Journal of Asian Scientific Research
ISSN(e): 2223-1331
ISSN(p): 2226-5724
DOI: 10.55493/5003.v16i2.5901
Vol. 16, No. 2, 61-74.

URL: www.aessweb.com

A semi-supervised deep learning framework for efficient PCB defect detection confidence-thresholded self-training with YOLOv5

Zhenxia Wang[1,2+]
Nurulazlina Ramli[1]
Tzer Hwai Gilbert Thio[1]

[1]*Centre for Sustainability in Advanced Electrical and Electronics Systems, Faculty of Engineering, Built Environment and Information Technology, SEGi University, Petaling Jaya 47810, Malaysia.*
[2]*Department of Electrical Engineering, Hebei Vocational University of Technology and Engineering, Xingtai, 054000, China.*
[1,2]*Email: wangzhenxia@hevute.edu.cn*
[1]*Email: azlinaramli@segi.edu.my*
[1]*Email: gilbertthioth@segi.edu.my*

(+ Corresponding author)

Article History
Received: 3 October 2025
Revised: 8 January 2026
Accepted: 6 February 2026
Published: 25 February 2026

Keywords
Deep learning
Printed circuit board
Defect detection
Pseudo-labeling
Semi-supervised learning
YOLOv5.

ABSTRACT

Printed Circuit Board (PCB) defect inspection is critical in electronics manufacturing, yet annotating large training sets is labor-intensive and costly. This paper proposes a semi-supervised PCB defect detection framework based on YOLOv5, which leverages a small labeled dataset and a larger pool of unlabeled images through an iterative self-training pipeline. A YOLOv5 detector is first trained on the limited labeled data, then used to generate pseudo-labels on unlabeled images; high-confidence detections are retained as defect annotations, and the model is retrained on the combination of labeled and pseudo-labeled data. This pseudo-labeling and retraining cycle is repeated for multiple iterations to progressively refine the detector. The approach is evaluated on a PCB defect dataset with 100 labeled and 1,000 unlabeled images, and shows significant gains over a fully supervised baseline. The proposed semi-supervised YOLOv5 achieves 91.6% mAP@0.5 with only 100 labeled images, outperforming both the baseline (87.0% mAP) and prior semi-supervised methods, while substantially improving recall and precision. The results demonstrate that the method effectively reduces annotation effort while maintaining high detection accuracy, providing a simple, confidence-thresholded self-training strategy for deploying PCB defect detectors under limited labeling resources. This work directly supports SDG 9 by enabling cost-efficient, high-accuracy AI-based PCB inspection that strengthens intelligent and sustainable manufacturing systems.

Contribution/ Originality: This study contributes to the literature by presenting confidence-thresholded self-training for PCB defect detection with YOLOv5. It uses a methodology without teacher–student consistency. This study is among the few investigating semi-supervised PCB detection. Its primary contribution is demonstrating that high-precision pseudo-labels improve mAP. The study documents a practical pipeline.

1. INTRODUCTION

Printed circuit boards (PCBs) are fundamental to modern electronics, and even small surface flaws, such as open circuits, shorts, spurious copper, or missing holes, can cause performance degradation or field failure, underscoring the need for accurate, scalable inspection [1-3]. Manual visual checks and electrical probing are labor-intensive, subjective, and costly to scale [4, 5]. Classical automated optical inspection (AOI), based on template matching, morphology, or similarity metrics, improves throughput but relies on carefully engineered priors and often generalizes poorly across layouts and process conditions [6]. These limitations have accelerated the transition to data-driven computer vision for robust PCB defect detection [7].

Deep learning now underpins many high-accuracy PCB inspection systems [8-10]. In particular, one-stage detectors from the YOLO family combine speed with strong detection quality for industrial deployment [11-13]. Prior studies report that task-tailored YOLO variants deliver competitive precision and real-time throughput on PCB benchmarks, motivating adoption in production pipelines, especially in Asian manufacturing contexts where high line speeds and tight cost control make label-efficient inline inspection attractive [7, 8, 12].

Despite this progress, assembling large, fully annotated PCB datasets remains difficult in practice [14]. Many defect types are rare, vary with illumination/materials, or require expert confirmation, making bounding-box annotation expensive and slow [15, 16]. With limited labels, purely supervised training can overfit and generalize poorly to new boards or process drift, motivating semi-supervised learning (SSL) to leverage abundant unlabeled imagery [17].

SSL has shown strong gains in classification via self-training and consistency regularization; for example, Noisy Student iteratively improves performance with large-scale pseudo-labels [18]. Extending SSL to detection, recent semi-supervised object detection (SSOD) methods adopt both end-to-end self-training and teacher–student paradigms, e.g., STAC as a self-training baseline [19], Unbiased Teacher with EMA guidance [20], and Soft Teacher with confidence-aware objective [21]. Surveys concur that unlabeled data can substantially narrow the gap to fully supervised detectors on generic datasets [22]. However, applying SSOD to PCB inspection, which must localize multiple small and often low-contrast instances under diverse backgrounds, has been less explored; most advances are still validated on everyday-object corpora rather than industrial boards [15, 16].

In sum, there is a practical need for a label-efficient, deployment-ready SSOD recipe for PCB inspection that (i) avoids complex teacher–student mechanisms, (ii) integrates cleanly with existing YOLO-based lines, and (iii) remains robust under extreme label scarcity. This study addresses that gap with a confidence-thresholded self-training scheme built on YOLOv5 that iteratively augments training with high-precision pseudo-labels, without EMA teachers or consistency losses, consistently improving mAP and recall under limited labels. The approach is immediately relevant to Asian electronics manufacturing, where rapid, label-efficient inspection is critical to technological advancement and cost control, consistent with reported YOLO-based deployments in the region [7, 8, 12].

2. METHODOLOGY

A three-stage semi-supervised learning pipeline for PCB defect detection is developed based on the YOLOv5 one-stage object detector. The proposed approach, referred to as PCB-SS YOLOv5, iteratively self-trains the model using a small initial labeled set alongside a larger pool of unlabeled PCB images. An overview of the training flow is as follows:

Phase I (Supervised Pre-training): Train the YOLOv5 model on the available labeled PCB images to obtain an initial detector.

Phase II (Pseudo-Label Generation): Run inference with the detector on unlabeled PCB images. Apply a confidence threshold to select high-confidence predicted bounding boxes as pseudo-labels for defects.

