VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Internship 2 - (CO5019)

An Autonomous Agentic Framework for Crop Health: Bridging Open-Set Detection and Expert Knowledge

Instructor: Dr. Nguyen Quang Hung
Students: Tran Quoc Anh – 2370386

Ho Chi Minh City, December 2025

Declaration of Authenticity

I declare that this research is our own work, conducted under the supervision and guidance of Dr. Nguyen Quang Hung. The result of our research is legitimate and has not been published in any forms prior to this. All materials used within this researched are collected myself by various sources and are appropriately listed in the references section.

In addition, within this research, we also used the results of several other authors and organizations. They have all been aptly referenced.

In any case of plagiarism, we stand by our actions and will be responsible for it. Ho Chi Minh City University of Technology therefore are not responsible for any copyright infringements conducted within our research.

Ho Chi Minh City, December 2025
Authors
Tran Quoc Anh

Abstract

Agriculture faces growing threats from diseases and climate change, making autonomous health management systems essential. Most current Artificial Intelligence systems rely on "closed-set" detectors that map inputs to a fixed list of known classes. These models often fail in real field conditions and cannot identify new pests without constant retraining. Additionally, there is a "Semantic Gap" between simply detecting a symptom and providing actual farming advice.

This thesis proposes an Autonomous Multimodal Agronomy Agent designed to bridge these gaps by unifying open-set perception with expert-level reasoning. The proposed architecture functions through a structured "Observe → Verify → Reason → Act" loop. First, the system utilizes Grounding DINO[19] to perform open-set object detection, enabling the identification of pests and diseases-ranging from general-purpose anomalies to specific Vietnamese crops on arbitrary natural language descriptions rather than predefined labels. Second, a fine-tuned CLIP[25] (Contrastive Language-Image Pre-training) model filters out false positives by matching visual data with scientific descriptions in a shared embedding space. Finally, an Agentic Retrieval-Augmented Generation (RAG) engine links these findings to a database of agricultural literature. This ensures that the final reports for farmers are fact-based and free from model hallucinations.

While these individual technologies are powerful, this work focuses on integrating them into a unified workflow capable of handling complex, whole-plant imagery. By moving beyond simple leaf classification, this framework offers a scalable way to deliver expert-level recommendations directly to agricultural practitioners.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Agriculture is important for the world's food supply, but it is under threat from new diseases and changes in the weather. Checking crops for health problems requires regular monitoring and expert knowledge. However, doing this manually is very difficult on large farms or in remote areas.

Artificial Intelligence has already been applied to this problem, but current systems primarily rely on a fixed set of predefined disease labels. These models often fail to generalize when encountering novel pests or complex "whole-plant" image in unstructured field conditions. Furthermore, there is a persistent "Semantic Gap" between simple visual detection and providing actionable, scientifically grounded advice. There is a need for a system that can see what is happening, verify the findings, and provide expert reasoning automatically to help farmers.

1.2 Goals

The main goal of this thesis is to build an autonomous system for crop health that combines vision and reasoning. Our specific goals are:

- Implement an Observe→Verify→Reason→Act loop to move from simple classification to a complete diagnostic workflow.
- Develop an open-set perception module to identify pests and diseases based on natural language descriptions instead of fixed class IDs.
- Integrate a fine-tuned CLIP verifier to reduce false positives by matching images with scientific descriptions.
- Build an Agentic RAG (Retrieval-Augmented Generation) engine that that links detections to a database of agricultural books and papers, eliminating the risk of Large Language Model (LLM) hallucinations.

1.3 Scopes

This study focuses on the development of a vision-based diagnostic agent with the following boundaries:

- The system is designed to process "whole-plant" image captured in field environments, rather than a single object.
- The system is built for general-purpose application, but I focus on Vietnamese crops like durian and tomato.
- The final output of the system is a diagnostic report and treatment recommendation.

1.4 Thesis Structure

This thesis is organized into six chapters, following the logical progression from theoretical foundations to experimental validation:

- **Chapter 1: Introduction** presents the motivation, objectives, and scope of the research.
- **Chapter 2: Theoretical Background** details the underlying mechanisms of Grounding DINO, CLIP, and Agentic RAG systems.
- **Chapter 3: Related Works** reviews the evolution of AI in agriculture and identifies the research gaps in current multimodal integration.
- **Chapter 4: Methodology** describes the proposed "Observe–Verify–Reason–Act" architecture and its specific implementation details.
- **Chapter 5: Prototype Development** show the demo application, which display step by step how the works done.
- **Chapter 6: Evaluation Plan and Expected Contribution** outlines the metrics for system validation, highlights technical contributions, and proposes future research directions.

Chapter 2

Theoretical Background

2.1 Deep Learning in Computer Vision

2.1.1 CNNs to Transformers

Since 2012, Convolutional Neural Networks (CNNs) have formed the backbone of most computer vision systems. Successive architectures, including VGG[32], ResNet[10], and EfficientNet[33], consistently achieved strong performance on large-scale benchmarks such as ImageNet[30] and COCO[18]. Besides that, CNN-based detectors like YOLO[27] became widely adopted in real-time applications due to their computational efficiency. CNNs are built to focus on local patterns and recognize them anywhere, which makes them efficient, lightweight, and ideal for running on edge devices.

For tasks requiring global context, this same locality bias is a bottleneck. Capturing long-range dependencies - such as spatial relationships between distant plant organs or complex scene configurations - often requires very deep CNN architectures or additional fine-tuned modules. These designs increase model complexity and may still be insufficient for fine-grained recognition tasks where entire scene understanding is essential.

The introduction of Vision Transformers (ViTs)[6] marked a fundamental change in computer vision. ViTs model global interactions from the first layer by using self-attention mechanisms and treating images as sequences of patches. Unlike the hierarchical aggregation used in CNNs, the self-attention mechanism allows every patch to attend to every other patch, which is particularly advantageous for capturing long-range dependencies in complex scenes.

There are two primary implications for modern vision systems:

- **Global Reasoning:** ViTs are better able to handle occlusion and complex spatial distributions due to their natural integration of global structure.
- **Multimodal Alignment:** Since Transformers process image patches and text tokens via the same attention mechanism, they make it easier to create unified embedding spaces . This is the foundation of models like CLIP, enabling the zero-shot and open-vocabulary capabilities required for adaptable crop health management in open-set agricultural environments.

2.1.2 Closed-Set vs. Open-Set Detection

Conventional object detectors, from two-stage models like Faster R-CNN[28] to one-stage architectures like YOLO, typically operate in a closed-set setting. These models learn a mapping from an image to a fixed, finite set of categories $\mathcal{Y}_{\text{known}}$ (e.g., the 80 classes in MS COCO)[18]. At test time, the model implicitly assumes that all visible objects belong to this known set. Any object outside this taxonomy - such as a rare pest or a novel disease symptom - is either forced into a known class (misclassification) or treated as background (suppression).

Open-set object detection (OSOD) generalizes this formulation. It requires a model to satisfy two primary objectives: (i) accurately classify instances from $\mathcal{Y}_{\text{known}}$, while (ii) identifying or acknowledging inputs from $\mathcal{Y}_{\text{unknown}}$ without overconfident misclassification [5]. While early OSOD work focused on rejecting unknowns, recent advances in *open-vocabulary* models provide a more flexible solution. By leveraging large-scale vision–language pre-training (e.g., CLIP), these models replace fixed classification layers with text-embedding similarity scores.[26, 35] This allows a detector not just to reject an unknown object, but to localize it based on descriptive text prompts (e.g.,"necrotic lesion with yellow halo") that were never seen during training.

This distinction is critical because agriculture is inherently an *open-set* environment. The biological landscape is dynamic: new pathogens emerge, invasive species migrate, and symptom appearances shift due to environmental stress.[9] A closed-set detector is brittle in this context; it cannot recognize a threat it has not been explicitly trained to see. Recent work in plant pathology underlines the necessity of models that can handle these "unknown" classes to prevent silent failures in the field.[31]

In this thesis, we address this limitation by treating crop health monitoring as an open-set problem driven by text. Rather than relying on a static list of pest categories, we propose a pipeline that integrates open-vocabulary detection with agronomic knowledge. The system is designed to localize both known and novel visual patterns by conditioning the detector on textual descriptions, thereby bridging the gap between rigid closed-set training and the dynamic reality of agricultural management.

2.2 Grounding DINO

Grounding-DINO[19] is a popular large vision-language model equipped with open-set object detection capabilities. It is designed to overcome the constraints of fixed-class detectors by recognizing novel objects defined solely by natural language input. The model achieves this by combining the high-performance transformer-based detection architecture of DINO with the grounded language-image pre-training methodology of GLIP. This integration allows Grounding DINO to perform zero-shot detection by aligning visual features with textual descriptions at a foundational level.

