BoardVision: Deployment-ready and Robust Motherboard Defect Detection with YOLO+Faster-RCNN Ensemble

Brandon Hill
University of Maryland, Baltimore County
bhill@umbc.edu

KMA Solaiman[*]
University of Maryland, Baltimore County
ksolaima@umbc.edu

Abstract

*Motherboard defect detection is critical for ensuring reliability in high-volume electronics manufacturing. While prior research in PCB inspection has largely targeted bare-board or trace-level defects, assembly-level inspection of full motherboards inspection remains underexplored. In this work, we present **BoardVision**, a reproducible framework for detecting assembly-level defects such as missing screws, loose fan wiring, and surface scratches. We benchmark two representative detectors - YOLOv7 and Faster R-CNN, under controlled conditions on the MiracleFactory motherboard dataset, providing the first systematic comparison in this domain. To mitigate the limitations of single models, where YOLO excels in precision but underperforms in recall and Faster R-CNN shows the reverse, we propose a lightweight ensemble, **Confidence-Temporal Voting** (CTV Voter), that balances precision and recall through interpretable rules. We further evaluate robustness under realistic perturbations including sharpness, brightness, and orientation changes, highlighting stability challenges often overlooked in motherboard defect detection. Finally, we release a deployable GUI-driven inspection tool that bridges research evaluation with operator usability. Together, these contributions demonstrate how computer vision techniques can transition from benchmark results to practical quality assurance for assembly-level motherboard manufacturing.*

1. Introduction

Motherboards are the backbone of modern electronics, and their reliability is essential for large-scale manufacturing. Even subtle assembly-level defects such as missing screws, loose wiring, or scratches, can compromise system reliability and incur significant costs. Automated optical inspection (AOI) has long been used to detect surface-level anomalies, but most prior research and industrial solutions have concentrated on bare-board or trace-level PCB inspection [1, 11, 12], leaving motherboard inspection comparatively underexplored.

[*]Corresponding Author

Deep learning has advanced vision-based inspection with detectors such as Faster R-CNN [15] and YOLO variants [20]. Extensions of YOLOv7 [2, 16, 22, 23], YOLOv5-based models [17, 18], and SSD variants [9] have shown strong results for PCB and related defects. But most studies end with accuracy reports on curated datasets, with limited focus on robustness to factory perturbations like lighting, blur, or orientation [14], and few translate into deployable operator-facing systems.

Single detectors also exhibit trade-offs: higher precision or higher recall, but rarely both are achieved together under imperfect dataset distributions. Leveraging multiple models together in a lightweight voting ensemble could provide a more balanced trade-off, motivating our design.

To address these gaps, we introduce **BoardVision**, a reproducible motherboard defect detection framework designed with both research benchmarking and industrial deployment in mind. Our contributions are four-fold:

1. the first controlled benchmarking of YOLOv7 and Faster R-CNN on an assembly-level motherboard dataset (MiracleFactory) [21],
2. the first robustness-oriented ensemble in this space, a lightweight Voter model with interpretable rules that balances precision and recall across detectors under diverse perturbations,
3. comprehensive robustness evaluation under sharpness, brightness, and orientation perturbations, and
4. the first deployable GUI-driven inspection system for motherboard defects, capable of both live and offline inference.

By uniting rigorous benchmarking, robustness evaluation, and practical deployment, BoardVision demonstrates how vision research can transition from isolated metrics to meaningful, real-world quality assurance.

2. Related Work

Automated inspection of bare-board and trace-level printed circuit boards (PCBs) has progressed from rule-based techniques such as template matching and pixel differencing, which were fast but brittle to lighting and alignment variations [1, 12], to classical machine learning approaches that relied on handcrafted descriptors such as HOG and SIFT with classifiers like SVM or k-NN [4, 5]. While these pipelines achieved moderate accuracy, they struggled particularly with subtle or small defect classes and were quickly surpassed by deep CNN-based methods [5].

Deep learning has since become the dominant paradigm, with convolutional detectors applied across public PCB datasets such as DeepPCB [19], DsPCBSD+ [10, 11], and HRIPCB [13]. While these benchmarks enabled progress, they primarily emphasize trace-level or component-level anomalies. In contrast, assembly-level defects in motherboards (e.g., missing screws, detached fan ports, scratches) remain underexplored [17, 21], motivating use of the MiracleFactory dataset [21].

Among detection families, two-stage models like Faster R-CNN [15] have offered strong accuracy but suffer from slower inference and reduced stability under class imbalance. In contrast, one-stage models such as the YOLO family has shown a strong real-time detector performance across vision benchmarks [20]. Recent studies have shown that YOLOv7 achieves state-of-the-art accuracy and throughput across PCB and related manufacturing tasks: [22] improved YOLOv7 with attention modules for fine-grained PCB defect detection, [16] applied YOLOv7 to automated remanufacturing defects, and [2] optimized YOLOv7 for semiconductor wafer inspection. However, few studies directly compare YOLOv7 and Faster R-CNN under matched setups for motherboard-level defects.

More recent studies address broader inspection challenges. [7] proposed an adaptive YOLOv2+Faster R-CNN ensemble for DIP soldering, highlighting accuracy degradation under lighting and orientation variations and addressing it with continual self-adaptation. [14] further analyzed how lighting, contamination, and pose shifts impact defect detection models, emphasizing the need for robustness-oriented evaluation. Ensemble approaches such as voting or hybrid pipelines have also been explored [6], though practical deployment remains limited.

Finally, most PCB defect detection studies conclude with benchmark metrics, without advancing to deployable tools or operator-facing systems. In contrast, our work provides both rigorous benchmarking and practical utility: we deliver the first head-to-head evaluation of YOLOv7 and Faster R-CNN on an assembly-level motherboard dataset, introduce a lightweight ensemble (CTV Voter) with interpretable rules to enhance robustness, and release a GUI-driven inspection tool. This combination bridges academic evaluation with factory-floor usability, positioning BoardVision as both a reproducible research artifact and a practical quality assurance aid.

3. System Overview

BoardVision is designed as a modular, deployment-ready pipeline for automated motherboard defect detection. The full system is shown in Fig. 1, which integrates three key stages: input processing, ensemble inference, and operator-facing visualization.

Input and Preprocessing. Images or video streams (either offline files or live camera feeds) are first passed through a preprocessing stage where YOLOv7 and Faster R-CNN detectors are initialized. These models then run on a CPU or GPU to produce bounding boxes and their corresponding confidence scores.

Ensemble Inference. Outputs from the detectors are reconciled by the Confidence–Temporal Voting (CTV) module. This layer matches detections across models using IoU criteria and applies either solo rules (e.g., high-confidence overrides, model preference) or agreement fusion (confidence- and F1-weighted averaging) before applying non-max suppression. The details of CTV are described in Section 4.

Visualization and GUI. The final predictions are overlaid on the input stream and presented through a PySide6-based GUI. The system supports two execution modes: (1) *streaming mode*, for live video inspection, and (2) *file mode*, for batch evaluation of offline images and videos. The interface logs model disagreements, exposes ensemble decisions, and allows operators to adjust key parameters in real time. Together, these components frame BoardVision as a practical QA tool, bridging algorithmic advances with user-facing deployment.

4. Method: Voter Algorithm

While YOLOv7 achieves high overall accuracy and Faster R-CNN produces more stable detections under class imbalance, each exhibits complementary failure modes. To exploit these strengths, we introduce a lightweight ensemble strategy called **Confidence-Temporal Voting (CTV)**. Fig. 1 includes the CTV workflow. The full pseudocode is provided in Algorithm 1 in Appendix.

4.1. Parameter Intuition

CTV is governed by a small set of tunable and interpretable parameters that control pairing, confidence weighting, and solo-detection rules:

- **IoU threshold** (t_{IoU}): minimum overlap required to pair detections across models.
- **Confidence exponent** (γ): emphasizes high-confidence predictions.

