TECHNISCHE HOCHSCHULE MITTELHESSEN

ITI Institut für Technik und Informatik

MASTER THESIS

Developing a Domain-Specific Foundation Model with Adapters and Self-Supervised Learning for Efficient Plant Detection in Grassland

A Data-Efficient Approach to Adaptable Plant Detection Models

Mirko Lehn
`mirko.lehn@mni.thm.de`

To obtain the academic degree

Master of Science (M.Sc.)

submitted to the
department Mathematics, Natural Sciences and Computer Science
of the University of Applied Sciences Mittelhessen

July 17, 2025, Giessen

Examiner:	Prof. Dr.-Ing. Seyed Eghbal Ghobadi
Co-Examiner:	Ing. Moritz Schauer, M.Sc.

Declaration of the use of Generative AI

In accordance with the recommendation of the *German Research Foundation* (DFG[1] - Deutsche Forschungsgemeinschaft) and that of the journal *Theoretical Computer Science*[2] I (the author) herby declare the use of generative AI.

During the preparation of this thesis, I used ChatGPT 4 to improve the readability and language as well as the code quality of the corresponding Python scripts. After using the output of ChatGPT 4, I have reviewed and edited the content as needed and take full responsibility for the content of this work.

Declaration of Independence

I declare that I have written this thesis independently and have only used the resources indicated. All passages that are taken from other works, including electronic media, in terms of wording or meaning, have been identified by me as borrowings by stating the source. Borrowings from the Internet are documented by stating the source and the date of access.

Gießen, 17.07.2025	[illegible]
Place, Date	Mirko Lehn

[1] https://www.dfg.de/en/news/news-topics/announcements-proposals/2023/info-wissenschaft-23-72

[2] https://www.sciencedirect.com/journal/theoretical-computer-science/publish/guide-for-authors

Abstract

The rise of foundation models has enabled generalizable solutions across diverse computer vision tasks. However, their performance often declines in specialized domains where data is limited, annotations are sparse, and subtle visual cues dominate. This thesis targets such a scenario: detecting individual plants in grassland fields where low inter-class variance, weak contrast, and small object sizes pose persistent challenges.

Instead of discarding prior work like DETReg, which was criticized for its lack of classification and reliance on generic object priors, this thesis revisits and extends it. Building on the Deformable Detection Transformer (Deformable DETR), the proposed method introduces an unsupervised, domain-aware pretraining pipeline that directly addresses DETReg's limitations. It uses custom pseudo boxes tailored to the target domain and gradually incorporates teacher model predictions via an Exponential Moving Average (EMA)-based training loop. Semantic learning is introduced early through a binary classification signal, and regularization mechanisms are applied to stabilize training and avoid collapse.

While results show only marginal gains when abundant labeled data is available, the benefits become pronounced under few-shot conditions. The proposed pretraining strategy consistently outperforms models initialized from unrelated domains such as COCO. Moreover, it is designed to operate with minimal supervision and for reducing not only labeled data requirements but also the need for large volumes of unlabeled data by leveraging curriculum schedules and early semantic cues. Lightweight adapter modules are optionally integrated to stabilize the training in the early phase and to test additional possibilities for data reduction, but are not decisive for the final performance.

This work presents a scalable and interpretable approach to domain-specific pretraining, aimed at data efficiency and practical applicability in real-world, low-resource scenarios. Applications include ecological monitoring, automated weed control, and other tasks requiring fine-grained recognition in complex natural environments.

Contents

Acronyms

AP	Average Precision
API	Application Programming Interface
AR	Average Recall
BYOL	Bootstrap Your Own Latent
CNN	Convolutional Neural Network
COCO	Common Objects in Context
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
CV	Computer Vision
DETR	Detection Transformer
DDETR	Deformable Detection Transformer
DL	Deep Learning
EMA	Exponential Moving Average of Weights
FFN	Feedforward Neural Network
GRVI	GreenRed Vegetation Index
GPU	Graphics Processing Unit
IoU	Intersection over Union
KIhUG	KI-gestützte hochautomatisierte Unkrautbekämpfung im Grünland
LAB	CIELAB
LLM	Large Language Model
MAE	Masked Autoencoder
ML	Machine Learning

mAP	mean Average Precision
NAS	Network-Attached Storage
NMS	Non-Maximum Suppression
PEFT	Parameter-Efficient Fine-Tuning
SAM	Segment Anything Model
SLURM	Simple Linux Utility for Resource Management
SSL	Self-Supervised Learning
YAML	Yet Another Markup Language
YOLO	You Only Look Once

List of Figures

List of Tables

Listings

1 | Introduction

Object detection is a key task in Computer Vision (CV), with applications in fields such as autonomous driving, medicine, and ecological monitoring. Recent advances have introduced large-scale foundation models capable of generalizing across tasks and domains through zero-shot or few-shot learning. While powerful in broad settings, these models often underperform in specialized domains that require sensitivity to subtle, domain-specific features. One such challenge is plant detection in natural grassland environments, where small object sizes, low contrast, and visual similarity between foreground and background complicate accurate detection. In such cases, generic architectures and large-scale pretraining are often insufficient, and task-specific adaptation becomes essential.

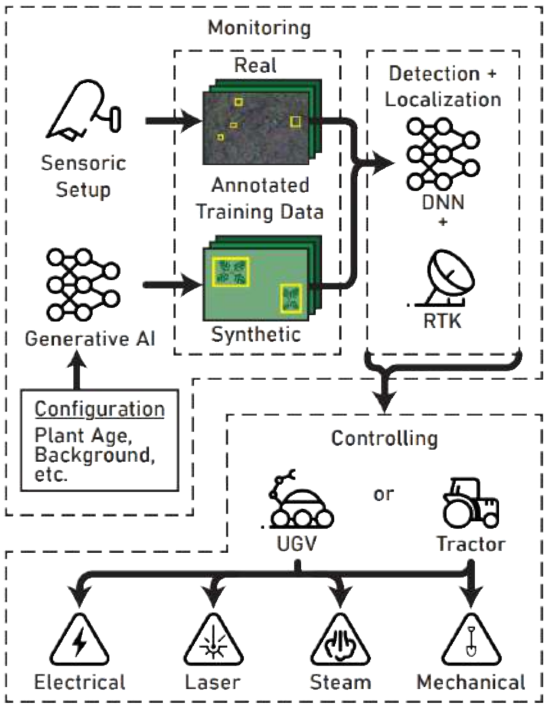
This work introduces a domain-specific pretraining strategy using Self-Supervised Learning (SSL) and a pre-trained Deformable Detection Transformer (DDETR) model [86]. Inspired by DETReg [3], the method uses pseudo-labels generated from unlabeled data to guide object-level learning. Unlike DETReg, which omits classification and relies on general-purpose region proposals, this approach introduces a custom pseudo-labeling pipeline tailored to the domain and leverages confidence-based classification in a single-class setting. To minimize computational cost while maintaining adaptability, lightweight adapter modules are integrated into the model. These allow the backbone to remain frozen while enabling task-specific specialization during pretraining.

The goal of this work is to demonstrate that a targeted, scalable approach can deliver strong results in challenging, data-scarce environments without relying on large general-purpose models that often fail in such cases. The findings contribute to ongoing research in SSL, domain adaptation, and efficient training for specialized object detection.

1.1 Motivation and Problem Statement

This research is conducted within the framework of the KI-gestützte hochautomatisierte Unkrautbekämpfung im Grünland (KIhUG), which aims to develop an AI-assisted robotic system for automated weed control in grasslands. One of the primary objectives of the project is the detection and removal of *Jacobaea vulgaris*, a toxic plant that poses risks to both livestock and humans when ingested [22]. Farmers have a strong incentive to control its spread, as its presence in hay poses both health risks and regulatory concerns.

The envisioned KIhUG system aims to identify and eliminate these plants using non-chemical methods such as mechanical removal or electrocution. However, a major bottleneck in achieving this goal is the requirement for large volumes



The 'Kihug' work pipeline. [63]

The creation of the dataset.

Figure 1.2

of high-quality training data. While precision agriculture has made progress in targeted interventions such as spot spraying, most current detection systems are optimized for "green-on-brown" scenarios, where weeds are clearly distinguishable against bare soil. In contrast, detecting *J. vulgaris* in dense, natural grasslands presents a far more challenging "green-on-green" problem, due to overlapping textures, similar color tones, and variations in lighting. [1]

A robust neural network for plant detection requires training on a diverse and accurately annotated dataset to ensure high recall and precision. However, dataset creation is both costly and time-intensive [4]. Labeling thousands of images demands expert supervision, detailed manual work, and significant logistical coordination. The variability in plant appearance across growth stages, weather conditions, and occlusions further complicates annotation. Similar challenges are common in other specialized domains where annotated data is scarce or expensive to obtain. At the time of writing, the KIhUG project focuses on a single target plant. For this species, a data set was compiled with considerable effort, which is used in this work (see Figure 1.3). Should the scope expand to include additional species, entirely new datasets would be needed, requiring further investment of time, resources, and domain knowledge.

These constraints highlight the need for approaches that reduce dependency on fully labeled datasets and minimize the burden of training from scratch, forming the foundation for the research directions in the following chapters.

1.2 Research Objectives and Contributions

The primary objective of this thesis is to explore whether a domain-specific foundation model can be developed with minimal reliance on labeled data, while still enabling effective fine-tuning for downstream tasks under similar constraints. This addresses a common challenge in specialized domains: the lack of large-scale annotated datasets and the absence of pre-trained models tailored to niche applications.

While the detection of *Jacobaea vulgaris* in the KIhUG project serves as the central use case, the broader aim is to develop a generalizable methodology for constructing lightweight, adaptable, domain-specific foundation models. Such models should support related detection tasks with minimal additional data, potentially enabling the training of models for similar weed species or agricultural applications using only small, weakly labeled, or synthetic datasets.

This work is organized around two central research themes, each defined by two guiding research questions. To address these questions, a set of corresponding objectives has been formulated, encompassing the design of training strategies, architectural modifications, and evaluation of model behavior under limited supervision.

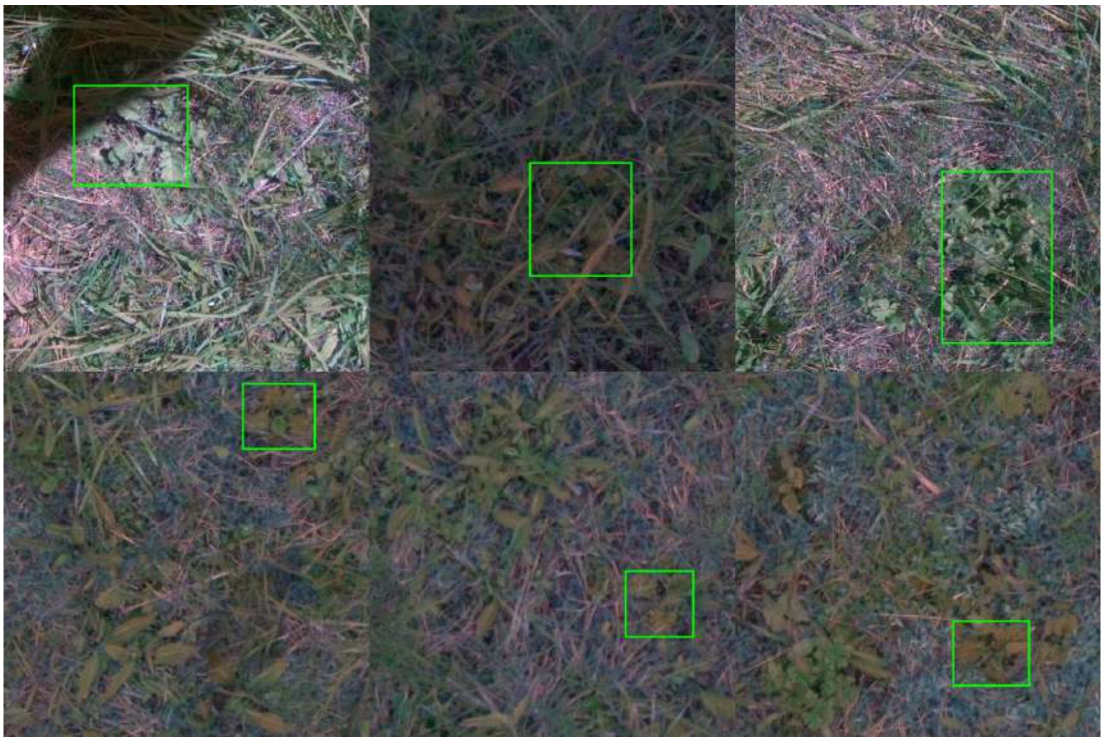


Figure 1.3: Six example image crops from the KIhUG dataset, showing labeled instances of *Jacobaea vulgaris* marked with green bounding boxes.

Research Theme 1: SSL Object Detection on Domain-Specific Data

RQ1: Can a SSL object detection model, trained solely on pseudo-labels extracted from unlabeled grassland images, learn to detect non-grass plant instances without manual annotations?

RQ2: Does domain-specific SSL pretraining improve convergence speed or final detection performance during downstream fine-tuning, compared to using randomly initialized or generically pretrained weights (e.g., Common Objects in Context (COCO) [41])?

Objectives:

- Design a pseudo-labeling pipeline for unlabeled grassland imagery to identify non-grass plant regions with sufficient quality for SSL training.
- Train an object detection model using only pseudo-labels and evaluate its performance in the absence of manual annotations.
- Compare downstream fine-tuning behavior across three initialization settings: domain-specific pretraining (proposed), generic pretraining (e.g., COCO), and random initialization.

Research Theme 2: Parameter-Efficient Fine-Tuning (PEFT)-Based Efficient Learning in Object Detection

RQ3: Can adapter modules [31] integrated into an object detection architecture reduce the number of trainable parameters during SSL training while preserving detection quality?

RQ4: Do adapters pretrained during SSL learning improve sample efficiency or convergence speed when reused in downstream fine-tuning on annotated target datasets?

Objectives:

- Integrate adapter modules into the object detection architecture and develop training routines that update only these modules and selected layers.
- Evaluate detection quality and training efficiency compared to full model fine-tuning.
- Reuse the pretrained adapters during downstream fine-tuning and assess improvements in convergence speed and data efficiency.

These research questions and objectives are addressed through the experimental design and evaluation presented in Chapter 4.

1.3 Thesis Outline

This thesis is structured into eight chapters, each contributing to a cohesive investigation of SSL and parameter-efficient learning techniques for domain-specific object detection in agricultural environments.

Chapter 2 provides the theoretical foundation, presenting an overview of the evolution and limitations of foundation models, with particular emphasis on their application in specialized domains. It introduces key concepts such as SSL learning and adapter modules, establishing the groundwork for the proposed approach.

Chapter 3 surveys related work in object detection, domain adaptation, and parameter-efficient learning. It critically examines existing methods and highlights their limitations in settings with scarce annotations and limited computational resources, thereby motivating the need for alternative solutions.

Chapter 4 outlines the proposed methodology. Following a formal problem definition, it details the strategy for leveraging pseudo-labels and integrating adapters into a SSL training pipeline. It also justifies the selection of the base architecture and its adaptation to the research goals.

Chapter 5 covers the practical implementation. It describes the architectural modifications required to support adapter modules, the construction of the training pipeline, and the generation and application of pseudo-labels. Additionally, it outlines the software stack and computational infrastructure used in the experiments.

Chapter 6 presents the experimental evaluation. It compares the proposed approach against relevant baselines using metrics such as detection accuracy, convergence speed, and parameter efficiency. The experiments are designed to isolate and assess the contributions of each component, including pseudo-labeling and adapter-based training.

Chapter 7 offers a critical discussion of the findings. It interprets the results in a broader context, reflecting on strengths, limitations, and potential generalization to other domains such as medical imaging or environmental monitoring.

Chapter 8 concludes the thesis by summarizing key contributions and insights. It revisits the research questions, evaluates how they were addressed, and outlines directions for future work, including extensions to multi-class detection and more complex datasets.

2 | Background and Context

Many fields of applied machine learning are highly specialized, often involving data distributions, feature patterns, or operational constraints that differ substantially from common benchmark settings. As a result, domain-specific strategies for model adaptation have long been necessary in areas such as medical imaging, remote sensing, and agricultural analysis.

In recent years, however, research and industry have shifted heavily toward general-purpose foundation models. For example, VisionLLM [77] shows how large language models can function as open-ended decoders for vision tasks such as detection and segmentation using natural language prompts. These models aim to unify learning across tasks and modalities, often relying on massive labeled datasets and compute-intensive training. The current trend favors architectures that generalize broadly with minimal customization, often omitting domain-specific considerations and replacing dataset curation with zero- or few-shot inference.

While highly successful in general contexts, these models face challenges in specialized domains like precision agriculture or ecological monitoring. Here, limited data availability, distinct visual patterns, and task-specific constraints hinder performance. Recent research has attempted to bridge the gap between data efficiency and effectiveness. Few-shot methods such as Meta-DETR [84] and TFA [71] adapt pretrained models to novel classes using limited annotations. Self-supervised approaches like DETReg [3], by contrast, aim to eliminate ground-truth labels altogether using object-level pseudo-labels.

Complementing these research directions, system-level tools are emerging that emphasize low-data or training-free deployment. Models like the Segment Anything Model (SAM) [36] offer prompt-based segmentation across diverse scenes, while platforms like Roboflow [59] and OpenAIs Application Programming Interfaces (APIs) [52] provide modular building blocks for vision pipelines.

This chapter provides the theoretical and practical background for this thesis. It reviews the evolution of foundation models, discusses the limitations of out-of-the-box approaches in specialized domains, and introduces key techniques (such as self-supervised learning and adapter-based fine-tuning) as scalable solutions for data scarce vision tasks.

2.1 Foundation Models: Trends and Challenges

Foundation models have gained substantial momentum in computer vision, with a strong trend toward generality, scalability, and multi-modality. Vision-language models such as VisionLLM [77], VL-SAM [43, 42], LLaFS [85], and GLEE [80] demonstrate that Large Language Models (LLMs) can serve as powerful open-ended decoders for a wide range of visual tasks, including object detection, segmentation, and image captioning. These systems often rely on prompt-based interfaces or APIs, allowing users to describe tasks in natural language, thus bypassing the need for explicit domain-specific training. VL-SAM, for example, integrates the prompt flexibility of LLMs with the segmentation capabilities of the SAM to enable open-ended, training-free segmentation across domains.

Recent literature, such as the review "Vision-Language Model for Object Detection and Segmentation," [21] highlights the strengths of these models in closed-set detection, domain adaptation, and few-shot learning scenarios. MarvelOVD [76], for example, enhances region-level grounding in object detection by refining the interaction between detectors and vision-language embeddings.

These developments collectively represent a paradigm shift: rather than developing specialized models for each task, many pipelines now combine pre-trained, general-purpose modules (like SAM or LLMs) in zero-shot or few-shot settings. Their ease of use through model hubs or API-accessible endpoints makes them particularly attractive for real-world deployment and prototyping.

Despite their impressive capabilities, these models often struggle in specialized domains that deviate from the distributions seen during pretraining. In grassland imagery, for instance, subtle differences between plant species, overlapping vegetation, and low inter-class variance pose serious challenges. General-purpose segmentation often lacks the spatial granularity and semantic sensitivity required for precise, localized detection, especially when visual features are faint, occluded, or visually similar to background textures.

This limitation is illustrated in Figure 2.1, which shows example predictions from GPT-4o and OWL-ViT across four grassland image crops of increasing realism and difficulty. The final column features a sample from the KIhUG dataset. The models were prompted using either natural language (e.g., Locate all larger plant centers) or tag-based queries (e.g., [plant, shrub]), with the task reduced to simply identifying plant-like regions. Despite this simplification, OWL-ViT consistently fails to produce meaningful results. GPT o3 performs somewhat better, especially in less complex scenes, but still returns vague or imprecise locations that fall far short of what is needed for reliable, high-precision plant detection. This example underscores the inadequacy of generic foundation models for even basic spatial localization in dense, visually ambiguous environments, reinforcing the need for domain-specific alternatives.

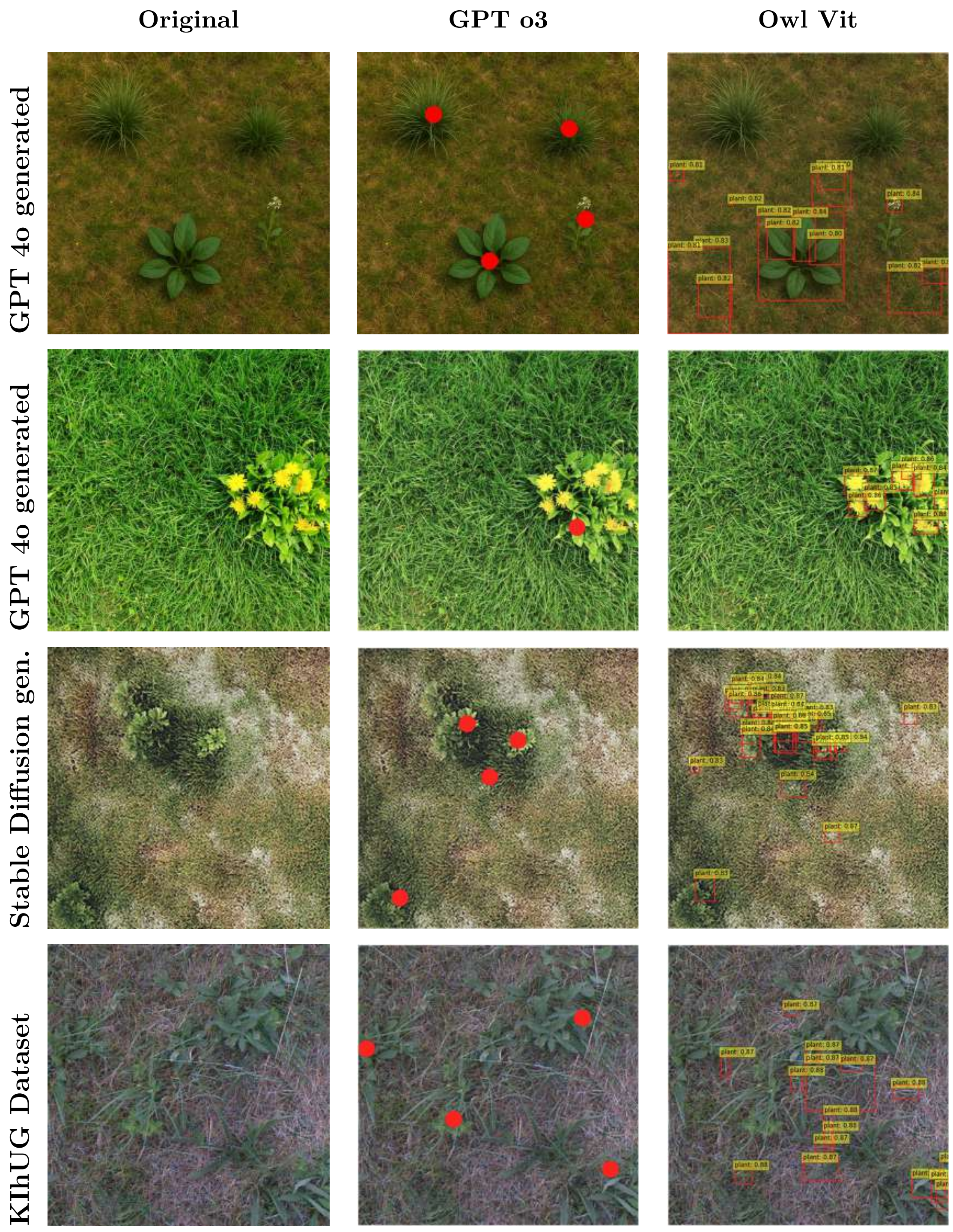


Figure 2.1: Example of plant predictions made by GPT o3 and OWL-ViT, organized by input image (rows) and evaluation model (columns). For GPT-4o, the prompt was: *Can you give me the relative coordinates of all larger plant centers you detect in the given image? The images upper-left corner is (0,0) and the bottom-right is (1,1). Can you try to guess without using classical approaches?* OWL-ViT, in contrast, was queried using a simple tag-based prompt: *["plant", "shrub"].*

2.2 Domain-Specific Model Requirements

While general-purpose foundation models perform impressively on diverse tasks, their effectiveness often drops in specialized domains. Settings that are characterized by rare object classes, subtle visual cues, or complex environmental conditions are typically underrepresented in large-scale pretraining datasets. As a result, applying off-the-shelf models to such domains often leads to suboptimal results.

Several studies highlight this issue. Li et al. [40] found that the SAM failed to generalize to permafrost mapping, revealing poor adaptability to out-of-distribution (OOD) environments. Likewise, Xu et al. [82] show that even adapted foundation models often underperform compared to simple supervised baselines on domain-specific tasks.

Chen et al. [12] further argue that effective domain adaptation requires architectural adjustments (e.g., adapters), specialized objectives (e.g., SSL), and efficient annotation strategies (e.g., pseudo-labeling). They emphasize **modularity** and **parameter efficiency** as key enablers for success in low-data domains.

In agricultural contexts like weed detection in dense grasslands, these demands are even more pronounced. Visual distinctions are subtle, occlusion is common, and annotated data is scarce. To better understand the trade-offs, it is helpful to contrast general-purpose and domain-specific models:

General-Purpose Models	**Domain-Specific Models**
• Trained on large, diverse web-scale datasets	• Trained on narrow, task-specific data
• Effective across many tasks out-of-the-box	• Require fine-tuning but adapt precisely
• Often lack precision in unfamiliar domains	• Capture subtle, context-sensitive features
• Zero-/few-shot capability reduces annotation needs	• Can use pseudo-labels to reduce annotation effort
• Typically large; support PEFT, but require more data for effective adaptation	• Support PEFT for efficient training

This comparison underscores that while generalist models prioritize breadth and convenience, domain-specific approaches are better suited for precision-critical tasks, especially when enhanced by SSL and PEFT. These considerations shape the methodological foundation of this thesis.

2.3 Importance of Plant Detection in Grassland Environments

Plant detection has become a key application area for machine learning in agriculture, with deep learning techniques increasingly adopted for tasks such as crop monitoring, yield estimation, and weed control [35]. However, most successes in this domain are found in highly structured settings (such as row crops) where plants are spatially arranged and their phenotypes are well-documented. In contrast, grassland environments present a more complex, unstructured challenge that remains underexplored in current research.

Grasslands contain a high diversity of plant species with overlapping morphologies and similar color profiles, often growing in dense, non-uniform patterns. As such, the visual signals needed to distinguish between harmful and harmless plants are often subtle and inconsistent. For example, species such as *Jacobaea vulgaris* may closely resemble other non-toxic flora like *Ajuga reptans*, particularly during early growth stages. This visual ambiguity complicates detection, even for human experts, and becomes a major bottleneck for automated approaches. A visual example of such visual similarity is shown in Figure 2.2.

Adding to the challenge is the fact that chemical control methods are either discouraged or outright banned in many ecologically sensitive grasslands. As a result, plant identification and removal are often performed manually – a task that is time-consuming, labor-intensive, and prone to error. Studies such as [24] emphasize the importance of precise within-field plant mapping as a basis for effective intervention strategies. However, this process often relies on aerial or proximal sensing data that must be interpreted with care, especially in vegetation-dense environments.

In this context, the role of machine learning is not just to replicate human identification, but to do so reliably across variable lighting, growth stages, and occlusions. Yet current segmentation and detection models often underperform when transferred from curated datasets to these unpredictable natural scenes. The work by van Marrewijk et al. [72] underlines how costly annotation and domain variation hinder the scalability of deep learning in practical agricultural tasks, advocating for efficient solutions such as active learning and domain-specific adaptation.

Together, these insights frame the relevance of this thesis: developing detection models that can perform under the nuanced demands of grassland environments without requiring extensive manual annotation. Accurate identification of harmful species like *J. vulgaris* is not only agriculturally beneficial but also critical for ecological management and regulatory compliance.

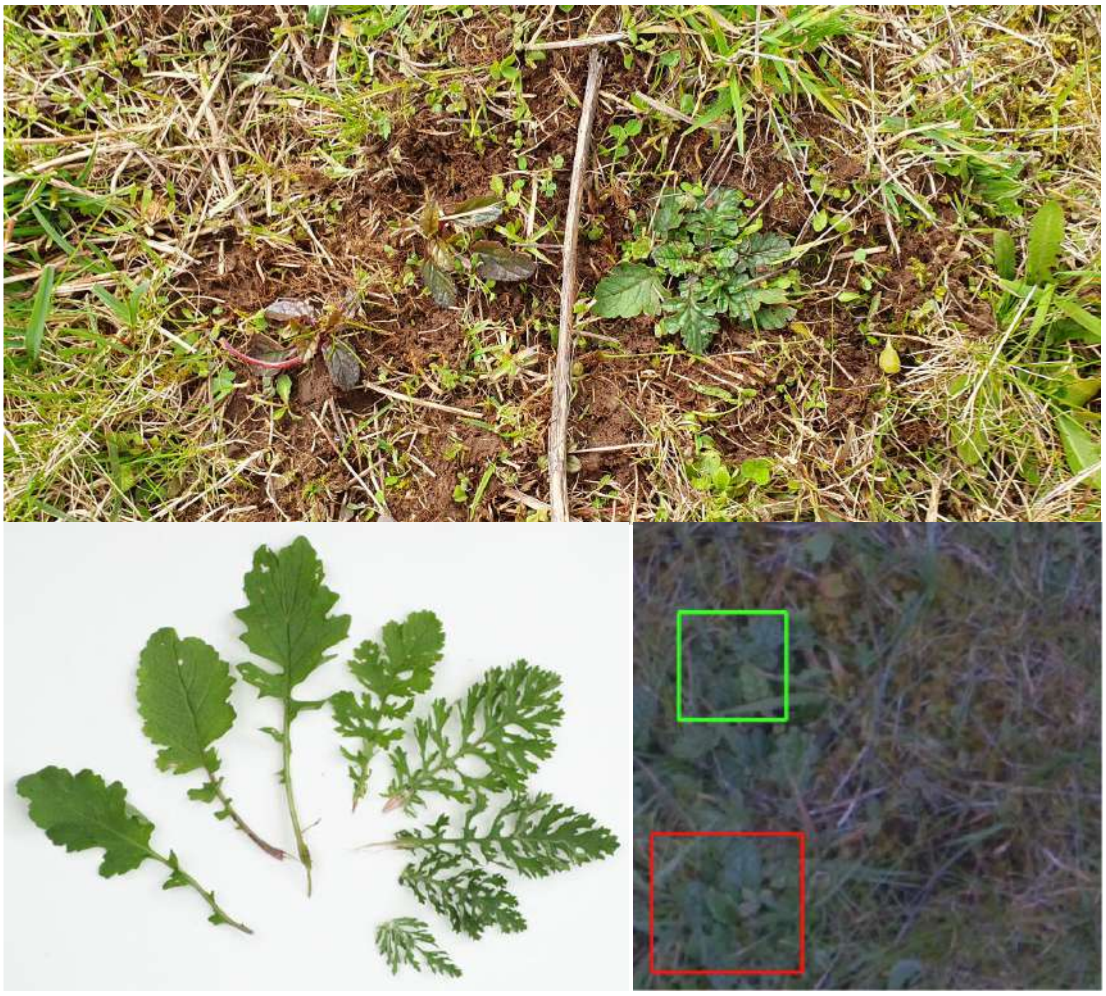


Figure 2.2: **Top:** High-resolution field image showing morphologically similar plant species – early-stage *Ajuga reptans* (left) and *Jacobaea vulgaris* (right) – growing side by side in a natural grassland environment. The image offers well-lit, detailed visual cues for species discrimination (Fig. 3.7 [73]).
Bottom Left: Leaf shape variations of *J. vulgaris* across different growth stages. As the plant matures, its leaves become more deeply lobed and feather-like, complicating species identification under field conditions. (Image credit: Pierre Aeby, Landwirtschaftliches Institut Grangeneuve.)
Bottom Right: Dataset example depicting the same species as above – *A. reptans* (red) and *J. vulgaris* (green) in early grow stages – under challenging real-world conditions: low lighting, low resolution, partial occlusion, and early growth stages. Accurate discrimination is particularly difficult due to the poor visual quality and subtle morphological differences.

