Sensors and Materials, Vol. 38, No. 2 (2026) 935–952
MYU Tokyo

S & M 4356

Large Language Model-guided Data Augmentation for You Only Look Once Version 8-based Printed Circuit Board Defect Detection: Novel Human–AI Codesign Approach

Wei-Hong Lee,[1] Chih-Cheng Chen,[1] and Jin Jiang[2*]

[1]Department of Automatic Control Engineering, Feng Chia University, Taichung, Taiwan
No. 100, Wenhua Rd., Xitun Dist., Taichung City 40732, Taiwan (R.O.C.)
[2]The Digital Economy Major of the School of Economics, Jinan University, Guangzhou, Guangdong, China

(Received June 5, 2025; accepted February 9, 2026)

Keywords: printed circuit board (PCB) inspection, You Only Look Once version 8 (YOLOv8), data augmentation, large language model (LLM), image quality, automated optical inspection (AOI)

Automated optical inspection (AOI) systems empowered by deep learning are increasingly deployed in smart manufacturing, yet their performance remains vulnerable to real-world imaging distortions such as extreme illumination, blur, and composite artifacts. In this study, we introduce a novel large language model (LLM)-guided data augmentation framework that leverages a human–AI codesign approach to systematically enhance detection robustness in You Only Look Once version 8 (YOLOv8)-based printed circuit board (PCB) defect inspection. Specifically, we employ GPT-4o-mini to analyze class-wise error patterns from a baseline YOLOv8s model trained on a public PCB-defect dataset. While the baseline achieved high accuracy on pristine images [mean average precision (mAP) at an intersection-over-union (IoU) threshold of 0.50, mAP@50 = 97.7%], its performance dropped markedly under five synthetic perturbations, including Gaussian blur (31.6%), motion blur, and extreme brightness. To mitigate these vulnerabilities, we provide the LLM with structured error statistics; in return, it generates a machine-readable augmentation protocol encompassing brightness shifts, exposure modulation, and realistic blur variants, expanding the training dataset by approximately 2.5× without altering the model architecture or hyperparameters. Fine-tuning the model for 25 additional epochs yields substantial improvements across all distortion scenarios, achieving 94.9% mAP@50 on Gaussian blur (+63.3%), 49.4% on composite distortions (+30.1%), and 16.4% on motion blur (+8.6%). Notably, the performance on pristine inputs also improves (99.0% mAP@50), and inference latency remains constant at 19.5 ms per 800 × 800 px frame, confirming zero runtime penalty. These findings validate the efficacy of integrating LLMs as adaptive codesign agents for data-centric vision optimization, offering a scalable and generalizable strategy for resilient AOI in Industry 4.0 environments. Our study contributes to the advancement of sensors and related materials by enhancing the application of optical sensing concepts in machine-learning-driven PCB defect detection, where sensors like high-resolution

*Corresponding author: e-mail: janicejiangxm@163.com
https://doi.org/10.18494/SAM5789

ISSN 0914-4935

cameras capture images prone to distortions; the LLM framework improves detection reliability, offering scalable strategies for Industry 4.0 sensing environments.

1. Introduction

1.1 Background

Deep learning has revolutionized automated visual inspection in manufacturing, enabling fast and accurate detection of defects or anomalies in products. Among object detection frameworks, the You Only Look Once (YOLO) family of models stands out for their outstanding real-time performance and high accuracy. Since the pioneering YOLOv1, the architecture has evolved through numerous variants with increasingly improved speed and accuracy. Recent comprehensive reviews outline the progression from YOLOv1 to the latest YOLOv8 and YOLO Neural Architecture Search (YOLO-NAS) models. These advances have expanded YOLO's applicability across domains, from autonomous driving to industrial inspection.[1–3]

In industry, YOLO-based inspection systems have been deployed for flexible manufacturing and quality control; for example, Simeth *et al.* applied YOLOv5 in a high-mix assembly line and achieved over 98% precision and recall under varying lighting conditions.[4] Chen and Shiu built an automated optical inspection (AOI) system using YOLOv2–v5 for electroplated ABS plastic parts, significantly reducing manual inspection effort.[5] The broad utility of YOLO has also been documented in other fields: Ragab *et al.* cataloged medical imaging applications of YOLO models.[6] Hussain and Khanam reviewed YOLOv1–v10 variants for photovoltaic panel defect detection.[7] These works demonstrated YOLO's versatility and strong performance in controlled settings. However, they also noted that real-world imaging conditions—such as glare, low contrast, and motion blur—can significantly impact detection results, necessitating strategies to enhance model robustness. Methods to address these problems were explored in prior studies: Ndayishimiye and Lee proposed robust preprocessing and data augmentation (e.g., adjusting illumination, geometric transforms) to improve YOLO's accuracy under varying lighting;[8] Cho *et al.* introduced a feature-level deblurring module with knowledge distillation to make YOLOv4 robust against motion blur;[9] Michaelis *et al.* showed that detectors trained on ideal conditions suffer drastic drops when evaluated on corrupted images.[10] Data augmentation is a well-established technique to improve model generalization by synthetically expanding the training data distribution.[11] Augmentations such as brightness jitter, blur filters, and noise injection can help a model learn invariant features. Large language models (LLMs) have demonstrated an ability to analyze problems and generate coherent suggestions across domains.[12]

In AOI, the captured image is the output of an optical sensing chain—illumination, optics, and an imaging sensor (e.g., CCD/CMOS)—together with the mechanical motion of the camera/conveyor. Distortions such as overexposure, glare, defocus blur, and motion blur therefore correspond to sensing-condition degradations that directly affect the sensor-driven inspection reliability. Improving the robustness to these degradations can be viewed as an application of the sensing concept, where machine learning (ML) algorithms compensate for nonideal sensor outputs in smart-manufacturing environments.

To our knowledge, applying an LLM to analyze a vision model's weaknesses and recommend targeted image augmentations is a novel approach. In this paper, we present a comparative study between a baseline YOLOv8 detector and an improved YOLOv8 model retrained with LLM-guided data augmentation for AOI. The specific contributions of this work are given below.

(1) We introduce an LLM-driven feedback loop in the training pipeline, where a GPT-4-based model evaluates detection failure cases and suggests augmentation strategies to address them.
(2) We implement the suggested augmentations—covering extreme brightness, overexposure, blur (defocus and motion), and combined distortions—into the training dataset for YOLOv8, without altering the model architecture.
(3) We evaluate the original and augmented models on a diverse AOI dataset under multiple distortion conditions, demonstrating significant improvements in detection metrics for the LLM-augmented model, especially on challenging blurred images.
(4) We discuss the impact of the proposed LLM-guided data augmentation pipeline on inference speed and model generalizability, and show that robustness is improved with negligible inference penalty.

