International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 10 | Oct - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

# Adaptive High-Resolution PCB Defect Detection Using YOLO11n with Tile-Based Processing and Global Coordinate Reconstruction

**Dr. Mrs. A. P. Rathkanthiwar , Rohan Virutkar, Rupesh Ingale , Shreya Pohekar ,Siddhant Chikate ,**

Department of ECE ,Priyadarshini College of Engineering, Nagpur, Maharashtra, India|
e-mail: anagharathkanthiwar@gmail.com
e-mail: rohan96virutkar@gmail.com, e-mail: rupeshingale579@gmail.com ,
e-mail: shreyapohekar3@gmail.com
e-mail: chikatesiddhant2@gmail.com

---

**Abstract**—Printed circuit board (PCB) inspection requires reliable detection of manufacturing defects that can occupy only a small fraction of the board image. Directly resizing a high-resolution PCB image to a fixed detector input size reduces the pixel information available for such defects. This paper presents a software-based inspection framework that combines YOLO11n with adaptive image tiling. The number of 640 × 640 tiles is computed from the dimensions of each input image, with 20% overlap between neighboring tiles to reduce boundary misses. Each tile is processed by a fine-tuned YOLO11n detector, tile-level boxes are mapped back to the coordinate system of the original image, and a global class-aware Non-Maximum Suppression (NMS) stage removes duplicates created by the overlap. The framework was evaluated on a six-class PCB defect dataset (missing hole, mouse bite, open circuit, short, spur, spurious copper). A baseline model reached 96.2% precision, 93.8% recall, 93.5% mAP@50 and 69.9% mAP@50–95; after fine-tuning, the model reached 98.6% precision, 85.6% recall, 84.6% mAP@50 and 48.1% mAP@50–95 on a 75-image test set. The results show that adaptive tiling is a practical way to preserve local detail in PCB inspection, while recall and bounding-box localization remain the main points for improvement.

**Index Terms**—Printed circuit board, defect detection, YOLO11n, adaptive tiling, small object detection, non-maximum suppression, computer vision.

## I. INTRODUCTION

Printed circuit boards (PCBs) are the electrical foundation of most electronic products, and their quality directly affects the reliability of the final system. During manufacturing, defects such as open circuits, short circuits, missing holes, mouse bites, spurs and spurious copper can appear, and even a small defect may impair electrical connectivity or mechanical integrity.

Manual inspection becomes tiring and inconsistent as board volume grows and defects become small. Automated optical inspection (AOI) based on machine vision is therefore widely used, and reference-comparison methods built on image registration, binarization and morphology have long been applied to this task [1]. More recently, deep-learning object detectors have been explored because they locate and classify defects in a single pass. The YOLO family [2] is attractive for this purpose, and YOLO11n is a lightweight variant intended to balance computational cost and accuracy [3].

A practical difficulty arises when the complete PCB is captured at high resolution. A detector with a 640 × 640 input must downscale a large image, and a defect that spans only a few pixels in the original may become almost invisible after resizing. In the dataset used here, defect boxes are typically only about 2–3% of the image width and height (Fig. 3), which illustrates how severe this effect can be.

This work approaches the problem from the input-processing side. Instead of modifying the detector architecture, the high-resolution image is divided into overlapping 640 × 640 regions whose number follows from the image size. The central idea is to *preserve local image detail first, and then run detection on smaller regions rather than aggressively resizing the whole board.*

Image tiling for small-object detection is an established idea [4], [5]. The contribution of this paper is a complete, modular software pipeline built around YOLO11n for PCB inspection, consisting of: (i) a tile layout computed adaptively from each image's dimensions, with overlap and border handling; (ii) reconstruction of tile-level detections into global PCB coordinates; and (iii) a global class-aware NMS stage that produces the final result in the original image space. The paper also reports the baseline and fine-tuned detector performance and analyzes where the detector succeeds and fails.