2.4 Adapters and Self-Supervised Learning: Concepts and Benefits

Two complementary strategies, PEFT (Parameter-Efficient Fine-Tuning) and SSL (Self-Supervised Learning), have emerged as promising solutions for adapting vision models to specialized domains where annotated data is scarce and computational resources are constrained. These methods form the conceptual basis of the approach proposed in this thesis.

Adapter-Based Fine-Tuning

Adapters are lightweight, trainable modules inserted into pre-trained neural networks to enable task-specific adaptation without modifying the full model. Originally introduced by Houlsby et al. [31], adapters have gained traction for their ability to preserve the general knowledge encoded in a model backbone while significantly reducing the number of parameters that require training. Instead of updating all weights during fine-tuning, only the adapter layers are optimized, making the approach more efficient in terms of both memory and compute.

This modular design supports flexible task-switching and multi-domain training setups, while also mitigating the risk of catastrophic forgetting. Several extensions have been proposed to further improve efficiency and expressiveness, including LoRA [32], which uses low-rank matrix decomposition, and Compacter [47], which employs shared hypercomplex adapters. In the context of computer vision, approaches like AdaptFormer [11] demonstrate the viability of adapter-based fine-tuning for vision transformers, though adoption in niche domains such as agriculture or remote sensing is still emerging.

Self-Supervised Learning

SSL provides a framework for learning meaningful representations from unlabeled data by defining pretext tasks that do not require manual annotations. These tasks exploit structural or semantic regularities in the data. For example, contrastive learning [10], where models are trained to distinguish between augmented views of the same image, or masked image modeling [27], where models learn to reconstruct missing image patches.

Such strategies promote the acquisition of features that generalize across tasks, such as object boundaries, textures, and spatial configurations which are characteristics particularly important for downstream tasks like object detection. In the case of object detection, methods like DETReg [3] show how self-supervised pretraining can develop objectness priors using pseudo-labels, even in the absence of manually annotated boxes.

Combining Adapters and SSL

While both adapter-based fine-tuning and SSL have been successfully applied in isolation, their combined use remains largely underexplored, particularly in domains with limited labeled and unlabeled data. The idea of applying self-supervised training to adapter modules, rather than to the full network, raises a number of open questions: Can adapters alone capture useful representations during pretraining? Can this setup offer a meaningful balance between efficiency and accuracy? And to what extent does it transfer to downstream tasks?

The experimental approach presented in this thesis investigates this possibility. Specifically, adapters are inserted into a DDETR architecture and trained using self-supervised object-level signals extracted from unlabeled domain data. These same adapters are then fine-tuned for detection on pseudo-labeled data. While this setup is unconventional and its theoretical foundations are not yet well-established in the literature, it offers a low-cost, modular pipeline for testing domain adaptation under realistic data constraints.

Rather than claiming state-of-the-art results, the aim is to evaluate whether this combined strategy can yield competitive performance relative to full-model training, especially in scenarios where resource or annotation budgets are tight. If adapters trained via SSL show even partial transferability, they may serve as a promising direction for building scalable, domain-specific models with minimal supervision.

The next chapter reviews related work in these areas and situates this investigation within the broader landscape of vision adaptation strategies.

3 | Related Work

This chapter reviews research relevant to the development of domain-specific foundation models, focusing on architectural design, data-efficient training strategies, and cross-domain adaptation. It covers key advances in object detection SSL frameworks, pseudo-labeling methods, and adapter-based fine-tuning. Emphasis is placed on approaches that reduce annotation needs and computational costs while preserving performance in specialized domains. By analyzing the strengths and limitations of current methods, this chapter motivates the combination of parameter-efficient adaptation and SSL explored in the following chapters.

3.1 Object Detection and DETR Variants

Object detection has long been a central task in computer vision, with foundational deep learning models such as Faster R-CNN [57] and You Only Look Once (YOLO) [56] achieving widespread success. YOLO set a benchmark for real-time detection, while Faster R-CNN introduced a two-stage pipeline prioritizing accuracy. Both architectures have since evolved through numerous follow-up versions (such as YOLOv12 [69]) while remaining competitive alongside newer designs. A major architectural shift came with Detection Transformer (DETR) [9], which redefined object detection as a direct set prediction problem. It eliminates components like anchor boxes and Non-Maximum Suppression (NMS) by employing a transformer-based encoder-decoder architecture and bipartite matching for loss computation. While DETR's end-to-end design is conceptually elegant and simplifies the detection pipeline, it suffers from high computational demands and slow convergence, especially on smaller datasets.

While DETR represents a breakthrough in architectural clarity, its practical adoption is hindered by slow training convergence and limited scalability to small or sparse datasets. Consequently, several improved variants have been proposed:

- **Conditional DETR** [48] accelerates training by refining how object queries attend to encoder outputs, introducing conditional spatial query embeddings.
- **DN-DETR** [39] proposes a denoising strategy during training by injecting known object information into queries, which improves robustness and accelerates convergence.
- **DDETR** [86] introduces multi-scale deformable attention mechanisms that focus on sparse keypoints rather than full feature maps, significantly reducing the computational cost and enabling application to high-resolution images.

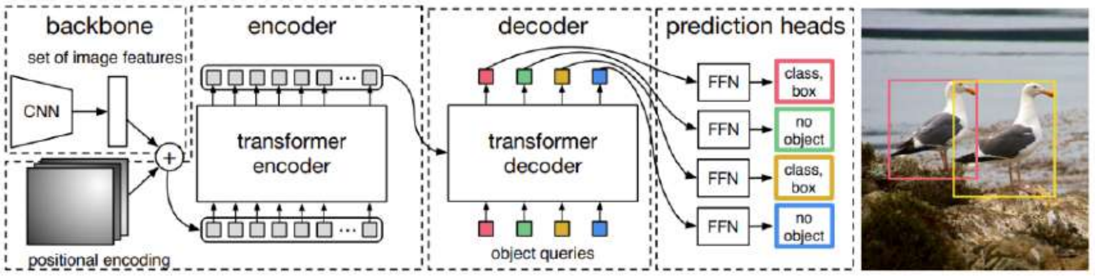


Figure 3.1: Overview of the DETR architecture. The image features a Convolutional Neural Network (CNN) backbone (typically a ResNet) that produces spatial feature maps, followed by a transformer encoder-decoder. A fixed set of learned object queries is passed through the decoder, which predicts object class labels and bounding box coordinates. Bipartite matching with the Hungarian algorithm enables end-to-end set prediction.

(Fig. 2 from [9])

These variants collectively address core limitations of the original DETR formulation. In particular, DDETR serves as the architectural basis for the work presented in this thesis, due to its efficient handling of dense and high-resolution imagery common in aerial and agricultural contexts. Overall, DETR and its extensions demonstrate how transformer-based architectures can be tailored for detection while maintaining the benefits of global context modeling. The following sections examine how such architectures can be adapted further through SSL and parameter-efficient fine-tuning techniques.

3.2 SSL for Detection

SSL has emerged as a powerful paradigm for learning visual representations without manual annotations. Instead of relying on labeled datasets, SSL methods define pretext tasks that exploit inherent structure in the data. These tasks encourage models to learn meaningful features that transfer well to downstream applications such as classification or object detection.

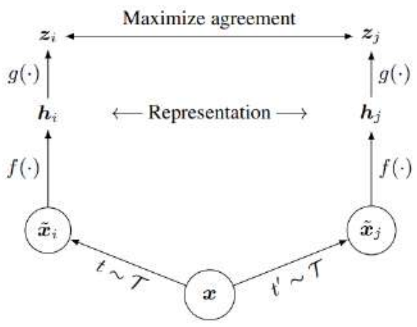
A wide range of SSL techniques has been proposed for image-level representation learning. Contrastive methods like SimCLR [14], MoCo [28], and BYOL [25] train networks to maximize agreement between different augmentations of the same image. Notably, Bootstrap Your Own Latent (BYOL) stabilizes training through the use of a momentum encoder updated via an Exponential Moving Average of Weights (EMA) of the student weights, a mechanism that has since influenced a broad set of teacher-student strategies in SSL pipelines. In parallel, masked image modeling approaches like Masked Autoencoders (MAEs) [27] reconstruct missing image regions from visible patches, enforcing semantic understanding of spatial context. Other vision transformer-based methods like DINO [10] rely on patch-level matching and self-distillation to extract transferable features.

Extending SSL to object detection introduces additional complexity due to the need for spatial localization alongside semantic classification. Detection-oriented approaches such as UP-DETR [17] and DETReg [3] address this by coupling vision transformers with auxiliary pretext tasks or spatial priors. UP-DETR incorporates location-based objectives, whereas DETReg leverages unsupervised region proposals to define object-centric training targets in the absence of ground truth.

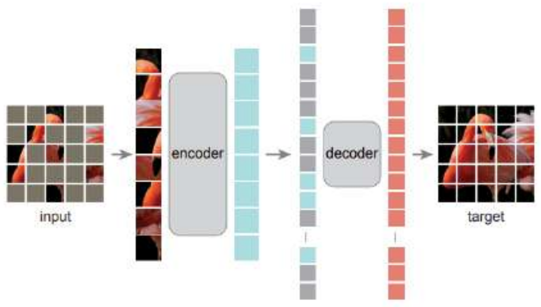
However, DETR-style SSL remains an area of active research. Recent findings by Ma et al. [46] questioned the robustness of DETRegs pretraining, suggesting that simpler strategies may offer more consistent performance across benchmarks.

Despite these uncertainties, the use of SSL for object detection remains attractive, particularly in domains where labeled data is scarce. Region-aware pretraining enables detection transformers to acquire objectness priors from raw inputs, reducing dependence on annotation-heavy pipelines.

Moreover, recent trends in student-teacher training frameworks often rely on EMA-based weight averaging to stabilize learning. Beyond BYOL, newer works leverage this principle by maintaining a teacher model as an exponential moving average of the student. This teacher supervises training via pseudo-labels such as bounding boxes or segmentation masks, enabling self-supervision even in structured prediction tasks. Studies such as Morales-Brotons et al. [49] demonstrate that EMA-based models yield better convergence stability and robustness, making them a valuable component of modern SSL pipelines.



SimCLR
(Fig. 2 from [14])



MAE
(Fig. 1 from [27])

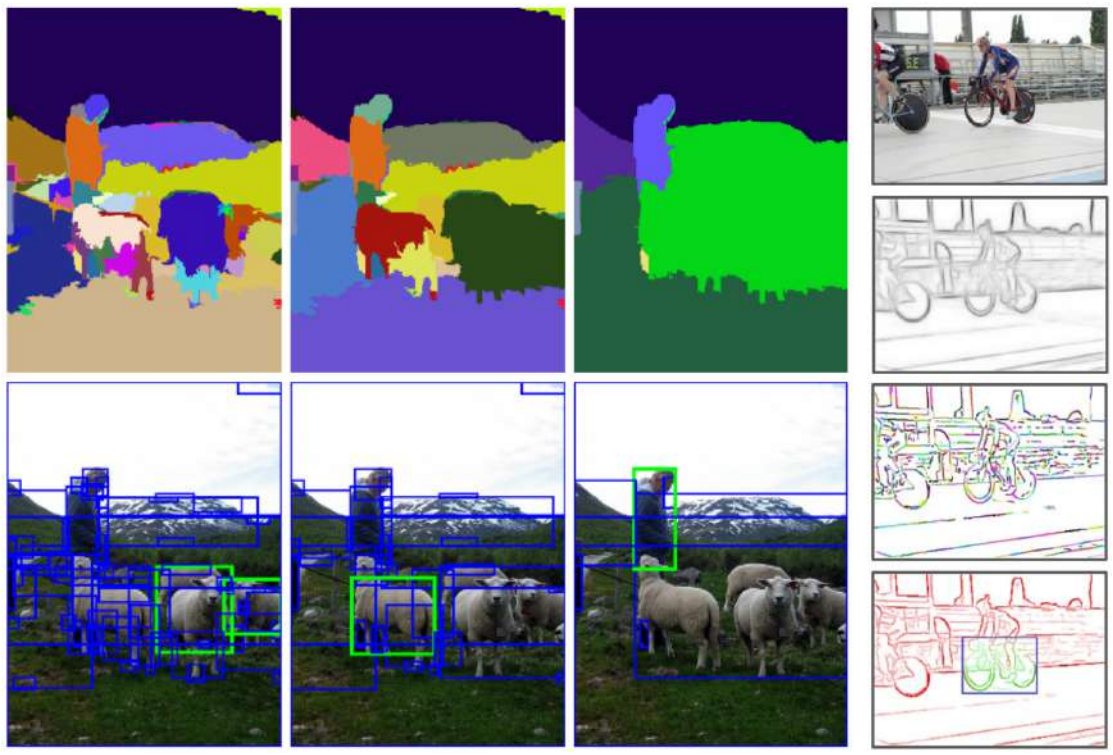
Figure 3.2: Comparison of two widely used SSL paradigms. **Left:** SimCLR [14] uses contrastive learning by generating two augmented views of the same image and training an encoder-projection head pipeline to maximize agreement between the resulting representations. **Right:** MAE [27] adopts a reconstruction-based approach by masking a large portion of image patches, encoding only the visible ones, and reconstructing the full image using a lightweight decoder. After training, the decoder (only MAE) and projection head (both) are discarded, and the encoder is reused for downstream tasks.

3.3 Pseudo-Labeling Approaches

The previous section highlighted the importance of SSL for learning visual representations without manual supervision. Among the most widely adopted strategies in this context is pseudo-labeling, which involves generating approximate ground-truth annotations from unlabeled data. These pseudo labels can take various forms, including class labels, bounding boxes, or segmentation masks, depending on the target task.

In object detection, pseudolabeling is often integrated into SSL pipelines to encourage the learning of objectness priors. DETReg [3], for instance, uses region proposals from a regionproposal generator to pretrain DETR-based architectures. Similarly, UP-DETR [17] employs synthetic targets through auxiliary tasks, such as patchbased localization, to bootstrap transformerbased detection models.

More classical region proposal techniques such as Selective Search [70] and Edge-Boxes [87] have historically been used for generating object hypotheses without supervision, and can also serve as sources of pseudo labels. These methods exploit visual cues such as edges, textures, and color similarity to generate candidate bounding boxes, which can be used for weak supervision or dataset bootstrapping.



selective search
(Adapted from Fig. 2 in [70])

edge box
(Fig. 1 [87])

Figure 3.4: Results and intermediate steps in the Selective Search and Edge Boxes pipelines, taken from the original sources.

Pseudo-labeling also plays a significant role in semantic segmentation. The SAM [36] has popularized prompt-based segmentation, allowing users to extract object masks from minimal input. While SAM enables pseudo-mask generation at scale, its performance deteriorates in highly specialized domains. For example, Li et al. [40] show that SAM fails to generalize to remote sensing imagery used in permafrost mapping, illustrating the limitations of foundation models when applied to niche datasets. This aligns with the broader critique raised in Chapter 2 regarding the overgeneralization of universal models in domain-specific tasks.

Beyond computer vision, pseudo-labeling is widely used in other areas of Machine Learning (ML), including natural language processing, speech recognition, and information retrieval. In these domains, it serves as a critical strategy for bootstrapping models from limited supervision, often in conjunction with confidence-based sampling or data augmentation. For example, selective search techniques adapted for text retrieval (such as MICO [78]) combine unsupervised clustering with mutual information co-training, demonstrating that pseudo-labeling and representation learning can be effectively co-optimized even in non-visual settings. In semi-supervised learning more broadly, methods like FixMatch [65] unify consistency regularization with confidence-based pseudo-label selection to achieve strong performance with minimal labeled data. Similarly, research on confirmation bias in pseudo-labeling [2] highlights both the potential and pitfalls of this technique, prompting refinements that make it more robust across diverse tasks. These approaches collectively emphasize the versatility and transferability of pseudo-labeling across modalities and problem domains.

Ultimately, there is no single pseudo-labeling method or hyperparameter configuration that universally outperforms others. As domains become more specialized, pseudo-labeling strategies must also be tailored to the data structure, noise, and semantic content. This adaptability makes pseudo-labeling not only a practical solution for resource-limited scenarios but also a crucial component of modern data-efficient learning systems.

As shown in Figure 3.4, both Selective Search and Edge Boxes highlight the sensitivity of proposal quality to hyperparameter choices. The left image illustrates how varying segmentation granularity in Selective Search can reveal different subsets of relevant objects, none of which fully capture all ground truth instances. Although combining multiple hyperparameter settings can improve recall, it also introduces redundancy and noise. This fundamental trade-off persists even in carefully tuned pipelines, emphasizing the need for domain-aware heuristics and adaptive thresholds in specialized applications. Moreover, the design choices made in algorithms like Edge Boxes (such as contour density thresholds and edge orientation heuristics) reflect handcrafted compromises that rarely generalize across datasets. As such, pseudo-labeling remains an inherently imperfect process, best guided by task-specific insight rather than universal recipes.

3.4 Adapter-Based Fine-Tuning and PEFT

As vision models grow in size and complexity, fully fine-tuning them for each new task or domain becomes increasingly inefficient. PEFT (Parameter-Efficient Fine-Tuning) addresses this challenge by modifying only a small subset of model parameters, enabling resource-efficient adaptation with minimal performance loss and significantly reduced memory footprint.

A foundational approach to PEFT is the use of *adapters*, introduced by Houlsby et al. [31]. These lightweight bottleneck modules are inserted between transformer layers and trained while the main model remains frozen. Formally, an adapter layer is typically defined as:

$$\text{Adapter}(h) = h + W_{\text{up}}\,\sigma(W_{\text{down}}\,h), \tag{3.1}$$

where h is the hidden representation, $W_{\text{down}} \in \mathbb{R}^{d\times r}$ and $W_{\text{up}} \in \mathbb{R}^{r\times d}$ are the down- and up-projection matrices, $r \ll d$, and σ is a non-linear activation function.

Several refinements of this idea have been proposed. **LoRA** (Low-Rank Adaptation) [32] introduces low-rank decompositions directly into the attention and feed-forward components of transformers, injecting trainable matrices without altering the original weights. This preserves compatibility with pre-trained checkpoints and accelerates training. **Compacter** [47] further compresses adapters using hypercomplex multiplication and shared parameter matrices, achieving strong performance in NLP tasks at an even smaller parameter cost.

Although originally developed for natural language processing, PEFT techniques are increasingly applied to vision tasks. **Visual Prompt Tuning** [33] adapts prompt-based learning by optimizing visual tokens that condition pre-trained vision models. Similarly, **AdaptFormer** [13] and **Vision Transformer Adapters for Dense Prediction** [16] extend adapter-based techniques to tasks like classification, segmentation, and object detection, demonstrating their efficacy in visual domains.

Recent surveys such as [81] offer a comprehensive overview of PEFT strategies for pre-trained vision models, highlighting trends, design patterns, and challenges. These findings underscore the versatility of adapters and related techniques as practical alternatives to full fine-tuning, particularly in low-resource or domain-adaptation settings.

Despite these advances, the integration of adapter-based tuning with SSL and object detection remains relatively underexplored. Most prior work focuses on supervised learning or simpler classification benchmarks. This thesis investigates the synergy between adapter-based tuning and self-supervised object detection, aiming to enable scalable and efficient adaptation of pre-trained models in domain-specific settings with limited supervision.

3.5 Domain Adaptation in Vision Models

Domain adaptation aims to transfer knowledge from a labeled source domain to an unlabeled or sparsely labeled target domain, addressing the performance degradation caused by domain shift. This is particularly important for real-world computer vision applications, where training and deployment environments often differ significantly in terms of appearance, texture, lighting, or sensor characteristics.

A foundational work in unsupervised domain adaptation is the Domain-Adversarial Neural Network (DANN) [23], which introduced a gradient reversal layer to align feature distributions across domains through adversarial training. By minimizing the task loss while simultaneously maximizing the domain confusion, DANN encourages the extraction of domain-invariant features. Building on this, Discriminative Adversarial Domain Adaptation (DADA) [67] refines adversarial alignment by ensuring that discriminative class boundaries are preserved in the target domain, thereby improving classification performance.

Alternative strategies include discrepancy-based methods, such as Maximum Classifier Discrepancy (MCD) [62]. Rather than aligning feature distributions directly, MCD trains multiple classifiers and encourages them to produce consistent outputs on the target domain. When disagreement arises, it indicates a domain gap, prompting the feature extractor to adjust its representation.

In the context of object detection, domain adaptation introduces further challenges, as spatial and structural consistency must also be preserved. Chen et al. [15] extended the Faster R-CNN framework by incorporating image-level and instance-level domain classifiers, enabling adaptation both globally and locally. Similarly, the Strong-Weak Distribution Alignment method [61] distinguishes between strong alignment at the image level and weak alignment at the instance level, resulting in better cross-domain generalization for object detectors.

These approaches demonstrate the growing diversity of domain adaptation techniques in computer vision, reaching from adversarial learning and discrepancy minimization to specialized detector modifications. Despite significant progress, domain adaptation for dense prediction tasks, particularly under resource and annotation constraints, remains an open challenge.

This thesis addresses domain adaptation from a different angle: rather than aligning feature distributions explicitly, it explores the use of parameter-efficient pretraining, pseudo-labeling, and SSL to adapt detection transformers to novel domains with minimal supervision.

4 | Methodology

This chapter outlines the methodological foundations and experimental design that underpin the development of a domain-specific object detection model for plant identification in grassland environments. The focus lies on the rationale behind key design decisions, particularly in relation to dataset selection, training strategies, and model architecture. Rather than detailing the technical implementation, which is covered in the subsequent chapter, this section emphasizes the conceptual reasoning that guided the overall approach. The chapter is divided into two main sections. The first section introduces the data sources used throughout this work, including internal, public, and synthetic datasets. These are discussed with respect to their relevance for both self-supervised pretraining and downstream finetuning. Particular attention is given to the challenges inherent in collecting representative visual data in outdoor environments and the strategies adopted to mitigate them. The second section focuses on the methodological aspects of model training, including the choice of architecture, the incorporation of adapter modules, and the use of self-supervised objectives to reduce dependence on manual annotation. The design of the pseudo-labeling pipeline and the application of regularization techniques to improve training stability are also discussed.

4.1 Data Sources and Preparation

Data availability is a central factor in developing effective machine learning models, particularly in specialized domains where annotated datasets are scarce or difficult to obtain. In this work, the goal is to train a domain-specific model using SSL, which removes the need for manual labels during the pretraining phase. However, this does not eliminate the importance of data quality: the visual data must still contain relevant domain features to support meaningful representation learning. The target domain in this thesis, the detection of plants in grassland environments, introduces unique challenges. Plants often blend with the background due to similar textures and colors, and their appearances vary widely depending on species, growth stage, and environmental conditions. Collecting a sufficiently diverse and representative dataset is non-trivial and often constrained by logistical and seasonal factors. This situation is not unique. Other domains, such as medical imaging or industrial inspection, may suffer from even stricter limitations due to data sensitivity, privacy concerns, or the high cost of acquisition. As such, choosing appropriate datasets is a critical step. In this thesis, a combination of real, synthetic, and publicly available image data is used to maximize diversity while minimizing manual effort. The following sections describe each of these sources and their specific roles in training and evaluation.

4.1.1 Internal Dataset (KIhUG)

The internal dataset used in this thesis was collected within the KIhUG project using a Basler Ace Pro 2 camera mounted approximately 60 cm above the ground. The dataset comprises a total of 4,250 high-resolution images with a resolution of 1920×1200 pixels, captured at 10-30 frames per second. Data acquisition was performed using two different setups: one involving a tractor-mounted camera and the other a manually guided handcart. Ground truth annotations were provided for 6,302 object instances and stored in YOLO format. Each annotation file contains one or more lines, with each line representing a single bounding box using center-normalized coordinates. These label files are organized in a parallel directory structure under `labels/`, with file names matching their corresponding images.

To prevent data leakage and ensure a clean separation between training and evaluation, a structured Yet Another Markup Language (YAML) file was created. It explicitly defines the dataset splits, listing image paths grouped into `train`, `val`, and `test` sections. The split was designed to separate entire image sequences and to prevent multiple views of the same plant from appearing across different subsets. Background-only images (without labeled instances) were excluded to avoid skewing the evaluation and learning process. Based on this primary split, additional YAML files were generated to define various sub-datasets used in later experiments. Figure 4.1 provides an overview of the image and object counts across these dataset variants. This split configuration forms the basis for all subsequent experiments and ensures reproducible and controlled comparisons across training regimes. The labeling format and annotation protocol were inspired by practices used in the Rumex Weeds dataset, described in the following section.

During training, this dataset can be easily processed through a data loader pipeline that applies scaling, normalization, and optional data augmentation strategies such as horizontal flipping or color jitter.

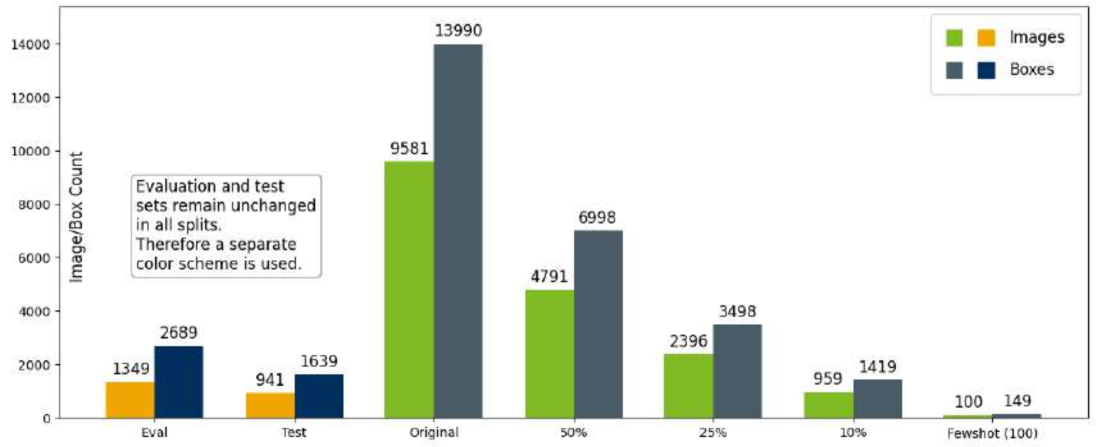


Figure 4.1: Distribution of images and labeled object instances across dataset splits defined in the different KIhUG YAML configurations.

4.1.2 Public Dataset (Rumex Weeds)

The Rumex Weeds dataset [26] is a publicly available resource created for precision agriculture and robotic weed detection. It targets the two species *Rumex obtusifolius* and *Rumex crispus* under challenging green-on-green conditions in natural grasslands. The dataset includes 5,510 RGB images captured from a top-down perspective across three different farms on four days during summer and autumn 2021. Altogether, it contains 15,519 manually annotated bounding boxes and 340 pixel-wise segmentation masks, making it one of the first publicly available datasets specifically designed for grassland weed detection.

In this thesis, the Rumex Weeds dataset is used as a second benchmark on the same domain to assess the generalization capability of the proposed self-supervised pretraining and adapter-based fine-tuning pipeline. By evaluating performance on this related yet distinct dataset, it is demonstrated that domain-specific pretraining confers transfer benefits even when the target dataset differs in species composition, image acquisition protocol, sensor or other characteristics.

The original dataset split as defined in the publication was preserved without modification to ensure comparability with existing benchmarks. In addition to using the full dataset, multiple smaller subsets were created based on fixed percentages of the original training set (e.g., 10%, 25%, 50%). These were constructed to reflect the same split ratios and structure as applied to the internal KIhUG dataset, allowing a consistent evaluation of the proposed methods performance in reduced-data scenarios.

Figure 4.2 provides an overview of the distribution of images and labeled objects in the original dataset splits.

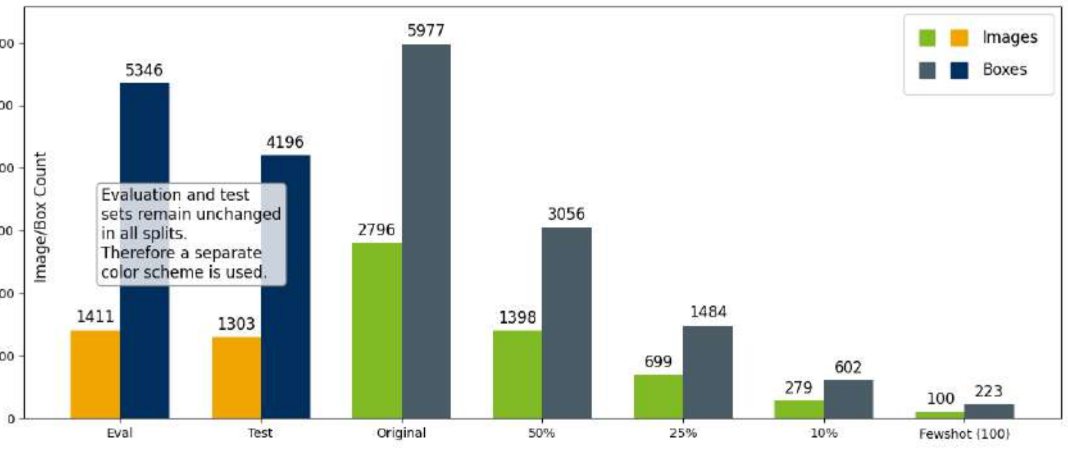


Figure 4.2: Distribution of images and labeled object instances as defined in the original Rumex Weeds dataset configuration, compared with sub-dataset splits created for this thesis.

4.1.3 Synthetic and Unlabeled Datasets

Unlabeled data form the cornerstone of the SSL strategy developed in this thesis. They enable representation learning without requiring manual annotation, thereby addressing one of the core challenges in domain-specific model development. To construct a sufficiently diverse and scalable pretraining corpus, several unlabeled image sources were selected based on visual similarity to the target domain and overall dataset size. The following paragraphs describe each of these sources in detail.

