Object Detection on a Small Custom Dataset

**PCB defect detection via fine-tuning of pre-trained object detection models*

1st Federico De Marinis
Department of Computer Science
IT University of Copenhagen
Copenhagen, Denmark
fedd@itu.dk

2nd Mattia Malipiero
Department of Computer Science
IT University of Copenhagen
Copenhagen, Denmark
mattm@itu.dk

I. Introduction

Printed Circuit Boards (PCBs) sit at the core of everyday electronics. As device complexity increases, production requires higher precision and often longer development cycles, making defect detection a critical part of mass manufacturing. Consequently, substantial investments have been made over the years in defect detection.

A. Error detection in PCB production

In this domain the main challenge lies in the nature of the anomalies themselves. Because defects are typically minute and visually similar, the detection system must be sensitive to subtle morphological features. Although classical pipelines (e.g., rule-based inspection and classical machine vision) can be effective in constrained settings, their reliance on handcrafted features has driven a shift toward deep learning, which learns features directly from data.

In recent years, convolutional neural networks for object detection and segmentation have become a popular choice, as they can learn discriminative features directly from data, thus reducing the need for manual inspection by humans or machines. However, deep learning based models are not without limitations; in particular, learning reliable representations for very small defects remains challenging as noted by Lim et al. feature representations of tiny PCB defects can be lost after successive convolutional layers, leading to a substantial drop in accuracy [1].

To address this issue without the need for massive, domain-specific datasets, the implementation of transfer learning has become a standard methodology. By utilizing models pre-trained on extensive benchmarks researchers can inherit generalized visual hierarchies. These foundational features are then adapted to the PCB domain through fine-tuning, a process where either a subset or the entirety of the network's weights are updated. There is a large body of research already available which focuses on taking pre-trained models and applying them to PCB defects datasets.

A literature review [2] of the PCB detection reveals domain a clear separation between two primary algorithmic categories:

Two-stage algorithms (e.g., Faster R-CNN) utilize a region proposal phase followed by classification. While these methods demonstrate high detection accuracy, they are often characterized by substantial computational workloads and slower inference speeds that may not meet the real-time requirements of industrial production.

One-stage algorithms (e.g., YOLO, SSD) redefine defect detection as a regression problem, concurrently predicting class and bounding box attributes. These models are valued for their exceptional speed and real-time processing capabilities, with some implementations reaching up to 90 FPS.

Among modern single-stage detectors, the YOLO family offers a strong accuracy-speed trade-off, which is attractive for industrial inspection settings without the need for a prohibitive training schedule in terms of computing power required.

Motivated by these challenges and the advancements in real-time architectures, PCB inspection is framed as an object-detection problem, where the goal is to both localize defects and assign them a class label. In this work, the focus is placed on transfer learning with the YOLOv8 family of models.

II. Methodology

The primary detection model adopted in this study is YOLOv8m, a medium-scale variant of the YOLOv8 family developed by Ultralytics [3]. YOLOv8 is a single-stage, anchor-free object detector, meaning that bounding box regression and classification are performed jointly in a single forward pass of the network. This design reduces computational overhead while maintaining strong detection performance, making it suitable for industrial inspection scenarios [4] [5].

YOLOv8 follows a backbone–neck–head architecture. The backbone extracts hierarchical feature representations from the input image, progressively transforming low-level spatial patterns into higher-level semantic features. The neck aggregates these representations across multiple spatial resolutions through a feature pyramid structure, enabling multi-scale detection. This multi-scale fusion is relevant for PCB defect detection, where defects can vary in size and may occupy only a small fraction of the image area. The detection head operates on the fused feature maps and outputs class probabilities together with bounding box coordinates for each detected instance.

The separation into backbone, neck, and head is central to this work, as different fine-tuning strategies selectively update specific components of the network. In particular, freezing the backbone preserves generic visual representations learned

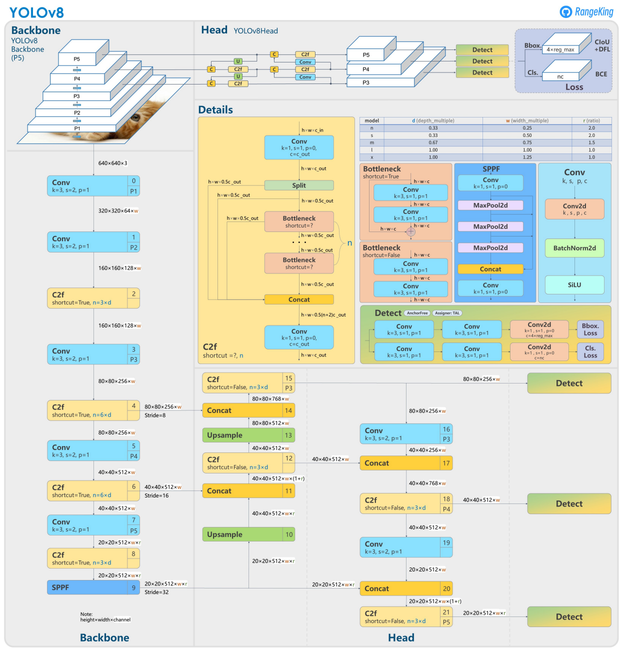


Fig. 1: Model structure of YOLOv8 detection models; although not specified in the image, the neck is composed by layers 10 through 21. Source: [6].

during pretraining, while adapting the neck and head allows the model to specialize to the PCB defect domain.

Among the available YOLOv8 variants, YOLOv8m provides a balance between representational capacity and computational efficiency. Compared to smaller variants, it offers increased feature extraction capability, which is beneficial for capturing fine-grained structural irregularities present in PCB surface defects, while remaining tractable for repeated training and controlled experimentation.

A. *Overall Training Objective*

The training of YOLOv8m is formulated as the minimization of a composite objective that jointly optimizes classification accuracy and bounding box localization precision. The total loss is defined as:

$$\mathcal{L}_{total} = \lambda_{cls}\mathcal{L}_{cls} + \lambda_{box}\mathcal{L}_{CIoU} + \lambda_{dfl}\mathcal{L}_{DFL},$$

where $\mathcal{L}_{cls}$ denotes the classification loss, $\mathcal{L}_{CIoU}$ represents the bounding box regression loss based on Complete IoU, and $\mathcal{L}_{DFL}$ is the Distribution Focal Loss term used to refine coordinate estimation. The weighting coefficients λ_{cls}, λ_{box}, and λ_{dfl} control the relative contribution of each component during optimization.

B. *Classification Loss*

YOLOv8 employs a focal-style classification loss that mitigates the imbalance between positive and negative samples. In object detection tasks, the majority of spatial locations correspond to background. A standard cross-entropy formulation may therefore be dominated by easy negative examples.

Focal Loss addresses this issue by down-weighting well-classified examples and emphasizing harder cases [7]. Its general formulation is:

$$\mathcal{L}_{cls} = -\alpha(1-p)^{\gamma}\log(p),$$

where p denotes the predicted probability for the ground-truth class, α is a balancing factor, and γ is a focusing parameter that reduces the contribution of easy samples.