2.2.1 DETR

DETR (Detection Transformer) [3] is a deep learning model that fundamentally shifts object detection from a regression and classification task to a direct set prediction problem. By combining the Transformer architecture with self-attention mechanisms, DETR captures global context and long-range dependencies across the image, addressing limitations of

CNN-based methods that primarily focus on local features. Furthermore, its set-based prediction mechanism utilizing bipartite matching eliminates the need for hand-crafted anchors and complex post-processing steps like Non-Maximum Suppression (NMS), streamlining the pipeline into an end-to-end process.

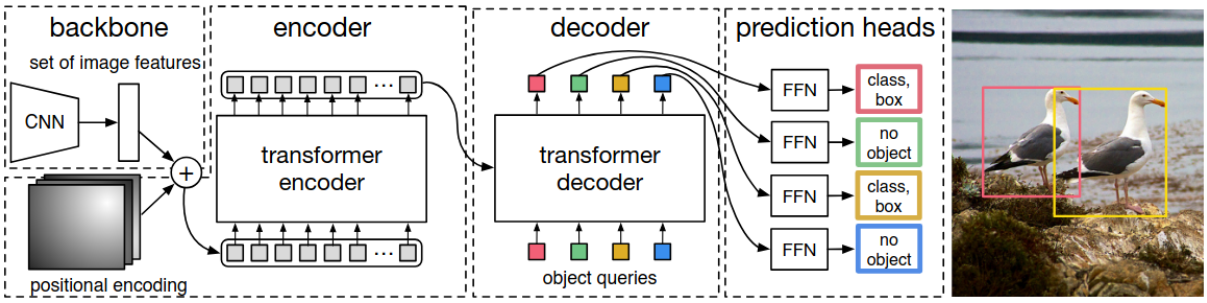

Figure 2.1: DETR architecture [3].

To achieve this, the DETR architecture follows a streamlined pipeline:

- **Backbone:** The input image is processed by a CNN to extract a low-resolution feature map. Positional encodings are added to these features to preserve the spatial structure that Transformers otherwise ignore.
- **Encoder:** The flattened features are passed through a Transformer Encoder. Here, self-attention layers aggregate information from across the entire image, enriching each pixel's representation with global context.
- **Decoder:** The Decoder utilizes learned object queries - vectors representing potential objects - to interact with the encoded image features. It reasons about the relations between objects and the global image context.
- **Prediction Heads:** Finally, the decoder output is passed to Feed-Forward Networks (FFNs) that predict class labels and bounding box coordinates for each query. If no object is detected for a specific query, it is assigned a "no object" class.

Despite its conceptual advantages, the vanilla DETR architecture faces significant limitations, specifically regarding slow convergence rates and difficulties in processing high-resolution images.

2.2.2 DINO

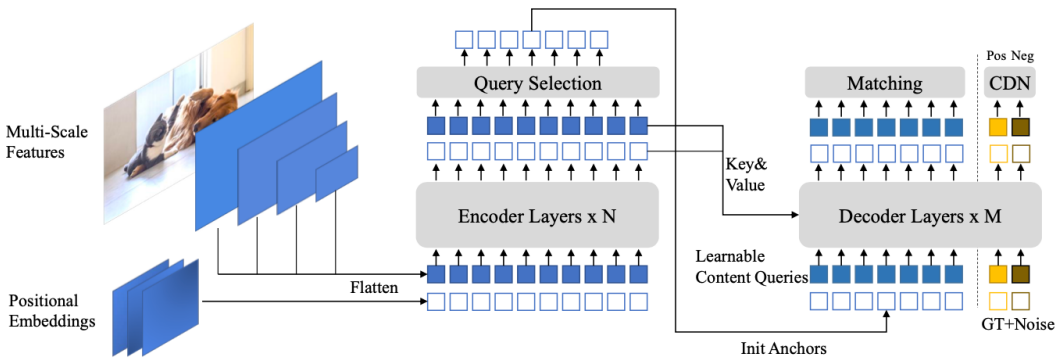

Figure 2.2: DINO Architecture (Source: Internet).

DINO (DETR with Improved deNoising anchOr boxes) builds upon the standard DETR framework by integrating specific architectural enhancements designed to optimize convergence speed and detection performance on high-resolution inputs. These improvements focus on two primary areas: attention efficiency and query stability.

Multi-Scale Deformable Attention
To process high-resolution feature maps efficiently, DINO adopts the Deformable Attention mechanism from Deformable DETR. Unlike global self-attention, which computes dependencies across all spatial locations resulting in quadratic computational complexity - deformable attention restricts the attention field to a small set of learned sampling points around a reference. This sparse attention mechanism allows the model to capture fine-grained details for small object detection without incurring prohibitive computational costs.

Query Optimization and Denoising
DINO mitigates the convergence latency of the original DETR through two complementary strategies:

- **Dynamic Anchor Boxes (DAB-DETR):** Static object queries are replaced with dynamic anchor boxes. These anchors explicitly formulate the query position as dynamic spatial priors (coordinates and scale), clarifying the spatial relationship between the query and the target features.
- **Contrastive Denoising Training (DN-DETR):** To stabilize the bipartite matching process, DINO introduces a denoising training task. The model is fed "noisy" ground-truth bounding boxes and trained to reconstruct the original coordinates. This auxiliary task reduces the ambiguity of bipartite matching during early training phases, significantly accelerating convergence.

2.2.3 GLIP and Grounded Pre-Training

Grounding DINO extends the capabilities of standard detection by incorporating language supervision, enabling "zero-shot" detection where the model can identify objects based on arbitrary text descriptions without re-training. This relies on the methodology introduced by GLIP, which reformulates object detection as a **phrase grounding** task, replacing fixed class IDs with alignable word embeddings.

The model's generalization capability stems from a diverse combination of datasets during pre-training:

- **Detection Data (e.g., Object365, LVIS):** Provides standard bounding boxes and class labels to maintain localization precision.
- **Caption Data (e.g., Cap24M):** Offers detailed textual descriptions of images to expand the semantic vocabulary.
- **Grounded Data (e.g., RefCOCO):** Crucially, this data links specific phrases in a sentence to specific image regions. This bridges the gap between visual features and linguistic semantics, allowing the model to learn the association between textual concepts and visual patterns directly.

2.2.4 Grounding DINO Architecture

The Grounding DINO architecture is designed to process image-text pairs simultaneously to achieve open-set detection. As illustrated in Figure 2.3, the model comprises several key components: a dual-backbone feature extractor, a feature enhancer for multi-modal

fusion, a language-guided query selection module, and a cross-modality decoder.

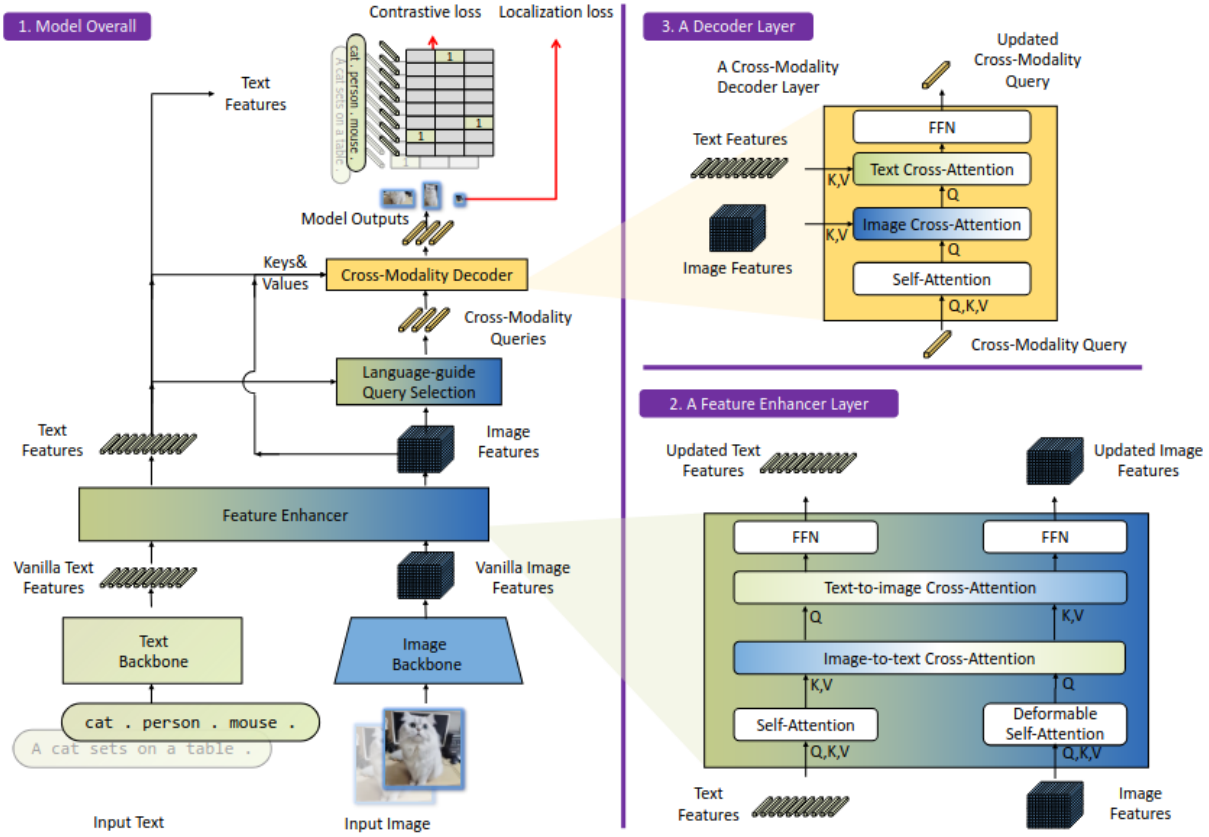

Figure 2.3: The framework of Grounding DINO [19].

