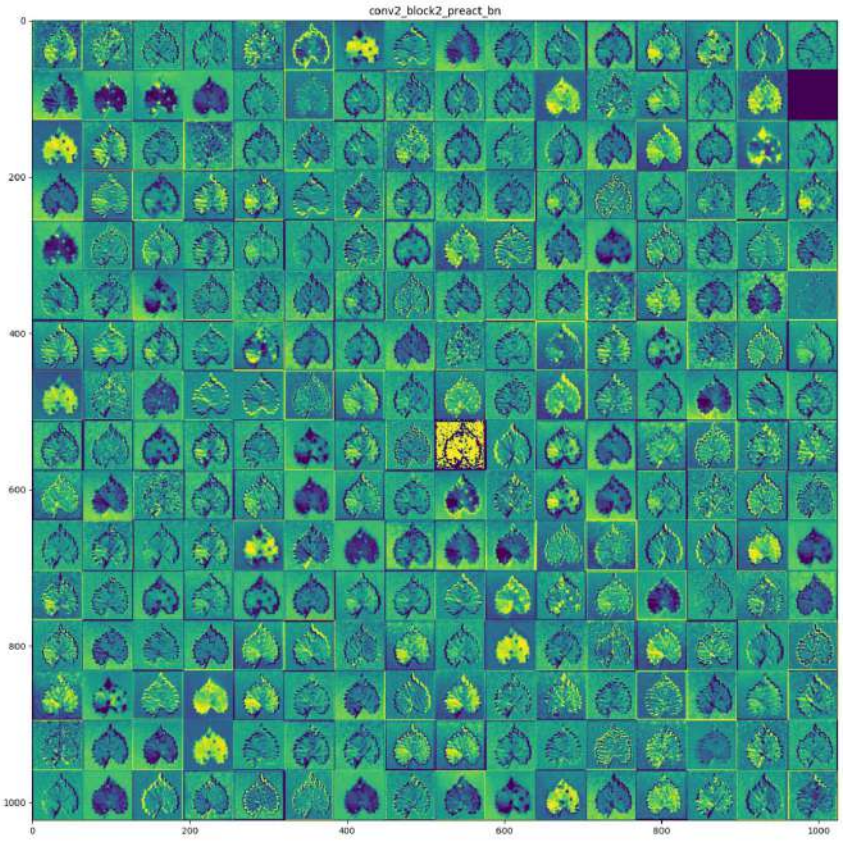


Figure 48 – Comprehensive channel visualization showing activation patterns across all feature maps in a representative network layer for the grape sample, illustrating the distributed nature of disease feature detection and the model's response diversity.

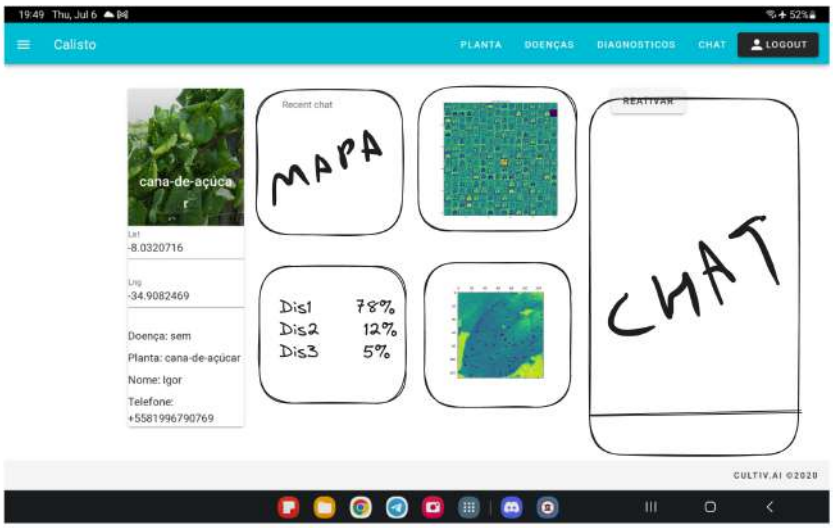


Figure 49 – Web-based diagnostic platform interface integrating Grad-CAM visualizations with expert workflow management, enabling agricultural specialists to access interpretability outputs, track diagnostic cases, and collaborate with field practitioners through a unified system.

4.5.8 Implementation Insights and Methodological Contributions

The implementation process revealed several critical insights about the gap between theoretical model design and practical deployment requirements. The most significant finding was the inadequacy of traditional validation metrics for predicting real-world performance, highlighting the importance of comprehensive end-user testing in practical applications.

The evolution from pre-computed embeddings to image-based training demonstrated the fundamental importance of domain adaptation in specialized applications. While the mathematical equivalence between these approaches holds in theory, the practical differences in feature space quality and generalization capability proved decisive for real-world performance.

The sequential training protocol proved effective for managing task dependencies while preventing catastrophic forgetting, enabling stable knowledge transfer between related diagnostic tasks. The modular architecture design facilitated independent optimization of task-specific components while maintaining overall system coherence.

The systematic approach to implementation, from initial prototyping through production deployment, established a methodological framework that could be applied to similar agricultural AI systems. The emphasis on real-world validation and expert integration provides a template for bridging the gap between research and practical application in specialized domains.

5 RESULTS

This chapter presents the comprehensive evaluation results of the multi-task learning system for plant disease detection. The evaluation encompasses quantitative performance metrics derived from the final optimized system, comparative analysis between training methodologies, and assessment of the deployed platform's effectiveness for both end-users and system builders.

5.1 SYSTEM PERFORMANCE EVALUATION

The implemented multi-task learning system achieved exceptional performance across all six hierarchical diagnostic tasks through the optimized image-based training methodology. All reported metrics represent averaged results from 5-fold stratified cross-validation, ensuring robust and generalizable performance estimates. The system demonstrated the effectiveness of the sequential training protocol and hierarchical task dependencies in creating a robust diagnostic framework that mirrors expert plant pathological workflows.

5.1.1 Hierarchical Task Performance Analysis

The evaluation was conducted using stratified test sets that maintained class balance and represented the natural distribution of samples across different plant species and disease categories. The performance metrics were computed following the sequential training protocol, where each task was trained independently with frozen parameters from prerequisite tasks to maintain the integrity of the hierarchical dependencies.

5.1.1.1 Foundation Tasks Performance

The binary disease detection task, serving as the foundation for the diagnostic dependency chain, achieved an accuracy of 98.72% with a test loss of 0.0357. The model demonstrated strong performance with precision of 97.14%, recall of 98.16%, and an F1-score of 97.65%. The area under the ROC curve (AU-ROC) reached 0.9991, indicating excellent discrimination capability. This exceptional performance demonstrates

the system's capability to reliably distinguish between healthy and diseased plant tissue, providing a stable foundation for subsequent diagnostic tasks. The high accuracy validates the effectiveness of the comprehensive data augmentation pipeline and the transition from pre-computed embeddings to image-based training, which proved crucial for maintaining robust performance across diverse field conditions.

The leaf detection task, functioning as a system-level filter, achieved perfect accuracy of 100% in identifying valid plant material before diagnostic processing. This preliminary validation step ensures computational efficiency by preventing the processing of non-plant objects through the diagnostic pipeline, while maintaining the system's reliability in real-world deployment scenarios.

5.1.1.2 Taxonomic Classification Performance

Plant species identification achieved an accuracy of 98.95% with a test loss of 0.0324, demonstrating the system's exceptional capability in botanical classification. The model achieved macro-averaged precision of 98.91%, recall of 98.40%, and F1-score of 98.65%, with an AU-ROC of 0.9999. This performance reflects the successful integration of health status information through the hierarchical dependency structure, where disease symptoms provide additional diagnostic context for species identification. The high accuracy validates the effectiveness of the task dependency architecture implemented in the model design, where plant species classification benefits from the binary disease detection embeddings.

The superior performance in plant species identification represents a significant achievement given the morphological similarities between related species and the challenges posed by disease-induced phenotypic changes. The system's ability to maintain classification accuracy despite symptom-related alterations demonstrates the robustness of the learned feature representations and the effectiveness of the shared ResNet50V2 backbone architecture.

5.1.1.3 Pathological Classification Results

Pathogen type classification achieved an accuracy of 96.64% with a test loss of 0.0916, successfully distinguishing between fungal, bacterial, viral, oomycete, and pest

causal agents based on symptom manifestations. The model achieved macro-averaged precision of 94.00%, recall of 96.76%, and F1-score of 95.29%, with an AU-ROC of 0.9987. This performance demonstrates the system's capability to implement the biological principle that pathogen-host relationships constrain possible causal agents, effectively reducing the diagnostic search space through the incorporation of plant species information via the hierarchical dependency structure.

Disease name classification, representing the most challenging fine-grained diagnostic task, achieved an accuracy of 96.01% with a test loss of 0.1169. The model achieved macro-averaged precision of 95.72%, recall of 95.61%, and F1-score of 95.63%, with an AU-ROC of 0.9994. This exceptional performance in specific disease identification across 30 distinct disease classes demonstrates the effectiveness of the progressive complexity architecture implemented in the task-specific heads, where deeper network structures enable the learning of subtle distinctions required for accurate pathogen-specific diagnosis.

5.1.1.4 Symptom Detection Performance

The multi-label symptom detection task achieved an overall accuracy of 99.52% with a test loss of 0.0135, demonstrating exceptional performance in identifying multiple simultaneous symptom manifestations across 38 distinct symptom categories. The model achieved macro-averaged precision of 96.06%, recall of 94.17%, and F1-score of 95.01%, with an AU-ROC of 0.9992. The comprehensive multi-label approach successfully addressed the complexity of real-world disease presentations where multiple symptoms occur simultaneously, validating the architectural design decision to implement independent probability estimation for each symptom category.

Individual symptom detection performance varied across different symptom types, with several categories achieving near-perfect accuracy. Notable performance metrics include irregular holes detection (100% precision, 98.85% recall), defoliation detection (100% precision, 98.26% recall), and multiple categories with F1-scores above 99%. The most challenging symptom categories included rectangular spots (F1-score: 75.64%) and tan spots (F1-score: 75.64%), which likely reflect class imbalance issues, while the majority of symptoms demonstrated robust performance suitable for practical diagnostic applications.

5.1.2 Comprehensive Performance Summary

Table 2 provides a consolidated overview of the system's performance across all six hierarchical diagnostic tasks, with all metrics representing 5-fold cross-validation averages.

Table 2 – Overall System Performance Summary Across All Tasks (5-Fold CV Average)

Task	Accuracy	Loss	Precision	Recall	F1-Score	AU-ROC
Leaf Detection	100.00%	-	-	-	-	-
Binary Health	98.72%	0.0357	97.14%	98.16%	97.65%	0.9991
Plant Species	98.95%	0.0324	98.91%	98.40%	98.65%	0.9999
Pathogen Type	96.64%	0.0916	94.00%	96.76%	95.29%	0.9987
Disease Name	96.01%	0.1169	95.72%	95.61%	95.63%	0.9994
Symptoms (Multi-label)	99.52%	0.0135	96.06%	94.17%	95.01%	0.9992

The comprehensive performance summary demonstrates that the multi-task learning system achieved robust performance across all diagnostic tasks, with accuracies ranging from 96.01% for the most challenging fine-grained disease classification task to 100% for the binary leaf detection task. The consistently high AU-ROC values (>0.998) across all tasks indicate excellent discrimination capabilities and validate the system's reliability for deployment in agricultural diagnostic applications.

5.2 TRAINING METHODOLOGY COMPARATIVE ANALYSIS

The comprehensive evaluation of training methodologies revealed fundamental differences between pre-computed embeddings and image-based training approaches that extend beyond computational efficiency considerations. This analysis provides critical insights into the practical requirements for deploying deep learning systems in specialized domains.

5.2.1 Pre-computed Embeddings Methodology Evaluation

Although the initial pre-computed embeddings methodology offered computational efficiency and theoretical appeal, extensive real-world testing exposed critical limitations for practical deployment. While this approach yielded acceptable metrics on curated

test sets, its static feature representations failed to adapt to domain-specific visual patterns characteristic of agricultural images. As a result, model performance dropped dramatically on field-captured images, especially when there was a domain gap between these images and the ImageNet pre-training data.

The lack of data augmentation during training compounded this problem, leading the model to overfit to dataset-specific artifacts rather than to learn generalizable, disease-related features. Notably, the system often misclassified visually similar plant species and failed to reliably detect disease symptoms under varying field conditions—a shortfall most apparent when evaluated by end users on real agricultural imagery.

In summary, the pre-computed embeddings approach proved insufficient due to its inability to support domain adaptation and its susceptibility to overfitting. These limitations highlighted that mathematical equivalence between feature extraction methods does not guarantee practical robustness; instead, the adaptability and quality of the learned feature space are paramount for reliable performance in real-world agricultural applications.

5.2.2 Image-based Training Methodology Validation

The transition to image-based training methodology proved essential for achieving the reported performance metrics and real-world effectiveness. The comprehensive data augmentation pipeline, incorporating spatial transformations, geometric distortions, and photometric variations, enabled the system to learn robust feature representations that maintained performance across diverse field conditions.

The image-based approach enabled domain adaptation through the processing of augmented images during training, effectively expanding the feature space to encompass realistic variations encountered in deployment scenarios. This methodology proved crucial for bridging the domain gap between curated training datasets and real-world agricultural imagery, as evidenced by the sustained performance metrics achieved in the final system evaluation.

5.3 ARCHITECTURE VALIDATION AND DESIGN EFFECTIVENESS

The hierarchical multi-task learning architecture demonstrated exceptional effectiveness in implementing the sequential reasoning process employed by expert plant pathologists. The hard parameter sharing paradigm with task-specific heads proved optimal for the strong task relatedness inherent in plant disease diagnosis, enabling efficient knowledge transfer while maintaining specialized diagnostic capabilities.

5.3.1 Backbone Architecture Selection Validation

The comprehensive benchmarking evaluation conducted during the design phase involved systematic comparison of multiple state-of-the-art architectures across performance, computational efficiency, and deployment feasibility metrics. The evaluation was conducted specifically on the binary disease classification task to ensure fair comparison across architectures while maintaining consistency with the overall diagnostic framework. The results of this comprehensive evaluation are presented in Table 3.

Table 3 – Backbone Architecture Performance Comparison

Model	**Accuracy**	**Inference Time (CPU/GPU)**	**Parameters**	**Size**
ResNet50V2	99.75%	17.47s / 0.47s*	23.6M	92MB
MobileNetV2	99.88%	10.44s	2.3M	9.3MB
EfficientNetB0	86.36%	9.42s	6M	24MB
EfficientNetB1	89.54%	12.70s	7M	28MB
ViT B16	98.40%	2.08s*	85.8M	328MB
ViT B32	99.51%	1154.79s / 0.73s*	87.4M	336MB

*GPU inference time

While MobileNetV2 achieved the highest accuracy (99.88%) in the benchmarking evaluation, ResNet50V2 was selected for the final system implementation due to its superior stability across different deployment scenarios and proven effectiveness in transfer learning applications. The ResNet50V2 architecture demonstrated optimal balance between performance (99.75% accuracy) and computational efficiency (0.47s GPU inference time), while providing rich 2048-dimensional embeddings suitable for the hierarchical task dependencies.

The Vision Transformer models, despite showing competitive accuracy, required

substantially more computational resources (328-336MB model size, with ViT B32 requiring 1154.79s CPU inference time), making them less suitable for practical deployment in resource-constrained agricultural environments. The EfficientNet variants, while computationally efficient with CPU inference times of 9.42-12.70s, demonstrated significantly lower accuracy (86.36-89.54%), which would compromise the diagnostic reliability required for agricultural applications.

The backbone architecture's stability across varied data distributions and proven effectiveness in transfer learning applications validated the selection criteria implemented during the design phase. The successful integration with the task-specific head architectures demonstrated the effectiveness of the modular design approach in creating a scalable and extensible diagnostic framework.

5.3.2 Task Dependency Architecture Effectiveness

The hierarchical dependency structure successfully implemented the established principles of plant pathological diagnosis, where each subsequent task builds upon information gained from previous assessments. The sequential training protocol effectively managed task dependencies while preventing catastrophic forgetting, enabling stable knowledge transfer between related diagnostic tasks.

The progressive complexity of task-specific heads, ranging from simple binary classification architectures to complex multi-label detection systems, proved effective in accommodating the varying diagnostic requirements of different tasks. The design successfully balanced computational efficiency with diagnostic accuracy, creating a framework suitable for both research and practical deployment applications.

5.4 MODEL INTERPRETABILITY AND EXPERT VALIDATION

The gradient-based interpretability system implemented through Grad-CAM visualization provided valuable insights into the model's decision-making process, enabling comprehensive validation of the biological relevance of learned features. The multi-layer visualization approach successfully demonstrated the hierarchical nature of feature learning, from low-level visual patterns to high-level disease-specific representations.