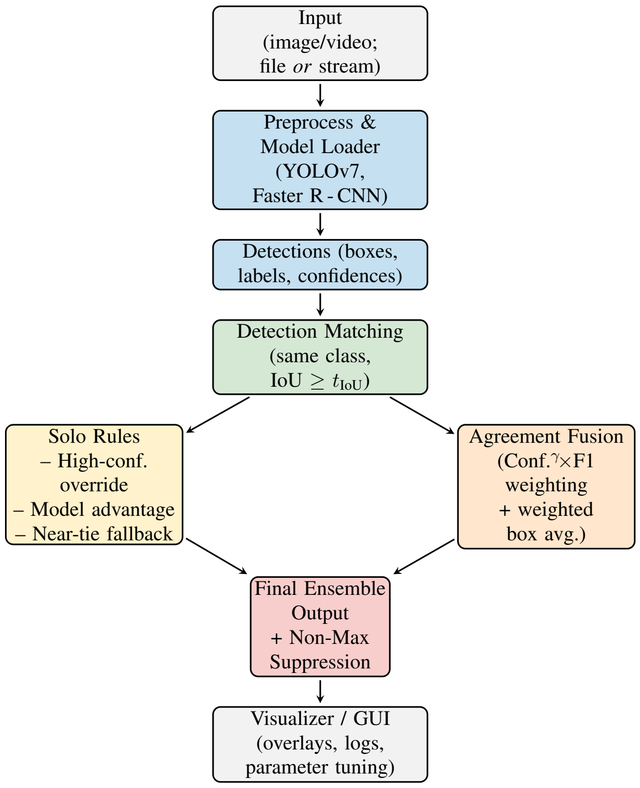

Input (image/video; file *or* stream)
Preprocess & Model Loader (YOLOv7, Faster R-CNN)
Detections (boxes, labels, confidences)
Detection Matching (same class, IoU ≥ t_{IoU})
Solo Rules – High-conf. override – Model advantage – Near-tie fallback
Agreement Fusion (Conf.$^\gamma$×F1 weighting + weighted box avg.)
Final Ensemble Output + Non-Max Suppression
Visualizer / GUI (overlays, logs, parameter tuning)

Figure 1. BoardVision pipeline: YOLOv7 and Faster R-CNN outputs are reconciled by the Confidence–Temporal Voting ensemble and visualized through a GUI for operator-facing deployment.

- **Class F1 margin** ($f1\_margin$): tolerance for treating two models as equivalent on a class.
- **Solo confidence thresholds**: govern when to retain unmatched detections - $conf\_thresh$, $solo\_strong$.

4.2. Detection Matching

Per-frame detections from YOLOv7 and Faster R-CNN are represented as bounding boxes with class labels and confidence scores. Let i and j be the indices for the detections from the YOLOv7 and Faster R-CNN models, respectively. For each YOLO detection y_i, we attempt to pair it with a Faster R-CNN detection f_j of the same class if their intersection-over-union (IoU) exceeds a threshold (t_{IoU}). Two cases then follow before the final decision:

- **Agreement:** If a valid agreement pair (y_i, f_j) is found, the detections are merged through a weighted voting scheme that balances instance confidence with class-level reliability. A worked example of this fusion process is provided in Appendix A.
- **Solo:** If no valid pair is found, we apply a solo-labeling heuristic that decides whether to retain the detection based on interpretable rules.

4.2.1. Agreement Cases

Confidence-weighted scoring An exponent γ is applied to model confidences, amplifying the influence of higher-confidence predictions during fusion. For an agreement pair of class c, we assign each model a **fusion score**:

$$S_{\text{YOLO}}(i) = \left(p_i^{\text{YOLO}}\right)^\gamma \cdot F1_{\text{YOLO},c} \tag{1}$$

$$S_{\text{FRCNN}}(j) = \left(p_j^{\text{FRCNN}}\right)^\gamma \cdot F1_{\text{FRCNN},c} \tag{2}$$

where $p_i^{\text{YOLO}}, p_j^{\text{FRCNN}}$ are the model confidence scores for YOLO and FRCNN detections, and $F1_{\cdot,c}$ is the model's validation F1 score for class c.

This formula balances instance-level confidence (per detection) with class-level trustworthiness (per model). A model that performs better historically on class c exerts more influence, even if its raw confidence is slightly lower.

Bounding Box for Fused Prediction The final bounding box is a weighted average between S_{YOLO} and S_{FRCNN}, weighted by bounding boxes from YOLO (b_Y) and FRCNN (b_R):

$$b^\star = \frac{S_{YOLO} \cdot b_Y + S_{FRCNN} \cdot b_R}{S_{YOLO} + S_{FRCNN}} \tag{3}$$

Finally, the fused confidence is calculated as $\max(p_Y, p_R)$.

4.2.2. Solo Cases

When no valid pair is found, CTV applies three interpretable rules to decide whether to keep a detection:

(I) **High-confidence override:** Retain any detection with confidence above the `solo_strong` threshold.

(II) **Model advantage:** Retain if the model's per-class F1 score is higher than its competitor and the confidence exceeds `conf_thresh`.

(III) **Near-tie fallback:** Retain if the two models perform similarly on a specific class, in terms of their F1-score (within $f1\_margin$) and the confidence is at least 0.95.

5. Experimental Setup and Results

5.1. Dataset

We evaluate on the publicly available **MiracleFactory Motherboard Defect Dataset** [21], which contains 389 high-resolution images and 2,860 annotated instances across 11 defect categories. This dataset was designed for visual quality assurance (QA) in motherboard manufacturing, but it poses several challenges: (1) the number of samples is modest compared to common detection benchmarks; (2) class distribution is highly imbalanced, with common classes such as *Screws* dominating, while rare but operationally important categories such as *Loose Screws* and *CPU_fan_port_detached* contain fewer than 100 instances; and (3) many labels represent fine-grained, visually similar categories (e.g., different screw conditions), making accurate discrimination difficult under imbalance. Table 1 summarizes the distribution.

Table 1. Distribution of annotated instances across defect classes.

Class Name	Instance Count
Screws	806
CPU_FAN_Screws	685
CPU_FAN_NO_Screws	326
CPU_fan	313
No_Screws	196
CPU_fan_port	159
CPU_FAN_Screw_loose	99
Scratch	95
Incorrect_Screws	63
CPU_fan_port_detached	60
Loose_Screws	58

5.2. Training and Evaluation Protocols

We compare YOLOv7 and Faster R-CNN under matched training setups to ensure fairness. Both were trained for 50 epochs with early stopping. Models were trained with stochastic gradient descent and comparable learning rates, using input size of 640×640. YOLOv7 followed its official PyTorch implementation with a base learning rate of 0.01, while Faster R-CNN used a ResNet-50 FPN backbone via torchvision with a base learning rate of 0.005. For the CTV approach, we used $t_{IoU} = 0.4$, $\gamma = 2$, and $f1\_margin = 0.05$, with $p_i^{\text{YOLO}} = 0.6$, $p_j^{\text{FRCNN}} = 0.9$, $conf\_thresh = 0.6$ and $solo\_strong = 0.95$ as default values.

Inference-time robustness tests uses image augmentations, including horizontal flipping, increased sharpness, and varied brightness. To address class imbalance, no resampling was applied, reflecting real world deployment conditions.

Training was performed on a local workstation (NVIDIA GTX 1080 GPU with 8GB VRAM, AMD 3900x CPU, 32GB RAM). Models were implemented in PyTorch 2.0 with CUDA 11.7. Checkpoints were saved every epoch, and the model with the highest validation performance was selected for final evaluation. This ensured consistent convergence monitoring while enabling reproducibility of results.