Feature Extraction and Encoding

The model processes the input image and text prompt through separate backbones:

- **Image Backbone:** A Swin Transformer[20] is employed to extract hierarchical visual features from different layers, capturing multi-scale visual information.
- **Text Backbone:** The text prompt is tokenized using Byte-Pair Encoding (BPE) and encoded by a BERT model [4]. This produces a sequence of text embeddings with dimension $N \times 768$, where N is the number of tokens.

Feature Enhancer Module

To bridge the gap between modalities, the raw image and text embeddings are fed into a Feature Enhancer. This module consists of multiple layers employing both self-attention and bi-directional cross-attention mechanisms. These layers facilitate deep interaction between text-to-image and image-to-text contexts, ensuring that the visual features are contextually aware of the linguistic prompts before detection begins.

Language-Guided Query Selection

Unlike standard DETR which uses learned static queries, Grounding DINO initializes object queries based on the input text. The model calculates the dot product between the enhanced text and image features to generate similarity scores. It then selects the top N_I image features with the highest scores to serve as the initial object queries. This ensures the queries are relevant to the specific objects mentioned in the text prompt.

Sub-Sentence Level Representation

Grounding DINO refines how text is processed by introducing a "sub-sentence" level representation. Previous methods either compressed an entire sentence into a single feature (losing fine-grained detail) or allowed all words to interact indiscriminately, which

created unnecessary dependencies between unrelated category names. To solve this, Grounding DINO employs attention masks within the text encoder. These masks selectively block the attention mechanism, preventing unrelated categories from influencing each other. This ensures the model retains precise, per-word features necessary for fine-grained understanding without introducing noise from irrelevant word associations.

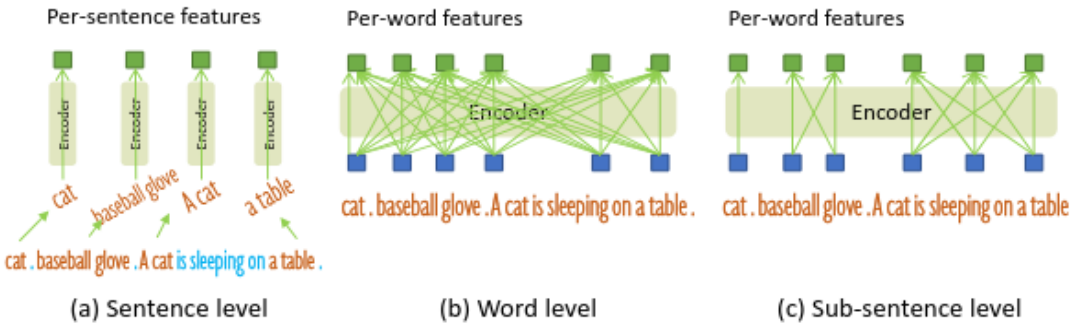

Figure 2.4: Comparisons of text representations [19].

Cross-Modality Decoder and Training Objectives
The selected queries are fed into a cross-modality decoder. Here, query features undergo self-attention followed by cross-attention with both the image and text feature maps to refine the object representations.

- **Prediction:** The output queries predict bounding boxes and class probabilities based on their similarity to the text features.
- **Loss Functions:** Training optimizes a combination of losses using bipartite matching. Classification utilizes a **contrastive loss**, where the dot product between query and text features produces logits optimized via Focal Loss [17]. Bounding box regression is supervised using L1 loss and Generalized IoU (GIoU) loss [29].

2.3 CLIP

CLIP[25] is a neural network adept at grasping visual concepts through natural language supervision. It operates by concurrently training a text encoder and an image encoder, focusing on a pre-training task that involves matching captions with corresponding images. This architecture allows CLIP[25] to adapt to a variety of visual classification benchmarks seamlessly. It does so by simply receiving the names of the visual categories to be recognized.

2.3.1 Learning from Unstructured Data

Traditional computer vision models have depended on supervised learning methods that need high-quality, human-annotated datasets. In standard benchmarks, models are trained to predict a fixed set of predetermined categories, which are manually verified for accuracy. While effective for closed-set tasks, this approach has two limitations: the high cost of scaling manual annotation to new domains and the inability of the model to generalize beyond its training taxonomy.

CLIP[25] resolves these limitations by changing from crowd-sourced labeling to natural language supervision. Instead of relying on a restricted ontology of categorical labels ,

CLIP learns from a massive dataset of 400 million image-text pairs collected from the public internet (WebImageText).

This method makes use of the semantic richness of unstructured text found on the web. By treating natural language descriptions as labels, the model is not confined to a static list of IDs but instead learns to associate visual features with a vast array of linguistic concepts. This transition enables the model to generalize unseen categories without task-specific retraining.

2.3.2 Contrastive Learning

At the core of CLIP[25] is a contrastive learning objective designed to align visual and textual representations into a shared embedding space. Unlike traditional classification tasks that map images to discrete labels, CLIP solves a proxy task of predicting which text description matches which image within a given batch.

For a batch of N (image, text) pairs, the model constructs an $N \times N$ similarity matrix by computing the cosine similarity between all image and text embeddings. The training objective is to maximize the similarity scores of the N correct pairings (the diagonal of the matrix) while minimizing the scores of the $N^2 - N$ incorrect pairings (the off-diagonal elements). This forces the model to learn a multi-modal embedding space where semantically similar images and texts are located close to each other.

This alignment is optimized using a symmetric cross-entropy loss over the similarity scores. Formally, for an image embedding I_i and a text embedding T_i, the loss function for the image-to-text direction is defined as:

$$\mathcal{L}_i^{(I \to T)} = -\log \frac{\exp(\text{sim}(I_i, T_i)/\tau)}{\sum_{j=1}^{N} \exp(\text{sim}(I_i, T_j)/\tau)} \tag{2.1}$$

where τ is a learnable temperature parameter that scales the logits. A symmetric loss $\mathcal{L}^{(T \to I)}$ is calculated for the text-to-image direction, and the total loss is the average of these two components.

2.3.3 Dual Encoders

CLIP processes two independent neural networks to process visual and textual inputs in parallel as show in Figure 2.5.

- **Image Encoder** (typically a Vision Transformer like ViT-B/32) maps images to feature vectors
- **Text Encoder** (a standard Transformer) processes natural language prompts.

Both encoders must project their outputs into a shared embedding space of identical dimension d via a learned linear layer. This dimensional alignment is important, as it allows the vectors from these different modalities to be compared directly using the dot product, enabling the contrastive mechanism described previously.

2.3.4 Zero-Shot Inference

Zero-shot inference enables the model to predict classes that were not explicitly encountered during training. This is achieved by synthesizing a linear classifier directly from natural

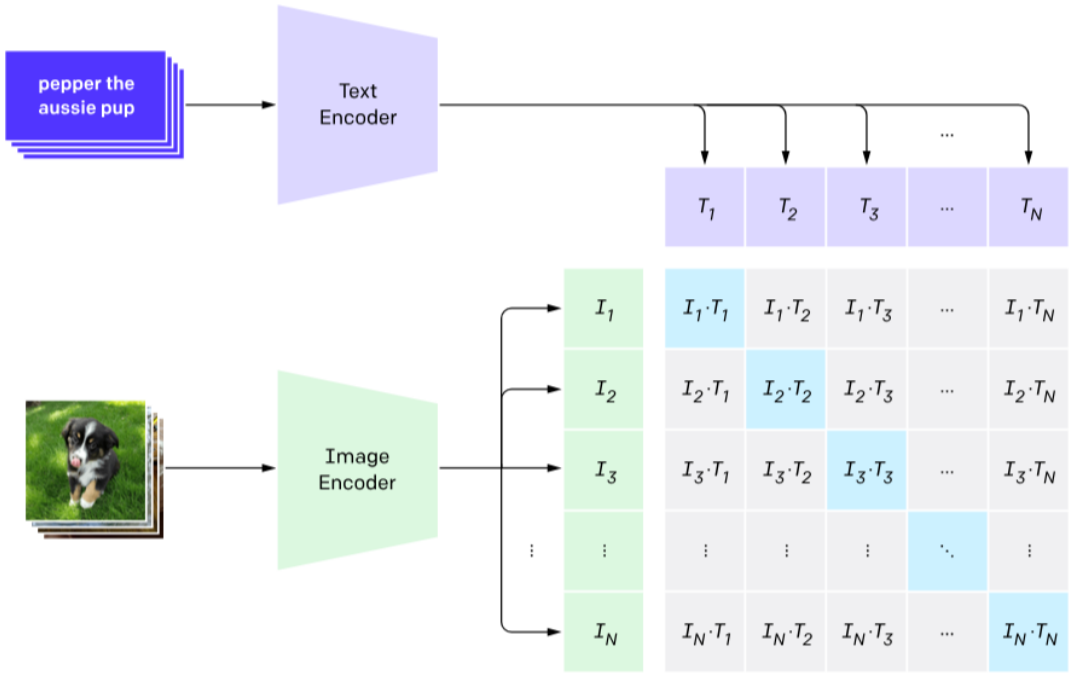

Figure 2.5: Text Encoder & Image Encoder (Source: Internet).

language descriptions.