This formulation allows the detector to focus on challenging and potentially rare defect categories rather than being dominated by abundant background predictions.

C. *Bounding Box Regression Loss*

Bounding box regression in YOLOv8 combines geometric overlap optimization with coordinate refinement. Two complementary components are used: Complete IoU (CIoU) and Distribution Focal Loss (DFL).

1) Complete IoU (CIoU): Intersection over Union (IoU) measures the overlap between predicted and ground-truth bounding boxes:

$$IoU = \frac{|B_{pred} \cap B_{gt}|}{|B_{pred} \cup B_{gt}|}.$$

However, IoU alone provides no gradient when boxes do not overlap and does not account for geometric alignment. Complete IoU (CIoU) extends this formulation by incorporating three factors: overlap area, center-point distance, and aspect ratio consistency [8]. The CIoU loss is defined as:

$$\mathcal{L}_{CIoU} = 1 - IoU + \frac{\rho^2(b, b^{gt})}{c^2} + \alpha v,$$

where $\rho(b, b^{gt})$ denotes the Euclidean distance between box centers, c is the diagonal length of the smallest enclosing box, and v captures aspect ratio similarity.

By considering both spatial alignment and shape consistency, CIoU provides stronger and more stable optimization signals than IoU-based losses alone.

2) Distribution Focal Loss (DFL): Rather than regressing bounding box coordinates as single continuous values, YOLOv8 models each coordinate as a discrete probability distribution over predefined bins. Distribution Focal Loss encourages the predicted probability mass to concentrate around the true coordinate value, leading to smoother gradients and sub-pixel precision. This formulation improves localization accuracy, particularly for small objects where minor coordinate deviations can significantly affect IoU.

D. *Relevance to PCB Defect Detection*

The combination of focal-style classification loss, CIoU, and DFL is relevant for PCB surface defect detection.

First, PCB inspection images typically contain large background regions with comparatively few defective areas. The focal-style classification loss mitigates the dominance of easy background predictions and encourages learning from harder or less frequent defect instances.

Second, most of PCB defects occupy very small spatial regions. Accurate geometric alignment is therefore critical. CIoU enhances convergence toward well-aligned bounding boxes, while DFL improves fine-grained coordinate precision.

Together, these components enable robust detection of small and potentially rare defects, which is essential in industrial quality-control settings where missed detections may lead to functional failure.

In addition to the standard YOLOv8m configuration, a YOLOv8m-P2 variant was evaluated. The P2 configuration introduces an additional high-resolution detection layer operating at an earlier stage of the feature pyramid. While the standard YOLOv8 architecture performs detection at three scales, the P2 variant extends the pyramid with a finer-resolution level, enabling predictions to be generated from feature maps of higher spatial granularity.

The motivation for incorporating a higher-resolution detection head is grounded in prior work on multi-scale detection architectures. Feature Pyramid Networks (FPN) demonstrate that leveraging higher-resolution pyramid levels improves sensitivity to small objects by preserving spatially localized information that may be attenuated in deeper layers [9], while Kim et al. [10] explicitly incorporated a P2 layer into YOLOv8 to enhance small-object detection performance, showing that augmenting the feature pyramid with a finer-resolution level can improve detection of small-scale targets.

Accordingly, the YOLOv8m-P2 variant was included in the experimental setup for comparative analysis with respect to the baseline model, to evaluate whether enhanced high-resolution feature utilization improves detection performance on our downstream task.

III. Dataset description

The experiments are conducted on the DsPCBSD+ dataset (Dataset of PCB Surface Defect), introduced by Lv et al. (2024) [11]. The dataset contains 10,259 images of real PCB surface defects collected from an industrial Automated Optical Inspection (AOI) system deployed at Guangzhou FastPrint Technology Co., Ltd. All images originate from actual production workflows and are captured using high-resolution line-scan cameras under controlled multi-LED illumination, ensuring realistic industrial conditions.

The dataset comprises 20,276 manually annotated defect instances across nine defect categories: Short (SH), Spur (SP), Spurious Copper (SC), Open (OP), Mouse Bite (MB), Hole Breakout (HB), Conductor Scratch (CS), Conductor Foreign Object (CFO), and Base Material Foreign Object (BMFO), see in Appendix Fig. 4. Each image has a fixed resolution of 226×226 pixels.

Annotations are provided in multiple formats, including YOLO-normalized bounding boxes. Multiple defects may appear within a single image, resulting in a multi-object detection setting with potential bounding-box overlap.

To characterize defect scale, the COCO [12] size convention is followed and instances are grouped into three bins based on bounding-box area. This yields the following distribution: 13,574 small (below 32×32 pixels), 5,797 medium (between 32×32 and 96×96 pixels), and 905 large (above 96×96 pixels). The class-wise size distribution in Fig. 6 shows a strongly imbalanced size distribution across classes. Most classes are dominated by small defects, while HB is mostly medium and CS has a substantial large component. This indicates strong class-specific scale patterns: classes such as SP/MB/OP/SC are primarily micro-defect classes, whereas HB and CS contain comparatively larger structures.

The distribution of defect instances across the nine categories is illustrated in Fig. 5. The dataset exhibits a clear class imbalance, with certain defect types occurring substantially more frequently than others. In particular, categories such as Spur (SP) dominate the dataset, while defects such as Short (SH) and Open (OP) appear considerably less often.

Despite their lower frequency, SH and OP defects are functionally critical, as they directly affect electrical continuity and may lead to circuit failure. This imbalance between frequent non-critical defects and rarer but high-impact defects introduces a non-uniform learning signal during training.

Such characteristics motivate the use of a focal-style classification loss, which down-weights easy and dominant classes while increasing the relative contribution of harder and less frequent categories. This design choice mitigates bias toward majority classes and supports improved sensitivity to rare but industrially critical defects.

The distribution of normalized bounding-box widths and heights is shown in Fig. 7(a)–(b). The mean normalized width and height are approximately 0.15 and 0.14, respectively, indicating that most defects occupy a small fraction of the image area. The majority of bounding boxes fall below a normalized size of 0.2, further confirming that the dataset predominantly represents a small-object detection setting.

This concentration of small defects introduces additional localization challenges, as minor coordinate deviations can significantly affect overlap-based metrics. Consequently, multi-scale feature aggregation becomes essential. The dominance of small objects directly motivates the use of enhanced multi-scale detection strategies, including the YOLOv8m-P2 variant.

Defect centers are not uniformly distributed across the image plane and concentrated near the patch center (Fig. 9). This pattern likely reflects the dataset construction process, in which defects are often centered when patches are created, suggesting that the dataset encodes positional priors that may be exploited during learning. Consequently, models trained on datasets composed of patches that are cropped and centered on defects, such as this one, may struggle to generalize to full-PCB imagery, where defects can appear at arbitrary locations; this bias should be considered when interpreting generalization performance.

