International Journal of Science and Engineering Applications
Volume 15-Issue 07, 83 – 87, 2026, ISSN:- 2319 - 7560
DOI: 10.7753/IJSEA1507.1014

Industrial Defect Dataset Construction and Detection Based on YOLOv8

Liao Wangwang
School of Electronic Information and Electrical Engineering, Yangtze University
Jingzhou, Hubei, China

Abstract: Detection of tiny defects on industrial surfaces is a critical procedure to guarantee product quality and production safety, and relevant research bears significant engineering value. Based on the YOLOv8n model, this paper proposes a lightweight improved YOLOv8 model for industrial tiny defect detection. The improvements are implemented in two dimensions. In terms of training strategy, the regression branches of the detection head are frozen to retain the localization knowledge from pre-trained weights and mitigate overfitting under small-sample conditions. In terms of network architecture, standard convolutions are replaced with Ghost Convolutions, and C2f modules are substituted with C3Ghost modules to substantially reduce the model's parameter count and computational cost.Experiments conducted on a self-built industrial defect dataset demonstrate that the improved model reduces the number of parameters and computational load by 42.9% and 37.8% respectively, with negligible performance degradation, and its overall performance outperforms the original YOLOv8n. The experimental results verify that the proposed method achieves model lightweighting while preserving detection accuracy, and effectively enhances the generalization capability under small-sample scenarios.

Keywords: Industrial defect detection; Deep learning; Few-shot learning; Model lightweighting; YOLOv8n

1. INTRODUCTION

Industrial surface defect detection serves as a core procedure to guarantee product quality and production safety. With the rapid advancement of intelligent manufacturing technology, industrial production lines have raised continuous requirements for the precision of defect detection[1]. Tiny defects, characterized by faint feature information and vulnerability to interference from background textures, pose great detection challenges and represent a major bottleneck in the field of industrial defect inspection. The emergence of deep learning has delivered innovative technical solutions for industrial defect detection, and convolutional neural network-based detection algorithms have gradually become the mainstream in the industry[2].

Before the popularization of deep learning, scholars at home and abroad mainly conducted research on industrial defect detection relying on handcrafted features, and developed a series of mature traditional visual inspection methods[3]-[8]. Liu et al.[4] proposed a multi-block local binary pattern algorithm. Benefiting from its concise and efficient computing performance, this algorithm realizes real-time detection of steel plate surface defects effectively. Zhu et al.[5] integrated the autocorrelation function and gray-level co-occurrence matrix algorithm. The detection window was adaptively determined according to texture cycles. Combined with feature distance calculation and threshold discrimination, the method can accurately identify various typical fabric defects. Lee et al.[6] adopted a morphological top-hat filtering algorithm to suppress the interference of pseudo-defects on billet surfaces. Verified by images captured from actual production lines, the algorithm achieves precise detection of surface defects on steel billets. Based on Fourier image reconstruction technology, Tsai et al.[7] realized effective identification of subtle defects on solar cells through frequency-domain reconstruction and comparison of image gray-scale differences. Yun et al.[8] optimized Gabor filters via the univariate dynamic search encoding algorithm. A cost function was constructed based on the criterion of energy separation between defects and backgrounds, and filter parameters were optimized iteratively. This method significantly enlarges the feature difference between defects and backgrounds, thus completing accurate defect detection of steel billets with scale. The above traditional algorithms feature simple deployment and favorable real-time performance. Nevertheless, they heavily rely on manual feature design and empirical parameter tuning, with limited generalization ability and adaptability to diverse working conditions, which makes them difficult to meet the detection requirements of high-precision tiny defects.