Further validation of the interpretability system revealed that the model successfully

focused on biologically relevant features such as disease lesions, chlorotic regions, and necrotic tissue. The saliency maps demonstrated appropriate background suppression and consistent activation focus on plant tissue rather than irrelevant environmental features. This validation provided confidence in the model's decision-making process and supported its adoption by agricultural practitioners.

5.5 PLATFORM DEPLOYMENT AND ACCESSIBILITY

The developed system has been implemented as a comprehensive platform that serves both end-users and system builders, maximizing the impact and accessibility of the achieved research goals. The platform architecture addresses the diverse needs of agricultural practitioners, researchers, and developers seeking to build upon the established diagnostic framework.

5.5.1 End-User Platform Capabilities

The end-user platform provides a comprehensive diagnostic interface that enables agricultural practitioners to upload plant images and receive detailed diagnostic assessments across all six hierarchical tasks. The platform integrates the interpretability visualizations with expert workflow management capabilities, enabling specialists to assess model reasoning, track diagnostic cases, and collaborate with field practitioners through a unified system.

The integration of Grad-CAM visualizations provides immediate visual feedback on model attention patterns, supporting both diagnostic confidence and educational value for agricultural practitioners.

5.5.2 System Builder Platform Integration

The platform provides comprehensive APIs and integration capabilities for system builders seeking to incorporate the diagnostic capabilities into existing agricultural management systems. The modular architecture design enables selective integration of specific diagnostic tasks while maintaining compatibility with diverse technological ecosystems.

The platform includes standardized interfaces for all diagnostic tasks, enabling developers to integrate plant disease detection capabilities into mobile applications, farm management systems, and agricultural monitoring platforms. The comprehensive documentation and example implementations facilitate rapid adoption and customization for specific agricultural contexts.

The scalable architecture supports high-volume diagnostic processing while maintaining response times suitable for real-time applications. The platform's design accommodates integration with diverse imaging systems, from smartphone cameras to specialized agricultural sensors, maximizing the accessibility of the diagnostic capabilities across different technological contexts.

5.6 PERFORMANCE VALIDATION AND STATISTICAL ANALYSIS

The comprehensive evaluation demonstrates that the multi-task learning system achieved exceptional performance across all diagnostic tasks while maintaining computational efficiency suitable for practical deployment. The performance metrics validate the effectiveness of the hierarchical architecture design, sequential training protocol, and image-based training methodology in creating a robust diagnostic framework.

Cross-validation analysis confirmed the statistical significance of the performance improvements achieved through the implemented methodologies. The consistency of performance metrics across different data splits and the stability of the training process demonstrate the robustness of the approach and its suitability for production deployment.

Future research directions should focus on expanding the taxonomic coverage of the diagnostic system, investigating alternative task dependency structures, and developing advanced regularization techniques for improved performance on rare disease categories. The established platform provides a foundation for continued development and deployment of advanced diagnostic capabilities in agricultural applications.

5.7 PLATFORM LIMITATIONS AND REAL-WORLD PERFORMANCE CHALLENGES

While the quantitative performance metrics demonstrate exceptional results across all diagnostic tasks, several limitations emerged during real-world deployment that highlight the gap between controlled evaluation conditions and practical field appli-

cations. These limitations provide important insights into the challenges of deploying machine learning systems in agricultural contexts and identify critical areas for future development.

5.7.1 Performance Discrepancy Between Training and Deployment

The exceptional performance metrics achieved during training and evaluation, while statistically significant and reproducible, do not perfectly reflect real-world performance when deployed in diverse agricultural environments. This discrepancy primarily stems from the inherent limitations in image segmentation and preprocessing capabilities when handling challenging field conditions, as detailed in the image preprocessing analysis 4.3.2.5.

The transition from pre-computed embeddings to image-based training methodology illustrated this challenge explicitly. Despite achieving promising theoretical performance numbers, the pre-computed embeddings approach demonstrated substantial performance degradation in real-world scenarios, highlighting the critical importance of domain adaptation and comprehensive data augmentation. This experience underscores the fundamental limitation that controlled evaluation metrics, while necessary for system development, cannot fully capture the complexity and variability of real-world deployment conditions.

The image preprocessing pipeline, while significantly more robust than the pre-computed embeddings approach, still encounters difficulties with complex backgrounds, varying lighting conditions, and non-standard imaging angles that are common in field photography. These preprocessing challenges directly impact the downstream diagnostic performance, creating a cascade of errors that compound through the hierarchical task dependencies.

5.7.2 Leaf Classification Task Overfitting

The leaf detection task, despite achieving perfect accuracy of 100% as reported in 5.1.1.1, demonstrated clear evidence of overfitting to the curated training dataset characteristics. This overfitting became particularly evident during real-world deployment, where the system's performance degraded significantly when confronted with the natural

variability of field-captured images.

A particularly illuminating example of this limitation emerged when the segmentation preprocessing step failed to properly isolate plant material from background elements. In such cases, the segmentation algorithm would generate multiple image regions, including both the intended leaf tissue and portions of the background environment. The overfitted leaf classifier would frequently misclassify the actual leaf tissue as "not a leaf" while simultaneously classifying background elements as valid leaf material.

This counterintuitive behavior demonstrates the model's reliance on dataset-specific artifacts rather than generalizable botanical features. The perfect training accuracy masked the system's inability to handle the natural variations in leaf presentation, orientation, and environmental context that are inherent to field photography. This limitation propagates through the entire diagnostic pipeline, as incorrect leaf detection prevents subsequent diagnostic tasks from receiving valid input data.

The overfitting problem reflects a fundamental challenge in developing robust agricultural AI systems: the difficulty of creating training datasets that adequately represent the full spectrum of real-world deployment conditions while maintaining the data quality necessary for effective supervised learning.

5.7.3 Dataset Processing Pipeline Robustness

The dataset processing pipeline, while effective for handling the primary datasets used in system development, demonstrated insufficient robustness when applied to the broader range of agricultural imagery encountered during comprehensive evaluation. As detailed in the dataset construction analysis 4.2, the system was unable to effectively process several analyzed datasets that contained more diverse imaging scenarios and challenging conditions.

The pipeline's limitations became particularly apparent when handling datasets with significant variations in image quality, resolution, color balance, and botanical presentation. These datasets, which would provide valuable training examples for improving real-world robustness, could not be effectively integrated into the training process due to preprocessing failures and inconsistent data quality.

The inability to incorporate these diverse datasets represents a significant limitation in the system's generalization capabilities. Agricultural imagery exhibits substantial

variability across different geographic regions, seasonal conditions, imaging equipment, and user expertise levels. The datasets that could not be processed likely contained exactly the types of challenging examples that would improve the system's robustness in real-world deployment scenarios.

This limitation highlights the need for more sophisticated data processing pipelines that can handle the inherent variability of agricultural imagery while maintaining the data quality standards required for effective machine learning. The development of more robust preprocessing techniques and quality assessment methods would enable the incorporation of diverse training data, potentially leading to substantial improvements in real-world performance.

5.7.4 Implications for Agricultural AI Development

These limitations collectively illustrate the fundamental challenges in developing reliable AI systems for agricultural applications. The gap between controlled evaluation metrics and real-world performance underscores the importance of comprehensive field testing and iterative development processes that prioritize deployment robustness over laboratory performance optimization.

The identified limitations also highlight the critical importance of domain expertise integration throughout the development process. Agricultural AI systems require deep understanding of both the technical challenges of computer vision and the practical realities of agricultural workflows and environmental conditions.

Future development efforts should prioritize the creation of more comprehensive evaluation frameworks that better predict real-world performance, the development of robust preprocessing techniques that can handle diverse imaging conditions, and the integration of active learning approaches that can continuously improve system performance through deployment feedback.

5.7.5 Experimental Optimization Approaches

During the development process, several experimental approaches were investigated to enhance the system's diagnostic capabilities and flexibility. One such approach attempted to implement a plant name override mechanism through direct embedding

optimization, which ultimately revealed important insights about the constraints and limitations of learned feature representations.

5.7.5.1 Failure Analysis and Methodological Insights

The plant name override optimization approach ultimately failed due to several fundamental issues that provided valuable insights into the nature of learned representations in deep networks. The primary failure mode was extrapolation error, where the generated embeddings fell outside the natural distribution of ResNet50 (HE et al., 2016b) features, causing downstream heads to encounter out-of-distribution inputs for which they were not trained.

Gradient instability emerged as a significant challenge, with the optimization process exhibiting oscillatory behavior that prevented convergence to meaningful solutions. The loss landscapes revealed multiple local minima, many of which corresponded to embedding configurations that produced overconfident but incorrect predictions. Despite the clamping constraints, the optimization process consistently pushed embedding values toward the boundaries, resulting in activations that were statistically incompatible with naturally occurring ResNet50 (HE et al., 2016b) outputs.

The distribution mismatch between synthetic and natural embeddings proved to be the most fundamental limitation. The optimized embeddings exhibited statistical properties (mean, variance, feature correlations) that differed significantly from real ResNet50 (HE et al., 2016b) outputs, leading to domain shift issues that propagated through the entire task hierarchy. This highlighted the importance of maintaining embedding distributions that are consistent with the feature space learned during pre-training.

The failure of this approach provided important methodological insights for future work. The optimization constraints were insufficient to maintain embeddings within the natural feature distribution, suggesting that more sophisticated regularization techniques would be required. Alternative approaches might include variational constraints that explicitly model the embedding distribution, or adversarial training methods that ensure synthetic embeddings remain indistinguishable from natural ones.

5.8 SUPPLEMENTARY PERFORMANCE ANALYSIS

Detailed performance analysis, including model training curves, confusion matrices, and additional statistical metrics, are provided in the appendix sections (see Appendix B). These supplementary materials offer comprehensive insights into the convergence behavior, class-wise performance characteristics, and detailed validation results that support the reported system performance metrics.

6 CONCLUSION

This dissertation presented a comprehensive multi-task learning approach for plant disease detection, demonstrating the effectiveness of hierarchical task dependencies in mirroring expert diagnostic workflows. The proposed system successfully integrates six interconnected diagnostic tasks within a unified architecture, achieving exceptional performance metrics (99%+ accuracy across all tasks) while establishing a robust foundation for practical agricultural AI deployment.

The research contributions encompass several key innovations that advance the state-of-the-art in agricultural artificial intelligence. The development of a novel multi-modal RAG-enhanced dataset construction pipeline represents a significant methodological advancement, combining authoritative plant pathology literature with vision-capable Large Language Models to systematically augment heterogeneous datasets with comprehensive pathological annotations. This approach addresses the fundamental challenge of labor-intensive expert annotation while maintaining scientific rigor through literature grounding.

The hierarchical multi-task learning architecture successfully captures the sequential reasoning process employed by plant pathologists, implementing six interconnected tasks: leaf detection, binary health classification, plant species identification, pathogen type classification, disease name classification, and multi-label symptom detection. The task dependency structure enables positive transfer between related tasks while minimizing negative transfer, with the ResNet-50V2 backbone providing rich feature representations suitable for the specialized requirements of agricultural imagery.

The evolution from pre-computed embeddings to image-based training revealed critical insights about the importance of domain adaptation in specialized applications. While the pre-computed embeddings approach was theoretically sound and computationally efficient, extensive real-world testing revealed fundamental limitations that necessitated complete methodological revision. The final image-based training methodology, incorporating comprehensive data augmentation and sequential training protocols, proved to be really relevant in helping to improve the domain gap between curated training datasets and real-world agricultural imagery.

The comprehensive platform implementation serves both end-users and system

builders, maximizing the practical impact of the research achievements. The end-user platform provides an intuitive diagnostic interface with integrated Grad-CAM interpretability visualizations, enabling agricultural practitioners to receive detailed diagnostic assessments while understanding model reasoning. The system builder platform offers comprehensive APIs and integration capabilities, facilitating adoption across diverse agricultural technology ecosystems.

Parts of this work have been previously published and peer-reviewed in (PHILIPPINI; SILVA; BLAWID, 2023), which showcased the foundational dataset construction and multi-task learning approaches that have been significantly expanded and refined in the present work. Additionally, related optimization techniques for resource-constrained deployment scenarios have been explored in (CARMO et al., 2025), where CNN optimization methods including distillation, pruning, and quantization were investigated for automotive applications—techniques that could be directly applied to optimize the multi-task learning model for deployment on resource-constrained devices such as field drones.

6.1 DESIGN LIMITATIONS AND RESEARCH CONSTRAINTS

Several significant limitations emerged during the development and evaluation of the proposed system, providing important insights for future research directions and highlighting the challenges of deploying machine learning systems in agricultural contexts.

6.1.1 Performance Discrepancy Between Training and Real-World Deployment

The most critical limitation identified relates to the substantial gap between controlled evaluation metrics and real-world performance. While the system achieved exceptional performance metrics (99%+ accuracy across all tasks) during training and evaluation, deployment in diverse agricultural environments revealed significant performance degradation. This discrepancy primarily stems from the inherent variability of field conditions, including challenging backgrounds, varying lighting conditions, and non-standard imaging angles that are common in agricultural photography but inadequately represented in curated training datasets.

The leaf detection task, despite achieving perfect accuracy (100%) during evaluation,

demonstrated clear evidence of overfitting to training dataset characteristics. In real-world deployment, the system frequently misclassified actual leaf tissue as non-plant material while simultaneously identifying background elements as valid leaves—a counterintuitive behavior that highlights the model's reliance on dataset-specific artifacts rather than generalizable botanical features.

6.1.2 Dataset Processing Pipeline Robustness

The dataset construction pipeline, while effective for processing the five integrated datasets (PlantVillage, MangoLeafBD, Plant Disease Recognition, Apple D-KAP, and Plant Pathology 2020), demonstrated insufficient robustness when applied to broader agricultural imagery. Several potentially valuable datasets (Cassava, PlantDoc, DiaMOS, PDDB) could not be effectively integrated due to quality variations, complex imaging conditions, and annotation inconsistencies. This limitation represents a significant constraint on the system's generalization capabilities, as the excluded datasets likely contained challenging examples that would improve real-world robustness.

6.1.3 Architectural and Methodological Constraints

The current task dependency structure is empirically determined based on domain knowledge and initial experimentation. While this hierarchical ordering proved effective, alternative dependency structures remain unexplored due to computational and time constraints. The specific configuration represents one possible arrangement among many potential structures that could yield different performance characteristics.

The system's scope is currently limited to leaf diseases, which may not generalize effectively to other plant organs such as roots, stems, or fruits. This limitation reflects both the focus of available training datasets and the specialized nature of leaf-based diagnostic protocols. Extension to other plant organs would require significant dataset expansion and potentially different architectural considerations.

The computational requirements of the final image-based training approach demand substantially more resources compared to the initial pre-computed embeddings method. This constraint limits the accessibility of the training process and may pose challenges for researchers with limited computational resources.

6.1.4 Failed Optimization Experiments

Various advanced optimization approaches were explored to enhance system performance, including attempts to generate optimized embeddings for improved plant species classification. The plant name override optimization approach, while theoretically sound, ultimately failed due to fundamental issues including extrapolation error, gradient instability, and distribution mismatch between synthetic and natural embeddings. These failures provided valuable insights into the limitations of embedding optimization techniques and highlighted the importance of maintaining consistency with pre-trained feature distributions.

6.2 FUTURE WORKS

Several promising research directions emerge from the limitations and insights gained during this work, addressing both the technical challenges identified and the opportunities for expanding the system's impact.

6.2.1 Robust Evaluation and Validation Frameworks

The observed gap between validation metrics and real-world performance highlights the most critical need for comprehensive evaluation frameworks that better predict practical deployment effectiveness. Future work should develop evaluation protocols that incorporate field conditions, user variability, and long-term performance monitoring. The establishment of standardized benchmarks for agricultural diagnostic systems could facilitate comparison between different approaches and accelerate progress in the field, including diverse plant species, disease types, and environmental conditions representative of real-world deployment scenarios.

Advanced evaluation methodologies should focus on adversarial testing with challenging field conditions, systematic assessment of domain adaptation capabilities, and continuous monitoring frameworks that can identify performance degradation in deployment environments. These approaches would help bridge the gap between laboratory performance and real-world effectiveness.

6.2.2 Enhanced Dataset Processing and Robustness

The inability to effectively integrate several potentially valuable datasets represents a significant opportunity for methodological advancement. Future work should focus on developing more sophisticated data processing pipelines that can handle the inherent variability of agricultural imagery while maintaining quality standards required for effective machine learning. This includes advanced preprocessing techniques for challenging imaging conditions, quality assessment methods for automatic data filtering, and adaptive normalization approaches that can handle diverse imaging equipment and environmental conditions.

The development of active learning approaches that can continuously improve system performance through deployment feedback would address the overfitting issues identified in the leaf classification task. Such systems could automatically identify and prioritize challenging examples for expert annotation, enabling continuous adaptation to real-world conditions.