In addition, the dataset exhibits non-trivial scene complexity. While the mean number of objects per image is approximately 1.98 (with a long tail up to 27), multi-label images are frequent (Fig. 7 d). Cross-class co-occurrence further confirms that defects often appear jointly within the same local crop (Fig. 8), indicating that the detection task is not purely single-instance. Consequently, training is performed under a multi-object, multi-class setting where overlapping and adjacent defects must be resolved reliably.

IV. Experiments and Results

All models were trained using the Ultralytics YOLOv8 framework and were all initialized from pretrained YOLO weights. Two YOLOv8 architectures (YOLOv8m and YOLOv8m-P2) were evaluated and fine-tuning strategies were compared; the preprocessing and augmentation recipe was kept fixed across runs for a fair comparison.

A. *Model variants fine-tuning strategies*

For YOLOv8m, three fine-tuning strategies were compared by freezing different portions of the network, following the backbone–neck–head decomposition (backbone: layers 0–9, neck: 10–21, head: 22). For YOLOv8m-P2, only a full fine-tuning experiment was conducted. Augmentation parameters values can be see in Fig. 11. Figure 10 summarizes, for each strategy, the split between trainable and frozen parameters, along with the corresponding trainable parameter counts.

B. *Metrics*

All final numbers are computed on the held-out test split with standard YOLOv8 inference settings and no test-time augmentation. Both operating-point error statistics at a fixed matching criterion and COCO-style average precision metrics that summarize performance across confidence thresholds [12] are reported.

1) Matching criterion: A predicted box is considered a correct detection if it is assigned to a ground-truth box of the same class with Intersection-over-Union (IoU) $\geq$ 0.50. Under this rule, true positives (TP) are correctly matched predictions, false positives (FP) are unmatched predictions, and false negatives (FN) are ground-truth objects with no matched prediction. Misclassifications (Miscls) are additionally reported, defined as predictions with IoU $\geq$ 0.50 to a ground-truth box but an incorrect class label.

2) Precision, Recall, F1: From the counts above, precision $P = \frac{TP}{TP+FP}$, recall $R = \frac{TP}{TP+FN}$, and $F1 = \frac{2PR}{P+R}$ are computed. Precision captures the false-alarm rate, recall captures missed-defect behavior, and F1 summarizes their trade-off at the chosen operating point.

3) COCO AP metrics: To evaluate detection quality independent of a single confidence threshold, mean Average Precision at IoU$=$ 0.50 (mAP50) and COCO mAP (mAP50–95) are reported, where AP is averaged over IoU thresholds 0.50:0.05:0.95 and then averaged across classes [12]. While mAP50 reflects detection capability under a lenient overlap requirement, mAP50–95 emphasizes localization accuracy.

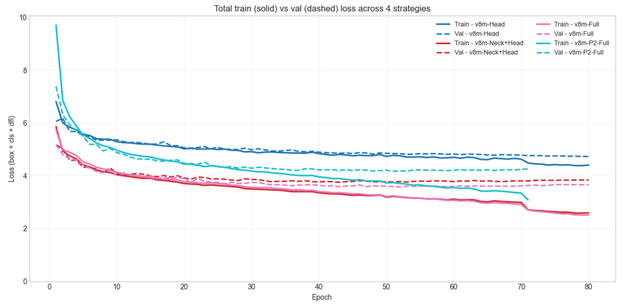


Fig. 2: Loss curves for Training (solid) and Validation (dashed) curves of the four runs. Maximum nr of epochs is 80, patience is set to 20

TABLE I: Global test-set metrics including COCO AP (AP$_S$, AP$_M$, AP$_L$).

Model	P	R	F1	mAP50	mAP50-95	AP$_S$	AP$_M$	AP$_L$
v8m-Head	0.633	0.599	0.615	0.635	0.352	0.214	0.243	0.236
v8m-Neck+Head	0.794	0.746	0.769	0.796	0.497	0.360	0.472	0.614
v8m-Full	0.793	0.824	0.808	0.844	0.545	0.395	0.526	0.585
v8m-P2-Full	0.801	0.792	0.797	0.841	0.531	0.375	0.494	0.383

4) Size-aware evaluation: Because DsPCBSD+ is skewed toward small instances, overall AP is complemented with COCO size-specific scores (AP$_S$, AP$_M$, AP$_L$) using the same COCO size definitions introduced earlier, which quantify how performance changes with object scale [12].

C. *Training and Validation curves analysis*

With early stopping, the v8m-P2-Full terminated before reaching the maximum of 80 training steps. Across all four strategies, optimization is rapid in the initial phase followed by a slower convergence regime. The v8m-P2-Full configuration starts from the highest initial loss, but converges to an intermediate training-loss level between the head-only and fully fine-tuned baselines. In later epochs, validation loss remains consistently above training loss for all models, indicating a stable generalization gap that is not large enough to be concerning. This gap is most pronounced for v8m-Head and v8m-P2-Full, and smaller for v8m-Full and v8m-Neck+Head. Toward the end of training, validation curves tend to plateau (or increase slightly) while training loss continues to decrease, which is consistent with mild overfitting. Overall, the curves suggest underfitting for the head-only setup and mild overfitting for the two YOLOv8m fine-tuning regimes (v8m-Neck+Head and v8m-Full). For the P2-tuned variant, the training curve appears close to diverging further from the validation curve, suggesting the start of a more pronounced overfitting, also P2 reaches its best validation point earlier (triggering early stopping) with a higher final validation loss than YOLOv8m-Full, indicating less stable optimization and an earlier validation ceiling under fixed hyperparameters.

D. *Results Analysis*

Table I reports results for all four runs using the best checkpoint achieved during training. For completeness, Table VI reports the metrics at the final epoch; however, the checkpoints with the strongest generalization performance are emphasized. Since the two fully fine-tuned models achieve the best overall results, the remainder of this analysis centers on their best-generalization weights.

In the standardized setting (identical augmentation and optimization protocol), the comparison between YOLOv8m-Full and YOLOv8m-P2-Full shows that adding the P2 detection scale does not improve aggregate performance on this dataset. On the test set at the default evaluation confidence, as can be seen in Table I, YOLOv8m-Full achieves higher mAP50-95 (0.545 vs 0.531) and slightly higher mAP50 (0.844 vs 0.841), while YOLOv8m-P2 shows only a marginal precision gain (0.801 vs 0.793) at the cost of a clearer recall drop (0.792 vs 0.824). This precision–recall trade-off is consistent with a stricter operating behavior: P2 tends to suppress more uncertain detections, which reduces some false positives but increases missed instances.

The size COCO metrics reinforce this conclusion. Contrary to the expected advantage of a higher-resolution P2 branch for small defects, AP_S is lower for P2 (0.375 vs 0.395). The gap is larger for medium objects (AP_M: 0.493 vs 0.526) and becomes very large for large objects (AP_L: 0.384 vs 0.584). Therefore, the observed performance decrease is not confined to a specific object size; it is global, with the strongest penalty on larger instances. This indicates that, in this training regime, the architectural shift toward an extra fine-scale prediction level did not translate into better localization quality across IoU thresholds.