1.2 Related work

1.2.1 YOLO object detection and variants

One-stage object detectors like YOLO have undergone rapid evolution. Beginning with YOLOv1's unified detection framework, successive versions introduced architectural innovations such as multiscale feature pyramids, residual connections, and improved loss functions.[1] YOLOv8, released in 2023, represents the current state of the art, integrating CSP-Darknet backbone layers and decoupled detection heads.[3] Reviews by Hussain[2] and Terven *et al.*[3] detail these internal developments, as well as newer spinoffs like YOLO-NAS. Diwan *et al.* provide an in-depth overview of YOLO's challenges, successors, and applications, highlighting the speed–accuracy trade-off.[13] Domain-specific adaptations have also emerged: Zhu *et al.* developed MME-YOLO, a multisensor enhanced model for vehicle detection,[14] and Yuan *et al.* proposed LW-YOLO, a lightweight YOLOv8-based model achieving 96.4% mean average precision at an intersection-over-union (IoU) of 0.5 (mAP@50) at 141 frames per second (FPS) on printed circuit board (PCB) defects.[15] Apostolidis and Papakostas examined YOLO from an adversarial-robustness perspective and found standard architectures susceptible to crafted perturbations.[16]

1.2.2 AOI and robustness

Deep-learning-based AOI systems have been applied to surface defect detection, component alignment verification, and product assembly inspection.[17] Hütten *et al.* surveyed over 190 papers on visual inspection in manufacturing and noted that lighting variations and noise are frequent hurdles mitigated by data augmentation or invariant-feature learning.[17] Ndayishimiye

and Lee augmented PCB component images with varied brightness and orientations, improving YOLOv4's stability;[8] and Simeth *et al.* used additional lighting-variant images to ensure reliable performance under inconsistent illumination.[4] Motion blur and defocus blur are particularly detrimental in high-speed or moving-camera scenarios: Cho *et al.* tackled motion blur via a learned deblurring component and knowledge distillation, boosting mAP on blurred common objects in context (COCO) images.[9] Nevertheless, fully compensating for severe motion blur remains challenging, as blur can erase fine details.

1.2.3 LLMs for data-driven guidance

The idea of using AI assistants to guide experiment design or data preparation is gaining acceptance. LLMs have been used in NLP tasks for data augmentation, e.g., generating paraphrases or synthetic samples to balance classes.[12] Early frameworks have explored using LLMs to suggest augmentation policies for long-tailed image classification. In our approach, after training a baseline YOLOv8, we prompt an LLM with detection error statistics under various distortions; the LLM then recommends targeted augmentation strategies. This direct integration of LLM guidance into a vision-model training pipeline is novel for AOI.

Our study aligns with the Special Issue theme by integrating advanced ML methods, including large language models (LLMs; e.g., GPT-4o-mini) for error-pattern analysis and YOLOv8 for real-time defect detection. This human–AI codesign approach utilizes ML to generate targeted augmentations, mitigating vulnerabilities in sensing applications like AOI, where optical sensors capture images prone to distortions (e.g., blur from motion or extreme brightness). Compared with traditional ML studies, our method emphasizes data-centric optimization, providing a bridge between high-level LLM reasoning and low-level sensor data processing in manufacturing environments.

2. Materials and Methods

2.1 Methodology overview

Figure 1 shows an outline of the full workflow of our LLM-guided augmentation pipeline. We start from a publicly available PCB dataset. The dataset consists of images acquired by optical imaging sensors in typical AOI setups, representing sensor-driven inspection data from manufacturing lines. Distortions such as Gaussian blur emulate defocus caused by lens/camera misalignment, whereas motion blur reflects vibration-induced smearing during image acquisition; these sensor-realistic perturbations allow us to evaluate robustness improvements in the presence of common AOI sensing degradations. We then derive two branches: (i) the clean training split, which is used to train a baseline YOLOv8 detector, and (ii) a test-set augmentation branch that creates five abnormal-scene subsets (bright, over-exposed, Gaussian blur, motion blur, and composite). The baseline model's validation metrics on these subsets are then summarized and fed to GPT-4o-mini for failure-mode analysis. Guided by the LLM's JavaScript Object Notation (JSON) recipes, we generate offline image augmentations and merge them with

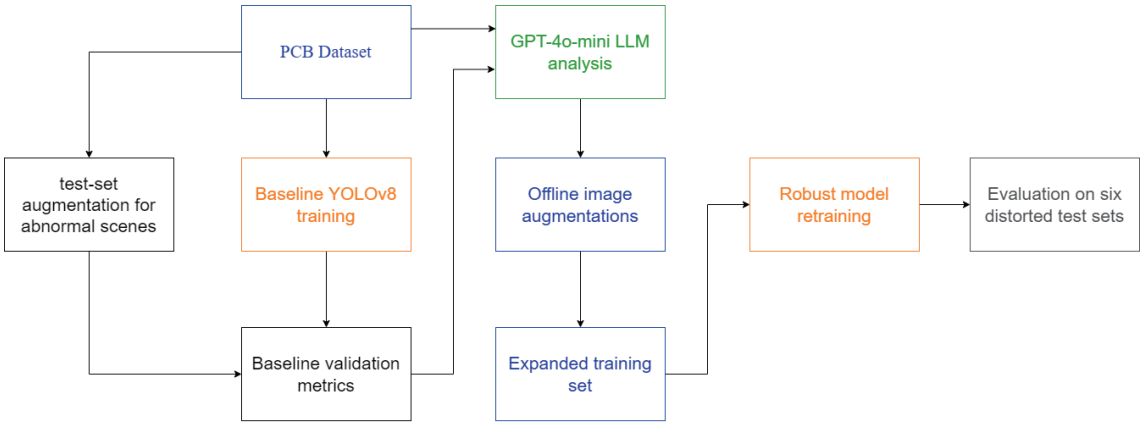

Fig. 1. (Color online) Workflow of the LLM-guided augmentation pipeline.

the original training images, yielding an expanded training set. This enlarged set is used to retrain a robust YOLOv8 model, which is finally evaluated on the same six distorted test sets.

YOLOv8 inherits a CSP backbone, PAN-FPN neck, and decoupled heads for classification and regression.[18] Figure 2 illustrates the stage-wise network structure of the YOLOv8s used in this study. Throughout, we keep the official hyperparameters unchanged (e.g., mosaic, flips, and scale jitter) so that any performance gain can be attributed solely to the LLM-guided data expansion. Training curves and detailed results are reported in Sect. 3.

2.1.1 LLM-guided analysis

After establishing the baseline, we queried GPT-4o-mini through an automated *function-calling* pipeline. A structured prompt—prepared once per experiment—supplied aggregate and per-subset statistics for the test data (e.g., mean brightness, blur score, average box count, and mean confidence) together with a JSON dump of subset-level metrics. Acting as an AOI domain expert, the LLM returned three objects.