To break through the performance bottlenecks of traditional inspection methods, deep learning technologies have been widely applied to industrial defect detection tasks, substantially improving the accuracy and speed of defect detection[9]-[12]. Oh et al.[10] constructed a welding defect detection model based on Faster R-CNN, which realizes integrated automatic detection of feature extraction and defect classification and raises the automation level of welding defect inspection. Li et al.[11] addressed the low sensitivity to tiny defects of photovoltaic cells and the conflict between detection accuracy and inference speed, and proposed an improved multi-scale context-aware YOLOv8 model that greatly boosts the detection performance for micro-defects. Hou et al.[12] comparatively analyzed the industrial detection performance of mainstream models including YOLOv5, YOLOv8, YOLOv10s and YOLOv11, and verified that lightweight YOLO-series models are suitable for industrial defect inspection scenarios requiring high precision and high efficiency. In summary, YOLO-series models possess remarkable advantages in industrial defect detection[13]. However, existing approaches fail to sufficiently exploit shallow detailed features, leaving room for further optimization in the detection accuracy of tiny defects.

Aiming at the insufficient training data and scarce defect categories encountered by existing detection algorithms under small-sample and class-imbalanced conditions, this paper

International Journal of Science and Engineering Applications
Volume 15-Issue 07, 83 – 87, 2026, ISSN:- 2319 - 7560
DOI: 10.7753/IJSEA1507.1014

proposes an improved YOLOv8 detection model for industrial defect inspection.

2. INDUSTRIAL DEFECT DETECTION MODEL

2.1 Baseline YOLOv8

YOLOv8 is a one-stage object detection algorithm released by Ultralytics in 2023. It achieves an excellent trade-off between detection speed and accuracy, and has been widely adopted in industrial defect detection. Its network architecture is illustrated in Figure 1.

The backbone network extracts multi-scale basic features from input images. Through alternating stacking of convolutional layers and C2f modules, progressive downsampling is performed. Finally, the SPPF module fuses feature information from different receptive fields and outputs high-level features with abundant semantic information.

The neck network adopts a bidirectional FPN+PAN structure. Bidirectional fusion of multi-scale features is realized via Upsample and downsampling operations, enabling feature maps to carry both high-level semantic information and low-level detailed information simultaneously.

The detection head consists of three parallel branches P3, P4 and P5, corresponding to feature maps of three scales: 80×80, 40×40 and 20×20. They are responsible for outputting bounding box coordinates and category information of targets.

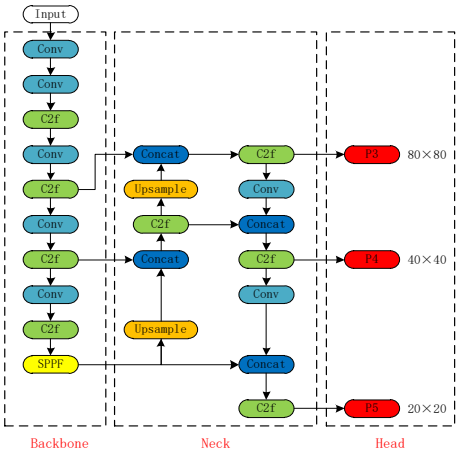


Figure 1 Network architecture of YOLOv8

2.2 Improved YOLOv8 Model

The network structure of the improved YOLOv8 is shown in Figure 2. For the backbone and neck, the first Conv layer is retained, while the remaining Conv layers are replaced with GhostConv, and the C2f modules are substituted with C3Ghost modules. As for the detection head, the regression branches of the three detection heads P3, P4 and P5 are frozen.

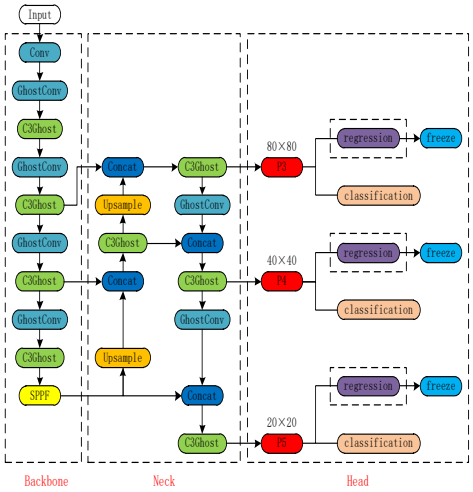


Figure 2 Network Architecture of Improved YOLOv8

2.2.1 GhostConv and C3Ghost

When processing images, multi-layer convolution operations of neural networks at different scales generate numerous redundant feature maps. GhostConv [14]combines standard convolution operations with low-computation linear transformations. It fully exploits the correlation between feature extraction and the redundancy of feature maps, reducing the model's parameter count and computational overhead while maintaining detection performance.