At inference time, the names of the target categories (e.g., "healthy", "early blight") are inserted into a prompt template, such as "A photo of a {label}." These prompts are processed by the Text Encoder to generate a set of class embeddings $\{T_1, T_2, \ldots, T_K\}$. Simultaneously, the input image is mapped to a feature vector I by the Image Encoder.

Classification is performed by computing the cosine similarity between the image vector I and the set of text vectors. The predicted class is simply the one whose text embedding maximizes the dot product with the image embedding:

$$\hat{y} = \arg\max_k(\text{sim}(I, T_k)) \tag{2.2}$$

This mechanism effectively treats the text embeddings as the dynamic weights of a classifier, allowing the model to adapt to any open-set taxonomy solely by modifying the input text prompts.

2.4 Large Language Models (LLMs) & Reasoning

2.4.1 Generative Transformer Architecture

Large Language Models (LLMs) are defined as transformer-based models scaled to billions of parameters. Unlike earlier architectures that process an entire sentence at once (like BERT), modern LLMs primarily use a decoder-only architecture. This design processes text sequentially-reading from left to right-to generate coherent responses token by token.

Pre-training

Next Token Prediction The fundamental "engine" of these models is built during a

massive pre-training phase. The model reads vast amounts of text and learns to solve a single objective: Next Token Prediction. Here is how it works by steps:

1. Given a sequence of words (e.g., "The crop leaves are turns..."), the model calculates the probability of the most likely next word (e.g., "yellow").
2. By repeating this billions of times, the model develops a deep statistical understanding of language, grammar, and facts.

Instruction Tuning & Alignment

While pre-training teaches the model how to speak, it does not teach it to follow orders. A raw pre-trained model might just continue a sentence rather than answer a question. To fix this, modern models undergo a second training phase called Instruction Tuning (Ouyang et al., 2022). This "fine-tuning" step teaches the model to understand user intents and generate helpful, safe responses rather than just random text completions.

2.4.2 Chain-of-Thought (CoT) Reasoning

Chain of thought (CoT) [34] is a prompt engineering technique that enhances the output of large language models (LLMs), particularly for complex tasks involving multistep reasoning. It facilitates problem-solving by guiding the model through a step-by-step reasoning process by using a coherent series of logical steps. Instead of giving a final answer immediately, the model generates middle reasoning steps. This process allows the model to think more, breaking down a complex problem into smaller, clearer parts before concluding.

Reasoning is often considered an "emergent ability", meaning it appears naturally as the size and complexity of a model increase. Larger models tend to perform better at this because they have learned more patterns from massive datasets. However, increasing model size is not the only way to improve this. Advances in "instruction tuning" now allow smaller models to perform CoT reasoning effectively by training them on examples of how to respond step-by-step.

2.4.3 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation [13] is an artificial intelligence (AI) application that connects a generative AI model with an external knowledge base. The data in the knowledge base augments user queries with more context so the LLM can generate more accurate responses. RAG enables LLMs to be more accurate in domain-specific contexts without needing fine-tuning.

Vector Embeddings

To retrieve this information, the system must first understand the "meaning" of the text. This is done using Vector Embeddings, which map text into a high-dimensional numerical space. For example, OpenAI's text-embedding-3-small model converts a sentence into a vector of 1,536 numbers. In this space, concepts that are semantically similar (like "yellow leaves" and "nitrogen deficiency") are located close to each other, even if they do not share the exact same words.

Vector Indexing

Finding the closest matching vector among millions of documents can be slow if done one by one. To solve this, vector databases (like ChromaDB) use an algorithm called

Hierarchical Navigable Small World (HNSW)[21], HNSW organizes data into a multi-layered graph structure. This allows the system to quickly zoom in on the relevant neighborhood of data points rather than scanning the entire database, ensuring the agent retrieves information almost instantly.

Semantic Search Metrics

The relevance of a document is measured using Cosine Similarity. This metric calculates the angle between the query vector and the document vector. If the score is close to 1, it means the documents have almost the same meaning; if it is close to 0, it means they are unrelated. The agent uses this score to rank the retrieved documents and select only the most relevant evidence to verify its diagnosis.

2.5 Agentic AI Frameworks

2.5.1 The ReAct Paradigm

ReAct stands for Reasoning and Acting. It is a paradigm that overcomes the limitations of static Large Language Models (LLMs) by interleaving reasoning traces with executable actions.

In a standard LLM interaction, the model maps an input directly to an output based solely on pre-trained probability weights. In contrast, a ReAct agent operates dynamically by maintaining a continuous feedback loop:

- **Thought (Reasoning):** The agent utilizes Chain-of-Thought (CoT) prompting to decompose a complex user request into logical intermediate steps. This internal monologue allows the model to plan before acting.
- **Action:** Instead of relying on internal parametric knowledge (which may be hallucinated), the agent invokes external tools or functions-such as querying a sensor database or calculating a specific crop metric-to retrieve ground-truth data.
- **Observation:** The agent perceives the output from the tool and appends this new information to its context window to formulate the next step.

The fundamental difference between a standard chatbot and a ReAct agent lies in their operational scope. A chatbot is designed to "Talk," utilizing patterns to simulate conversation. A ReAct agent is designed to "Do," perceiving the environment through observations and affecting it through actions.

2.5.2 Neuro-Symbolic Integration (Structured Outputs)

While Large Language Models (LLMs) excel at probabilistic generation, they inherently lack deterministic control. In agricultural and industrial applications, this stochastic nature presents a safety risk; a system cannot "hallucinate" a chemical dosage or a sensor ID. To mitigate this, the system employs a Neuro-Symbolic approach.

Neuro-Symbolic AI integrates two distinct paradigms:

- Neural Component: The LLM, which handles intuition, semantic understanding, and unstructured data
- Symbolic Component: A logic-based validator that enforces strict rules, types, and constraints

In this thesis, this integration is implemented via Schema-Guided Generation (technically realized using Pydantic). By forcing the LLM to collapse its probability distribution into a pre-defined JSON schema, we ensure that the output is not merely text, but a valid executable object. This acts as a guardrail, ensuring that even if the agent's reasoning is creative, its final actuation commands remain syntactically and semantically valid.

Chapter 3

Related Works

3.1 The Big Picture of AI in Crop Health Management

The application of Artificial Intelligence in agriculture has evolved through three distinct paradigms, progressing from manual statistical analysis to the current use of generative reasoning. Recent surveys by Liakos et al. [15] and Kamilaris and Prenafeta-Boldú [11] classify these techniques based on their structural complexity and level of independence. Initially, techniques depended on fixed, manually created rules; however, the field has gradually moved towards data-driven Deep Learning, and most recently, toward Foundation Models that allow agents to do the reasoning rather than just classify.

3.1.1 Traditional Machine Learning Approaches

Prior to the widespread adoption of Deep Learning, crop health monitoring was dominated by feature engineering. In this paradigm, domain experts manually defined the visual characteristics of a disease-such as specific color histograms, texture patterns (e.g., GLCM), or shape descriptors. These features were then processed by classical classifiers like Support Vector Machines (SVM) or Random Forests [15].

While pioneers like Pydipati et al. [24] demonstrated success in controlled environments, these systems lacked robustness. They struggled to generalize in real-world conditions where lighting varies, or when symptoms appear subtly different across plant varieties. The rigidity of manual feature extraction meant that the system could only detect what it was explicitly programmed to look for, creating a bottleneck in scalability.