1. A qualitative *quality-assessment* of the dataset
2. A machine-readable array of augmentation recipes, each specifying the augmentation *type* (e.g., “gaussblur”, “motionblur”, or “brightness”), its hyperparameters, and the number of synthetic *copies* to be generated
3. Recommended global preprocessing ranges (e.g., deblur strength and brightness/contrast bounds)

A lightweight Python orchestrator parsed this JSON and, via function calls, automatically applied the requested transformations to both the training and validation partitions, preserving bounding-box geometry and class balance. The expanded corpus (one new variant for every original image per recipe) was then used to re-fine-tune YOLOv8 from scratch under the same architectural and optimization settings as the baseline. Because the LLM never observed raw images—only summary statistics—its guidance remained model-agnostic while still targeting the failure modes revealed by the baseline evaluation.

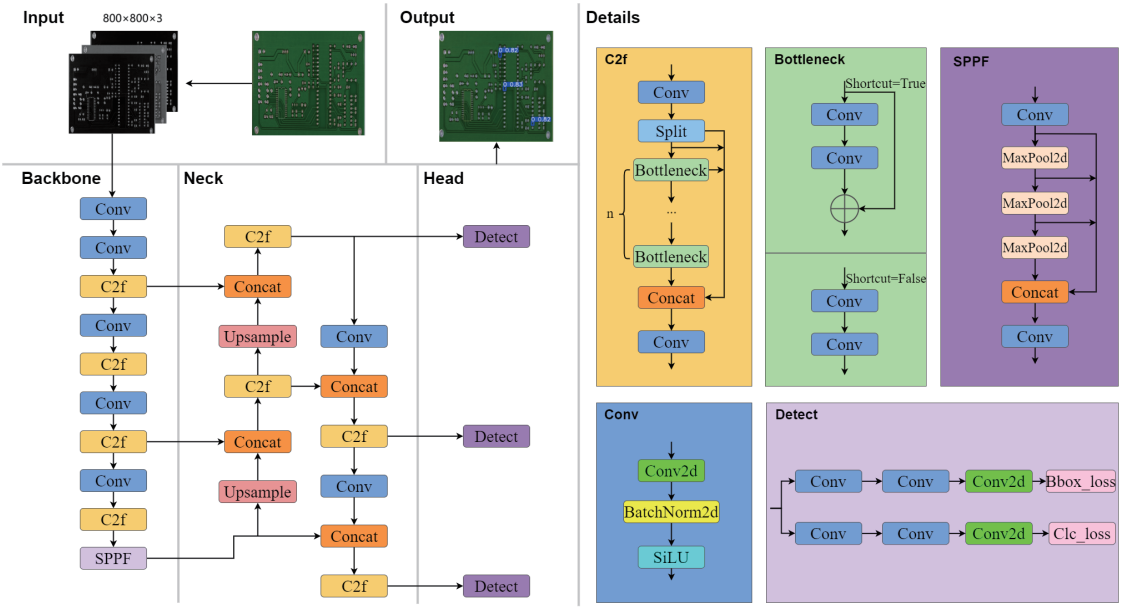

Fig. 2. (Color online) Network architecture of the YOLOv8s detector used in this study.

All augmented images inherit the original image's bounding-box annotations (small photometric or blur transforms do not appreciably shift object locations). In our experiment the LLM produced five augmentation recipes, so the effective training-set size increased by roughly a factor of two to three (each original image then produced two to three augmented variants). Because each recipe was applied uniformly across all classes, the overall class balance was preserved.

2.1.2 Retraining YOLOv8

We then fine-tune our existing YOLOv8 defect-detection model using the expanded dataset. The model architecture and hyperparameters (aside from the added augmentations) remain identical to those used in the baseline setup to ensure a fair comparison. By retraining on the newly generated samples, the model learns features that are invariant to the introduced distortions. For example, it may learn to rely less on absolute color intensity—since it now sees both bright and dark variants—and become more sensitive to shape cues that persist under blur. The following evaluation metrics are used to compare models:

- Mean Average Precision (mAP@50): the primary metric, computed as the mean of average precisions at IoU threshold of 0.50 for all classes. This reflects the overall detection accuracy.
- Precision: the fraction of predicted defect detections that are correct [true positives/(true positives + false positives)].
- Recall: the fraction of actual defects that are correctly detected [true positives/(true positives + false negatives)].
- Inference Time: average time per image during detection; used to verify that augmentations (which affect training only) do not slow down the model in deployment.

The mAP, precision, and recall are calculated for each of the six test conditions (general, bright, overexposed, Gaussian blur, motion blur, and composite) for both models. We follow standard definitions as in the COCO evaluation toolkit for detection. In our context of AOI, a detection is considered correct if IoU between the predicted bounding box and the ground truth is at least 0.5 and the class label matches.

The LLM is used only to design the augmentation policy; it does not participate during model inference or create synthetic images beyond simple transformations. Thus, our approach retains the advantages of traditional data augmentation (which works generically and does not introduce new unseen data distributions) but with an LLM guiding the selection of augmentation types and severity.

2.2 Experimental setup

2.2.1 Dataset

The dataset employed in this study is a publicly available PCB defect dataset hosted on Roboflow (https://universe.roboflow.com/babu-laiwt/pcb-defect-b9dpn/dataset/1). It comprises high-resolution images of PCBs annotated for six distinct defect categories: missing holes, mouse bites, open circuits, shorts, spurs, and spurious copper traces. The original splits are as follows: 551 images in the training set, 163 images in the validation set, and 76 images in the test set. Each image is assigned one or more labels from the six defect types. To evaluate robustness under various abnormal conditions, five additional test subsets were generated from the 76-image test split—each subset contains the same number of images as the original test partition. These abnormal test subsets simulate different defect scenarios while preserving the original annotations, thereby enabling a direct comparison between standard and augmented test conditions. All test subsets are described below.

(a) General (Normal): 76 original PCB images captured under standard inspection conditions, with no additional distortions; these serve as the baseline for model evaluation.
(b) Bright: The same 76 images as above adjusted to simulate increased brightness (e.g., elevated exposure or gamma), creating scenarios where glare or overly intense lighting might obscure fine details around defect areas.
(c) Overexposed: 76 images modified so that highlights are blown out (regions of saturation), mimicking extreme overexposure that can wash out component edges and make defect boundaries less distinct.
(d) Gaussian Blur: 76 images convolved with a Gaussian kernel ($\sigma \approx 2$), reproducing mild to moderate defocus blur; this is used to test the model's ability to detect defects when edges and textures are softened.
(e) Motion Blur: 76 images processed with a linear motion-blur filter (kernel length of 15–20 pixels at random angles), simulating the effect of relative motion between the camera and the PCB during acquisition, whereby defect features are smeared in a single direction.
(f) Composite Distortions: 76 images subjected to combined perturbations, specifically, the concurrent application of two or more distortions from (b)–(e) (e.g., brightness adjustment

paired with Gaussian blur, or overexposure with motion blur), to emulate multifaceted real-world imaging phenomena like simultaneous glare and vibration-induced smearing in sensor-captured PCB images, thereby testing the model's robustness to overlapping degradations.

