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

Graduation Thesis Proposal - Semester 211

Development of a Feature-Based Social Bot Detection Tool

Instructor:	Assoc. Prof. Nguyen Manh Hung	
Students:	Tran Quoc Anh	– 2370386

Ho Chi Minh City, December 2025

Declaration of Authenticity

We declare that this research is our own work, conducted under the supervision and guidance of Assoc. Prof. Nguyen Manh 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 ourself 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

Acknowledgment

We would like to acknowledge and express our gratitude to everyone who have supported us during our university life. Not only the professors in Ho Chi Minh City University of Technology who have taught us immense knowledge and given us their aspiring guidance, invaluable constructive criticism and friendly advice, but also our friends and our family members of each member in our team who let us have a good physical and mental health during our study.

Foremost, we would like to give our sincere gratitude to our supervisor, Associate Professor Ph.D Nguyen Manh Hung as well as our mentors for their continuous support. Their guidance has brought up our motivation and enthusiasm in research for this thesis. We could not have imagined having a better support for our study.

Last but not least, we would like to thank Ho Chi Minh City University of Technology for giving us the opportunity to learn great lessons of theory and practical experience in the field of Computer Science.

Abstract

Online social networks, such as Twitter and Facebook, have become an important part of people's daily lives in recent years since they allow them to connect with others, communicate with friends, share personal content with others, and obtain information, especially during the ongoing worldwide disaster—the COVID-19 pandemic. However, it also creates opportunities for social bots that are designed to replicate the behaviors of normal human accounts. Most of these bots are used for nefarious purposes such as disseminating false information, artificially amplifying the popularity of a person or movement or spreading spam. This proposal presents a review of some techniques that have emerged and designed to differentiate between social bot accounts and human accounts. We restrict the analysis to the detection of social bots on the Twitter social media platform for the time being, with plans to expand to Facebook later. We also compare the experiments of each technique on two different data sets to point out the most accurate one. Besides experimenting, our final destination is to develop a social bot detection tool that can be deployed as a web application. Finally, we highlight the challenges that remain in the domain of social bots detection and consider future directions for research efforts that are designed to address this problem.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

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[21], and EfficientNet[33], consistently achieved strong performance on large-scale benchmarks such as ImageNet[30] and COCO[24]. Besides that, CNN-based detectors like YOLO[28] 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)[19] 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[29] 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)[24]. 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 (i) accurately classify instances from $\mathcal{Y}_{\text{known}}$, while (ii) acknowledging inputs from $\mathcal{Y}_{\text{unknown}}$ without overconfident misclassification.[18] 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.[27, 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.[20] 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 operationalizing crop health monitoring as a *text-conditioned, open-set* problem. 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[25] 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) [17] 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.

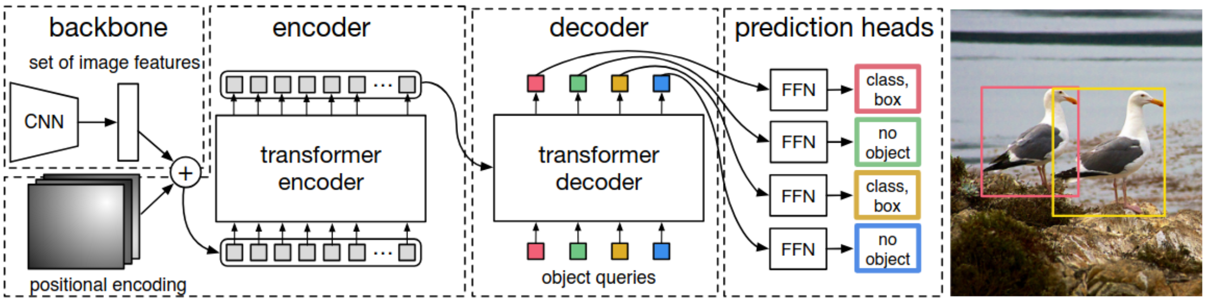

Figure 2.1: DETR architecture [17].

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

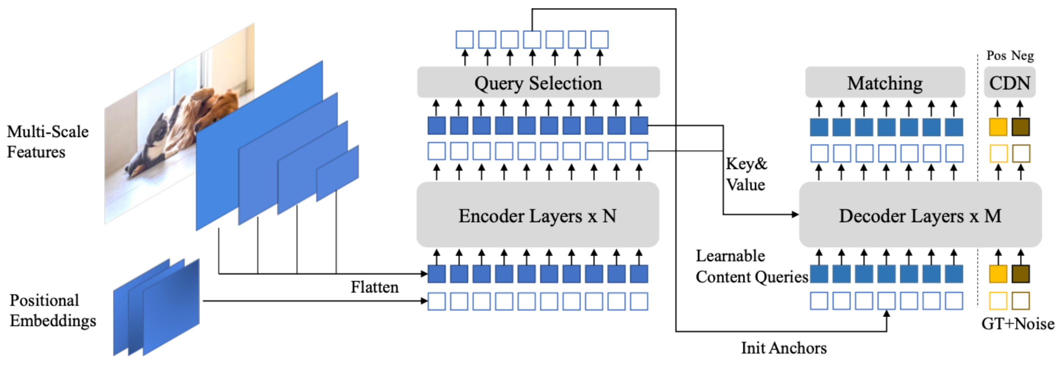

Figure 2.2: DINO Architecture (Source: Internet).