3.1.2 Supervised Deep Learning Dominance

The introduction of Convolutional Neural Networks (CNNs) marked a pivotal shift, allowing systems to automatically learn features from raw data. Early works by Mohanty et al. [22] and Ferentinos [8] utilized architectures like AlexNet and GoogLeNet on the PlantVillage dataset, achieving human-level accuracy in disease classification.

This era evolved rapidly from simple classification (labelling an entire image) to real-time object detection, often referred to as the "YOLO Era". Single-stage detectors such as YOLO (You Only Look Once) became the industry standard for field scouting, enabling

the localization of specific pests or lesions in milliseconds [14]. However, these systems operate on a "closed-set" assumption: they can only detect the specific classes they were trained on. If a supervised model encounters a novel pest or an undocumented disease variant, it will either miss it entirely or confidently misclassify it, highlighting a critical limitation in open-field adaptability.

3.1.3 The Emergence of Foundation Models and Agentic AI

The current frontier addresses the limitations of closed-set supervision through **Foundation Models**. Unlike their predecessors, models like CLIP [25] and Grounding DINO [19] are trained on massive, internet-scale datasets of image-text pairs. This enables "Zero-Shot" detection, where a system can identify a pest it has never seen before, simply by understanding its textual description.

Furthermore, the integration of Large Language Models (LLMs) has given rise to **Agentic AI**. Rather than acting as passive detectors, these systems function as autonomous agents capable of "Active Perception"-observing a symptom, verifying it, and retrieving external knowledge (RAG) to form a diagnosis. This shift moves the goalpost from simple detection accuracy to holistic, actionable decision support.

3.2 Problem Definitions

To properly contextualize the challenges in autonomous crop health management, we formally define the technical limitations inherent in traditional supervised learning and generation tasks. These definitions establish the necessity for the open-set and retrieval-augmented architecture proposed in this thesis.

3.2.1 The Open-Set Recognition Problem

Standard object detection systems operate under a "Closed-Set" assumption. In this paradigm, a model is trained on a dataset $D_{train} = \{(x_i, y_i)\}_{i=1}^{N}$, where x_i is an input image and y_i belongs to a fixed set of predefined classes $C_{train} = \{c_1, c_2, \ldots, c_k\}$. The detection function f is optimized to map an input to one of these known classes:

$$f(x) \rightarrow \hat{y} \in C_{train}$$

The fundamental problem arises during deployment when the system encounters a novel pest or disease instance x_{new} belonging to a class c_{new} such that $c_{new} \notin C_{train}$. A closed-set detector is forced to map x_{new} to the closest existing class in C_{train}, resulting in a confident error or false positive.

Open-Set Recognition, therefore, requires a model capable of zero-shot transfer. This is defined as learning a function $f(x, t) \rightarrow \hat{y}$, where t is a natural language description, allowing the identification of classes where $C_{test} \nsubseteq C_{train}$.

3.2.2 The Semantic Gap and Hallucination

While visual detectors identify spatial features (e.g., bounding boxes), they lack the capacity to reason about causal implications. Conversely, Large Language Models (LLMs)

possess general reasoning capabilities but lack visual grounding. We define the **Semantic Gap** as the disconnect between the visual feature representation V (e.g., pixel patterns of a yellow leaf) and the domain-specific knowledge K required to diagnose the cause (e.g., distinguishing between nitrogen deficiency and early blight).

Relying solely on generative models to bridge this gap introduces the problem of **Hallucination**. Given a user query Q and a visual context I, a standard LLM generates a response $\hat{y}$ by maximizing the probability of the next token based on training correlations rather than factual correctness:

$$\hat{y} = \arg\max_{y} P(y|I, Q)$$

Without external grounding, the model may generate plausible but factually incorrect agronomic advice. This necessitates a retrieval mechanism R to condition the generation on verified documents D, altering the objective to maximize $P(y|I, Q, R(Q))$, ensuring the output is grounded in retrieved evidence rather than parametric memory alone.

3.3 Review of Single-Modality Approaches

Most research in agricultural AI focuses on either only images or only text. While these models work well for specific tasks, they have several problems when used for automatic crop management in real fields.

3.3.1 Limitations of Vision-Only Supervised Detectors

Supervised deep learning models, particularly CNNs and the YOLO family, have been the cornerstone of plant disease detection [11, 8]. However, these models exhibit significant technical drawbacks when deployed outside of curated datasets:

- **Data Bias and Overfitting:** Mohanty et al. [22] demonstrated high accuracy on the PlantVillage dataset, but subsequent studies by Barbedo [2] found that these models rely on "shallow" features like leaf shape or background color rather than the actual pathology. When the background shifts from a laboratory setting to a complex field, performance degrades significantly.
- **The Feature Dependency Gap:** In supervised learning, a classifier $f(x)$ learns a mapping from high-dimensional pixels to a class index $y \in \{0, \ldots, N\}$. Because the model lacks a semantic understanding of the label, it cannot distinguish between a "Bacterial Spot" and a visually similar artifact (e.g., a mud splash or shadow) unless it was explicitly trained on those "negative" examples.
- **High Cost of Annotation:** Annotation engineering is labor-intensive. Manually labeling thousands of bounding boxes for every new pest strain is a non-scalable process for global agronomy.

3.3.2 Limitations of Text-Only Large Language Models

While LLMs have shown sophisticated reasoning for agronomic decision support, they operate in a state of "Visual Blindness." The technical limitations include:

- **Subjectivity in Symptom Description:** Without a vision-language alignment, the system relies on human-provided text descriptions. As noted in recent evaluations

[12], this introduces a "Semantic Gap" where the user's subjective description of a leaf ("yellow spots") may not align with the scientific definition required for an accurate diagnosis.

- **Lack of Spatial Grounding:** Text-only models cannot localize an infection. In a field scenario, knowing *what* a disease is is insufficient without knowing *where* it is and its *severity density*. This lack of spatial context prevents LLMs from being used for localized spray planning or robotic intervention.
- **Hallucination Risk:** LLMs generate responses based on token probabilities $P(y|Q)$ rather than physical evidence. In agricultural contexts, a "plausible-sounding" but incorrect pesticide recommendation can lead to crop death or environmental damage.

3.4 Review of Multi-Modal and Foundation Models

The current State-of-the-Art (SOTA) in agricultural perception is represented by the move toward multimodal foundation models. A technical overview of these models' adaptation to the agricultural domain is given in this section, with particular attention to grounded reasoning, open-set detection, and semantic verification.

3.4.1 Open-Set Object Detection (Grounding DINO)

Grounding DINO [19] has recently emerged as a solution for agricultural tasks where traditional closed-set annotations are difficult to acquire. Recent studies have explored its application in diverse field scenarios:

- **Few-Shot Adaptation:** Research by researchers on agricultural datasets (e.g., GWHD, PhenoBench) demonstrates that Grounding DINO can achieve up to 24% higher mAP than fully fine-tuned YOLO models under few-shot conditions [1]. This proves its ability to localize complex features like wheat heads and insect pests with minimal training data.
- **Livestock Monitoring:** In cattle management, Grounding DINO has been utilized for muzzle detection without any task-specific training, achieving a mean Average Precision (mAP@0.5) of 76.8%. This highlights the model's robustness in varied environmental backgrounds.
- **Auto-Labeling Teacher Models:** Some frameworks employ Grounding DINO to automatically generate labels for smaller, real-time models (like YOLO), significantly reducing the manual annotation burden in agricultural computer vision.

3.4.2 Contrastive Vision-Language Verification (CLIP)

To address the precision-recall trade-off in open-set detection, researchers are increasingly utilizing CLIP [25] as a semantic filter.

- **Multimodal Disease Classification:** Studies utilizing the PlantVillage and FieldPlant datasets have demonstrated that augmenting visual models with CLIP's textual annotations leads to notable gains in F1-score over vision-only baselines [23]. These systems use textual descriptions of symptoms (e.g., "irregular brown lesions with yellow halos") to provide a more sophisticated interpretation of the visual data.

- **Few-Shot Frameworks (VLCD):** The Vision-Language model for Crop Disease (VLCD) framework integrates CLIP to improve few-shot classification accuracy. By using a "Cache Model" to blend CLIP's prior knowledge with task-specific training features, researchers have achieved high accuracy in identifying crop leaf diseases with very few samples.

3.4.3 Retrieval-Augmented Generation (RAG)

The most recent evolution in agricultural decision support is the deployment of Agentic RAG systems to mitigate the hallucination risks of standard LLMs.

- **AgroLLM and Expert Chatbots:** Frameworks like AgroLLM [7] deliver contextually relevant responses derived from a comprehensive corpus of agricultural textbooks and scientific articles. This "non-parametric" grounding ensures that the advice provided is factually accurate and geographically relevant.
- **Q&A Systems for Pest Management:** Recent Q&A systems developed for crop pests use adaptive retrieval to handle diverse farmer queries. By combining knowledge graphs with RAG, these systems can refine search queries to retrieve specific pesticide protocols and outbreak patterns, significantly reducing the probability of erroneous diagnostic advice.