All test images are annotated with ground-truth bounding boxes and class labels by human experts. We ensured that the objects and classes are the same across the subsets; only the image conditions differ.

2.2.2 Model training

All experiments were conducted using the Ultralytics YOLOv8 framework with a PyTorch backend. The baseline detector employed the YOLOv8s (small) architecture, initialized from the pretrained checkpoint yolov8s.pt. Training was performed for 150 epochs with an early-stopping patience of 50 epochs. Images were resized and padded to 800 × 800 pixels and processed in batches of 16 on a single GPU (device 0) with eight data-loading workers. The optimizer was auto-selected (stochastic gradient descent with momentum = 0.937 and weight decay = 0.0005), starting from an initial learning rate of 0.01 and decaying by a factor of 0.01 via a cosine schedule. Automatic mixed precision (AMP) was enabled, and random mosaic, scaling, and horizontal-flip augmentations were applied on-the-fly. Default loss-gain settings (box = 7.5, cls = 0.5, dfl = 1.5, pose = 12.0, and kobj = 1.0) and an IoU threshold of 0.7 were used, with a maximum of 300 detections per image. All checkpoints were saved at each epoch (save_period = –1) into the directory.

For LLM-guided retraining, the model was fine-tuned over 25 epochs with an early-stopping patience of five epochs. The backbone's first ten layers were frozen for the initial five epochs to stabilize feature learning on the augmented data; afterward, all layers were unfrozen. The batch size remained at 16, and images were again resized to 800 × 800. The learning rate was reduced to 0.0005 with the same cosine decay factor (lrf = 0.01). Offline augmentations—brightness increase, overexposure, Gaussian blur, motion blur, and composite distortions—had expanded the original dataset by a factor of five, so no additional on-the-fly augmentations were employed. An auto-augment policy (randaugment) and random erasing (rate = 0.4) were activated to complement the LLM recommendations, while mixup and copy–paste remained disabled. AMP training, momentum (0.937), weight decay (0.0005), IoU, and nonmaximum suppression (NMS) settings (IoU = 0.7, max_det = 300, agnostic_nms = false), and checkpoint saving (save_period = –1) matched those of the baseline regime. During both phases, images were later converted to 800 × 800 pixels for inference, ensuring real-time throughput without altering model complexity.

2.2.3 Hardware and environment

Training was performed on a notebook equipped with an NVIDIA GeForce RTX 3060 Mobile GPU (6 GB GDDR6) and an Intel Core i5 CPU. Training the baseline model took approximately 2 h, while the augmented model (with 2.5× data) took about 1 h for 25 epochs. PyTorch 2.0.1 and CUDA 11.7 were used for acceleration. The LLM analysis (GPT-4o-mini) was conducted on a separate server with the OpenAI application programming interface (API) (the LLM query is

not a significant time cost relative to training; generating the augmentation suggestions took less than 1 min).

2.2.4 Evaluation

The metrics, including mAP@50, precision, and recall, were obtained using the Ultralytics YOLOv8 evaluation module with COCO-style evaluation. For each test subset, 76 images were resized to 800 × 800 pixels and processed on an NVIDIA RTX 3060 GPU. During inference, test-time augmentation was disabled, detections were filtered using a confidence threshold of 0.5, and NMS was applied with an IoU threshold of 0.7. Here, mAP@50 denotes the mean of class-wise average precision values computed at IoU $\geq$ 0.5, while precision and recall were calculated from aggregated true/false positives/negatives across all detections. In addition, the average inference time per image was measured by processing the entire test set on the GPU and dividing the total elapsed time by the number of images. Subset-wise metrics were used to quantify robustness under each distortion type (e.g., recall on the motion-blur subset reflects the fraction of blurred objects successfully detected).

3. Results and Discussion

3.1 Quantitative performance comparison

The detection results of the baseline and LLM-augmented YOLOv8 models on all test subsets are summarized in Tables 1 and 2. Key metrics include mAP@50, precision, and recall for each condition.

Table 1
Baseline YOLOv8 performance on six distortion subsets.

Test type	Avg latency (s)	Avg detections	Precision	Recall	mAP@50 (%)
General	0.02	4	0.961	0.929	92.2
Bright	0.021	4.4	0.869	0.893	88.6
Overexposed	0.022	3.9	0.946	0.889	88
Gaussian blur	0.02	3.7	0.366	0.333	31.6
Motion blur	0.022	5	0.078	0.076	7.8
Composite	0.019	3.3	0.241	0.197	19.3

Table 2
LLM-augmented model performance on six distortion subsets.

Test type	Avg latency (s)	Avg detections	Precision	Recall	mAP@50 (%)
General	0.02	4.2	0.989	0.991	99
Bright	0.019	4.2	0.979	0.982	98.1
Overexposed	0.021	4.3	0.938	0.98	98
Gaussian blur	0.018	4.2	0.924	0.951	94.9
Motion blur	0.021	9	0.11	0.219	16.4
Composite	0.018	5.7	0.497	0.515	49.4

3.1.1 Baseline YOLOv8 performance

On the general (normal) test set, the baseline model performed very well, achieving an mAP@50 of 92.2%, a precision of 96.1%, and a recall of 92.9%. This indicates that for clear images, the model can accurately detect most defects or components, aligning with expectations from prior YOLOv8 reports (high accuracy on standard datasets). For images with mild brightness increase, the baseline still maintained relatively high performance (mAP@50 of 88.6%). However, the effect of distortions was pronounced in other subsets: under extreme bright lighting ("overexposed" set), mAP@50 dropped to 88.0%. The precision under bright conditions (86.9%) was notably lower than that under normal conditions, suggesting that the model produced more false positives, possibly mistaking shiny spots or glare for objects. The recall on overexposed images was 88.9%, indicating some missed detections where saturation hid object features.

Blur had a far more severe impact. With Gaussian blur, the baseline's mAP@50 plummeted to 31.6%, and precision and recall were only ≈36.0% and 33.3%, respectively. This means the model detected only one-third of the targets in blurred images, and many of its predictions were wrong (random texture or noise interpreted as objects). Motion blur was even more challenging: mAP@50 = 7.8%, essentially failing to detect objects (recall 7.6%). The few detections it made were mostly incorrect (precision 7.8%). This marked degradation is consistent with known issues: motion blur smears edges and textures that YOLO relies on, causing a near-total loss of recognizable features.