Phase III (Semi-Supervised Re-Training): Re-train (fine-tune) the YOLOv5 model on the combined dataset containing all original labeled data plus the newly pseudo-labeled images. This process yields an improved detector, which can then be used again for Phase II of the next iteration.

These three phases (Generate pseudo-labels → retrain) can be repeated for multiple iterations to gradually refine the detector. Each iteration should ideally increase the amount of trustworthy training data (via pseudo-labels) and thereby improve detection performance until a plateau is reached.

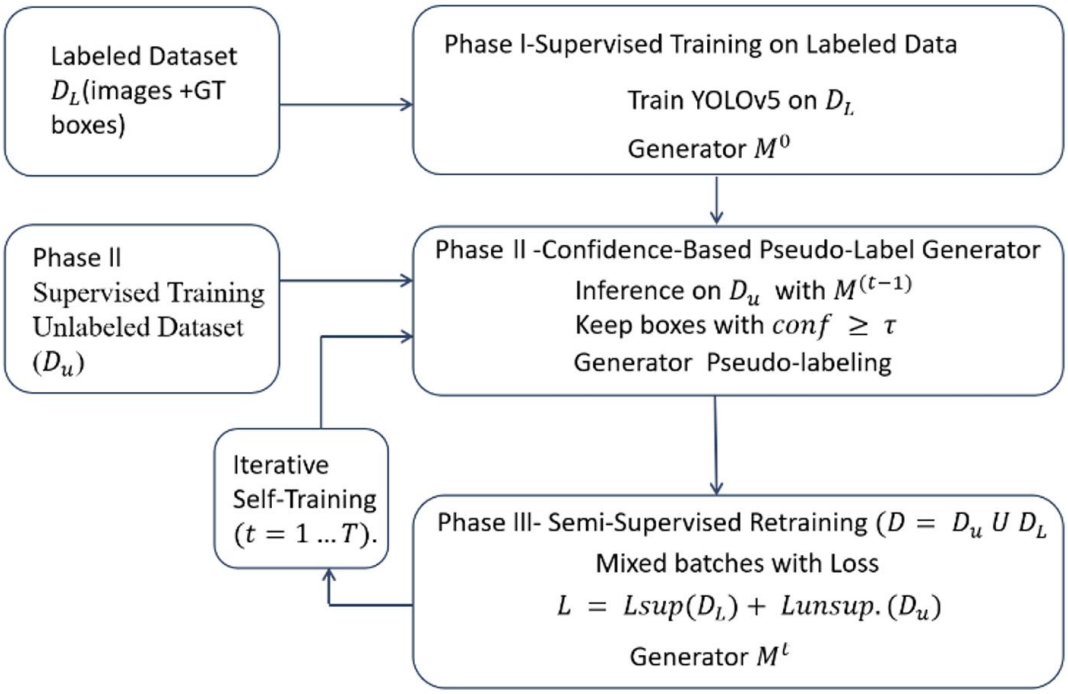

Figure 1. Confidence-thresholded self-training.

Note: Phase I: supervised training on D_L to obtain $M^{(0)}$. Phase II: run inference on D_U with $M^{(t-1)}$ and select pseudo-labels using threshold τ. Phase III: retrain on $D_L \cup D_U^{(t)}$ with mixed batches and loss $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{sup}} + \lambda\mathcal{L}_{\text{unsup}}$ to obtain $M^{(t)}$. Iterate for t=1…T.

2.1. Phase I — Supervised Training on Labeled Data

Formally, let$D_L = \{(x_i, y_i)\}_{i=1}^N$ denote the labeled dataset of PCB images, where y_i includes the ground-truth defect bounding boxes and classes for the image x_i. The YOLOv5 model is initialized with official pretrained weights (from COCO, as is standard) and then trained on D_L using the standard YOLOv5 loss functions. The total supervised loss can be written as a sum of three components.

$$\mathrm{L}_{\text{sup}} = \lambda_{obj}\mathrm{L}_{obj} + \lambda_{cls}\mathrm{L}_{cls} + \lambda_{box}\mathrm{L}_{box} \quad (1)$$

Here $\mathcal{L}_{obj}$ is the objectness loss (confidence for object vs. non-object), $\mathcal{L}_{cls}$ is the classification loss for defect classes, and $\mathcal{L}_{box}$ is the bounding-box regression loss. The positive scalars λ_{obj},λ_{cls},$\lambda_{box}$are the YOLOv5x gains that weight the objectness, classification, and box terms, respectively. Let $D_L = \{(x_i, y_i)\}_{i=1}^{n_L}$ denote the labeled training set with n_L images and annotations. To mitigate overfitting on this small labeled set, we use the standard YOLOv5 augmentations (random scaling, horizontal flips, mosaic) during Phase-I training, which yields the initial detector $M^{(0)}$ optimized only on D_L.

2.2. Phase II — Confidence-Based Pseudo-Label Generation

In Phase II, unlabeled images are utilized to generate additional training targets. Let $D_U = \{x_j\}$ be the unlabeled dataset (images without annotations). Inference is performed on each image in$x \in D_U$ using the current YOLOv5 model. (Starting with $M^{(0)}$ from Phase I, and later $M^{(t-1)}$ in iteration t). This produces a set of predicted bounding boxes with confidence scores for each image. A confidence threshold τ is subsequently applied to these predictions: any predicted box with a confidence score ≥ τ is accepted as a pseudo-labeled defect. Formally, a pseudo-label set for iteration t is obtained as follows:

$$\widehat{D}_U^{(t)} = \{(x, \hat{B}) \mid x \in D_U,\ \hat{B} = M^{(t-1)}(x),\ \text{conf}(\hat{B}) \geq \tau\} \quad (2)$$

Here $D_U = \{u_j\}_{j=1}^{n_U}$ is the unlabeled pool; $M^{(t-1)}$ is the detector before iteration t; let $\hat{B} = M^{(t-1)}(x)$ denote the predicted box set with confidence $conf(b) \in [0,1]$ and class $c(b)$ for each $b \in \hat{B}(x)$; with a fixed threshold $\tau \in [0,1]$, we keep the NMS-filtered subset$\hat{B}_\tau = \{b \in \hat{B}: conf(b) > \tau\}$; the pseudo-labeled set is $\widehat{D}_U^{(t)} = \{(x, \hat{B}_\tau(x)) \mid x \in D_U\}$

For each unlabeled image, all model-predicted bounding boxes with confidence scores greater than or equal to τ (e.g., $\tau = 0.5$) are included. Each such prediction is treated as a pseudo ground truth with a defect class label assigned by the model. This rule filters out low-confidence detections that are likely to be false positives, while retaining high-confidence predictions as new training samples. A fixed confidence threshold is deliberately chosen, avoiding more complex selection schemes. This design choice ensures that the pipeline remains fully automatic, without manual intervention for pseudo-label verification, and eliminates the need for additional hyperparameters. Unlike certain SSOD methods, the approach does not employ a teacher model with EMA or consistency losses between augmented views; instead, it relies solely on the confidence of the YOLO model's predictions to determine which unlabeled instances are incorporated.