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 **??**, 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.

Feature Extraction and Encoding

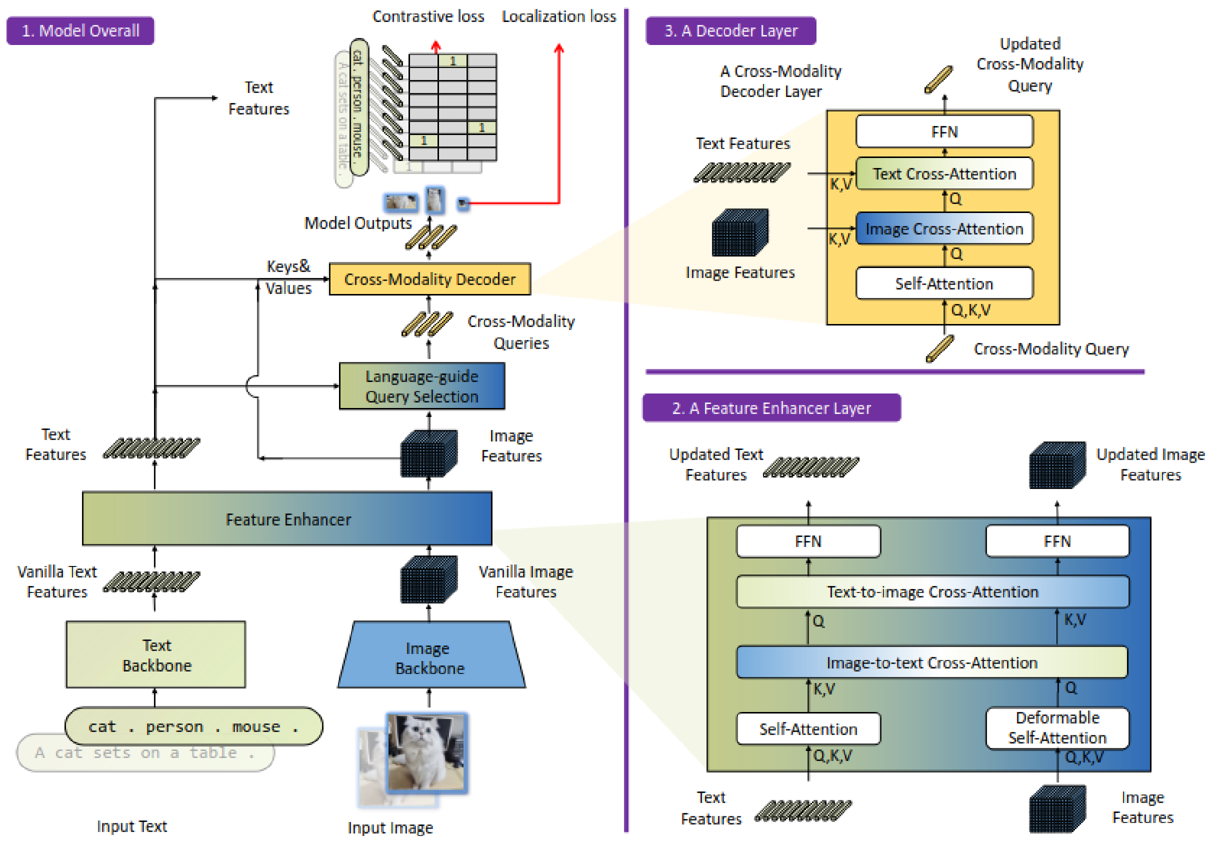

Figure 2.3: The framework of Grounding DINO [25].

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

- **Image Backbone:** A Swin Transformer [?] 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 [?]. 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.