## II. PROBLEM DEFINITION

Let the input PCB image have size W × H. A conventional pipeline resizes it to 640 × 640 before detection. For a high-resolution board this strongly reduces the relative size of small defects. The proposed system instead decomposes the image into a set of tiles

$$I \rightarrow \{T_1, T_2, \ldots, T_n\} \quad (1)$$

where each $T_i$ is a 640 × 640 region and n is not fixed but depends on W and H. This dependence on the input size gives the framework its adaptive behavior.

## III. PROPOSED METHODOLOGY

The software pipeline (Fig. 1) has six stages: (1) *image input* by upload or camera; (2) *adaptive tiling*, in which the image dimensions are analyzed and tile positions are computed; (3) *YOLO11n detection* on every tile; (4) *coordinate reconstruction*, converting tile-level boxes to coordinates in the full image; (5) *global class-aware NMS*, removing duplicate predictions from overlapping tiles; and (6) *result generation*, in which final detections are drawn on the original image and stored in structured JSON/HTML reports.

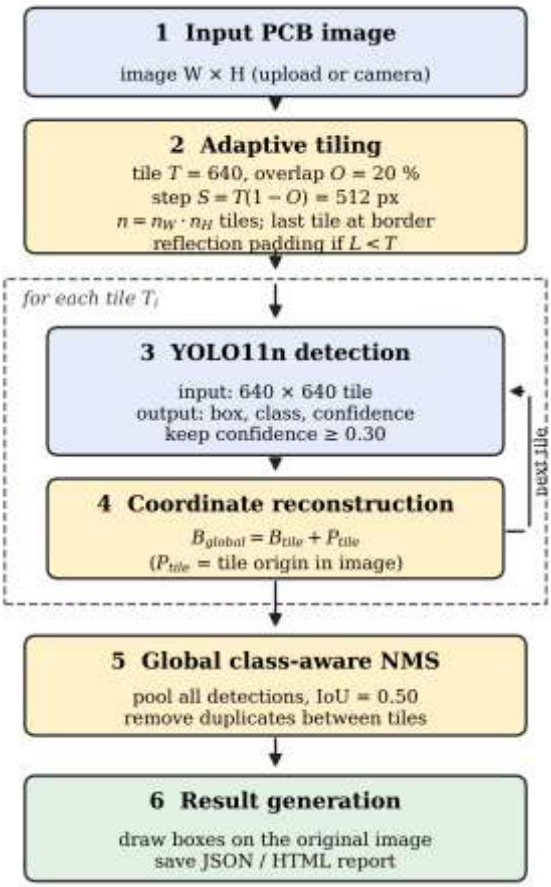



**Fig. 1.** Flow chart of the proposed software pipeline. Blue: detector stages; yellow: adaptive-input and post-processing components added around the detector; green: output.

## A. Adaptive Tiling Strategy

The tile size is T = 640 and the overlap ratio is O = 0.20, so the step between adjacent tiles is

$$S = T(1 - O) = 640(1 - 0.20) = 512\ px \qquad (2)$$

For an image dimension L ∈ {W, H} larger than T, the number of tiles along that axis and their start positions are

$$n_L = \lceil (L - T) / S \rceil + 1 \qquad (3)$$

$$p_k = min(kS, L - T), \quad k = 0, \ldots, n_L - 1 \qquad (4)$$

and the total number of tiles is n = $n_W$ $n_H$. The min(·) term shifts the last tile back to the image border so that the right and bottom edges are fully covered. If L ≤ T, a single tile position is used along that axis.

For example, a 1536 × 1024 image yields 3 × 2 = 6 tiles and a 1900 × 1200 image yields 4 × 3 = 12 tiles (Fig. 2). Because the grid is derived from the image size, computation grows with the board image rather than being tied to a fixed tile count.

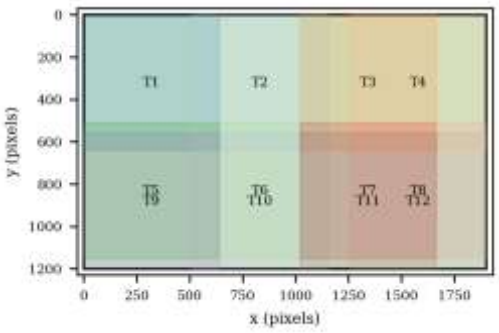



**Fig. 2.** Adaptive tile grid for a 1900 × 1200 image (T = 640, O = 20%, S = 512), giving 12 tiles. The last column and row are shifted to the border.