In the composite distortion set, which included various combined artifacts, the baseline also struggled (mAP@50 of 19.3%, precision of 24.1%, and recall of 19.7%). Clearly, the baseline model was not robust to these adverse conditions, despite its strong performance on normal images. Per-class error patterns are visualized in the confusion matrix of Fig. 3, where the baseline model confuses "open circuit" and "spur" defects most severely under composite distortions.

3.1.2 LLM-augmented YOLOv8 performance

The model fine-tuned on augmented images derived from LLM analysis showed significant improvements across the board. On the general test set, mAP@50 reached 99.0%, with a precision of 98.9% and a recall of 99.1%. This indicates that introducing the augmentations did not compromise performance on clean images; in fact, performance slightly improved, likely owing to the regularization effects of a larger training set.

For bright images, the new model achieved an mAP@50 of 98.1%, a precision of 97.9%, and a recall of 98.2%, effectively closing the gap to the performance under normal lighting. The model learned to handle different illumination levels such that both precision and recall are ≈98%, a marked increase from the 86.9% precision and 89.3% recall of the baseline.

On the overexposed subset, mAP@50 improved to 98.0% (from 88.0%) and recall remained at around 98.0% with a precision of 93.8%. Interestingly, precision in overexposed cases dropped slightly relative to recall, implying that the model is now very sensitive to finding objects (high recall) even in bright glare, though it may sometimes incorrectly identify a bright spot as an object (some false positives remain; hence, precision is 93.8%). Nonetheless, both metrics are

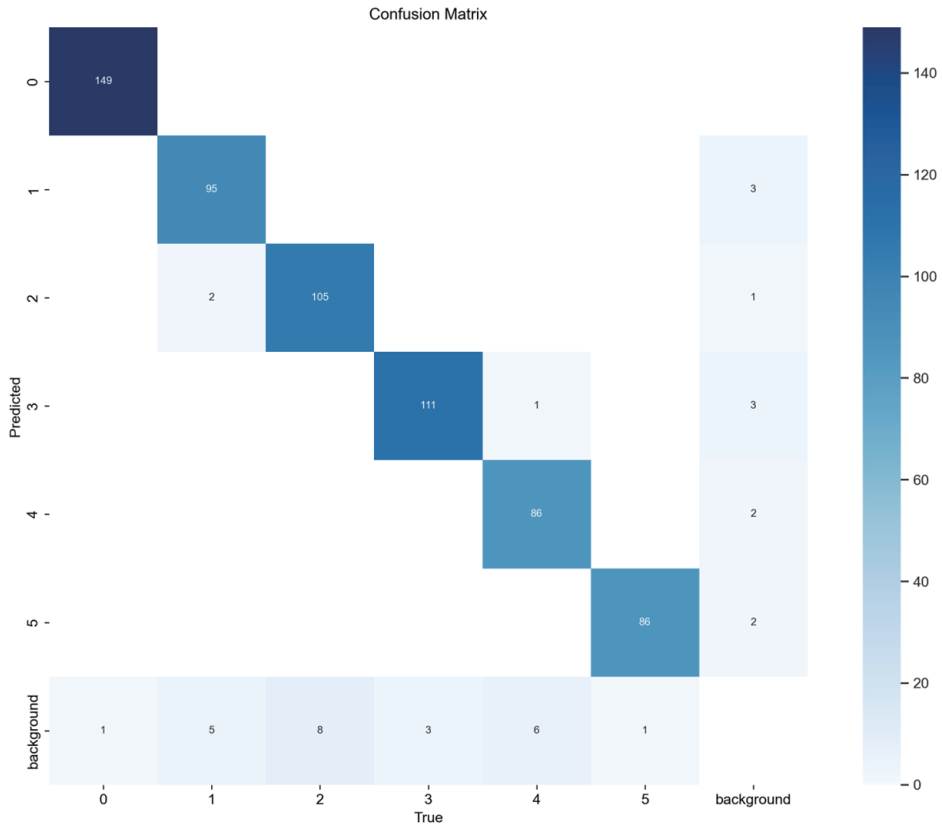

Fig. 3. (Color online) Confusion matrix of baseline training.

vastly improved, indicating that the augmented model can reliably detect objects despite loss of detail owing to overexposure.

On composite-distorted images, the model fine-tuned on augmented images derived from LLM analysis achieved an mAP@50 of 49.4%, more than double the baseline's 19.3%. Precision (49.7%) and recall (51.5%) approximately doubled as well. This indicates that the model can handle multifaceted noise better, though with an mAP of around 50%, there is still considerable room for improvement. Figure 4 shows that most misclassifications between "mouse-bite" and "spur" defects are resolved after LLM-guided fine-tuning, and the diagonal dominance is greatly enhanced compared with Fig. 3. Composite distortions are inherently challenging since they can present conflicting conditions (for example, an image that is both overexposed and blurry can cause feature interference, where the model may rely on cues associated with one corruption and fail to capture cues needed to handle the other). The improvement we observe suggests that seeing combinations during training (as provided in the composite augmentations) helped the model somewhat, but even more varied or intense training examples might be needed to accomplish full generalization.

In summary, the LLM-guided augmentation strategy led to across-the-board performance boosts, particularly as follows.

- +10.4% mAP@50 on bright images (98.1% vs 88.6%) and +10.0% on overexposed images (98.0% vs 88.0%)