1) Per Class analysis: To better understand the aggregate results, per-class performance is analysed in Tables III and IV. P2 improves AP50-95 in a minority of classes (e.g., SP, MB, BMFO, HB), but these gains are small and are outweighed by larger losses in other classes (especially SH, then CS and OP). In other words, P2 redistributes performance rather than increasing it consistently. This explains why precision can increase slightly while mAP50-95 decreases overall: the model becomes more selective but less uniformly accurate across classes and IoU strictness.

E. Threshold Sweep and Operating-Point Selection

Because model behavior depends strongly on confidence threshold, a confidence sweep was performed and precision/recall/F1 trade-offs were analyzed. To avoid test leakage, threshold selection was performed on the validation set, then results were reported on the test set at the selected threshold. Confidence-threshold tuning was performed only on the validation set by sweeping thresholds from 0.05 to 0.60 and and the threshold with best F1 was selected. The selected threshold was then applied once on the test set.

As expected, increasing confidence improved precision but reduced recall. For YOLOv8m-P2, moving from the default threshold (0.25) to the validation-selected threshold (0.40) increased precision (0.801 to 0.810) but decreased recall (0.792 to 0.787), with slight decreases in mAP50 and mAP50-95. This indicates that threshold tuning changed the operating point (precision-recall tradeoff) but did not improve overall ranking quality. It is noted that threshold tuning was investigated as a post-training calibration step, not as a substitute for model-level improvements.

TABLE II: YOLOv8m-P2 test metrics at default confidence and validation-selected confidence.

Setting	Conf.	mAP50	mAP50-95	P	R
P2 test @ default conf	0.250	0.841	0.531	0.801	0.792
P2 test @ val-selected conf*	0.400	0.828	0.529	0.810	0.787

Note: Full metrics table in both settings can be found in appendix at V

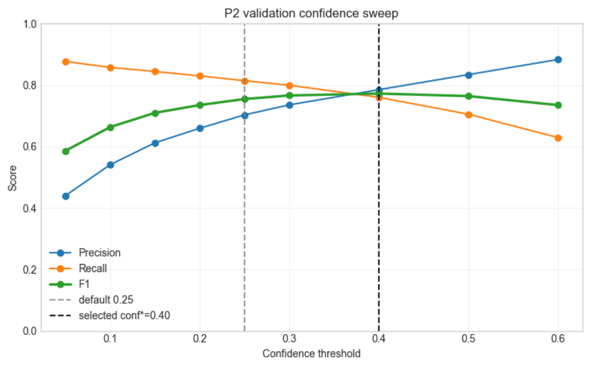


Fig. 3: Validation confidence sweep for YOLOv8m-P2. As the confidence threshold increases, precision rises while recall drops; F1 peaks around 0.40. The gray dashed line marks the default threshold (0.25), and the black dashed line marks the validation-selected operating point ($\text{conf}^* = 0.40$).

V. Conclusion and Limitations

PCB surface-defect detection was formulated as an object-detection task on the DsPCBSD+ industrial AOI dataset, where most defects are small and classes are imbalanced. Transfer learning with pre-trained YOLOv8 models was evaluated under controlled training conditions, with the focus placed on how much of the network should be updated during fine-tuning.

Across YOLOv8m freezing strategies, performance increased with the amount of trainable capacity: head-only fine-tuning underperformed, freezing only the backbone improved substantially, and full fine-tuning achieved the strongest overall detection quality on the held-out test set. Hypothesis: higher performance was expected when more layers were allowed to adapt, since a larger trainable portion of the network can better specialize to PCB-specific defect morphologies compared to updating the head alone.

A YOLOv8m-P2 variant (adding a higher-resolution detection level) was also assessed to test whether improved high-resolution feature utilization benefits small-defect detection; however, under the tested setup, this architectural change did not produce a consistent improvement over the fully fine-tuned YOLOv8m baseline. Overall, the controlled experiments show that threshold choice changes deployment behavior (precision vs. recall), while model architecture and training determine most of the final AP performance. For this PCB dataset, full fine-tuning of YOLOv8m remains a strong baseline, while YOLOv8m-P2 requires additional targeted investigation (e.g. class-specific imbalance handling, loss re-weighting, or broader hyperparameter search) before a consistent advantage can be established.

It is acknowledged that performance may improve with further training and stronger regularization to mitigate the already visible overfitting; however, due to the limited compute budget, a broader hyperparameter search or extended training schedule could not be investigated.

As discussed in the exploratory data analysis, the dataset consists of cropped PCB patches rather than full-board images. This limits the availability of global contextual information and may reduce generalization when defect interpretation depends on surrounding board structure. A relevant future direction is the inclusion of full-board imagery (or a mixed patch + full-image training setup), while keeping the training and inference format aligned with the intended acquisition pipeline. It is hypothesized that increasing data diversity in this way would improve robustness and generalization performance.

VI. Acknowledgment

Generative AI tools (GAI) were used for rephrasing, coding support and asking feedback on our interpretation of the results.

References

[1] J. Lim, J. Lim, V. M. Baskaran, and X. Wang, “A deep context learning based pcb defect detection model with anomalous trend alarming system,” Results in Engineering, vol. 17, p. 100968, 2023.

[2] X. Chen, Y. Wu, X. He, and W. Ming, “A comprehensive review of deep learning-based pcb defect detection,” IEEE Access, vol. 11, pp. 139 017–139 038, 2023.

[3] Ultralytics, “Yolov8 documentation,” Online, 2023. [Online]. Available: https://docs.ultralytics.com/models/yolov8/

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” 2016. [Online]. Available: https://arxiv.org/abs/1506.02640

[5] M. Yaseen, “What is yolov8: An in-depth exploration of the internal features of the next-generation object detector,” 2024. [Online]. Available: https://arxiv.org/abs/2408.15857

[6] RangeKing, “Dl-diagram: Computer vision diagrams (release: cv),” Online, 2024. [Online]. Available: https://github.com/RangeKing/DL-Diagram/releases/tag/cv

[7] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” 2018. [Online]. Available: https://arxiv.org/abs/1708.02002

[8] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, “Distance-IOU loss: Faster and better learning for bounding box regression,” Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 07, pp. 12 993–13 000, 4 2020. [Online]. Available: https://doi.org/10.1609/aaai.v34i07.6999

[9] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature pyramid networks for object detection,” in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 936–944.

[10] J.-H. Kim, N. Kim, and C. S. Won, “High-speed drone detection based on yolo-v8,” in ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2023, pp. 1–2.

[11] S. Lv, B. Ouyang, Z. Deng, T. Liang, S. Jiang, K. Zhang, J. Chen, and Z. Li, “A dataset for deep learning based detection of printed circuit board surface defect,” Scientific Data, vol. 11, no. 1, p. 811, 7 2024. [Online]. Available: https://doi.org/10.1038/s41597-024-03656-8

[12] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: Common objects in context,” in Computer Vision – ECCV 2014, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Springer, 2014, pp. 740–755.