3.5 Research Gap

The primary research gap identified in this study is the **lack of an orchestrated agentic workflow** that unifies perception, verification, and reasoning. Specifically:

1. **Absence of Semantic Verification:** While open-set detectors like Grounding DINO offer high recall, there is a lack of research into using contrastive models (CLIP) as a real-time Verifier to suppress the false positives inherent in unstructured field data.
2. **The Perception-Reasoning Disconnect:** There is a distinct absence of Agentic RAG implementations where a vision system's output directly initializes a scientific retrieval process without human intervention.
3. **Lack of Actionable Autonomy:** Most current models provide a *label* (e.g., "Tomato Leaf Mold"), but they do not provide a *verified treatment plan* grounded in real-time localized knowledge.

To bridge these gaps, this thesis proposes an "Autonomous Multimodal Agronomy Agent". By moving away from the individual assumption of single-task models, this work establishes a unified loop of *Observe* $\rightarrow$ *Verify* $\rightarrow$ *Reason* $\rightarrow$ *Act*. This architectural method is essential for transitioning from simple disease detection to autonomous crop health management.

Chapter 4

Methodology

4.1 Conceptual Agent Architecture

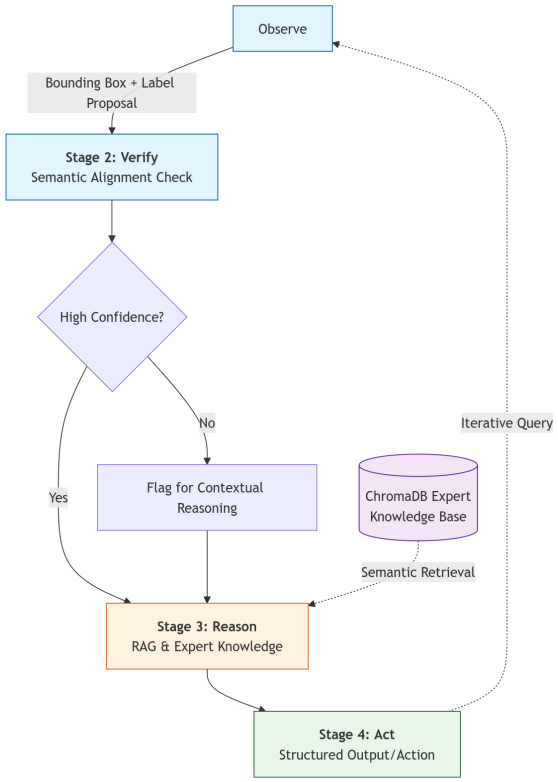

Figure 4.1: Conceptual Agent Architecture

The diagram 4.1 illustrates the logical flow and the decoupling of the vision, knowledge, and orchestration layers. The agent operates in a four-stage cycle that separates finding an object from identifying its condition:

1. **Observe (Localization):** The agent uses Grounding DINO[19] to scan the image for general agricultural objects. It uses simple prompts like "leaf", "bug", or "worm". The goal here is just to find the location and create a bounding box around the object.
2. **Verify (Fine-Grained Classification):** Once a box is found, the agent crops that image and sends it to the fine-tuned CLIP model. This model checks for specific

diseases or pests, such as "Durian leaf with Algal Leaf Spot" or "Grasshopper". This acts as a verification layer to ensure the diagnosis is precise.

3. **Reason (Knowledge Retrieval):** After the disease or pest is identified, the agent queries the RAG system. It retrieves expert advice from the database about that specific condition.
4. **Act (Response Generation):** The agent combines the visual findings and the retrieved expert knowledge to create a final report.

State Management

To make the agent understand the whole context, it needs to remember its progress. during a diagnostic session. I implemented a state management process that included these functions:

- Tracking Objects: The agent keeps track of every leaf or bug it has already checked. This prevents it from wasting time analyzing the same spot twice.
- Handling Unclear Results: If the CLIP model is not sure about a disease (low similarity score), the agent's memory records this uncertainty. It then uses the Reasoning stage to look for "look-alike" symptoms in the knowledge base instead of just guessing.
- System Flexibility: Because the architecture is modular, I use a general detector (Grounding DINO) to find anything and a specialized classifier (Fine-tuned CLIP) to identify specifics. This makes it easy to add new plants or pests to the system later just by updating the CLIP prompts or the RAG database.

4.2 Perception & Verification

The perception layer is responsible for turning raw images into useful data. This process is split into two simple parts: first, finding where the objects are (Localization), and second, identifying exactly what they are (Verification).

4.2.1 Open-Set Detection Strategy (Grounding DINO)

The first part of the vision system uses Grounding DINO[19] to find objects in the image. This is an open-set detector, meaning it can find things based on simple text descriptions without needing to be retrained for every new plant or insect.

- **Model Backbone:** This implementation employs the Swin-B (Base) backbone. Compared to the smaller Swin-T variant, the Swin-B architecture provides a richer feature representation, which helps the agent better identify agricultural objects in complex, high-entropy backgrounds.
- **Prompting Strategy:** The agent is configured with a general prompt: *"leaf . bug . worm"*. This strategy enables the model to scan the entire plant structure simultaneously, identifying both the leaves and pests.
- **Detection Hyperparameters:** To maximize recall while maintaining precision, the following thresholds are applied:
  - **Box Threshold ($\tau_{box} = 0.3$):** Filters bounding boxes to ensure the agent only processes if it is at least 30% sure an object is there.
  - **Text Threshold ($\tau_{text} = 0.25$):** Ensures the detected visual features align with the semantic tokens in the input prompt.

- **Cropping:** Grounding DINO generates coordinates for each detection. These regions will be cropped and normalized to be passed into the subsequent verification layer.

4.2.2 Semantic Verification Layer (CLIP)

After the objects are found, the agent must check their health or identify the species. This stage uses two different versions of the CLIP model to ensure the results are accurate.

- **Checking Leaves (Disease Identification):** If the object is a leaf, it is sent to a fine-tuned CLIP model (`keetawan/clip-vit-large-patch14-plant-disease` [16]). This model has been specifically trained to recognize plant diseases, such as *"Durian leaf with Algal Leaf Spot"* or *"Healthy Tomato leaf"*.
- **Checking Pests (Bug Identification):** If the object is a bug or a worm, it is sent to the standard CLIP model (`openai/clip-vit-base-patch16`). This version is better at general identification and helps identify insects like a *"Ladybug"* or *"Grasshopper"* without getting confused by leaf features.

4.2.3 Comparative Analysis of Perception Stages

The following table summarizes the functional separation between the localization and verification stages within the agentic loop.

Table 4.1: Comparison of Detection vs. Verification Stages

Feature	**Grounding DINO (Observe)**	**CLIP (Verify)**
Goal	Spatial Localization (Bounding Boxes)	Semantic Classification (Health/Species)
Input	General Categories (e.g., *leaf, bug*)	Specific Labels (e.g., *Late Blight, Black rot*)
Architecture	Open-set Detector (Swin-B)	Multimodal Classifier (Finetuned/Base)
Output	Region Coordinates	Confidence Scores per Class

4.3 Knowledge Retrieval (RAG)

After a specific disease or pest is identified, the agent must provide actionable advice. This is achieved through a Retrieval-Augmented Generation (RAG) system, which allows the agent to look up expert facts from a reliable external source.

4.3.1 Knowledge Base Construction

The RAG system relies on a curated library of agricultural information. This ensures that the agent's recommendations are grounded in expert knowledge rather than just statistical guesses.

- **Data Collection:** The knowledge base includes data on symptoms, causes, and treatments for various crops, with a specific focus on the Tomato varieties found in the PlantVillage dataset.
- **Chunking and Embedding:** Large expert documents are divided into smaller, manageable text segments. These segments are converted into numerical vectors using an embedding model, which represents the semantic meaning of the agricultural advice.

4.3.2 Semantic Search and Indexing (ChromaDB)

To retrieve the correct information during the diagnostic loop, the agent uses **ChromaDB** as its vector database.

- **Vector Storage:** All embedded knowledge chunks are indexed within ChromaDB. This allows for high-speed searching during the "Reason" stage of the agentic loop.
- **Similarity Search:** When the CLIP model identifies a condition (e.g., *"Tomato Late Blight"*), the agent generates a query vector. ChromaDB then performs a similarity search to find the most relevant treatment protocols.
- **Contextual Filtering:** The search is restricted based on the current context (such as the plant species). This ensures that the retrieved information is specific to the user's current problem and ignores irrelevant data.

4.4 Orchestration and Logic

The orchestration layer serves as the central control unit of the agent. It manages the flow of information between the vision models and the knowledge base, ensuring the agent follows a logical decision-making path.