**Algorithm 1.** Adaptive tiling inference

**Input:** image I (W × H), T = 640, O = 0.20, $C_{min}$ = 0.30, $\tau_{IoU}$ = 0.50
S ← T(1 − O); compute position sets $P_x$, $P_y$
D ← ∅
**for each** ($x_t$, $y_t$) ∈ $P_x$ × $P_y$ **do**
  $T_i$ ← crop of I at ($x_t$, $y_t$) (reflection-pad if needed)
  {(b, c, s)} ← YOLO11n($T_i$); keep s ≥ $C_{min}$
  add (b + [$x_t$, $y_t$, $x_t$, $y_t$], c, s) to D
**end for**
D ← class-aware NMS(D, $\tau_{IoU}$)
**return** D

## B. Edge Handling

When a tile would extend beyond the image, which occurs when an image dimension is smaller than 640 pixels, the tile is completed to 640 × 640 by reflection padding rather than by resizing. This keeps the detector input size constant without introducing artificial black borders. Only detections that lie inside the original image area are retained.

## C. YOLO11n Detection

YOLO11n is used as the detector. For each accepted prediction it provides a bounding box, a class label and a confidence score [3]. A confidence threshold of $C_{min}$ = 0.30 is applied at inference. This threshold is a filtering parameter and should not be interpreted as model accuracy.

## D. Coordinate Reconstruction

Let a tile start at ($x_t$, $y_t$) in the original image and let a detection in tile coordinates be ($x_1$, $y_1$, $x_2$, $y_2$). The global box is

$$B_{global} = B_{tile} + P_{tile}, \quad P_{tile} = (x_t, y_t, x_t, y_t) \qquad (5)$$

that is, $X_1 = x_t + x_1$, $Y_1 = y_t + y_1$, $X_2 = x_t + x_2$, $Y_2 = y_t + y_2$. This step is required so that results are displayed on the original board image rather than on individual tiles.

## E. Global Class-Aware NMS

Overlapping tiles intentionally share image regions, so one physical defect can be detected more than once (for example, a mouse bite reported in two tiles with confidences 0.91 and 0.84). All reconstructed detections are pooled and filtered by class-aware NMS, which uses the intersection over union

$$IoU = Area(B_p \cap B_g) / Area(B_p \cup B_g) \qquad (6)$$

to identify overlapping boxes [6]. Within each class, the higher-confidence box is retained and boxes above the IoU threshold of 0.50 are suppressed. Performing NMS globally, rather than per tile, is necessary because duplicates arise *between* tiles.

## IV. DATASET AND EXPERIMENTAL SETUP

### A. Dataset

The PCB defect dataset of Huang and Wei [1] was used. It contains 1386 images of bare PCBs with six synthetically produced defect types: missing hole, mouse bite, open circuit, short, spur and spurious copper, and each image contains three to five defects of one category. Annotations are distributed in Pascal VOC XML format. They were parsed, mapped to class indices and converted to normalized YOLO coordinates before the train/validation/test split. Fig. 3 shows that the training instances are balanced across classes (386–402 per class) and that the boxes are very small relative to the image, with most widths and heights between roughly 0.01 and 0.04 of the image dimension. The missing-hole class appears as “pin-hole” in the figures produced by the training tool. Fig. 4 shows an annotated training batch.

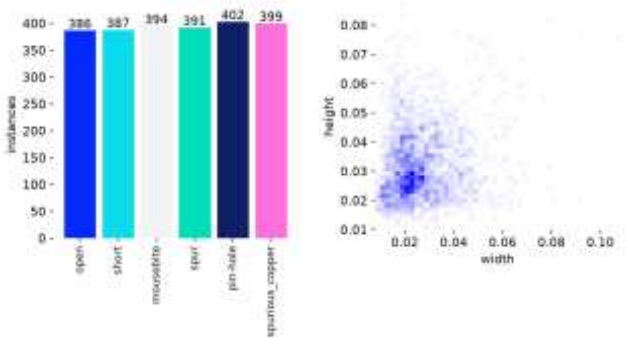



**Fig. 3.** Training label statistics: instances per class (left) and normalized box width versus height (right). Most defects are only a few percent of the image size.