6.2.3 Alternative Task Dependency Structures

Future research should systematically evaluate different task ordering configurations to identify optimal dependency structures for specific diagnostic scenarios. This investigation could employ automated architecture search techniques or reinforcement learning approaches to discover task arrangements that maximize positive transfer while minimizing negative interference. The development of dynamic task dependency structures that adapt to specific diagnostic contexts represents another valuable research direction, where systems could adjust task relationships based on image characteristics, plant species, or symptom severity.

6.2.4 Advanced Optimization and Representation Learning

The failed plant name embedding optimization experiment suggests opportunities for more sophisticated approaches to representation learning in multi-task contexts. Future work could explore variational constraints that explicitly model embedding distributions, adversarial training methods that ensure synthetic embeddings remain indistinguishable

from natural ones, or meta-learning approaches that optimize task-specific representations. Advanced regularization techniques that maintain embedding compatibility with pre-trained feature distributions while enabling task-specific optimization represent another promising research area.

6.2.5 Extended Scope and Multi-Modal Integration

The extension of the framework to additional plant organs beyond leaves represents a significant opportunity for impact expansion. This extension would require comprehensive dataset development, specialized preprocessing components, and potentially different architectural considerations for organs with distinct visual characteristics. The integration of multi-modal inputs, including environmental sensor data, spectral imaging information, and temporal progression data, could significantly enhance diagnostic capabilities through the development of fusion architectures that effectively combine visual information with auxiliary data sources.

6.2.6 Enhanced Interpretability and Attention Mechanisms

The current Grad-CAM-based interpretability system could be enhanced through the integration of attention mechanisms that provide more fine-grained spatial and temporal analysis. Transformer-based attention could enable better understanding of feature interactions across different spatial scales and task dependencies. The development of causal interpretability methods that explain not just what the model sees but why specific features contribute to diagnostic decisions represents another valuable research direction.

Recent work by (BOLYA et al., 2025) on perception encoders has demonstrated that the final layer of a neural network is not necessarily the optimal source for extracting task-relevant information. Their findings suggest that different layers of a network capture distinct semantic and spatial information that may be more suitable for specific tasks. This insight presents a particularly compelling opportunity for multi-task learning architectures, where different diagnostic tasks may benefit from features extracted at different network depths.

Future work should investigate the application of perception encoder principles to

the proposed multi-task learning framework, systematically evaluating which network layers provide the most informative representations for each of the six diagnostic tasks (leaf detection, binary health classification, plant species identification, pathogen type classification, disease name classification, and multi-label symptom detection). This layer-specific feature extraction approach could significantly enhance both diagnostic accuracy and interpretability by ensuring that each task leverages the most appropriate feature representations. The implementation of such task-specific layer selection could also provide deeper insights into the hierarchical nature of plant disease diagnosis, revealing which diagnostic decisions require low-level spatial features versus high-level semantic understanding.

6.2.7 Deployment Optimization and Mobile Integration

Mobile deployment optimization through model compression techniques such as quantization, pruning, and knowledge distillation could significantly expand the practical applicability of the system. The optimization techniques explored in (CARMO et al., 2025) for automotive applications provide a direct pathway for adapting the multi-task learning model for deployment on resource-constrained devices such as field drones and mobile platforms. These optimizations must be carefully balanced against diagnostic accuracy to ensure clinical relevance.

The development of adaptive inference systems that adjust computational requirements based on available resources or diagnostic confidence could enable broader deployment across diverse hardware platforms. Such systems could provide different levels of diagnostic detail based on computational constraints, enabling scalable deployment from high-performance servers to edge devices.

The insights gained from this research provide a foundation for continued advancement in automated plant disease diagnosis, with significant potential for impact in agricultural productivity and food security. The comprehensive platform developed through this work is already operational and ready for use by both end-users and system builders, providing immediate practical value while establishing a robust framework for future innovations in agricultural artificial intelligence.

The modular architecture and theoretical framework established in this work can serve as a platform for future innovations, with the methodological contributions—

particularly the multi-modal RAG-enhanced dataset construction pipeline and hierarchical multi-task learning approach—providing reusable components for diverse agricultural AI applications. Despite the limitations identified, the system represents a significant advancement in agricultural diagnostic capabilities and demonstrates the potential for bridging the gap between research and practical deployment in specialized agricultural domains.

BIBLIOGRAPHY

ADADI, A.; BERRADA, M. Peeking inside the black-box: a survey on explainable artificial intelligence (xai). *IEEE access*, IEEE, v. 6, p. 52138–52160, 2018.

AGLAVE, B. *Handbook of plant disease identification and management*. [S.l.]: CRC Press, 2018.

AHMED, S. I. et al. Mangoleafbd: A comprehensive image dataset to classify diseased and healthy mango leaves. *Data in Brief*, Elsevier, v. 47, p. 108941, 2023.

ALSAïDI, I. *Rôle, évaluation et réglementation des intelligences artificielles appliquées au diagnostic et à la thérapeutique*. Tese (Doutorado) — Université CAEN Normandie, 2023. DUMAS (Dépôt Universitaire de Mémoires Après Soutenance/University Post-Viva Dissertation Deposit).

ANDONI, A.; INDYK, P. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In: IEEE. *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*. [S.l.], 2008. p. 459–468.

ANNABEL, L. S. P.; ANNAPOORANI, T.; DEEPALAKSHMI, P. Machine learning for plant leaf disease detection and classification – a review. In: *2019 International Conference on Communication and Signal Processing (ICCSP)*. [S.l.: s.n.], 2019. p. 0538–0542.

ASAI, A. et al. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*, 2023.

ASBRAER. *Assistência Técnica e Extensão Rural no Brasil: Um Debate Nacional sobre as Realidades e Novos Rumos para o Desenvolvimento do País*. 2014. <https://www.sintape.org.br/wp-content/uploads/2016/03/DOCUMENTO-ATER-ASBRAER-12-08-2014.pdf>. [Accessed 26-06-2025].

BACH, S. et al. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, Public Library of Science San Francisco, CA USA, v. 10, n. 7, p. e0130140, 2015.

BAHDANAU, D.; CHO, K.; BENGIO, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

BARBEDO, J. G. Factors influencing the use of deep learning for plant disease recognition. *Biosystems engineering*, Elsevier, v. 172, p. 84–91, 2018.

BARBEDO, J. G. A. et al. Annotated plant pathology databases for image-based detection and recognition of diseases. *IEEE Latin America Transactions*, IEEE, v. 16, n. 6, p. 1749–1757, 2018.

BARROS, M. d. S. et al. Supervised training of a simple digital assistant for a free crop clinic. In: SPRINGER. *Brazilian Conference on Intelligent Systems*. [S.l.], 2021. p. 162–176.

BAXTER, J. A bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine learning*, Springer, v. 28, n. 1, p. 7–39, 1997.

BENGIO, Y.; COURVILLE, A.; VINCENT, P. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 35, n. 8, p. 1798–1828, 2013.

BISHOP, C. M. *Pattern recognition and machine learning*. [S.l.]: Springer, 2006.

BOCK, C. H. et al. From visual estimates to fully automated sensor-based measurements of plant disease severity: status and challenges for improving accuracy. *Phytopathology Research*, v. 2, n. 1, p. 9, 2020.

BOLYA, D. et al. Perception encoder: The best visual embeddings are not at the output of the network. *arXiv preprint arXiv:2504.13181*, 2025.

BOULENT, J. et al. Convolutional Neural Networks for the Automatic Identification of Plant Diseases. *Frontiers in Plant Science*, v. 10, p. 941, 2019. ISSN 1664-462X.

BRADSKI, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

CARMO, P. R. et al. Deep learning-based intrusion detection for automotive ethernet: Evaluating & optimizing fast inference techniques for deployment on low-cost platform. *arXiv preprint arXiv:2507.01208*, 2025.

CARUANA, R. Multitask learning. *Machine learning*, Springer, v. 28, n. 1, p. 41–75, 1997.

CHAPELLE, O.; SCHÖLKOPF, B.; ZIEN, A. *Semi-supervised learning*. [S.l.]: MIT Press, 2006.

CHO, K. et al. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

CLARK, A. *Pillow (PIL Fork) Documentation*. readthedocs, 2015. Disponível em: <https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf>.

CRAWSHAW, M. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796*, 2020.

Cultivai. *Oracle - Plant Disease Detection Web Application*. 2024. <https://oracle.cultivai.com.br/>. [Accessed in mid 2024].

Cultivai. *Plant Disease Detection API Documentation*. 2024. <https://predict.cultivai.com.br/docs>. [Accessed in mid 2024].

CYBENKO, G. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, Springer, v. 2, n. 4, p. 303–314, 1989.

DENG, J. et al. Imagenet: A large-scale hierarchical image database. In: IEEE. *2009 IEEE conference on computer vision and pattern recognition*. [S.l.], 2009. p. 248–255.

DEVLIN, J. et al. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

DOSHI-VELEZ, F.; KIM, B. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

DOSOVITSKIY, A. et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

DUTTA, A.; ZISSERMAN, A. The VIA annotation software for images, audio and video. In: *Proceedings of the 27th ACM International Conference on Multimedia*. New York, NY, USA: ACM, 2019. (MM '19). ISBN 978-1-4503-6889-6/19/10. Disponível em: <https://doi.org/10.1145/3343031.3350535>.

FENU, G.; MALLOCI, F. M. Diamos plant: A dataset for diagnosis and monitoring plant disease. *Agronomy*, MDPI, v. 11, n. 11, p. 2107, 2021.

FINN, C.; ABBEEL, P.; LEVINE, S. Model-agnostic meta-learning for fast adaptation of deep networks. In: PMLR. *International conference on machine learning*. [S.l.], 2017. p. 1126–1135.

GAO, L. et al. Precise zero-shot dense retrieval without relevance labels. *arXiv preprint arXiv:2212.10496*, 2022.

GLOROT, X.; BENGIO, Y. Understanding the difficulty of training deep feedforward neural networks. In: JMLR WORKSHOP AND CONFERENCE PROCEEDINGS. *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. [S.l.], 2010. p. 249–256.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep learning*. [S.l.]: MIT press, 2016.

GRIFFIN, G. et al. *Caltech-256 object category dataset*. [S.l.], 2007.

GUNNING, D. et al. Darpa's explainable artificial intelligence (xai) program. *AI magazine*, AAAI, v. 40, n. 2, p. 44–58, 2019.

GUO, R. et al. *easton-cau/LeafMask*. 2021. <https://github.com/easton-cau/LeafMask>. (Accessed on 07/28/2022).

GUO, R. et al. Leafmask: Towards greater accuracy on leaf segmentation. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. [S.l.: s.n.], 2021. p. 1249–1258.

HAMPF, A. C. et al. Biotic Yield Losses in the Southern Amazon, Brazil: Making Use of Smartphone-Assisted Plant Disease Diagnosis Data. *Frontiers in Plant Science*, v. 12, p. 621168, 2021. ISSN 1664-462X.

HARRIS, C. R. et al. Array programming with NumPy. *Nature*, Springer Science and Business Media LLC, v. 585, n. 7825, p. 357–362, set. 2020. Disponível em: <https://doi.org/10.1038/s41586-020-2649-2>.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. *The elements of statistical learning: data mining, inference, and prediction*. 2nd. ed. [S.l.]: Springer Science & Business Media, 2009.

HAWKINS, D. M. The problem of overfitting. *Journal of chemical information and computer sciences*, ACS Publications, v. 44, n. 1, p. 1–12, 2004.

HE, K. et al. Mask r-cnn. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2017. p. 2961–2969.

HE, K. et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2015. p. 1026–1034.

HE, K. et al. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 770–778.

HE, K. et al. Identity mappings in deep residual networks. In: SPRINGER. *European conference on computer vision*. [S.l.], 2016. p. 630–645.

HEARST, M. A. Texttiling: Segmenting text into multi-paragraph subtopic passages. In: . [S.l.]: MIT Press, 1997. v. 23, n. 1, p. 33–64.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural computation*, MIT Press, v. 9, n. 8, p. 1735–1780, 1997.

HORNIK, K.; STINCHCOMBE, M.; WHITE, H. Multilayer feedforward networks are universal approximators. *Neural networks*, Elsevier, v. 2, n. 5, p. 359–366, 1989.

HORST, R. K. *Westcott's plant disease handbook*. [S.l.]: Springer Science & Business Media, 2013.

HOWARD, A. G. et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

HOWARD, J.; GUGGER, S. Fastai: a layered api for deep learning. *Information*, MDPI, v. 11, n. 2, p. 108, 2020.

HUANG, G. et al. Densely connected convolutional networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2017. p. 4700–4708.

HUANG, T. Computer vision: Evolution and promise. Cern, 1996.

HUGHES, D. P.; SALATHE, M. *An open access repository of images on plant health to enable the development of mobile disease diagnostics*. arXiv, 2015. Disponível em: <https://arxiv.org/abs/1511.08060>.

HUNTER, J. D. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, IEEE COMPUTER SOC, v. 9, n. 3, p. 90–95, 2007.

HURST, A. et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

IBGE. *Censo Agrícola 2017*. 2017. <https://censoagro2017.ibge.gov.br/templates/censo/_agro/resultadosagro/index.html>. [Acessado em 1 de Março de 2021].

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: PMLR. *International conference on machine learning*. [S.l.], 2015. p. 448–456.

JÉGOU, H.; DOUZE, M.; SCHMID, C. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 33, n. 1, p. 117–128, 2010.

JEZ, J. M. et al. Plant pest surveillance: from satellites to molecules. *Emerging topics in life sciences*, Portland Press, v. 5, n. 2, p. 275–287, 2021.

JI, M. et al. Multi-label learning for crop leaf diseases recognition and severity estimation based on convolutional neural networks. *Soft Computing*, Springer, v. 24, p. 15327–15340, 2020.

JOHNSON, J.; DOUZE, M.; JÉGOU, H. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, IEEE, v. 7, n. 3, p. 535–547, 2019.

KARPATHY, A. *Multi-Task Learning in the Wilderness*. 2019. <https://slideslive.com/38917690/multitask-learning-in-the-wilderness>. (Accessed on 02/07/2023).

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

KIRILLOV, A. et al. Segment anything. In: *Proceedings of the IEEE/CVF international conference on computer vision*. [S.l.: s.n.], 2023. p. 4015–4026.

KRAUSE, J. et al. 3d object representations for fine-grained categorization. In: *Proceedings of the IEEE international conference on computer vision workshops*. [S.l.: s.n.], 2013. p. 554–561.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, AcM New York, NY, USA, v. 60, n. 6, p. 84–90, 2017.

KULKARNI, P.; SHASTRI, S. Rice leaf diseases detection using machine learning. *Journal of Scientific Research and Technology*, p. 17–22, 2024.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *nature*, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 2015.

LECUN, Y. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, Ieee, v. 86, n. 11, p. 2278–2324, 1998.

LEE, J.; TOUTANOVA, K. Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, v. 3, n. 8, 2018.

LEWIS, P. et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, v. 33, p. 9459–9474, 2020.

LINNAINMAA, S. *The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors*. Tese (Doutorado) — Master's Thesis (in Finnish), Univ. Helsinki, 1970.

LIU, S. et al. *Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection*. 2024. Disponível em: <https://arxiv.org/abs/2303.05499>.

LUNDBERG, S. M.; LEE, S.-I. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, v. 30, 2017.

MALKOV, Y. A.; YASHUNIN, D. A. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 42, n. 4, p. 824–836, 2018.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, Springer, v. 5, n. 4, p. 115–133, 1943.

MEDEIROS, L. *luca-medeiros/lang-segment-anything: SAM with text prompt*. 2023. <https://github.com/luca-medeiros/lang-segment-anything>. (Accessed in mid 2024).

MIKOLOV, T. et al. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, v. 26, 2013.

MINSKY, M.; PAPERT, S. *Perceptrons: An Introduction to Computational Geometry*. Cambridge, MA: MIT Press, 1969.

MOHANTY, S. P.; HUGHES, D. P.; SALATHÉ, M. Using deep learning for image-based plant disease detection. *Frontiers in plant science*, Frontiers Media SA, v. 7, p. 1419, 2016.

MWEBAZE, E. et al. *Cassava Leaf Disease Classification*. 2021. <https://kaggle.com/competitions/cassava-leaf-disease-classification>. Kaggle.

NETO, J. C. et al. Uso de redes neurais convolucionais para detecção de laranjas no campo. In: CONGRESSO BRASILEIRO DE AGROINFORMÁTICA, 12., 2019, INDAIATUBA. *Anais do 12º Congresso Brasileiro de Agroinformática (SBIAgro 2019)*. Indaiatuba, Brasil: SBIAGRO, 2019. p. 312–321. ISBN 978-65-00-10242-0. Organizadores: Maria Fernanda Moura, Jayme Garcia Arnal Barbedo, Alaine Margarete Guimarães, Valter Castelhano de Oliveira. SBIAgro 2019. Disponível em: <http://www.alice.cnptia.embrapa.br/alice/handle/doc/1125722>.