2.3. Phase III — Semi-Supervised Retraining with Pseudo-Labels

After generating pseudo-labels, Phase III involves retraining the detector on the combined labeled and pseudo-labeled data. Let $D_{L+U}^{(t)} = D_L \cup \widehat{D}_U^{(t)}$ denote the union of the original labeled set and the newly pseudo-labeled set for the current iteration t. Mini-batches are constructed from $D_{L+U}^{(t)}$ such that both original labeled examples and pseudo-labeled examples are present in each batch. In practice, a fixed unlabeled sampling ratio ρ (e.g., $\rho = 1$, corresponding to equal numbers of labeled and unlabeled images per batch) is used to ensure balance during training.

The training loss in Phase III is defined as a combination of the supervised loss on real labels and an unsupervised loss on pseudo-labels. The notation$\mathcal{L}_{\text{sup}}$, as defined in Equation 1, is reused for the loss computed on true labeled data, while an analogous loss on pseudo-labeled data $\mathcal{L}_{\text{unsup}}$, is introduced with the same formulation, except that the "ground truth" consists of pseudo-label boxes and classes predicted by the model in Phase II. The overall semi-supervised training objective is expressed as.

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{sup}} + \lambda \mathcal{L}_{\text{unsup}} \tag{3}$$

Here, $\lambda \in \mathbb{R}_+$ is the scalar weight for the unsupervised loss computed on pseudo-labels (to avoid confusion with Equation 1 gains, we refer to this term as the *unsupervised weight*). A smaller λ down-weights potentially noisy pseudo-labels, whereas a larger λ increases their influence. Let $D_{L+U}^{(t)} = D_L \cup \widehat{D}_U^{(t)}$ denote the training set at iteration t, formed by the labeled data $D_L$and the accepted pseudo-labels $\widehat{D}_U^{(t)}$ from Equation 2. Mini-batches mix labeled and unlabeled images with an unlabeled fraction $\rho \in (0,1]$ (default $\rho = 0.5$). The detector MMM is then fine-tuned by minimizing $\mathcal{L}_{\text{total}}(\mathcal{D}_{L+U}^{(t)})$for N epochs to obtain the updated model $M^{(t)}$. This completes one self-training round; we repeat this process for $t = 1, \ldots, T$.

2.4. Iterative Self-Training

The pseudo-labeling and retraining process can be iterated to further enhance performance. After obtaining $M^{(1)}$ from the first round, the model can be fed again into Phase II to generate new pseudo-labels, potentially more numerous and of higher quality than those from the first round, and then into Phase III to obtain $M^{(2)}$, and so on. At each iteration t, the latest model $M^{(t-1)}$ is used to pseudo-label the unlabeled set. As the model improves, a greater number of defect instances in unlabeled images surpass the confidence threshold τ in later iterations. Consequently, the size of the pseudo-labeled set $\widehat{D}_U^{(t)}$ may increase, and the accuracy of those labels may also improve as false positives are corrected and false negatives (previously missed detections) begin to be identified. The threshold τ is kept fixed across iterations for simplicity. In practice, an organic growth in both the quantity and quality of pseudo-labels is observed over a few iterations.

Iteration cannot continue indefinitely, as improvements eventually taper off. Empirical results indicate that $T = 3$ self-training iterations yield clear gains on the dataset, while going beyond three provides diminishing returns and, in some cases, a slight decrease in certain metrics due to the accumulation of incorrect pseudo-labels. Stopping after three iterations thus offers the optimal balance between maximizing the use of unlabeled data and avoiding overfitting to noise.

Algorithm 1 outlines the complete training procedure for the PCB-SS YOLOv5 framework. The YOLOv5 model parameters are initialized with pretrained weights and first trained on the labeled set D_L alone (Phase I). The iterative loop then begins: for each iteration $t = 1 \cdots T$, pseudo-labels are generated from the unlabeled set (Phase II) using the current model, combined with the labeled data, and used to retrain the model (Phase III). An optional validation step may be conducted at each round to monitor progress, and after all iterations, the final model (or the best-performing model on validation) is selected for evaluation on the test set.

Algorithm 1: Confidence-Thresholded Self-Training for Semi-Supervised PCB Defect Detection

Input: Labeled dataset D_L; Unlabeled dataset D_U; Confidence threshold τ; Unlabeled batch fraction ρ; Unlabeled loss weight λ; Number of iterations T.

Output: Trained detector $M^{(T)}$ (after T self-training rounds).

Initialize YOLOv5 model M with pretrained weights.

Phase I – Supervised Training: Train M on D_L alone (optimize $\mathcal{L}_{\text{sup}}$, Equation 1) for N epochs; obtain initial model $M^{(0)}$.

For t = 1 to T do.

a. Phase II – Pseudo-Label Generation: Use the model $M^{(t-1)}$ to infer on all $x \in D_U$. Construct pseudo-labeled set $\widehat{D}_U^{(t)}$ by selecting predictions with confidence ≥ τ (Equation 2).

b. Form a combined dataset $D_{L+U}^{(t)} = D_L \cup \widehat{D}_U^{(t)}$.

c. Phase III – Semi-Supervised Retraining: Fine-tune model M on $D_{L+U}^{(t)}$ for N epochs, using mixed batches (ratio ρ of unlabeled: labeled). Optimize total loss $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{sup}} + \lambda \mathcal{L}_{\text{unsup}}$ (Equation 3) to obtain an updated model $M^{(t)}$.

d. (Optional) Evaluate $M^{(t)}$ on validation set; keep best checkpoint.

end for

Return the final model $M^{(T)}$ (or best validation model).