TABLE III: Per-class AP comparison (AP50 and AP50–95) across four fine-tuning strategies (best checkpoint).

Class	v8m-Head AP50	v8m-Neck+Head AP50	v8m-Full AP50	v8m-P2-Full AP50	v8m-Head AP50-95	v8m-Neck+Head AP50-95	v8m-Full AP50-95	v8m-P2-Full AP50-95
BMFO	0.743	0.852	0.899	0.896	0.386	0.470	0.518	0.521
CFO	0.476	0.629	0.706	0.688	0.293	0.408	0.464	0.453
CS	0.299	0.640	0.707	0.693	0.159	0.398	0.452	0.421
HB	0.924	0.955	0.968	0.976	0.757	0.820	0.836	0.839
MB	0.631	0.752	0.810	0.814	0.297	0.377	0.436	0.441
OP	0.627	0.882	0.901	0.893	0.295	0.544	0.596	0.568
SC	0.732	0.820	0.846	0.859	0.394	0.502	0.528	0.521
SH	0.656	0.856	0.916	0.911	0.317	0.583	0.658	0.591
SP	0.626	0.779	0.841	0.844	0.267	0.374	0.416	0.421

TABLE IV: Per-class precision and recall across four fine-tuning strategies (best checkpoint).

Class	v8m-Head Precision	v8m-Neck+Head Precision	v8m-Full Precision	v8m-P2-Full Precision	v8m-Head Recall	v8m-Neck+Head Recall	v8m-Full Recall	v8m-P2-Full Recall
BMFO	0.721	0.819	0.818	0.811	0.681	0.828	0.909	0.866
CFO	0.516	0.629	0.653	0.669	0.427	0.586	0.703	0.634
CS	0.410	0.655	0.688	0.701	0.251	0.553	0.680	0.600
HB	0.720	0.920	0.907	0.890	0.935	0.926	0.945	0.960
MB	0.664	0.809	0.803	0.798	0.552	0.688	0.793	0.769
OP	0.651	0.851	0.824	0.825	0.597	0.837	0.897	0.862
SC	0.732	0.803	0.818	0.841	0.714	0.760	0.802	0.793
SH	0.636	0.847	0.823	0.814	0.655	0.827	0.885	0.880
SP	0.647	0.817	0.803	0.863	0.577	0.703	0.807	0.767

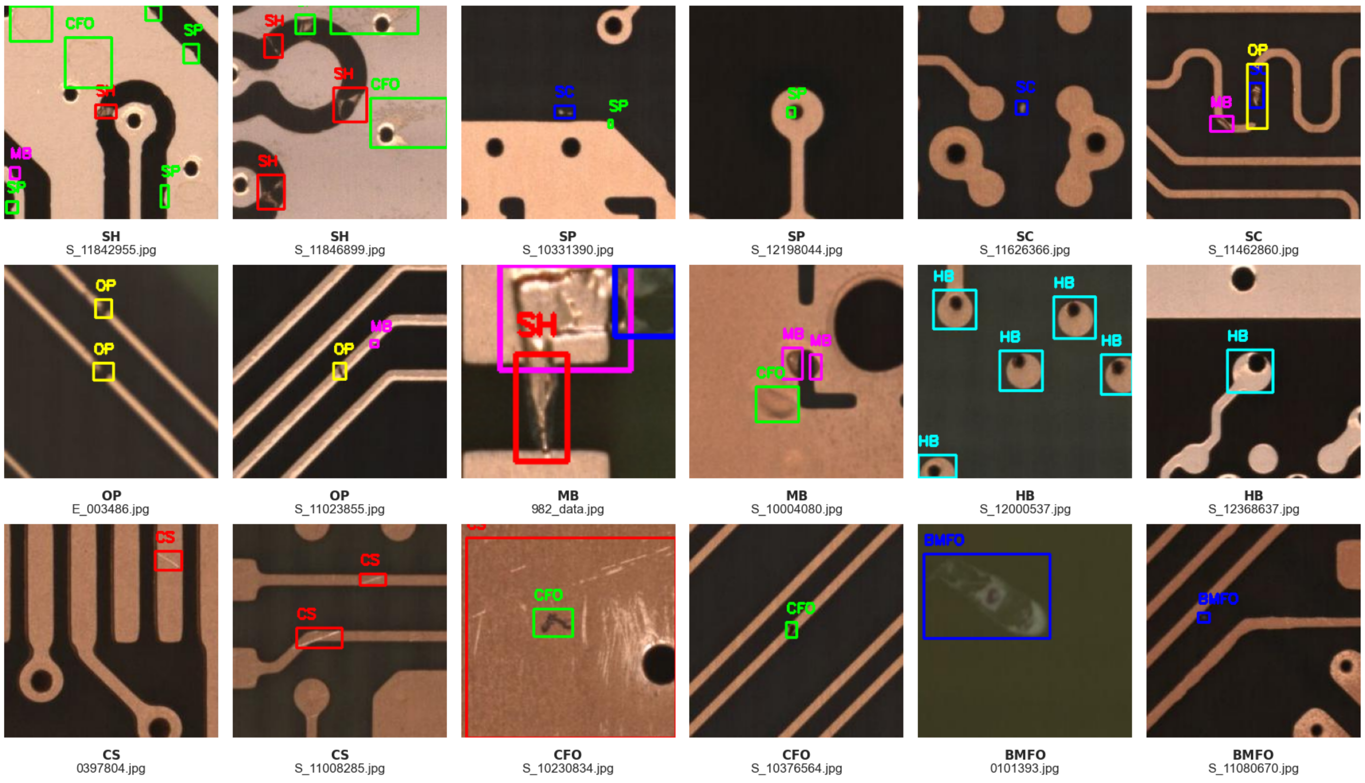


Fig. 4: Examples of data instances across the nine DsPCBSD+ classes.

Appendix A
Additional Figures

TABLE V: YOLOv8m-P2 test metrics at default confidence and validation-selected confidence.

Setting	Conf.	mAP50	mAP50-95	P	R	AP_S	AP_M	AP_L
P2 test @ default conf	0.250	0.841	0.531	0.801	0.792	0.375	0.493	0.383
P2 test @ val-selected conf*	0.400	0.828	0.529	0.810	0.787	0.362	0.475	0.376

TABLE VI: Global test-set metrics at last checkpoint (last.pt), including COCO sizs-aware AP (AP_S, AP_M, AP_L).