Figure 3 illustrates the differences between GhostConv and standard convolution. GhostConv first generates a small number of feature maps according to the specified kernel size. Afterwards, it enriches features and expands the number of channels through linear operations with low computational cost. Finally, feature extraction and linear transformation are executed in parallel to preserve the intrinsic features of input images and prevent the loss of valuable feature map information. Unlike standard convolution, which maps all channels to extract feature information, GhostConv does not perform convolution processing on every generated feature map.

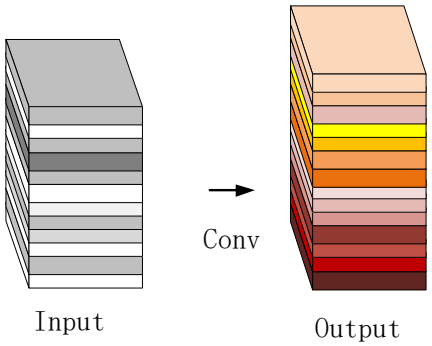
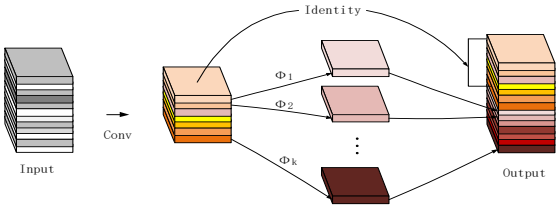


Figure 3(a) Standard Convolution



International Journal of Science and Engineering Applications
Volume 15-Issue 07, 83 – 87, 2026, ISSN:- 2319 - 7560
DOI: 10.7753/IJSEA1507.1014

Figure 3(b) GhostConv

Suppose there is an input image $X \in R^{c \times h \times w}$ with c input channels, height h and width w. The output feature map with n output channels obtained via standard convolution is denoted as $Y \in R^{h' \times w' \times n}$, and the convolution kernel is $f \in R^{c \times k \times k \times n}$. Accordingly, the floating-point operations (FLOPs) required by standard convolution are calculated as:

$$N_{FLOPs} = n \times h' \times w' \times c \times k \times k$$

For GhostConv, convolution kernels of the specified size are adopted to generate m intrinsic feature maps, where $m < n$. Linear transformations are sequentially performed on these m intrinsic feature maps to produce $s - 1$ additional ghost feature maps, satisfying $n = m\ s$. The final feature set is obtained by concatenating the intrinsic feature maps and newly generated ghost feature maps, yielding a total of $m + m \times (s - 1)$ feature maps. Hence, the floating-point operations ($FLOPs, N'_{FLOPs}$) consumed by GhostConv are expressed as:

$$N'_{FLOPs} = c \times k \times k \times m \times h' \times w' + m \times k \times k \times (s-1) \times h' \times w'$$

Thus, the speedup ratio between standard convolution and GhostConv can be derived as:

$$r_s = \frac{c \times k \times k \times ms \times h' \times w'}{c \times k \times k \times m \times h' \times w' + m \times k \times k \times (s-1) \times h' \times w'}$$

In GhostConv, it generally holds that $c >> s$. Therefore, Equation (3) can be simplified as:

$$r_s \approx \frac{s \times c}{s + c - 1} \approx s$$

The model compression ratio of standard convolution to GhostConv is defined as:

$$r_c = \frac{c \times k \times k \times ms}{c \times k \times k \times m + m \times k \times k \times (s-1)} = \frac{c \times s}{c + s - 1} \approx s$$

The theoretical speedup ratio is equal to the compression ratio, both equal to s. This demonstrates that GhostConv reduces FLOPs to a certain extent while preserving the detection performance of the model.

The C3Ghost module replaces the original Bottleneck layer with the GhostBottleneck layer, which effectively eliminates redundant computations introduced by standard convolutions inside Bottleneck while retaining feature extraction capability. Such replacement not only cuts down the number of parameters but also guarantees lightweight effects.