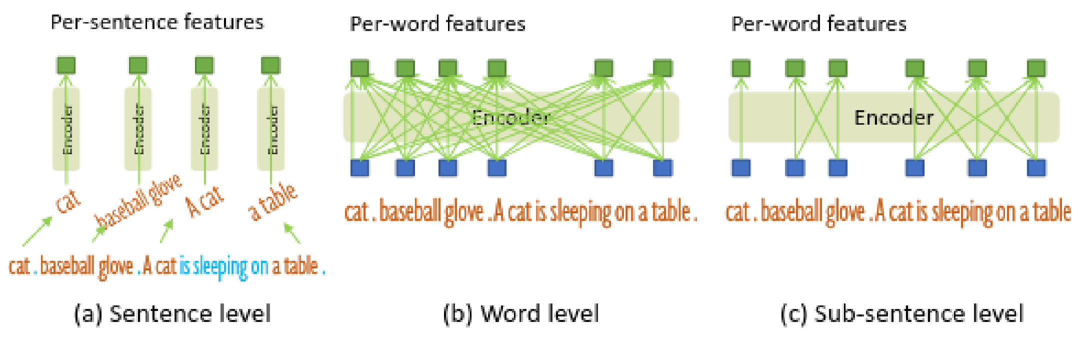

Figure 2.4: Comparisons of text representations [25].

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 [?]. Bounding box regression is supervised using L1 loss and Generalized IoU (GIoU) loss [?].

2.3 CLIP

CLIP 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 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 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 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:

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

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 language descriptions.

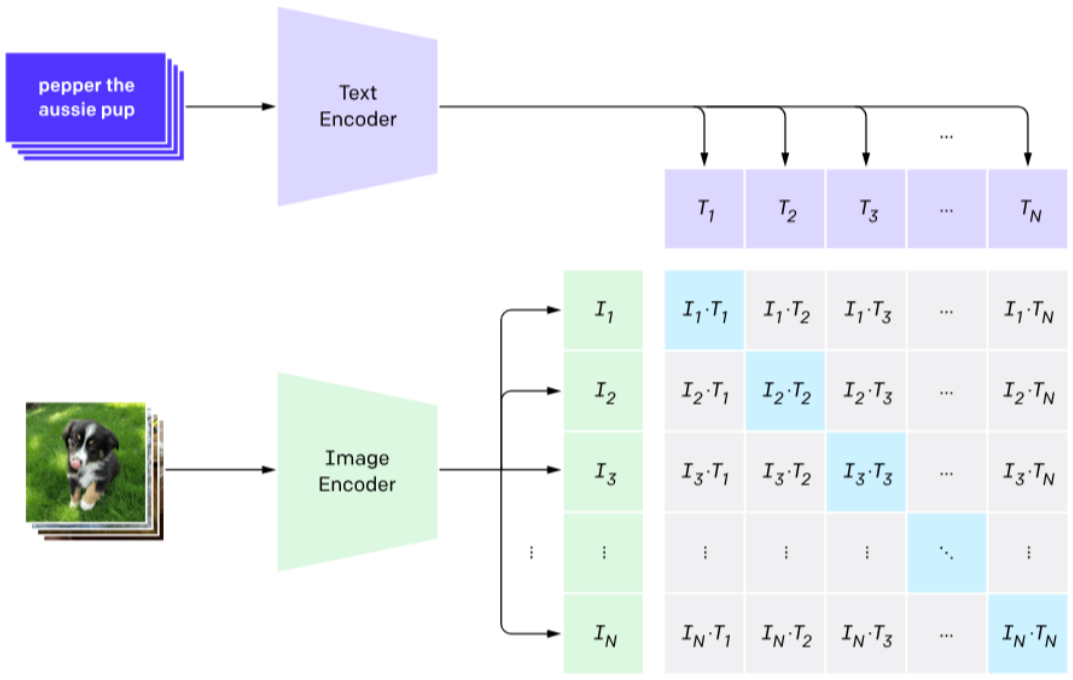
pepper the aussie pup
Text Encoder
Image Encoder
T_1 T_2 T_3 ... T_N
I_1 I_2 I_3 ⋮ I_N
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
$I_2 \cdot T_1$ $I_2 \cdot T_2$ $I_2 \cdot T_3$ ... $I_2 \cdot T_N$
$I_3 \cdot T_1$ $I_3 \cdot T_2$ $I_3 \cdot T_3$ ... $I_3 \cdot T_N$
$I_N \cdot T_1$ $I_N \cdot T_2$ $I_N \cdot T_3$ ... $I_N \cdot T_N$

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

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 [23] 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)[26], HNSW organizes data into a multilayered 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

A ReAct agent is an AI agent that uses the "reasoning and acting" (ReAct) framework to combine chain of thought (CoT) reasoning with external tool use. The ReAct framework enhances the ability of a large language model (LLM) to handle complex tasks and decision-making in agentic workflows.