Model	P	R	F1	mAP50	mAP50-95	AP_S	AP_M	AP_L
v8m-Head	0.633	0.599	0.615	0.635	0.352	0.214	0.243	0.236
v8m-Neck+Head	0.785	0.756	0.770	0.796	0.498	0.362	0.472	0.622
v8m-Full	0.838	0.793	0.815	0.838	0.542	0.388	0.527	0.599
v8m-P2-Full	0.811	0.807	0.809	0.846	0.534	0.372	0.527	0.509

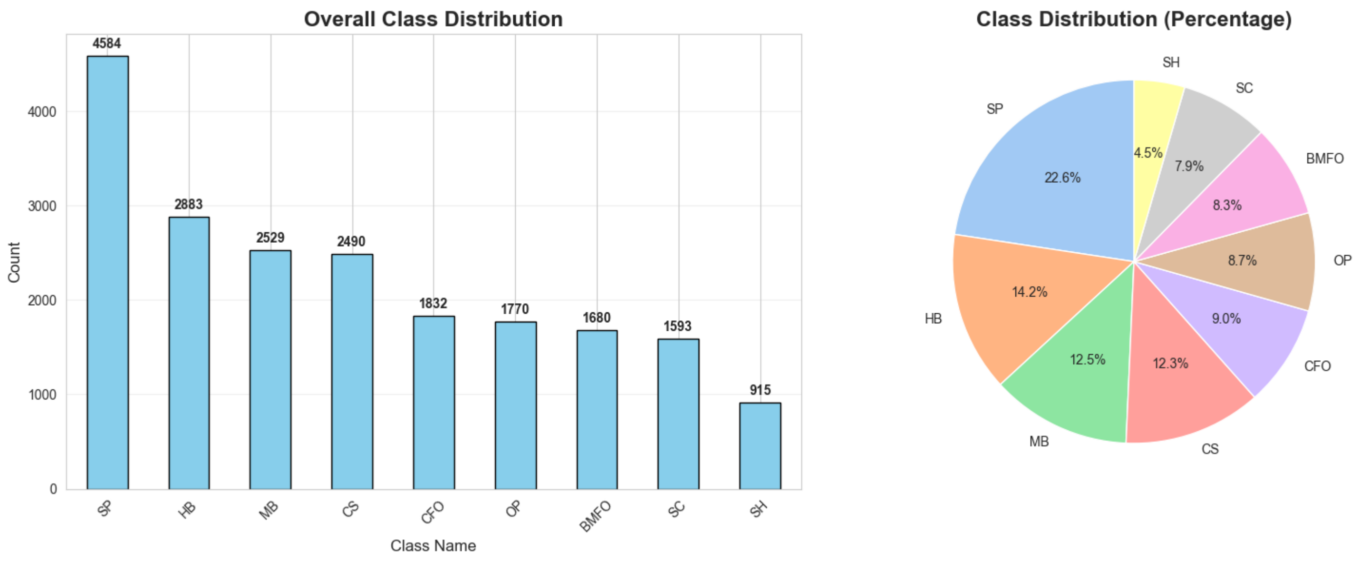


Fig. 5: Class distribution of defect instances across the nine DsPCBSD+ categories.