By using Pydantic schemas, the architecture ensures that all data passed between tools (such as bounding box coordinates or classification labels) is strictly formatted. This prevents errors that often occur when passing unstructured text between different AI models.

Decision Routing Logic

The agent uses a routing mechanism to control the execution of tasks. This allows the system to be autonomous and adaptive.

- Conditional Routing: The system can change its behavior based on findings. For example, if no pests or diseases are found during the verification stage, the agent is programmed to terminate the loop early and report a "Healthy" status, saving computational resources.
- Result Synthesis: The orchestration layer combines the visual evidence from CLIP with the textual advices from ChromaDB to generate a single, structured report for the end-user.

Chapter 5

Prototype Development

5.1 Prototype

The framework is made publicly available on https://github.com/dirtyg2120/multimodal-crop-agent.

The website's homepage is very simple, just a place for user to upload an image.

Figure 5.1: Main page

The main page after user upload an image. There are 3 custom fields: "caption, tex_threshold and box_threshold" to custom the parameters for running Grouding DINO.

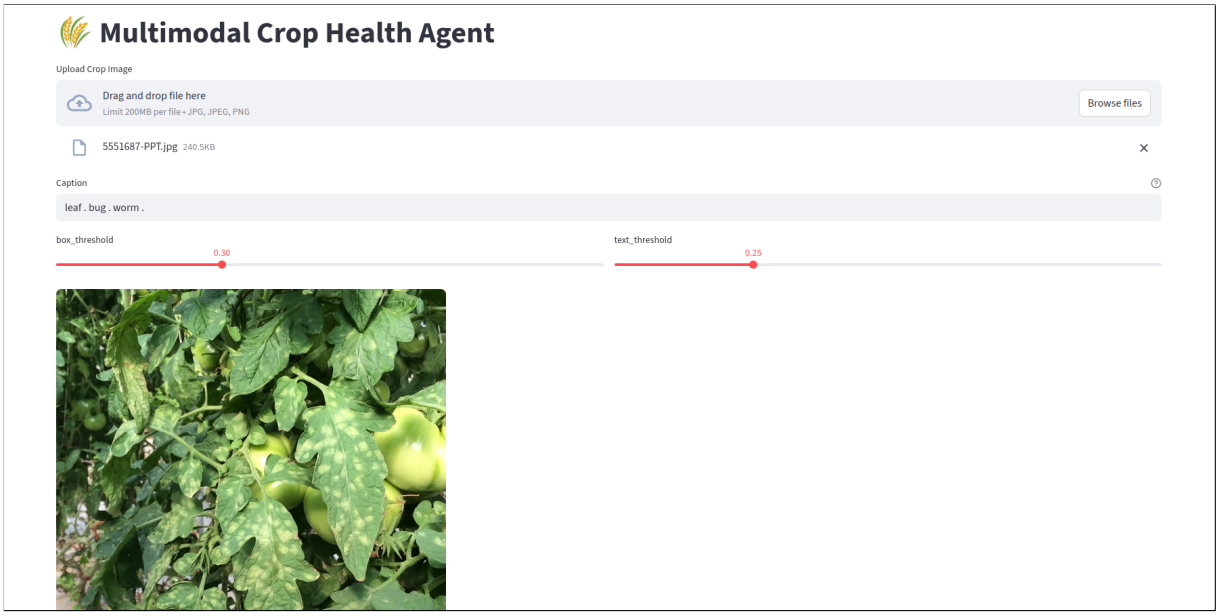

Figure 5.2: Main page after uploading image

After user click "Analyze Image", the model will be applied and return labels with bounding box as shown.

Figure 5.3: Visualize the output result from running Grouding DINO

After Grounding DINO detect all the objects, the system will crop all the bounding boxes and send to CLIP verifier.

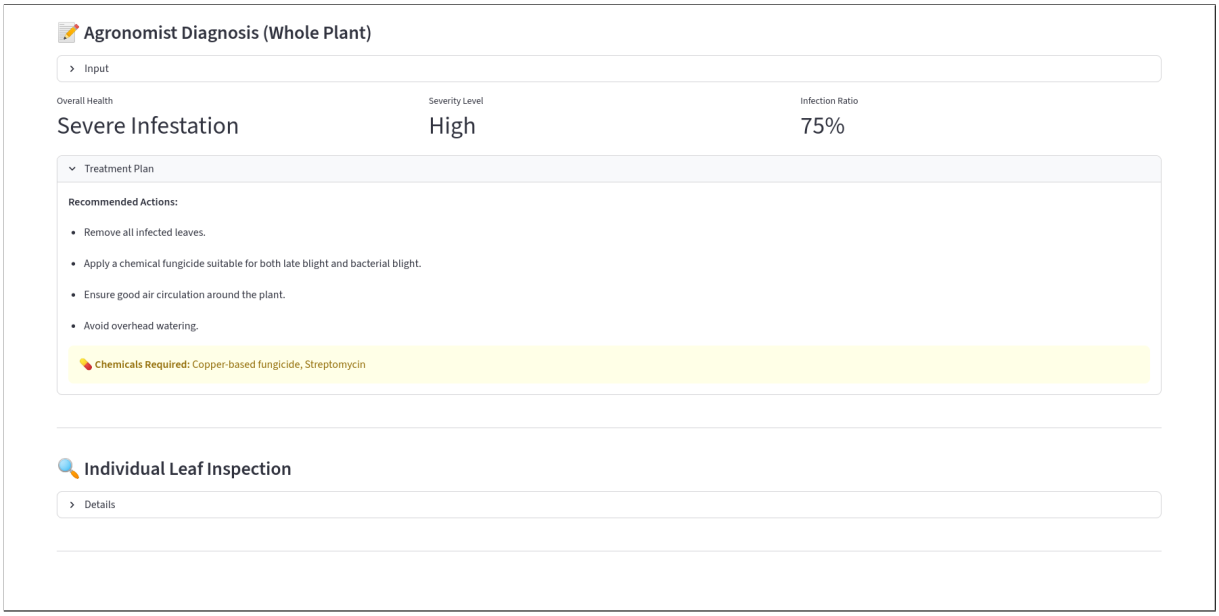

Figure 5.4: Result show after all steps (Run in the background)

If user click on the "Details" expander of "Invidual Leaf Inspection" part. It will display all cropped images and their labels

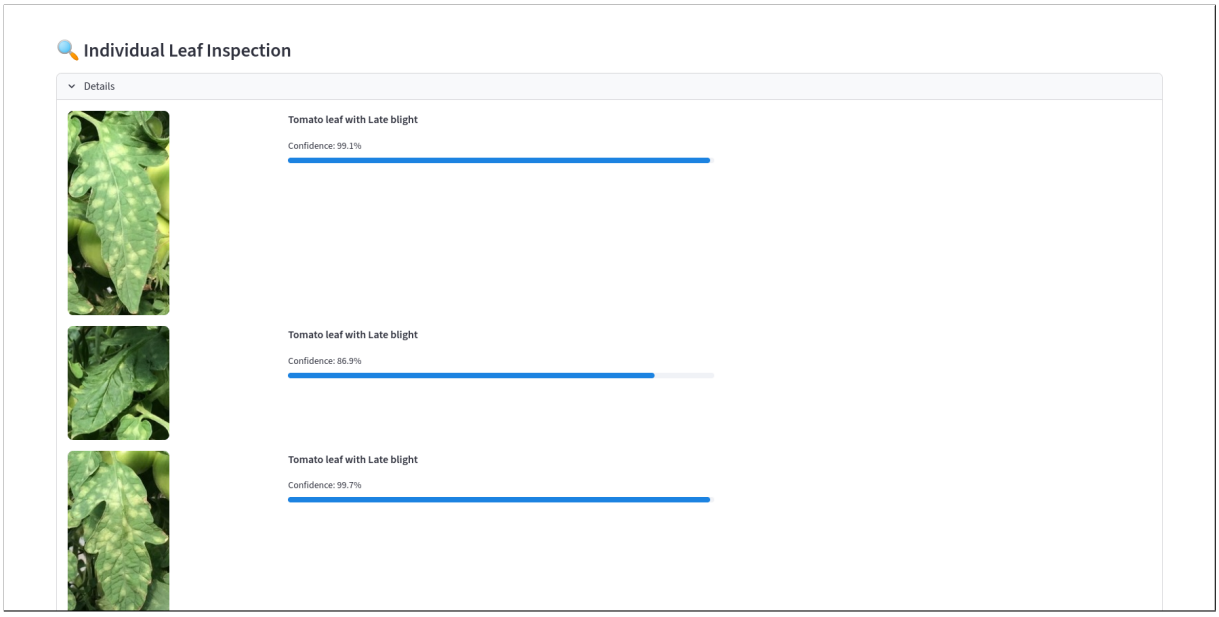

Figure 5.5: Additional pages to see the label of each object

If user click on the "Input" expander of "Agronomist Diagnosis" part. It will display the input which will be send to AI Agent, and the part of retrieving RAG is run in the background.