2.5. Methodological Contrast to Prior SSOD

Unlike teacher–student SSOD approaches (e.g., Soft Teacher, Unbiased Teacher), our pipeline does not maintain an EMA teacher, impose view-consistency losses, or calibrate/weight losses by teacher confidence. Instead, we adopt a single-model, fixed-threshold pseudo-label selection (τ) and a simple two-loss objective with an unsupervised weight α. This design reduces engineering overhead (one network, fewer hyperparameters), avoids degenerate behavior when teacher/student drift, and proves sufficient on PCB imagery where high-contrast, localized defects benefit from high-precision pseudo-labels. In contrast, teacher–student methods typically rely on consistency under strong augmentations and dynamic confidence weighting, which can excel on generic object corpora but add complexity for factory deployment.

3. EXPERIMENTAL EVALUATION

3.1. Data and Training Setup

Dataset: The evaluation is conducted on a PCB defect dataset [9] in Pascal VOC format, referred to as VOC-PCB. This dataset contains high-resolution images of printed circuit boards with various defect types, including missing holes, mouse bites, open circuits, shorts, spurs, and spurious copper. The data are divided into labeled, unlabeled, validation, and test sets, as summarized in Table 1.

Table 1. *VOCPCB* dataset splits and usage across training phases.

Subset	Images	Purpose
Labeled train	100	Used for Phase I supervised training (initial model)
Unlabeled pool	1000	Used for Phase II pseudo-label generation in semi-supervised training
Validation	500	Model selection, hyperparameter tuning, and early stopping
Test	2134	Held-out evaluation of final performance (not used in training)

This split reflects an extreme label-scarce scenario, with only 200 labeled images (containing on the order of a few hundred annotated defect instances) and a large pool of 1,000 unlabeled PCB images to be exploited by our semi-supervised approach. All experiments and ablations use this same dataset split for consistency.

Training protocol: In Phase I, a YOLOv5x model (the extra-large variant of YOLOv5) was initialized with pretrained weights and trained on 100 labeled training images for 100 epochs. This was followed by $T = 3$ self-training iterations (Phase II + Phase III cycles). In each iteration, Phase II involved running inference on 1,000 unlabeled images with the current model to produce pseudo-labels, using a fixed confidence threshold $\tau = 0.5$ to accept predictions. In Phase III, the model was retrained on the combined labeled and pseudo-labeled set for an additional 200 epochs. Performance plateaued by the third round, indicating that three iterations were sufficient (see results in Section 3.3.2).

Optimization and implementation details. Stochastic Gradient Descent (SGD) was used with an initial learning rate of 0.01, momentum of 0.937, and weight decay of 0.0005, consistent with the default YOLOv5 hyperparameters. Each training run used a batch size of 32 and an input resolution of 640×640 pixels, with images resized using padding. Standard YOLOv5 data augmentations, including mosaic augmentation, random scale jitters, and horizontal flips, were applied during all training phases to enhance generalization. Training was performed on a single RTX 4090 GPU with 24 GB of memory; each self-training iteration of 200 epochs on the combined dataset required several hours. Model checkpoints were saved at each iteration and evaluated on the validation set to track progress, with the best-performing model selected for final evaluation on the test set.

Software environment. Experiments were conducted with Python 3.10 and PyTorch (CUDA 12.1 build) on Ubuntu Linux.

Statistical Analysis & Robustness. Each experiment is repeated with six independent random seeds. We report mean ± standard deviation (s.d.) for mAP@0.5 and provide 95% confidence intervals (CI) using a two-sided Student's t-interval (df=5).

3.2. Evaluation Metrics

Precision (P): proportion of predicted defect boxes that are correct.

$$\text{Precision} = \frac{TP}{TP+FP} \quad (4)$$

Here, TP and FP denote the numbers of true and false positive detections after NMS; a detection b is counted as TP if it uniquely matches a ground-truth box b* of the same class with IoU ≥ δ; δ is the IoU threshold (default δ=0.5); Precision is computed by aggregating TP and FP across all classes (micro-average) unless otherwise stated.

Recall (R): proportion of ground-truth defects detected.

$$\text{Recall} = \frac{TP}{TP+FN} \quad (5)$$

Here, FN is the number of false negatives under the same matching rule and thresholds δ as in Equation 4; a ground-truth box not matched by any prediction with IoU ≥ δ is counted as FN. Recall is reported as a micro-average across classes, unless otherwise noted.

Intersection over Union (IoU): overlap between a predicted and ground-truth box.

$$IoU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \quad (6)$$

Here, B_p and B_{gt} are the predicted and ground-truth boxes, respectively.

AP: area under the precision–recall curve for a class at a given IoU threshold.

mAP@0.5: mean AP over all classes at $IoU = 0.5$.

Table 2. Comparison of semi-supervised object detection methods on VOC-PCB under a label-limited protocol (100 labeled, 1000 unlabeled). Reported values are mAP@0.5 on the held-out test set. STAC adopts augmentation-driven self-training; Soft Teacher uses an EMA teacher with confidence-aware objectives; Ours applies fixed-threshold, iterative self-training without EMA or consistency losses.

Approach	Teacher–Student?	Pseudo-label rule	Dataset/labels	mAP@0.5	Distinctive point
STAC	No	Augmentation + self-training	VOC-PCB, 100 L + 1000 U	0.901	Early SSOD self-training baseline
Soft teacher	Yes (EMA)	Confidence-weighted losses	VOC-PCB, 100 L + 1000 U	0.904	Strong EMA teacher, consistency
Ours	No	Fixed confidence threshold; iterative	VOC-PCB, 100 L + 1000 U	**0.916**	Simplest pipeline; no EMA/Consistency

3.3. Results and Discussion

3.3.1. Overall Performance on VOC-PCB

The proposed semi-supervised YOLOv5 model (after three self-training iterations $T=3$) is first compared against a supervised baseline trained solely on the 100 labeled images. Table 2 compares our proposed semi-supervised YOLOv5 with STAC and Soft Teacher on the VOC-PCB dataset in terms of mAP@0.5. Table 3 presents the results of the PCB test set. The semi-supervised approach achieves a clear improvement on all metrics. The baseline YOLOv5x (100 labeled images, no unlabeled data) attains mAP@0.5 = 0.870, Precision = 0.919, and Recall = 0.835. In contrast, our semi-supervised model reaches mAP@0.5 = 0.916, Precision = 0.958, and Recall = 0.876. This is a substantial gain of +4.6 mAP points (from 87.0% to 91.6%), +3.9-point increase in precision, and +4.1-point increase in recall. The higher Recall indicates the semi-supervised model is detecting more defect instances than the baseline (fewer misses), thanks to having learned from the additional unlabeled data. At the same time, the Precision improvement suggests it is also making fewer false positive errors – likely because the decision boundary for what constitutes a defect is refined with more examples, including the pseudo-labeled ones. Together, these boosts in Recall and Precision lead to a significantly higher mAP, reflecting better overall defect detection performance.