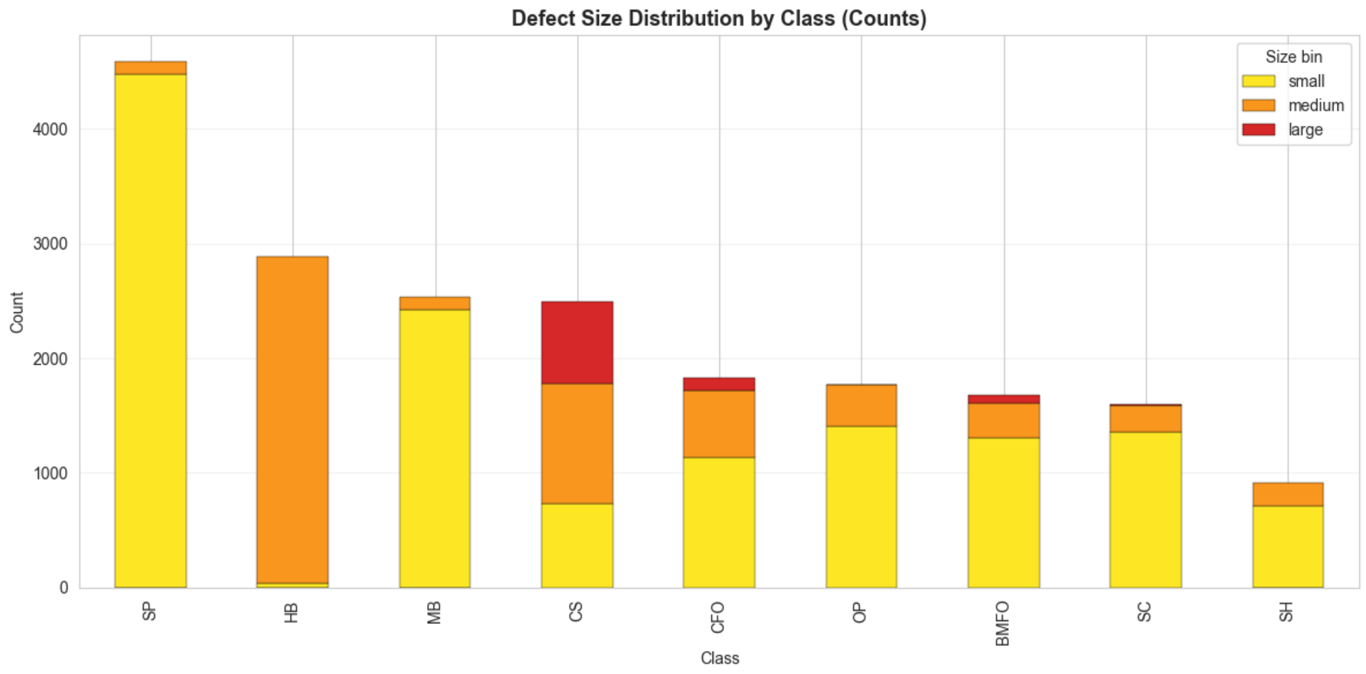


Fig. 6: Size distribution by class

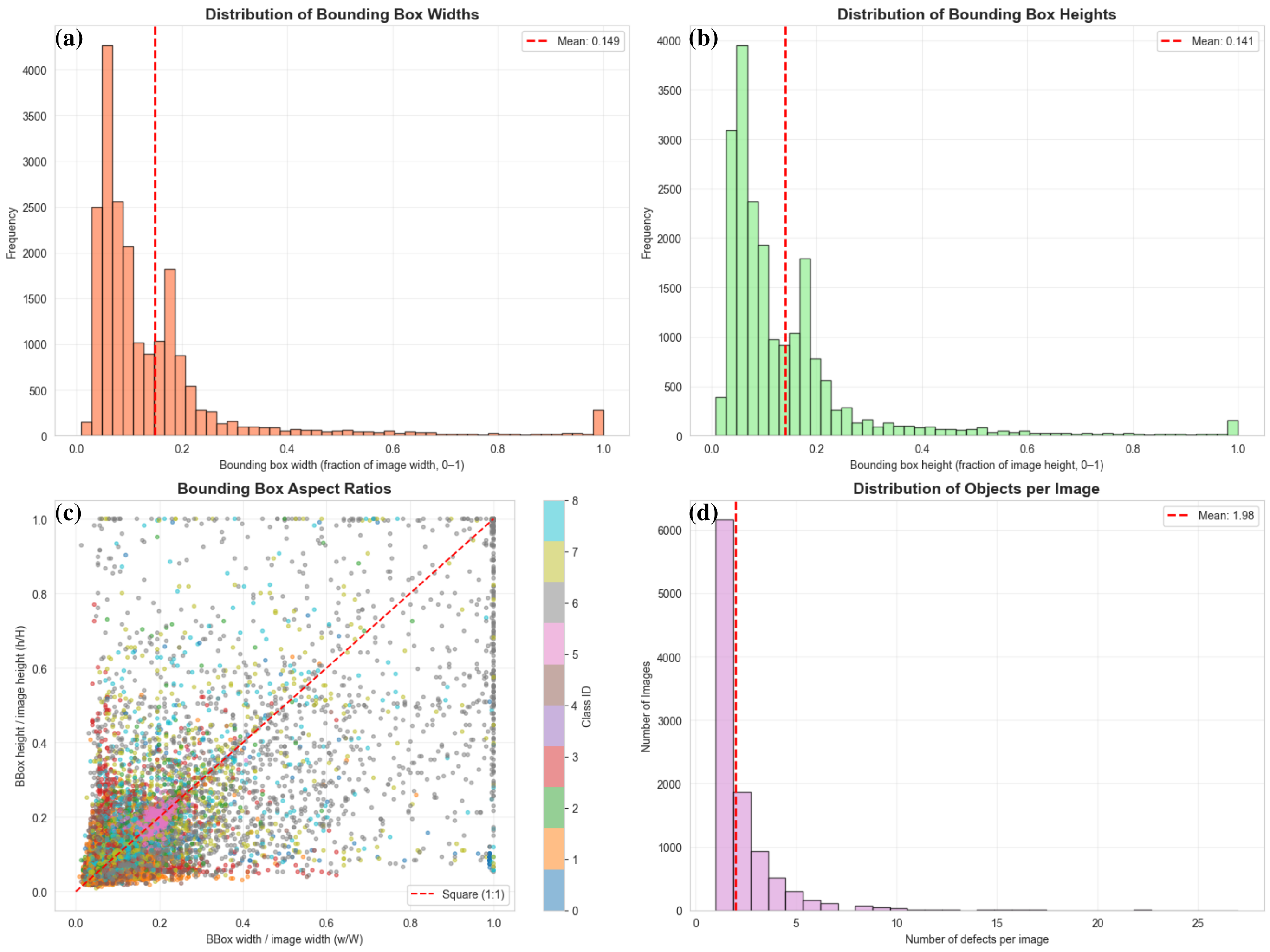


Fig. 7: All bounding-box dimensions are YOLO-normalized relative to each image; values are unitless ratios, not pixels.

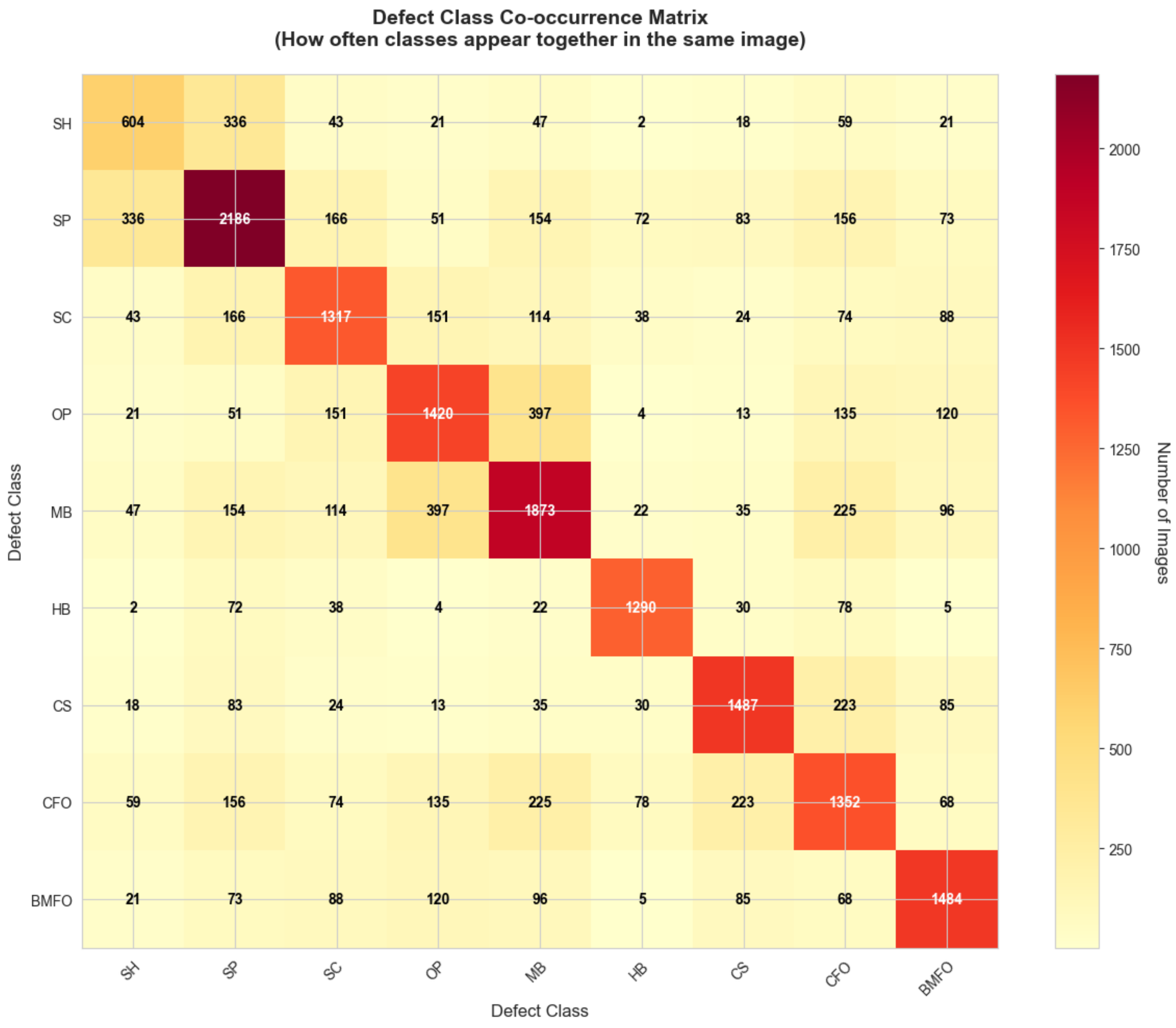


Fig. 8: Defect Class Co-occurrence Matrix - How often classes appear together in the same image.