GrassClover The GrassClover dataset [64] comprises 31,600 real-world RGB images of grass-clover mixtures captured outdoors. These images feature variable lighting conditions, occlusions, and a mixture of plant species, including common weeds, with ground-sampling distances ranging from 4-8 px/mm. Additionally, the dataset includes 8,000 synthetic images with hierarchical and instance-level annotations; however, only the real, unlabeled images were used in this work. Due to its scale and domain relevance, GrassClover served as the primary source of unlabeled training data.

Internal drone/field shots This collection includes real-world images captured during an internal KIhUG project experiment, using a drone-mounted camera. The original goal was to evaluate whether drone imagery could serve as a practical basis for plant-level datasets. However, due to insufficient resolution at feasible flight altitudes, the resulting images did not allow accurate identification of individual plant species. Despite their limitations, these images remain visually consistent with the target domain and were therefore used as unlabeled inputs during self-supervised pretraining. Since they do not introduce label bias, they serve as a low-risk source of domain-relevant visual features. These images have not been published and are only used within this research context.

Synthetic renderings A set of synthetic grassland images was generated using 3D modeling environments as part of a Bachelor's thesis within the KIhUG project [37]. These renderings aimed to simulate diverse environmental conditions and vegetation structures under controlled variability. Despite featuring the target plant class, the visual quality and realism of the synthetic images were limited due to modeling constraints, particularly with regard to capturing intra-species diversity and growth-stage variations. The thesis results suggest that these synthetic samples introduced no significant bias when used in training of a model, due to what is commonly referred to as the "reality gap." Consequently, the images are useful for self supervised pretraining and are not used for fine-tuning or evaluation.

Potato tuber microscopy dataset To assess the transferability of self- supervised representations beyond the target domain, an unrelated dataset was included: microscopy images of potato tuber cross-sections [6], comprising 9,811 unstained and 6,127 stained RGB images. Although these were not used for fine-tuning, they serve as a cross-domain benchmark to evaluate the generality of the learned feature space.

To investigate the impact of data volume on self-supervised training, two nested subsets were derived from the combined SSL dataset. First, a 50% subset was randomly sampled using a custom YAML configuration script. This reduced dataset was then used as the input for a second subsampling step, producing a 25% subset of the full corpus. These subsets were employed in experiments assessing the trade-offs between dataset size, representation quality, and adapter-based parameter efficiency. An overview of the subset sizes and their composition is provided in Figure 4.3.

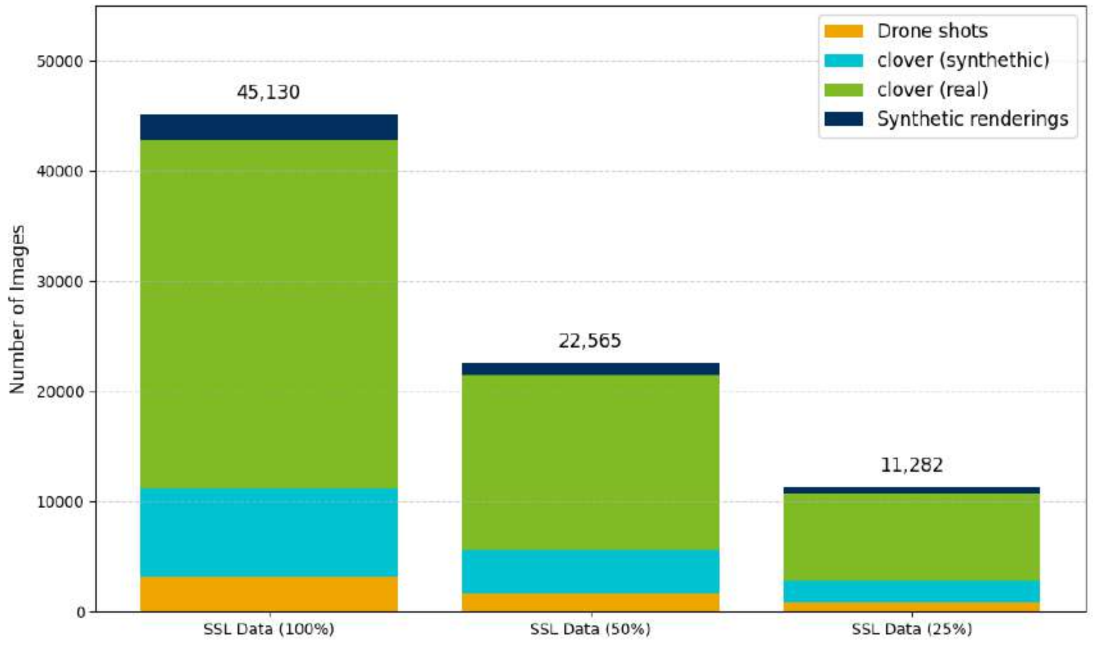


Figure 4.3: Combined and subset sizes of the unlabled datasets for SSL Training.

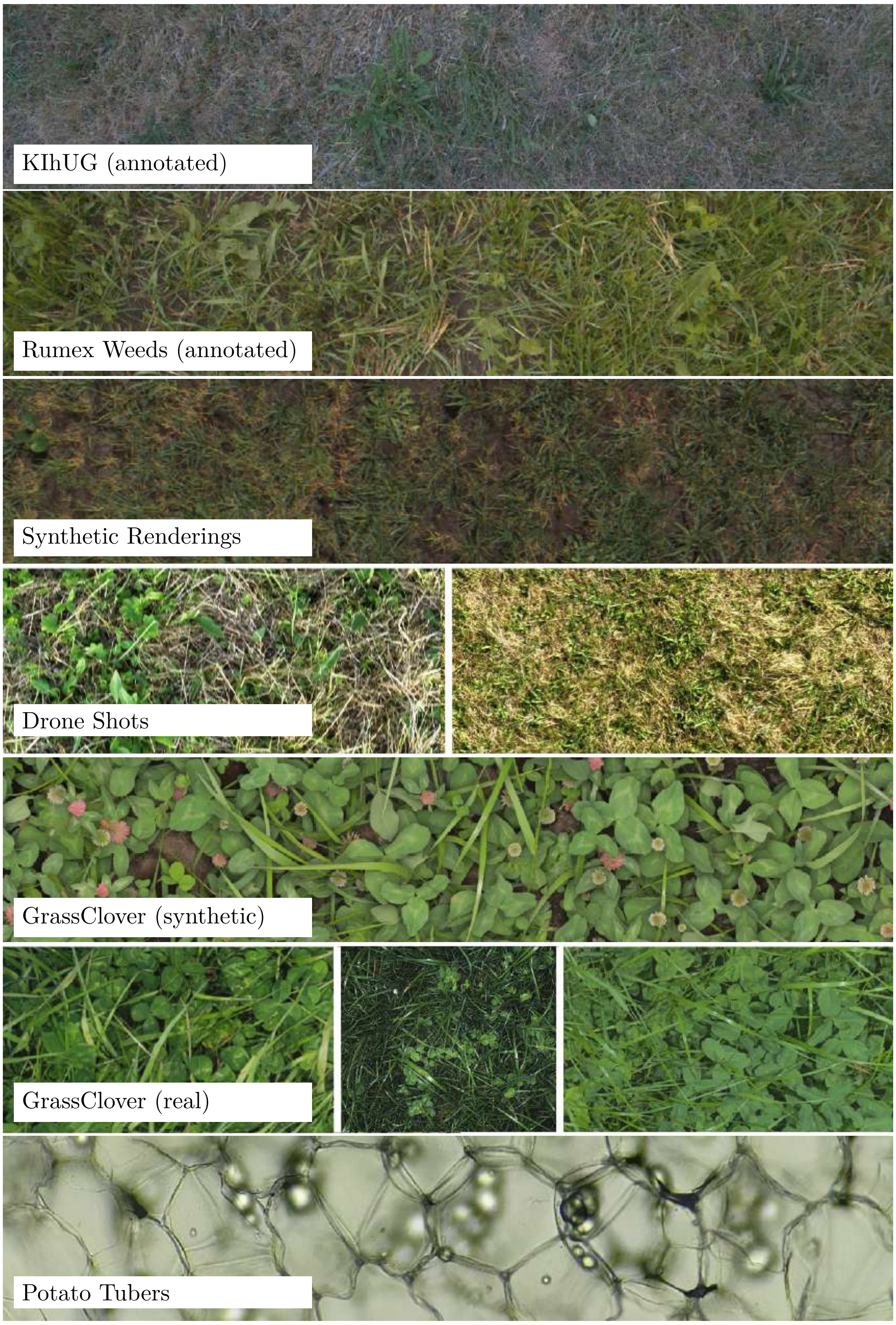


Figure 4.4: Visual samples from all datasets used in this thesis. Dataset names are embedded as labels inside each image. All images are representative crops, not shown to scale.

4.2 Training and Model Design

This section outlines the core methodology behind the proposed self-supervised object detection pipeline, including the architectural design, training strategy, and evaluation plan. The goal is to develop a model capable of learning robust and transferable visual representations from unlabeled domain-specific data, while minimizing reliance on manual annotations and maintaining parameter efficiency.

To this end, the pipeline combines pseudo-labeling with teacher–student training and curriculum learning, targeting challenging detection scenarios in heterogeneous grassland environments. Parameter-Efficient Fine-Tuning (PEFT) is employed using adapter modules to enable scalable transfer learning within both pretraining and fine-tuning stages.

A deformable transformer-based detector is selected as the base model for its capacity to handle geometric variation and dense scenes. The pseudo-labeling pipeline supports self-supervised training through bounding box proposals, while adapter integration provides a lightweight mechanism for injecting domain knowledge without updating the full model.

Training stability is further enhanced by incorporating a mean teacher model and regularization mechanisms to mitigate representation collapse. Metrics are defined both for evaluating downstream performance and for monitoring the progress of unsupervised representation learning. This section focuses on the conceptual and methodological design, while technical and implementation details such as hyperparameters and code structure are deferred to Chapter 5.

4.2.1 Problem Definition

The challenges of detecting individual plants in grassland environments have already been outlined in Chapters 1 and 2 (see also Figure 2.2). These challenges include minimal color contrast, overlapping plant structures, and high intra-class variability across growth stages. Such conditions complicate object detection even for human annotators and can render conventional deep learning models trained on structured datasets like COCO largely ineffective in this domain.

This section briefly revisits the problem to emphasize its implications for model design. Specifically, it motivates the need for a SSL pipeline that can leverage unlabeled data, avoid overfitting to scarce annotations, and generalize across species, camera setups, and growth stages. Moreover, the domain-specific visual ambiguities inherent in grassland imagery call for inductive biases that support fine-grained shape and texture cues that are ideally learned from the target domain itself rather than transferred from unrelated datasets.

4.2.2 Pseudo-Label Generation for Self-Supervised Training

Training object detection models directly on their target task has been shown to yield superior results compared to contrastive or masked pretraining approaches. DETReg [3], for example, demonstrates that pretraining with task-specific objectives (by predicting pseudo bounding boxes instead of global image features) can improve downstream performance. Their method relies on Selective Search to generate region proposals from unlabeled images.

However, subsequent analysis has highlighted critical shortcomings in this approach. In particular, Ma et al. [46] revisit DETReg and report that Selective Search proposals are noisy, low-quality, and ill-suited for detection pretraining. They argue that domain-agnostic region proposals may actively hinder representation learning, especially in complex real-world scenes. A recent survey by Kage et al. [34] confirms that pseudo-label generation, though widely used in computer vision, must be adapted to the domain and task to be effective.

These findings motivate a domain-specific pseudo-labeling pipeline for this thesis, targeting green-on-green plant structures in grassland imagery. The implemented method is intentionally simple and computationally efficient, combining classical vision techniques to produce coarse object masks suitable for online training. A more dynamic and classification-aware pseudo-labeling approach exists [75], but was deemed unnecessarily complex for the unlabeled settings addressed here.

The proposed pipeline involves the following steps:

- **Green-Red Vegetation Index (GRVI):** enhances vegetation-specific color signals [50]. GRVI is computed as:

  $$\text{GRVI} = \frac{G - R}{G + R}$$

  where G and R are green and red channel intensities, respectively. As shown by Motohka [50], values above zero typically indicate plant matter, distinguishing it from background elements like soil or litter.
- **Total Variation (TV) Denoising:** reduces texture noise while preserving meaningful edges [60].
- **Otsu Thresholding:** binarizes the GRVI map using an adaptive threshold [53].
- **Watershed Segmentation:** separates overlapping plant regions into discrete segments [74].
- **Bounding Box Filtering:** removes candidates that are too small, irregular, or elongated.

The resulting pseudo labels, while imperfect, are visually coherent and tailored to the domain. They serve as training targets during the self-supervised pretraining phase described in Section 4.2.5, and their visual outputs are illustrated in Chapter 5, Section 5.4.1.

4.2.3 Model Selection: Deformable DETR as the Base Model

DDETR [86] was selected as the base architecture for object detection due to its proven strengths in detecting small, fine-grained objects and its computational efficiency when working with high-resolution images. The model builds upon the original DETR framework but replaces its costly global attention mechanism with a more focused and efficient strategy: deformable attention.

Instead of computing attention across the full spatial extent of the image, as done in standard transformers [18], deformable attention dynamically samples a small set of key points around each query location. These offsets are learned during training, allowing the model to concentrate on semantically relevant regions while avoiding the overhead of dense attention. This localized, multi-scale focus allows DDETR to detect small, partially occluded, or visually subtle targets such as individual plants in cluttered grassland scenes, while maintaining efficient training and inference behavior.

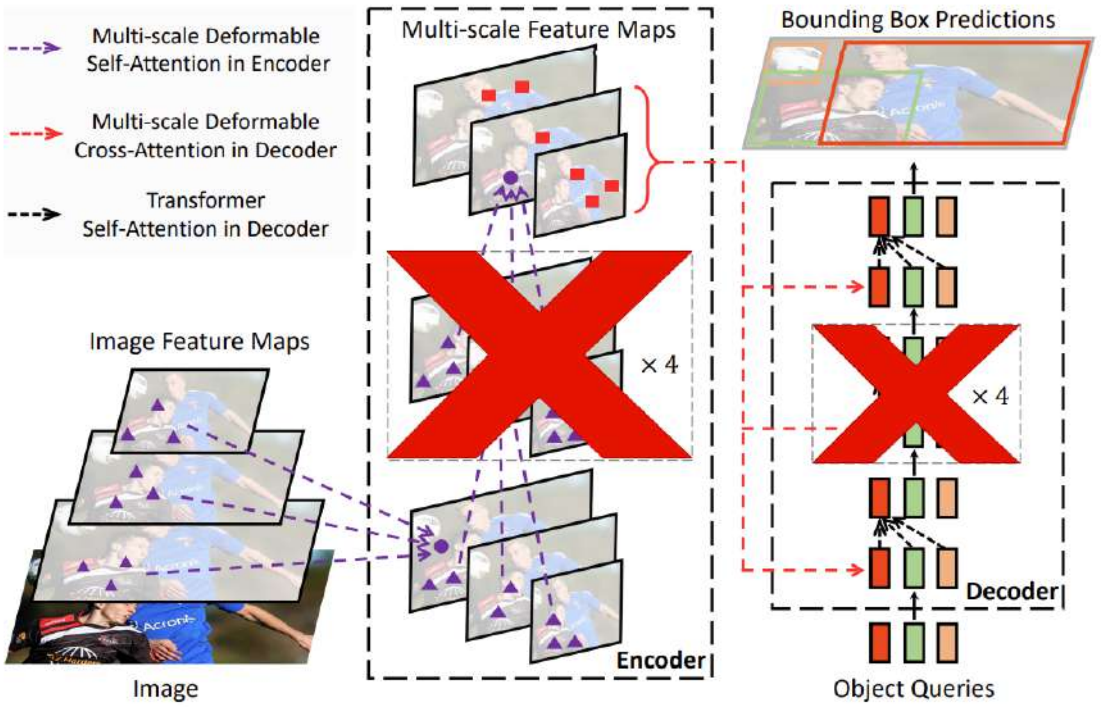

Multi-scale Deformable Self-Attention in Encoder
Multi-scale Deformable Cross-Attention in Decoder
Transformer Self-Attention in Decoder
Multi-scale Feature Maps
Bounding Box Predictions
Image Feature Maps
× 4
× 4
Image
Encoder
Decoder
Object Queries

Figure 4.5: Overview of the default DDETR architecture with reduced encoder and decoder depth. The red × symbols indicate the parts of the original architecture that were omitted in this thesis (Four of the six standard encoder and decoder layers). This reduction was made to decrease model complexity, mitigate overfitting, and ensure compatibility with lightweight deployment environments such as embedded systems.

(Adapted from Fig. 1 in [86])

A key innovation in DETR and by inheritance in DDETR is its use of a *one-to-one* assignment between model predictions and ground-truth boxes, implemented via the Hungarian algorithm [38, 51]. At each training step, the model produces Q candidate boxes and associated class scores for each image. The Hungarian matcher then finds the optimal bipartite matching between the Q predictions and the N ground-truth boxes by minimizing a composite cost:

$$\mathrm{Cost}(i,j) = \underbrace{c_{\mathrm{cls}}\left(-\log p_{i,j}\right)}_{\text{classification cost}} + \underbrace{c_{\mathrm{bbox}}\left\| b_i - b_j \right\|_1}_{\text{L1 (box) cost}} + \underbrace{c_{\mathrm{giou}}\left(1 - \mathrm{GIoU}(b_i, b_j)\right)}_{\text{GIoU cost}} \tag{4.1}$$

where

- $p_{i,j}$ is the predicted probability (after softmax or sigmoid) that query i matches ground truth j,
- b_i and b_j are the normalized box coordinates (like the center-width-height format) for prediction i and ground truth j respectively,
- c_{cls}, c_{bbox}, and c_{giou} are weighting hyperparameters controlling the relative importance of classification, ℓ_1 box distance, and generalized IoU loss.

Once the optimal assignment is found, each matched pair contributes to the loss via cross-entropy (or focal/BCE) for classification and a combination of ℓ_1 and GIoU regression losses. Unmatched queries incur only an 'no object' classification penalty. This *global matching* strategy prevents duplicate detections and ensures a clean one-to-one supervision signal, which is particularly valuable in cluttered scenes.

Transformer-based vision models are particularly well-known for their capacity to model global context and long-range dependencies, which has contributed to significant improvements in image classification, detection, and segmentation tasks [18, 54]. In many domains, such as autonomous driving, these contextual cues are crucial: a blurry object might still be correctly identified as a car simply because it appears on a road surrounded by other vehicles [19]. However, this reliance on context offers limited utility in ecological monitoring. In grassland environments, visual context is often ambiguous or misleading. The background consists of mostly indistinct foliage, and the presence of one plant has little predictive value for others. Object categories often differ only in subtle, local details such as texture, shape, or boundary smoothness. In this case, the design of deformable attention, which restricts the models focus to local, learned sampling regions, proves advantageous. While it may sacrifice some global reasoning ability (as discussed in [54, 83]), this design choice is well suited for unstructured scenes with high visual clutter and minimal contextual correlation.

In this thesis, the DDETR encoder and decoder depth was reduced from 6 to 2 layers each. This modification serves several practical and methodological purposes: it speeds up training, reduces overfitting when working with limited data, and reflects a realistic deployment scenario where lightweight models are preferred, for example in embedded systems. Prior work has shown that reducing model depth can in some cases improve generalization and convergence in data-scarce settings [83]. Moreover, the architectural modifications introduced here are compatible with the broader DDETR framework and can be scaled back up to the original depth in future work or production settings.

Finally, DDETRs modularity makes it especially amenable to the integration of adapter modules, enabling parameter-efficient updates during SSL and fine-tuning. These combined characteristics (efficiency, adaptability, strong fine-grained detection, and compatibility with lightweight training strategies) make it a suitable foundation for the model pipeline proposed in this thesis.

4.2.4 Adapter Integration Strategy

As introduced in Section 3.4, this work builds on the Houlsby adapter architecture, which inserts lightweight, bottleneck-style modules into transformer layers for efficient fine-tuning [31]. That section covered the adapter structure, mathematical formulation, and motivation from a parameter-efficient training perspective. A visual overview of the original adapter design is shown in Figure 4.6.

Adapters were inserted into both the encoder and decoder blocks of the transformer. Specifically, one adapter was added to each of the two feedforward sublayers (fc1 and fc2) per transformer block. Each adapter follows a bottleneck structure consisting of a down-projection, a nonlinearity (ReLU), and an up-projection back to the original size:

$$\text{Adapter}(x) = W_{\text{up}} \cdot \sigma(W_{\text{down}} \cdot x) \tag{4.2}$$

where $W_{\text{down}} \in \mathbb{R}^{d \times r}$, $W_{\text{up}} \in \mathbb{R}^{r \times d}$, and σ denotes a nonlinearity (here, ReLU). This allows the adapter to learn task-specific transformations with minimal additional parameters.

Following ideas from AdaptFormer [13], a **gated residual connection** is used to modulate the adapters influence on the transformer layer:

$$x' = x + \alpha \cdot \text{Adapter}(x) \tag{4.3}$$

where $\alpha \in \mathbb{R}^d$ is a learnable gating vector, initialized to a small value (0.3 for example). This mechanism softly regulates the integration of adapter features into the residual stream, offering control over adaptation strength while maintaining stable optimization. It builds on concepts from highway networks [66] and residual learning [29], and has shown to be especially effective in parameter-efficient transfer learning scenarios [13].

In addition to adapters, feature alignment modules were introduced between the CNN backbone and the transformer encoder. These modules normalize and project the multi-scale CNN outputs into a representation that is more compatible with the transformers expectations. A gated fusion mechanism that is also implemented using learnable scalar weights is applied here to balance the influence of raw and adapted features. While this alignment design is inspired by standard normalization and fusion principles, no specific publication was found to cite for this exact combination; it is introduced here as a practical solution to stabilize adapter-only training with frozen backbones.

This adapter-based strategy was selected over other low-rank or hypercomplex PEFT approaches such as Compacter [47] or LoRA [32], due to its interpretability, modularity, and demonstrated effectiveness in enabling parameter-efficient domain adaptation.

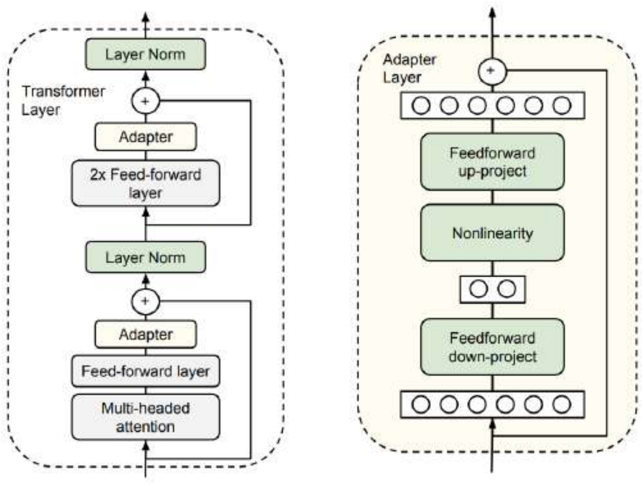


Figure 4.6: **Houlsby adapter bottleneck design.** The original design proposed by Houlsby [31] consists of a down-projection to a smaller dimensionality, a nonlinearity, and an up-projection back to the original size. The adapter is inserted as a residual connection in transformer layers to enable efficient fine-tuning.
(Fig. 2 [31])

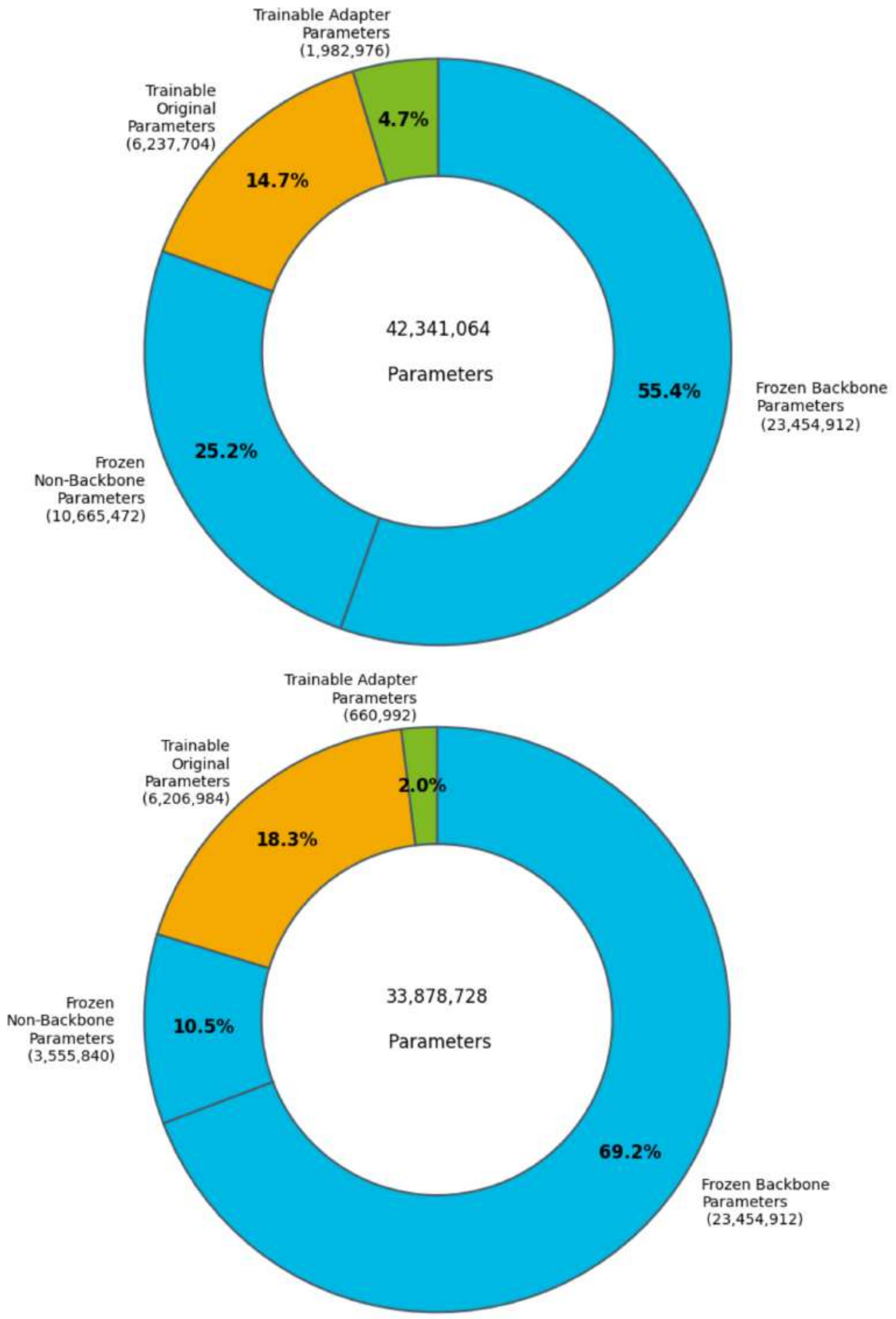


Figure 4.7: Distribution of model parameters across key components in two architectural configurations: the default DDETR with six encoder and decoder layers (top), and a reduced version using only two of each (bottom). Colors indicate the parameter role during adapter-based training: frozen weights in blue (including the backbone and selected encoder/decoder parameters), partially trainable original weights in orange, and added adapter modules in green.

4.2.5 Self-Supervised Training: A Teacher/Student Approach

To learn robust feature representations without manual supervision, this thesis adopts a teacher-student framework inspired by recent developments in SSL, most notably the BYOL paradigm [25]. The key idea is to train a student model to mimic the predictions of a teacher model, where both models share the same architecture, but the teacher is an EMA of the student weights. This asymmetry stabilizes training and prevents representational collapse, even in the absence of negative samples or contrastive pairs.

EMA Teacher Updates The teacher weights are updated after every training step using an EMA strategy, following the formulation:

$$\theta_t^{\text{teacher}} \leftarrow \tau \cdot \theta_{t-1}^{\text{teacher}} + (1 - \tau) \cdot \theta_t^{\text{student}} \tag{4.4}$$

where τ is a decay factor close to 1 (0.995 for example). This formulation makes the teacher a slowly evolving ensemble of past student weights, improving stability and convergence. Recent work [7] provides a detailed analysis on how to select and adapt the EMA decay parameter based on the learning rate and training scale. Their findings motivate the use of dynamic or context-aware EMA scheduling to improve teacher responsiveness without introducing instability. While this thesis uses a fixed τ, insights from [7] informed early tuning experiments and highlight the sensitivity of EMA-based training to hyperparameter choices.

Augmentation Asymmetry Following practices from prior work in SSL [25], different augmentation strategies were applied to the student and teacher inputs. The student model receives stronger or more varied augmentations, encouraging it to develop invariance and robust representations. Meanwhile, the teacher is exposed to milder or canonical views, ensuring the targets remain stable and semantically meaningful. This asymmetry improves training dynamics and helps the student generalize beyond superficial image variations.

Curriculum-Driven Pseudo-Label Selection Instead of providing all pseudo labels from the teacher from the start, a curriculum learning strategy [5] is employed. Early training epochs only use high-confidence teacher predictions to supervise the student, gradually relaxing the threshold to allow more diverse and potentially ambiguous pseudo labels.

The curriculum is encoded as a progressive scheduling function that determines how many teacher-predicted boxes are used per image at each epoch. This also implicitly addresses noise in early predictions by excluding uncertain or redundant proposals at the start of training.

To further improve robustness, the confidence threshold for accepting teacher predictions is computed dynamically for each batch using a percentile-based strategy:

- Scores from teacher predictions are aggregated and sorted.
- A threshold is computed as the average between the maximum score and the 75th percentile score (These values where chosen based on experiments).
- Only boxes above this threshold are passed to the student as pseudo labels.

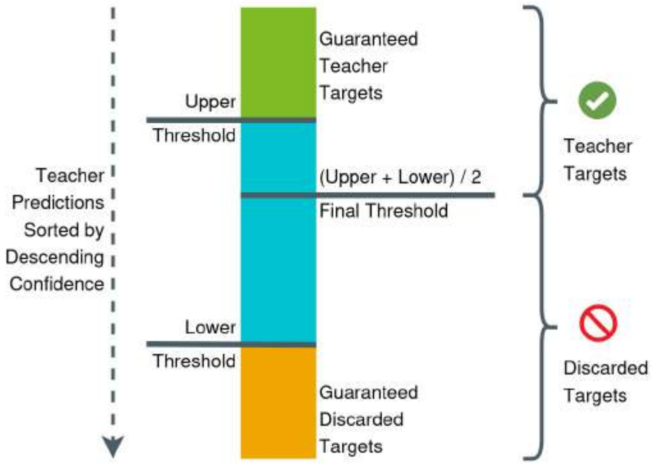

Teacher Predictions Sorted by Descending Confidence
Upper Threshold
Lower Threshold
Guaranteed Teacher Targets
(Upper + Lower) / 2
Final Threshold
Guaranteed Discarded Targets
Teacher Targets
Discarded Targets

Figure 4.8: Example for dynamic thresholding.