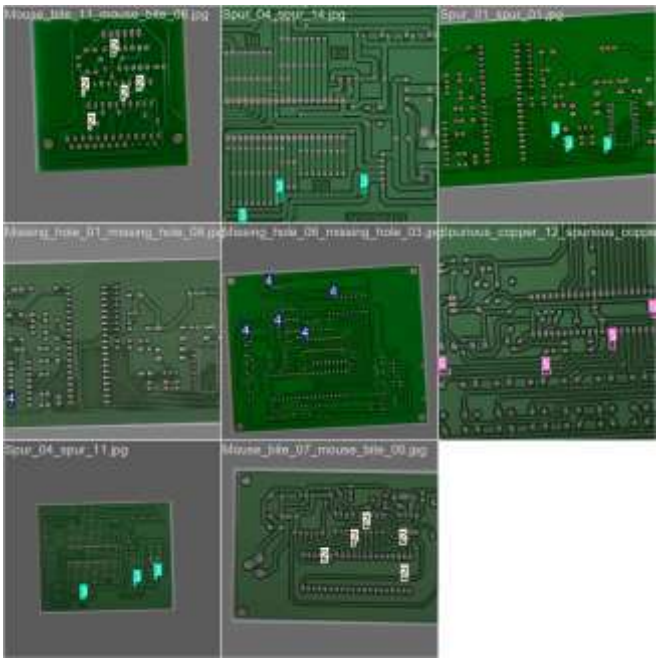


**Fig. 4.** Annotated training batch. Class indices: 0 open, 1 short, 2 mouse bite, 3 spur, 4 missing hole, 5 spurious copper.

### B. Implementation Details

Training and inference used Python 3.12, PyTorch, Ultralytics, OpenCV and NumPy on an NVIDIA RTX 3050 Laptop GPU. The settings are summarized in Table I. Training ran for about 145 epochs, and Fig. 5 shows that the training and validation losses decrease steadily and the validation metrics level off near the end of training.

**TABLE I** — EXPERIMENTAL SETTINGS

| Parameter | Value |
|---|---|
| Detector | YOLO11n |
| Tile / input size | 640 × 640 px |
| Tile overlap O | 20% (step S = 512 px) |
| Confidence threshold $C_{min}$ | 0.30 |
| NMS IoU threshold | 0.50 (class-aware) |
| Number of classes | 6 |
| Test set | 75 images |
| GPU | NVIDIA RTX 3050 Laptop |