2.2.2 Freeze Regression Branches Training Strategy

This paper proposes a training strategy that freezes regression branches. Specifically, during the training phase, all trainable parameters of the regression branches belonging to the three detection heads P3, P4 and P5 in YOLOv8n are frozen, so they do not participate in backpropagation and weight updating. The classification branches, backbone and neck are trained normally.

In engineering implementation, this strategy consists of three steps. Step 1: Automatic localization of detection head modules. The model structure is traversed to match detection head layers with the class name Detect, avoiding version compatibility issues caused by manually specifying layer indices. If no Detect module exists in the model structure, an exception will be thrown to prompt users to check the network architecture, ensuring that subsequent freezing operations act on the correct layers.

Step 2: Accurate filtering of parameters to be frozen. All parameters under the detection heads are screened layer by layer: only parameters belonging to regression branches (cv2) are frozen, while classification branch parameters (cv3) are excluded to guarantee undisturbed training of classification branches. This filtering logic ensures precise freezing—neither mistakenly freezing classification branches nor missing any regression branch parameters.

Step 3: Freezing verification. Before training starts, all parameters are traversed to count their frozen status. The names of all frozen parameters are printed one by one, and the ratio of trainable parameters to total parameters is calculated. An excessively low ratio implies potential accidental freezing of irrelevant layers; an excessively high ratio indicates that freezing may not take effect. This verification mechanism provides verifiable guarantees for the correct execution of the freezing strategy.

The core idea of this strategy is as follows: in small-sample scenarios, the localization capability of regression branches has been sufficiently learned during COCO pre-training. Freezing these parameters preserves the pre-trained localization prior knowledge and prevents localization overfitting induced by fine-tuning with limited samples. Meanwhile, limited gradient signals are concentrated to optimize classification branches and feature extraction networks, enabling the model to learn discriminative features of the specific defect categories in this dataset more efficiently.

3. EXPERIMENTS AND ANALYSIS

3.1 Dataset

The images of the dataset used in this paper are collected from the actual production line of a cooperative factory, and backlight imaging is adopted to highlight the edge contours of products. The resolution of raw images captured by industrial cameras reaches hundreds of millions of pixels, while most edge defects of products only occupy tens to hundreds of pixels, resulting in an extremely large scale difference between defects and backgrounds.

There are three types of product edge defects studied in this paper: burrs (sharp protrusions on edges), under-etching (gentle outward bulges on edges), and over-etching (gentle inward depressions on edges). Since over-etching defects do not exist on the current production line, artificial simulation and image acquisition are carried out on separate equipment, forming a dataset with a "2+1" mode. Specifically, two types of real defects constitute the core detection task, and one type of artificially generated defect is used for prospective verification. Images of each type of defect are shown in Figure 4.

International Journal of Science and Engineering Applications
Volume 15-Issue 07, 83 – 87, 2026, ISSN:- 2319 - 7560
DOI: 10.7753/IJSEA1507.1014

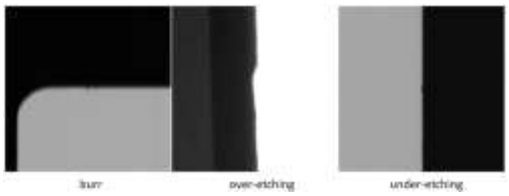


Figure 4 Images of various types of defects

Given the drastic size mismatch between raw images and tiny defects, this paper adopts a preprocessing pipeline of "labeling first, cropping afterwards". First, preliminary labeling is performed on the original full-resolution images with loose labeling criteria. Then, a 640×640 sub-image is cropped centering on each single defect. All cropped patches are further filtered strictly based on three criteria: definition, exposure and integrity.

A total of 187 high-quality samples are retained. The quantity statistics of each defect category are listed in Table 1, including 83 burr samples, 22 under-etching samples and 82 over-etching samples. The category ratio of approximately 4:1 between burrs and under-etching truly reflects the real distribution on production lines. With merely 22 samples, under-etching is a typical small-sample class. The dataset is split into training set and validation set at a ratio of roughly 9:1, and all subsequent experiments are conducted on this partition.