Progressive Loss Scaling Training is supervised by a combination of two loss terms: a custom loss based purely on box regression (see Section 4.2.6), and the standard DDETR loss [86], which includes classification. In early epochs, the classification component may be too unreliable due to noisy pseudo labels. To address this, a progressive scaling schedule is applied to the default loss. Specifically, the loss is scaled by a factor of 0.1 in the first epoch, 0.2 in the second, and so forth which allows the model to focus initially on spatial localization before gradually introducing classification learning. This incremental weighting ensures that the model builds a robust spatial understanding before engaging in the more error-prone task of label discrimination under noisy supervision.

Learning Rate Scheduling A cosine decay schedule [44] is applied to the learning rate, allowing for larger updates at the start and more refined tuning toward the end of training. This dynamic schedule facilitates better exploration of the parameter space in early stages and reduces oscillations as the model converges, which is particularly beneficial for self-supervised objectives that rely on noisy pseudo labels.

Simplified Student-Teacher Evaluation To monitor whether the student is successfully imitating the teacher or collapsing entirely, a simple safety metric is computed at the end of each epoch. On a small set of images from the target domain (KIhUG dataset), both models make predictions using the same input. The students predicted boxes are compared against the teachers pseudo boxes using intersection-over-union and a fixed matching strategy. The percentage of matched boxes is logged. This serves as a safeguard to detect undesirable training behaviors such as representational collapse (close to 0% match rate) or excessive copying (100% matches too early). However, it is not used as a formal performance metric and does not evaluate detection quality.

Motivation and Benefits This EMA-based teacher-student setup offers a lightweight and stable approach for learning domain-relevant representations without labels. It complements the architectural constraints introduced by the reduced model size and adapter-based fine-tuning, forming a cohesive strategy for low-resource transfer learning. The use of curriculum learning, adaptive thresholds, dynamic loss scaling, and cosine learning rate scheduling creates a well-aligned training regime that improves stability, reduces overfitting, and accelerates convergence.

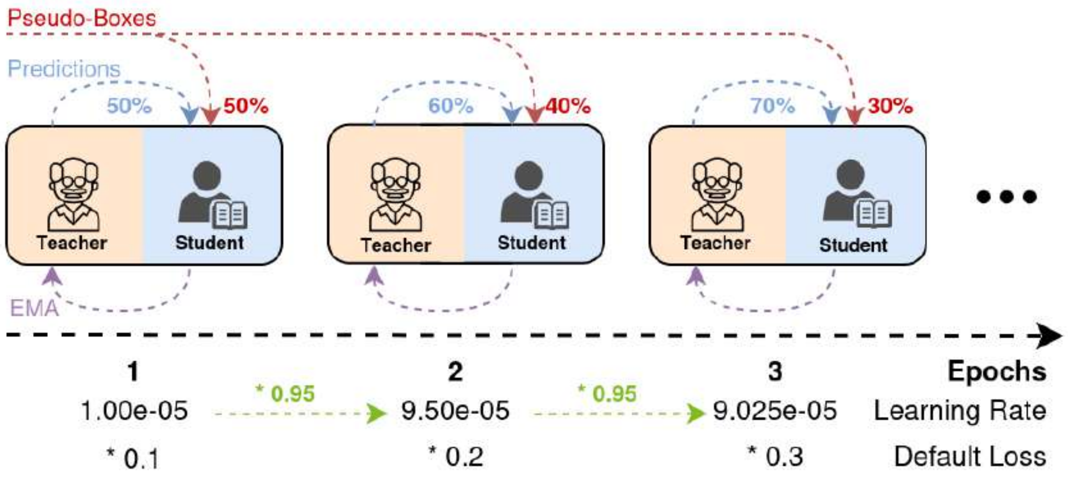


Figure 4.9: Visualization of the SSL training strategy combining EMA teacher-student learning with curriculum scheduling. Each epoch block illustrates how the student learns from pseudo labels generated by the EMA teacher. The share of teacher-labeled data increases over epochs (e.g., 25% 50% 75%), while the learning rate is decreased progressively. Additionally, the influence of the standard DDETR (which includes classification) is gradually increased via an epoch-dependent weight multiplier. This approach stabilizes training and gradually refines the models internal representations while softly introducing the classification task.

4.2.6 Loss Functions and Optimization Techniques

The loss function used during self-supervised training combines multiple components, each designed to optimize a specific part of the prediction task. As described in Section 4.2.5, a curriculum learning strategy gradually increases the influence of the default DDETR loss during early training epochs.

Custom Intersection over Union (IoU)-Based Localization Loss A custom matching loss based on spatial alignment is used to supervise the student model's bounding box predictions. For each target box in the pseudo-label set, the students best-matching predicted box (based on maximum IoU) is selected.

The final localization loss is the sum of:

- Mean absolute error (L1 loss) between matched box pairs in center-size format.
- Generalized IoU (GIoU) loss [58] between the same boxes converted to corner format.

Only matched pairs with an IoU above 0.5 are included in the loss calculation, a threshold commonly used in object detection benchmarks to define valid matches [41]. This focus on sufficiently overlapping predictions helps avoid penalizing the model for unmatched or low-confidence boxes during early stages.

Curriculum-Weighted Default Loss In parallel, the default DDETR loss [86] is included with a gradually increasing weight over training epochs. This loss encompasses classification, bounding box regression, and auxiliary losses from intermediate decoder layers. These auxiliary losses help stabilize optimization by enforcing useful gradients throughout the entire decoder stack, not just the final output layer. As part of the teacher-student curriculum (Section 4.2.5), the classification component starts with low influence and grows in importance as the student becomes more aligned with the teacher.

Optimization Strategy The AdamW optimizer [45] is used together with a cosine learning rate schedule [44]. This combination balances early exploration and late-stage convergence, and is well-suited for noisy supervision signals and fine-grained learning dynamics.

Additional loss terms used as regularizers are discussed in the Section 4.2.8.

4.2.7 Collapse Risks in Self-Supervised Training

SSL with pseudo labels introduces failure modes that differ from those in supervised training. These issues can arise from architectural limitations, loss design, spectral properties of learned representations [79], or unstable dynamics in EMA-based frameworks [7]. They include both degenerate optimization behaviors and subtle dynamics linked to noisy supervision or unstable signals. While many are known from general SSL literature, such as representation collapse in BYOL [25], overconfidence in FixMatch [65], and teacher-student drift in EMA models [68], they manifest differently in detection tasks.

Table 4.1 provides an overview of observed risks. Each row lists a collapse mode, its training stage, and a brief description. Though not all issues are unique to this work, they were empirically encountered. Some problems – like underconfident predictions or teacher–student agreement without learning – are especially common in dense prediction tasks with weak visual signal.

Following the risk overview, a set of mitigation strategies is presented to address these failure modes.

Risk or Collapse Mode	**During**	**Description**
Representation Collapse	SSL	The student model converges to trivial or uniform outputs, failing to encode meaningful features. This often occurs when entropy is not encouraged and diversity is lost.
Overconfident Predictions	SSL	The model becomes too certain about noisy pseudo labels early in training, locking in incorrect patterns and reducing generalization.
Underconfident Predictions	Both	The model avoids making strong predictions, hovering around < 0.5 logits to minimize classification loss. Common in ambiguous tasks with subtle class distinctions.
Overfitting to Noisy Labels	SSL	The model memorizes pseudo labels (including their systematic errors) instead of learning to generalize beyond them.
Degenerate Box Outputs	Both	The model exploits box loss functions by producing unrealistic boxes (extremely small or stretched), unless regularized properly.
Teacher-Student Drift	SSL	The EMA teacher and the student model diverge from one another, destabilizing the learning signal and breaking knowledge transfer.
Training Instability	Both	Caused by sudden loss spikes, poor scheduling, or unbalanced objectives, leading to divergence or degraded convergence.
Pseudo-Label Mimicry	SSL	The model learns to imitate the heuristics of the pseudo-label generator rather than develop general object understanding, leading to high loss performance but poor real-world generalization.
Exploration Decay	Both	Instead of discovering new signal, the model gradually loses confidence and predicts only background, especially in difficult or uncertain regions. This leads to poor feature diversity.

Table 4.1: Common risks and failure modes encountered during self-supervised object detection using pseudo labels. The **During** column indicates whether the issue can arise only during SSL or also during Fine-Tuning (Both). Some failure modes are well-documented in the literature on SSL (like representation collapse or teacher-student drift), while others were observed empirically in the specific context of domain-adapted or noisy-label training.

4.2.8 Regularization and Stabilization Techniques

To ensure stable training and avoid collapse during SSL, a combination of regularization techniques and architectural modifications were employed. The following paragraphs describe each strategy in detail, including both standard and custom components.

EMA Stability Considerations Recent work [7] suggests that the stability and effectiveness of EMA-based training is sensitive to the choice of decay rate τ, especially as model size and learning rate scale. While this thesis uses a fixed $\tau = 0.995$ during training (see Section 4.2.5), future work could explore adaptive schedules or temperature-aware scaling strategies to improve robustness.

Weight Decay To discourage overfitting and promote simpler models, a weight decay term is applied to the model parameters. The loss is defined as

$$L_{\text{weight decay}} = \lambda \|\theta\|^2$$

where θ denotes the model weights and λ is a small regularization constant. This penalizes large parameter values and encourages the model to distribute importance more evenly across the network.

Learning Rate Scheduling To guide optimization, a cosine learning rate schedule inspired by SGDR is employed:

$$\eta(t) = \eta_0 \cdot 0.9^{\lfloor t \rfloor}$$

where η_0 is the initial learning rate and t represents the current epoch. This decay schedule helps prevent training instabilities by allowing large updates early and gradually stabilizing learning in later stages.

Dropout Variants Dropout is used in multiple forms to regularize different parts of the network. In its classical form, dropout applies a random mask to neuron outputs:

$$y = x \cdot \text{mask}$$

where the `mask` is sampled from a Bernoulli distribution with dropout probability p. This is applied to the fully connected layers to prevent co-adaptation of features.

In attention mechanisms, *attention dropout* is applied to the attention weights rather than to the activations themselves. Similarly, *activation dropout* is employed at the output of activation layers to reduce overfitting by introducing controlled noise.

LayerDrop Inspired by structured dropout, LayerDrop randomly disables entire transformer layers during training:

$$\text{Layer}_l = \begin{cases} 1 & \text{with probability } 1-p \\ 0 & \text{with probability } p \end{cases}$$

As shown in [20], this encourages the network to learn redundant representations, which can later be pruned or compressed during deployment without retraining.

Auxiliary Losses The standard Deformable DETR loss includes auxiliary predictions at intermediate decoder layers. The total loss is given by:

$$L_{\text{total}} = L_{\text{det}} + \sum_{l=1}^{L_{\text{layers}}} \lambda_{\text{aux}} \cdot L_{\text{aux}}^{(l)}$$

as proposed by [86]. This strategy provides early supervision and smooth gradients to all decoder layers, facilitating convergence and reducing vanishing gradient problems.

Spectral Regularization Awareness Beyond structural and loss-based strategies, recent studies highlight the role of spectral properties in SSL collapse [79]. While this thesis does not explicitly modulate spectral responses, the introduced regularization techniques like entropy loss and orthogonal projection help indirectly encourage a diverse spectrum of feature responses.

Entropy Loss To avoid overconfident predictions and promote diversity in the output, an entropy-based regularization term is applied to the predicted class logits:

$$L_{\text{ent}} = -\left(\text{score}\log(\text{score} + 1e^{-8})\right) - (1 - \text{score})\log(1 - \text{score} + 1e^{-8})$$

This penalizes low-entropy outputs and prevents the model from collapsing into a degenerate solution where all predictions are constant or binary.

Box Shape and Size Regularization Two additional custom losses are used to prevent degenerate box predictions. The box size penalty is defined as:

$$L_{\text{size}} = \sum_i |\text{width}_i - \text{height}_i|$$

which encourages boxes to be approximately square, mitigating extreme aspect ratios.

The box shape penalty complements this by directly penalizing the aspect ratio:

$$L_{\text{shape}} = \sum_i \left(\frac{\text{width}_i}{\text{height}_i} - 1\right)^2$$

Together, these losses improve the plausibility and interpretability of box proposals and reduce the risk of degenerate boxes being favored during optimization.

Orthogonal Projection Loss To further encourage feature separability and improve clustering in the learned embeddings, an orthogonal projection loss is applied:

$$\mathcal{L}_{\text{OPL}} = (1 - s) + \gamma \cdot |d|$$

as introduced by [55]. This ensures that learned features span a broader, more useful space, promoting robustness in downstream tasks.

Box Jittering and Dropout To improve robustness and avoid overfitting to noisy pseudo boxes, spatial data augmentation is applied directly to the pseudo-labels. Box jittering randomly shifts and scales bounding boxes slightly, while box dropout randomly omits a subset of pseudo boxes before training. These strategies encourage the model to generalize beyond the exact geometry of the pseudo-labels, preventing memorization and reinforcing spatial invariance.

Student-Specific Augmentation In the teacher-student setup, predictions are generated by the teacher on already augmented images. After this step, additional student-specific augmentations (such as random color jitter) are applied before passing the image to the student model. This enforces consistency and encourages the student to learn invariance under different viewing conditions without affecting the pseudo-label quality.

4.2.9 Metrics

Evaluating self-supervised object detection models requires carefully chosen metrics that can meaningfully track learning progress and model quality in the absence of human-annotated labels. This section outlines the metrics used throughout the training and evaluation phases of this work and contrasts them with strategies used in related literature such as DETReg [3].

Comparison with Prior Work DETReg [3], a closely related self-supervised object detection method, does not evaluate its pretraining stage using standard object detection metrics. Instead, it assesses performance indirectly: the self-supervised model is used to initialize a downstream fine-tuning run, and the quality of the pretraining is inferred from improved convergence speed and higher mean Average Precision (mAP) values after fine-tuning. This makes it difficult to analyze or compare self-supervised representations in isolation.

During SSL Training: Consistency Check via IoU To monitor training progress and detect collapse during self-supervised training, a simple yet informative consistency metric is used. A fixed set of test images from the KIhUG dataset is passed through both the student and teacher model after each epoch. The predicted boxes are matched, and their Intersection-over-Union (IoU) is computed. The average IoU is logged as:

```
"avg_iou": 0.7495
```

This value reflects the alignment between teacher and student outputs. If the value stagnates near zero or one, it may indicate collapse (one model blindly copies the other, or both produce trivial predictions). A healthy SSL process maintains a moderate and gradually improving `avg_IoU`.

Additional statistics such as average L1 and GIoU loss, entropy, aspect ratio penalties, orthogonality losses, and cardinality error are also logged. These help track model behavior, identifying different kinds of model collapse and support debugging if necessary.

After SSL Training: Measuring Recall Although traditional metrics like Average Precision (AP) are not meaningful in the absence of labels, the final SSL model can be directly evaluated using COCO-style metrics [41] against the KIhUG dataset. While most AP values are expected to be low without supervised fine-tuning, recall remains a reliable metric to quantify how many relevant objects the model is able to detect without supervision.

An example output from the COCO evaluation tool shows:

```
"AR_100": 0.183, "Recall@IoU=0.5": 0.507, ...
```

These recall metrics offer valuable insight into the coverage of the pseudo-labeling system and demonstrate whether the model has learned any meaningful object localization ability.

During Fine-Tuning: Loss Monitoring During supervised fine-tuning, training and validation loss curves are logged for each epoch. These provide insight into convergence speed, stability, and potential overfitting. Because the model has already undergone SSL pretraining, the loss is expected to decrease faster than in a training-from-scratch setup.

After Fine-Tuning: Final COCO Evaluation Finally, the fine-tuned model is evaluated on the KIhUG test set using the standard COCO evaluation protocol. All COCO metrics are reported, including AP, AP_{50}, AP_{75}, and various recall values. These metrics allow for direct performance comparisons with other detection approaches.

4.2.10 Experimental Design and Training Plan

The following experimental plan outlines the key training runs used to evaluate the proposed SSL approach and its impact on downstream object detection performance. It is structured to allow both absolute and relative performance comparisons across different model variants and training configurations. Specifically, the experiments are divided into four categories: (i) SSL pretraining, (ii) standard baselines with or without COCO initialization, (iii) fine-tuning comparisons to test the effectiveness of SSL in low-data regimes, and (iv) adapter-based experiments, which isolate the benefits of training small subsets of parameters.

This plan provides the foundation for a structured and fair comparison of performance, generalization, and data efficiency, particularly focusing on whether SSL pretraining improves results over supervised baselines. A list of the first important runs (KIhUG dataset only) is given in Table 4.2, including identifiers, descriptions, and references to saved checkpoint and log files. These runs form the core of the evaluation presented in chapter 6.

ID	**Description**	**Checkpoint File**	**Log File**
	SSL Pretraining		
P1	SSL EMA curriculum pretraining, 2×Encoder/Decoder	ssl_pretrain.pth	ssl_pretrain.json
	Baseline (Finetuning without SSL Pretraining)		
B1	Default Deformable DETR, random init, full data	baseline_rand.pth	baseline_rand.json
B2	Default Deformable DETR, COCO pretrained, full data	baseline_coco.pth	baseline_coco.json
B3	Small Deformable DETR (2x2), COCO pretrained, full data	baseline_small.pth	baseline_small.json
	Finetuning Comparison (is SSL helpful?)		
C1	P1, Small D-DETR (2×2), SSL pretrained, full data	small_ssl.pth	small_ssl.json
B3.1	B3 with 50% data	small_coco_50.pth	small_coco_50.json
B3.2	B3 with 25% data	small_coco_25.pth	small_coco_25.json
B3.3	B3 with 10% data	small_coco_10.pth	small_coco_10.json
B3.4	B3 with few-shot (100)	small_coco_fs.pth	small_coco_fs.json
C1.1	C1 with 50% data	small_ssl_50.pth	small_ssl_50.json
C1.2	C1 with 25% data	small_ssl_25.pth	small_ssl_25.json
C1.3	C1 with 10% data	small_ssl_10.pth	small_ssl_10.json
C1.4	C1 with few-shot (100)	small_ssl_fs.pth	small_ssl_fs.json
	Adapter-Based Experiments (many more necessary		
A1	SSL EMA pretraining, 2×2 DETR with Adapters only	ssl_adapt.pth	ssl_adapt.json
A2	Finetune B2 (coco init full DETR) + Adapters only	baseline_adapt.pth	baseline_adapt.json
A3	Finetune A1 (SSL + Adapters), 2×2 + Adapters only	ssl_base_adapt.pth	ssl_base_adapt.json

Table 4.2: Rough overview of planned training runs (KIhUG only).

5 | Implementation

This chapter describes the technical realization of the methods outlined in Chapter 4, focusing on how the proposed training strategy was translated into a functioning and reproducible system. While the methodology chapter focused on conceptual aspects (such as the training loop, regularization techniques, and pseudo-label generation) this chapter covers the engineering details: code structure, execution flow, and integration with compute infrastructure.

The implementation is organized as a modular pipeline, with clearly separated components for data preprocessing, model loading, loss computation, training orchestration, evaluation, and logging. Emphasis is placed on flexibility and reproducibility: configuration files define each experiment in a declarative manner, and all training runs are launched in isolated environments via Simple Linux Utility for Resource Management (SLURM) job scripts.

Where applicable, this chapter explicitly references the design decisions presented in Chapter 4. For instance, the teacher−student setup described in Section 4.2.5 is matched by an EMA-based training script, and confidence scheduling mechanisms are implemented exactly as proposed in that section. The structure of this chapter roughly mirrors the chronological order of a typical experiment: starting with system setup and model configuration, continuing with the implementation of the self-supervised pretraining pipeline, and ending with supervised fine-tuning, evaluation tools, and practical concerns such as logging and monitoring. Code listings and figures are included where they help clarify implementation choices, but overly detailed or redundant listings are avoided in favor of conceptual transparency.

5.1 System and Environment Setup

This section provides an overview of the computational infrastructure and software environment used to execute the training pipelines. Emphasis is placed on reproducibility, modularity, and scalability. These are crucial aspects for managing complex training workflows, particularly under SSL regimes with large-scale data.

All experiments were conducted on a dedicated SLURM-managed cluster comprising two high-performance compute nodes. Each node was equipped with a modern multi-core Central Processing Unit (CPU), ample memory, and an NVIDIA GeForce RTX 4090 Graphics Processing Unit (GPU). Datasets were centrally stored on a 14 TB Network-Attached Storage (NAS) volume mounted to both nodes, enabling fast and shared access during training. The hardware configuration used throughout this work is summarized in Table 5.1.

To ensure consistent execution and reduce dependency conflicts, all training was performed inside containerized environments using SLURMs native container support. This setup guaranteed full Compute Unified Device Architecture (CUDA) acceleration for GPU workloads. Broadly used core libraries for Deep Learning (DL), data augmentation, and image processing (including PyTorch, Albumentations, OpenCV, and Pillow) were bundled within the container image. Table 5.2 outlines the key components of the used software stack.

Table 5.1: Hardware Configuration of the SLURM Cluster Node

Component	**Specification**
Hostname	`iti-dl-1.mni.thm.de`
Operating System	Ubuntu 24.04.1 LTS (Noble Numbat)
Kernel Version	5.15.0-140-generic
CPU	Intel Core i9-13900K (13th Gen), 32 threads
Cores / Threads	24 cores (P+E), 32 threads, 1 socket
Cache	L1: 896 KiB, L2: 32 MiB, L3: 36 MiB
Memory (RAM)	125 GiB total
Swap Space	8.0 GiB
GPU	NVIDIA GeForce RTX 4090, 24 GiB VRAM
CUDA Version	12.8
GPU Driver Version	570.133.20
Disk Storage	4.6 TiB (local), 14 TiB (NAS, 9.9 TiB used)
Mounted Datasets	`/datasets`

Table 5.2: Software Environment Inside the Training Container

Component	**Version / Details**
Container OS	Ubuntu 24.04.1 LTS
Python	3.12.3
PyTorch	2.6.0a0
CUDA Support in PyTorch	Enabled (12.6 runtime)
NVIDIA Container Toolkit	Enabled via SLURM
SLURM Job Management	Used for distributed job execution
Dataset Access	Direct mount of NAS: `/datasets`

5.2 Training Infrastructure and Script Structure

Figure 5.1 provides an overview of the complete training setup, including the two core workflows – self-supervised pretraining (left) and supervised fine-tuning (right). Each pipeline follows a modular structure, with separate scripts responsible for configuration loading, model setup, data handling, core training logic, and logging. Script dependencies are indicated by arrows, while color coding highlights functional roles across the system.

Experiment configuration is managed via a single YAML file that defines all relevant settings for both training stages. This includes hyperparameters, dataset splits, augmentation strategies, logging paths, and checkpoint management. The YAML file is parsed at runtime and replaces the need for complex command-line argument parsing. While this single-profile setup does not support simultaneous experiments, it offers clarity, traceability, and consistency for sequential runs.

```yaml
# === Paths and Directories ===
BASE_DIR: "/home/mlhn64/ssl_ddetr_mlhn64"
ROOT_DIR: "/datasets"

# === EMA Specific ===
EMA_EPOCHS: 6
EMA_LEARNING_RATE: 0.0001
EMA_DECAY: 0.999
TEACHER_TEMPERATURE: 1.1
STUDENT_TEMPERATURE: 1.0

# === Fine-tuning Specific ===
FINETUNE_EPOCHS: 50
FINETUNE_LEARNING_RATE: 0.00005
```

Listing 5.1: Example Snippet from `config.yaml`

Training jobs are launched using SLURM scripts that handle job scheduling, containerized execution, and dataset mounting. Each SLURM script wraps a specific entry point – either the self-supervised training script (`ema_ddetr.py`) or the fine-tuning script (`finetune_ddetr.py`) – within a reproducible execution environment. These job files, while simple, are essential for consistent deployment across shared computing infrastructure and are designed to minimize configuration drift.

For completeness, the full SLURM job scripts for self-supervised and fine-tuning stages are provided in Appendix A.2, along with the complete YAML configuration file used throughout the experiments in Appendix A.1.

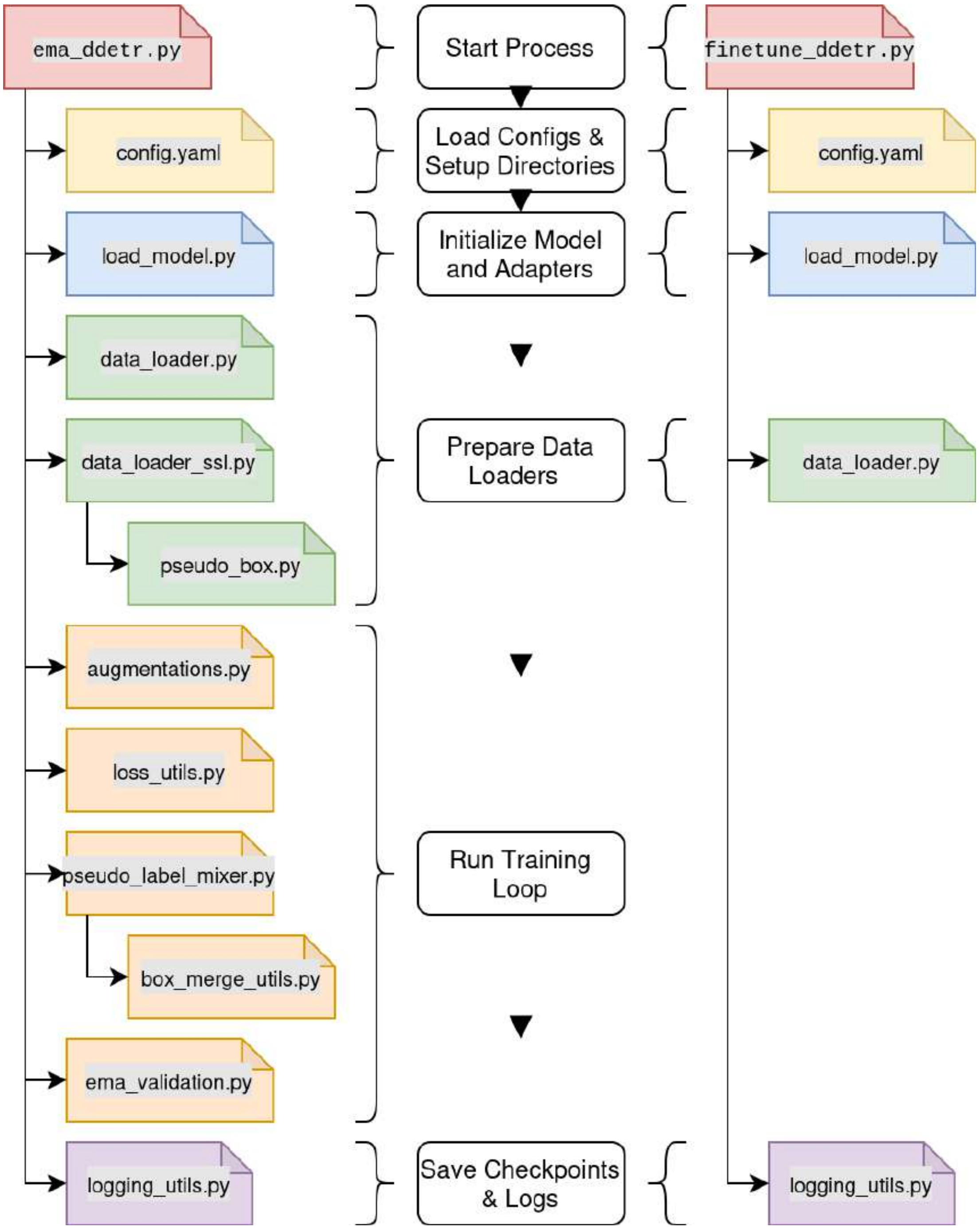


Figure 5.1: Schematic overview of the training pipelines and associated scripts. **Left:** Self-supervised pretraining based on an EMA teacher-student strategy with pseudo labels. **Right:** Supervised fine-tuning with ground truth labels. Functional components such as configuration loading, model preparation, data input, and logging are color-coded for clarity. Arrows denote script dependencies.

5.3 Model Integration and Adapter Setup

This section describes how the base DDETR model is extended for lightweight, parameter-efficient training via adapter injection. The modular script `load_model.py` consolidates all core logic, including model instantiation, adapter integration, checkpoint loading, and trainability settings. Adapters are inserted into both encoder and decoder Feedforward Neural Network (FFN) layers, allowing fine-grained updates without modifying the full model. A gated residual mechanism fuses adapter outputs with original activations, and optional alignment layers wrap the input projection to improve consistency between student and teacher features. The setup supports full fine-tuning or selective freezing of weights, controlled via configuration flags. Temperature scaling is optionally applied to classification logits during EMA-based training, as outlined in Section 4.2.5. Key features of `load_model.py` are:

- Loads DDETR from Hugging Face with optional checkpoint initialization.
- Injects lightweight adapters into all transformer FFN layers.
- Wraps input projection layers with feature alignment modules.
- Supports freezing most model weights while keeping adapters trainable.
- Includes optional logit temperature scaling for teacher-student setups.

```python
class Adapter(nn.Module):
    def __init__(self, input_dim, adapter_dim=64):
        super().__init__()
        self.down = nn.Linear(input_dim, adapter_dim)
        self.act = nn.ReLU()
        self.up = nn.Linear(adapter_dim, input_dim)

    def forward(self, x):
        return self.up(self.act(self.down(x)))

class ModifiedFFN(nn.Module):
    def __init__(self, original_fc, adapter):
        super().__init__()
        self.original_fc = original_fc
        self.adapter = adapter
        self.norm = nn.LayerNorm(original_fc.in_features)
        self.gate = nn.Parameter(torch.full((original_fc.in_features,)
            ↪ ,0.3))

    def forward(self, x):
        x = x + self.gate * self.adapter(x)
        x = self.norm(x)
        return self.original_fc(x)

def integrate_adapter(layer, target_attr, adapter_dim=64):
    orig = getattr(layer, target_attr)
    adapter = Adapter(orig.in_features, adapter_dim)
    modified = ModifiedFFN(orig, adapter)
    setattr(layer, target_attr, modified)
```

Listing 5.2: Injecting Adapters into Transformer Layers

5.4 Self-Supervised Pretraining Pipeline (EMA)

This section details the implementation of the self-supervised pretraining workflow built around a teacher-student architecture with EMA-based weight updates. The goal is to train the DDETR backbone on unlabeled, in-domain images using pseudo labels generated by the teacher model. This enables the model to learn robust, domain-specific object representations without requiring ground-truth annotations.

The training pipeline integrates several mechanisms to improve learning stability and data efficiency. These include curriculum-based pseudo-label scheduling, dynamic confidence thresholding, temperature-scaled logits, and auxiliary losses for semantic alignment and regularization. Key components like the pseudo label algorithm, loss functions or augmentation strategies are implemented in a modular way to allow flexible experimentation and targeted ablations.