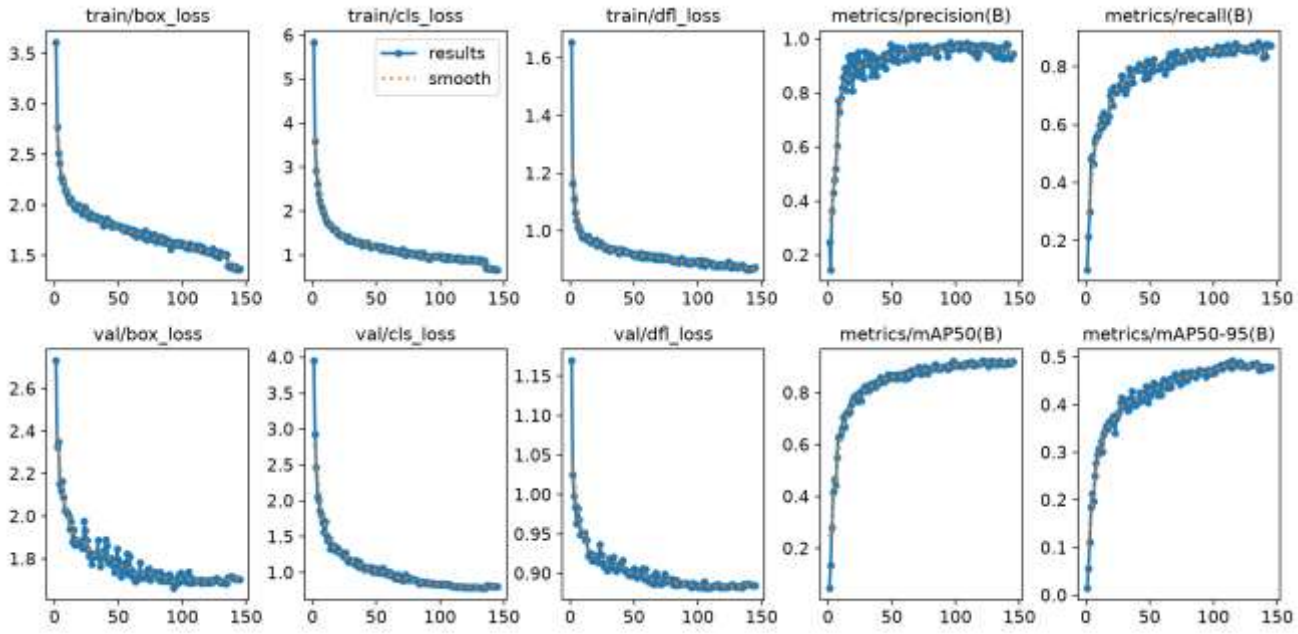



**Fig. 5.** Training and validation curves: box, classification and DFL losses, and precision, recall, mAP@50 and mAP@50–95 over epochs.

### C. Evaluation Metrics

Precision, recall, mAP@50 and mAP@50–95 are reported. Precision is the fraction of predicted defects that are correct, recall is the fraction of ground-truth defects that are found, mAP@50 averages precision over recall at an IoU threshold of 0.5, and mAP@50–95 averages it over IoU thresholds from 0.5 to 0.95 in steps of 0.05, which penalizes imprecise localization.

## V. RESULTS AND DISCUSSION

### A. Overall Performance

Table II lists the baseline and fine-tuned results. The fine-tuned model was evaluated on a 75-image test set.

**TABLE II** — DETECTION PERFORMANCE

| Model | P (%) | R (%) | mAP@50 | mAP@50–95 |
|---|---|---|---|---|
| Baseline | 96.2 | 93.8 | 93.5 | 69.9 |
| Fine-tuned (75 test images) | 98.6 | 85.6 | 84.6 | 48.1 |

The fine-tuned model is highly precise: when it reports a defect, the report is usually correct. Its recall is lower, so some real defects are missed. For PCB inspection this matters, because a missed manufacturing defect is generally more costly than an extra candidate that a human can verify. Future optimization should therefore aim at higher recall while keeping false detections under control.

The gap between mAP@50 (84.6%) and mAP@50–95 (48.1%) indicates that many defects are found but their boxes do not align tightly with the ground truth at stricter IoU thresholds. For very small defects, a displacement of only a few pixels changes the IoU considerably, so localization accuracy is as important as classification confidence.

### B. Per-Class Behavior

Fig. 6 shows the confusion matrices on the evaluation set, which together account for 275 ground-truth defects. Short circuits (1.00) and missing holes (0.96) are detected most reliably, followed by open circuits (0.87) and spurs (0.84). Spurious copper (0.77) and mouse bite (0.70) are the weakest. Almost all errors are of one kind: a real defect predicted as background. Missed defects number 10 for mouse bite, 10 for spurious copper, 7 for spur, 6 for open circuit and 2 for missing hole, whereas confusion between defect classes is rare (two mouse bites predicted as open circuit and one as spur) and only six background regions were falsely reported as defects. This agrees with the high precision and lower recall in Table II. Mouse bites and spurious copper are small, low-contrast changes of the copper outline, which probably explains why they are the hardest to detect.