We evaluate all models on a held-out test set of 45 images (640×640), containing 365 annotated instances across all 11 classes. Standard object detection metrics were used to evaluate model performance [3, 8]:

- **mAP@0.5**: Mean Average Precision at an Intersection over Union (IoU) threshold of 0.5, measuring bounding box localization and classification accuracy.
- **mAP@0.5:0.95**: A stricter metric averaging mAP across IoU thresholds from 0.5 to 0.95 (in 0.05 increments), capturing both coarse and fine localization quality.
- **Precision** (**P**): Ratio of correctly predicted defect instances to total predictions, reflecting false positive rates.
- **Recall** (**R**): Proportion of ground truth instances correctly identified, reflecting false negative rates.
- **F1-score** (**F1**): Harmonic mean of precision & recall.
- **FPS (frames/sec)**: The number of images processed per second, indicating real-time viability.

5.3. Full Quantitative Results

Table 2 compares YOLOv7, Faster R-CNN (FRCNN), and the proposed voter ensemble (CTV) across aggregate detection metrics. As expected, YOLOv7 delivers the highest overall performance, achieving superior mAP@0.5, precision, and F1-score, while also running over twice as fast as Faster R-CNN. Faster R-CNN lags across most metrics, reflecting its sensitivity to small, cluttered objects and the dataset's class imbalance. The ensemble improves upon YOLOv7 by balancing precision and recall, yielding the highest F1-score (0.964) and top precision (0.967). This confirms that the voter algorithm enhances stability without sacrificing YOLOv7's high recall, and provides an interpretable pathway for integrating complementary detectors.

Table 2. Aggregate performance comparison across models.

Metric	YOLOv7	FRCNN	CTV
mAP@0.5	0.914	0.766	**0.921**
mAP@0.5:0.95	**0.606**	0.495	0.604
Precision	0.964	0.953	**0.967**
Recall	0.956	0.718	**0.962**
Mean F1-score	0.960	0.819	**0.964**
FPS (frames/sec)	22–25	8–10	8–10

Table 3 provides a per-class breakdown of F1-scores across the 11 defect categories. YOLOv7 achieves near-perfect performance on most of the classes, including frequent categories such as *Screws* and *CPU_FAN_Screws*, as well as rare but operationally critical types like *Loose_Screws*. Faster R-CNN, in contrast, collapses on rare or visually subtle categories (e.g., *CPU_FAN_NO_Screws*), reflecting its sensitivity to class imbalance. The voter ensemble maintains YOLOv7's strong scores while selectively improving specific categories, most notably *No_Screws*, where it raises F1 from 0.926 to **0.963** by selectively retaining high-confidence Faster R-CNN detections. These results confirm that the ensemble preserves YOLOv7's dominance while mitigating edge-case failures that are critical in deployment.

Table 3. Per-class F1-scores across models.

Class	YOLOv7	FRCNN	CTV
CPU_FAN_NO_Screws	**0.907**	0.000	**0.907**
CPU_FAN_Screw_loose	**0.933**	0.615	**0.933**
CPU_FAN_Screws	**1.000**	0.959	**1.000**
CPU_fan	0.987	**1.000**	**1.000**
CPU_fan_port	**0.974**	0.789	**0.974**
CPU_fan_port_detached	**0.667**	0.462	**0.667**
Incorrect_Screws	**0.857**	0.667	**0.857**
Loose_Screws	**1.000**	0.800	**1.000**
No_Screws	0.926	0.851	**0.963**
Scratch	**0.900**	**0.900**	**0.900**
Screws	**0.987**	0.895	**0.987**

Overall, these results show that ensemble recovers stability on difficult classes without hurting throughput. This validates the value of the proposed confidence-temporal voting scheme for industrial QA.

Failure Modes and Error Analysis. The per-class F1 scores (Table 3) and error profiles (Table 4) highlight systematic strengths and weaknesses of each detector. YOLOv7 achieves strong separation on most categories, but struggles with *CPU_fan_port_detached*, where only 0.67 of samples are correctly identified, consistent with its lower F1 score for this class. Faster R-CNN shows broader distributed errors, with 103 false negatives overall, reflecting a strong recall deficit. In contrast, YOLOv7 produces more false positives (13) despite higher recall.

The proposed voter ensemble inherits YOLOv7's precision while reducing Faster R-CNN's false negatives, achieving 351 true positives with only 12 false negatives. Notably, categories such as *No_Screws* improve from 0.93 F1 with YOLOv7 to 0.96 with the ensemble, and *Loose_Screws* is preserved at 1.00 compared to 0.67 for Faster R-CNN. These results confirm that the ensemble provides more balanced error trade-offs, especially for rare but high-cost classes. These stability gains extend under perturbations, as further analyzed in Section 6.

Full confusion matrices are included in the appendix for reference, but the main operational insights are captured by the compact FP/FN counts and per-class F1 scores.

Table 4. Error profile across models.

Method	TP	FP	FN
YOLOv7	349	13	16
Faster R-CNN	262	13	103
Voter	351	12	14

5.4. Qualitative Examples

In this section we illustrate how each model performs on real motherboard images. By comparing side-by-side predictions from YOLOv7 and Faster R-CNN (Figure 3) on the same test sample, we can better understand practical differences and failure modes in their behavior.

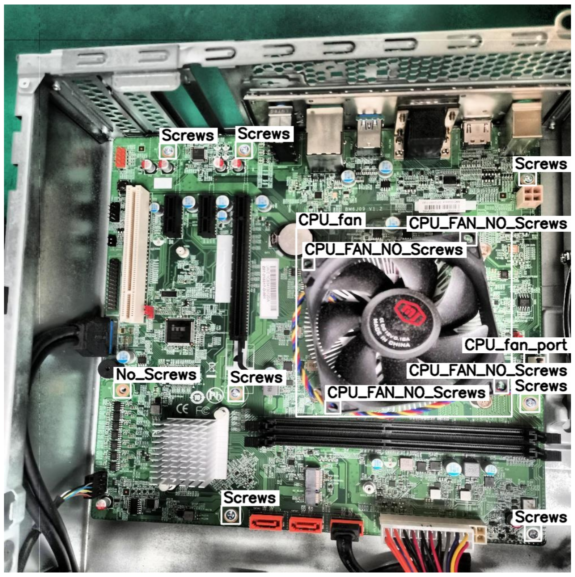
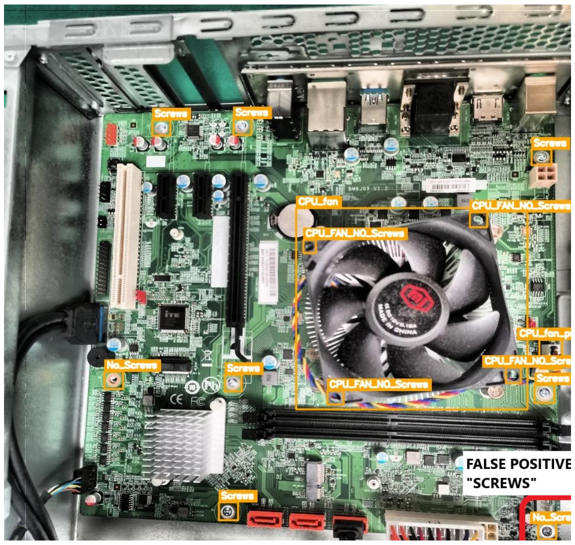


Figure 2. Ground truth annotations for the motherboard test image used in Figures 3 and 4.