The remainder of this section explains the key components of the pipeline and how they interact in the training loop. Additional implementation details on curriculum and threshold scheduling are referenced from Section 4.2.5.

5.4.1 Pseudo Label Generation

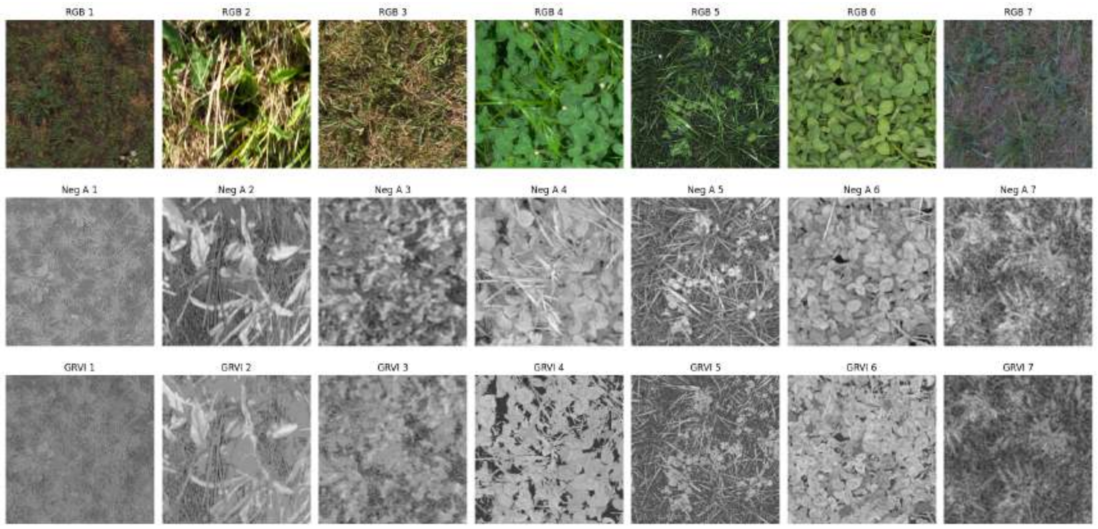


Figure 5.2: Visual comparison of seven image samples from different datasets used in this thesis. The first row shows the original RGB input. The second row presents the corresponding negated A-channel from the CIELAB (LAB) color space, highlighting foreground structures. The third row displays the GreenRed Vegetation Index (GRVI) maps, which emphasize vegetation similarly but often with stronger contrast. Both are computationally efficient and suitable for on-the-fly pseudo-label generation.

A central contribution of this thesis is the development of a fast and effective pseudo-label generation algorithm tailored to DETReg-style training. As outlined in Section 4.2.5, DETReg originally relied on Selective Search to generate object proposals. An approach that was critically questioned in earlier work and was considered inefficient in terms of localization in complex scenes [30].

During the exploratory phase of this work, several alternative pseudo-labeling strategies were considered. HyperPixel using SLIC superpixels showed promising segmentation quality but was ultimately excluded due to its computational overhead, which rendered it impractical for on-the-fly generation during training.

To enable scalable and real-time pseudo-label generation for SSL, a lightweight strategy was adopted based on color-space transformation and vegetation index analysis, as initially introduced in Section 4.2.2. Initially, converting input images to the LAB color space and negating the A-channel (which encodes the green-magenta axis) proved to be a simple yet effective method for isolating green plant structures. In parallel, the Green-Red Vegetation Index (GRVI), a variant often used in vegetation analysis, was evaluated. Interestingly, GRVI maps produced qualitatively similar foreground masks to the negated A-channel, but often with improved contrast − making them preferable in most scenarios.

Figure 5.2 illustrates a side-by-side comparison across seven representative samples from datasets used for SSL and fine-tuning. It shows the original RGB input (top), the negated LAB A-channel (middle), and the GRVI output (bottom), supporting the use of both techniques as viable preprocessing steps.

Building on these observations, a complete pseudo-labeling pipeline was implemented, transforming raw visual cues into binary masks and, ultimately, bounding boxes suitable for object detection pretraining. The full process is shown in Figure 5.3, which outlines each step in the pipeline: GRVI computation, total variation denoising, Otsu thresholding, morphological filtering, and final segmentation via watershed transformation. A slightly simplified version of the core logic (focusing only on the pseudo-label generation) is provided in Appendix A.3.

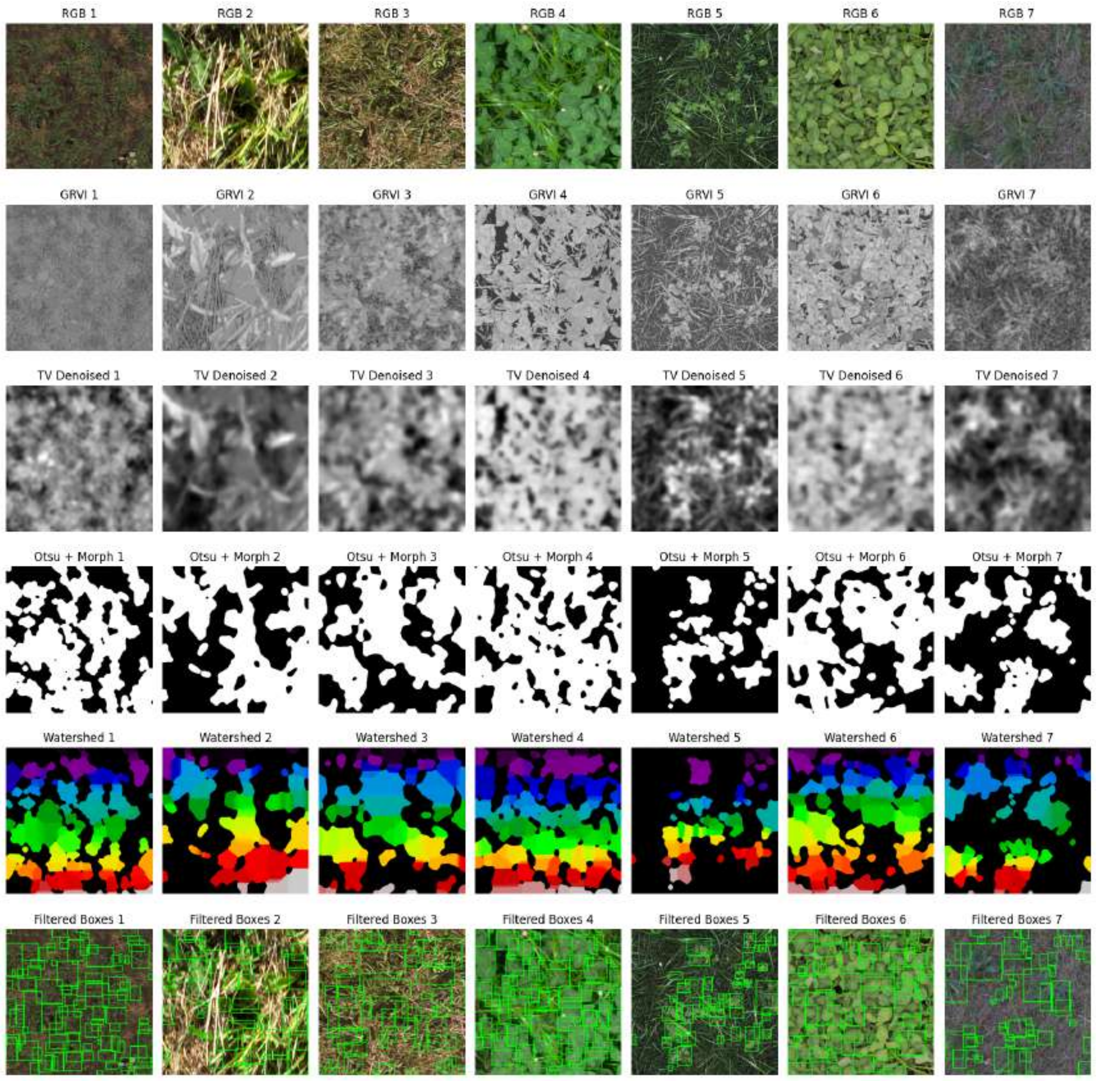


Figure 5.3: Visualization of the pseudo-label generation pipeline used during self-supervised training. Each column shows one dataset sample (used in either SSL or fine-tuning). Each row corresponds to a processing stage: from top to bottom – original image, GRVI transformation, total variation denoising, Otsu thresholding, morphological filtering (erosion/dilation), watershed segmentation, and final bounding box generation. The method is tailored for detecting green-on-green plant structures in natural grassland environments.

5.4.2 Teacher-Student Model Updates with EMA

To stabilize learning under weak supervision, this work employs a teacher-student architecture based on EMA updates [68, 25]. The teacher model is initialized as a frozen copy of the student and is not directly trained. Instead, it evolves as a smoothed version of the student throughout training, thereby serving as a more stable source of pseudo labels.

The teacher is initialized by copying all weights from the student and freezing its parameters, as shown in Listing 5.4.2. This ensures that the teacher model does not receive gradients and remains a lagged version of the student.

```
# Copy student weights to teacher model
for t_param, s_param in zip(
    teacher_model.parameters(),
    student_model.parameters()
):
    t_param.data.copy_(s_param.data)
    t_param.requires_grad = False
```

Listing 5.3: One-time Initialization of the Teacher-Model

```
@torch.no_grad()
def update_teacher_weights(teacher, student, decay):
    for t_param, s_param in zip(teacher.parameters(), student.parameters
        ↪ ()):
        t_param.data = decay * t_param.data + (1.0 - decay) * s_param.
            ↪ data
```

Listing 5.4: EMA Update to Synchronize Teacher with Student

During training, after every student update, the teacher is synchronized using the EMA rule. This is implemented in the function shown in Listing 5.4.2. The update rule can be written as:

$$\theta_t \leftarrow \alpha \cdot \theta_t + (1 - \alpha) \cdot \theta_s$$

where θ_t and θ_s are the teacher and student weights respectively, and α is a decay constant (typically set between 0.95 and 0.999). This formulation ensures that the teacher adapts gradually, filtering out noise and short-term variance from the student's updates. In this implementation, α is configurable via the config YAML file, as shown in Appendix A.1.

This EMA-based teacher mechanism is critical for robust pseudo-label generation, especially in early epochs where the students outputs are unstable. By maintaining a slowly updating teacher, the training pipeline benefits from smoother supervision and improved convergence behavior.

5.4.3 Dynamic Thresholding and Curriculum Scheduling

This work replaces static thresholds and fixed schedules with dynamic, data-driven alternatives to improve robustness during SSL. The confidence threshold for accepting teacher predictions is recalculated on a per-batch basis from the distribution of predicted logits. As shown in Listing 5.6, the threshold is computed as the average between the maximum and 75th-percentile confidence scores across all predictions. This strategy allows the system to adaptively respond to the evolving reliability of the teacher model.

In parallel, a curriculum mechanism regulates the mixing of teacher and pseudo labels. Early in training, a higher proportion of pseudo boxes is favored, while teacher predictions are gradually phased in as the model stabilizes. This ratio-controlled mixing is implemented in Listing 5.5, which samples and merges the appropriate number of boxes from each source.

```python
def mix_teacher_pseudo_boxes(
    teacher_boxes, teacher_labels,
    pseudo_boxes, pseudo_labels,
    teacher_ratio, pseudo_ratio,
    device
):
    # Sample fixed ratio of teacher and pseudo boxes
    num_teacher = int(teacher_ratio * len(teacher_boxes))
    num_pseudo = int(pseudo_ratio * len(pseudo_boxes))

    t_sample = random.sample(
        list(zip(teacher_boxes, teacher_labels)),
        k=min(num_teacher, len(teacher_boxes))
    )
    p_sample = random.sample(
        list(zip(pseudo_boxes, pseudo_labels)),
        k=min(num_pseudo, len(pseudo_boxes))
    )

    combined = t_sample + p_sample
    random.shuffle(combined)

    # Return mixed boxes and labels
    boxes_combined = torch.tensor(
        [b for b, _ in combined],
        dtype=torch.float32, device=device
    )
    labels_combined = torch.tensor(
        [l for _, l in combined],
        dtype=torch.int64, device=device
    )

    return boxes_combined, labels_combined
```

Listing 5.5: Combining Teacher/Pseudo Boxes According to Curriculum Ratios

```python
# Flatten teacher logits across batch and compute scores
filtered_teacher_scores = [
    torch.sigmoid(teacher_logits[i])[:,0] for i in range(len(
        ↪ teacher_boxes))
]
all_teacher_scores = torch.cat(filtered_teacher_scores)

# Sort and calculate dynamic threshold
sorted_scores, _ = torch.sort(all_teacher_scores, descending=True)
middle_idx = int(len(sorted_scores) * 0.75)
highest_score = sorted_scores[0]
middle_score = sorted_scores[middle_idx]
CONF_THRESHOLD = (highest_score + middle_score) / 2
```

Listing 5.6: Computing a Dynamic Confidence Threshold from Teacher Scores

5.4.4 Regularization and Collapse Prevention Strategies

Training a student model using pseudo labels, like those derived from a dynamically updating teacher, carries a high risk of collapse. This can manifest as overconfident but incorrect predictions, loss of box diversity, or convergence to degenerate solutions (like empty boxes or extreme aspect ratios). To mitigate these issues, this work employs a combination of regularization losses and aggressive augmentation strategies.

Loss-Based Regularization The student model includes several regularization terms in addition to the base detection loss. Orthogonality constraints are applied to both adapter modules and feature alignment layers, promoting diversity in learned representations. These constraints are enforced via the orthogonal projection loss introduced in [55] and described earlier in Section 4.2.8. This loss encourages decorrelated projection weights and prevents representational redundancy.

Entropy regularization is also applied to avoid early overconfidence. It promotes prediction uncertainty in the initial training stages, enabling the model to explore alternative hypotheses. Additionally, box geometry constraints are enforced to penalize extreme aspect ratios and outlier bounding box sizes. These priors help guide the student away from degenerate configurations, which is important when pseudo labels are noisy in early epochs.

Augmentation-Based Stability Beyond loss functions, the pipeline includes augmentations tailored to stabilize training. Before being used for supervision, the mixed set of EMA-based teacher boxes and pseudo boxes is perturbed using spatial jitter and box dropout. These introduce stochasticity, discourage overfitting, and simulate label noise.

Further, teacher and student models receive different augmented views of the same image. This is achieved by applying a dedicated set of image-level transformations (such as color jitter, blur, and channel shifts) only to the students input, using the Albumentations library [8]. This asymmetry reinforces consistency by requiring the student to match teacher outputs across visual diverse inputs.

```
# Apply jitter to box in [cx, cy, w, h] format
cx += np.random.uniform(-w * jitter_ratio, w * jitter_ratio)
cy += np.random.uniform(-h * jitter_ratio, h * jitter_ratio)
w += np.random.uniform(-w * jitter_ratio, w * jitter_ratio)
h += np.random.uniform(-h * jitter_ratio, h * jitter_ratio)

# Clamp to [0, 1] and recompute box
x1 = np.clip(cx - 0.5 * w, 0.0, 1.0)
y1 = np.clip(cy - 0.5 * h, 0.0, 1.0)
x2 = np.clip(cx + 0.5 * w, 0.0, 1.0)
y2 = np.clip(cy + 0.5 * h, 0.0, 1.0)

# Final box
cx = (x1 + x2) / 2
cy = (y1 + y2) / 2
w = x2 - x1
h = y2 - y1
```

Listing 5.7: Computing Jittered Bounding Boxes in Normalized Coordinates

```
def evaluate_iou_alignment(student_model, teacher_model, dataloader,
    ↪ device):
    student_model.eval()
    teacher_model.eval()
    total_iou, total_gt, total_matched = 0.0, 0, 0

    with torch.no_grad():
        for images, _ in dataloader:
            images = images.to(device)
            s_boxes = student_model(images).pred_boxes
            t_boxes = teacher_model(images).pred_boxes

            for s_b, t_b in zip(s_boxes, t_boxes):
                s_xyxy = cxcywh_to_xyxy(s_b)
                t_xyxy = cxcywh_to_xyxy(t_b)
                ious = box_iou(s_xyxy, t_xyxy)
                best_ious, _ = ious.max(dim=0)

                total_iou += best_ious.sum().item()
                total_matched += (best_ious > 0.3).sum().item()
                total_gt += best_ious.numel()

    return {
        "avg_iou": total_iou / total_gt,
        "iou>0.3": total_matched / total_gt
    }
```

Listing 5.8: Evaluating Teacher-Student Alignment via IoU at Each Epoch

Monitoring for Collapse To detect model drift and collapse, this implementation evaluates IoU-based statistics between teacher and student predictions at the end of each epoch. While not used to alter training dynamics directly, these metrics act as sanity checks to ensure continued alignment between models.

Together, these regularization and monitoring strategies form a comprehensive defense against collapse. By combining principled loss functions with targeted augmentations and diagnostics, the training loop remains robust even under weak supervision constraints.

```python
# Extract classification scores and boxes
logits = outputs.logits             # [B, num_queries, 1]
boxes = outputs.pred_boxes          # [B, num_queries, 4]
scores = torch.sigmoid(logits)      # class-0 probability

# --- Entropy Regularization ---
# Penalizes overconfident predictions to promote exploration
entropy = - (scores * torch.log(scores + 1e-8) +
             (1 - scores) * torch.log(1 - scores + 1e-8)).mean()

# --- Aspect Ratio Penalty ---
# Discourages extreme box shapes
widths = (boxes[..., 2] - boxes[..., 0]).clamp(min=1e-4)
heights = (boxes[..., 3] - boxes[..., 1]).clamp(min=1e-4)
aspect_ratio = widths / heights
aspect_reg = ((aspect_ratio < 0.1)|(aspect_ratio > 10)).float().mean()

# --- Size Regularization ---
# Penalizes boxes that deviate too much from a target size
target_size = 0.2
penalty_scale = 2.0
size_reg_w = penalty_scale * ((widths - target_size) ** 2).mean()
size_reg_h = penalty_scale * ((heights - target_size) ** 2).mean()
size_reg = size_reg_w + size_reg_h

# --- Adapter Orthogonality Loss ---
# Promotes diversity in adapter projection weights
orth_loss_adapters = 0.0
for module in student_model.modules():
    if isinstance(module, Adapter):
        orth_loss_adapters += compute_weight_orthogonality_loss(
            module.down, gamma=GAMMA_ADAPTERS
        )

# --- Feature Aligner Orthogonality Loss ---
# Same idea as above, but for pre-adapter aligners
orth_loss_aligners = 0.0
for module in student_model.modules():
    if isinstance(module, PreAdapterFeatureAligner):
        orth_loss_aligners += compute_weight_orthogonality_loss(
            module.proj, gamma=GAMMA_ALIGNERS
        )

# --- Combine into Total Regularization Loss ---
reg_loss = (
    0.5 * aspect_reg +
    0.5 * size_reg +
    1.0 * entropy +
    1.0 * orth_loss_adapters +
    1.0 * orth_loss_aligners
)
```

Listing 5.9: Loss Terms for Entropy, Box Geometry, and Adapter Orthogonality

5.5 Supervised Fine-Tuning

The pretrained model is transferred to the target domain through fully supervised learning on manually annotated bounding boxes. The existing training framework, including dataset interfaces, configuration parsing, logging utilities, and the core training loop, is reused to ensure simplicity, consistency, and reproducibility across both self-supervised and supervised stages.

No novel algorithms are introduced at this stage. Depending on the chosen configuration, either the entire DDETR network is finetuned or only its lightweight adapter modules are updated via residual connections. In adapter-only mode, all backbone weights remain frozen and only the adapter parameters undergo optimization, yielding an efficient approach for lowdata regimes.

To mitigate the instability of oneclass DETR heads-where low initial confidence can cause the Hungarian matcher to find no positives and stall trainingan optional warmup phase is applied. During the first T epochs, the overall loss is computed as a linear interpolation between the standard classification loss and a reduced initial weight, gradually increasing the classification component until the full loss formulation is restored.

- A custom box-only loss (ℓ_1 + GIoU), which bootstraps localization regardless of confidence
- The standard DETR loss (classification + box + GIoU), introduced gradually to build reliable confidence

Once the warm-up is complete ($t > T$), everything is switched to 100% standard DETR loss. Empirically this prevents under-confidence collapse early on and yields markedly better convergence.

Listing 5.10 shows the core of this strategy; note that setting $T = 0$ entirely skips the warm-up and reduces the loop to the usual DETR fine-tuning.

```
for epoch in range(1, EPOCHS + 1):
    model.train()
    for images, targets in dataloader_train:
        images = images.to(device)
        targets = [{"class_labels": t["labels"].to(device),
                    "boxes": t["boxes"].to(device)} for t in targets]

        optimizer.zero_grad()
        loss = model(images, labels=targets).loss
        loss.backward()
        optimizer.step()

    scheduler.step()
    torch.save(model.state_dict(), CHECKPOINT_PATH)
```

Listing 5.10: Supervised Fine-Tuning Using Ground-Truth Annotations

```
# Warm-up weight computation (inside epoch loop)
if epoch <= WARMUP_EPOCHS:
    w_custom = 1.0 - (epoch - 1) / WARMUP_EPOCHS
    w_detr   = 1.0 - w_custom
    # optional: anneal focal alpha
    model.config.focal_alpha = ALPHA_START + (epoch - 1) / WARMUP_EPOCHS
        ↪ * (ALPHA_END - ALPHA_START)
else:
    w_custom, w_detr = 0.0, 1.0
```

Listing 5.11: Supervised Fine-Tuning Using Ground-Truth Annotations

5.6 Evaluation and Visualization Tools

This section outlines the tools and procedures used to quantitatively and qualitatively assess model performance. Evaluation is conducted using standardized COCO metrics to ensure comparability across training phases, while custom visualization scripts offer insights into prediction quality and training dynamics.

5.6.1 Evaluation Strategy and COCO Metrics

To evaluate detection performance consistently across both the self-supervised pretraining and supervised fine-tuning phases, this work uses a standardized evaluation pipeline based on the official `pycocotools` API. All predictions are exported in COCO-compatible JSON format and processed using the same script to ensure metric comparability across training stages.

The evaluation computes standard COCO metrics, including mAP, AP at specific IoU thresholds (like AP_{50} or AP_{75}), and Average Recall (AR). To better track alignment performance during early training, the script also reports recall at additional thresholds (e.g., $\text{IoU} = 0.3, 0.5, 0.7$).

A single evaluation script is reused across all experiments by modifying only the input checkpoint and dataset split, reinforcing consistency and reproducibility. The core evaluation logic is illustrated in Listing 5.12.

```
from pycocotools.coco import COCO
from pycocotools.cocoeval import COCOeval

# Load COCO-format ground truth and predictions
coco = COCO("coco_gt.json")
coco_dt = coco.loadRes("coco_dt.json")

# Run standard COCO evaluation
evaluator = COCOeval(coco, coco_dt, iouType="bbox")
evaluator.evaluate()
evaluator.accumulate()
evaluator.summarize()
```

Listing 5.12: Standardized COCO Evaluation Used Across Training Stages

5.6.2 Visualization of Predictions

To qualitatively assess training dynamics and supervision quality, custom visualization tools were implemented for both the self-supervised and supervised training phases. Each tool corresponds to one of the two implemented data loaders (`data_loader_ssl.py` and `data_loader.py`) and supports flexible inspection of model predictions, pseudo labels, and ground-truth annotations.

Both visualization scripts share a common set of functionalities: they display image batches in a grid (typically 3×2, adjustable via batch size) and allow interactive toggling between different box sources. These include no boxes, pseudo labels (or ground truth), model predictions, or overlays of both. Additionally, data augmentations can be toggled on or off via loader parameters, enabling analysis of their visual impact. The visualizations are primarily used as a qualitative tool to identify issues such as label noise, overconfident predictions, and domain shift effects. While this section presents an overview figure for illustration purposes, these tools are used extensively throughout the results and discussion chapter to support the interpretation of experimental findings and training behavior.

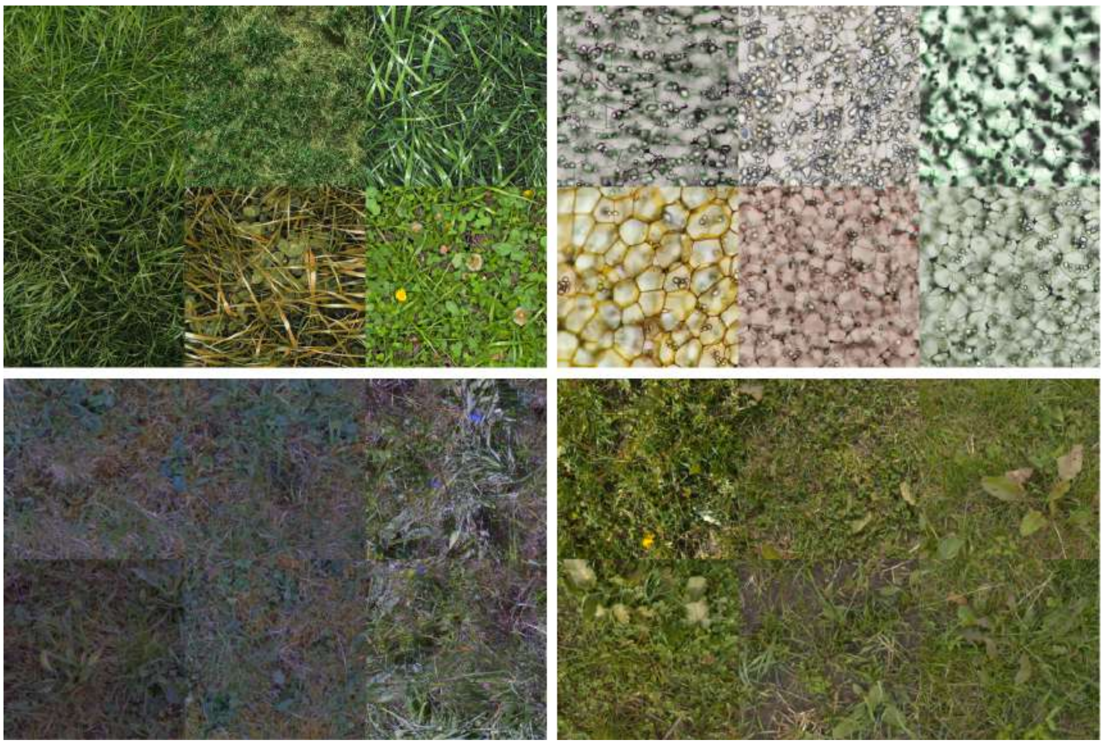


Figure 5.4: Visual overview of batch visualizations used during training and evaluation. **Top row:** images from the SSL dataset collection (left) and the potato tuber dataset (right). **Bottom row:** images from the KIhUG *Jacubaea vulgaris* dataset (left) and the Rumex weed dataset (right). No bounding boxes are shown in this overview to keep the presentation uncluttered. Boxes (pseudo labels, predictions, or both) can be interactively visualized using the tool described in this section.

6 | Results

This chapter presents the empirical results obtained from training and evaluating the proposed object detection model. It covers both quantitative metrics and qualitative visualizations for different model configurations. The focus lies on assessing the performance of the baseline fine-tuning approach, the effect of self-supervised pretraining, and the efficiency of adapter-based learning. Evaluations are conducted on the KIhUG and, where applicable, on the Rumex Weeds dataset. All results are interpreted in the context of the research questions outlined in section 1.2.

6.1 Performance Metrics and Evaluation Criteria

The evaluation of object detection performance in this thesis relies on two complementary forms of assessment: quantitative metrics computed using `pycocotools`[1] and qualitative visual inspection of predicted bounding boxes. Both methods were introduced in earlier chapters but are summarized here for clarity and completeness.

Quantitative results are reported using standard COCO evaluation metrics, which include mean mAP, precision, and recall across different IoU thresholds. Before evaluation, all model predictions are post-processed with NMS to remove duplicate boxes. These metrics allow consistent comparison across training configurations and datasets. Specifically, the following values are used:

- **Average Precision (AP)**: The area under the precision-recall curve. AP is averaged over multiple IoU thresholds (from 0.5 to 0.95 in steps of 0.05, COCO-style), referred to as:

$$\mathrm{mAP}_{50:95} = \frac{1}{10}\sum_{i=0}^{9} \mathrm{AP}_{0.5+0.05i}$$

- **AP@50** and **AP@75**: Average precision at fixed IoU thresholds of 0.5 and 0.75, representing more lenient and stricter matching conditions, respectively.
- **Precision**: The ratio of correct positive detections to all detections made:

$$\text{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

[1]`https://github.com/ppwwyyxx/cocoapi` (accessed July 2025).

- **Recall**: The ratio of correct positive detections to the total number of ground truth objects:
$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

`pycocotools` is also able to divide the ground-truth objects into size-based categories and evaluate them separately. While the COCO benchmarks define *small* as area $\leq 32^2$, *medium* as $32^2 <$ area $\leq 96^2$, and *large* as area $\geq 96^2$, these cutoffs suit everyday objects and not our plant imagery. Instead, we set dataset-specific thresholds at the 33rd and 67th percentiles of the *absolute* box areas:

$$0 \;\leq\; \textit{small} \;\leq\; A_{\text{sm}} \;<\; \textit{medium} \;\leq\; A_{\text{md}} \;<\; \textit{large} \;\leq\; \infty$$

KIhUG Dataset

$$A_{\text{sm}} = 06136.6\ \text{px}^2 \quad (33\,\%\text{ile})$$
$$A_{\text{md}} = 11090.8\ \text{px}^2 \quad (67\,\%\text{ile})$$

Rumex Weeds Dataset

$$A_{\text{sm}} = 26724.6\ \text{px}^2 \quad (33\,\%\text{ile})$$
$$A_{\text{md}} = 64444.7\ \text{px}^2 \quad (67\,\%\text{ile})$$

This data-driven binning ensures that each size category contains roughly one third of all boxes and yields stable, meaningful AP/AR measurements across the scales present in each dataset.

In addition to these metrics, self-supervised training results are evaluated indirectly by applying the pretrained model to the final annotated target dataset. The resulting recall provides insight into how well the domain-specific pretraining captured relevant patterns for the downstream task, even in the absence of supervised labels during the initial training phase.

Qualitative evaluation complements the numerical results by visualizing predictions on validation images. These examples help identify typical error patterns, verify detection coverage, and illustrate the impact of different model configurations.