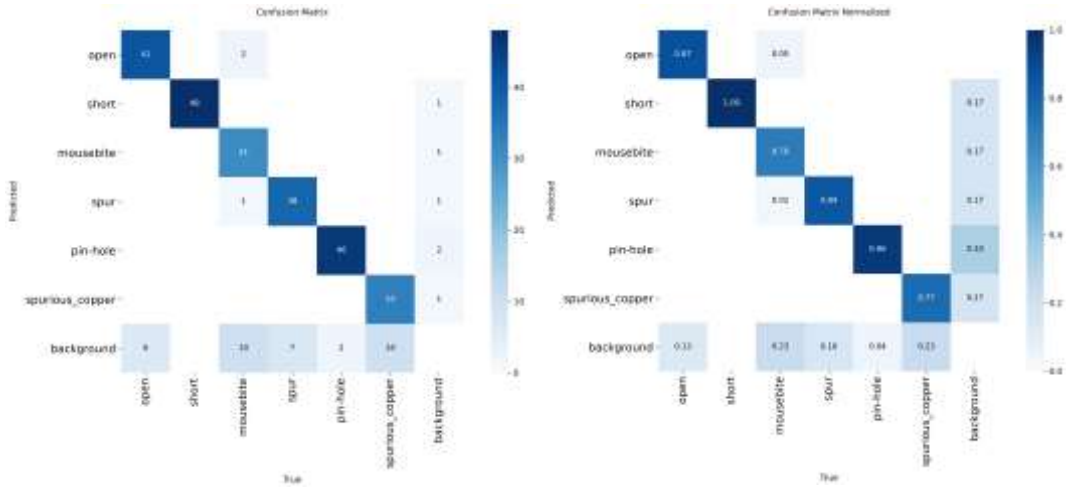



**Fig. 6.** Confusion matrix (left) and normalized confusion matrix (right). Rows are predicted classes and columns are true classes.

## C. Confidence Behavior

The F1–confidence curve (Fig. 7, left) peaks at an F1 of 0.90 for all classes at a confidence of about 0.36 and stays near this level over a wide range of thresholds (roughly 0.15 to 0.65), so the operating point is not sensitive to the exact threshold. The chosen $C_{min}$ = 0.30 lies within this plateau. The precision–confidence curve (right) reaches 1.00 at a confidence of 0.885, which shows that the most confident predictions are reliable. Because recall is the limiting factor, a lower threshold or class-specific thresholds for mouse bite and spurious copper are promising directions, provided the extra false positives remain acceptable.

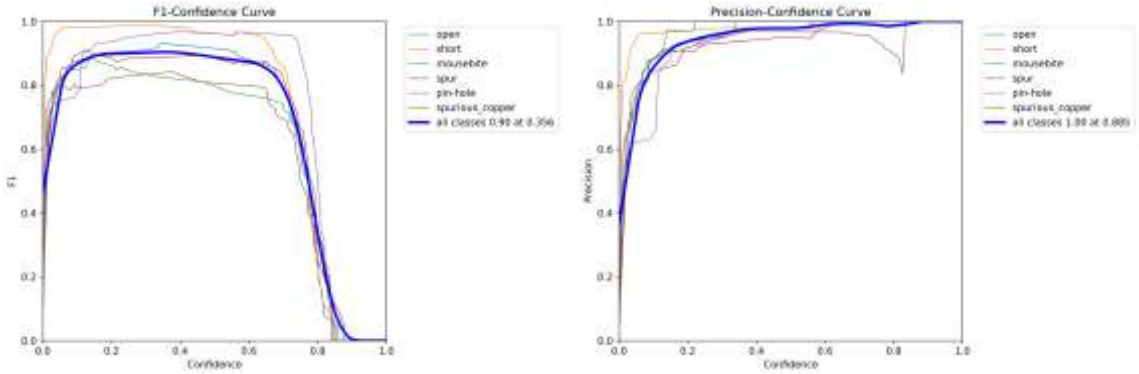



**Fig. 7.** F1–confidence curve (left) and precision–confidence curve (right) for all classes.

## D. Qualitative Results

Fig. 8 shows predictions on validation batches. The detector localizes and labels all six defect types, with confidences mostly between 0.6 and 0.9. Some boxes overlap, and a few low-confidence detections occur on mouse bite and spurious copper, consistent with the per-class results above.



