Semi-Supervised Weed Detection With YOLO & Graph Neural Networks

Hostel ID: **78**

5 February 2025

1 Introduction

Traditional deep learning-based object detection models require a large volume of labeled data, which is expensive and time-consuming to obtain. This work explores the use of **semi-supervised learning** (SSL), which combines labeled and unlabeled data to improve weed detection accuracy. This study presents a novel pipeline integrating **YOLO-based object detection**, **pseudo-labeling**, and **Graph Neural Networks (GNNs)** for label propogation. The key highlights of this work are:

- A YOLO11-based weed detection model trained on limited labeled data.
- A pseudo-labeling strategy for generating labels for unlabeled data to extend training.
- Graph Neural Networks (GNNs) using transductive label propagation to refine predictions.

The dataset consists of agricultural field images containing sesame crops and weeds. The objective is to accurately identify and localize weeds using a semi-supervised approach. The dataset is represented as:

$$\mathcal{D} = \mathcal{D}_L \cup \mathcal{D}_U$$

where:

- $\mathcal{D}_L = \{(x_i, y_i)\}_{i=1}^{N_L}$ represents the labeled dataset with N_L images and their annotations.
- $\mathcal{D}_U = \{x_i\}_{i=N_L+1}^{N}$ denotes the unlabeled dataset containing N_U images.

Each labeled image follows the YOLOv8 annotation format:

$$y_i = \{(c_k, x_k, y_k, w_k, h_k)\}_{k=1}^{M_i}$$

where c_k represents the class (weed or crop), and (x_k, y_k, w_k, h_k) denote the bounding box parameters.

2 Methodology

The proposed approach involves three main stages:

1. Supervised fine-tuning of YOLO11 on labeled dataset.
2. Pseudo labeling using Graph Neural Network.
3. Final Training on the complete labeled and pseudo labled dataset.

2.1 Supervised Learning with YOLO11

Before introducing pseudo-labels from the unlabeled dataset, the model must first be trained on high-quality labeled data to develop a robust feature extractor. The labeled dataset provides accurate object bounding boxes and class labels, ensuring that the YOLO11 backbone network effectively learns meaningful feature representations. This is essential because, without a well-trained backbone, the model would generate low-confidence pseudo-labels for unlabeled images, negatively impacting the entire semi-supervised learning process. The loss function in YOLO follows:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{CLS}} + \mathcal{L}_{\text{BOX}} + \mathcal{L}_{\text{DFL}}$$

where $\mathcal{L}_{\text{CLS}}$ is the classification loss, responsible for correctly assigning object categories, $\mathcal{L}_{\text{BOX}}$ is the complete IoU loss, which ensures accurate bounding box localization and $\mathcal{L}_{\text{DFL}}$(Distribution Focal Loss) refines bounding box regression by improving the precision of object boundaries.

2.2 Pseudo-Labeling & GNN

We first construct the graph by connecting each node to its k most similar neighbors based on feature embeddings extracted from the YOLO11 backbone. The GNN was trained on a graph where nodes represent images, and edges are defined based on feature similarity. Let $G = (V, E)$ be a graph where:

- V is the set of nodes, where each node is an image.
- f_i is the feature embedding of node i, extracted using the YOLO11 backbone.
- E is the set of edges between nodes, whose weights are calculated based on feature similarity.

The edges E are assigned weights using a similarity function:

$$A_{ij} = \begin{cases} \exp(-\|f_i - f_j\|^2) & \text{if } j \in \text{KNN}(i) \\ 0 & \text{otherwise} \end{cases}$$

where A_{ij} represents the adjacency matrix of the graph and $\text{KNN}(i)$ is the set of nearest neighbors of node i, determined using the K-Nearest Neighbors (KNN) algorithm. Then GNN propagates labels through the graph using the Graph Attention Network (GAT) formulation:

$$H^{(l+1)} = \sigma(D^{-1/2} A D^{-1/2} H^{(l)} W^{(l)})$$

where A is the adjacency matrix, D is the degree matrix, $H^{(l)}$ is the node embedding at layer l, $W^{(l)}$ is the trainable weight matrix and σ is an activation function (ELU in our case). The GNN training process optimizes three loss components: supervised loss, pseudo-label loss, and smoothness loss. The supervised loss ($\mathcal{L}_{\text{supervised}}$) ensures correct classification of labeled nodes using negative log-likelihood. The pseudo-label loss ($\mathcal{L}_{\text{pseudo}}$) trains the model on high-confidence pseudo-labels from the unlabeled dataset, incorporating only predictions above a confidence threshold τ. The smoothness loss ($\mathcal{L}_{\text{smoothness}}$) enforces consistency between predictions of similar nodes in the feature graph by minimizing the difference between connected node probabilities. The total loss function used for training is given by:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{supervised}} + \lambda_{\text{pseudo}} \mathcal{L}_{\text{pseudo}} + \lambda_{\text{smooth}} \mathcal{L}_{\text{smoothness}}$$

Table 1: Performance Metrics and Class Distribution Before and After Pseudo-Labeling

Stage	**Box Loss**	**Cls Loss**	**DFL Loss**	**mAP@50**	**mAP@50-95**
Before Pseudo-Labeling	0.4392	0.3081	1.216	0.915	0.637
After Pseudo-Labeling	0.3148	0.2069	0.8689	0.907	0.651
Class-wise Metrics (Test Set)	**Images**	**Instances**	**Precision**	**Recall**	**mAP@50**	**mAP@50-95**
All Classes	50	74	0.929	0.848	0.905	0.647
Weed	24	42	1.000	0.852	0.901	0.680
Crop	26	32	0.858	0.844	0.908	0.615
Overall Metrics	-	-	**0.929**	**0.8478**	**0.9047**	**0.6474**
Class Distribution in Datasets	**Total Samples**	**Weed**	**Crop**			
Original Labeled Dataset (200 Samples)	200	88	112			
Pseudo-Labeled Dataset (1200 Samples)	1200	561	639			