6.2 Baseline Performance

To establish a reliable baseline for later comparisons, this section starts with an evaluation of three different configurations of the DDETR model, fine-tuned on the KIhUG dataset (auxiliary loss and the custom warm-up phase were *not* used in these experiments):

- the standard model with random initialization
- the same model initialized with COCO-pretrained weights
- a reduced model variant with only two encoder and decoder layers, also initialized from COCO

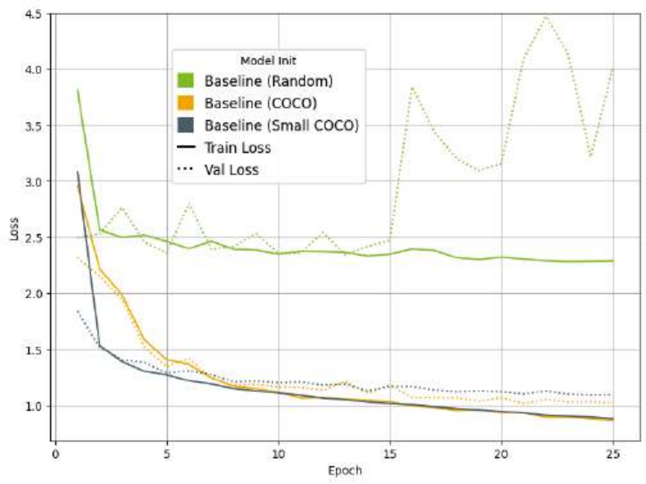


Figure 6.1: Training and validation loss curves over the first 25 epochs for three DDETR configurations on the KIhUG dataset. Green: randomly initialized model. Orange: COCO-pretrained baseline. Black: reduced model with 2 encoder/decoder layers. Note that auxiliary losses and the custom warm-up phase were disabled for this comparison.

As shown in Figure 6.1, the randomly initialized model fails to converge, highlighting the need for prior knowledge in low-data regimes. The COCO-pretrained baseline performs stably and achieves good results. Interestingly, the reduced model exhibits faster initial convergence and matches the performance of the full model despite its significantly smaller size. This finding supports its use throughout the remainder of this thesis as a more efficient alternative without sacrificing accuracy.

6.2.1 Fine-Tuning on the KIhUG Dataset

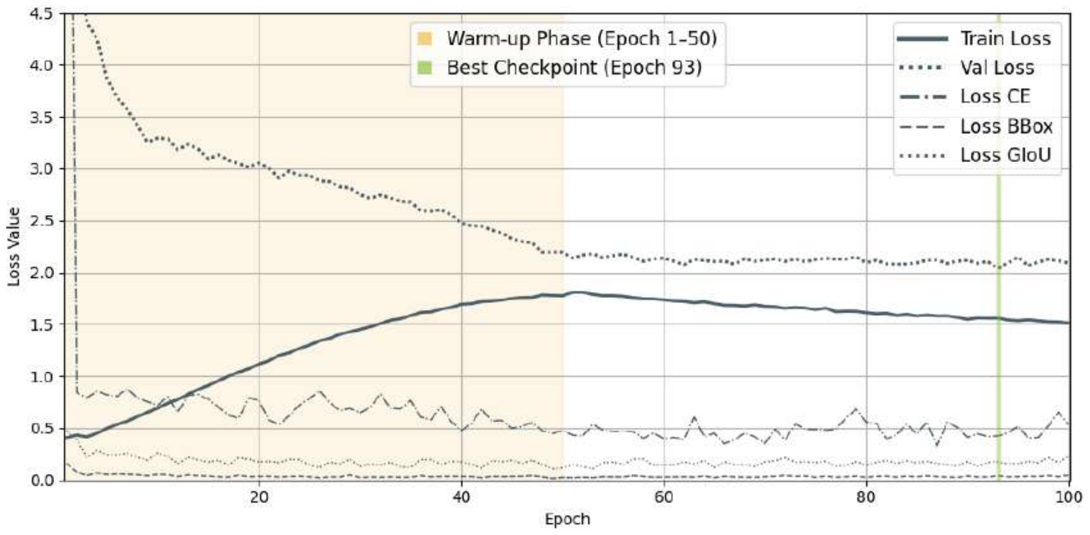
Figure 6.2 shows the full 100-epoch fine-tuning run of the depth-reduced DDETR on the KIhUG dataset. The shaded orange region marks the 50-epoch warm-up phase, during which a custom box-only loss is blended with the standard DETR loss to prevent under-confidence collapse; the green dashed line indicates the best-performing checkpoint at epoch 93. After warm-up, the model relies entirely on the standard classification+box+GIoU objective, with a smoothly decaying learning rate, and continues to improve until the final epochs.

In addition to the overall training and validation losses, the plot also breaks out the three components of the DETR loss:

- **Cross-entropy (CE)** classification loss,
- **BBox** (ℓ_1) bounding-box regression loss,
- **GIoU** loss for overlap maximization.

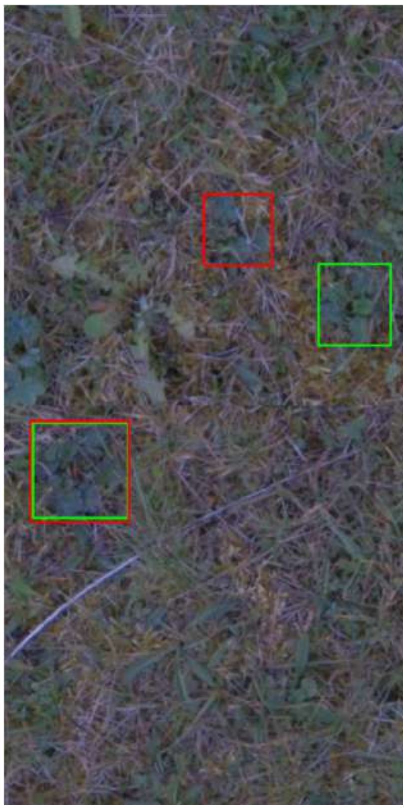
Each of these curves is multiplied by its respective weight in the matching and loss-weight configuration, so that their sum (together with the warm-up custom loss when active) yields the total training loss shown. This decomposition highlights how classification, localization, and overlap objectives evolve over the course of fine-tuning, and illustrates that gains in overall loss can stem from improvements in any one or a combination of these components.

Figure 6.2: Trainings plot of full baseline on KIhUG dataset.



Metric	Value
mean AP@[.50:.95] (mAP)	0.4537
mAP@IoU=0.50 (mAP50)	0.7928
mAP@IoU=0.75 (mAP75)	0.4670
AP (Small Objects)	0.1265
AP (Medium Objects)	0.4664
AP (Large Objects)	0.5461
Average Recall @ 1 Detection (AR1)	0.4683
Average Recall @ 10 Detections (AR10)	0.6064
Average Recall @ 100 Detections (AR100)	0.6157
AR (Small Objects)	0.3987
AR (Medium Objects)	0.6267
AR (Large Objects)	0.6730
Recall @ IoU=0.50	0.9769
Recall @ IoU=0.60	0.9122
Recall @ IoU=0.70	0.7290

(a) Quantitative evaluation metrics for the baseline DDETR model fine-tuned on the KIhUG dataset.



(b) Two qualitative results of the KIhUG baseline.

Figure 6.3: Performance of the baseline DDETR model on the KIhUG dataset. Model predictions are shown with red boxes, ground truth labels with green boxes. Small plants are often missed. Many false positives are actually accidentally unlabeled plants (see example in subplot b).

Table 6.3a summarizes the COCO-style evaluation on the held-out test set. The model achieves an overall AP@[.50:.95] of 0.4537 (AP@50 = 0.7928, AP@75 = 0.4670), with improved performance on small plants (AP = 0.1265) while maintaining strong results on large objects (AP = 0.5461). Recall at 1 detection is 0.4683, at 10 detections 0.6064, and at 100 detections 0.6157, indicating most plants are found when enough candidates are considered. Continuous recall at IoU thresholds of 0.50, 0.60, and 0.75 are 0.9769, 0.9122, and 0.7290, respectively. Notably, throughout training the classification (cross-entropy) loss remained consistently higher than both the 11 and GIoU terms (see the logs), highlighting that in these cluttered grassland scenes it is relatively easy to localize candidate regions but far harder to discriminate between plant structures and background or among similar plant species. This classification bottleneck was anticipated given the subtle visual differences and annotation noise in the dataset.

Figure 6.3b presents example detections (red) together with ground-truth boxes (green). Extensive qualitative visualization reviews showed, that the model reliably localizes medium and large plants, while missing many tiny specimens. Occasional false positives correspond to unlabeled plants in the annotations rather than outright errors, as shown in the figure. Together, these plots and tables establish a solid supervised baseline for subsequent comparisons.

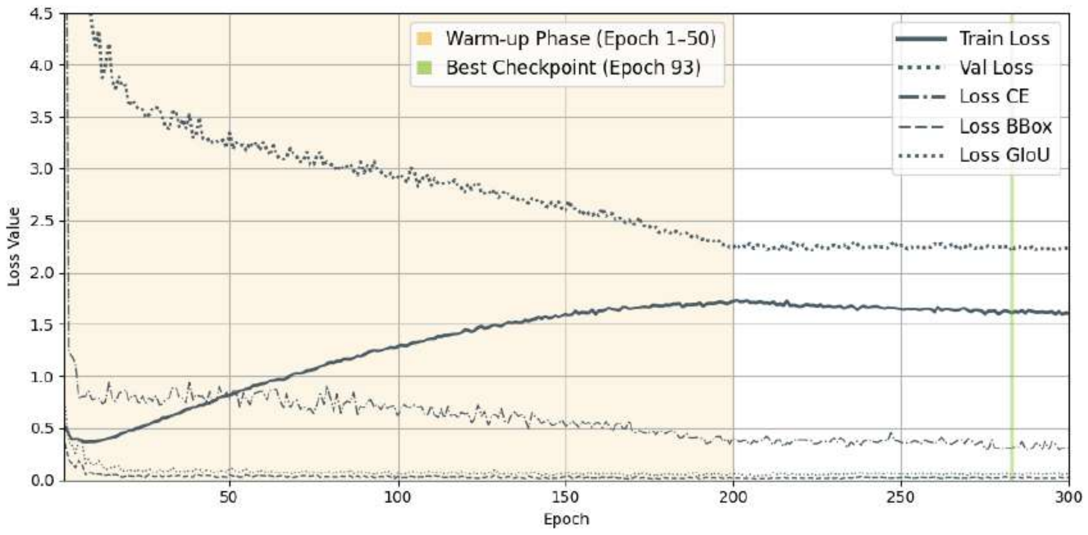
6.2.2 Fine-Tuning on the Rumex Weeds Dataset

To establish a comparative baseline, this section begins by referencing results reported in the original Rumex Weeds dataset paper [26]. The authors evaluated multiple YOLOX variantsranging from the lightweight YOLOX-nano to the deeper YOLOX-m on their weed detection benchmark. These models differ primarily in parameter count, computational cost, and overall inference speed. Their performance is summarized in Table 6.2.

Model	Params(M)	FLOPs(G)	mAP$_{50:95}$	mAP$_{50}$	Avg. Inference T. (ms)	
					V100	Jetson NX
YOLOX-m	25.3	73.5	27.8	54.1	14.7	48.7
YOLOX-s	8.9	26.6	27.1	51.4	12.2	25.9
YOLOX-tiny	5.0	15.1	26.9	52.8	11.1	21.4
YOLOX-nano	0.9	2.5	26.4	52.2	13.88	18.9

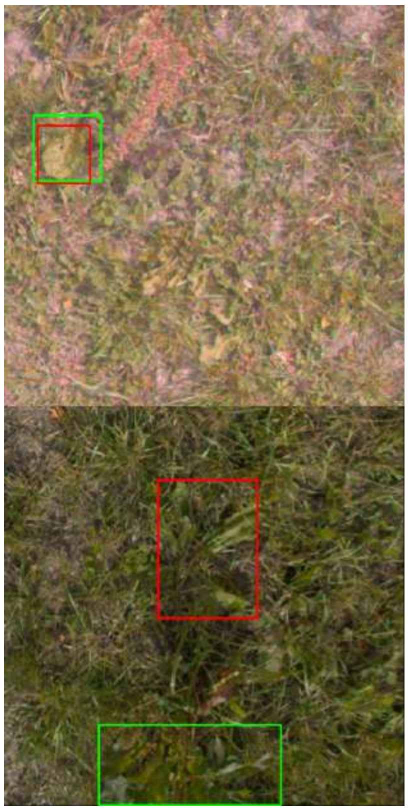
Table 6.2: Reported detection performance and inference speed of different YOLOX variants on the RumexWeeds dataset, reproduced from [26]. YOLOX-tiny is selected in the original work for its favorable trade-off between detection performance and real-time inference capability.

Figure 6.4: Trainings plot of full baseline on Rumex dataset.



Metric	Value
mean AP@[.50:.95] (mAP)	0.2749
mAP@IoU=0.50 (mAP50)	0.5672
mAP@IoU=0.75 (mAP75)	0.2356
AP (Small Objects)	0.1533
AP (Medium Objects)	0.4548
AP (Large Objects)	0.4587
Average Recall @ 1 Detection (AR1)	0.2750
Average Recall @ 10 Detections (AR10)	0.4717
Average Recall @ 100 Detections (AR100)	0.5118
AR (Small Objects)	0.4205
AR (Medium Objects)	0.6323
AR (Large Objects)	0.7038
Recall @ IoU=0.50	0.9265
Recall @ IoU=0.60	0.8195
Recall @ IoU=0.70	0.6032

(a) Quantitative evaluation metrics for the baseline DDETR model fine-tuned on the Rumex dataset.



(b) Two qualitative results of the RumexWeeds baseline.

Figure 6.5: Performance of the baseline DDETR model on the Rumex dataset. Model predictions are shown with red boxes, ground truth labels with green boxes. Many predictions are under-confident and are only visible by reducing the threshold for display in the visualization script.

While the original authors prioritized inference efficiency, this thesis uses the DDETR model for all experiments. Due to its significantly larger capacity and architectural advantages, it is expected to outperform the lightweight YOLOX variants. To establish a meaningful comparison, the model is fine-tuned on the RumexWeeds dataset using only the default COCO-pretrained weights, mirroring the baseline approach previously used for the KIhUG dataset. The losses during training progression are visualized in Fig. 6.4. The quantitative and qualitative results are shown in Fig. 6.5.

Despite its greater capacity, the DDETR baseline only marginally outperforms the original Rumex Weeds benchmarks. Several factors likely contribute to this result:

1. **Limited training data.** The Rumex datasets training split is smaller than KIhUG, offering fewer examples for the model to learn robust features.
2. **Extreme scale variance.** Rumex plants often occupy an entire 512×512 tile, resulting in very large bounding boxes that leave little background context. This contrasts with the smaller, more varied crops in KIhUG and can confuse the models localization and scale priors. This extreme variance is also detailed in Section 6.1, where custom size thresholds for use with `pycocotools` are defined.
3. **Ambiguous label boundaries.** In case of bigger plants, large, lobed leaves and overlapping canopies make it difficult to decide where one Rumex plant ends and another begins, introducing annotation noise that challenges the detector.

Together, these challenges constrain the improvements attainable by a high-capacity transformer on Rumex. However, at least the limited-data issue can be mitigated through self-supervised pretraining before the fine-tuning.

6.3 Self-Supervised Pretraining

This section presents the self-supervised pretraining (Section 4.2.5) outcome on unlabeled grassland and related data. We first examine convergence behavior, then show qualitative evidence of domain-aware feature learning, and finally quantify improvements over the raw pseudo-label generator.

Figure 6.6 confirms stable alignment between teacher and student across six epochs, with loss components smoothly balancing regression and location based terms while slowly introducing classification. The sustained IoU agreement (0.7-0.8) indicates successful avoidance of representational collapse.

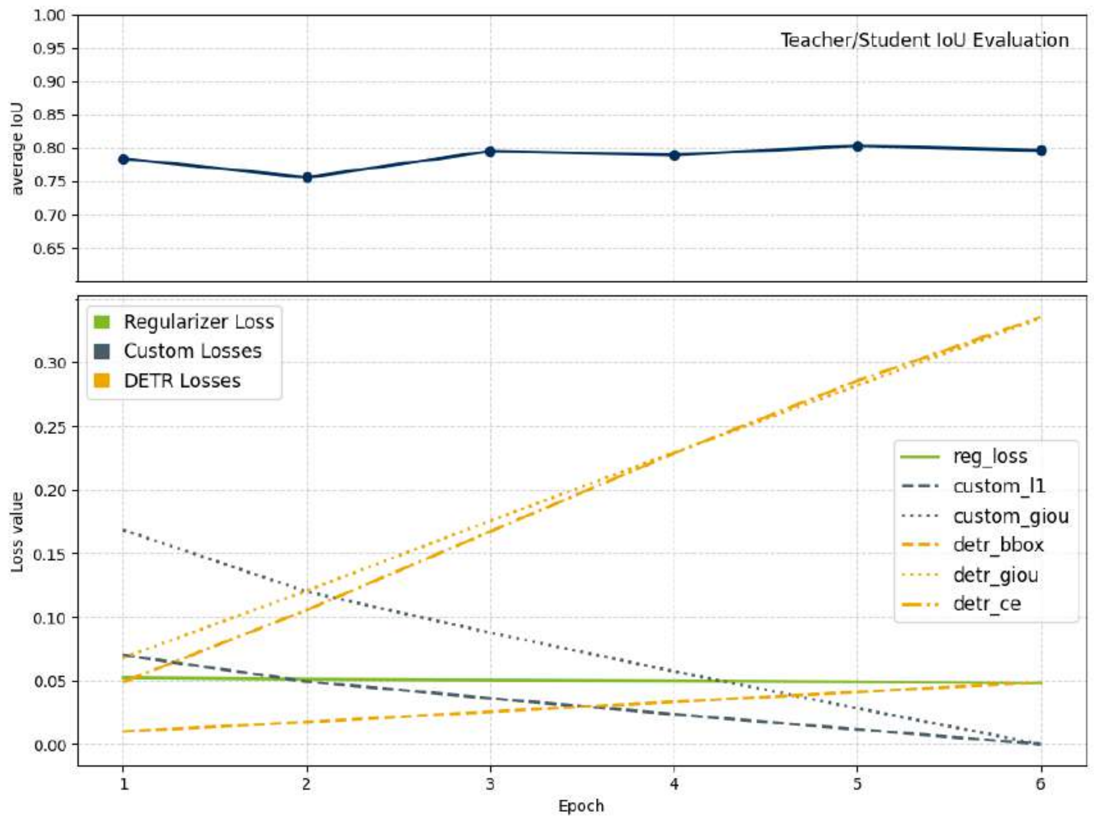
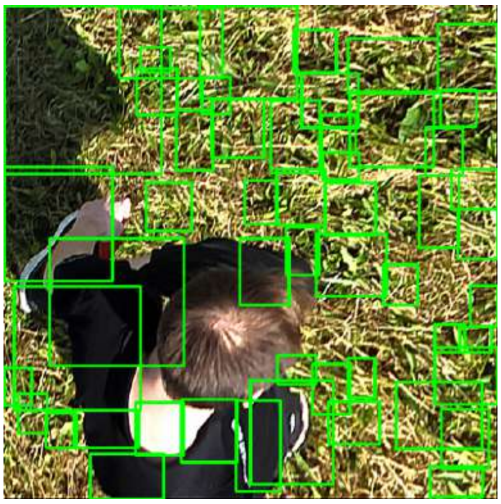


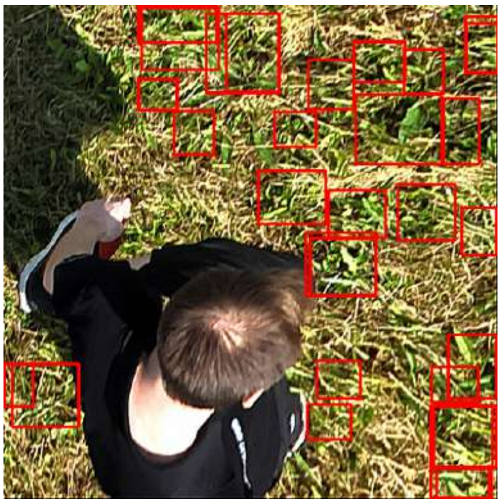
Figure 6.6: Self-supervised pretraining progress over 6 epochs. **Top:** Epoch-wise IoU agreement between teacher and student networksstability in the 0.7-0.8 range indicates successful feature alignment without collapse. **Bottom:** Total loss decomposition into custom location-only loss, standard DETR loss (classification + bbox + GIoU), and regularization terms, showing smooth convergence and balanced contributions from each component.

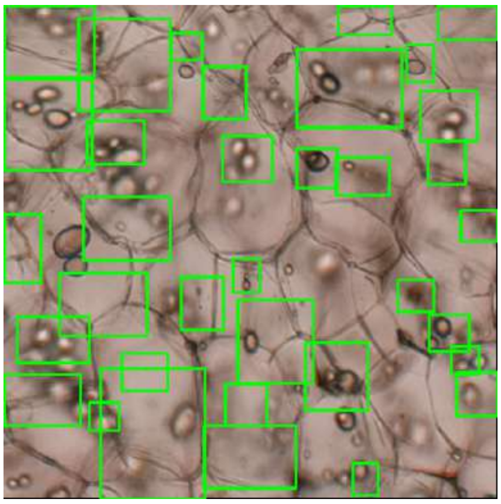
Beyond mere convergence, the model must learn semantic plant features rather than replicate low-level pseudo-label heuristics. Figure 6.7 contrasts raw algorithmic boxes with EMA-refined detections on both grassland and out-of-domain tuber images.

Encouraged by improved precision on pseudo samples, we next evaluate the pretrained detector qualitatively on unseen KIhUG validation images. Figure 6.8 illustrates that many target plants are correctly localized (despite a conservative confidence threshold) highlighting genuine feature understanding.

To quantify this refinement, Table 6.4 compares recall and AR@100 metrics between the raw pseudo-label generator and the EMA-trained detector on the KIhUG split. The substantial gains across IoU thresholds demonstrate that the SSL model converts noisy proposals into accurate plant detections.


(a) SSL pseudo-labels on grassland


(b) Pretrained model predictions


(c) SSL pseudo-labels on potato tubers

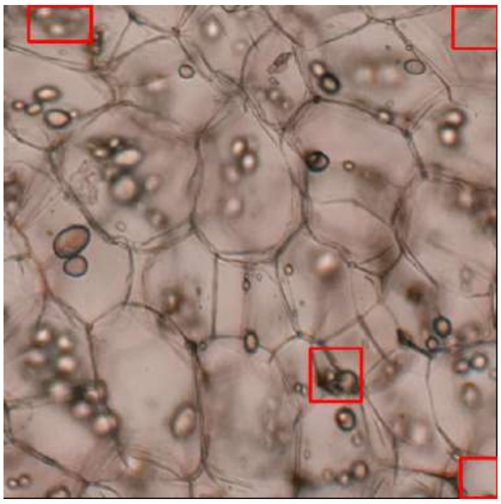

(d) Pretrained model predictions

Figure 6.7: Comparison of pseudo-labels and EMA-pretrained model predictions on in-domain grassland and out-of-domain potato tuber images.
(a) Raw pseudo-labels over grassland input.
(b) EMA-pretrained detector output on the same grassland image.
(c) Raw pseudo-labels over potato tuber input.
(d) EMA-pretrained detector output on the same tuber image.
The reduction of false and irrelevant boxes on the grassland image (like boxes including the person) as well as rare predictions in the tuber image demonstrates that the EMA-pretrained model learned domain-specific plant features rather than merely replicating the pseudo-label generator.

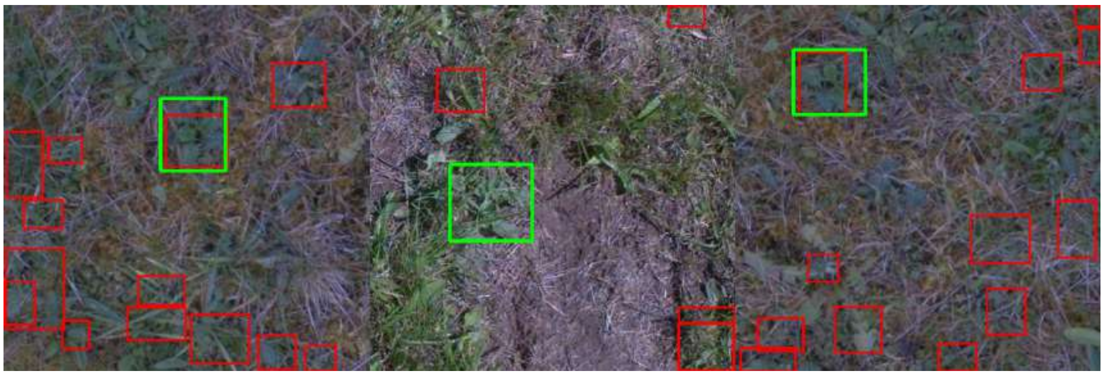


Figure 6.8: Three unseen KIhUG validation samples (not used during pretraining) with EMA-pretrained model predictions (red) overlaid on ground truth annotations (green). Extensive visualizations show that many target plants remain undetected or only partially covered. By lowering the confidence threshold slightly below 0.5 many more good predictions are made, but the visualization becomes more cluttered.

Table 6.4: Quantitative recall comparison on domain-specific data (KIhUG validation split) between the raw SSL pseudo-box generator and the EMA-pretrained model. The detectors higher recall at all IoU thresholds and AR@100 demonstrates that the model refines pseudo-labels into more accurate plant detections.

Metric	**Pseudo**	**Model**
AR@100	0.1372	0.2256
Recall@IoU$_{0.5}$	0.4032	0.6157
Recall@IoU$_{0.6}$	0.2326	0.4029
Recall@IoU$_{0.7}$	0.1192	0.2147

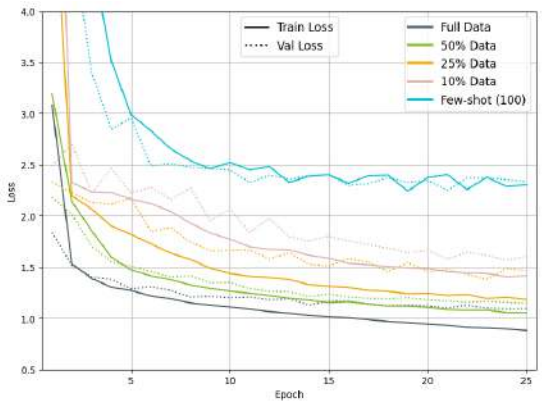
Taken together, these results show that the EMA-based pre-training on pseudoboxes not only remains stable, but also learns meaningful, domain-specific features. This sets the stage for improved downstream fine-tuning, which is explored in the following section.

6.4 Fine-Tuning After SSL Pretraining

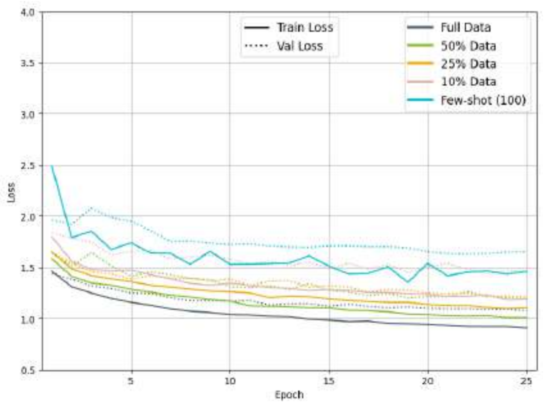
Building on the baseline results of COCO-initialized models (section 6.2), we now assess the impact of initializing from our EMA-SSL checkpoint. In each experiment, we fine-tune the same DDETR architecture with identical hyperparameters and data fractions on the KIhUG and RumexWeeds datasets. By overlaying the SSL-initialized loss curves on top of the obtained COCO-initialized baseline, it is possible to quantify how domain-aware pretraining accelerates convergence and improves performance in low-data regimes.

The experiments begin by directly overlaying the SSL-initialized fine-tuning curves on top of the simple COCO-initialized baselines (no warm-up phase) from Section 6.2. Figure 6.9 displays training and validation losses across the five different data regimes on both KIhUG and RumexWeeds. This comparison isolates the effect of the initial weight choice under identical optimization settings.

Fine-tuned on KIhUG

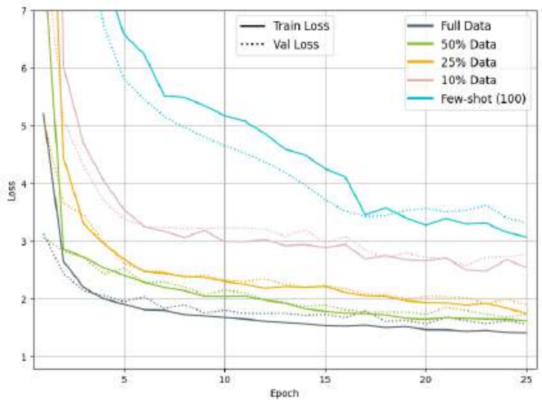


(a) COCO-pretrained

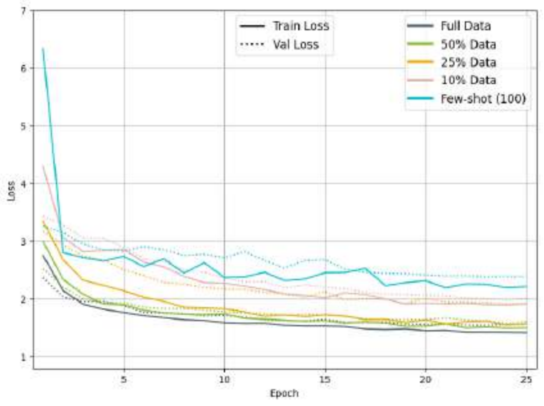


(b) EMA-pretrained

Fine-tuned on Rumex



(c) COCO-pretrained



(d) EMA-pretrained

Figure 6.9: Comparison of training and validation loss across data regimes and pretraining strategies. Each subplot shows loss curves for five different training set sizes (100%, 50%, 25%, 10%, and few-shot), using either COCO-based or EMA-based pretraining. Top: Fine-tuning on the KIhUG dataset. Bottom: Fine-tuning on the Rumex weed dataset. These plots display only the first 25 epochs. Note that auxiliary losses and the custom warm-up phase were disabled for this comparison.

Notably, the gap between COCO and EMA-SSL initializations widens as the available labeled data shrinks. To pinpoint the source of this advantage in the few-shot case, we zoom into the 100-image curves in Figure 6.10 and decompose the total loss into its classification, localization, and regularization components.