Figure 5.6: Additional pages to see the input for AI Agent

5.2 Technologies

5.2.1 PyTorch

PyTorch is a popular open-source machine learning library primarily used for computer vision and natural language processing. In this project, PyTorch acts as the engine that runs our vision-related models. I use it to load and execute pre-trained models from the Hugging Face library, such as Grounding DINO for detecting objects and CLIP for verifying them.

The main reason for choosing PyTorch is its flexibility and its ability to perform high-speed calculations on a GPU. This is essential for our agent because processing high-resolution images of crops requires significant computing power. By using PyTorch, the agent can analyze images quickly. It also provides a wide range of tools for handling image data, making it easier to integrate the vision system with the rest of the software stack.

5.2.2 Pydantic AI

Pydantic AI is a specialized framework designed to manage how AI agents behave and make decisions. It serves as the "brain"that controls the flow of our "Observe-Verify-Reason-Act"loop. While many other tools focus only on sending prompts to a model, Pydantic AI ensures that every piece of information moving through the system follows a strict format.

I use this framework to define exactly what the input and output should look like for each step. For example, when the vision module identifies a disease, Pydantic AI checks that the data is sent in the correct structure (such as a specific list of names and confidence scores) before passing it to the reasoning module. This strict data validation prevents the agent from making errors caused by messy or unexpected information. It makes the system more reliable because the AI is forced to follow a predictable path,

reducing the chance of "hallucinations or logical mistakes.

5.2.3 LlamaIndex

LlamaIndex is a data framework used to connect our Large Language Model (LLM) with external data sources. This process is known as Retrieval-Augmented Generation (RAG). In this project, the agent needs more than just general knowledge; it needs specific, expert-level information about tomato diseases, pests, and treatment protocols.

LlamaIndex allows us to take large documents, such as agricultural textbooks and diagnostic guides, and turn them into a format the agent can easily search. When the agent identifies a specific problem on a plant, LlamaIndex finds the most relevant sections of text from our knowledge base and provides them to the LLM. This ensures that the advice the agent gives to a farmer is grounded in real agricultural science rather than just a guess. It handles the difficult task of indexing and retrieving text so that the agent always has the right context at the right time.

5.2.4 ChromaDB

ChromaDB is a vector database that works alongside LlamaIndex to store and retrieve information based on its meaning. Traditional databases search for exact words, but ChromaDB stores data as embeddings, which are mathematical representations of the meaning of the text. This allows the agent to perform a "semantic search".

For instance, if the agent detects "early blight" but the documentation uses slightly different terms, ChromaDB can still find the correct information because it understands that the topics are related. I chose ChromaDB because it is lightweight, fast, and easy to set up within a Python environment. It stores all our embedded agricultural data in an organized way, allowing the agent to find the correct treatment protocols in milliseconds. This speed is vital for maintaining a responsive system when the agent has to look through thousands of pages of expert documentation.

5.2.5 Streamlit

Streamlit is an open-source framework used to create the web-based user interface for our application. It allows us to build a professional-looking dashboard using only Python, without needing to write complex HTML, CSS, or JavaScript code.

Through the Streamlit interface, a user can upload a photo of a crop and see the agent's work in progress. The dashboard displays the original image, marks the areas where problems were detected, and shows the final diagnostic report. I chose Streamlit because it is perfect for data-heavy applications. It allows us to visualize the agent's reasoning process in a way that is easy for a human to understand. This transparency is important for building trust with users, as they can see exactly why the agent is making a specific recommendation.

Chapter 6

Evaluation Plan and Expected Contributions

6.1 Proposed Evaluation Metrics

The system will be evaluated using three primary metrics to ensure each part of the loop is performing correctly:

- **Mean Average Precision:** Used to assess the localization accuracy of Grounding DINO in identifying leaves, bugs, and worms.
- **Accuracy:** Used to measure the performance of the dual-CLIP verification layer in identifying specific pathologies.
- **Retrieval Relevance:** Evaluates the RAG system's ability to provide the correct expert treatment advice from ChromaDB based on the identified condition.
- **Visual Case Analysis:** This analysis focuses on the agent's robustness in many other images from the Internet with complex context, ensuring the reasoning and recommendation still perform well.

6.2 Expected Contributions

This thesis aims to contribute to the field of Agentic AI in agriculture by:

- Providing a solid "Observe-Verify-Reason-Act" framework that bridges the gap between raw vision and expert reasoning.
- Demonstrating the use of Open-Set Detection to handle the diverse and changing nature of farm environments without constant retraining.
- Developing a method for knowledge grounding that ensures AI recommendations are based on verified agricultural facts.

The primary contribution of this research is the development of a stateful agentic loop that integrates open-set detection with expert knowledge. By moving away from static pipelines, this work demonstrates a more flexible and reliable approach to precision agriculture.

6.3 Future Work

Future developments for this project include:

- Image Quality Check: Adding a layer to detect motion blur, poor lighting, or extreme occlusion. If the quality is too low, the agent will request a new image rather than risk a wrong diagnosis.
- SAHI Integration: Combining Slicing Aided Hyper Inference (SAHI) with these quality checks to help Grounding DINO handle high-resolution images, ensuring small pests are not missed due to image downscaling.
- Stateful Memory: The system will save every detection result (with timestamps) into a long-term database. The agent will be able to query this history to identify patterns.
- Specialist Model Fallback: Implementing a multi-tier verification system for high-uncertainty cases.

Bibliography

[1] Anonymous Authors. Few-shot adaptation of grounding dino for agricultural domain. *arXiv preprint*, 2024.

[2] Jayme Garcia Arnal Barbedo. Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification. *Computers and electronics in agriculture*, 153:46–53, 2018.

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *CoRR*, abs/2005.12872, 2020.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

[5] Akshay Dhamija, Manuel Günther, Jonathan Ventura, and Terrance E. Boult. The overlooked elephant of object detection: Open set. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1021–1030, 2020.

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021. arXiv:2010.11929.

[7] M. Samuel et al. Agrollm: Connecting farmers and agricultural practices through large language models. *ResearchGate*, 2024.

[8] Konstantinos P Ferentinos. Deep learning models for plant disease detection and diagnosis. *Computers and Electronics in Agriculture*, 145:311–318, 2018.

[9] Alvaro Fuentes, Sook Yoon, Taehyun Kim, and Dong Sun Park. Open set self and across domain adaptation for tomato disease recognition with deep learning techniques. *Frontiers in Plant Science*, 12:758027, 2021.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. IEEE, 2016.

[11] Andreas Kamilaris and Francesc X. Prenafeta-Boldú. Deep learning in agriculture: A survey. *Computers and Electronics in Agriculture*, 147:70–90, 2018.

[12] Enkelejda Kasneci et al. Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and Individual Differences*, 103, 2023.

[13] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. *CoRR*, abs/2005.11401, 2020.

[14] Dawei Li, Yuan Shao, et al. A comprehensive review of computer vision in plant phenotyping and agriculture. *Journal of Field Robotics*, 2021.

[15] Konstantinos G. Liakos, Patrizia Busato, Dimitrios Moshou, Simon Pearson, and Dionysis Bochtis. Machine learning in agriculture: A review. *Sensors*, 18(8), 2018.

[16] Keetawan Limaroon. clip-vit-large-patch14-finetuned-disease: A fine-tuned model for plant disease classification and captioning, 2024.

[17] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[18] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*, volume 8693 of *Lecture Notes in Computer Science*, pages 740–755. Springer, 2014.

[19] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023.

[20] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10012–10022, 2021.

[21] Yury A. Malkov and Dmitry A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *CoRR*, abs/1603.09320, 2016.

[22] Sharada P Mohanty, David P Hughes, and Marcel Salathé. Using deep learning for image-based plant disease detection. *Frontiers in plant science*, 7:1419, 2016.

[23] K. Oladele. Using natural language processing to enhance visual models for plant leaf diseases classification. *Thesis, AAU*, 2024.

[24] R Pydipati, TF Burks, and WS Lee. Identification of citrus disease using color texture features and discriminant analysis. *Computers and electronics in agriculture*, 52(1-2):49–59, 2006.

[25] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. *International conference on machine learning*, pages 8748–8763, 2021.

[26] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 2021.

[27] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788. IEEE, 2016.

[28] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster r-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

[29] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 658–666, 2019.

[30] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[31] Yefeng Shen, Md Zakir Hossain, Khandaker Asif Ahmed, and Shafin Rahman. An open set model for pest identification. *Computational Biology and Chemistry*, 108:108002, 2024.

[32] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

[33] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 2019.

[34] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *CoRR*, abs/2201.11903, 2022.

[35] Chaoyang Zhu and Long Chen. A survey on open-vocabulary detection and segmentation: Past, present, and future. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(15):8954–8975, 2024.