Table 3. Overall defect detection performance on the VOC-PCB test set. The semi-supervised YOLOv5 (Ours, after T = 3 iterations with 100 labeled + 1000 unlabeled images) is compared to a fully supervised baseline (100 labeled images only) and prior semi-supervised methods. Metrics reported are mean Average Precision at IoU 0.5 (mAP@0.5), Precision, and Recall.

Method	mAP@0.5	Precision	Recall
Supervised YOLOv5 (100 labeled)	0.87	0.919	0.835
+ STAC (CVPR 2021) [19]	0.901	0.930	0.842
+ Soft Teacher (ICCV 2021) [21]	0.904	0.943	0.851
+ Ours (Semi-Supervised, T=3)	0.916	0.958	0.876

For context, Table 3 also reports the performance of two recent semi-supervised detection methods adapted to the dataset: STAC and a method based on the Soft Teacher framework. These were implemented using the same 100 labeled images + 1000 unlabeled images setup. STAC achieves mAP@0.5 = 0.901, Precision = 0.930, Recall = 0.842 on our PCB test set, which indeed is higher than the supervised baseline (confirming that even a relatively basic SSL method like STAC provides benefit).

The Soft Teacher-based approach performs slightly better, with mAP@0.5 = 0.904, Precision = 0.943, Recall = 0.851. Notably, our simpler self-training approach outperforms both STAC and Soft Teacher in this case, achieving the highest mAP as well as Precision and Recall among the compared methods. This demonstrates that a carefully tuned confidence-thresholded self-training can be as effective as more complex consistency-based or teacher-student methods, at least for the task of PCB defect detection. The strong performance of the method can be attributed to the high precision of the pseudo-labels (controlled by the threshold) and the iterative process that gradually expands the training set with reliable examples.

Table 4. Results are from six independent random seeds; values are percentages. Metrics are reported as mean ± standard deviation and 95% t-CI (df=5).

Number of experiments	1	2	3	4	5	6
Results (%)	91.6	90.8	90.6	92.0	92.1	91.4
Mean value of results (%)	91.42					
Standard Deviation (%)	0.615					
95% CI (%)	[90.77,92.06]					

Table 4 presents the statistical robustness analysis of mAP@0.5 over six independent runs, reporting the mean, standard deviation, and 95% confidence interval. The best run reached 92.10%, and the worst was 90.60%; the 1.50-point spread indicates modest seed-to-seed fluctuation. Across six independent seeds, the average mAP@0.5 was 91.42% ± 0.62%, with a 95% *t*-CI (df=5) of [90.77%, 92.06%]. After removing the single best and worst runs, the remaining four results ranged from 90.8% to 92.0% (mean 91.45%, s.d. 0.50%), suggesting stable performance under the unified training/evaluation protocol.

3.3.2. Impact of the Number of Self-Training Iterations T

The evolution of performance with the number of self-training iterations is analyzed next. Starting from $T = 0$ (corresponding to the supervised baseline with no unlabeled data), pseudo-labeling rounds are incrementally added, and the model is evaluated after each round on the test set. The results are summarized in Table 5.

Table 5. Effect of the number of self-training iterations (T) on test set performance. $T = 0$ corresponds to no semi-supervised learning (supervised baseline).

Self-training iterations (T)	mAP@0.5	Precision	Recall
0(Supervised only)	0.87	0.919	0.835
1	0.907	0.934	0.866
2	0.911	0.938	0.892
3	0.916	0.958	0.876
4	0.915	0.935	0.904
5	0.912	0.953	0.885

With $T = 1$ (one round of self-training), a substantial improvement in accuracy is observed: mAP@0.5 increases from 0.870 to 0.907, precision from 0.919 to 0.934, and recall from 0.835 to 0.866. This single iteration of unlabeled data utilization provides the biggest boost, indicating that even the first wave of pseudo-labels (despite being generated by a relatively weak initial model) adds significant new information for the detector. Going to $T = 2$, The trend continues but with smaller gains: mAP improves slightly to 0.911, Precision to 0.938, and recall to 0.892. By $T = 3$, the model reaches a mAP of 0.916, a precision of 0.958, and a recall of 0.876, representing the best result observed. Notably, from $T = 2$ to $T = 3$, Precision jumps quite a bit (0.938 → 0.958), while recall sees a slight dip (0.892 → 0.876). This could indicate that, by the third iteration, the model becomes more conservative (fewer false positives, hence higher precision) at the minor expense of missing a few more defects (lower recall). The overall mAP still improves $T = 3$, because the precision gain outweighs the recall loss.

To further investigate, the experiment was extended to $T = 4$ and $T = 5$ rounds. At $T = 4$, the model achieved a mAP of 0.915 (essentially unchanged from 0.916, within run-to-run variation), with a precision of 0.935 and recall of 0.904. At $T = 5$, mAP slipped slightly to 0.912, precision 0.953, recall 0.885. The fluctuations observed beyond $T = 3$ suggest that performance saturation has been reached. After three iterations, the model appears to have extracted most of the useful signal from the unlabeled data, and additional pseudo-labeling rounds introduce as many noisy labels as correct ones, resulting in no net benefit. The slight decrease in mAP at $T = 5$ may be attributed to the accumulation of incorrect pseudo-labels that begin to hinder training, although no catastrophic drop is observed, indicating good stability of the method.

In summary, one or two self-training iterations provide the majority of the benefit, and three iterations are sufficient to maximize performance on this dataset. This experiment justifies our choice of $T = 3$ for the main results. It also indicates that in practice, one does not need an unbounded iterative process; a few rounds are enough before diminishing returns set in. This insight can be useful for deciding how many self-training cycles to run when deploying such a system, as each additional iteration has a computational cost. In our case, stopping at $T = 3$ offered an excellent trade-off between accuracy and training time.

3.3.3. Impact of Labeled-Set Size

Up to this point, 100 labeled images have been used as the default size of D_L. This section examines the performance of the semi-supervised approach when the number of labeled images is varied. This simulates different annotation budgets, from extremely scarce labels to a generously labeled dataset. For this experiment, labeled set sizes of 20, 50, 100, 200, 400, and 1000 images are considered, while the unlabeled pool is kept fixed at 1000 images (except in the 1000-labeled case, where no unlabeled data remains). In each scenario, the full algorithm is executed for $T = 3$ iterations and evaluated on the test set. The results are summarized in Table 6.