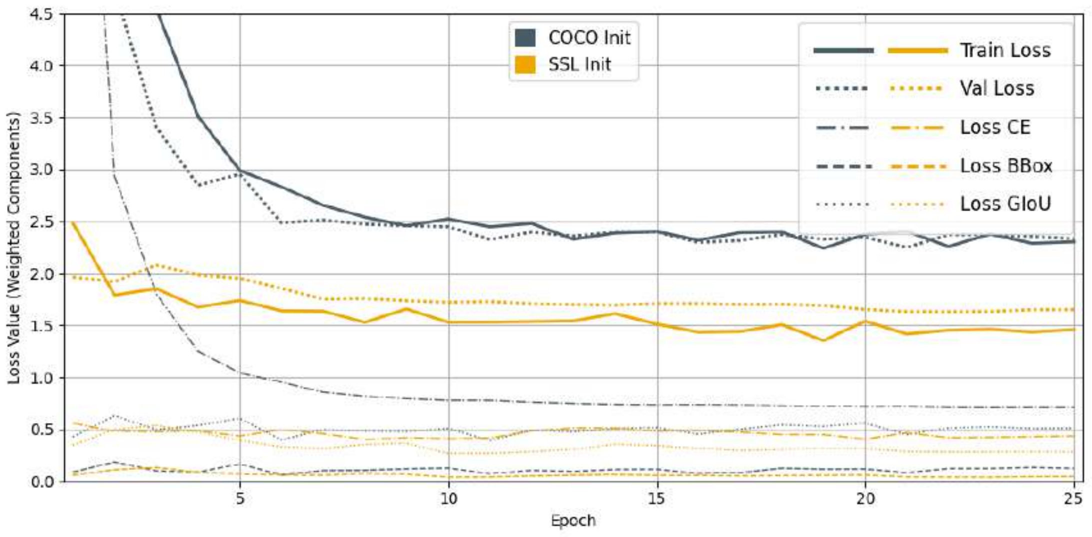


Figure 6.10: Detailed loss decomposition during few-shot fine-tuning on 100 KIhUG images, comparing COCO-pretrained (dark gray) and EMA-SSL-pretrained (orange) initializations. These curves correspond to the few-shot traces from Fig. 6.9 (top-row, in blue), now recolored and expanded to show the total loss alongside its three components over training epochs. The lower starting CE loss for SSL highlights how self-supervised pretraining provides stronger initial classification confidence on plant targets.

While the initial 25-epoch experiments clearly demonstrate an early advantage for the SSL-initialized model, they also revealed that a simple warm-up strategy could mitigate the difficulty of optimizing the cross-entropy loss. Motivated by these findings, a dedicated warm-up phase was introduced (see Section 5.5), during which the classification loss weight is gradually increased from zero to its full value over the first N epochs (configurable in the YAML file, see Appendix A.1). Figure 6.11 presents the results of this extended fine-tuning protocol on both the KIhUG and Rumex Weeds datasets. Despite the stronger COCO baseline achieved through warm-up, the EMA-SSL initialization consistently maintains lower cross-entropy loss throughout, confirming that domain-aware self-supervised pretraining imparts lasting benefits beyond what warm-up alone can provide.

To rigorously evaluate performance under extremely limited labels, additional runs of 500 epochs with a 400-epoch warm-up were conducted on both KIhUG and Rumex Weeds. As shown in Figure 6.12, both setups produce more fluctuating loss curves, yet the EMA-SSL start yields smoother validation trends and a marginally lower final loss. These results confirm that domain-aware pretraining provides durable advantages even when paired with extended warm-up schedules, although the overall gain diminishes in the most extreme few-shot scenario.

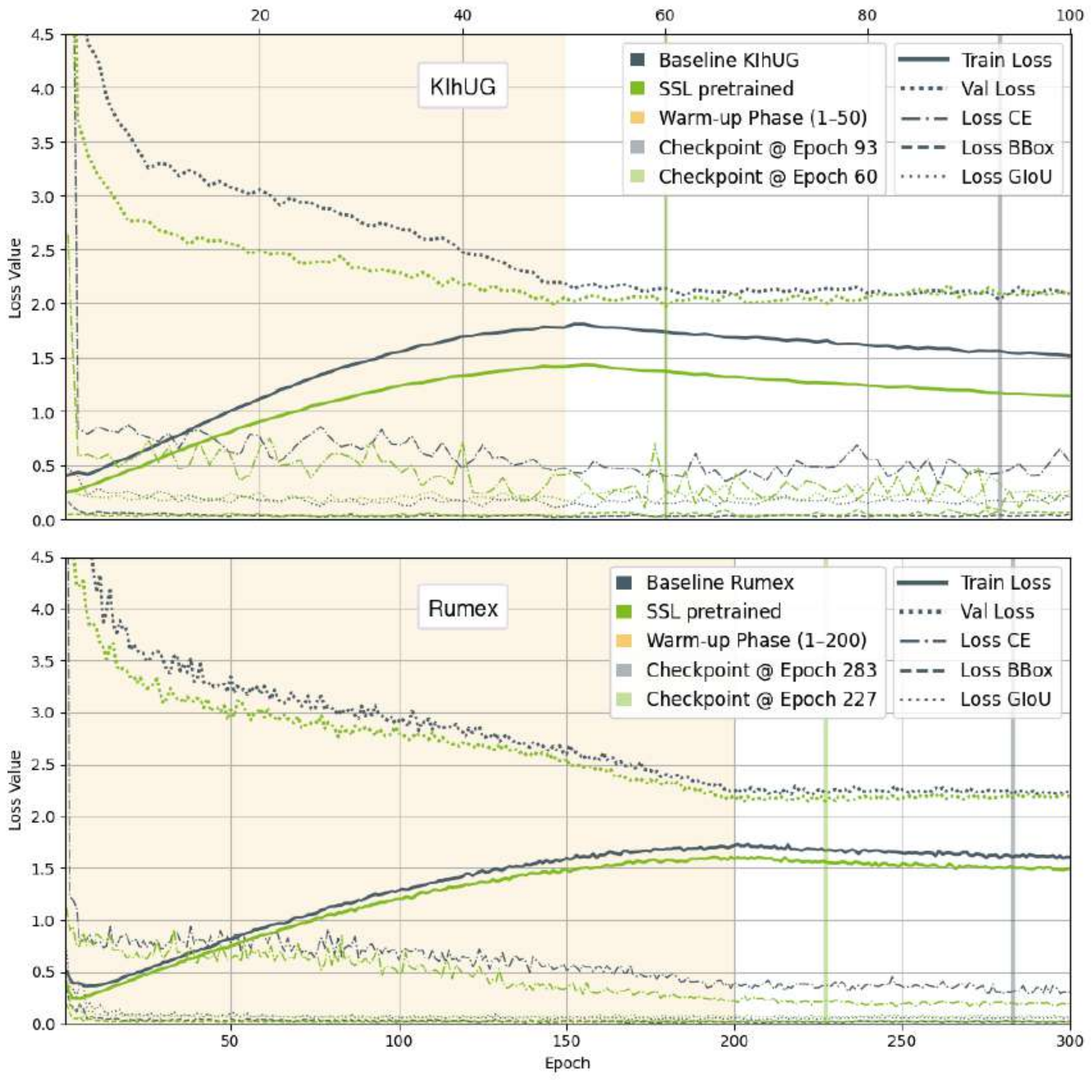


Figure 6.11: Training loss curves for fine-tuning on KIhUG (top) and Rumex Weeds (bottom), comparing the baseline from section 6.2 (COCO-initialized) and EMA-SSL-pretrained checkpoints under identical hyperparameters (including warm-up phase and total epochs). The SSL-initialized runs maintain a lower cross-entropy loss throughout the warm-up phase (despite both models gradually introducing classification loss) demonstrating that the domain knowledge from SSL pretraining persists and accelerates convergence. Extending the warm-up period (200 epochs for Rumex Weeds because of less data for training) partially closes the gap but does not eliminate the advantage of SSL pretraining.

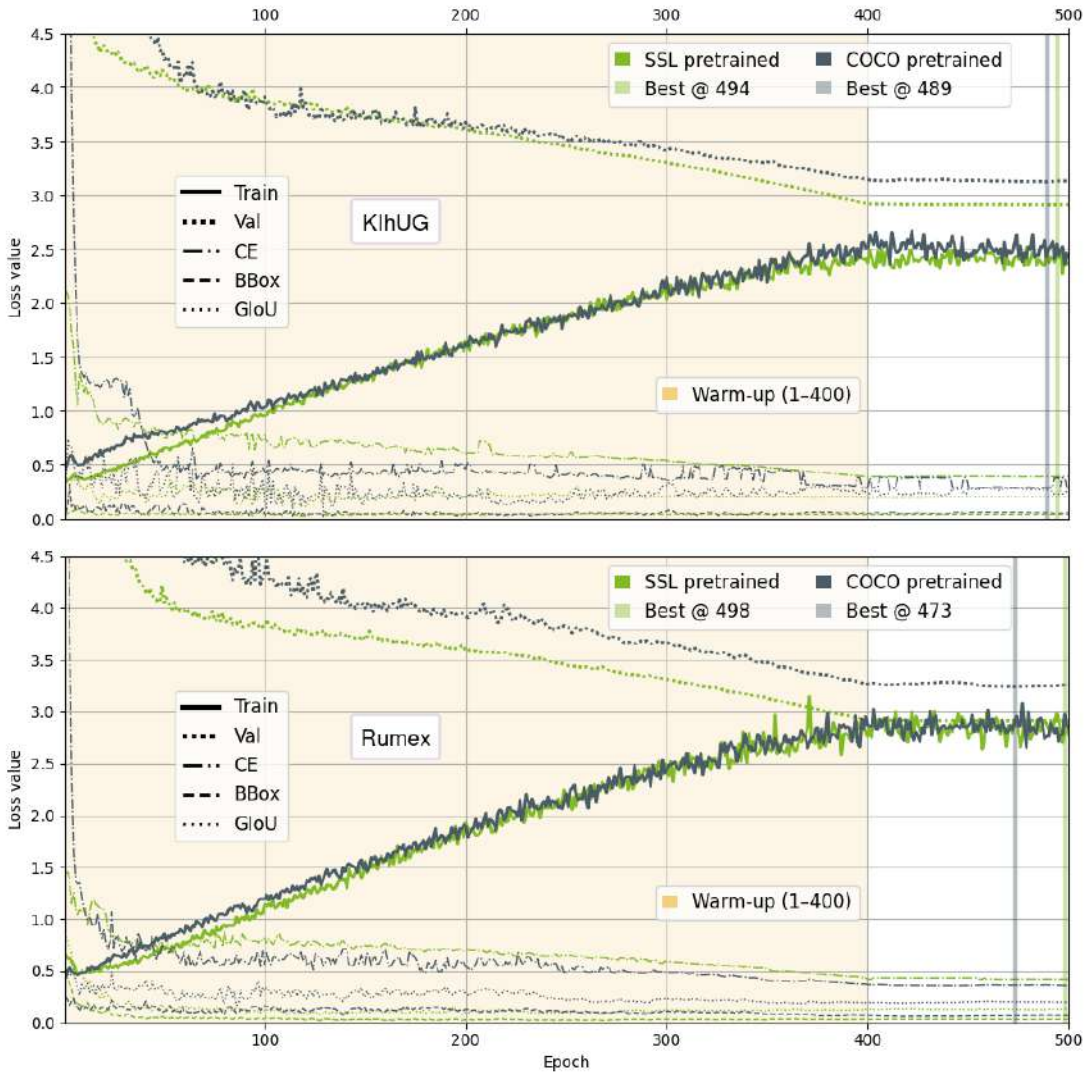


Figure 6.12: Extended 500-epoch few-shot fine-tuning (400-epoch warm-up) on KIhUG (top) and Rumex Weeds (bottom), comparing COCO-initialized (gray) and SSL-pretrained (green) runs under the same aggressive augmentations and learning-rate schedule. In contrast to the short 25-epoch comparison in Fig. 6.10, here both pipelines suffer large training loss fluctuations, but the SSL-initialized model shows significantly smoother validation loss curves with fewer spikes and achieves a slightly lower final loss. This stability and the slightly better convergence show that the self-supervised domain pretraining provides meaningful prior knowledge that cannot simply be replicated by a long warm-up with COCO weights alone, even if the overall training loss behavior appears similar.

	Full Fine-tuning		**Few-shot Fine-tuning**	
Metric	**COCO**	**SSL**	**COCO**	**SSL**
KIhUG (Jacobaea vulgaris)				
AP (50:95)	0.4537	0.4117	0.1093	0.2106
AP@50	0.7928	0.6948	0.3566	0.4108
AP@75	0.4670	0.4342	0.0268	0.2039
AP (small)	0.1265	0.0777	0.0056	0.0164
AP (medium)	0.4664	0.4034	0.1637	0.2374
AP (large)	0.5461	0.5245	0.1936	0.2640
AR@100	0.6157	0.5997	0.3252	0.4699
AR (small)	0.3987	0.3963	0.1547	0.2907
AR (medium)	0.6267	0.6070	0.3513	0.5086
AR (large)	0.6730	0.6602	0.3663	0.4981
Recall@0.50	0.9769	0.9609	0.8422	0.9151
Recall@0.70	0.7290	0.7166	0.2930	0.5448
Rumex Weeds				
AP (50:95)	0.2749	0.2805	0.0786	0.1159
AP@50	0.5672	0.5742	0.2655	0.3245
AP@75	0.2356	0.2383	0.0188	0.0598
AP (small)	0.1533	0.1604	0.0339	0.0518
AP (medium)	0.4548	0.4531	0.1849	0.2093
AP (large)	0.4587	0.5732	0.2100	0.1476
AR@100	0.5118	0.5164	0.2953	0.3818
AR (small)	0.4205	0.4323	0.2339	0.3198
AR (medium)	0.6323	0.6299	0.3873	0.4774
AR (large)	0.7038	0.7044	0.3277	0.4064
Recall@0.50	0.9265	0.9300	0.8197	0.8844
Recall@0.70	0.6032	0.6140	0.2509	0.3933

Table 6.5: Quantitative comparison of COCO versus SSL checkpoints on KIhUG and Rumex Weeds. Columns 2/3 show full fine-tuning results while columns 4/5 show few-shot fine-tuning.

Loss curves alone cannot fully capture detection performance. Models may optimize the loss without corresponding improvements in true detection metricsfor example, by overfitting or producing many low-confidence (≈ 0.5) predictions that never exceed threshold. To provide a definitive assessment, Table 6.5 reports final AP and AR values for both full-data and few-shot fine-tuning on KIhUG and RumexWeeds. By directly comparing runs initialized from our EMA-SSL checkpoint against those starting from generic COCO weights under identical training settings, this table quantifies the actual gains enabled by domain-aware self-supervised pretraining and illustrates where those gains are most pronounced.

It confirms that domain-aware SSL pretraining yields clear benefits in low-data regimes. Under few-shot conditions (100 images), the SSL-initialized runs more than double $\text{AP}_{50:95}$ on KIhUG ($0.109 \longrightarrow 0.211$) and achieve substantial gains on RumexWeeds ($0.079 \longrightarrow 0.116$), along with marked improvements in AR@100 and Recall@0.70. Even when ample labeled data is available, the COCO baseline can nearly match SSL performance after extended warm-up, but only the SSL initialization consistently maintains stronger classification confidence and stability throughout training. While these quantitative gains may not yet reach production-grade robustness, they confirm that domain-specific self-supervised pretraining substantially enhances detection quality under extreme data scarcity. The qualitative examples in Figure 6.12 further underscore the practical promise of this approach, revealing reliable plant identification, even when trained on just 100 labeled images.

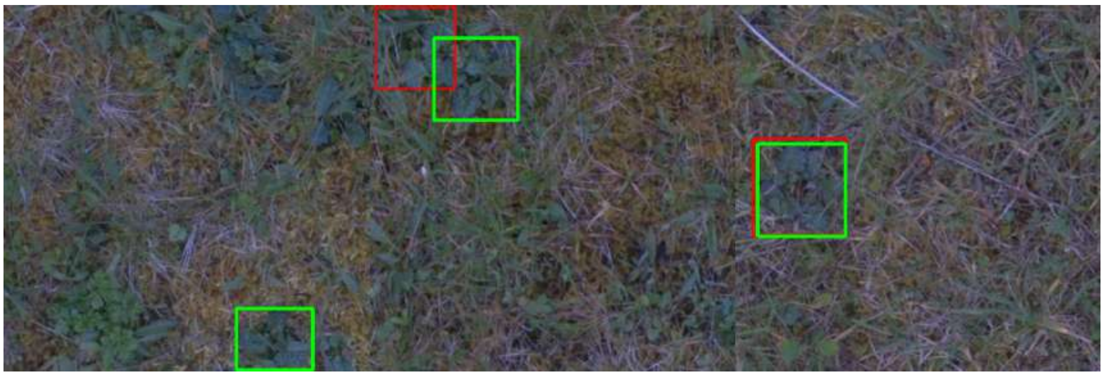


Figure 6.13: Three example validation crops from the KIhUG dataset after few-shot fine-tuning starting from the EMA-SSL checkpoint (100 images, confidence threshold 0.3). Extensive visualizations show that despite the extreme data scarcity, the detector yields few false positives and captures a meaningful subset of plant instances, demonstrating qualitatively strong performance that outpaces what quantitative PyCOCO metrics alone suggest. These results indicate real-world potential for automated removal tasks under minimal annotation budgets, but should always be taken with a grain of salt due to possible observer bias.

6.5 Adapter-Based Experiments

This section examines adapter modules (as introduced previously in 4.2.4) in self-supervised and supervised stages. By training only a small subset of parameters, adapters can cut computational overhead and annotation needs while maintaining object detection performance. First, adapter-only EMA pretraining on reduced unlabeled subsets is evaluated. Then, the effectiveness of these results during downstream fine-tuning, as well as adapter finetuning itself, is assessed.

6.5.1 Adapter-Based EMA-Training

To test parameter-efficient SSL, adapters are trained with the EMA teacher-student framework on limited portions of the unlabeled corpus. This protocol is used to evaluate whether updating only the adapter weights enables stable training and meaningful feature learning with 25% and 50% of the data.

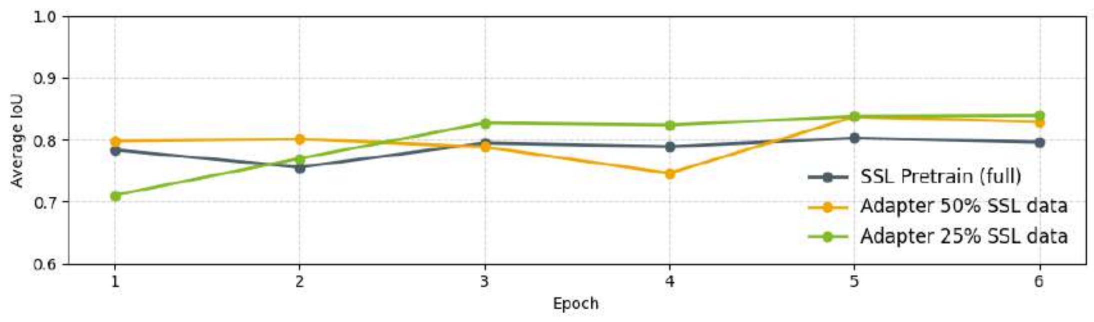


Figure 6.14: Teacher-Student IoU over six epochs for three EMA-based pretraining strategies: full-model fine-tuning on the entire SSL corpus (already shown in Fig. 6.6) and adapter-only on 50% and 25% of SSL data. All three have a constant IoU value (≈0.8), which indicates stable training. The complete training diagrams are not shown here to avoid unnecessary clutter.

Table 6.6: Recall comparison on KIhUG validation split: pseudo-boxes, full-model SSL pretraining, and adapter-only tuning on 50% and 25% of SSL data.

Metric	**Pseudo**	**Full SSL**	**Adapter**$_{50\%}$	**Adapter**$_{25\%}$
AR@100	0.1372	0.2256	0.1398	0.1390
Recall@0.50	0.4032	0.6157	0.4332	0.4532
Recall@0.60	0.2326	0.4029	0.2517	0.2555
Recall@0.70	0.1192	0.2147	0.1085	0.0974

Adapter-only training on 50% of the data achieves over 70% of the full-model recall, while 25% still delivers meaningful results (Table 6.6). This confirms that updating only a small fraction of DDETR parameters can capture domain-specific features with only a modest drop in completeness. A qualitative comparison (see 6.15) illustrates plant localization in unseen samples.

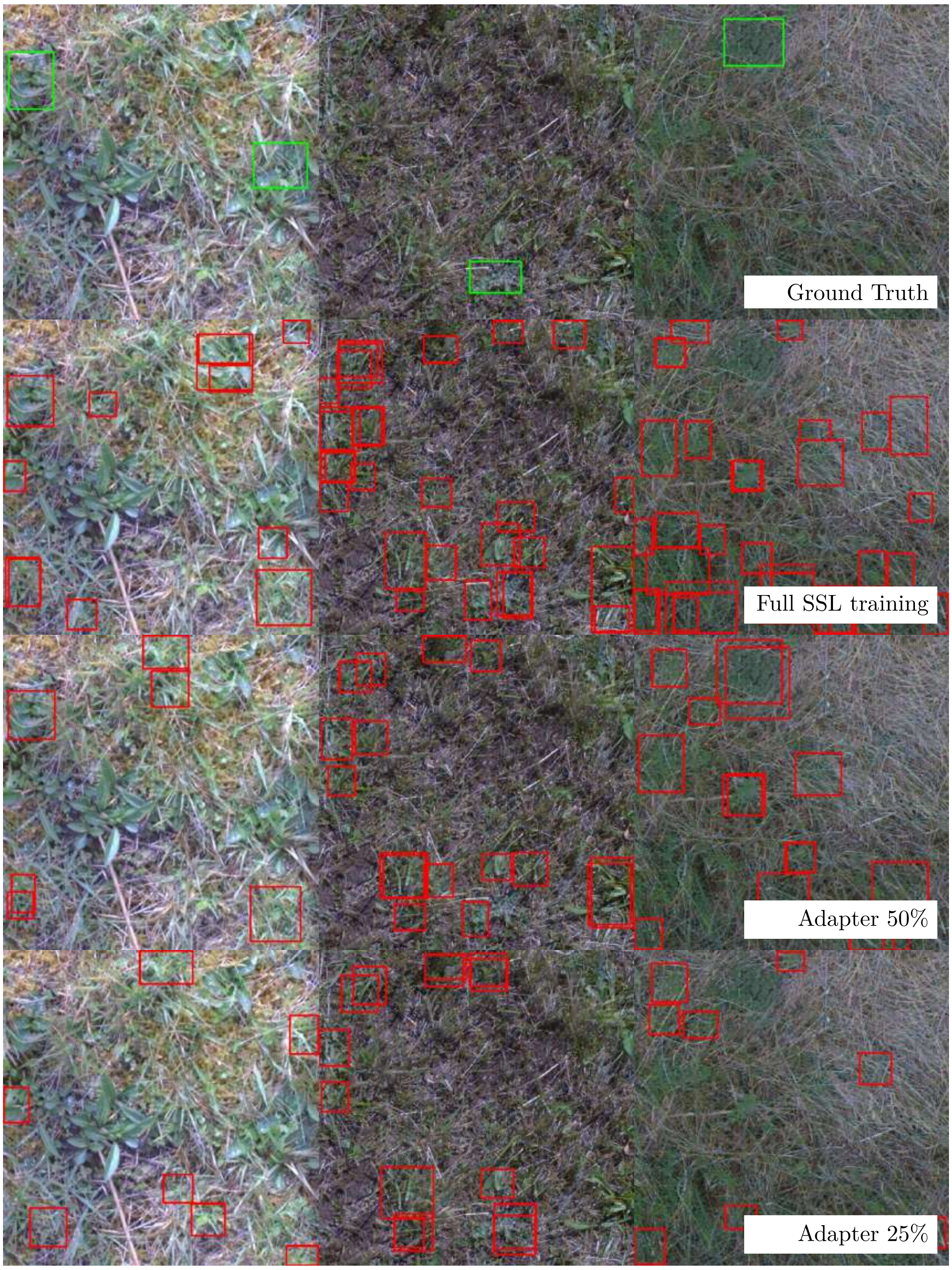


Figure 6.15: Qualitative detection results on three unseen KIhUG images, using three EMA-based SSL pretraining variants (full-model fine-tuning and adapter-only on 50% and 25% data). Predictions are shown at a 0.4 confidence threshold (red) and are compared with ground truth (green). Although adapter-only tuning yields reasonable plant localizations, overall confidence and completeness remain highest with the full-model SSL checkpoint.

6.5.2 Adapter-Based Fine-Tuning

Building on the adapter-only pretraining results (Section 6.5.1), eight few-shot fine-tuning runs are conducted to assess how different initialization and update strategies affect object detection quality. Four initial checkpoints are used: COCO-pretrained, full-model SSL (see section 6.3), and the two adapter-only SSL variants trained on 50% and 25% of the unlabeled corpus. Each is fine-tuned under identical hyperparameters using either full-model or adapter-only updates. Only the KIhUG dataset is used for this already extensive comparison.

Table 6.7 summarizes the final detection metrics obtained via PyCOCO tools, quantifying the trade-off between parameter efficiency and performance. For clarity, the exact combinations of starting checkpoints, fine-tuning variants, and resulting checkpoint names are listed below:

Starting Checkpoints

- `coco`
- `ssl-full`
- `ssl-adpt50`
- `ssl-adpt25`

Fine-tuning Variants

- `full-ft`
- `adpt-ft`

Final Checkpoints

- `coco_full-ft`
- `coco_adpt-ft`
- `ssl-full_full-ft`
- `ssl-full_adpt-ft`
- `ssl-adpt50_full-ft`
- `ssl-adpt50_adpt-ft`
- `ssl-adpt25_full-ft`
- `ssl-adpt25_adpt-ft`

Adapter-only fine-tuning on 100 images achieves surprisingly strong results when starting from the SSL checkpoints (Table 6.7). Although full-model updates yield the highest absolute AP and AR scores (like $AP_{50:95}$ up to 0.2106 for `ssl-full`), adapter-only runs retain over 90% of that performance while updating only 20% of the parameters. By contrast, adapter-only fine-tuning from the COCO baseline falls well below both SSL variants ($AP_{50:95} = 0.0619$), confirming that domain-aware pretraining is essential for parameter-efficient adaptation in few-shot regimes.

These results demonstrate two key insights: (1) self-supervised, domain-specific pretraining provides a persistent boost that survives rigid downstream protocols, and (2) adapter modules alone can suffice to transfer this knowledge, yielding stable and competitive detectors despite freezing the bulk of the backbone. In practice, this means that a lightweight adapter-only workflow, coupled with an EMA-SSL checkpoint, can drastically reduce both compute and annotation costs without sacrificing much detection quality in extreme low-data settings.

	coco		**ssl-full**		**ssl-adpt50**		**ssl-adpt25**	
Metric	full-ft	adpt-ft	full-ft	adpt-ft	full-ft	adpt-ft	full-ft	adpt-ft
AP (50:95)	0.1093	0.0619	0.2106	0.1206	0.1489	0.1504	0.2035	0.1461
AP@50	0.3566	0.2234	0.4108	0.2958	0.3669	0.4249	0.4813	0.4150
AP@75	0.0268	0.0126	0.2039	0.0772	0.0944	0.0609	0.1418	0.0490
AP (small)	0.0056	0.0039	0.0164	0.0085	0.0024	0.0040	0.0148	0.0165
AP (medium)	0.1637	0.0676	0.2374	0.1547	0.1855	0.1694	0.2584	0.1476
AP (large)	0.1936	0.1127	0.2640	0.1972	0.2445	0.2146	0.2909	0.2119
AR@1	0.1770	0.1035	0.2406	0.1704	0.2104	0.1987	0.2581	0.2123
AR@10	0.3033	0.2081	0.4164	0.3298	0.2895	0.3215	0.3624	0.3346
AR@100	0.3252	0.2331	0.4699	0.3984	0.3167	0.3577	0.3741	0.3579
AR (small)	0.1547	0.1579	0.2907	0.2793	0.1481	0.1527	0.1813	0.1932
AR (medium)	0.3513	0.2394	0.5086	0.4300	0.3474	0.3619	0.4012	0.3753
AR (large)	0.3663	0.2517	0.4981	0.4125	0.3466	0.4196	0.4138	0.3925
Recall@0.5	0.8422	0.6922	0.9151	0.9034	0.7004	0.8665	0.7908	0.8580
Recall@0.6	0.6134	0.4331	0.7891	0.7169	0.5491	0.6686	0.6503	0.6657
Recall@0.7	0.2930	0.1826	0.5448	0.4242	0.3459	0.3561	0.4312	0.3720

Table 6.7: Quantitative few-shot evaluation of eight fine-tuning runs on a 100-image split, comparing COCO and SSL pretrained initializations with full-model vs. adapter-only updates. Metrics include AP at standard IoU thresholds and AR for up to 100 detections. Adapter-only variants retain much of the performance of full-model fine-tuning while updating far fewer parameters.

7 | Discussion

The experiments presented in Chapter 6 demonstrate the practical impact of domain-aware self-supervised pretraining and adapter-based parameter efficiency for challenging 'green-on-green' object detection tasks. In what follows, the strengths and limitations of the approach are analyzed in depth, key insights are distilled, and directions for future work are proposed. This discussion ties together the quantitative findings, qualitative observations, and methodological lessons learned throughout this thesis.

7.1 Analysis of Results

The experimental results demonstrate that targeted, domain-specific pretraining via SSL with an EMA teacher-student framework can greatly reduce the need for large annotated datasets. By comparing models initialized with COCO weights, random weights, and our EMA-SSL checkpoint, and by evaluating both full-model and adapter-only setups, several key advantages of the proposed approach become evident:

- **Early convergence:** Fine-tuning from the EMA-SSL checkpoint converges to lower training and validation loss more rapidly than COCO-initialized or randomly initialized counterparts on KIhUG.
- **Improved recall in low-data regimes:** Under few-shot conditions (100 images), recall@0.5 and AR@100 increase substantially when starting from EMA-SSL weights compared to COCO baselines (see Table 6.5).
- **Cross-domain transferability:** The same EMA-SSL pretrained model, without any additional adaptation, achieves strong detection performance on the Rumex Weeds dataset despite no shared training images.
- **Parameter efficiency with adapters:** Adapter-only training-updating only ∼20% of the models parameters-retains over 70% of the full-model recall, illustrating that lightweight modules can capture essential domain features (Table 6.6).

7.2 Strengths and Limitations of the Proposed Approach

The domain-specific SSL pipeline presented here-combining handcrafted pseudo-labels with an EMA teacher-student framework and lightweight adapters demonstrates a powerful proof of concept for few-shot object detection in challenging green-on-green scenarios. By injecting task-focused signals from the very first iteration, the model learns meaningful plant features and converges far more quickly than generic baselines. Adapter modules further enable PEFT, freezing over 80% of the network (including the backbone) while still achieving within 90-95% of full-fine-tuning recall (Table 6.6). This combination of targeted pseudo-labeling, stability via EMA updates, and parameter-efficient tuning offers a compelling route to reduce annotation burden across multiple related datasets.

Yet the road to a robust SSL detector is strewn with pitfalls. Early in development, unchecked collapse modes like underconfidence, teacher-student drift, or trivial box-replication-were rampant and required elaborate tuning of confidence schedules, loss weights, and warm-up phases. Figure 7.1 illustrates a typical failure: after only a few epochs the teacher and student converge to identical but meaningless box patterns, halting any real learning. In practice, if only a single dataset is needed, simply annotating a few hundred images often proves faster and more reliable than wrestling with these dynamics. However, when multiple related tasks or species must be supported, the upfront investment in domain-specific SSL can pay dividends, amortizing annotation effort across many follow-on datasets without repeating the entire labeling process.