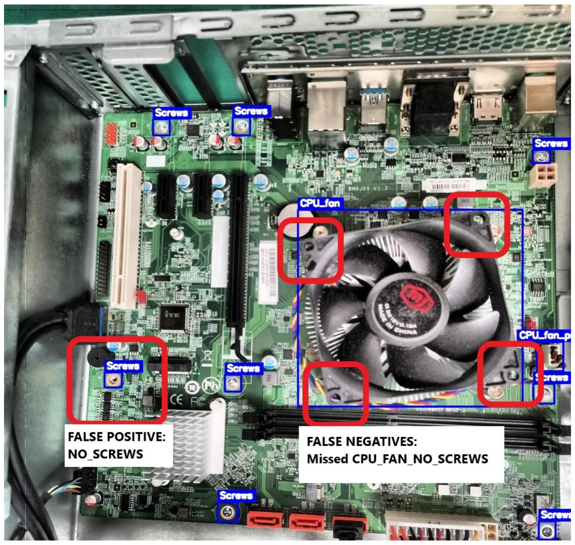
Both single models identify key components like *Screws* and *CPU_fan*, but differences in granularity and class distinction are evident. In Figure 3a, YOLOv7 outputs precise, non-overlapping predictions with clear boundaries and minimal redundancy. It correctly identifies multiple *Screws* regions and distinguishes *CPU_fan* components without overlap or redundancy. YOLOv7's predictions are visually coherent and class-consistent, reflecting its superior per-class F1 scores.

Figure 3 highlights typical failure modes. In both cases, a false positive for *No_Screws* appears in a region where screws exist, indicating that metallic reflections and shadows near screw holes can trigger confusion. Faster R-CNN also misses a true *CPU_FAN_NO_Screws* instance within the fan mount, suggesting reduced sensitivity to small absences in dense areas. YOLOv7 incorrectly labels an existing screw as *No_Screws* in a crowded region.

The voter ensemble consistently corrects these issues. By fusing detections from both models with agreement-aware scoring and cross-model suppression, the voter removes spurious *No_Screws* predictions, restores missed *CPU_FAN_NO_Screws* near the fan mount, and preserves confident *Screws* and *CPU_fan* boxes. Qualitatively, Figure 4 shows tighter boxes, fewer label flips, and better separation of visually similar classes.



(a) YOLOv7 – FP: Screws



(b) Faster R-CNN – FP: No Screws, FN: CPU_FAN_NO_Screws

Figure 3. Annotated error cases for YOLOv7 and FRCNN. False positives (FP) and false negatives (FN) are annotated in red to illustrate specific failure modes.

6. Robustness and Sensitivity Analysis

Beyond baseline performance, we analyze the stability of BoardVision under both external perturbations (image augmentations) and internal variations (ensemble parameters and ablation).

Robustness Evaluation. To evaluate robustness, we tested all models on the original held-out test set and on augmented variants designed to simulate real-world perturbations:

- **flip**: Horizontal mirroring of images to simulate viewpoint changes and test invariance to orientation.
- **increased sharpness**: via Gaussian unsharp masking to mimic sensor or compression artifacts.
- **increased brightness**: via linear-RGB exposure adjustment, simulating overexposed, well-lit conditions.
- **decreased brightness**: using the same method, simulating underexposed or poorly lit conditions.

Such perturbations are representative of deployment scenarios where camera streams may vary in lighting, sharpness, or orientation, particularly in low-cost or factory-floor settings. The voter was run with default parameters, with per-class weights initialized from preliminary F1 scores of the base models. Tables 5-7 refer to N = Normal, F = Flip, SUp = Sharpness Up, BUp = Brightness Up, BDn = Brightness Down.

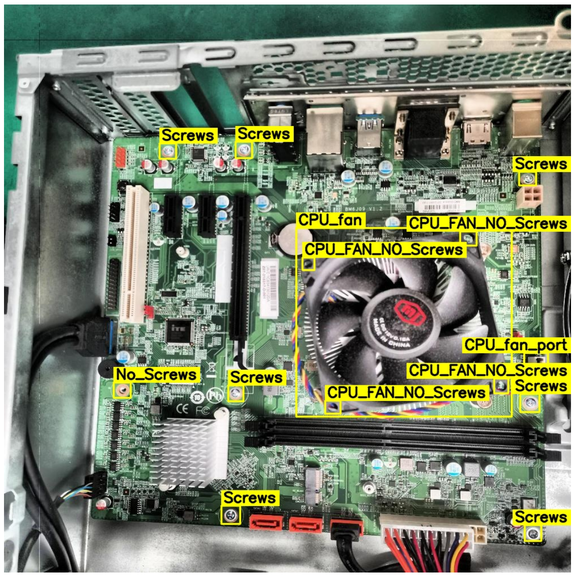


Figure 4. Voter Predictions for motherboard test image.

Table 5. YOLOv7 Performance under Perturbations.

Metric	Mean ± Std	N	F	SUp	BUp	BDn
P	0.962 ± 0.006	0.964	0.969	0.950	0.964	0.967
R	0.949 ± 0.009	0.956	0.953	0.934	0.951	0.959
F1	**0.958 ± 0.007**	0.960	0.961	0.942	0.957	0.963
FPS	22–25					

Table 6. Faster R-CNN Performance under Perturbations.

Metric	Mean ± Std	N	F	SUp	BUp	BDn
P	0.954 ± 0.008	0.953	0.967	0.945	0.945	0.959
R	0.713 ± 0.003	0.718	0.721	0.710	0.712	0.712
F1	0.816 ± 0.005	0.819	0.826	0.811	0.812	0.818
FPS	8-10					

Across augmentations shown in Table 5, YOLOv7 maintains strong Recall but suffers drops under *Sharp_Up* (F1

Table 7. CTV Performance under Perturbations.

Metric	Mean ± Std	N	F	SUp	BUp	BDn
P	**0.964 ± 0.006**	0.967	0.972	0.953	0.961	0.967
R	**0.954 ± 0.009**	0.962	0.948	0.940	0.956	0.962
F1	*0.957 ± 0.006*	0.964	0.960	0.946	0.959	0.964
FPS	8-10					

= 0.942), exposing some fragility to high-frequency distortions. Faster R-CNN degrades more substantially, with Mean F1 = 0.816. Its heavier reliance on region proposals makes it sensitive to small feature perturbations, leading to lower stability across all conditions. The Voter Ensemble in Table 7 matches YOLOv7's F1 (0.957 vs. 0.958) while delivering higher precision and lower variance. Across augmentations, the Voter's F1 ranges from 0.946 (Sharp_Up) to 0.964 (Normal), consistently narrowing YOLOv7's fluctuations. Even when YOLOv7 dips under augmentation like in Sharp_Up, the Voter stabilizes performance by leveraging cross-model agreement. The voter's stability across perturbations demonstrates robustness: even when a single model wavers, the ensemble preserves consistent performance, making it more reliable for deployment on noisy or imperfect inspection streams.

Table 8. Sensitivity of CTV parameters across IoU, γ, and solo thresholds. Bold values indicate notable deviations.

Voter Params	mAP@0.5	mAP@0.5:0.95	P	R
t_{IoU} = 0.3	0.921	0.604	0.967	0.962
t_{IoU} = 0.5	0.921	0.604	0.967	0.962
t_{IoU} = 0.7	**0.919** $\downarrow$	0.604	**0.949** $\downarrow$	0.962
γ = 0	**0.917** $\downarrow$	**0.605** $\uparrow$	**0.964** $\downarrow$	**0.959** $\downarrow$
γ = 1	0.921	**0.605** $\uparrow$	0.967	0.962
γ = 2	0.921	0.604	0.967	0.962
γ = 3	0.921	**0.596** $\downarrow$	0.967	0.962
solo=0.90	**0.910** $\downarrow$	**0.595** $\downarrow$	**0.944** $\downarrow$	0.962
solo=0.95	**0.912** $\downarrow$	**0.597** $\downarrow$	**0.956** $\downarrow$	0.962
solo=0.98	0.921	0.604	0.967	0.962

Sensitivity to Voter Parameters. As Table 8 shows, CTV is stable across a wide range of parameters, but extremes degrade performance. IoU thresholds of 0.3–0.5 perform identically, while raising to 0.7 reduces precision (0.949) and mAP@0.5 (0.919). The exponent γ performs best at 1–2, whereas $\gamma = 0$ removes YOLO's precision bias and lowers recall (0.959), and $\gamma = 3$ suppresses mAP@0.5:0.95 (0.596). Solo thresholds below 0.98 admit too many weak detections, collapsing precision to 0.944–0.956. The default ($t_{IoU} = 0.5, \gamma = 2, solo = 0.98$) sits near the optimal balance.

