Semi-Supervised Weed Detection Using YOLO

Introduction:

Weeds compete with crops for resources such as water, nutrients, and sunlight, making them a major hindrance to agricultural productivity. Effective weed management can significantly enhance crop yield and reduce farming costs. However, traditional weed detection methods, which rely on manual identification, are time-consuming and labor-intensive. **Deep learning-based object detection models** like **YOLO (You Only Look Once)** offer a fast, automated solution for identifying weeds in agricultural fields. In this project, we adopt a semi-supervised learning approach to improve weed detection by leveraging both manually labeled and pseudo-labeled data. The integration of pseudo-labels allows the model to benefit from large amounts of data without requiring extensive manual annotations.

Objective:

The primary objective of this project is to develop a YOLO-based object detection model capable of identifying weeds and crops with high accuracy. Traditional methods of manually labeling data for detection of weed for training deep learning models can be time-consuming and expensive. Hence, by utilizing a semi-supervised learning approach, we aim to minimize annotation costs while maximizing detection performance. The project also focuses on improving the model's generalization by using advanced training techniques such as **data augmentation**, **adaptive optimization**, and **regularization**.

Workflow Overview:

The workflow follows a step-by-step procedure to ensure a systematic and efficient approach to building the weed detection model. It is divided into several stages and each stage is elaborately described below.

Step 1: Labeled Data Preparation

The initial dataset comprises images that have been manually annotated with two object classes: ‘*weed’* and ‘*crop’*. Each image has an associated annotation file in the YOLO format. This format encodes bounding box information as follows:

- **Class ID** (0 for weed, 1 for crop).
- **Center coordinates (x, y)** of the bounding box(normalized by the image dimensions).
- **Width** and **height** of the bounding box (normalized by the image dimensions).

These labeled images form the foundation for initial model training. High-quality annotations are critical at this stage to ensure accurate learning during the supervised phase.

Step 2: Pseudo-Label Generation

The third step involves generating pseudo-labels for an unlabeled dataset using a petrained model. This process allows the model to learn from additional data without requiring manual annotations. The procedure is as follows:

❖ **Model Inference:**
The **YOLO11s** model, trained on the labeled dataset, is used to predict objects in each image of the unlabeled dataset. For each image, the model outputs **bounding boxes**, **class labels** and **confidence scores**.

❖ **Confidence Filtering:**
Predictions with confidence scores below a threshold **(0.5)** are discarded to reduce noise. This threshold ensures that only high-confidence predictions are included in the training dataset.

❖ **Label Formatting:**
Valid predictions are saved in text files using the YOLO format. Each line contains the **class ID**, **normalized coordinates**, and **dimensions of the bounding box**.

- **Image and Label Storage:**
  Images corresponding to valid predictions are copied to a new directory. Annotation files are saved in a separate directory under a unified dataset structure.
- **Validation:**
  The pseudo-labels are parsed and validated to detect and remove errors such as:
  1) Invalid class IDs (outside the defined range of 0–1).
  2) Bounding box coordinates that fall outside the normalized range [0, 1].

Step 3: Dataset Integration and Combination
After generating pseudo-labels, the labeled and pseudo-labeled data are combined into a single dataset. The dataset follows a directory structure compatible with YOLO's training requirements:
- **Images Directory:** Contains both labeled and pseudo-labeled images.
- **Labels Directory:** Contains annotation files for all images.

A YAML configuration file is also created to define dataset properties such as the the path to the dataset, training and validation image directories, information about the number of classes (*which was 2 in our case*) and the class names. This integrated dataset serves as the input for semi-supervised training.

Step 4: Model Training
The training process consists of two phases: initial supervised training and semi-supervised training with the combined dataset.

Initial Training:
In this phase, the YOLO model (yolo11s.pt) is trained using only the manually labeled dataset. This establishes a baseline model, which is later refined through semi-supervised training. The model is configured with basic training parameters, including the number of epochs, learning rate, and batch size.

Semi-Supervised Training
During the semi-supervised phase, the combined dataset (labeled and pseudo-labeled data) is used to enhance the model's performance. Several advanced techniques are employed to optimize training:

Parameter	**Value**	**Functionality**	**Benefits**
epochs	200	Number of complete passes through the dataset.	Longer training allows better learning of complex patterns.
imgsz	256 x 256	Input image size for the model.	Balances accuracy and training speed.
batch	32	Number of images processed in one iteration.	Larger batch sizes improve gradient stability.
lr0	0.0002	Initial learning rate for gradient descent.	A low starting rate prevents large, unstable updates.
lrf	0.002	Final learning rate for cosine scheduling.	Ensures gradual decay in learning rate over time.
optimizer	RAdam	Adaptive optimizer combining Adam and RMSProp.	Adapts learning to different features dynamically.

weight_decay	0.0001	Regularization technique to prevent overfitting.	Reduces model complexity by penalizing large weights.
momentum	0.98	Controls how much of the previous gradient is retained.	Stabilizes convergence by smoothing weight updates.
augment	True	Enables data augmentation techniques such as flipping and rotation.	Increases dataset variability, improving model robustness.

Since this model is trained on a CPU device due to hardware limitations, regular validation checks help monitor performance and prevent overfitting.

Step 5: Model Evaluation

The trained model is evaluated on a separate test set to assess its detection capabilities. Evaluation metrics provide insights into the model's precision, recall, and overall accuracy:

Metric	**Description**	**Result**
Precision	The proportion of true positive detections out of all detected objects.	0.8500
Recall	The proportion of true positives detected out of all actual objects in the dataset.	0.8300
mAP@50	Mean Average Precision at a 0.5 Intersection over Union (IoU) threshold.	0.8700
mAP@50-95	Average Precision across multiple IoU thresholds from 0.5 to 0.95.	0.7800
F1-Score	The harmonic mean of precision and recall.	0.8400
Final Score	Weighted combination of F1-Score and mAP@50-95.	0.8100

Challenges Faced and Limitations:

1. **Pseudo-Label Noise:** Predictions from the initial model introduced noisy pseudo-labels, which affected training quality. A static confidence threshold of 0.5 reduced noise but led to the exclusion of some valid labels.
2. **Class Imbalance:** The dataset contained more objects of one class than the other, causing biased predictions.
3. **Limited Hardware Resources:** Training on a CPU slowed the process and limited experimentation with larger models and higher-resolution images. Reducing image size and batch size alleviated some issues, but future training on GPUs is necessary for optimal performance.
4. **Hyperparameter Tuning Complexity:** Manual tuning of hyperparameters was time-consuming. Suboptimal configurations impacted both learning and generalization.
5. **Limited Edge Deployment:** Real-time performance on edge devices was not tested due to hardware constraints. Techniques like model pruning and quantization are required for efficient deployment.

Conclusion:

This project demonstrates the potential of semi-supervised learning to revolutionize agricultural technology. By combining labeled and pseudo-labeled data, our YOLO model achieved high accuracy in detecting weeds and crops. This scalable, cost-effective approach reduces annotation effort, making AI-driven weed detection more accessible and impactful for precision farming.