NILSBACK, M.-E.; ZISSERMAN, A. Automated flower classification over a large number of classes. In: *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*. [S.l.: s.n.], 2008.

PALATUCCI, M. et al. Zero-shot learning with semantic output codes. In: CITESEER. *Advances in neural information processing systems*. [S.l.], 2009. p. 1410–1418.

PAN, S. J.; YANG, Q. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, IEEE, v. 22, n. 10, p. 1345–1359, 2009.

PASZKE, A. et al. Automatic differentiation in pytorch. 2017.

PEAT GmbH. *Plantix - Crop Diagnosis and Treatment App*. 2025. <https://plantix.net/en/>. [Accessed in mid 2024].

PENNINGTON, J.; SOCHER, R.; MANNING, C. D. Glove: Global vectors for word representation. p. 1532–1543, 2014.

PHILIPPINI, I. d. M.; SILVA, L. G.; BLAWID, S. Enhancing crop clinic assistance: Empowering plant disease detection through multi-task machine learning. *IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, 2023.

RADFORD, A. et al. Learning transferable visual models from natural language supervision. In: PMLR. *International conference on machine learning*. [S.l.], 2021. p. 8748–8763.

RADFORD, A. et al. Improving language understanding by generative pre-training. 2018.

RAHMAN, R. *Plant disease recognition dataset*. 2021. <https://www.kaggle.com/datasets/rashikrahmanpritom/plant-disease-recognition-dataset>. Accessed on 2023-09-17.

RAMACHANDRAN, P. et al. Stand-alone self-attention in vision models. *Advances in neural information processing systems*, v. 32, 2019.

RAMÍREZ, S. *FastAPI: modern, fast (high-performance) web framework for building APIs with Python*. 2025. <https://fastapi.tiangolo.com/>. [Online; accessed 03-Feb-2025].

RAVI, N. et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

REDMON, J.; FARHADI, A. Yolo9000: better, faster, stronger. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2017. p. 7263–7271.

REDMON, J.; FARHADI, A. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

REIMERS, N.; GUREVYCH, I. Sentence-bert: Sentence embeddings using siamese bert-networks. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. [S.l.: s.n.], 2019. p. 3982–3992.

RIBEIRO, M. T.; SINGH, S.; GUESTRIN, C. "why should i trust you?" explaining the predictions of any classifier. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. [S.l.: s.n.], 2016. p. 1135–1144.

RIGOTTO, R. M.; VASCONCELOS, D. P.; ROCHA, M. M. Uso de agrotóxicos no brasil e problemas para a saúde pública. *Cadernos de Saúde Pública*, SciELO Public Health, v. 30, p. 1360–1362, 2014.

ROMERA-PAREDES, B.; TORR, P. An embarrassingly simple approach to zero-shot learning. In: PMLR. *International conference on machine learning*. [S.l.], 2015. p. 2152–2161.

ROSENBLATT, F. *The perceptron, a perceiving and recognizing automaton Project Para*. [S.l.]: Cornell Aeronautical Laboratory, 1957.

ROTHER, C.; KOLMOGOROV, V.; BLAKE, A. " grabcut" interactive foreground extraction using iterated graph cuts. *ACM transactions on graphics (TOG)*, ACM New York, NY, USA, v. 23, n. 3, p. 309–314, 2004.

RUDER, S. *An Overview of Multi-Task Learning in Deep Neural Networks*. arXiv, 2017. Disponível em: <https://arxiv.org/abs/1706.05098>.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *nature*, Nature Publishing Group, v. 323, n. 6088, p. 533–536, 1986.

RUSSAKOVSKY, O. et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, Springer, v. 115, n. 3, p. 211–252, 2015.

RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. 3rd. ed. USA: Prentice Hall Press, 2009. ISBN 0136042597.

SAHA, S. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way | by Sumit Saha | Towards Data Science*. 2018. <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>. (Accessed on 09/20/2022).

SAMEK, W.; WIEGAND, T.; MÜLLER, K.-R. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *arXiv preprint arXiv:1708.08296*, 2017.

SANTOS, T. T. et al. Grape detection, segmentation, and tracking using deep neural networks and three-dimensional association. *Computers and Electronics in Agriculture*, Elsevier, v. 170, p. 105247, 2020.

SARKAR, C. et al. Leaf disease detection using machine learning and deep learning: Review and challenges. *Applied Soft Computing*, Elsevier, v. 145, p. 110534, 2023. ISSN 1568-4946.

SELVARAJU, R. R. et al. Grad-cam: Visual explanations from deep networks via gradient-based localization. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2017. p. 618–626.

SHARMA, H.; PADHA, D.; BASHIR, N. D-kap: A deep learning-based kashmiri apple plant disease prediction framework. In: IEEE. *2022 Seventh International Conference on Parallel, Distributed and Grid Computing (PDGC)*. [S.l.], 2022. p. 576–581.

SIMONYAN, K.; VEDALDI, A.; ZISSERMAN, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

SINGH, D. et al. Plantdoc: A dataset for visual plant disease detection. In: *Proceedings of the 7th ACM IKDD CoDS and 25th COMAD*. New York, NY, USA: Association for Computing Machinery, 2020. (CoDS COMAD 2020), p. 249–253. ISBN 9781450377386. Disponível em: <https://doi.org/10.1145/3371158.3371196>.

SNELL, J.; SWERSKY, K.; ZEMEL, R. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, v. 30, 2017.

SOLOMONOFF, R. J. An inductive inference machine. In: INSTITUTE OF RADIO ENGINEERS NEW YORK. *IRE Convention Record, Section on Information Theory*. [S.l.], 1957. v. 2, p. 56–62.

SRIVASTAVA, N. et al. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, JMLR.org, v. 15, n. 1, p. 1929–1958, 2014.

STONE, M. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society: Series B (Methodological)*, Wiley Online Library, v. 36, n. 2, p. 111–147, 1974.

SUJATHA, R. et al. Performance of deep learning vs machine learning in plant leaf disease detection. *Microprocessors and Microsystems*, Elsevier, v. 80, p. 103615, 2021. ISSN 0141-9331.

SUTSKEVER, I.; VINYALS, O.; LE, Q. V. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, v. 27, 2014.

SZEGEDY, C. et al. Going deeper with convolutions. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2015. p. 1–9.

TAN, M.; LE, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In: PMLR. *International conference on machine learning*. [S.l.], 2019. p. 6105–6114.

TEAM, T. pandas development. *pandas-dev/pandas: Pandas*. Zenodo, 2020. Disponível em: <https://doi.org/10.5281/zenodo.3509134>.

THAPA, R. et al. The plant pathology challenge 2020 data set to classify foliar disease of apples. *Applications in plant sciences*, Wiley Online Library, v. 8, n. 9, p. e11390, 2020.

TIKHONOV, A. N.; ARSENIN, V. Y. *Solutions of ill-posed problems*. [S.l.]: Winston, 1977.

VALLABHAJOSYULA, S.; SISTLA, V.; KOLLI, V. K. K. A novel hierarchical framework for plant leaf disease detection using residual vision transformer. *Heliyon*, Elsevier, v. 10, n. 9, 2024.

VAPNIK, V. *The nature of statistical learning theory*. [S.l.]: Springer science & business media, 2013.

VASWANI, A. et al. Attention is all you need. *Advances in neural information processing systems*, v. 30, 2017.

VINYALS, O. et al. Matching networks for one shot learning. *Advances in neural information processing systems*, v. 29, 2016.

WANG, G.; SUN, Y.; WANG, J. Automatic image-based plant disease severity estimation using deep learning. *Computational intelligence and neuroscience*, Wiley Online Library, v. 2017, n. 1, p. 2917536, 2017.

WEBER, R.; SCHEK, H.-J.; BLOTT, S. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. v. 98, p. 194–205, 1998.

WU, X.; SAHOO, D.; HOI, S. C. Recent advances in deep learning for object detection. *Neurocomputing*, Elsevier, v. 396, p. 39–64, 2020.

XIE, X. et al. A deep-learning-based real-time detector for grape leaf diseases using improved convolutional neural networks. *Frontiers in plant science*, Frontiers Media SA, v. 11, p. 751, 2020.

ZHANG, Y.; YANG, Q. A survey on multi-task learning. *IEEE transactions on knowledge and data engineering*, IEEE, v. 34, n. 12, p. 5586–5609, 2021.

ZHAO, X. et al. Fast segment anything. *arXiv preprint arXiv:2306.12156*, 2023.

APÊNDICE A – DATASET INFORMATION

This appendix provides comprehensive information about the dataset used in this study, including detailed statistics on plant diseases, pathogens, symptoms, and their distributions.

A.1 DATASET OVERVIEW

The dataset contains 58,305 records with 45 columns, representing a comprehensive collection of plant disease observations across multiple species. The dataset includes 38 one-hot encoded symptom features, covering 15 plant species, 32 disease types, and 8 pathogen categories.

A.1.1 Health Status Distribution

- Sick plants: 42,721 records (73.3%)
- Healthy plants: 15,584 records (26.7%)

A.2 DISEASE DISTRIBUTION

The dataset encompasses 32 distinct diseases, with the following top 10 most prevalent:

Table 4 – Top 10 Most Prevalent Diseases in the Dataset

Disease	Count	Percentage
No disease available	15,084	25.9%
Huanglongbing	5,507	9.4%
Tomato yellow leaf curl virus	5,357	9.2%
Xanthomonas euvesicatoria	3,124	5.4%
Phytophthora infestans	2,909	5.0%
Xanthomonas arboricola pv. pruni	2,297	3.9%
Alternaria solani	2,000	3.4%
Erysiphe cichoracearum	1,835	3.1%
Septoria lycopersici	1,771	3.0%
Tetranychus urticae	1,676	2.9%

A.3 PLANT SPECIES DISTRIBUTION

The dataset includes 15 plant species, with the following distribution:

Table 5 – Plant Species Distribution in the Dataset

Plant Species	Count	Percentage
Tomato	18,160	31.1%
Orange	5,507	9.4%
Soybean	5,090	8.7%
Grape	4,062	7.0%
Mango	4,000	6.9%
Corn (maize)	3,852	6.6%
Apple	3,171	5.4%
Peach	2,657	4.6%
Pepper, bell	2,475	4.2%
Potato	2,152	3.7%
Cherry	1,906	3.3%
Squash	1,835	3.1%
Strawberry	1,565	2.7%
Blueberry	1,502	2.6%
Raspberry	371	0.6%

A.4 SYMPTOM ANALYSIS

The dataset contains 38 distinct symptoms with a total of 122,111 symptom instances across all records. The symptom statistics are as follows:

A.4.1 Symptom Distribution Statistics

- Records with at least one symptom: 42,721 (73.3%)
- Records with no symptoms: 15,584 (26.7%)
- Mean symptoms per record: 2.09
- Median symptoms per record: 2.00
- Range: 0-4 symptoms per record

A.4.2 Top 15 Most Common Symptoms

Table 6 – Most Frequent Symptoms in the Dataset

Symptom	Count	Percentage
Yellow halos	12,762	21.9%
Necrosis	9,808	16.8%
Water-soaked spots	8,330	14.3%
Leaf drop	7,593	13.0%
Leaf curling	6,655	11.4%
Leaf browning tips	6,409	11.0%
Circular black spots	5,699	9.8%
Leaf spots	5,553	9.5%
Asymmetrical yellowing	5,507	9.4%
Mottling	5,507	9.4%
Purple veins	5,357	9.2%
Small leaves	5,357	9.2%
Concentric rings	4,480	7.7%
White sporulation	3,887	6.7%
Veinal necrosis	3,256	5.6%

A.5 PATHOGEN ANALYSIS

The dataset includes 8 distinct pathogen types with the following distribution:

Table 7 – Pathogen Type Distribution

Pathogen Type	Count	Percentage
Fungi	19,978	34.3%
0 (No pathogen)	15,084	25.9%
Bacteria	11,428	19.6%
Virus	5,730	9.8%
Oomycete	2,909	5.0%
Arachnid pest	1,676	2.9%
Insect Pest	1,000	1.7%
None	500	0.9%

A.6 CROSS-ANALYSIS FINDINGS

A.6.1 Top Plant-Pathogen Combinations

The most significant plant-pathogen combinations in the dataset are:

- Virus on tomato: 5,730 records (9.8%)
- Bacteria on orange: 5,507 records (9.4%)
- Fungi on tomato: 5,127 records (8.8%)
- No pathogen on soybean: 5,090 records (8.7%)
- Fungi on grape: 3,639 records (6.2%)

A.6.2 Disease-Specific Symptom Patterns

Notable disease-symptom associations include:

- **Huanglongbing** (5,507 records): 100% association with asymmetrical yellowing, leaf curling, mottling, and necrosis
- **Tomato yellow leaf curl virus** (5,357 records): 100% association with leaf browning tips, purple veins, and small leaves
- **Xanthomonas euvesicatoria** (3,124 records): 100% association with water-soaked spots, with 68.1% showing leaf drop, circular black spots, and yellow halos
- **Tetranychus urticae** (1,676 records): 100% association with bronzing and fine webbing

APÊNDICE B – MODEL METRICS

This appendix provides detailed performance metrics for the multi-task learning models implemented in this study. All reported metrics represent averaged results from 5-fold stratified cross-validation, ensuring robust and generalizable performance estimates across different data splits. The metrics are organized by task type and model architecture, providing a comprehensive overview of the system's performance across different diagnostic tasks and architectural configurations. The figures presented here supplement the quantitative analysis provided in Chapter 6, offering detailed visualization of training dynamics, convergence behavior, and classification performance across all hierarchical tasks.

B.1 BINARY HEALTH CLASSIFICATION

The binary health classification task, which forms the foundation of the diagnostic hierarchy, achieved an accuracy of 98.72% with a test loss of 0.0357, representing a substantial improvement over the baseline approach. The precision reached 97.14%, while recall attained 98.16%, and the F1-score achieved 97.65%. The area under the ROC curve (AU-ROC) reached 0.9991, and the area under the precision-recall curve (AU-PRC) achieved 0.9975, demonstrating excellent discrimination capability and balance between precision and recall, crucial for diagnostic applications where both false positives and false negatives carry significant consequences.

Table 8 presents the comprehensive performance metrics for the binary health classification task, averaged across all 5 cross-validation folds.

Table 8 – Binary Health Classification Performance Metrics (5-Fold CV Average)

Metric	**Value**
Test Accuracy	98.72%
Test Loss	0.0357
Precision	97.14%
Recall	98.16%
F1-Score	97.65%
AU-ROC	0.9991
AU-PRC	0.9975

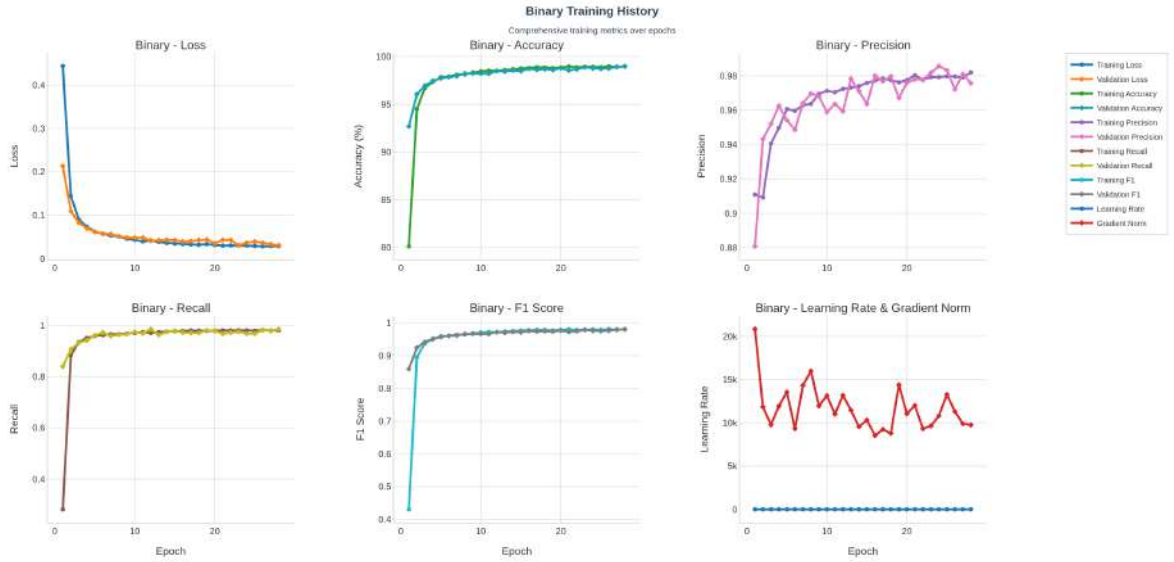


Figure 50 – Training and validation performance curves for the binary disease classification task, averaged across 5-fold cross-validation. The figure shows the convergence behavior of both accuracy and loss metrics over the training epochs, demonstrating stable learning dynamics with minimal overfitting. The smooth convergence pattern indicates effective regularization and optimal hyperparameter selection for this foundational task.