Table 6. Effect of the labeled training set size on semi-supervised performance (test set results, with T = 3 iterations and a fixed 1000 unlabeled images).

Labeled images	mAP@0.5	Precision	Recall
20	0.553	0.618	0.561
50	0.814	0.915	0.789
100	0.916	0.958	0.876
200	0.951	0.965	0.937
400	0.967	0.964	0.961
1000(All labeled, no SSL)	0.980	0.977	0.974

Even with as few as 20 labeled images, semi-supervised training can learn meaningful representations, achieving mAP@0.5 = 0.553, Precision = 0.618, and Recall = 0.561. Although these numbers are relatively low in an absolute sense, indicating the model detects only approximately 56% of defects with about 62% precision, they are significantly better than chance and establish a baseline under conditions of extreme label scarcity. Notably, increasing the number of labeled images from 20 to 50 results in a substantial performance improvement, mAP rises to 0.814, Precision to 0.915, and recall to 0.789. This indicates that the initial few dozen labeled examples are highly informative, and having at least around 50 images provides the model with sufficient grounding to begin generalizing and effectively utilizing unlabeled data. At 50 labels, the precision is already very high at 91.5%, meaning the model makes few false detections; the recall is lower at 78.9%, implying some defects are still missed. Overall, the mean Average Precision (mAP) of approximately 81% is within a usable range, demonstrating the model's practical effectiveness in defect detection tasks.

With 100 labeled images (the main setting), the model achieves a mean Average Precision (mAP) of 0.916, a precision of 0.958, and a recall of 0.876, as previously discussed. Increasing the number of labels to 200 further improves performance, with a mAP of 0.951, precision of 0.965, and recall of 0.937. The semi-supervised approach continues to yield strong results as additional labels are introduced, indicating that a larger labeled set provides a stronger starting point while the unlabeled data offers diminishing yet still positive returns. At 400 labeled images, the model reaches a mAP of 0.967, a precision of 0.964, and a recall of 0.961. Interestingly, precision does not increase from 200 to 400 (it stays around 0.964–0.965), but recall increases from 93.7% to 96.1%, indicating that adding those extra 200 labels helped catch a few more of the remaining defects. Finally, with 1000 labeled images, which is effectively the fully supervised case on the entire training set, the model achieves a mAP of 0.980, a precision of 0.977, and a recall of 0.974. This can be regarded as the upper bound (since all available images are labeled and used in training). The gap between 400 and 1000 labels is relatively small: 0.967 to 0.980 in mAP (only +1.3 points), and

precision actually decreases slightly (0.977 vs 0.964, which is likely not significant, with both around 96-97%), while recall increases from 0.961 to 0.974 (+1.3 points). This indicates a clear trend of diminishing returns; beyond a few hundred labeled images, each additional annotation contributes only marginal gains to accuracy.

Several useful insights can be derived from these results. First, semi-supervised learning is beneficial across the board, from 20 labels up to 400 labels and beyond. Even when labeling extremely few samples (20 or 50), the ability to incorporate unlabeled data helps the model achieve much higher accuracy than it would with those labels alone. (For reference, a supervised YOLOv5 trained on only 50 images would be expected to perform substantially worse than the 81.4% mAP obtained with SSL. Prior studies have noted that detection models often struggle to converge on such small datasets in the absence of unlabeled data.) Second, as the labeled set grows, the relative improvement from semi-supervised training shrinks because the supervised baseline itself gets better. In the limit of 1000 labels (which in our case means no unlabeled data left), our method naturally reverts to fully supervised training (and yields the highest absolute results). But importantly, our semi-supervised approach reaches very close to that fully supervised upper bound with far fewer labels. For instance, with only 200 labeled images (20% of the dataset), the model achieves a mAP of 0.951, corresponding to approximately 97% of the fully supervised mAP (0.980) and reflecting a deficit of about three mAP points. With 400 labeled images (40% of the dataset), the performance increases to 0.967 mAP, representing 98.7% of the fully supervised result. In practical terms, this means 60% of the manual labeling effort can be saved while still obtaining nearly the same accuracy by using our semi-supervised learning strategy. This is a highly attractive proposition for real-world deployments, where labeling hundreds of PCB images is much cheaper and faster than labeling thousands.

It is also interesting to note how Precision and Recall behave in Table 6. Precision remains relatively high (>0.95) once 50 or more labels are available and reaches a peak of 0.977 under full supervision. Recall shows a more steady increase as labels increase, going from 0.561 at 20 labels up to 0.974 at 1000 labels. This suggests that the main benefit of having more labeled data is in finding more defects (raising recall), while precision is easier to achieve even with fewer labels (perhaps because the model learns to avoid obvious false positives early on, and pseudo-labeling with a high confidence threshold further reinforces precision). For mission-critical applications, one might decide on a target recall (say 95%) and observe from Table 6 that approximately 400 labeled images were enough to reach about 96% recall in our case. This kind of analysis can guide how many images need to be labeled to meet a certain detection quality, using our semi-supervised training approach.

3.3.4. Qualitative Results and Error Analysis

Qualitative inspection of detection examples was conducted to analyze the types of errors made by the baseline compared with the semi-supervised model. Overall, the visual results corroborate the quantitative improvements reported. The semi-supervised YOLOv5 model produces tighter bounding boxes around small defect regions and detects certain subtle defects that the supervised model often misses. For instance, in images containing a faint hairline open circuit or a tiny spur defect, the baseline model (trained on 100 images) sometimes failed to detect those, likely due to insufficient examples of such subtle defects in the labeled set. After three rounds of self-training, the model accumulated a substantial number of pseudo-labeled examples of these defect types from the unlabeled data, and it successfully detected the majority of them, producing accurate bounding boxes even around very small blemishes.

Typical scenarios are illustrated in Figure 2. A spurious copper trace (extraneous copper) that was misclassified or overlooked by the baseline is correctly identified by the semi-supervised model. In addition, a significant reduction in false positives is observed. The baseline had a tendency to fire occasionally on regular structures of the PCB (like via holes or circuit patterns) that are not defects but might superficially resemble one. With the benefit of unlabeled data, the model seems to have learned a more robust representation that differentiates defect vs. non-defect patterns, resulting in a cleaner output with fewer false alarms. Figure 2 illustrates typical qualitative detection results, where

the proposed semi-supervised model (blue boxes) reduces false positives and missed defects compared with the supervised baseline (red boxes).