**Fig. 8.** Predicted defects with class names and confidence scores on two validation batches.

### E. Effect of Adaptive Tiling, Overlap and Cost

The purpose of adaptive tiling is to keep the local spatial information that would be lost when a complete high-resolution board is reduced to a fixed detector resolution. This matters for mouse bites, spurs and small copper abnormalities, which occupy a very small part of the board. A fixed tile count would process every board with the same pattern regardless of its size, whereas the adaptive layout generates only as many tiles as the image requires.

The 20% overlap reduces the chance that a defect on a tile border is split between two tiles with insufficient context in either. The trade-off is that higher overlap means more tiles to process and more duplicate predictions, which is why the global NMS stage is an essential part of the method and not an optional post-process. The main cost of tiling is computation: larger images produce more tiles and each requires an inference call. This is acceptable in inspection settings where finding a small defect matters more than minimizing the number of model calls. Since the tiling and coordinate-reconstruction layers are independent of the detector, YOLO11n could be replaced by another detector without changing them.

### F. Limitations

The experiments support the performance of the overall pipeline but do not isolate the contribution of tiling: no ablation comparing the same model with direct 640 × 640 resizing, fixed tiling and adaptive tiling has been run yet. Table III shows the planned comparison. The recall of the fine-tuned model is also limited, and the possible causes include very small defect size, similar appearance of some classes, limited dataset diversity, bounding-box localization difficulty and differences between the synthetic dataset and real production boards. Finally, the dataset contains synthetically generated defects on a small number of template boards (ten in the original dataset [1]), so generalization to industrial data has not been demonstrated.

**TABLE III** — PLANNED ABLATION STUDY (TO BE COMPLETED)

| Configuration | P | R | mAP@50 | mAP@50–95 | Time |
|---|---|---|---|---|---|
| Direct 640 × 640 resize | – | – | – | – | – |
| Fixed tiling | – | – | – | – | – |
| Adaptive, O = 20% | 98.6 | 85.6 | 84.6 | 48.1 | – |

## VI. CONCLUSION

This paper presented a software-based framework for high-resolution PCB defect inspection that combines YOLO11n with adaptive overlapping tiling, coordinate reconstruction and global class-aware NMS. The method addresses the loss of small-defect information that occurs when a high-resolution board is directly reduced to the detector input size. The baseline model achieved 96.2% precision, 93.8% recall, 93.5% mAP@50 and 69.9% mAP@50–95, and the fine-tuned model achieved 98.6% precision, 85.6% recall, 84.6% mAP@50 and 48.1% mAP@50–95 on a 75-image test set. High precision does not imply complete coverage: the lower recall, concentrated in mouse-bite and spurious-copper defects, and the gap between mAP@50 and mAP@50–95 show that missed detections and box localization are the main remaining problems.

## FUTURE WORK

Future work includes (i) the ablation study of Table III comparing direct resizing, fixed tiling and adaptive tiling; (ii) overlap values of 10–30% and tile sizes from 512 to 1024; (iii) recall optimization through threshold tuning, class-specific thresholds, additional augmentation and hard-example mining; (iv) improved localization for small defects; (v) computational optimization by batch inference, FP16, ONNX/TensorRT deployment and selective tile processing; and (vi) evaluation on real, larger and more diverse industrial PCB datasets.

## REFERENCES

[1] W. Huang and P. Wei, “A PCB dataset for defects detection and classification,” *arXiv preprint arXiv:1901.08204*, 2019.

[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[3] G. Jocher and J. Qiu, “Ultralytics YOLO11,” 2024. [Online]. Available: https://docs.ultralytics.com/models/yolo11

[4] F. O. Ünel, B. O. Özkalaycı, and C. Çiğla, “The power of tiling for small object detection,” in *Proc. IEEE/CVF CVPR Workshops*, 2019.

[5] F. C. Akyon, S. O. Altinuc, and A. Temizel, “Slicing aided hyper inference and fine-tuning for small object detection,” in *Proc. IEEE Int. Conf. Image Processing (ICIP)*, 2022, pp. 966–970.

[6] A. Neubeck and L. Van Gool, “Efficient non-maximum suppression,” in *Proc. Int. Conf. Pattern Recognition (ICPR)*, 2006, pp. 850–855.