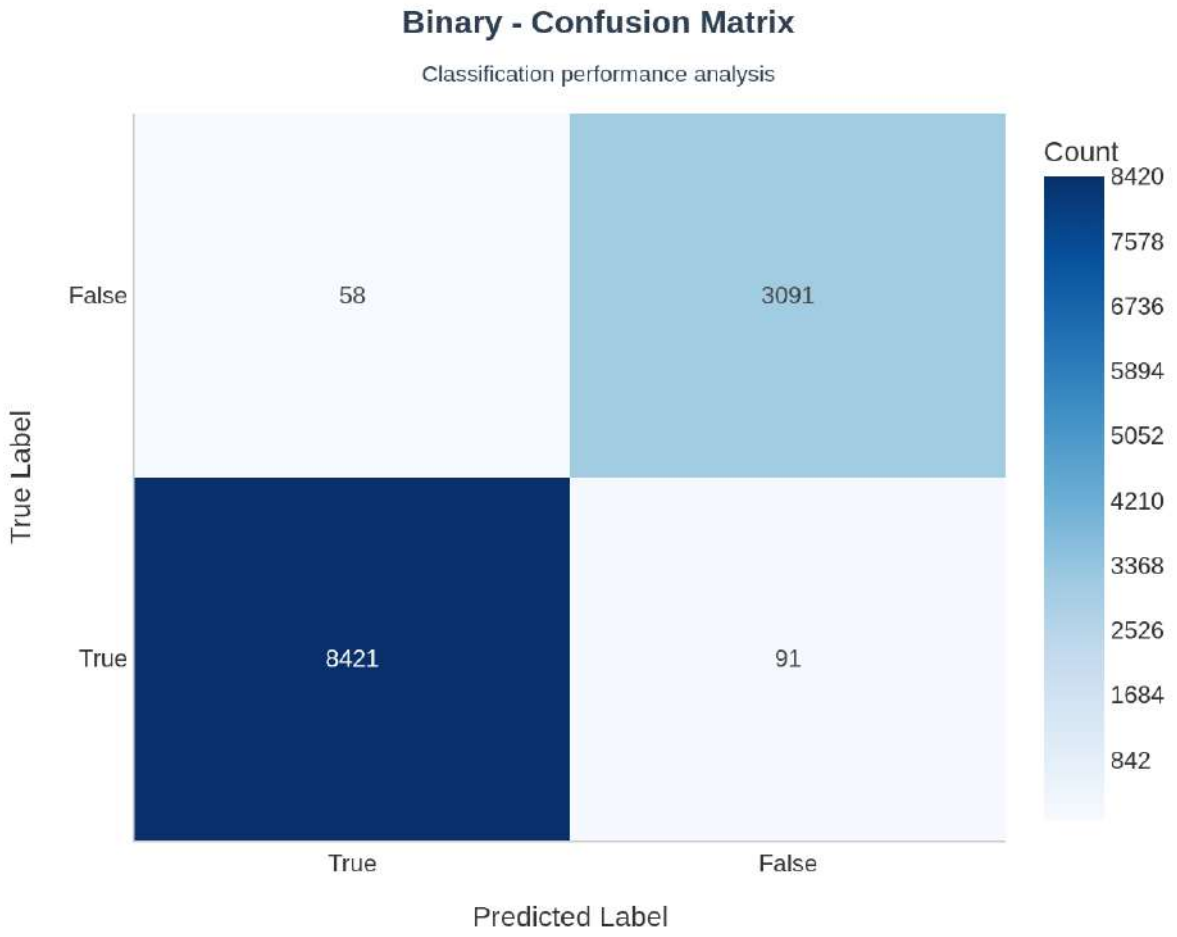


Figure 51 – Confusion matrix for binary disease classification task averaged across 5-fold cross-validation. The matrix demonstrates the system's exceptional capability to distinguish between healthy and diseased plant tissue, with minimal misclassification errors. The high true positive and true negative rates validate the reliability of this foundational diagnostic component.

B.2 LEAF DETECTION PERFORMANCE

The leaf detection task serves as a crucial preprocessing filter, ensuring that only valid plant material is processed through the diagnostic pipeline. This task achieved perfect accuracy of 100% in the controlled evaluation environment, though real-world deployment revealed sensitivity to image quality variations as discussed in Chapter 6.

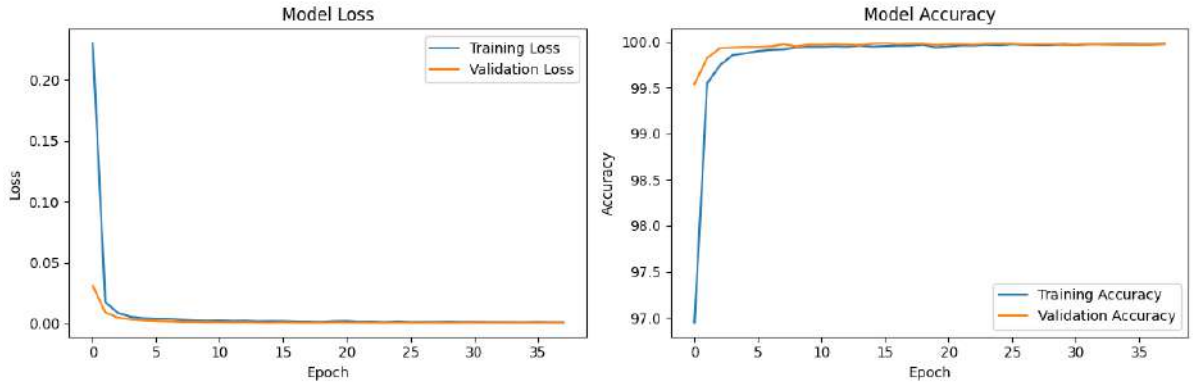


Figure 52 – Training history for leaf detection task. The rapid convergence to perfect accuracy demonstrates the relative simplicity of this binary classification problem when applied to segmented image regions. The minimal validation loss and stable training progression indicate effective feature learning for distinguishing plant material from background elements.

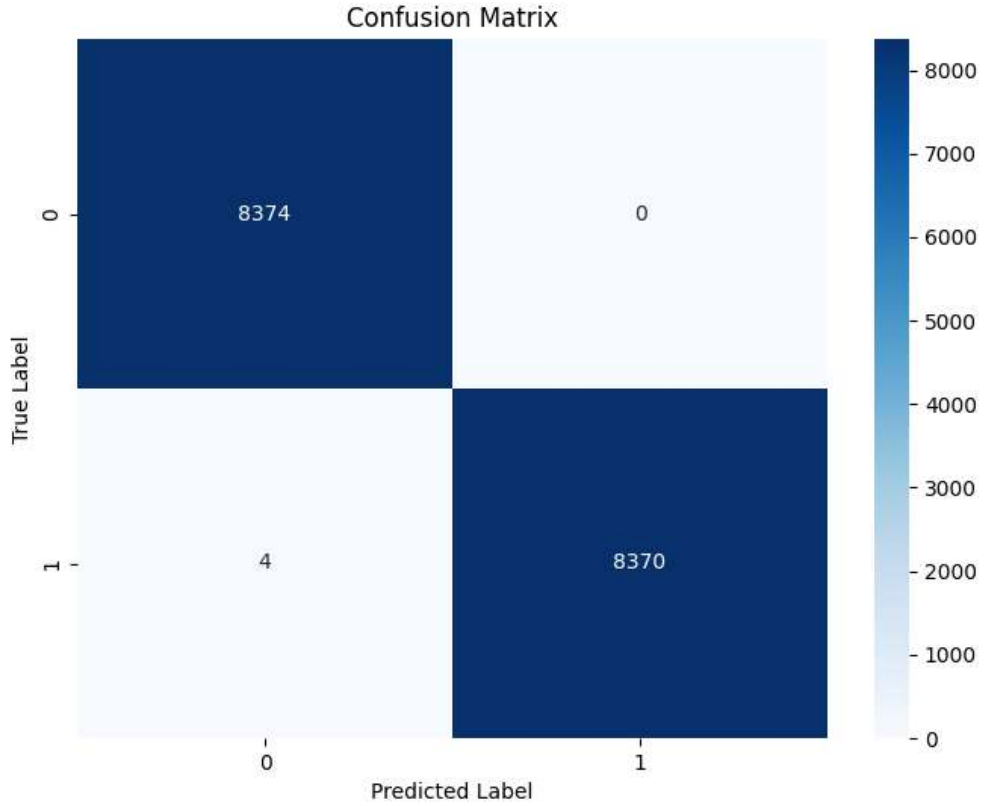


Figure 53 – Confusion matrix for leaf detection task showing perfect classification performance on the test set. While this matrix demonstrates flawless performance in controlled conditions, the real-world deployment challenges discussed in Chapter 6 highlight the importance of comprehensive field testing for validation of preprocessing components.

B.3 PLANT SPECIES CLASSIFICATION

Plant species identification achieved an accuracy of 98.95% with a test loss of 0.0324, demonstrating exceptional capability in botanical classification across 15 plant species. The model achieved macro-averaged precision of 98.91%, recall of 98.40%, and F1-score of 98.65%, with an AU-ROC of 0.9999. This performance reflects the successful integration of health status information through the hierarchical dependency structure, where disease symptoms provide additional diagnostic context for species identification.

Table 9 presents the overall performance metrics, while Table 10 provides detailed per-species performance metrics.

Table 9 – Plant Species Classification Overall Performance (5-Fold CV Average)

Metric	**Value**
Test Accuracy	98.95%
Test Loss	0.0324
Macro Precision	98.91%
Macro Recall	98.40%
Macro F1-Score	98.65%
AU-ROC	0.9999

Table 10 – Plant Species Classification Per-Class Performance (5-Fold CV Average)

Plant Species	**Precision**	**Recall**	**F1-Score**
Corn (Maize)	100.00%	99.61%	99.81%
Grape	99.41%	99.41%	99.41%
Tomato	98.68%	99.53%	99.10%
Orange	99.63%	99.81%	99.72%
Apple	98.84%	96.60%	97.70%
Peach	99.23%	99.04%	99.13%
Pepper (Bell)	98.12%	97.10%	97.61%
Blueberry	97.53%	99.06%	98.29%
Potato	96.95%	95.60%	96.27%
Cherry	98.38%	96.55%	97.46%
Raspberry	100.00%	96.20%	98.06%
Soybean	99.01%	98.82%	98.91%
Squash	99.72%	99.72%	99.72%
Strawberry	98.53%	99.12%	98.82%
Mango	99.63%	99.76%	99.70%

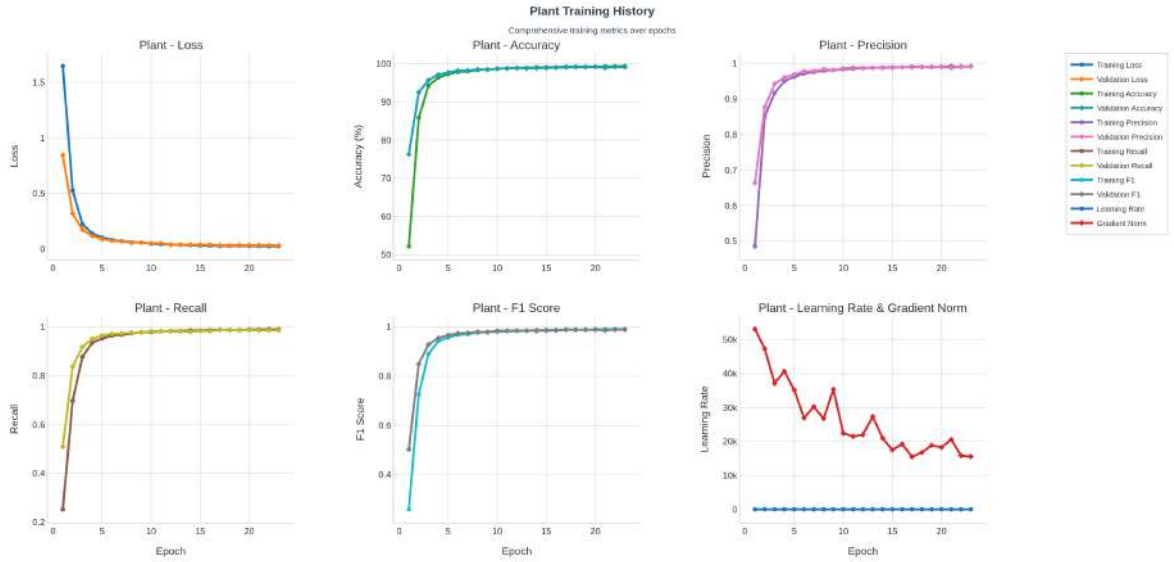


Figure 54 – Training history for plant species classification, averaged across 5-fold cross-validation. The smooth convergence pattern with minimal overfitting demonstrates the effectiveness of the hierarchical architecture in leveraging shared representations from the binary disease detection task. The stable validation performance indicates robust feature learning across diverse plant species.

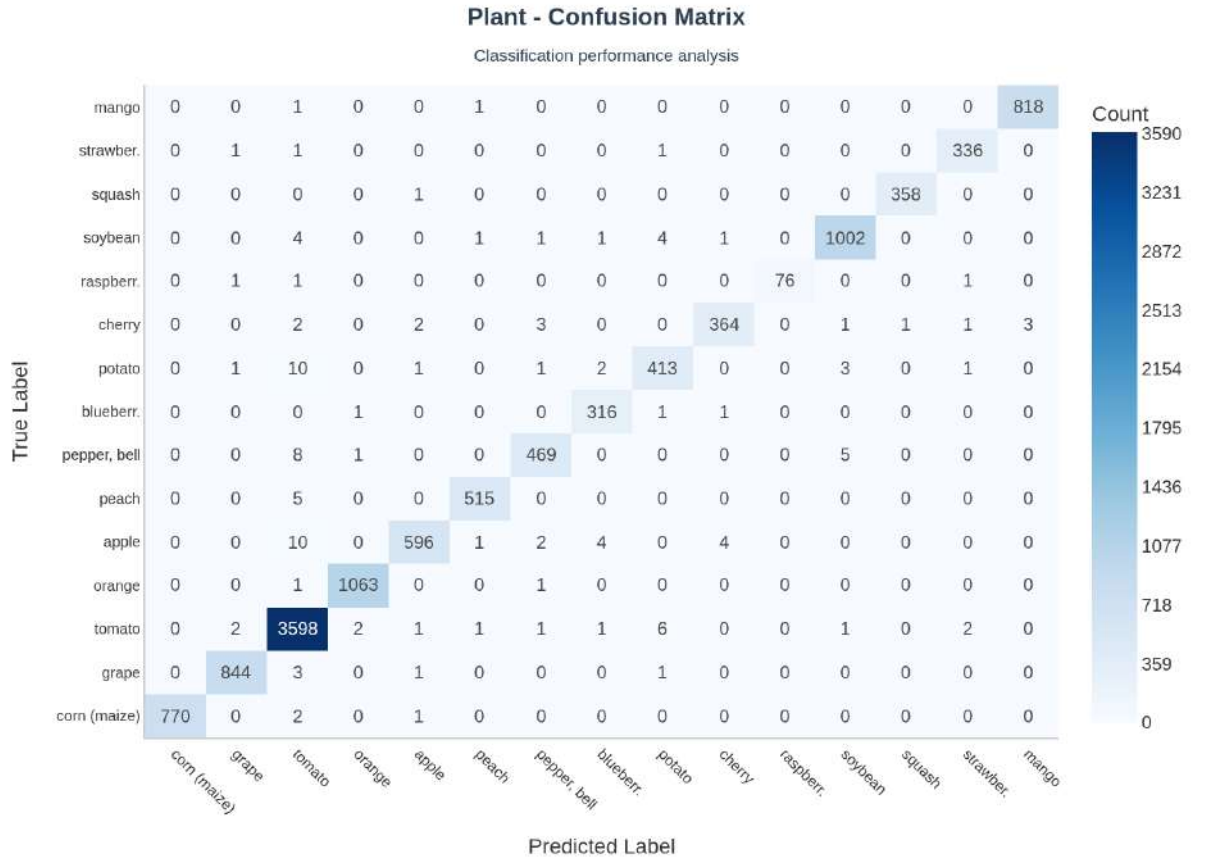


Figure 55 – Confusion matrix for plant species classification showing detailed performance across all plant categories in the dataset, averaged across 5-fold cross-validation. The matrix reveals the system's capability to maintain high accuracy across taxonomically diverse species, with minimal inter-species confusion even in the presence of disease-induced morphological changes.