Single-Factor Ablation of Ensemble Rules. The ablation study in Table 9 confirms that the full CTV ensemble achieves the strongest balance, with mAP@0.5 of 0.921 and precision/recall both above 0.96. Removing the high-confidence override notably degrades precision (-2.3%) and reduces mAP, showing its importance for reducing false positives and driving measurable precision gains. Eliminating the per-class F1 weighting or relaxing tie rules causes only minor drops, indicating they fine-tune performance but are not as critical as the override. Overall, the study highlights that the ensemble rules improve robustness by preserving high-confidence detections and stabilizing recall without sacrificing precision.

Table 9. Ablation study of Ensemble Components.

Config	mAP@0.5	mAP@0.5:0.95	P	R
Full Ensemble	0.921	0.604	0.967	0.962
No High confidence	0.910	0.595	0.944	0.962
No Per-class F1 weighting	0.912	0.601	0.956	0.962
Always-tie rule	0.912	0.597	0.956	0.962

Table 10. IoU and γ sensitivity. Defaults: t_{IoU}=0.4, $\gamma = 2$.

t_{IoU}	γ	mAP@0.5	mAP@0.5:0.95	P	R
0.30	0	**0.917** $\downarrow$	0.605	0.964	**0.959** $\downarrow$
0.30	1	0.921	0.605	0.967	0.962
0.50	0	**0.917** $\downarrow$	0.605	0.964	**0.959** $\downarrow$
0.50	1	0.921	0.605	0.967	0.962
0.70	0	0.919	**0.606** $\uparrow$	**0.949** $\downarrow$	0.962
0.70	3	0.919	**0.596** $\downarrow$	**0.949** $\downarrow$	0.962

Multi-Factor Stress Tests. Table 10 shows that IoU and γ interact smoothly near defaults, but extreme values combine poorly: e.g., t_{IoU}=0.70 with $\gamma = 3$ reduces precision to **0.949** $\downarrow$ and mAP@0.5:0.95 to **0.596** $\downarrow$.

Table 11. Impact of per-class F1 weighting under stress.

t_{IoU}	γ	f1_margin	mAP@0.5	P	R
0.30	0	Original	0.917	0.964	0.959
0.30	0	All-ones	**0.764** $\downarrow$	0.970	**0.718** $\downarrow$
0.50	1	Original	0.921	0.967	0.962
0.50	1	All-ones	**0.768** $\downarrow$	0.974	**0.721** $\downarrow$
0.70	3	Original	0.919	0.949	0.962
0.70	3	All-ones	**0.767** $\downarrow$	0.970	**0.718** $\downarrow$

Table 11 reveals the hidden importance of per-class F1 weighting. When replaced with uniform all-ones weights, recall collapses by **24%** $\downarrow$ (0.962 $\rightarrow$ 0.718) and mAP@0.5 drops to **0.76** $\downarrow$. This shows that F1 weighting is essential under class imbalance, ensuring rare but critical defects are not ignored.

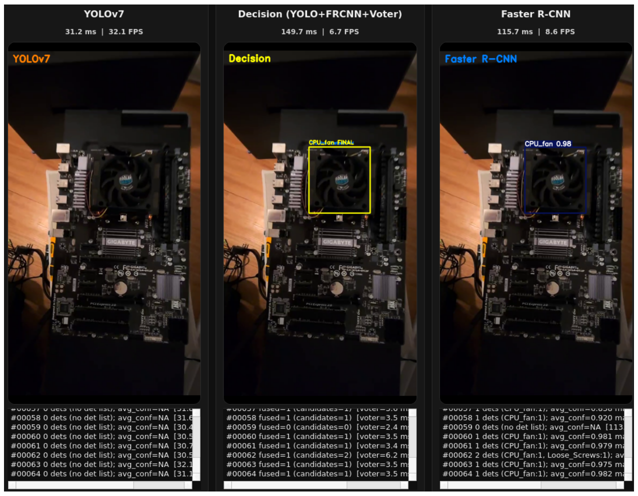
The sensitivity and ablation analyses jointly demonstrate that CTV's design is not redundant. The high-confidence override provides the main precision boost, while F1 weighting and tie rules act as stabilizers - modest in isolation but essential under stress.

7. BoardVision GUI Application

To bridge benchmarking with deployment, we developed a lightweight graphical user interface (GUI) for BoardVision using Python's PySide6/Qt framework.

With a **3-column synchronized view**, the GUI enables side-by-side inspection of YOLOv7 and Faster R-CNN predictions, while transparently displaying the fused ensemble output in real time. This design operationalizes the performance trends described in Section 5.2 and makes model disagreements directly observable to practitioners.

The interface supports flexible input sources (local files, webcams, network streams) and allows users to adjust key ensemble parameters such as `conf_thresh`, `solo_strong`, and `iou_thresh`. User can also adjust runtime behavior (stride for frame skipping, pause/resume, or stop). Real-time logs and overlays expose the Confidence–Temporal Voting rules (Section 4), enabling practitioners to interpret how detections are promoted, suppressed, or fused during operation. **The left panel of the GUI** givess access to these controls.



```
#00058 0 dets (no det list); avg_conf=NA  [31.6
#00059 0 dets (no det list); avg_conf=NA  [30.4
#00060 0 dets (no det list); avg_conf=NA  [30.5
#00061 0 dets (no det list); avg_conf=NA  [30.7
#00062 0 dets (no det list); avg_conf=NA  [30.5
#00063 0 dets (no det list); avg_conf=NA  [32.1
#00064 0 dets (no det list); avg_conf=NA  [31.1
```

```
#00058 fused=1 (candidates=1)  [voter=3.5 ms
#00059 fused=0 (candidates=0)  [voter=2.4 ms
#00060 fused=1 (candidates=1)  [voter=3.5 ms
#00061 fused=1 (candidates=1)  [voter=3.4 ms
#00062 fused=1 (candidates=2)  [voter=6.2 ms
#00063 fused=1 (candidates=1)  [voter=3.5 ms
#00064 fused=1 (candidates=1)  [voter=3.5 ms
```

```
#00058 1 dets (CPU_fan:1); avg_conf=0.920 ma
#00059 0 dets (no det list); avg_conf=NA  [113
#00060 1 dets (CPU_fan:1); avg_conf=0.981 ma
#00061 1 dets (CPU_fan:1); avg_conf=0.979 ma
#00062 2 dets (CPU_fan:1, Loose_Screws:1); av
#00063 1 dets (CPU_fan:1); avg_conf=0.975 ma
#00064 1 dets (CPU_fan:1); avg_conf=0.982 ma
```

Figure 5. Live inference on a low-light, low-resolution stream. YOLOv7 misses a CPU fan, Faster R-CNN detects it with high confidence ($p = 0.98$), and the voter ensemble fuses the result into the final decision (yellow box).

8. Discussion and Future Work

Our study surfaced several practical lessons for applied defect detection. First, ensemble design can remain simple yet effective: interpretable rules such as high-confidence overrides and per-class weighting delivered stability without requiring complex meta-models. Second, dataset imbalance emerged as the primary bottleneck: rare but operationally critical defects consistently challenged single models, underscoring the need for either targeted data collection or ensemble mechanisms that preserve rare detections. Third, robustness analysis showed that stability across perturbations is as important as raw mAP, since factory conditions rarely match controlled benchmarks. Finally, deployment through the GUI demonstrated that transparency is crucial: visualizing candidate and fused boxes not only improved user trust but also made failure modes visible in ways that metrics alone cannot. These insights position BoardVision as a reproducible blueprint for building interpretable, deployment-ready vision systems in manufacturing.