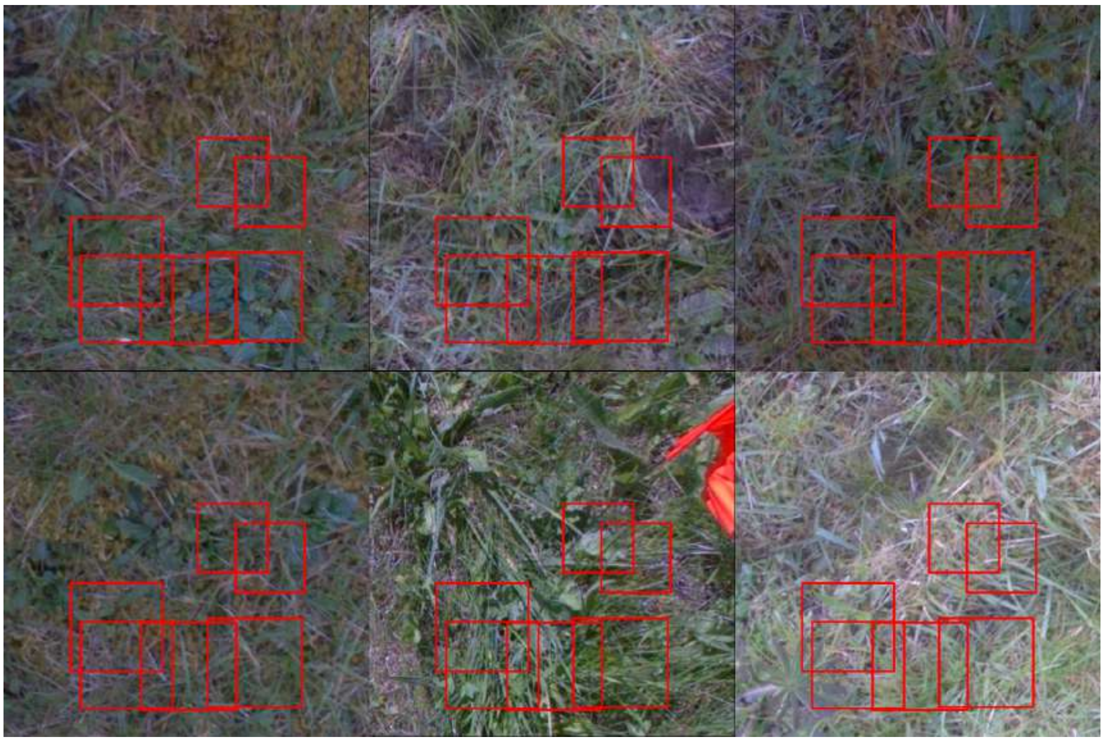


Figure 7.1: Illustration of a collapsed SSL training run: after several epochs, both teacher and student produce identical, trivial bounding-box predictions on all inputs.

7.3 Practical Implications for Domain-Specific Tasks

The methods developed in this thesis demonstrate clear benefits for real-world applications where annotation budgets and domain shifts pose significant challenges. By leveraging SSL with custom pseudo-labels and EMA-based teacher-student updates, practitioners can bootstrap object detectors in new environments without the overhead of large hand-labeled datasets. The integration of adapter modules further enables PEFT, keeping most of a pretrained DDETR backbone frozen while fine-tuning only a small fraction of parameters-ideal for rapid prototype cycles and limited compute budgets.

This pipeline is particularly attractive when:

- **Annotation costs are prohibitive:** Manual labeling of thousands of images requires domain experts and specialized tools, whereas pseudo-label generation and SSL pretraining automate much of the process.
- **Domain shift degrades generic models:** Off-the-shelf detectors like COCO-trained DDETR often fail on 'green-on-green' tasks. Domain-aware pretraining aligns the model to subtle texture and color patterns common in specialized datasets.
- **Resource constraints demand efficiency:** Freezing 80% of the network and updating only adapters drastically reduces memory footprint and training time, making real-time retraining feasible on GPU-equipped edge devices or modest NAS-backed clusters.
- **Multiple related tasks share features:** Once adapters capture domain-specific cues, they can be reused or lightly fine-tuned for related detection tasks (like different weed species), amplifying the return on initial pretraining investment.

Potential use cases extend beyond precision agriculture and weed removal to ecological surveys, medical imaging in niche specialties, and industrial inspection of specialized parts. In all these scenarios, the workflow presented here provides a scalable, reproducible path from unannotated imagery to deployable detection models with minimal manual effort.

7.4 Potential Improvements and Future Work

Although the presented EMA-based SSL and adapter framework demonstrates considerable promise, several enhancements could further strengthen its applicability and robustness:

This work identified multiple failure modes during self-supervised training, ranging from collapse to overconfidence, and showed that careful tuning is essential. Future research should explore:

- **Improved pseudo-label generation:** Develop automated or learned-pseudo label filters, for example via ensemble agreement or uncertainty estimation, to reduce reliance on hand-crafted image processing heuristics and improve initial teacher quality.
- **Hybrid pretraining strategies:** Combine object-level SSL (as in this thesis) with contrastive or masked-image-modeling approaches (like DINO or MAE) that target backbone representations, yielding richer feature hierarchies.
- **Broader PEFT comparisons:** Benchmark against emerging PEFT, such as LoRA, prefix tuning, or distillation frameworks like lightly, to quantify trade-offs between trainable parameter count, training speed, and detection accuracy on the same domain data.
- **Adapter architecture evolution:** Compare alternative adapter designs like compacter to study how adapter size and structure impact stability, plasticity, and downstream performance.
- **Robust out-of-domain performance:** Design mechanisms (like domain-mixup or adversarial augmentation) to maintain detection quality under shifts in illumination, background appearance, or when applied to related but unseen weed species.
- **Scalable multi-class extension:** Instead of single class pseudolabels, assigning random provisional class IDs to generated boxes and presenting them to the EMA teacher-student loop could refine both localization and category predictions. Over successive epochs, the teacher model's soft labels should dominate, effectively bootstrapping multi-class specialization with minimal manual intervention.
- **Automated hyperparameter optimization:** A more dynamic adjustment of the decay of EMA, the step sizes in the curriculum or the warm-up duration would shorten the time for manual testing and improve reproducibility.

These directions aim to deepen the domain alignment of self-supervised detectors, reduce manual overhead, and broaden the methods applicability to diverse, data-scarce object detection tasks.

7.5 Mapping Results to Research Questions

The experiments conducted in this thesis directly address the four key research questions posed at the end of the chapter 1. The following briefly summarizes how the results answer each question and demonstrate the effectiveness and limitations of the proposed domain-specific pretraining and adapter-based strategies.

RQ1: Pseudo-label-only SSL capability A pure SSL object detector trained solely on handcrafted pseudo-boxes demonstrates a measurable ability to localize non-grass plant instances beyond its initial seeds. Qualitative comparisons (Fig. 6.7) and recall gains (Table 6.4) indicate that the model has captured useful domain features, even though some predictions remain imprecise. Despite the mismatch between the self-supervised pretraining corpora and the KIhUG and Rumex Weeds target domains, many detections still correspond to valid plant instances.

RQ2: Impact on downstream fine-tuning Models initialized from our domain-aware EMA-SSL checkpoint converged faster and achieved higher recall in few-shot regimes than COCO or random starts (Figs. 6.9-6.12, Table 6.5). This demonstrates clear benefits in both convergence speed and final detection performance under limited labels.

RQ3: Parameter efficiency via adapters Adapter-only SSL training on 25-50% of the unlabeled corpus retains over 70% of the recall achieved by full-model pretraining (Table 6.6), demonstrating that updating a small subset of parameters can suffice for domain adaptation. These experiments were conducted starting from COCO-pretrained weights and focused on grassland imagery; extending this approach to other domains (like medical imaging) would remain an avenue for future investigation.

RQ4: Adapter transfer to few-shot tasks In downstream few-shot fine-tuning, adapters pretrained during SSL delivered stable learning and respectable performance, sometimes up 15% improvement compared to full-model update, while freezing 80% of weights (Table 6.7). This confirms that adapter modules carry forward valuable domain knowledge and improve sample efficiency.

Taken together, these results validate the central thesis: a refurbished DETReg-inspired pipeline, enhanced with domain-specific pseudo-labels and lightweight adapters, enables effective object detection with minimal annotation effort. While large-scale annotations still yield the best absolute performance, our approach shines when labels are scarce or new related tasks emerge, offering a scalable path for specialized vision applications.

8 | Conclusion

In this work, the DETReg-like pretraining for object recognition in specialized, data-poor domains has been revised and extended. By combining a hand-crafted pseudo-label pipeline with an EMA teacher-learner strategy and adapter modules, it is shown that domain-aware SSL can significantly reduce annotation requirements while maintaining competitive recognition performance.

8.1 Summary of Contributions

- **Revival of DETReg:** Identified and addressed key limitations of the original DETReg method, namely the lack of a classification signal and bad generic region proposals, by introducing a binary foreground/background loss and domain-specific pseudo-labels.
- **Domain-tailored pseudo-labeling:** Developed a simple yet effective classical vision pipeline (GRVI $\rightarrow$ denoise $\rightarrow$ threshold $\rightarrow$ watershed) to seed SSL pretraining on grassland imagery, enabling object-level learning without manual boxes.
- **EMA teacherstudent framework:** Integrated exponential moving average updates to refine pseudo-labels over time and stabilize training against collapse modes. (Similar to DETReg)
- **Adapter-based PEFT:** Injected small adapter modules into the DDETR backbone, freezing 80% of parameters and demonstrating that only 20% trainable weights suffice for both pretraining and downstream few-shot fine-tuning.
- **Comprehensive evaluation:** Extensive experiments on both labeled datasets (KIhUG and RumexWeeds) have shown that domain-specific SSL pretraining accelerates convergence, increases recognition rate on small datasets, and outperforms COCO-based baselines under few-shot conditions.
- **Practical training insights:** Introduced a gradual warm-up of classification loss to prevent underconfidence and further demonstrated that even this simple strategy can match or exceed SSL gains when abundant labels are available.

8.2 Key Findings

- **Annotation savings:** In few-shot scenarios (100 images), SSL-initialized models achieved up to a $\times 2$ recall improvement over COCO baselines (Recall@0.5 on KIhUG: 0.92 vs. 0.84).
- **Adapter efficacy:** Adapter-only pretraining on 50% of unlabeled data recovers over 70% of full-model recall, while freezing the backbone entirely.
- **Training stability:** The EMA + adapter combination mitigates common collapse modes, yet required extensive tuning. One example (see Fig. 7.1) visualizes the challenges of noisy supervision in SSL.
- **Domain shift resilience:** The same SSL checkpoint transfers effectively between KIhUG and RumexWeeds, despite no shared imagery.
- **Diminishing returns:** When large labeled datasets are available, simple warm-up strategies close the performance gap, indicating that SSL pretraining shines primarily under label scarcity.

8.3 Final Remarks

The work shows that an 'improved' DETReg approach (extended with domain-specific pseudo-labels, EMA updates, and parameter-efficient adapters) offers a viable way to reduce annotation overhead in specialized vision tasks. While the method requires careful tuning and cannot replace extensive supervised training when there are many labels, it opens up promising possibilities for applications where labeled data is expensive, expert-derived, or rapidly evolving. Future research integrating contrastive backbone pre-training, ensemble pseudo-labeling or distillation techniques could further strengthen these foundations and extend their impact in practice.

9 | Bibliography

[1] M. Hasan A. S. F. Sohel, D. Diepeveen, H. Laga, and K. Jones M. G.˙ A survey of deep learning techniques for weed detection from images. *Computer and Electronics in Agriculture*, 185:106110, 2021. Highlights challenges like green-on-green detection due to similar colors, textures, and shapes:contentReference[oaicite:2]index=2.

[2] Eric Arazo, Diego Ortego, Paul Albert, Noel E. O'Connor, and Kevin McGuinness. Pseudo-labeling and confirmation bias in deep semi-supervised learning, 2020.

[3] Amir Bar, Xin Wang, Vadim Kantorov, Colorado J Reed, Roei Herzig, Gal Chechik, Anna Rohrbach, Trevor Darrell, and Amir Globerson. Detreg: Unsupervised pretraining with region priors for object detection, 2023.

[4] Soroush A. Barzegar, Jeremy Maitin-Shepard, and Benjamin Recht. On the state of data in computer vision: Human annotations remain indispensable, 2023.

[5] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 4148, New York, NY, USA, 2009. Association for Computing Machinery.

[6] Sumona Biswas and Shovan Barma. A largescale optical microscopy image dataset of potato tuber for deep learning based plant cell assessment. *Scientific Data*, 7(1):371, 2020.

[7] Dan Busbridge, Jason Ramapuram, Pierre Ablin, Tatiana Likhomanenko, Eeshan Gunesh Dhekane, Xavier Suau, and Russ Webb. How to scale your ema, 2023.

[8] Alexander Buslaev, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A. Kalinin. Albumentations: Fast and flexible image augmentations. *Information*, 11(2), 2020.

[9] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers, 2020.

[10] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.

[11] Daquan Chen, Jianmin Bao, Xu Dong, Lu Yuan, Dongdong Chen, Jie Yuan, Lei Zhang, and Fang Wen. Adaptformer: Adapting vision transformers for scalable visual recognition. In *NeurIPS*, 2022.

[12] Haolong Chen, Hanzhi Chen, Zijian Zhao, Kaifeng Han, Guangxu Zhu, Yichen Zhao, Ying Du, Wei Xu, and Qingjiang Shi. An overview of domain-specific foundation model: key technologies, applications and challenges, 2025.

[13] Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition, 2022.

[14] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020.

[15] Yuhua Chen, Wen Li, Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Domain adaptive faster r-cnn for object detection in the wild, 2018.

[16] Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. Vision transformer adapter for dense predictions, 2023.

[17] Zhigang Dai, Bolun Cai, Yugeng Lin, and Junying Chen. Unsupervised pre-training for detection transformers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 111, 2022.

[18] 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, 2021.

[19] Istiaq Ahmed Fahad, Abdullah Ibne Hanif Arean, Nazmus Sakib Ahmed, and Mahmudul Hasan. Automatic vehicle detection using detr: A transformer-based approach for navigating treacherous roads, 2025.

[20] Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout, 2019.

[21] Yongchao Feng, Yajie Liu, Shuai Yang, Wenrui Cai, Jinqing Zhang, Qiqi Zhan, Ziyue Huang, Hongxi Yan, Qiao Wan, Chenguang Liu, Junzhe Wang, Jiahui Lv, Ziqi Liu, Tengyuan Shi, Qingjie Liu, and Yunhong Wang. Vision-language model for object detection and segmentation: A review and evaluation, 2025.

[22] Bradley J. Fletcher and John T. Smith. Jacobaea vulgarisan overview of toxicity and livestock impacts. *ScienceDirect Topics*, 2024. Jacobaea contains pyrrolizidine alkaloids; risks include liver damage in cattle and poisoning through contaminated hay:contentReference[oaicite:1]index=1.

[23] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation, 2015.

[24] R.J. Godwin and P.C.H. Miller. A review of the technologies for mapping within-field variability. *Biosystems Engineering*, 84(4):393–407, 2003. Precision Agriculture - Managing Soil and Crop Variability for Cereals.

[25] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H.

Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning, 2020.

[26] Ronja Güldenring, Frits K. van Evert, and Lazaros Nalpantidis. Rumexweeds: A grassland dataset for agricultural robotics. *Journal of Field Robotics*, 2023.

[27] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

[28] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning, 2020.

[29] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[30] Jan Hosang, Rodrigo Benenson, and Bernt Schiele. How good are detection proposals, really?, 2014.

[31] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp, 2019.

[32] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

[33] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning, 2022.

[34] Patrick Kage, Jay C. Rothenberger, Pavlos Andreadis, and Dimitrios I. Diochnos. A review of pseudo-labeling for computer vision, 2025.

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

[36] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything, 2023.

[37] Till Jacob Koch. Generierung synthetischer modellbasierter bilddaten zum training von künstlichen neuronalen netzen: Vergleich der anwendbarkeit von “cut, paste and learn” und 3d-modellierungsumgebungen auf reale daten. Bachelor thesis, Technische Hochschule Mittelhessen, Fachbereich MNI, December 2023. Supervised by Prof. Dr.-Ing. Seyed Eghbal Ghobadi and Moritz Schauer, M.Sc.

[38] Harold W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 52, 1955.

[39] Feng Li, Hao Zhang, Shilong Liu, Jian Guo, Lionel M. Ni, and Lei Zhang.

Dn-detr: Accelerate detr training by introducing query denoising, 2022.

[40] Wenwen Li, Chia-Yu Hsu, Sizhe Wang, Yezhou Yang, Hyunho Lee, Anna Liljedahl, Chandi Witharana, Yili Yang, Brendan M. Rogers, Samantha T. Arundel, Matthew B. Jones, Kenton McHenry, and Patricia Solis. Segment anything model can not segment anything: Assessing ai foundation model's generalizability in permafrost mapping, 2024.

[41] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[42] Zhiwei Lin and Yongtao Wang. Vl-sam-v2: Open-world object detection with general and specific query fusion, 2025.

[43] Zhiwei Lin, Yongtao Wang, and Zhi Tang. Training-free open-ended object detection and segmentation via attention as prompts, 2024.

[44] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2017.

[45] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[46] Yan Ma, Weicong Liang, Bohan Chen, Yiduo Hao, Bojian Hou, Xiangyu Yue, Chao Zhang, and Yuhui Yuan. Revisiting detr pre-training for object detection, 2023.

[47] Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. Compacter: Efficient low-rank hypercomplex adapter layers. In *Advances in Neural Information Processing Systems*, 2021.

[48] Depu Meng, Xiaokang Chen, Zejia Fan, Gang Zeng, Houqiang Li, Yuhui Yuan, Lei Sun, and Jingdong Wang. Conditional detr for fast training convergence, 2023.

[49] Daniel Morales-Brotons, Thijs Vogels, and Hadrien Hendrikx. Exponential moving average of weights in deep learning: Dynamics and benefits, 2024.

[50] Takeshi Motohka, Kenlo N. Nasahara, Hiroyuki Oguma, and Satoshi Tsuchida. Applicability of greenred vegetation index for remote sensing of vegetation phenology. *Remote Sensing*, 2(10):2369–2387, 2010.

[51] James R. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957.

[52] OpenAI. Gpt-4 technical report, 2024.

[53] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[54] Gracile Astlin Pereira and Muhammad Hussain. A review of transformer-based models for computer vision tasks: Capturing global context and spatial

relationships, 2024.

[55] Kanchana Ranasinghe, Muzammal Naseer, Munawar Hayat, Salman Khan, and Fahad Shahbaz Khan. Orthogonal projection loss, 2021.

[56] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016.

[57] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.

[58] 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, 2019.

[59] Roboflow. Roboflow model library. `https://roboflow.com/models`, 2023. Accessed: 2024-05-30.

[60] Leonid I. Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268, 1992.

[61] Kuniaki Saito, Yoshitaka Ushiku, Tatsuya Harada, and Kate Saenko. Strong-weak distribution alignment for adaptive object detection, 2019.

[62] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation, 2018.

[63] Moritz Schauer, Renke Hohl, Dennis Vaupel, Diethelm Bienhaus, and Seyed Eghbal Ghobadi. Towards automated regulation of jacobaea vulgaris in grassland using deep neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 702–711, October 2023.

[64] Soren Skovsen, Mads Dyrmann, Anders K. Mortensen, Morten S. Laursen, Rene Gislum, Jorgen Eriksen, Sadaf Farkhani, Henrik Karstoft, and Rasmus N. Jorgensen. The grassclover image dataset for semantic and hierarchical species understanding in agriculture. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[65] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence, 2020.

[66] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks, 2015.

[67] Hui Tang and Kui Jia. Discriminative adversarial domain adaptation, 2019.

[68] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results, 2018.

[69] Yunjie Tian, Qixiang Ye, and David Doermann. Yolov12: Attention-centric

real-time object detectors, 2025.

[70] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013.

[71] Taha ValizadehAslani, Yiwen Shi, Jing Wang, Ping Ren, Yi Zhang, Meng Hu, Liang Zhao, and Hualou Liang. Two-stage fine-tuning: A novel strategy for learning class-imbalanced data, 2022.

[72] Bart M. van Marrewijk, Charbel Dandjinou, Dan Jeric Arcega Rustia, Nicolas Franco Gonzalez, Boubacar Diallo, Jérôme Dias, Paul Melki, and Pieter M. Blok. Active learning for efficient annotation in precision agriculture: a use-case on crop-weed semantic segmentation, 2024.

[73] Dennis Vaupel. Generation of synthetic training data for neural network-based weed detection. Master thesis, Technische Hochschule Mittelhessen, Fachbereich MNI, March 2024. Supervised by Prof. Dr.-Ing. Seyed Eghbal Ghobadi and Moritz Schauer, M.Sc.

[74] L. Vincent and P. Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598, 1991.

[75] Hui Wang, Hao Li, Wanli Qian, Wenhui Diao, Liangjin Zhao, Jinghua Zhang, and Daobing Zhang. Dynamic pseudo-label generation for weakly supervised object detection in remote sensing images. *Remote Sensing*, 13(8), 2021.

[76] Kuo Wang, Lechao Cheng, Weikai Chen, Pingping Zhang, Liang Lin, Fan Zhou, and Guanbin Li. Marvelovd: Marrying object recognition and vision-language models for robust open-vocabulary object detection, 2024.

[77] Wenhai Wang, Zhe Chen, Xiaokang Chen, Jiannan Wu, Xizhou Zhu, Gang Zeng, Ping Luo, Tong Lu, Jie Zhou, Yu Qiao, and Jifeng Dai. Visionllm: Large language model is also an open-ended decoder for vision-centric tasks, 2023.

[78] Zhanyu Wang, Xiao Zhang, Hyokun Yun, Choon Hui Teo, and Trishul Chilimbi. Mico: Selective search with mutual information co-training, 2022.

[79] Xi Weng, Yunhao Ni, Tengwei Song, Jie Luo, Rao Muhammad Anwer, Salman Khan, Fahad Shahbaz Khan, and Lei Huang. Modulate your spectrum in self-supervised learning, 2024.

[80] Junfeng Wu, Yi Jiang, Qihao Liu, Zehuan Yuan, Xiang Bai, and Song Bai. General object foundation model for images and videos at scale, 2023.

[81] Yi Xin, Jianjiang Yang, Siqi Luo, Haodi Zhou, Junlong Du, Xiaohong Liu, Yue Fan, Qing Li, and Yuntao Du. Parameter-efficient fine-tuning for pre-trained vision models: A survey, 2025.

[82] Jiarui Xu, Mingjie Zhang, Shikun Ye, Zitian Hu, and Jifeng Dai. Specialized foundation models struggle to beat supervised baselines, 2025. ICLR 2025

Workshop Poster, Foundation Models for Decision Making.

[83] Yanqi Xu, Yiqiu Shen, Carlos Fernandez-Granda, Laura Heacock, and Krzysztof J. Geras. Understanding differences in applying detr to natural and medical images. *Machine Learning for Biomedical Imaging*, 3(May 2025):152170, May 2025.

[84] Gongjie Zhang, Zhipeng Luo, Kaiwen Cui, and Shijian Lu. Meta-detr: Image-level few-shot object detection with inter-class correlation exploitation, 2021.

[85] Lanyun Zhu, Tianrun Chen, Deyi Ji, Jieping Ye, and Jun Liu. Llafs: When large language models meet few-shot segmentation, 2024.

[86] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection, 2021.

[87] Larry Zitnick and Piotr Dollar. Edge boxes: Locating object proposals from edges. In *European Conference on Computer Vision*, September 2014.

A | Appendices

A.1 Full Example - Configuration File

```
# === Paths and Directories ===
BASE_DIR: "/home/mlhn64/ssl_ddetr_mlhn64"
ROOT_DIR: "/datasets"
KIHUG_DIR: "/datasets/kihug_raw_dataset/detect"
RUMEX_DIR: "/datasets/RumexWeeds"

# EMA Training Files
EMA_YAML: "dataset_configs/ssl_data_25.yaml"
EVAL_YAML: "../../datasets/kihug_raw_dataset/detect/dataset_split.yaml"
EMA_CHECKPOINT_START: "checkpoints/no.pth"
EMA_CHECKPOINT_OUTPUT: "checkpoints/ssl_pretrain_adapter_25.pth"
EMA_LOG_FILE: "logs/ssl_pretrain_adapter_25.json"

# Fine-tuning Files
TRAIN_RUMEX: false
FINETUNE_TRAIN_YAML: "../../datasets/kihug_raw_dataset/detect/
    ↪ dataset_split_fewshot_100.yaml"
FINETUNE_CHECKPOINT_START: "checkpoints/no_ssl_pretrain.pth"
FINETUNE_CHECKPOINT_OUTPUT: "checkpoints/coco_adapt-ft.pth"
FINETUNE_LOG_FILE: "logs/coco_adapt-ft.json"

# === EMA Specific ===
EMA_BATCH_SIZE: 14
EMA_WEIGHT_DECAY: 0.0001
EMA_EPOCHS: 6
EMA_LEARNING_RATE: 0.0001
SCHEDULAR_GAMMA: 0.95
EMA_DECAY: 0.995
TEACHER_RATIO_START: 0.5
PSEUDO_RATIO_START: 0.5
CURRICULUM_STEP: 0.1
TEACHER_TEMPERATURE: 1.1
STUDENT_TEMPERATURE: 0.9
ORTHO_ALIGNER_WEIGHT: 2.0
ORTHO_ADAPTER_WEIGHT: 0.2
JITTER_RATIO: 0.1
DROPOUT_PROB: 0.25
FREEZE_MODEL: true # adapter only
WARMUP_MODEL: false # freeze backbone only ...
WARMUP_STEPS: 1000  # ... for N epochs

# === Fine-tuning Specific ===
FINETUNE_BATCH_SIZE: 4 # 16 (full) / 4 (few shot learning)
FINETUNE_WEIGHT_DECAY: 0.0001
FINETUNE_EPOCHS: 500
FINETUNE_LEARNING_RATE: 0.0001
FINETUNING_SCHEDULAR_GAMMA: 0.99
FINETUNING_FREEZE_MODEL: true # adapter only
FINETUNING_WARMUP_EPOCHS: 400 # introduce CE loss slowly over N epochs
```

A.2 SLURM Job Scripts

```bash
#!/bin/bash
#SBATCH --job-name=ema_ddetr
#SBATCH --output=%j_ema.out  # Combined output and error log
#SBATCH --time=42:00:00
#SBATCH --mem=32G
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --gres=gpu:1

# SLURM container settings
#SBATCH --container-image='/home/mlhn64/image.sqfs'
#SBATCH --container-mounts='/datasets,/datasets/clover_unlabeled,/
    ↪ datasets/RumexWeeds,/datasets/clover_synthethic_Images,/datasets
    ↪ /20240826_Drohnenaufnahmen_Rennerod,/datasets/3D-Sim-v1'
#SBATCH --container-workdir='/home/mlhn64/'
#SBATCH --no-container-remap-root
#SBATCH --container-mount-home

export PYTORCH_CUDA_ALLOC_CONF=expandable_segments:True

# Run the self-supervised training script
python ssl_ddetr_mlhn64/ema_ddetr.py
```

Listing A.2: SLURM Job File for SSL Pretraining with EMA

```bash
#!/bin/bash
#SBATCH --job-name=finetune_ddetr
#SBATCH --output=%j_finetune.out  # Combined output and error log
#SBATCH --time=42:00:00
#SBATCH --mem=32G
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --gres=gpu:1  # Request a GPU if needed

# SLURM container settings
#SBATCH --container-image='/home/mlhn64/image.sqfs'  # Use your saved
    ↪ image
#SBATCH --container-mounts='/datasets,/datasets/RumexWeeds/20210806
    ↪ _hegnstrup,/datasets/RumexWeeds/20210806_stengard,/datasets/
    ↪ RumexWeeds/20210807_lundholm,/datasets/RumexWeeds/20210908_lundholm
    ↪ ,/datasets/RumexWeeds/20211006_stengard'
#SBATCH --container-workdir='/home/mlhn64/'         # Set working
    ↪ directory inside the container
#SBATCH --no-container-remap-root                   # Ensure root
    ↪ permissions inside container
#SBATCH --container-mount-home                      # Mount home
    ↪ directory inside the container

# Run the test script inside the container
python ssl_ddetr_mlhn64/finetune_ddetr.py
```

Listing A.3: SLURM Job File for Supervised Fine-Tuning

A.3 Pseudo-Label Generation: Core Logic

```python
# Pseudo Label Extraction Using GRVI and Classical CV Techniques
def extract_pseudo_boxes(image):
    boxes = []
    image = image.astype(np.float32)
    h, w = image.shape[:2]

    # Compute GRVI: (G - R) / (G + R)
    R, G, _ = image[:, :, 2], image[:, :, 1], image[:, :, 0]
    brightness = (R + G + image[:, :, 0]) / 3
    grvi = np.zeros_like(G, dtype=np.float32)
    mask = brightness > 50  # Avoid division noise in dark areas
    grvi[mask] = (G[mask] - R[mask]) / (R[mask] + G[mask] + 1e-6)
    grvi = (grvi + 1) / 2  # Normalize to [0, 1]

    # Apply denoising, thresholding, and watershed
    A_tv = denoise_tv_chambolle(grvi, weight=0.7)
    A_uint8 = cv2.normalize(A_tv, None, 0, 255, cv2.NORM_MINMAX).astype(
        ↪ np.uint8)
    _, binary = cv2.threshold(A_uint8, 0, 255, cv2.THRESH_BINARY + cv2.
        ↪ THRESH_OTSU)

    morph = cv2.erode(binary, None, iterations=3)
    morph = cv2.dilate(morph, None, iterations=3)

    dist = cv2.distanceTransform(morph, cv2.DIST_L2, 5)
    dist[dist < 0.1 * dist.max()] = 0
    local_max = local_maxima(dist)
    markers, _ = ndi.label(local_max)
    label_map = watershed(-dist, markers, mask=morph)

    # Convert segments to normalized bounding boxes
    for sl in find_objects(label_map):
        if sl:
            y_min, y_max = sl[0].start, sl[0].stop
            x_min, x_max = sl[1].start, sl[1].stop
            cx, cy = ((x_min + x_max) / 2) / w, ((y_min + y_max) / 2) / h
            bw, bh = (x_max - x_min) / w, (y_max - y_min) / h
            boxes.append((cx, cy, bw, bh))

    return boxes
```

Listing A.4: Key Snippet from the Pseudo-Label Extraction Pipeline

A.4 Full Code and Resources

The implementation and related resources for this thesis are available in the GitLab repository hosted by Technische Hochschule Mittelhessen. Please note that access may require institutional permissions.

`https://git.thm.de/institut-f-r-technik-und-informatik/projects/kihug/ssl_ddetr_mlhn64`