Table 1 Sample Quantity Statistics of Each Defect Category

Defect Category	Number of Samples	Proportion
burr	83	44.4%
under-etching	22	11.8%
over-etchng	82	43.8%
total	187	100%

3.2 Experimental Environment

The hardware platform is configured with an Intel Core i5-9300H CPU, 8 GB RAM, and an NVIDIA GTX 1650 graphics card with 4 GB video memory. The software configuration includes Python 3.9 programming language, PyTorch 2.2.2 deep learning framework, and CUDA 11.8 parallel computing architecture.

3.3 Evaluation Metrics

To verify the performance of the detection model, widely adopted evaluation metrics in the object detection field are selected: mean average precision (mAP), precision and recall. Detecting defects is the core target of industrial defect inspection. Considering the errors existing in workpiece defect shapes and annotations, the strict metric mAP@0.5:0.95 with high IoU thresholds is not suitable for this scenario. In contrast, mAP@0.5 only requires the intersection over union of bounding boxes to exceed 0.5 to confirm successful defect detection, which conforms to practical production line requirements and serves as a universal evaluation metric for industrial defect datasets.

1) mAP@0.5

mAP@0.5, defined as the mean average precision under an IoU threshold of 0.5, is adopted as the primary accuracy evaluation criterion. Its calculation formula is given as follows:

$$\text{mAP@0.5} = \frac{1}{N}\sum_{i=1}^{N} AP_i$$

2) Precision

Precision is defined as the proportion of correctly predicted samples among all samples predicted as positive. The calculation formula of precision is as follows:

$$\text{Precision} = \frac{TP}{TP + FP}$$

3) Recall

Recall is defined as the proportion of truly positive samples that are correctly predicted as positive. The formula for recall is as follows:

$$Recall = \frac{TP}{TP + FN}$$

3.4 Comparative Experiments

To verify the effectiveness of the proposed lightweight algorithm, all models are trained on the training and validation sets without data augmentation and evaluated on the test set. The evaluation results are presented in Table 2 (bold values represent optimal performance).

It can be observed that compared with YOLOv8n, the proposed lightweight model achieves a 3% drop in Recall and a 1.3% drop in mAP@0.5, while the number of model parameters is reduced by 42.9% and computational cost is cut by 37.8%. Meanwhile, Precision is improved by 13.7%. Although mAP@0.5:0.95 decreases significantly, mAP@0.5 serves as the core metric for industrial defect detection since only rough localization of defects is required.

Compared with the baseline model, the proposed lightweight model drastically reduces parameters while retaining competitive detection accuracy, which facilitates practical deployment on production lines. Among other YOLO-series models, our method achieves the smallest parameter size, lowest computational overhead and highest Precision. Even though YOLOv5n obtains a higher mAP@0.5, its overall performance is inferior to our lightweight model.

Table 2 Comparative Experiments of Improved YOLOv8 Models

Model Structure	FLOPs	Params	Recall	Precision	mAP@0.5
YOLOv3-tiny	19.0	12.13	0.645	0.734	0.657
YOLOv5n	7.2	2.51	0.655	0.737	0.668
YOLOv7-tiny	13.2	6.02	0.648	0.674	0.652
YOLOv8n	8.2	3.01	0.661	0.7	0.665
Ours	5.1	1.72	0.631	0.837	0.652

3.5 Ablation Experiments

To intuitively demonstrate the influence of each improved component of the proposed lightweight model on network

performance, ablation experiments are carried out on the dataset. Table 3 presents the experimental results. Here, "ghost" denotes replacing standard convolutions with Ghost Conv and C3Ghost modules, while "freeze" refers to freezing the regression branches of the P3, P4 and P5 detection heads.

Table 3 Ablation Experiments of Improved YOLOv8 Model

Model Structure	FLOPs	Params	Recall	Precision	mAP@0.5
YOLOv8n	8.2	3.01	0.661	0.7	0.665
YOLOv8n +ghost	5.1	1.72	0.598	0.787	0.622
YOLOv8n +freeze	8.2	3.01	0.668	0.735	0.679
Ours	5.1	1.72	0.631	0.837	0.652

It can be seen from Table 3 that in the second group of experiments, replacing standard convolutions with Ghost Conv and C3Ghost reduces the model parameters and computational cost by 42.9% and 37.8% respectively, but causes a noticeable degradation in defect detection accuracy. In the third group of experiments, applying the freezing strategy alone brings 0.7%, 3.5% and 1.4% improvements in Recall, Precision and mAP@0.5 correspondingly. The fourth group integrates GhostConv, C3Ghost and the freezing strategy simultaneously, which realizes model lightweighting while preserving competitive detection performance.