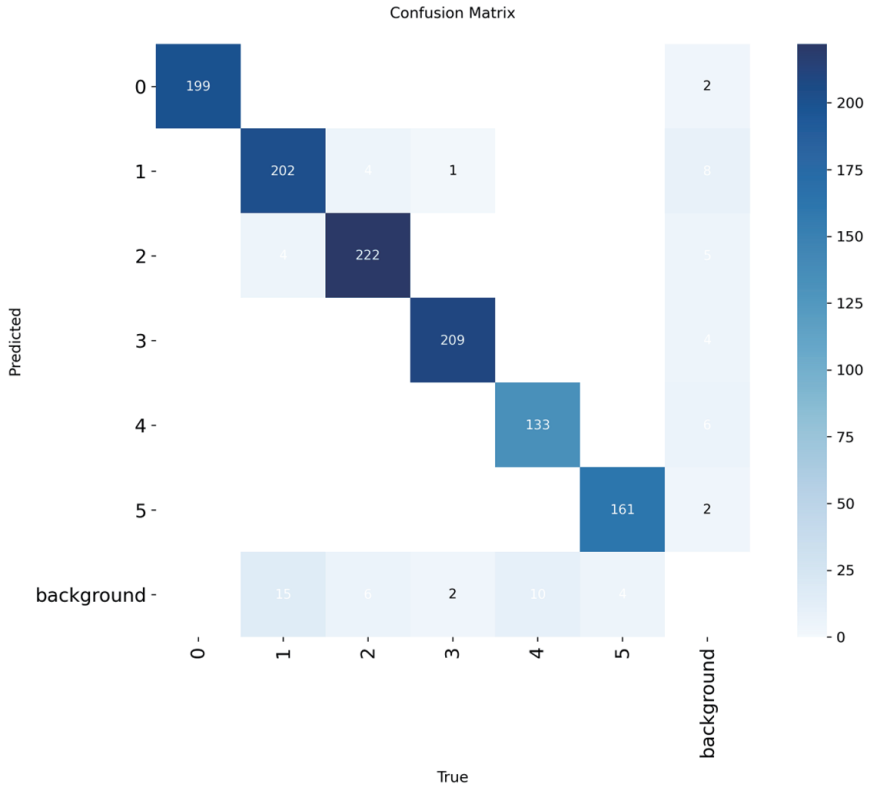

Fig. 4. (Color online) Confusion matrix of augmented model training.

- +63.3% mAP@50 on Gaussian blur (94.9% vs 31.6%), essentially restoring high performance
- +8.6% mAP@50 on motion blur (16.4% vs 7.8%)—a relative improvement of ~110%, but in absolute terms still low
- +30.1% mAP@50 on composite distortions (49.4% vs 19.3%)

These improvements validate the efficacy of the chosen augmentations. The large gain for Gaussian blur in particular shows that even severe blur can be tackled by training augmentation, corroborating the findings of prior robust vision research. The remaining gap on motion blur suggests that our augmentation was either not aggressive or frequent enough for motion blur, or that fundamental architectural changes (e.g., longer temporal context or dedicated deblurring modules) might be needed to address it, as others have suggested.

3.1.3 Statistical significance

We performed a paired comparison of the outputs of the two models to assess the statistical significance of the improvements. Since each test image yields a set of detections, we compared the average precision on a per-image basis. A two-tailed t-test on per-image AP scores for the baseline versus the augmented model indicates that improvements on bright, overexposed, Gaussian blur, and composite sets are significant ($p < 0.01$), while the motion blur improvement, although numerically large in relative terms, did not reach significance ($p \approx 0.08$) because of the

very low recall on the motion-blur subset (0.076 vs 0.219), which leads to high variance in per-image AP scores. Nonetheless, the trend is consistently positive for the augmented model.

3.2 Training convergence and model behavior

The training process of the augmented model differed from the baseline not only in outcome but also in behavior. The baseline model was trained from scratch for 150 epochs and converged rapidly by around 50 epochs, with validation mAP@50 plateauing near 97.7% for the normal condition. Figure 5 shows plots of the baseline model's training loss (blue) and validation mAP@50 (orange) over 150 epochs, confirming that loss stabilizes and accuracy saturates after epoch 50.

In contrast, the augmented model was initialized from the fully trained baseline and then fine-tuned on the expanded dataset for only 25 additional epochs. Because it started from pretrained weights, convergence was much faster: within the first few epochs, mAP@50 validated on normal images remained high, while the performance on distorted validation subsets initially oscillated—e.g., one epoch showed slightly higher mAP@50 on blurred images but lower on bright, and vice versa. By epoch 20, these oscillations had dampened, indicating that the model had found a balanced solution that works across distortions. As shown in Fig. 6, the augmented model's fine-tuning curves converge within ≈20 epochs; note the narrower gap between training and validation loss, indicating reduced overfitting. The final training loss of the augmented model was slightly higher than the baseline's final loss (since it was optimizing a harder, augmented task), but this did not harm its overall performance; rather, it prevented overfitting to the original distribution. This behavior is consistent with the regularization effects observed when using heavy augmentation in other studies.

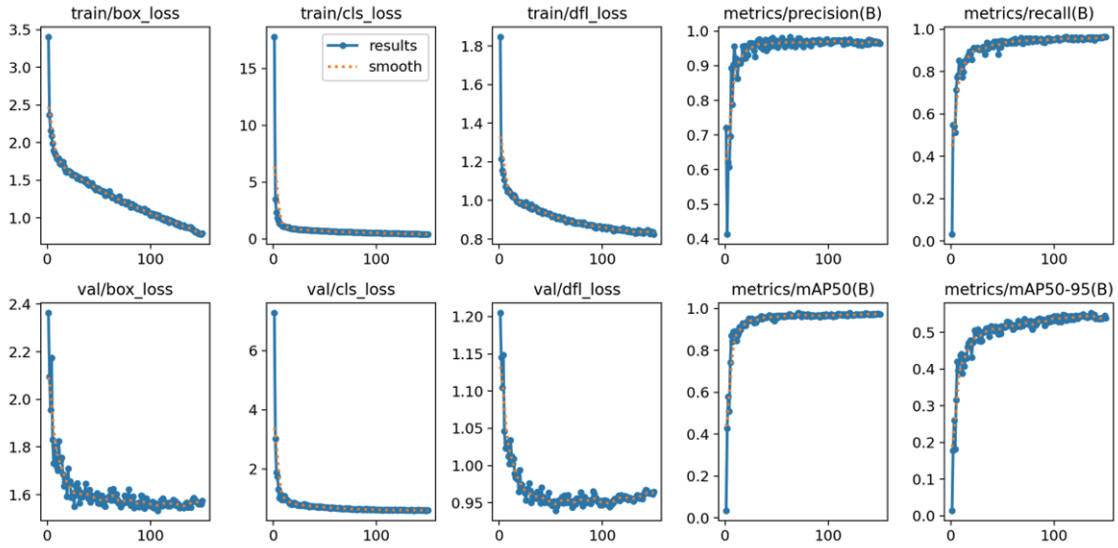

Fig. 5. (Color online) Baseline training results.