Future work will focus on three directions: expanding dataset coverage to address class imbalance (particularly for rare but high-cost classes), adapting ensemble rules dynamically to improve generalization under distribution shifts, and integrating BoardVision more tightly into manufacturing pipelines through headless QA modules, automated reporting, and human-in-the-loop inspection.

Ethics and Societal Impact. This work does not involve personal or sensitive data. Potential risks include overreliance on automated inspection, which could miss rare but critical defects if deployed without human oversight. We recommend human-in-the-loop verification to ensure safety and reliability in practical use.

Use of Large Language Models. Parts of the BoardVision software were prototyped with assistance from OpenAI's GPT-5 model. The LLM generated Python scaffolding for the graphical interface and the ensemble evaluation pipeline. All outputs were reviewed, debugged, and refactored by the authors to ensure correctness. Consistent with WACV policy, we take full responsibility for the final implementation.

9. Conclusion

This paper introduced BoardVision, an assembly-level motherboard defect detection system that combines YOLOv7 and Faster R-CNN through a lightweight, interpretable ensemble. The proposed Confidence–Temporal Voting (CTV) framework reduced failure modes on rare but high-cost classes and delivered more stable performance than either detector alone. Experiments on the MiracleFactory dataset confirmed that the ensemble not only improved F1 and precision but also maintained robustness under perturbations. In live inference, the ensemble consistently outperformed either single model: running both detectors in parallel reduced indecisiveness on borderline cases, stabilized frame-to-frame decisions under low light or clutter, and filtered spurious boxes via cross-verification.

Beyond metrics, a deployment-oriented GUI demonstrated how the ensemble's decision rules can be visualized and tuned in real time, bridging the gap between research benchmarks and practical motherboard inspection. We will publicly release the code, models, and the BoardVision software to support reproducibility in manufacturing QA and related applications.

References

[1] DB Anitha and Mahesh Rao. A survey on defect detection in bare pcb and assembled pcb using image processing techniques. In *2017 International conference on wireless communications, signal processing and networking (WiSPNET)*, pages 39–43. IEEE, 2017. 1, 2

[2] Enrique Dehaerne, Bappaditya Dey, Sandip Halder, and Stefan De Gendt. Optimizing yolov7 for semiconductor defect detection. page 77, 2023. 1, 2

[3] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 4

[4] Yuji Iwahori, Yohei Takada, Tokiko Shiina, Yoshinori Adachi, Manas Kamal Bhuyan, and Boonserm Kijsirikul. Defect classification of electronic board using dense sift and cnn. *Procedia Computer Science*, 126:1673–1682, 2018. 2

[5] Volkan Kaya and İsmail Akgül. Detection of defects in printed circuit boards with machine learning and deep learning algorithms. *Avrupa Bilim ve Teknoloji Dergisi*, (41):183–186, 2022. 2

[6] Ka Nam Canaan Law, Mingshuo Yu, Lianglei Zhang, Yiyi Zhang, Peng Xu, Jerry Gao, and Jun Liu. Enhancing printed circuit board defect detection through ensemble learning. In *2024 IEEE 1st International Workshop on Future Intelligent Technologies for Young Researchers (FITYR)*, pages 37–44. IEEE, 2024. 2

[7] Yu-Ting Li, Paul Kuo, and Jiun-In Guo. Automatic industry pcb board dip process defect detection system based on deep ensemble self-adaption method. *IEEE Transactions on Components, Packaging and Manufacturing Technology*, 11 (2):312–323, 2021. 2

[8] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 4

[9] Yang Liu, Bo Chen, and Qiang Wu. Multi-target detection of pcb defects based on improved ssd network with attention mechanisms. *International Journal of Computer and Electrical Engineering*, 8(6):319–325, 2022. 1

[10] Shengping Lv. Dspcbsd+. figshare, 2024. DOI: https://doi.org/10.6084/m9.figshare.24970329.v1. 2

[11] Shengping Lv, Bin Ouyang, Zhihua Deng, Tairan Liang, Shixin Jiang, Kaibin Zhang, Jianyu Chen, and Zhuohui Li. A dataset for deep learning based detection of printed circuit board surface defect. *Scientific Data*, 11(1):811, 2024. 1, 2

[12] Jianjie Ma. Defect detection and recognition of bare pcb based on computer vision. In *2017 36th Chinese Control Conference (CCC)*, pages 11023–11028. IEEE, 2017. 1, 2

[13] MA007. Hripcb dataset. `https://universe.roboflow.com/ma007/hripcb`, 2023. visited on 2025-09-20. 2

[14] Joon-Hyung Park, Yeong-Seok Kim, Hwi Seo, and Yeong-Jun Cho. Analysis of training deep learning models for pcb defect detection. *Sensors*, 23(5):2766, 2023. 1, 2

[15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015. 1, 2

[16] Guru Ratan Satsangee, Hamdan Al-Musaibeli, and Rafiq Ahmad. A defect detection method based on yolov7 for automated remanufacturing. *Applied Sciences*, 14(13):5503, 2024. 1, 2

[17] Minghui Shen, Yujie Liu, Jing Chen, Kangqi Ye, Heyuan Gao, Jie Che, Qingyang Wang, Hao He, Jian Liu, Yan Wang, et al. Defect detection of printed circuit board assembly based on yolov5. *Scientific Reports*, 14(1):19287, 2024. 1, 2

[18] Zhihao Tang, Jun Liu, Peng Zhao, Yifan Zhang, and Jie Li. An improved detection algorithm of pcb surface defects based on yolov5: Pcb-yolo. *Sustainability*, 15(7):5963, 2023. 1

[19] tangsanli5201. On-line pcb defect detector on a new pcb defect dataset. github, 2021. Available: `http://github.com/tangsanli5201/DeepPCB`. 2

[20] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 7464–7475, 2023. 1, 2

[21] Y. Xin et al. Computer motherboard production defects dataset. Kaggle, 2023. Available: `https://www.kaggle.com/datasets/yuelinxin/computer-motherboard-production-defects`, Accessed: Sep. 19, 2025. 1, 2, 3

[22] Yujie Yang and Haiyan Kang. An enhanced detection method of pcb defect based on improved yolov7. *Electronics*, 12(9):2120, 2023. 1, 2

[23] Yang Yang and Qin Qiang. An improved pcb defect detection algorithm for yolov7-tiny. In *International Conference on Mechatronic Engineering and Artificial Intelligence (MEAI 2023)*, pages 646–652. SPIE, 2024. 1

Algorithm 1: Confidence–Temporal Voting (CTV)

```
Input: YOLO detections $Y$, FRCNN detections $R$
       (each as boxes $b$, label $\ell$, confidence $p$);
IoU threshold $t_{IoU}$; confidence exponent $\gamma$;
per-class validation scores $F1_{YOLO,c}$, $F1_{FRCNN,c}$;
solo thresholds $conf\_thresh$, $solo\_strong$; near-tie
margin $f1\_margin$;
flag $fuse\_coords$ (average coordinates if true)
Output: Final fused detections $F$
$F \leftarrow \emptyset$ ;
Form candidate matches between $Y$ and $R$ with
  same label and $\mathrm{IoU}(b_Y, b_R) \geq t_{IoU}$ (greedy by
  IoU) ;
foreach matched pair $(y_i, r_j)$ do
    $S_Y \leftarrow (p_Y)^\gamma \cdot F1_{YOLO,\ell}$,
      $S_R \leftarrow (p_R)^\gamma \cdot F1_{FRCNN,\ell}$ ;
    if $fuse\_coords$ then
        $b^\star \leftarrow \frac{S_Y\, b_Y + S_R\, b_R}{S_Y + S_R}$;
    else
        $b^\star \leftarrow \arg\max_{b \in \{b_Y, b_R\}} \{S_Y, S_R\}$;
    $\ell^\star \leftarrow \ell$,   $p^\star \leftarrow \max(p_Y, p_R)$ ;
    Append $(b^\star, \ell^\star, p^\star)$ to $F$ ;
    Mark $y_i$ and $r_j$ as used ;
foreach unmatched detection $d = (b, \ell, p)$ in $Y \cup R$
  do
    let $m \in \{\text{YOLO}, \text{FRCNN}\}$ be the model of $d$,
      and $m'$ the other model ;
    if $p \geq solo\_strong$ then
        Append $d$ to $F$; continue
    if $F1_{m,\ell} > F1_{m',\ell}$ and $p \geq conf\_thresh$ then
        Append $d$ to $F$; continue
    if $|F1_{YOLO,\ell} - F1_{FRCNN,\ell}| \leq f1\_margin$ and
      $p \geq 0.95$ then
        Append $d$ to $F$; continue
Apply class-wise Non-Maximum Suppression to $F$;
  return $F$ ;
```

