Semi-Supervised Weed Detection using YOLOv8

Team ID:

46

Traditional Weed Detection Issues:

- Requires large labeled datasets (costly & time-consuming).
- Supervised models struggle with limited data.

Challenge:

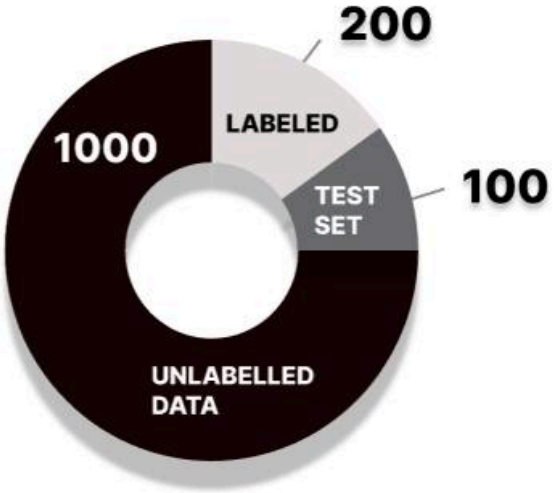
- Train an object detection model with semi-supervised learning.
- Utilize 1000 unlabeled images to improve accuracy.

Dataset:

- Labeled: 200 images (weeds & crops, YOLO format).
- Unlabeled: 1000 images.
- Test Set: 100 images..

Preprocessing Steps:

- Resize images, normalize, convert to YOLO format.
- Filter pseudo-labels with confidence threshold > 0.8.



- Labeled data contains both the images and their corresponding annotations in YOLO format, which includes the bounding box coordinates (i.e., xcenter, ycenter, width, height) along with the class labels (weed or crop)

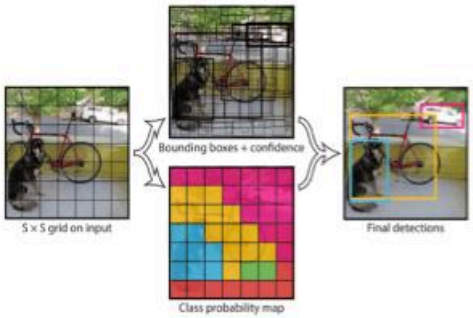
YOLOv8

You Only Look Once model

Why YOLOv8?

- YOLOv8 is a state-of-the-art deep learning model
- Majorly used in real-time object detection in computer vision applications.
- Enables Fast & accurate object detection.
- Lightweight & real-time performance.
- Single-stage detection

Input image → Feature extraction → Grid division → Bounding box prediction → Class probability prediction → Non-Maximum Suppression (NMS)



Motivation for using Yolov8

- We selected YOLO due to its status as a state-of-the-art (**SOTA**) object detection model. **Even its base model offers a strong foundation for our task**.
- **Experimental results demonstrate that fine-tuning** the last few layers of YOLO on our labeled data leads to substantial performance gains. This targeted approach maximizes the model's learning on our specific dataset.
- **The availability of labels in YOLO format drastically reduced redundant work.** This streamlined our data preparation process and saved valuable time and resources.

MODEL ARCHITECTURE

Core Components:

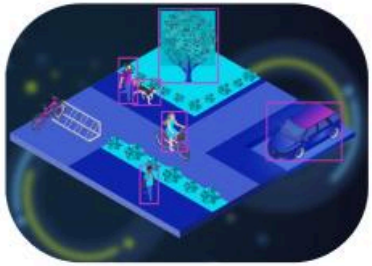
Backbone:
- Extracts rich features from the input image.

Neck:
- Combines and refines features from the backbone.

Head:
- Makes the final predictions (bounding boxes and object classes).

How it Works:

1. Input Image: The image is fed into the model.
2. Backbone: The backbone extracts features.
3. Neck: The neck combines and refines features.
4. Head: The head makes predictions (bounding boxes and classes).
5. Output: The model outputs the detected objects with their locations and labels.



BACKBONE

- The backbone is like the "eyes" of the model. It's a convolutional neural network (CNN) that learns to identify patterns and features in the image.

Key Elements:

- Convolutional Layers
- C2f Modules
- SPPF (Spatial Pyramid Pooling Fast)

NECK

- The neck acts as a "feature integrator." It takes the features from the backbone and combines them to get a better understanding of the scene.