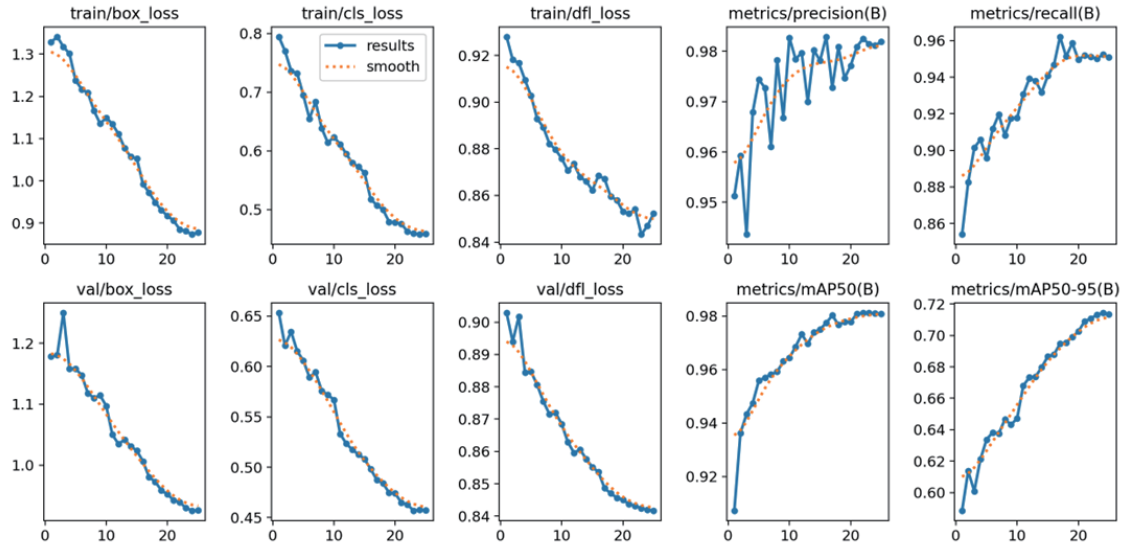

Fig. 6. (Color online) Augmented model training results.

From a qualitative perspective, the inspection of detection outputs reveals how the augmented model differs. On bright/overexposed images, the baseline often missed objects that were nearly white against white backgrounds, or it localized only parts of a saturated object. The augmented model, on the other hand, detected objects in these scenarios more completely. We suspect that the model learned to utilize contextual cues and shape continuity: even if the center of an object is washed out, the model picks up on the visible edges or surrounding components to infer an object's presence. On Gaussian blurred images, baseline detections were often misaligned or of the wrong size, whereas the augmented model placed tighter bounding boxes, indicating better localization despite blur. It likely learned more robust shape features (perhaps akin to edge detectors that still fire under blur). For motion blur, the augmented model occasionally detected faint elongated shapes as objects (which the baseline almost never did), but many objects remained undetected when the blur was very strong. This suggests the need for further research—possibly with the integration of a deblurring step or using sequence information if available.

3.2.1 Inference time

The average inference time per image (800 × 800 input) for the baseline model was determined to be 20.6 and 19.5 ms for the augmented model on the RTX 3060 GPU. This difference (<2%) is within run-to-run variance. Essentially, both models run ~50 FPS, which is expected since they share the YOLOv8s architecture. The negligible difference confirms that the improved robustness comes at no cost to runtime efficiency, a critical factor for real-world AOI where inspection must keep up with production line speeds.

3.2.2 Model size

Both models have the same number of parameters. Training with augmented data did not inflate the model size, it only adjusted weights. The checkpoint file sizes are identical. This means that the deployment requirements (memory and storage) remain the same, making our approach attractive for industrial stakeholders since better performance can be achieved without needing larger hardware.

3.3 Benefits of LLM guidance and generalizability

The core of our approach was using an LLM to guide augmentation. Analyzing the outcome, it is worth asking: could we have manually chosen those augmentations and achieved similar results? Possibly yes, since a knowledgeable practitioner might have hypothesized that blur and brightness augmentations were needed. However, the LLM provided a systematic and unbiased analysis, confirming the intuition and ensuring no major factor was missed (for example, it also suggested the composite augmentation, which we might not have initially considered but which proved beneficial). The LLM essentially acted as an automated consultant, saving time in the trial-and-error selection of augmentation types and intensities. In more complex cases, an LLM could reveal nonobvious augmentation strategies or even data collection strategies, especially as these models ingest vast literature and empirical knowledge. Our experiment demonstrates the feasibility of integrating such AI guidance in the computer vision training loop.[19]

One notable observation is the improvement on the *unaugmented* normal images as well. We expected robustness improvements on distortions, but the new model also showed slight improvement under normal conditions. This can be attributed to the regularization effect: by seeing more diverse examples, the model likely avoided overfitting to peculiarities of the original training set (e.g., it did not latch onto a specific lighting condition as a cue for certain classes). Instead, it had to truly learn the object features in a variety of contexts, which improved overall discrimination. This hints that even when distortions are not a primary concern, augmentation guided by failure modes can boost baseline performance.

3.3.1 Generality

While our case study is in AOI for electronics, the approach is general. Any vision application where a model's performance degrades in certain scenarios could potentially use an LLM to suggest targeted data augmentation or data collection. For example, an autonomous driving model that struggles at night could be augmented with more nighttime data as suggested by an LLM analyzing the errors. The key is that the LLM can parse error patterns (e.g., "most missed detections happen when the object is small and in motion") and translate that to an actionable training modification (e.g., "use image scaling augmentation to simulate small distant objects and motion blur augmentation"). This bridges the gap between high-level analysis and low-level implementation, which traditionally relied on human engineers.

3.3.2 Limitations

Our approach is not without limitations. The LLM's suggestions are only as good as the information provided. If the user does not accurately describe the model's issues, the LLM might overlook a needed augmentation. In our case, we guided the LLM with specific performance numbers; in fully automated scenarios, one would need a script to summarize model errors for the LLM. Additionally, LLMs might not suggest advanced techniques beyond augmentation; for instance, if a completely new architecture component is needed (like an attention mechanism for lens flare), the LLM might not "know" to propose it unless explicitly asked about architecture. We specifically constrained our scope to augmentations.

The persistent difficulty with motion blur suggests that augmentation alone might not suffice for certain distortions. Even though we augmented motion-blurred images, the model could not fully recover performance. This is consistent with prior work like Cho *et al.* who had to incorporate a deblurring sub-network.[9] This indicates that some problems require specialized solutions (e.g., algorithmic preprocessing or multiframe analysis) beyond what single-frame augmentation can solve. In practice, one could incorporate such solutions in a pipeline (for example, use a motion deblurring GAN before feeding images to YOLO or use longer exposure to avoid blur). Our study remains within the paradigm of single-frame detection without extra modules, which is a challenging setting.

Another limitation is that our augmentations, while diverse, do not cover all real-world distortions (e.g., sensor noise, color shifts, and lens distortions). We focused on the major factors affecting our scenario. An LLM might be further used iteratively: after one round of improvement, analyze remaining errors (perhaps now noise might emerge as the next factor) and suggest more augmentations. This iterative refinement could continue until the model's performance converges across conditions.