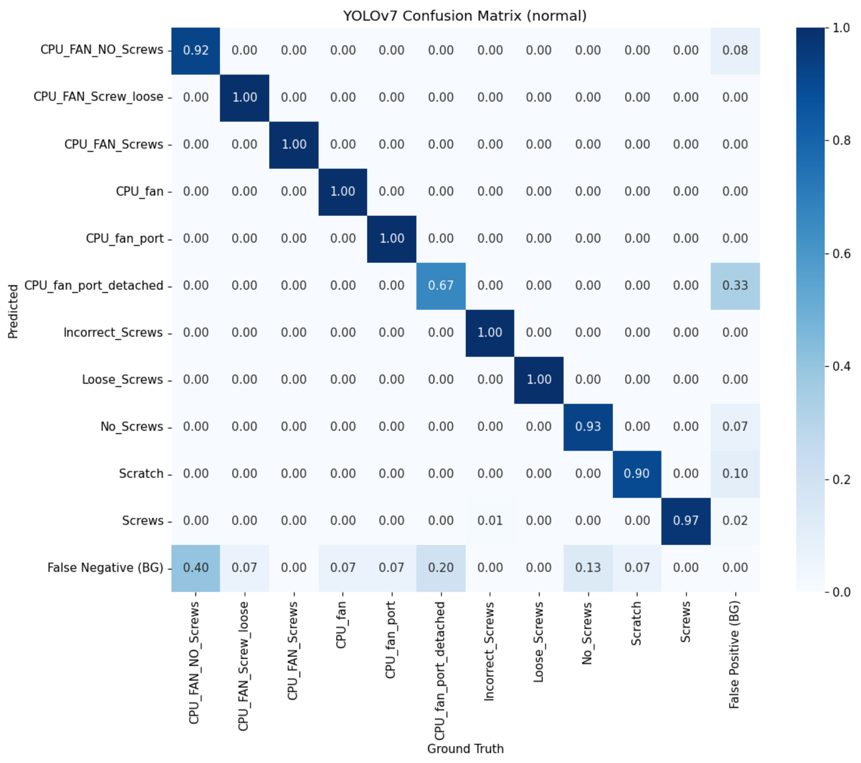
Example of Fusion

YOLOv7 predicts [100,100,200,200] with confidence 0.9, Faster R-CNN predicts [110,105,195,205] with confidence 0.8. After weighting with class-level F1 and $\gamma = 1.5$, YOLO's score is 0.75, FRCNN's 0.54.
Weighting the coordinates yields:

$$x_1^\star = \frac{0.75 \cdot 100 + 0.54 \cdot 110}{0.75 + 0.54} \approx 104.2, \quad y_1^\star \approx 102.0,$$

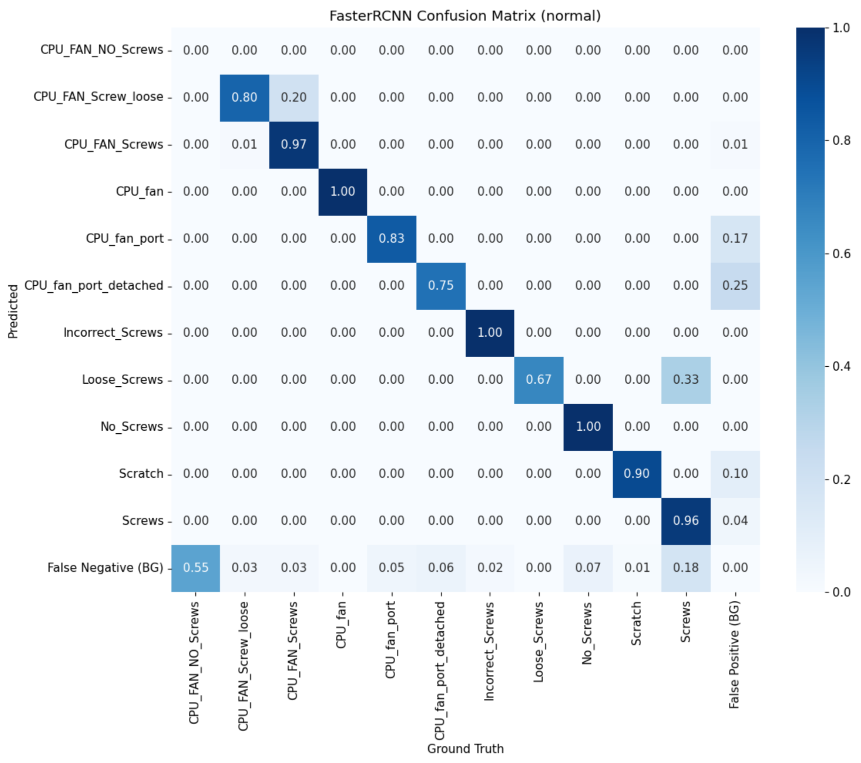
and similarly for (x_2, y_2). So The fused box lies at [104.2, 102.0, ...], closer to YOLO's prediction, reflecting its higher score, while still adjusting toward FRCNN's box.

Figure 6. CTV Fusion Example.



	CPU_FAN_NO_Screws	CPU_FAN_Screw_loose	CPU_FAN_Screws	CPU_fan	CPU_fan_port	CPU_fan_port_detached	Incorrect_Screws	Loose_Screws	No_Screws	Scratch	Screws	False Positive (BG)
CPU_FAN_NO_Screws	0.92	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.08
CPU_FAN_Screw_loose	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
CPU_FAN_Screws	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
CPU_fan	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
CPU_fan_port	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
CPU_fan_port_detached	0.00	0.00	0.00	0.00	0.00	0.67	0.00	0.00	0.00	0.00	0.00	0.33
Incorrect_Screws	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
Loose_Screws	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00
No_Screws	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.93	0.00	0.00	0.07
Scratch	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.90	0.00	0.10
Screws	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.97	0.02
False Negative (BG)	0.40	0.07	0.00	0.07	0.07	0.20	0.00	0.00	0.13	0.07	0.00	0.00

Figure 7. Normalized confusion matrix for YOLOv7 predictions.



	CPU_FAN_NO_Screws	CPU_FAN_Screw_loose	CPU_FAN_Screws	CPU_fan	CPU_fan_port	CPU_fan_port_detached	Incorrect_Screws	Loose_Screws	No_Screws	Scratch	Screws	False Positive (BG)
CPU_FAN_NO_Screws	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
CPU_FAN_Screw_loose	0.00	0.80	0.20	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
CPU_FAN_Screws	0.00	0.01	0.97	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01
CPU_fan	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
CPU_fan_port	0.00	0.00	0.00	0.00	0.83	0.00	0.00	0.00	0.00	0.00	0.00	0.17
CPU_fan_port_detached	0.00	0.00	0.00	0.00	0.00	0.75	0.00	0.00	0.00	0.00	0.00	0.25
Incorrect_Screws	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
Loose_Screws	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.67	0.00	0.00	0.33	0.00
No_Screws	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00
Scratch	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.90	0.00	0.10
Screws	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.96	0.04
False Negative (BG)	0.55	0.03	0.03	0.00	0.05	0.06	0.02	0.00	0.07	0.01	0.18	0.00

Figure 8. Confusion matrix for Faster R-CNN predictions.