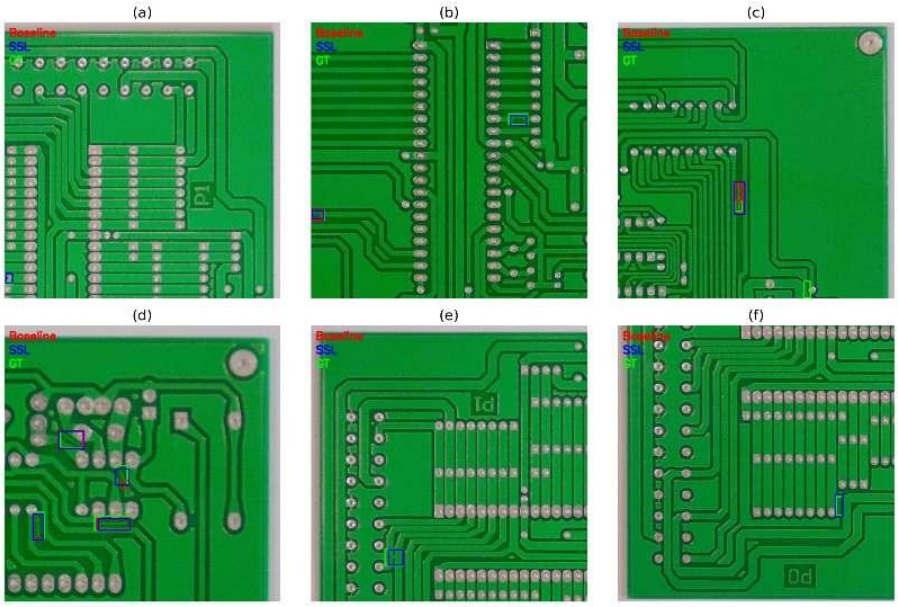

Figure 2. Typical scenarios. (a–f) Each panel shows a PCB test image with ground-truth boxes (Green), baseline predictions (Red), and semi-supervised predictions (Blue). Semi-supervised training reduces false positives on regular, non-defect structures.

Despite these improvements, some challenging cases remain. The most common errors by the semi-supervised model are: (i) missed detections of extremely small or low-contrast defects. For example, a very tiny pinhole or a slight discoloration might still be missed if it never appeared clearly in the training/pseudo data. These errors show up as false negatives (impacting recall). (ii) Occasional false positives on benign patterns, typically when a regular PCB feature mimics the shape of a defect. For instance, a complex copper trace corner might be incorrectly flagged as a spurious copper defect, or a solder bead might be mistakenly detected as a short. Such errors are rarer after self-training but still occur in a minority of images.

These failure modes suggest possible directions for further refinement. One idea is to adjust the confidence threshold τ on a per-class basis: if a certain defect type (say, open circuit) is still being under-detected, one could use a slightly lower τ for that class in Phase II to encourage more pseudo-labels of that type, thereby improving recall for it. Conversely, if another class is yielding more false positives, one might raise τ or apply a stricter criterion for that class. Another idea is to impose a penalty for spatially repeated false positives – for example, if the model erroneously flags a specific location as a defect in iteration 1, and repeats that in further iterations, one could identify that region and treat it with skepticism (since it might be a systematic false trigger, not an actual defect). Implementing such heuristics could further improve the precision without hurting recall. Nonetheless, even in its current form (with a fixed threshold and no special-case handling), our method yielded consistent and significant gains over the baseline. The qualitative results emphasize that leveraging unlabeled data through self-training has made the detector more robust: defect localization is more precise, and fewer true defects go unnoticed.

4. CONCLUSION

A confidence-thresholded semi-supervised learning approach for PCB defect detection is introduced, utilizing YOLOv5 as the backbone detector and an iterative self-training paradigm to leverage unlabeled data. The method is notable for its simplicity, as it does not require multiple networks or complex loss functions, but only a sensible threshold to filter model predictions as pseudo-labels. Experiments on a PCB inspection dataset demonstrate that this approach can significantly improve detection performance under extremely limited labeling. With only 100

labeled PCB images and 1,000 unlabeled images, the semi-supervised model achieves a mean Average Precision at IoU 0.5 (mAP@0.5) of 91.6%, outperforming a fully supervised baseline by approximately five percentage points and surpassing more elaborate recent methods such as STAC and Soft Teacher on the same data. The results further indicate that the benefits of unlabeled data are most pronounced when labels are scarce (e.g., 50 or 100 images), while still providing incremental gains as more labels are added, enabling near fully supervised accuracy with only a fraction of the dataset annotated. These findings highlight the practical value of semi-supervised learning for visual inspection, demonstrating that manufacturers can substantially reduce annotation costs while maintaining high defect detection accuracy.

Immediate relevance to Asian electronics manufacturing is evident, given that rapid, label-efficient inspection is critical to technological advancement and cost control. Potential limitations include error accumulation from pseudo-labels beyond three rounds and sensitivity to τ; future work may examine class-adaptive thresholds and hybrid teacher–student variants.

In terms of contributions to the field, our work provides a case study of self-training in an industrial application and confirms that even a relatively straightforward pipeline can yield state-of-the-art results when properly tuned. The key insights include the importance of high-precision pseudo-label selection (to avoid introducing too much noise) and the finding that a handful of self-training iterations (in our case, three) are sufficient before reaching diminishing returns. An analysis of common error modes on PCB defects shows that most remaining errors are attributable either to extremely subtle defects or to look-alike false positives. These observations suggest possible future research directions. One direction is to introduce adaptive pseudo-labeling strategies, for example, class-specific confidence thresholds or dynamic threshold adjustment based on the proportion of false positives versus false negatives observed.

This could help in addressing the residual recall gap for the hardest defect types. Another direction is to combine our self-training approach with other forms of weak supervision or active learning. An interactive training loop could be envisioned where an expert only verifies or corrects a small subset of the pseudo-labels (perhaps those of lowest confidence), further improving the quality of the training data. This would still be far less effort than full annotation and could push performance closer to the fully-supervised upper bound.