This formulation ensures accurate classification, effective pseudo-labeling, and smooth feature propagation within the GNN. Mathematically, pseudo-labeling involves assigning:

$$\tilde{y} = \arg\max_{c} P(c|x)$$

where $\tilde{y}$ is the pseudo-label assigned to an unlabeled image. $P(c|x)$ is the probability that the image belongs to class c. To ensure only high-confidence pseudo-labels are used, a thresholding function is applied

$$\mathbb{1}(P(\tilde{y}|x) > \tau)$$

where, τ is a confidence threshold (typically set to 0.9). This helps filter out low-confidence predictions, ensuring that only reliable pseudo-labels contribute to the learning process.

2.3 Final Training of YOLO11 on combined dataset

After training the Graph Neural Network (GNN) for label propagation, pseudo-labels were assigned to unlabeled data based on high-confidence predictions. These pseudo-labeled dataset were merged with the original labeled dataset, creating an expanded training dataset for fine-tuning YOLO11. The final training dataset was defined as:

$$D_{\text{final}} = D_{\text{labeled}} \cup D_{\text{pseudo}}$$

YOLO11 now learns from a larger dataset (D_{final}), improving generalization. Pseudo-labeling increases dataset diversity, reducing bias. The final model achieves better object detection performance, improving mAP@50-95.

3 Experiments and Results

The initial fine-tuning and the final training of the YOLO11l (large) was done with the same parameters. Rectified Adam (RAdam) optimizer along with Cosine Scheduler was used for stable training as the DFL Loss $\mathcal{L}_{\text{DFL}}$ was unstable. The training was done for 500 epochs with early stopping at a learning rate of 0.0005. The GNN defined in the Figure 2 was trained for 250 epochs with AdamW optimizer. The image size was used as it was given that is 512*512. The results of our experimentation are shown in the Table 1. The losses over time and the outputs of the final YOLO11 model are shown in Figure 1. The YOLOv11 inference speed breakdown includes preprocessing (0.1ms) for image loading and resizing, inference (6.6ms) for object detection, and postprocessing (7.2ms) for Non-Maximum Suppression (NMS) and filtering. With a total processing time of 13.9ms per image (72 FPS), the model is highly efficient for real-time detection.

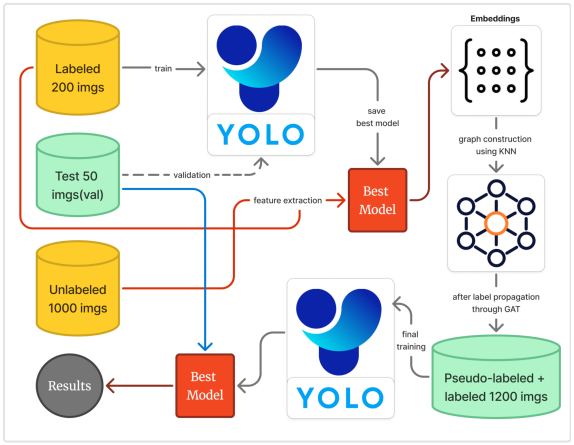

Figure 1: Complete model architecture and training process including the initial fine-tuning, graph attention, pseudo-labeling, and then the final training.

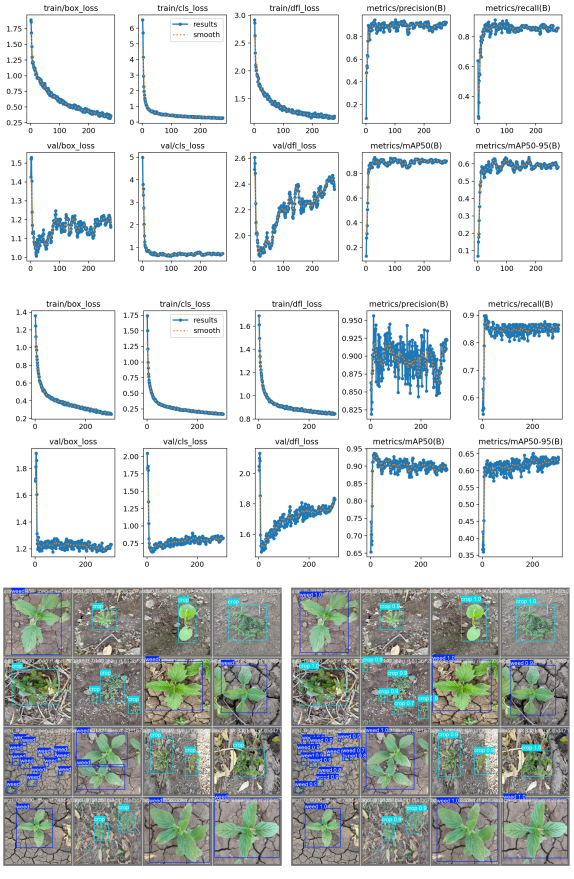

Figure 2: Top: Metrics for initial fine-tuning, Middle: Metric for final training on pseudo-labeled data, Bottom: Predicted and Labeled classes for the given test data.