Key Elements:

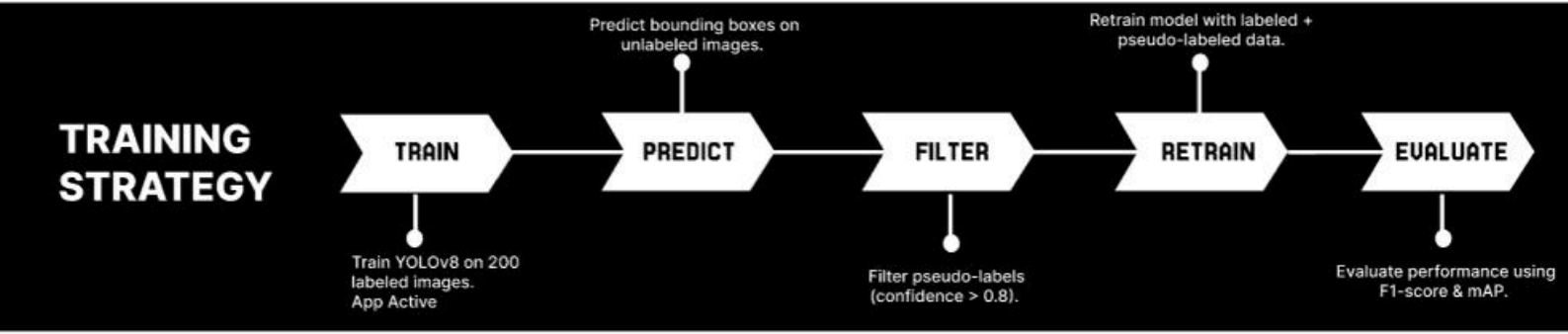
- Feature Pyramid Network (FPN)
- Path Aggregation Network (PAN)

HEAD

- The head is where the magic happens! It takes the refined features from the neck and makes the final predictions.

Key Elements:

- Decoupled Head
- Anchor-Free Detection



Baseline Model

- Trained on only labeled data.
- Low performance due to limited training samples.

Semi-Supervised Model:

- Uses pseudo-labels to expand dataset.
- Self-learning improves generalization & accuracy.

Baseline Model :
(200 labeled images only)

Score: 0.17

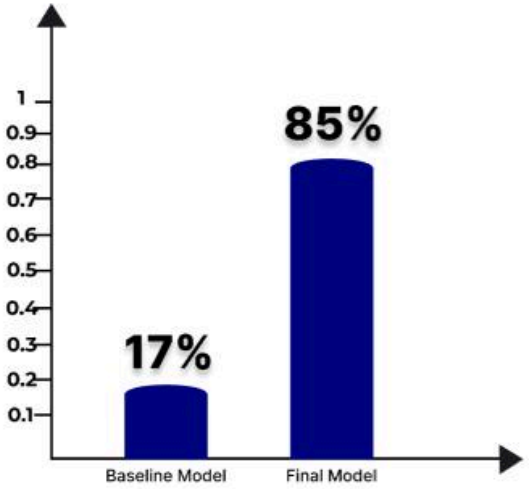
Final Model :
(Labeled + Pseudo-Labeled Data)

Score: 0.85

Performance Gain:

Improvement in detection accuracy.

mAP@[.5:.95] & F1-Score increased significantly



CHALLENGES

- Pseudo-label noise → Low-quality labels can degrade performance.
- Computational Cost → Retraining on large data is expensive.
- Hyperparameter tuning → Finding optimal confidence threshold.

SOLUTIONS

- Threshold filtering (confidence > 0.8).
- Colab Pro GPU acceleration for faster training.
- Data augmentations to improve generalization

ENHANCEMENTS

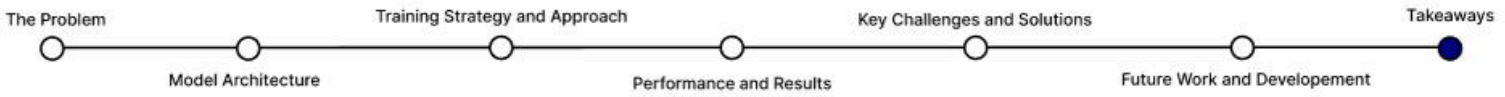
- Iterative self-training to refine pseudo-labels.
- Improve model generalization with stronger augmentations.
- Fine-tune confidence threshold dynamically.

Real-World DEPLOYEMENT

- Deploy model on edge devices (Jetson Nano, Raspberry Pi).
- Integrate with precision farming for real-time weed removal.

TAKEAWAYS

- By fine-tuning the model on a small labeled dataset and augmenting it with pseudo-labels from unlabeled data, we were able to significantly improve performance.
- This approach offers an efficient solution for scenarios with limited labeled data and holds great potential for scaling as more unlabeled data becomes available.
- Semi-Supervised Learning effectively enhances weed detection.
- YOLOv8 with pseudo-labeling led to accuracy improvement.
- Future Goal: Optimize & deploy for real-time agricultural automation.



Thank You