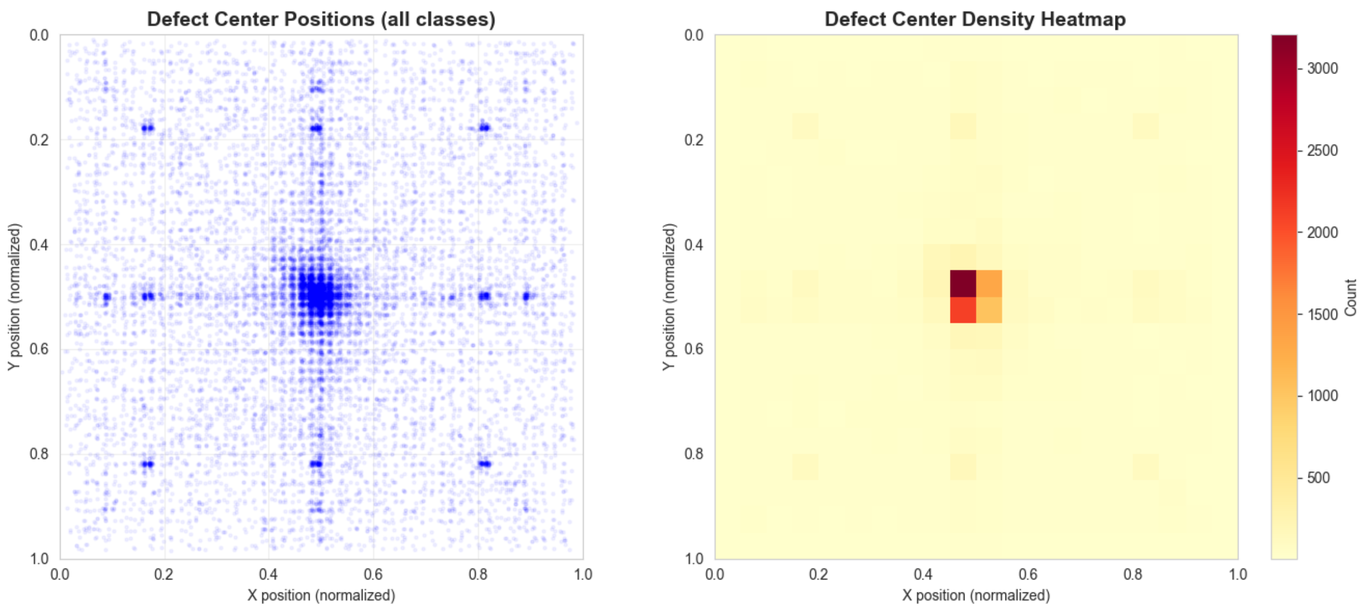


Fig. 9: Spatial distribution of defect centers in patch coordinates. Each point corresponds to the center (x_c, y_c) of a labeled bounding box from YOLO annotations.

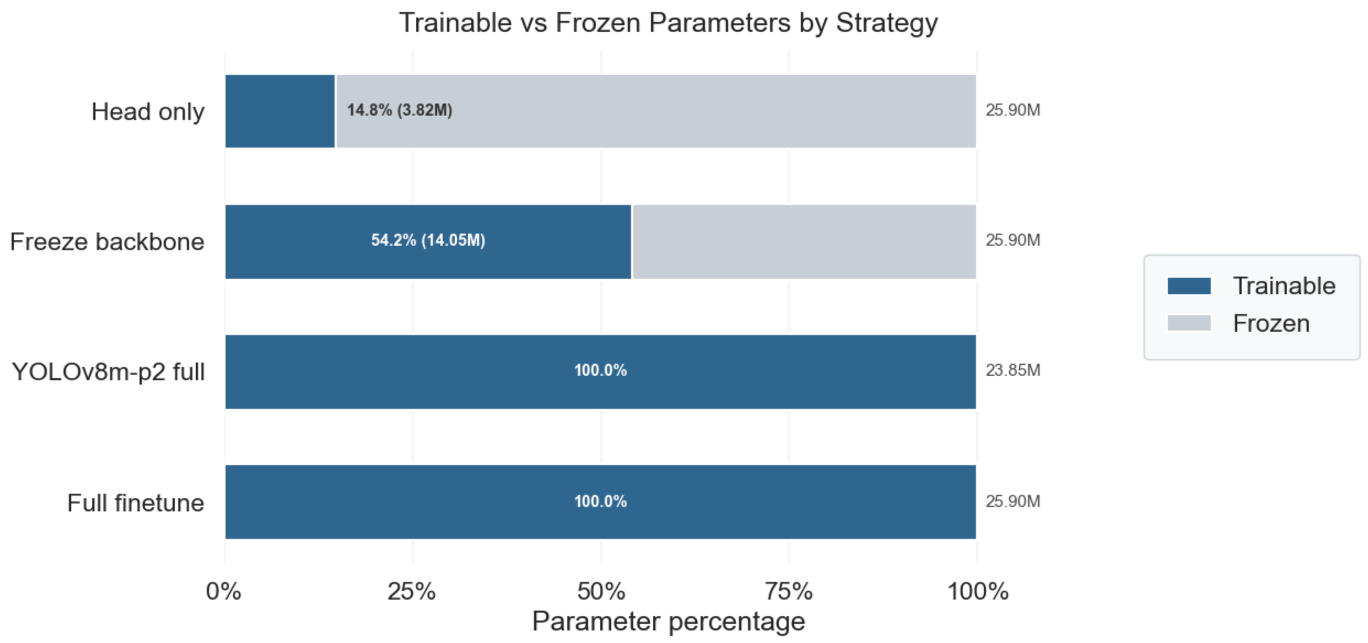
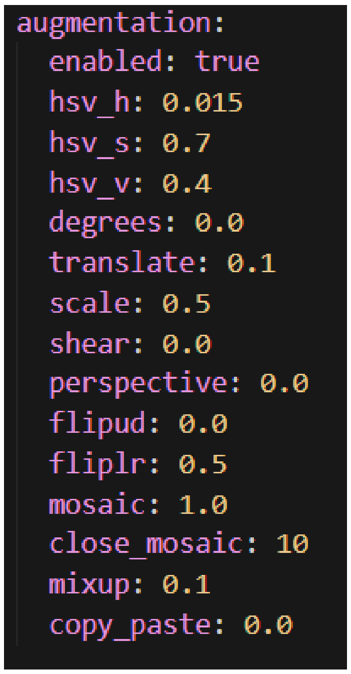


Fig. 10: Trainable versus frozen parameter composition for the four training strategies. Bars show the percentage of trainable (blue) and frozen (gray) parameters; text inside bars reports trainable share and trainable parameter count (in millions). Total parameter counts are shown at the bar ends (25.90M for YOLOv8m, 23.85M for YOLOv8m-p2) Head only, Freeze backbone and Full fine-tuning are names used to refer to training strategies of the YOLOv8m model.



```
augmentation:
  enabled: true
  hsv_h: 0.015
  hsv_s: 0.7
  hsv_v: 0.4
  degrees: 0.0
  translate: 0.1
  scale: 0.5
  shear: 0.0
  perspective: 0.0
  flipud: 0.0
  fliplr: 0.5
  mosaic: 1.0
  close_mosaic: 10
  mixup: 0.1
  copy_paste: 0.0
```

Fig. 11: Values assigned to the augmenation arguments