2.5.2 Neuro-Symbolic Integration (Structured Outputs)

2.5.3 Control Flow: Workflows vs. Agents

Chapter 3

Related Works

3.1 The Big Picture of AI in Crop Health Management

Recent surveys show that AI for agriculture has evolved from feature-engineered machine vision and classical machine learning to deep learning perception systems deployed on UAV/edge platforms, and more recently toward foundation-model and LLM-based decision support. Liakos et al. [1] reviewed machine learning applications across agricultural production systems, while Kamilaris and Prenafeta-Boldú [2] summarized early deep learning adoption in agriculture. More recent overviews emphasize practical constraints such as dataset bias, domain shift, and deployment challenges in real farms [?, ?, ?].

From the viewpoint of crop health management (disease/pest/stress monitoring and recommendations), the literature can be grouped into: (i) sensor and remote-sensing driven monitoring, (ii) feature-engineered machine learning pipelines, (iii) supervised deep learning for classification/detection/segmentation (including the "YOLO era" for real-time detection), (iv) open-set/open-world recognition to handle unknown diseases/pests, and (v) knowledge-grounded language/agent systems (e.g., Retrieval-Augmented Generation) for actionable decision support [3, 22, 4, 5].

3.1.1 Sensor and Remote-Sensing Driven Monitoring

A major branch of AI in agriculture builds on field sensing (weather/soil/IoT) and remote sensing (UAV/satellite) to infer crop health indicators (stress, canopy changes, outbreaks) at scale. Drone-based pipelines are often designed around mapping and scouting, combining imagery with geospatial processing and learning models [6]. While these systems can cover large areas, they frequently provide *symptom signals* rather than fine-grained causal explanations, and may struggle to translate detections into agronomically grounded recommendations.

3.1.2 Feature-Engineered Machine Learning Pipelines

Before deep learning dominance, crop disease inspection commonly followed a multi-stage pipeline: preprocessing/segmentation → handcrafted color/texture/shape features (e.g., GLCM/LBP) → classifier (e.g., SVM, RF). For example, Pydipati et al. [7] used color-texture features for citrus disease recognition, and Barbedo [8] surveyed classical digital image processing and ML techniques for detecting and classifying plant diseases from visible symptoms. These approaches can work well in controlled setups but typically require careful feature engineering and are sensitive to illumination, background clutter, cultivar differences, and symptom variability.

3.1.3 Supervised Deep Learning for Disease Recognition

CNN-based learning replaced manual feature design with end-to-end representation learning, achieving strong performance on curated datasets. Mohanty et al. [9] demonstrated high-accuracy plant disease classification using a large leaf image dataset, and Ferentinos [10] explored CNN architectures for plant disease detection/diagnosis at scale. However, multiple works highlight that performance can degrade under real field conditions due to dataset bias and limited variety; Barbedo [11] specifically studied how dataset size/variety affects deep learning effectiveness. This motivates research on robustness, generalization, and handling unseen conditions.

3.1.4 The "YOLO Era": Real-Time Detection and Field Deployment

Beyond image-level classification, practical crop monitoring often requires *localization*: detecting pest insects, lesions, or symptomatic regions in complex scenes. One prominent trend is adopting one-stage detectors for real-time performance (e.g., YOLO variants) on UAV/edge devices. Reviews summarize deep learning pipelines for pest detection in field imagery [12], and recent works report optimized YOLO variants for small-object pest detection in open fields [13]. While these systems enable real-time scouting, they still typically assume a *closed set* of classes: anything novel is forced into a known label or treated as background.

3.2 Problem Definitions

3.3 Review of Single-Modality Approaches

3.4 Review of Multi-Modal Approaches

3.5 Research Gap

Chapter 4

Methodology

4.1 System Overview

The "Observe → Verify → Reason → Act" Loop.
High-level architectural diagram.

4.2 Module A: Adaptive Perception (The Vision Pipeline)

4.3 Module B: Knowledge Retrieval (The RAG Pipeline)

4.4 Module C: Orchestration (The Agent)

Chapter 5

Dataset

5.1 Description

5.2 Data Processing

Chapter 6

Prototype Development

6.1 User Requirements

6.2 System Architecture

6.3 Technologies

6.3.1 Core AI Models

6.3.2 Orchestration & Backend

6.3.3 User Interface

Chapter 7

Experiments and Results

7.1 Experimental Setup

7.2 Perception Benchmarking (Vision)

7.3 Cognition Benchmarking (Logic)

7.4 Discussion

Bibliography

[1]

[2]

[3]

[4]

[5]

[6]

[7]

[8]

[9]

[10]

[11]

[12]

[13]

[14]

[15]

[16]

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] K. J. Joseph, Salman Khan, Fahad Shahbaz Khan, and Vineeth N. Balasubramanian. Towards open world object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5830–5840, 2021.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[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.