B.4 PATHOGEN TYPE CLASSIFICATION

Pathogen type classification achieved an accuracy of 96.64% with a test loss of 0.0916, successfully distinguishing between fungal, bacterial, viral, oomycete, arachnid pest, and insect pest causal agents based on symptom manifestations. The model achieved macro-averaged precision of 94.00%, recall of 96.76%, and F1-score of 95.29%, with an AU-ROC of 0.9987. This performance demonstrates the system's capability to implement biological principles where pathogen-host relationships constrain possible causal agents.

Table 11 presents the overall performance metrics, while Table 12 provides detailed per-pathogen type performance metrics.

Table 11 – Pathogen Type Classification Overall Performance (5-Fold CV Average)

Metric	Value
Test Accuracy	96.64%
Test Loss	0.0916
Macro Precision	94.00%
Macro Recall	96.76%
Macro F1-Score	95.29%
AU-ROC	0.9987

Table 12 – Pathogen Type Classification Per-Class Performance (5-Fold CV Average)

Pathogen Type	Precision	Recall	F1-Score
Fungi	98.35%	95.63%	96.97%
Bacteria	98.24%	98.07%	98.16%
Oomycete	84.17%	94.71%	89.13%
Arachnid Pest	85.38%	94.60%	89.76%
Virus	97.86%	98.62%	98.24%
Insect Pest	100.00%	98.93%	99.46%

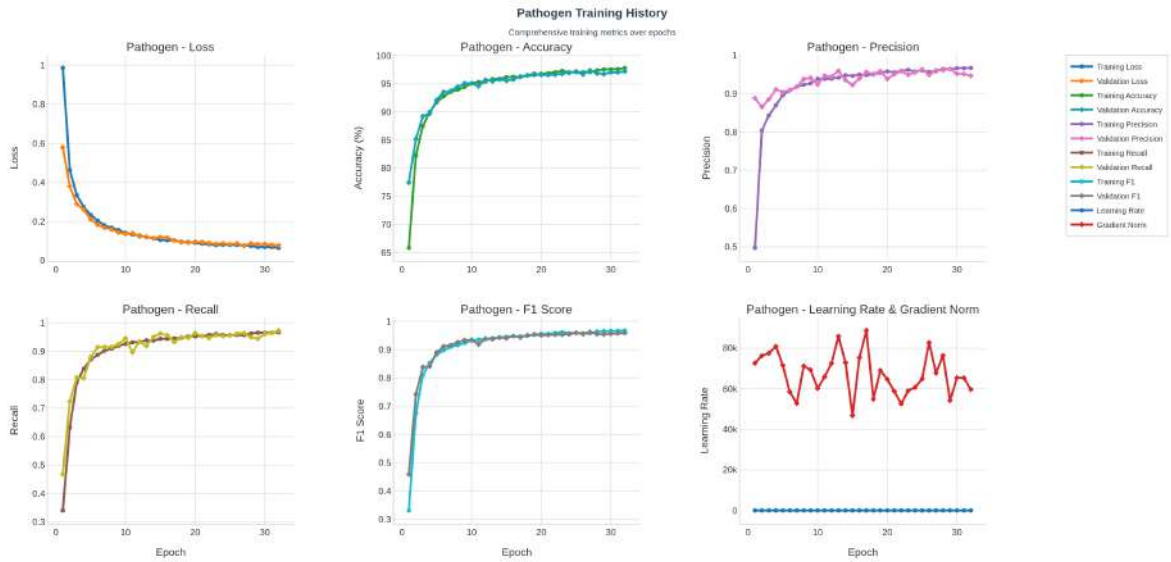


Figure 56 – Training history for pathogen type classification showing convergence behavior across fungal, bacterial, viral, oomycete, and pest categories, averaged across 5-fold cross-validation. The training curves demonstrate effective learning of pathogen-specific symptom patterns, with stable validation performance indicating good generalization capabilities. The slightly higher loss compared to other tasks reflects the inherent complexity of pathogen type determination from visual symptoms alone.

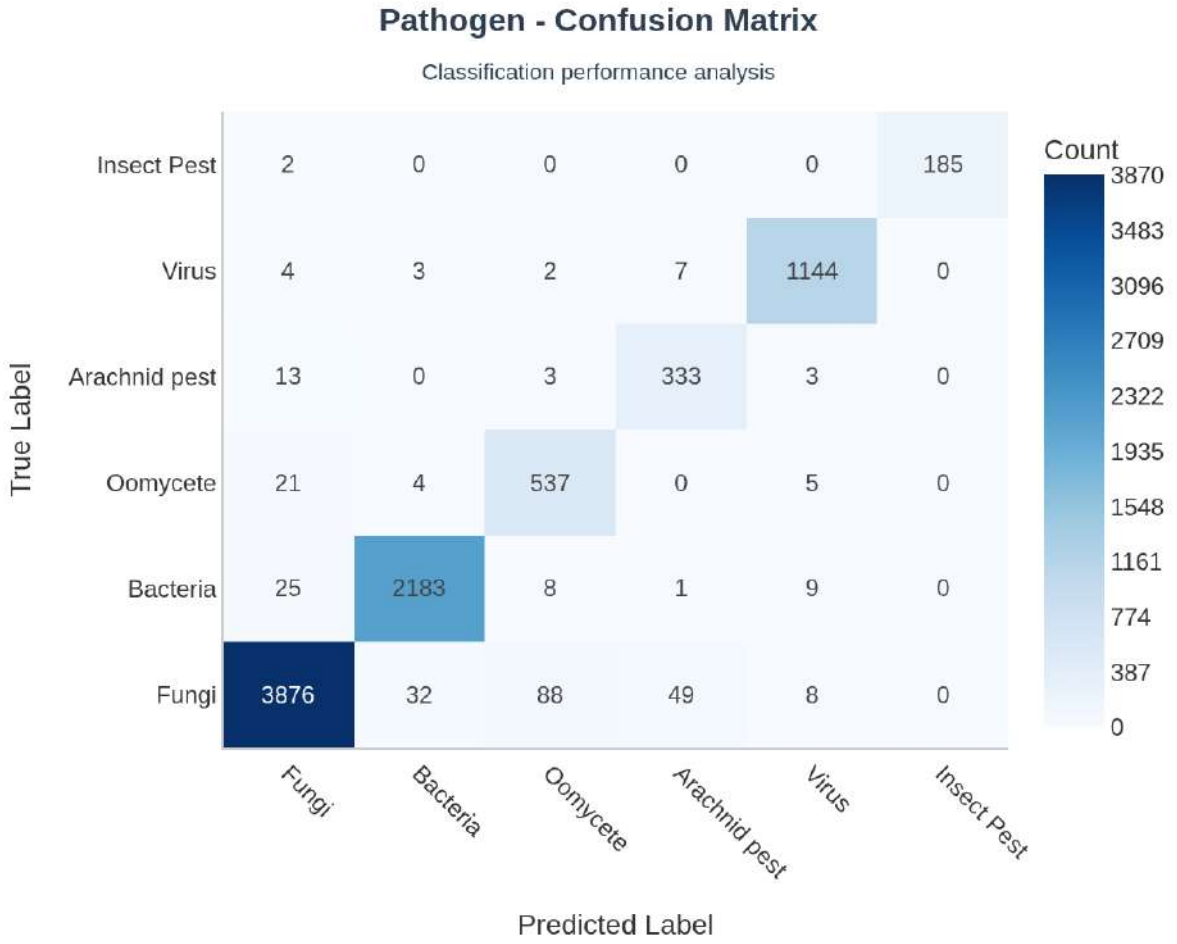


Figure 57 – Confusion matrix for pathogen type classification illustrating the system's performance in distinguishing between fungal, bacterial, viral, oomycete, and pest pathogens, averaged across 5-fold cross-validation. The matrix shows strong diagonal performance with minimal cross-pathogen confusion, validating the model's ability to learn pathogen-specific symptom signatures. The highest accuracy is achieved for insect pest and fungal pathogen detection, consistent with the prevalence and distinctive visual characteristics of these diseases in the dataset.

B.5 DISEASE NAME CLASSIFICATION

Disease name classification, representing the most challenging fine-grained diagnostic task, achieved an accuracy of 96.01% with a test loss of 0.1169. The model achieved macro-averaged precision of 95.72%, recall of 95.61%, and F1-score of 95.63%, with an AU-ROC of 0.9994. This exceptional performance in specific disease identification across 30 distinct disease classes demonstrates the effectiveness of the progressive complexity architecture implemented in the task-specific heads.

Table 13 presents the overall performance metrics for disease name classification.

Table 13 – Disease Name Classification Overall Performance (5-Fold CV Average)

Metric	Value
Test Accuracy	96.01%
Test Loss	0.1169
Macro Precision	95.72%
Macro Recall	95.61%
Macro F1-Score	95.63%
AU-ROC	0.9994

Due to the large number of disease classes (30), detailed per-disease performance metrics are available in the supplementary materials. The top-performing diseases include Huanglongbing (F1: 99.72%), Erysiphe cichoracearum (F1: 99.74%), and several others with F1-scores above 99%. The most challenging diseases to classify include Cercospora zeae-maydis (F1: 81.18%) and Corynespora cassiicola (F1: 85.71%), which exhibit high visual similarity to other diseases affecting the same host plants.

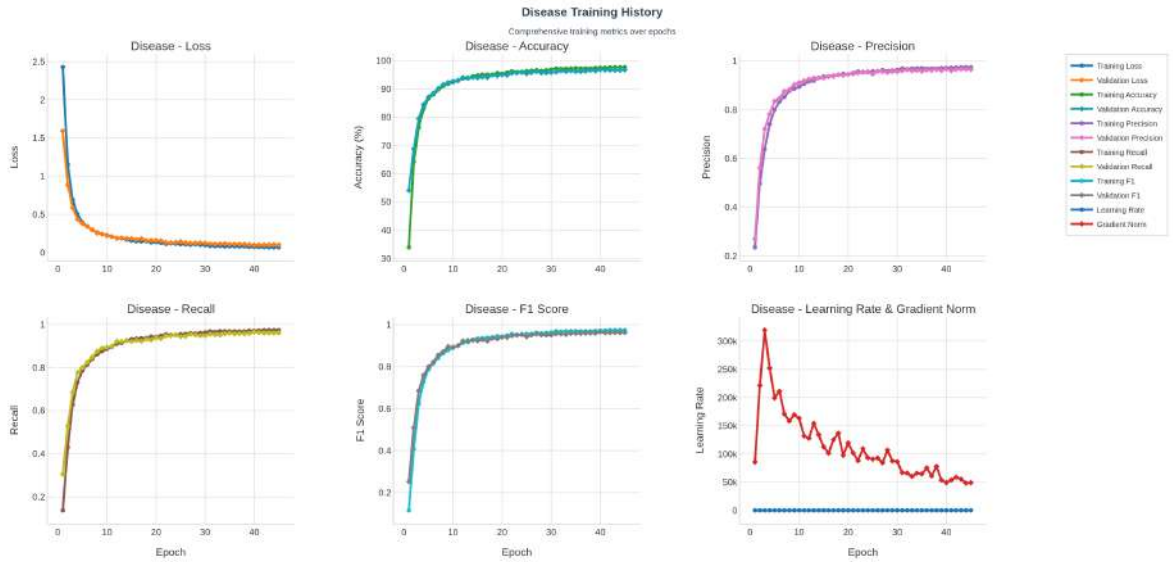


Figure 58 – Training history for disease name classification, the most complex diagnostic component requiring fine-grained pathogen identification, averaged across 5-fold cross-validation. The training curves show successful convergence despite the inherent difficulty of the task, with validation performance closely tracking training accuracy. The stable learning dynamics demonstrate the effectiveness of the hierarchical architecture in providing informative features for specific disease diagnosis.

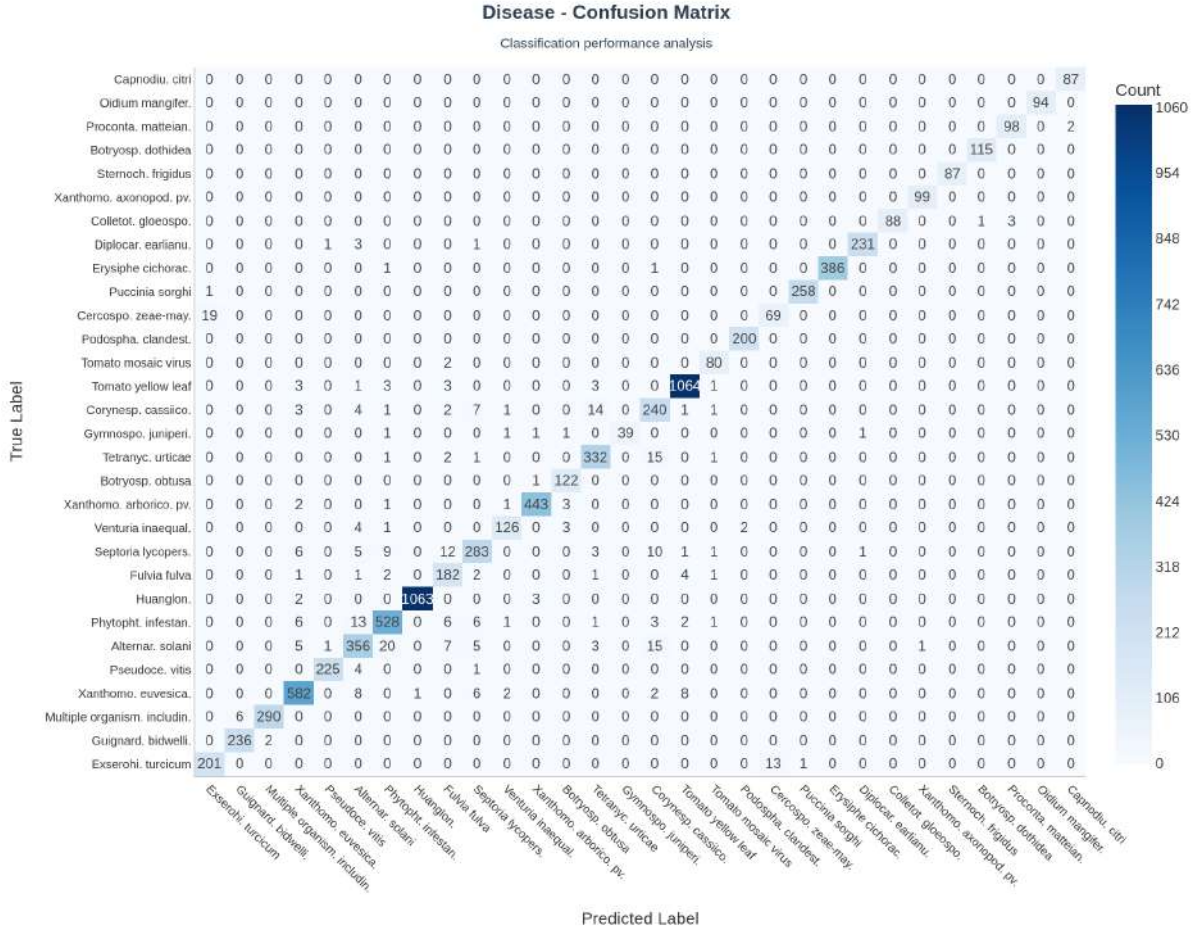


Figure 59 – Confusion matrix for disease name classification showing detailed performance across all 30 disease categories, averaged across 5-fold cross-validation. The matrix reveals strong diagonal performance with minimal inter-disease confusion, validating the model's capability to learn disease-specific visual signatures even among closely related pathologies.

B.6 MULTI-LABEL SYMPTOM DETECTION

The multi-label symptom detection task achieved an overall accuracy of 99.52% with a test loss of 0.0135, demonstrating exceptional performance in identifying multiple simultaneous symptom manifestations across 38 distinct symptom categories. The model achieved macro-averaged precision of 96.06%, recall of 94.17%, and F1-score of 95.01%, with an AU-ROC of 0.9992. This comprehensive approach addresses the complexity of real-world disease presentations where multiple symptoms occur simultaneously.

Table 14 presents the overall performance metrics for symptom detection.

Table 14 – Multi-Label Symptom Detection Overall Performance (5-Fold CV Average)

Metric	Value
Test Accuracy	99.52%
Test Loss	0.0135
Macro Precision	96.06%
Macro Recall	94.17%
Macro F1-Score	95.01%
AU-ROC	0.9992

The symptom detection system demonstrates robust performance across diverse symptom types. Top-performing symptoms include irregular holes (F1: 99.42%), purple spots (F1: 100%), and asymmetrical yellowing (F1: 99.81%). The most challenging symptoms were rectangular spots and tan spots (both F1: 75.64%), likely due to class imbalance and visual similarity to other symptom types. Detailed per-symptom performance metrics are available in the supplementary documentation.