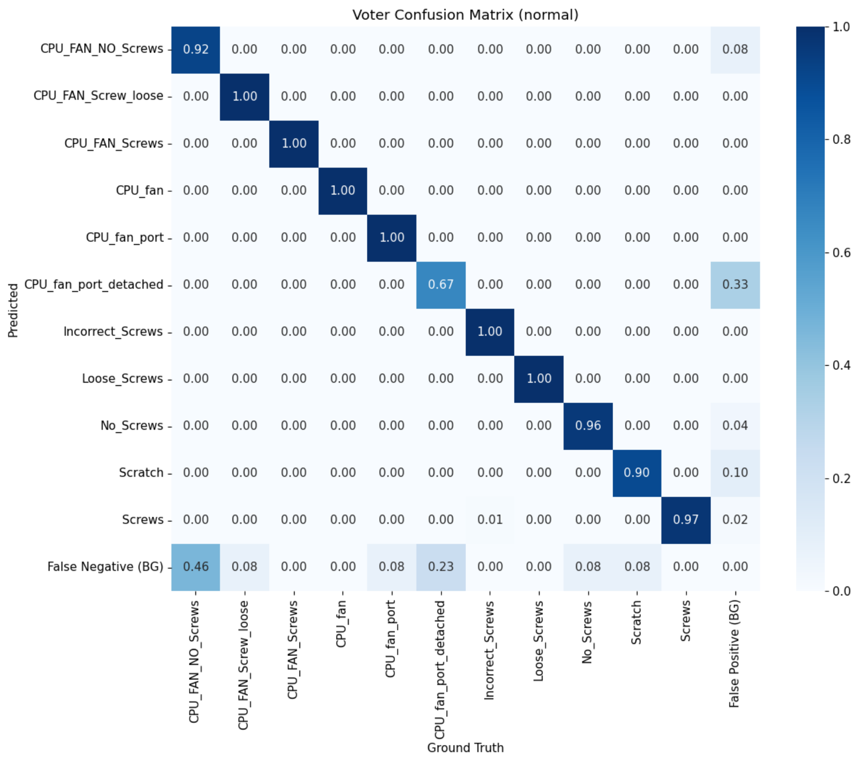
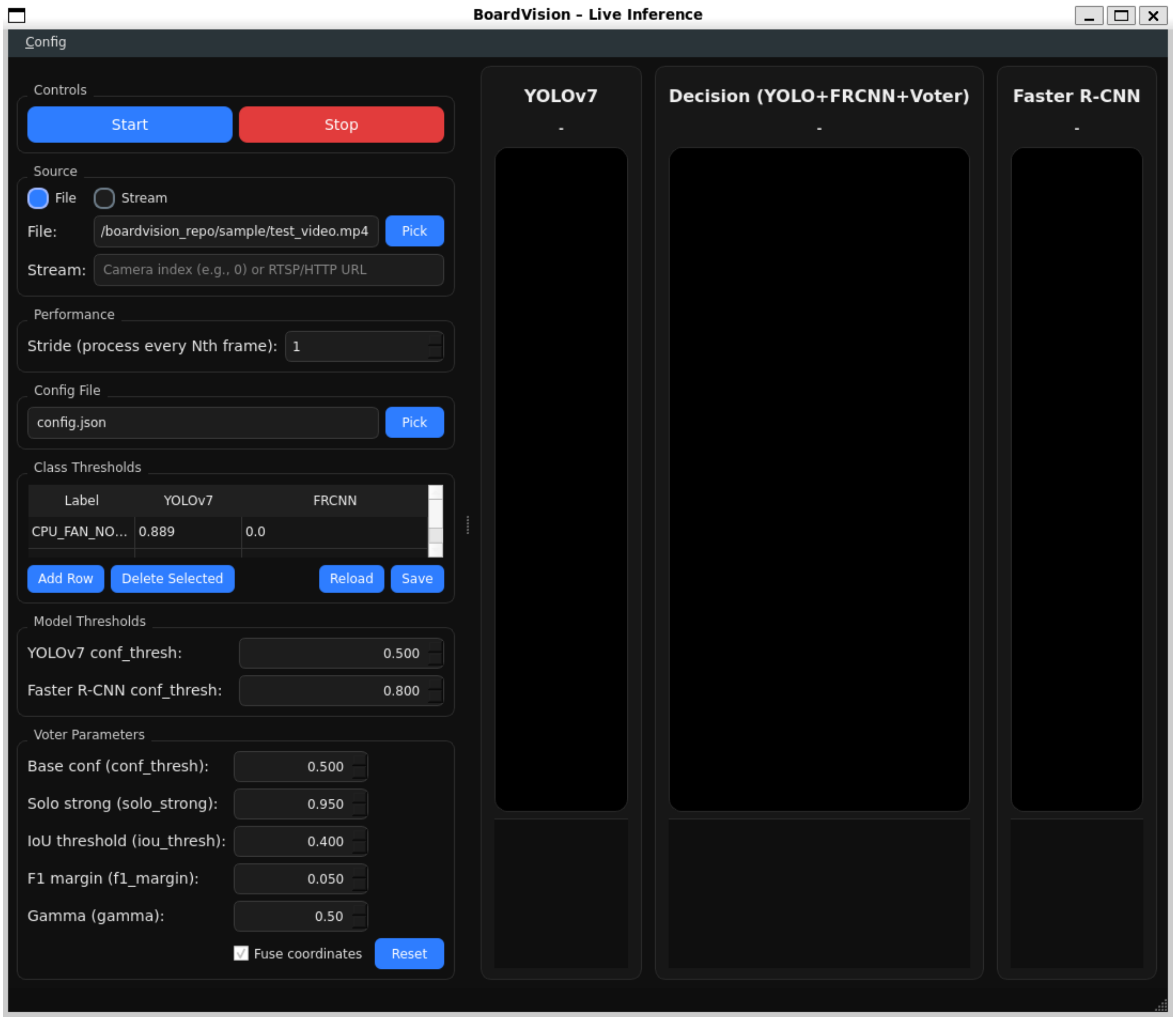


Figure 9. Confusion matrix for the proposed Voter Ensemble. Misclassifications are sparser, reflecting improved reliability on rare and ambiguous classes through ensemble fusion.


BoardVision - Live Inference

Config

Controls

Start Stop

Source

[✓ File] Stream

File: /boardvision_repo/sample/test_video.mp4 Pick

Stream: Camera index (e.g., 0) or RTSP/HTTP URL

Performance

Stride (process every Nth frame): 1

Config File

config.json Pick

Class Thresholds

Label	YOLOv7	FRCNN
CPU_FAN_NO...	0.889	0.0

Add Row Delete Selected Reload Save

Model Thresholds

YOLOv7 conf_thresh: 0.500

Faster R-CNN conf_thresh: 0.800

Voter Parameters

Base conf (conf_thresh): 0.500

Solo strong (solo_strong): 0.950

IoU threshold (iou_thresh): 0.400

F1 margin (f1_margin): 0.050

Gamma (gamma): 0.50

[✓] Fuse coordinates Reset

YOLOv7 -

Decision (YOLO+FRCNN+Voter) -

Faster R-CNN -

Figure 10. BoardVision GUI prior to selecting an input source, shown with synchronized YOLOv7, Faster R-CNN, and voter ensemble views.