4. Conclusions

We utilized an LLM to analyze data-augmented images for fine-tuning the original defect detection model. We presented a novel approach to improving the robustness of an object detection model for AOI by leveraging an LLM to guide data augmentation. The baseline YOLOv8 model, while accurate on pristine images, suffers substantial performance drops under common image distortions such as extreme brightness and blur. By prompting an LLM (GPT-4o-mini) with the baseline's failure patterns, we obtained targeted augmentation recommendations and used them to retrain YOLOv8. The resulting model showed markedly improved detection metrics on distorted images—most notably increased mAP@50 on blurred images from 31.6 to 94.9% (a threefold improvement) and similarly strong gains in handling overexposure and combined artifacts. The retrained model achieved almost the same precision and recall on challenging images as it did on normal ones, reflecting a significant enhancement in generalization. Importantly, these benefits came with no additional inference cost or model complexity, as we did not modify the network architecture, only the training data. This maintains the real-time capability essential for inline AOI systems on production lines.

Our findings underscore that data, especially intelligently augmented data, can be as powerful as architectural innovations in addressing model weaknesses. The use of an LLM adds a new dimension to this: it provides a high-level "eye" to identify what the model needs more exposure to, which is a task that can be tedious for human engineers when dealing with many failure cases. The LLM essentially distilled the best practices (like augment with blur for blur issues) from its knowledge base, tailored to our scenario. This approach could be readily applied to other computer vision tasks where specific failure modes are evident. For instance, in surveillance, if a model fails in rain or fog, an LLM might suggest simulating rain/fog in training data. We envision a future workflow where developers routinely consult LLMs for improving models, creating a loop of continuous model refinement.

In the broader context of smart manufacturing, deploying robust AI inspectors increases the reliability of quality control. Our LLM-augmented YOLOv8 could, for example, be used in an optical inspection station that sometimes sees motion-blurred images (due to conveyor belt vibration) or varying brightness (due to shiny metallic parts). The improved model would likely reduce false negatives (missing defects) and false positives (raising false alarms), thereby saving cost and ensuring product quality.

In summary, our LLM-guided data augmentation framework demonstrates the power of ML techniques in advancing sensing applications. By relating our work to related technologies, that is, ML techniques, we highlight how LLMs can serve as adaptive agents to mitigate vulnerabilities in sensor-based detection systems, paving the way for more robust and efficient quality control in manufacturing.

Future Work: While our approach significantly improved robustness, the residual gap in cases like heavy motion blur invites further research. One direction is to integrate pre-processing learned models (like a small CNN to enhance or deblur the image) in conjunction with augmentation. Another is to explore *generative* data augmentation using LLMs or other generative models—for example, using a diffusion model to synthetically generate training images with distortions not easily captured by simple filters. Additionally, an interesting extension would be to use the LLM not just for augmentation advice but also for error explanation. For instance, having the LLM explain why the model might be failing (e.g., "the model confuses glare with object edges") could guide more nuanced corrections like adjusting the model's NMS or using polarizing filters in imaging. Finally, evaluating this approach on other object detectors (like two-stage R-CNN variants or transformer-based detectors) and other datasets (e.g., autonomous driving benchmarks, and medical images) would help validate the universality of LLM-guided training enhancement.

In conclusion, this work contributes to sensors and related materials by demonstrating how LLM-guided augmentation optimizes the application of optical sensing in PCB defect detection. By addressing distortions in data from high-resolution cameras and 3D sensors, our approach enhances reliability in Industry 4.0 and thus solves remaining challenges through iterative AI-sensor integration and paves the way for future advancements in sensing technologies.

Acknowledgments

This study was supported in part by National Science and Technology Council Grants (114-2221-E-035-081- and NSTC 113-2221-E-035-047-).

References

1 J. Redmon, S. Divvala, R. Girshick, and A. Farhadi: Proc. 2016 IEEE Conf. Computer Vision and Pattern Recognition (CVPR) (IEEE, 2016) 779–788. https://doi.org/10.1109/CVPR.2016.91
2 M. Hussain: IEEE Access **12** (2024) 42816. https://doi.org/10.1109/ACCESS.2024.3378568
3 J. Terven, D. M. Córdova-Esparza, and J. A. Romero-González: Mach. Learn. Knowl. Extr. **5** (2023) 1680. https://doi.org/10.3390/make5040083
4 A. Simeth, A. A. Kumar, and P. Plapper: J. Intell. Manuf. **36** (2025) 3447. https://doi.org/10.1007/s10845-024-02411-5
5 Y.-W. Chen and J.-M. Shiu: Int. J. Adv. Manuf. Technol. **119** (2022) 8257. https://doi.org/10.1007/s00170-022-08676-5
6 M. G. Ragab, S. J. Abdulkader, A. Muneer, A. Alqushaibi, E. H. Sumiea, R. Qureshi, S. M. Al-Selwi, and H. Alhussian: IEEE Access **12** (2024) 57815. https://doi.org/10.1109/ACCESS.2024.3386826
7 M. Hussain and R. Khanam: Solar **4** (2024) 351. https://doi.org/10.3390/solar4030016
8 F. Ndayishimiye and J. J. Lee: J. Multimedia Inf. Syst. **8** (2021) 211. https://doi.org/10.33851/JMIS.2021.8.4.211
9 S.-J. Cho, S.-W. Kim, S.-W. Jung, and S.-J. Ko: IEEE Access **10** (2022) 79491. https://doi.org/10.1109/ACCESS.2022.3194898
10 C. Michaelis, B. Mitzkus, R. Geirhos, E. Rusak, O. Bringmann, A. S. Ecker, M. Bethge, and W. Brendel: arXiv:1907.07484 (2019). https://doi.org/10.48550/arXiv.1907.07484
11 C. Shorten and T. M. Khoshgoftaar: J. Big Data **6** (2019) 60. https://doi.org/10.1186/s40537-019-0197-0
12 F. K. Sufi: Information **15** (2024) 99. https://doi.org/10.3390/info15020099
13 T. Diwan, G. Anirudh, and J. V. Tembhurne: Multimedia Tools Appl. **82** (2023) 9243. https://doi.org/10.1007/s11042-022-13644-y
14 J. Zhu, X. Li, P. Jin, Q. Xu, Z. Sun, and X. Song: Sensors **21** (2021) 27. https://doi.org/10.3390/s21010027
15 Z. Yuan, X. Tang, H. Ning, and Z. Yang: Symmetry **16** (2024) 418. https://doi.org/10.3390/sym16040418
16 K. D. Apostolidis and G. A. Papakostas: Electronics **14** (2025) 1624. https://doi.org/10.3390/electronics14081624
17 N. Hütten, M. A. Gomes, F. Hölken, K. Andricevic, R. Meyes, and T. Meisen: Appl. Syst. Innov. **7** (2024) 11. https://doi.org/10.3390/asi7010011
18 H. Yaseen: arXiv:2502.07592 (2025). https://doi.org/10.48550/arXiv.2502.07592
19 L. Fang, G.-G. Lee, and X. Zhai: arXiv:2310.18365 (2023). https://doi.org/10.48550/arXiv.2310.18365