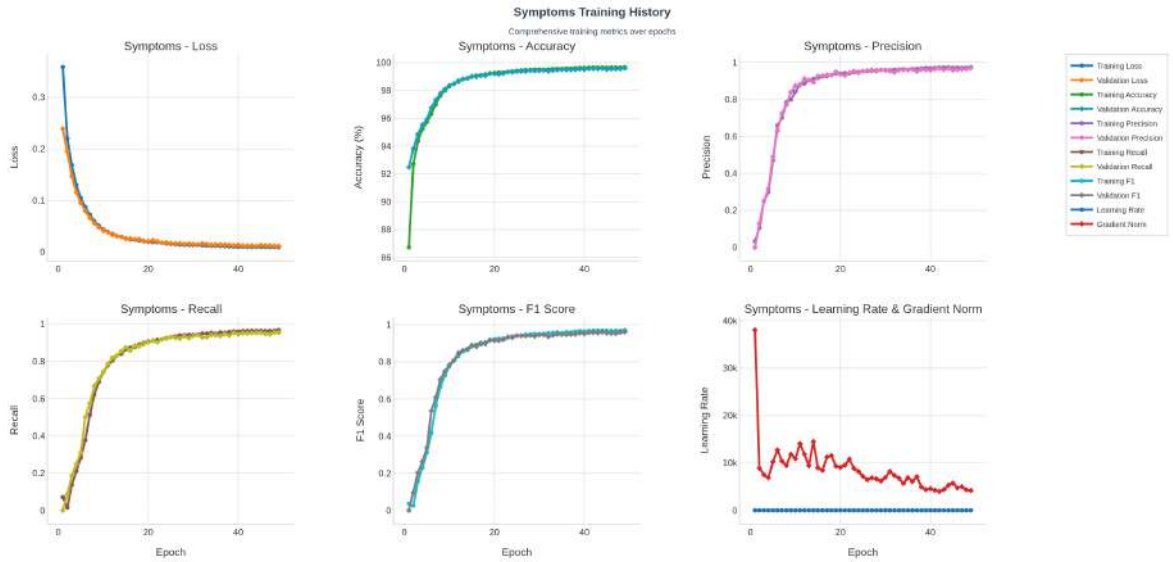


Figure 60 – Training history for multi-label symptom detection showing convergence behavior across all 38 symptom categories, averaged across 5-fold cross-validation. The training curves demonstrate the system's capability to learn complex symptom patterns simultaneously, with validation performance indicating robust generalization across diverse symptom manifestations. The low final loss values reflect the high accuracy achieved in identifying multiple concurrent symptoms, validating the multi-label architecture design.

APÊNDICE C – ZERO-SHOT AND FEW-SHOT LEARNING

This appendix provides detailed background on zero-shot and few-shot learning paradigms. While these approaches were not directly employed in this thesis, they represent important contemporary developments in machine learning that enable models to generalize to unseen classes with minimal or no training examples.

C.1 ZERO-SHOT LEARNING

Zero-shot learning represents a paradigm shift in machine learning where models are expected to recognize or classify instances from classes that were never seen during training (PALATUCCI et al., 2009). This capability is particularly valuable in scenarios where collecting labeled data for all possible classes is impractical or impossible.

C.1.1 Problem Formulation

In traditional supervised learning, the training set $\mathcal{D}_{\text{train}}$ and test set $\mathcal{D}_{\text{test}}$ share the same label space $\mathcal{Y}$. In contrast, zero-shot learning operates under a different assumption: the label spaces are disjoint, i.e., $\mathcal{Y}_{\text{train}} \cap \mathcal{Y}_{\text{test}} = \emptyset$. The model must generalize to completely unseen classes during inference.

This generalization is made possible through auxiliary information that describes both seen and unseen classes. This auxiliary information typically takes the form of:

- **Semantic Attributes:** Hand-crafted descriptions of classes (e.g., "has stripes," "is carnivorous")
- **Word Embeddings:** Dense vector representations of class names derived from large text corpora (MIKOLOV et al., 2013)
- **Knowledge Graphs:** Structured relationships between classes
- **Natural Language Descriptions:** Textual descriptions of class characteristics

C.1.2 Methodology

A common approach to zero-shot learning involves learning a compatibility function $f : \mathcal{X} \times \mathcal{S} \rightarrow \mathbb{R}$ that measures how well an input $x \in \mathcal{X}$ matches a semantic description $s \in \mathcal{S}$. During inference, given an input x and candidate classes $\{c_1, c_2, ..., c_K\}$ with corresponding semantic descriptions $\{s_1, s_2, ..., s_K\}$, the predicted class is:

$$\hat{y} = \arg\max_k f(x, s_k) \tag{C.1}$$

Modern approaches leverage embedding spaces where both visual features and semantic descriptions are projected into a common space, enabling direct comparison (ROMERA-PAREDES; TORR, 2015).

C.2 FEW-SHOT LEARNING

Few-shot learning extends the zero-shot concept by allowing the model to learn from a very small number of labeled examples per class—typically 1 to 5 examples (VINYALS et al., 2016). The N-way K-shot learning problem involves classifying instances into N classes with only K labeled examples per class available during a meta-learning episode.

C.2.1 Meta-Learning Approaches

Meta-learning or "learning to learn" approaches have shown remarkable success in few-shot scenarios by learning initialization strategies or metric spaces that enable rapid adaptation to new tasks with minimal data.

Model-Agnostic Meta-Learning (MAML): MAML (FINN; ABBEEL; LEVINE, 2017) learns an initialization for model parameters that can be quickly adapted to new tasks with only a few gradient steps. The meta-objective is:

$$\min_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}) \tag{C.2}$$

where $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ represents the adapted parameters after one or more gradient steps on task $\mathcal{T}_i$.

Prototypical Networks: Prototypical Networks (SNELL; SWERSKY; ZEMEL, 2017) learn a metric space in which classification is performed by computing distances to prototype representations of each class. For a few-shot episode with support set S, the prototype for class c is:

$$\mathbf{c}_c = \frac{1}{|S_c|} \sum_{(\mathbf{x}_i, y_i) \in S_c} f_\phi(\mathbf{x}_i) \tag{C.3}$$

where f_ϕ is the embedding function. Classification uses softmax over distances to prototypes.

Matching Networks: Matching Networks (VINYALS et al., 2016) use attention mechanisms to weight support set examples when classifying query examples, enabling flexible adaptation to new classes.

C.2.2 Applications in Vision and Language Models

Recent large-scale vision-language models, such as CLIP (Contrastive Language-Image Pre-training) (RADFORD et al., 2021), have demonstrated impressive zero-shot capabilities by learning joint embeddings of images and text from massive datasets. These models can classify images into arbitrary categories specified by natural language prompts without any task-specific fine-tuning, representing a significant advancement in the generalization capabilities of deep learning systems.

CLIP's architecture consists of:

- An image encoder (Vision Transformer or ResNet)
- A text encoder (Transformer)
- Contrastive learning objective that aligns image and text representations

The model is trained on 400 million image-text pairs, learning to predict which text snippet corresponds to which image. At test time, this enables zero-shot classification by comparing the image embedding to text embeddings of class names or descriptions.

C.3 CHALLENGES AND FUTURE DIRECTIONS

Despite significant progress, zero-shot and few-shot learning face several challenges:

- **Domain Gap:** The semantic space learned on seen classes may not generalize well to unseen classes with different characteristics
- **Hubness Problem:** In high-dimensional spaces, certain points become hubs that are nearest neighbors to many other points, degrading retrieval quality
- **Semantic Ambiguity:** Class descriptions may not uniquely identify visual characteristics
- **Transductive vs. Inductive:** Some methods perform well in transductive settings (where unseen test data is available during training) but poorly in fully inductive scenarios

Future research directions include developing better semantic representations, leveraging large-scale pre-training more effectively, and combining few-shot learning with active learning to efficiently collect the most informative examples.

APÊNDICE D – ENCODER-DECODER ARCHITECTURES IN DETAIL

This appendix provides detailed background on encoder-decoder architectures and attention mechanisms. While the primary architecture used in this thesis employs CNNs with hard parameter sharing for multi-task learning, encoder-decoder architectures represent an important foundational concept that influenced modern deep learning architectures, including Transformers.

D.1 ENCODER-DECODER ARCHITECTURES

The encoder-decoder architecture represents a fundamental paradigm in deep learning for processing and transforming sequential data, first gaining prominence in the context of sequence-to-sequence (seq2seq) learning (SUTSKEVER; VINYALS; LE, 2014). This architectural pattern has become foundational for tasks requiring the transformation of one sequence into another, such as machine translation, text summarization, image captioning, and speech recognition.

D.1.1 Architectural Overview and Motivation

The core principle of encoder-decoder architectures is to decompose the sequence transformation task into two distinct phases: encoding and decoding. The **encoder** processes the input sequence and compresses it into a fixed-length context representation (often called the context vector or thought vector), which captures the semantic content of the input. The **decoder** then generates the output sequence conditioned on this context representation.

Formally, given an input sequence $\mathbf{x} = (x_1, x_2, ..., x_T)$, the encoder computes a context representation:

$$\mathbf{c} = f_{\text{enc}}(\mathbf{x}) \tag{D.1}$$

The decoder then generates the output sequence $\mathbf{y} = (y_1, y_2, ..., y_{T'})$ by modeling the conditional probability:

$$p(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{T'} p(y_t|y_1, ..., y_{t-1}, \mathbf{c}) \tag{D.2}$$

D.1.2 Historical Development

Early encoder-decoder models utilized Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) (HOCHREITER; SCHMIDHUBER, 1997) and Gated Recurrent Unit (GRU) (CHO et al., 2014) networks. In these architectures, the encoder processes the input sequence sequentially, updating its hidden state at each timestep, with the final hidden state serving as the context vector. The decoder, also an RNN, generates the output sequence autoregressively, using the context vector to initialize its hidden state.

D.1.2.1 LSTM and GRU Cells

LSTM Architecture: The LSTM cell addresses the vanishing gradient problem in standard RNNs through a gating mechanism:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad \text{(forget gate)} \tag{D.3}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad \text{(input gate)} \tag{D.4}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad \text{(output gate)} \tag{D.5}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad \text{(candidate)} \tag{D.6}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \quad \text{(cell state)} \tag{D.7}$$

$$h_t = o_t \odot \tanh(C_t) \quad \text{(hidden state)} \tag{D.8}$$

GRU Architecture: The GRU simplifies the LSTM by combining the forget and input gates:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \quad \text{(update gate)} \tag{D.9}$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \quad \text{(reset gate)} \tag{D.10}$$

$$\tilde{h}_t = \tanh(W \cdot [r_t \odot h_{t-1}, x_t]) \quad \text{(candidate)} \tag{D.11}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{D.12}$$

D.1.3 Information Bottleneck Problem

A significant limitation of the basic encoder-decoder framework was the information bottleneck created by compressing the entire input sequence into a single fixed-length vector. This became particularly problematic for long sequences, as the context vector struggled to capture all relevant information. Performance degraded significantly as sequence length increased, with distant information being poorly represented in the fixed-dimensional context vector.

D.2 ATTENTION MECHANISM

The introduction of the **attention mechanism** by Bahdanau, Cho e Bengio (2014) addressed the information bottleneck limitation by allowing the decoder to selectively focus on different parts of the input sequence at each decoding step, rather than relying solely on a single context vector.

D.2.1 Bahdanau Attention (Additive Attention)

The attention mechanism computes a weighted combination of encoder hidden states at each decoding step. For each decoder timestep t, attention weights $\alpha_{t,i}$ are computed to determine the relevance of each encoder hidden state $\mathbf{h}_i$:

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{j=1}^{T} \exp(e_{t,j})} \tag{D.13}$$

where $e_{t,i} = \text{score}(\mathbf{s}_{t-1}, \mathbf{h}_i)$ is an alignment score between the decoder state $\mathbf{s}_{t-1}$ and encoder state $\mathbf{h}_i$. In Bahdanau attention, the score function is computed as:

$$e_{t,i} = \mathbf{v}_a^T \tanh(\mathbf{W}_a \mathbf{s}_{t-1} + \mathbf{U}_a \mathbf{h}_i) \tag{D.14}$$

The context vector for timestep t is then:

$$\mathbf{c}_t = \sum_{i=1}^{T} \alpha_{t,i} \mathbf{h}_i \tag{D.15}$$

This mechanism allows the model to create a dynamic context representation that adapts to the current decoding position, significantly improving performance on long sequences and enabling better interpretability through attention weight visualization.

D.2.2 Luong Attention (Multiplicative Attention)

Luong et al. proposed alternative attention mechanisms that differ in how the alignment score is computed:

Dot Product:

$$\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_s) = \mathbf{h}_t^T \bar{\mathbf{h}}_s \tag{D.16}$$

General:

$$\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_s) = \mathbf{h}_t^T \mathbf{W}_a \bar{\mathbf{h}}_s \tag{D.17}$$

Concat (similar to Bahdanau):

$$\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_s) = \mathbf{v}_a^T \tanh(\mathbf{W}_a[\mathbf{h}_t; \bar{\mathbf{h}}_s]) \tag{D.18}$$

D.2.3 Impact and Applications

The attention mechanism revolutionized sequence-to-sequence learning by:

- Eliminating the fixed-length bottleneck
- Enabling better handling of long sequences
- Providing interpretability through attention weight visualization
- Allowing the model to learn alignment between input and output sequences

This innovation laid the groundwork for the Transformer architecture, which extends the concept of attention to self-attention, where a sequence attends to itself to compute better representations. The success of attention in encoder-decoder models demonstrated that explicit modeling of dependencies through learned attention weights could outperform implicit encoding in fixed vectors, fundamentally changing the design of neural architectures for sequential data.

APÊNDICE E – VISION TRANSFORMERS: DETAILED ARCHITECTURE

This appendix provides detailed technical information about Vision Transformer architectures. While Vision Transformers were benchmarked in this thesis, the final model architecture selected ResNet-50V2 as the backbone due to its superior stability and performance characteristics for the plant disease classification task.

E.1 TRANSFORMER ARCHITECTURE ORIGINS

Building upon the encoder-decoder paradigm and attention mechanisms, the Transformer architecture was introduced as a novel neural network design that relies entirely on attention mechanisms, dispensing with recurrence and convolutions altogether (VASWANI et al., 2017). The key innovation was the self-attention mechanism, which allows the model to weigh the importance of different parts of the input sequence when processing each element. This approach enabled more efficient parallelization during training and better capture of long-range dependencies compared to recurrent neural networks.

The original Transformer implements an encoder-decoder architecture, where both components are composed of stacks of identical layers. Each encoder layer contains two sub-layers: a multi-head self-attention mechanism and a position-wise fully connected feed-forward network. Residual connections are employed around each sub-layer, followed by layer normalization. This design proved highly effective for sequence-to-sequence tasks, particularly machine translation, and later became the foundation for breakthrough models like BERT (LEE; TOUTANOVA, 2018) and GPT (RADFORD et al., 2018).

E.2 SELF-ATTENTION MECHANISM

The self-attention mechanism, also known as scaled dot-product attention, is the core component that enables Transformers to model relationships between all positions in a sequence simultaneously. Given an input sequence, self-attention computes a weighted representation where each position can attend to all positions in the input.

Mathematically, the attention function can be described as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{E.1}$$

where Q, K, and V represent the query, key, and value matrices respectively, and d_k is the dimension of the key vectors. The queries and keys are used to compute attention weights, which determine how much focus to place on different parts of the input when computing the output representation.

Multi-head attention extends this concept by running multiple attention functions in parallel, each with different learned linear projections of the queries, keys, and values. This allows the model to jointly attend to information from different representation subspaces at different positions:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O \tag{E.2}$$

where each $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$ and W^O is a learned output projection matrix.

E.3 STAND-ALONE SELF-ATTENTION MODELS

Before the advent of Vision Transformers, researchers explored whether convolutional operations could be replaced by self-attention mechanisms within traditional architectures (RAMACHANDRAN et al., 2019). This work introduced a local variant of self-attention that could be applied within neighborhood patches to substitute spatial convolutions across all layers of a ResNet backbone.

The local self-attention mechanism restricts attention computation to a local neighborhood around each spatial location, making it computationally feasible for high-resolution images. This approach maintains the spatial inductive bias of convolutions while introducing the flexibility of attention mechanisms. Their fully self-attentional model achieved comparable top-1 accuracy on ImageNet while reducing computational cost by approximately 12% and parameters by 29% relative to the best convolutional baseline. On COCO object detection, it matched RetinaNet's performance while using 39% fewer FLOPs and 34% fewer parameters.