Moreover, while our study focused on PCB surface defects, the approach is generic and can be extended to other industrial inspection tasks – such as detecting defects in solar panels, steel surfaces, or semiconductor wafers – where unlabeled data is often abundant but labeled data is limited. Future work could evaluate the method's generality across such domains and also explore integration with advanced architectures (e.g., using the latest YOLOv7/YOLOv8 or transformer-based detectors) to further boost the absolute performance. In conclusion, our work reinforces that semi-supervised object detection is a powerful paradigm for real-world applications, enabling high-performance detection "on a budget" of annotations.

These findings are expected to encourage wider adoption of semi-supervised strategies in industrial AI systems and to inspire continued research into simple yet effective SSL techniques. By enabling accurate, label-efficient, and scalable AI-driven defect inspection suited for modern production environments, this work directly advances *SDG 9: Industry, Innovation and Infrastructure*, supporting smarter, more resilient, and sustainable manufacturing systems.

Funding: This study received no specific financial support.
Institutional Review Board Statement: Not applicable.
Transparency: The authors state that the manuscript is honest, truthful, and transparent, that no key aspects of the investigation have been omitted, and that any differences from the study as planned have been clarified. This study followed all writing ethics.
Competing Interests: The authors declare that they have no competing interests.
Authors' Contributions: All authors contributed equally to the conception and design of the study. All authors have read and agreed to the published version of the manuscript.

REFERENCES

[1] Y. Zhou, M. Yuan, J. Zhang, G. Ding, and S. Qin, "Review of vision-based defect detection research and its perspectives for printed circuit board," *Journal of Manufacturing Systems*, vol. 70, pp. 557-578, 2023. https://doi.org/10.1016/j.jmsy.2023.08.019

[2] Q. Ling and N. A. M. Isa, "Printed circuit board defect detection methods based on image processing, machine learning and deep learning: A survey," *IEEE Access*, vol. 11, pp. 15921-15944, 2023. https://doi.org/10.1109/ACCESS.2023.3245093

[3] M. A. E. Abd Al Rahman and A. Mousavi, "A review and analysis of automatic optical inspection and quality monitoring methods in electronics industry," *IEEE Access*, vol. 8, pp. 183192-183271, 2020. https://doi.org/10.1109/ACCESS.2020.3029127

[4] M. Moganti, F. Ercal, C. H. Dagli, and S. Tsunekawa, "Automatic PCB inspection algorithms: A survey," *Computer Vision and Image Understanding*, vol. 63, no. 2, pp. 287-313, 1996. https://doi.org/10.1006/cviu.1996.0020

[5] V. H. Gaidhane, Y. V. Hote, and V. Singh, "An efficient similarity measure approach for PCB surface defect detection," *Pattern Analysis and Applications*, vol. 21, no. 1, pp. 277-289, 2018. https://doi.org/10.1007/s10044-017-0640-9

[6] E. H. Yuk, S. H. Park, C.-S. Park, and J.-G. Baek, "Feature-learning-based printed circuit board inspection via speeded-up robust features and random forest," *Applied Sciences*, vol. 8, no. 6, p. 932, 2018. https://doi.org/10.3390/app8060932

[7] X. Liao, S. Lv, D. Li, Y. Luo, Z. Zhu, and C. Jiang, "YOLOv4-MN3 for PCB surface defect detection," *Applied Sciences*, vol. 11, no. 24, p. 11701, 2021. https://doi.org/10.3390/app112411701

[8] V. A. Adibhatla, H.-C. Chih, C.-C. Hsu, J. Cheng, M. F. Abbod, and J.-S. Shieh, "Defect detection in printed circuit boards using you-only-look-once convolutional neural networks," *Electronics*, vol. 9, no. 9, p. 1547, 2020. https://doi.org/10.3390/electronics9091547

[9] R. Ding, L. Dai, G. Li, and H. Liu, "TDD-net: A tiny defect detection network for printed circuit boards," *CAAI Transactions on Intelligence Technology*, vol. 4, no. 2, pp. 110–116, 2019. https://doi.org/10.1049/trit.2019.0019

[10] B. Hu and J. Wang, "Detection of PCB surface defects with improved faster-RCNN and feature pyramid network," *IEEE Access*, vol. 8, pp. 108335-108345, 2020. https://doi.org/10.1109/ACCESS.2020.3001349

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779-788.

[12] V. A. Adibhatla, H.-C. Chih, C.-C. Hsu, J. Cheng, M. F. Abbod, and J.-S. Shieh, "Applying deep learning to defect detection in printed circuit boards via a newest model of you-only-look-once," *Mathematical Biosciences and Engineering*, vol. 18, no. 4, pp. 4411-4428, 2021. https://doi.org/10.3934/mbe.2021223

[13] B. Du, F. Wan, G. Lei, L. Xu, C. Xu, and Y. Xiong, "YOLO-MBBi: PCB surface defect detection method based on enhanced YOLOv5," *Electronics*, vol. 12, no. 13, p. 2821, 2023. https://doi.org/10.3390/electronics12132821

[14] S. Tang, F. He, X. Huang, and J. Yang, "Online PCB defect detector on a new PCB defect dataset," *arXiv preprint arXiv:1902.06197*, 2019. https://doi.org/10.48550/arXiv.1902.06197

[15] Y. Wang, Z. Liu, and S. Lian, "Semi-supervised object detection: A survey on recent research and progress," *arXiv preprint arXiv:2306.14106*, 2023. https://doi.org/10.48550/arXiv.2306.14106

[16] T. T. A. Pham, D. K. T. Thoi, H. Choi, and S. Park, "Defect detection in printed circuit boards using semi-supervised learning," *Sensors*, vol. 23, no. 6, p. 3246, 2023. https://doi.org/10.3390/s23063246

[17] Y.-C. Liu *et al.*, "Unbiased teacher for semi-supervised object detection," *arXiv preprint arXiv:2102.09480*, 2021. https://doi.org/10.48550/arXiv.2102.09480

[18] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, "Self-training with noisy student improves ImageNet classification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10687-10698.

[19] Q. Zhou, C. Yu, Z. Wang, Q. Qian, and H. Li, "Instant-teaching: An end-to-end semi-supervised object detection framework," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4081-4090.

[20] Y. Tang, W. Chen, Y. Luo, and Y. Zhang, "Humble teachers teach better students for semi-supervised object detection,"

in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3132-3141.

[21] M. Xu *et al.*, "End-to-end semi-supervised object detection with soft teacher," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 3060-3069.

[22] Y. Shao *et al.*, "Semi-supervised object detection with multi-scale regularization and bounding box re-prediction," *Electronics*, vol. 13, no. 1, p. 221, 2024. https://doi.org/10.3390/electronics13010221