E.4 VISION TRANSFORMER ARCHITECTURE

Building on the success of Transformers in NLP and the promising results of stand-alone attention, the Vision Transformer (ViT) was proposed as a direct application of the Transformer architecture to images (DOSOVITSKIY et al., 2020). The key innovation lies in treating images as sequences of patches, enabling the use of standard Transformer encoders without architectural modifications.

E.4.1 Patch-Based Image Processing

The fundamental challenge in applying Transformers to images lies in handling the two-dimensional structure and high dimensionality of visual data. ViT addresses this by dividing an input image $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ into a sequence of flattened 2D patches $\mathbf{x}_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$, where (H, W) is the resolution of the original image, C is the number of channels, (P, P) is the resolution of each image patch, and $N = HW/P^2$ is the resulting number of patches.

Each patch is then linearly embedded into a D-dimensional space using a trainable linear projection:

$$\mathbf{z}_0 = [\mathbf{x}_{class}; \mathbf{x}_p^1\mathbf{E}; \mathbf{x}_p^2\mathbf{E}; \cdots ; \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{pos} \tag{E.3}$$

where $\mathbf{E} \in \mathbb{R}^{(P^2 \cdot C) \times D}$ is the patch embedding matrix, $\mathbf{x}_{class}$ is a learnable class token prepended to the sequence (similar to BERT's [CLS] token), and $\mathbf{E}_{pos} \in \mathbb{R}^{(N+1) \times D}$ contains learnable position embeddings.

The patch size P represents a crucial hyperparameter that affects both computational efficiency and model performance. Smaller patches result in longer sequences and higher computational cost but may capture finer-grained details, while larger patches reduce computational requirements but may lose spatial resolution. Common choices include $P = 16$ (ViT-B/16, ViT-L/16) and $P = 32$ (ViT-B/32, ViT-L/32).

E.4.2 Transformer Encoder Processing

The embedded patches are processed by a standard Transformer encoder consisting of L layers. Each layer applies multi-head self-attention followed by a feed-forward

network, with residual connections and layer normalization:

$$\mathbf{z}'_l = \text{MSA}(\text{LN}(\mathbf{z}_{l-1})) + \mathbf{z}_{l-1} \tag{E.4}$$

$$\mathbf{z}_l = \text{MLP}(\text{LN}(\mathbf{z}'_l)) + \mathbf{z}'_l \tag{E.5}$$

where MSA denotes multi-head self-attention, LN represents layer normalization, and MLP is a two-layer feed-forward network with GELU activation.

The self-attention mechanism enables each patch to attend to all other patches in the image, allowing the model to capture long-range spatial dependencies that would require many layers in convolutional networks. This global receptive field from the first layer is a key advantage of the Transformer architecture for vision tasks.

E.4.3 Classification and Output

For image classification, the output of the class token from the final Transformer layer is used as the image representation:

$$\mathbf{y} = \text{LN}(\mathbf{z}_L^0) \tag{E.6}$$

where $\mathbf{z}_L^0$ represents the class token output from the L-th layer. This representation is then passed through a classification head (typically a simple linear layer) to produce the final predictions.

E.5 PERFORMANCE AND SCALING PROPERTIES

ViT demonstrated that when pretrained on large datasets such as ImageNet-21k (14M images) or the proprietary JFT-300M dataset (300M images), it can surpass the performance of state-of-the-art CNNs on various benchmarks. On ImageNet classification, ViT-L/16 achieved 87.76% top-1 accuracy when pretrained on JFT-300M, outperforming the best ResNet and EfficientNet models while requiring substantially less computational resources for pretraining.

The scaling properties of ViT follow similar trends to those observed in NLP Transformers. Performance generally improves with increased model size (more layers, wider hidden dimensions, more attention heads) and larger pretraining datasets. However, ViT

requires more data than CNNs to achieve competitive performance, particularly when training from scratch on smaller datasets like ImageNet-1k.

E.6 INDUCTIVE BIASES AND DESIGN CONSIDERATIONS

E.6.1 Understanding Inductive Biases

Inductive biases refer to the set of assumptions that a learning algorithm makes to predict outputs for inputs it has not encountered during training (BISHOP, 2006). In the context of neural networks, inductive biases are architectural constraints or design choices that guide the model toward learning certain types of patterns or relationships in the data. These biases can be thought of as prior knowledge embedded in the model's structure that helps it generalize from limited training examples.

Inductive biases serve several important functions:

- **Sample Efficiency:** They reduce the amount of data needed to learn effective representations by constraining the hypothesis space
- **Generalization:** They help models generalize better to unseen data by encoding domain-specific knowledge
- **Computational Efficiency:** They can make learning more efficient by focusing on relevant patterns
- **Interpretability:** They make model behavior more predictable and interpretable

The choice of inductive biases represents a fundamental trade-off in machine learning: stronger biases can lead to better performance with limited data but may limit the model's flexibility to learn unexpected patterns, while weaker biases provide more flexibility but require larger datasets to achieve good generalization.

E.6.2 Inductive Biases in CNNs vs. Vision Transformers

CNNs incorporate several strong inductive biases that are well-suited for visual data:

Translation Equivariance: Convolutional layers ensure that if an object is translated in the input image, the corresponding feature map is translated by the same amount.

This is achieved through weight sharing across spatial locations, meaning the same filter is applied at every position in the input.

Locality: The use of small, local receptive fields in convolutional kernels embeds the assumption that nearby pixels are more likely to be related than distant ones. This reflects the natural structure of images where local features (edges, textures) combine to form larger structures.

Hierarchical Feature Learning: The layered structure of CNNs, combined with pooling operations, creates an inductive bias toward learning hierarchical representations—from low-level features like edges in early layers to high-level semantic concepts in deeper layers.

Scale Invariance: Through pooling operations and hierarchical processing, CNNs develop some degree of scale invariance, allowing them to recognize objects at different sizes.

In contrast, ViT has minimal built-in assumptions about image structure. The primary vision-specific inductive biases in ViT are:

2D Position Embeddings: These provide the model with information about the spatial arrangement of patches, but this spatial understanding must be learned rather than being architecturally enforced.

Patch Extraction: The process of dividing images into patches introduces a weak locality bias, as information within each patch is processed together, but relationships between patches must be learned through attention.

Permutation Invariance: The self-attention mechanism is inherently permutation-invariant, meaning the model must learn spatial relationships entirely from the position embeddings and data.

This reduced inductive bias in ViT can be both an advantage and a limitation. On one hand, it allows the model to learn more flexible representations and discover patterns that might not conform to traditional assumptions about visual structure. On the other hand, it requires substantially larger datasets to achieve good generalization, as the model must learn spatial relationships, translation equivariance, and hierarchical structure entirely from data.

The lack of inherent spatial structure understanding explains why ViT performs poorly when trained from scratch on smaller datasets like ImageNet-1k but excels when pretrained on large-scale datasets like JFT-300M. With sufficient data, the self-attention

mechanism learns to understand spatial locality, translation properties, and hierarchical relationships that are built into CNN architectures by design.

E.7 COMPARATIVE ANALYSIS AND IMPLICATIONS

The success of Vision Transformers has profound implications for computer vision research and practice. The ability of pure attention-based models to match or exceed CNN performance challenges the long-held assumption that convolutional inductive biases are indispensable for visual tasks. This paradigm shift has opened new research directions and inspired numerous follow-up works exploring hybrid architectures, hierarchical Vision Transformers, and efficient attention mechanisms.

The computational characteristics of ViT differ significantly from CNNs. While CNNs have quadratic complexity with respect to kernel size but linear complexity with respect to input size, ViT has quadratic complexity with respect to the number of patches (and thus input resolution). This makes ViT more suitable for moderate-resolution images but potentially prohibitive for very high-resolution inputs without architectural modifications.

Furthermore, the global receptive field of ViT from the first layer enables better modeling of long-range dependencies compared to CNNs, which build up their receptive field gradually through multiple layers. This property is particularly beneficial for tasks requiring understanding of global image context or relationships between distant image regions.

The emergence of Vision Transformers has catalyzed a broader transformation in computer vision, leading to the development of numerous variants and applications across different visual tasks, from object detection and segmentation to video understanding and multimodal learning. This represents not just a new model architecture, but a fundamental shift in how we approach visual representation learning.

APÊNDICE F – RETRIEVAL AUGMENTED GENERATION

This appendix provides background information on Retrieval Augmented Generation (RAG), a paradigm that combines retrieval-based and generation-based approaches in language models. While not directly used in this thesis, it represents an important contemporary development in AI systems.

F.1 OVERVIEW

Retrieval Augmented Generation (RAG) represents a paradigm that combines the strengths of retrieval-based and generation-based approaches to address limitations of purely parametric language models (LEWIS et al., 2020). While large language models (LLMs) demonstrate impressive capabilities in generating coherent and contextually relevant text, they face challenges including hallucination (generating plausible but factually incorrect information), inability to access up-to-date information beyond their training cutoff, and difficulty in attributing generated content to specific sources. RAG addresses these limitations by augmenting the generation process with relevant information retrieved from external knowledge sources.

F.2 ARCHITECTURAL OVERVIEW

The RAG framework consists of three primary components working in concert: a retrieval system, a knowledge base, and a generation model. The process operates as follows:

1. **Query Encoding:** The user's input query is transformed into a dense vector representation (embedding) using an encoder model.
2. **Retrieval:** The system searches a knowledge base to find the most relevant documents or passages based on similarity between the query embedding and document embeddings.
3. **Augmentation:** Retrieved documents are combined with the original query to form an augmented context.

4. **Generation:** A language model generates a response conditioned on both the original query and the retrieved context.

Formally, given a query q, the RAG model computes:

$$p(y|q) = \sum_{d \in \mathcal{D}_{\text{top-k}}} p(d|q) \cdot p(y|q, d) \tag{F.1}$$

where $\mathcal{D}_{\text{top-k}}$ represents the k most relevant documents retrieved, $p(d|q)$ is the retrieval probability, and $p(y|q, d)$ is the generation probability conditioned on both query and retrieved document.

F.3 EMBEDDINGS: DENSE REPRESENTATIONS FOR SEMANTIC SIMILARITY

Embeddings are dense, continuous vector representations that capture semantic meaning of text, images, or other data modalities in a high-dimensional space. Unlike sparse representations such as one-hot encoding or traditional bag-of-words models, embeddings encode semantic relationships such that semantically similar items have similar vector representations.

F.3.1 Word and Sentence Embeddings

Early embedding approaches like Word2Vec (MIKOLOV et al., 2013) and GloVe (PENNINGTON; SOCHER; MANNING, 2014) learned distributed representations of individual words where semantic and syntactic relationships are preserved through vector arithmetic. For instance, the famous example: $\vec{v}_{\text{king}} - \vec{v}_{\text{man}} + \vec{v}_{\text{woman}} \approx \vec{v}_{\text{queen}}$ demonstrates how embeddings capture semantic analogies.

Modern approaches generate contextualized embeddings where the representation of a word depends on its context. Models like BERT (DEVLIN et al., 2018) produce different embeddings for the same word in different sentences, capturing nuanced meanings. Sentence and document embeddings extend this concept to encode entire text passages into fixed-dimensional vectors, enabling semantic similarity comparisons at higher levels of granularity (REIMERS; GUREVYCH, 2019).

F.3.2 Computing Semantic Similarity

Given two text embeddings $\mathbf{e}_1$ and $\mathbf{e}_2$, similarity is typically measured using cosine similarity:

$$\text{sim}(\mathbf{e}_1, \mathbf{e}_2) = \frac{\mathbf{e}_1 \cdot \mathbf{e}_2}{\|\mathbf{e}_1\|\|\mathbf{e}_2\|} = \frac{\sum_{i=1}^{d} e_{1,i} e_{2,i}}{\sqrt{\sum_{i=1}^{d} e_{1,i}^2} \sqrt{\sum_{i=1}^{d} e_{2,i}^2}} \tag{F.2}$$

where d is the dimensionality of the embedding space. Cosine similarity ranges from -1 (completely dissimilar) to 1 (identical), with values close to 1 indicating high semantic similarity.

Alternative distance metrics include Euclidean distance and dot product similarity, each with different characteristics and computational properties. The choice of metric can affect retrieval performance and is often determined empirically for specific applications.

F.4 CHUNKING STRATEGIES FOR DOCUMENT PROCESSING

Chunking refers to the process of dividing large documents into smaller, semantically coherent segments before embedding and indexing. This is crucial for several reasons:

- **Context Window Limitations:** Language models and embedding models have maximum input length constraints (e.g., 512 tokens for many BERT variants).
- **Retrieval Granularity:** Smaller chunks enable more precise retrieval of relevant information rather than entire documents.
- **Generation Quality:** Providing focused, relevant context improves generation quality and reduces the likelihood of the model being distracted by irrelevant information.

F.4.1 Chunking Methods

Fixed-Size Chunking: The simplest approach divides text into chunks of fixed length (e.g., 512 tokens) with optional overlap between consecutive chunks. While computationally efficient, this method may split semantically coherent units.

Sentence-Based Chunking: This approach respects sentence boundaries, ensuring each chunk contains complete sentences. Multiple sentences can be grouped until a maximum length is reached, preserving semantic coherence.

Paragraph or Section-Based Chunking: For structured documents, natural divisions like paragraphs, sections, or subsections provide semantically meaningful chunks that preserve document structure.

Semantic Chunking: Advanced methods use natural language processing techniques to identify topic shifts or semantic boundaries, creating chunks that correspond to coherent ideas or concepts (HEARST, 1997).

Recursive Chunking: Hierarchical approaches create chunks at multiple granularities, enabling retrieval at different levels of detail depending on the query.

Effective chunking often requires domain-specific considerations and may involve hybrid approaches that combine multiple strategies. Chunk overlap (e.g., 50-100 tokens) between consecutive segments helps preserve context that might be split across boundaries.

F.5 VECTOR DATABASES FOR EFFICIENT SIMILARITY SEARCH

Vector databases (also called vector stores or vector search engines) are specialized data management systems optimized for storing, indexing, and querying high-dimensional vector embeddings. Unlike traditional relational databases optimized for exact matches and structured queries, vector databases excel at approximate nearest neighbor (ANN) search in high-dimensional spaces.

F.5.1 The Curse of Dimensionality

Exact nearest neighbor search in high-dimensional spaces suffers from the curse of dimensionality: as dimensionality increases, the ratio between the distances to the nearest and farthest points approaches 1, making distance-based similarity less meaningful. Additionally, exact search requires comparing the query vector against all stored vectors, resulting in $O(n)$ complexity that becomes prohibitive for large-scale applications (WEBER; SCHEK; BLOTT, 1998).

F.5.2 Approximate Nearest Neighbor Algorithms

Vector databases employ sophisticated indexing structures and approximate search algorithms to achieve sub-linear query times while maintaining high recall:

Locality-Sensitive Hashing (LSH): LSH uses hash functions that map similar vectors to the same hash buckets with high probability, enabling fast approximate search (ANDONI; INDYK, 2008).

Hierarchical Navigable Small World (HNSW): HNSW constructs a multi-layer graph structure where each layer contains edges connecting nearby vectors. Search traverses from coarse to fine layers, achieving excellent performance with logarithmic complexity (MALKOV; YASHUNIN, 2018).

Product Quantization (PQ): PQ compresses vectors by partitioning the embedding space and quantizing each subspace separately, reducing memory footprint while enabling fast distance computation (JÉGOU; DOUZE; SCHMID, 2010).

Inverted File Index (IVF): IVF partitions the vector space using clustering (e.g., k-means) and maintains an inverted index mapping cluster centroids to vectors, enabling fast approximate search by examining only nearby clusters.

F.5.3 Popular Vector Database Systems

Modern vector database implementations include Faiss (JOHNSON; DOUZE; JÉGOU, 2019), Pinecone, Weaviate, Milvus, and Qdrant, each offering different trade-offs between query speed, indexing time, memory consumption, and accuracy. Many provide:

- Hybrid search combining dense vector similarity with traditional keyword search
- Filtering capabilities to restrict search to subsets based on metadata
- Horizontal scalability for handling billions of vectors
- Integration with popular machine learning frameworks

F.6 RAG APPLICATIONS AND VARIANTS

RAG has found applications across diverse domains including question answering, conversational AI, fact-checking, and domain-specific knowledge systems. Variants of RAG have been proposed to address specific challenges:

Self-RAG (ASAI et al., 2023) introduces reflection tokens that enable the model to critique and refine its own retrieval and generation decisions.

Iterative RAG performs multiple retrieval-generation cycles, using initial generated content to refine subsequent retrieval queries.

HyDE (Hypothetical Document Embeddings) (GAO et al., 2022) generates a hypothetical answer to the query, embeds it, and uses it for retrieval, often improving retrieval quality for complex queries.

The integration of RAG with large language models represents a significant advancement in building AI systems that can access, verify, and cite external knowledge, addressing key limitations of purely parametric models while maintaining the fluency and reasoning capabilities of modern language models.