4. CONCLUSION

This paper implements lightweight improvement based on YOLOv8n with two optimization strategies only. First, the regression branches of detection heads are frozen to retain the general localization prior contained in pre-trained weights, so as to alleviate localization overfitting during small-sample defect detection. Second, all standard convolutions and C3 modules in the network are replaced with GhostConv and C3Ghost structures.By combining lightweight modules and the freezing strategy, the model drastically cuts down parameters and computational overhead while effectively maintaining detection accuracy, achieving a good trade-off between lightweight property and defect detection performance. The number of model parameters and FLOPs are reduced by 42.9% and 37.8% respectively, making the model more suitable for real-time detection of tiny industrial defects under limited computing power.In summary, the proposed lightweight improvement scheme balances detection accuracy and inference efficiency. Freezing regression branches stabilizes localization performance on small-sample datasets, while Ghost lightweight modules significantly reduce model storage cost and computational overhead. The model exhibits stronger robustness in tiny industrial defect detection tasks, and can be deployed on computing-limited devices such as industrial controllers and embedded boards. It provides a feasible lightweight solution for real-time detection of tiny industrial defects.

5. REFERENCES

[1] Wu L, Hao H Y, Song Y. Review on industrial metal surface defect detection based on computer vision[J]. Acta Automatica Sinica, 2024, 50(7): 1261-1283 (in Chinese). DOI: 10.16383/j.aas.c230039.

[2] Wang J M, Ning C K. Texture defect detection based on multi-scale residual feature matching[J]. Manufacturing Automation, 2025, 47(1): 82-88 (in Chinese).

[3] Chen Y, Ding Y, Zhao F, et al. Surface defect detection methods for industrial products: A review[J]. Applied Sciences, 2021, 11(16): 7657.

[4] Liu Y, Xu K, Xu J. An improved MB-LBP defect recognition approach for the surface of steel plates[J]. Applied Sciences, 2019, 9(20): 4222.

[5] Zhu D, Pan R, Gao W, et al. Yarn-dyed fabric defect detection based on autocorrelation function and GLCM[J]. Autex research journal, 2015, 15(3): 226-232.

[6] Lee D, Kang Y, Park C, et al. Defect detection algorithm in steel billets using morphological top-hat filter[J]. IFAC Proceedings Volumes, 2009, 42(23): 209-212.

[7] Tsai D M, Wu S C, Li W C. Defect detection of solar cells in electroluminescence images using Fourier image reconstruction[J]. Solar Energy Materials and Solar Cells, 2012, 99: 250-262.

[8] Yun J P, Choi S H, Kim J W, et al. Automatic detection of cracks in raw steel block using Gabor filter optimized by univariate dynamic encoding algorithm for searches (uDEAS)[J]. Ndt & E International, 2009, 42(5): 389-397.

[9] Li S B, Yang J, Wang Z, et al. Review on development and application of defect detection technology[J]. Acta Automatica Sinica, 2020, 46(11): 2319-2336 (in Chinese). DOI: 10.16383/j.aas.c180538.

[10] Oh S, Jung M, Lim C, et al. Automatic detection of welding defects using faster R-CNN[J]. Applied Sciences, 2020, 10(23): 8629.

[11] Li L Z, Yang J H, Wang T, et al. Surface defect detection method for photovoltaic cells based on improved YOLOv8[J]. Journal of Yangzhou University (Natural Science Edition), 2026, 29(1): 1-11+50 (in Chinese). DOI: 10.19411/j.1007-824x.2025.12.003.

[12] Hou X L, Huang L X, Yao X B, et al. Comparative study on industrial defect detection performance based on multi-version YOLO models[J]. Computer Knowledge and Technology, 2025, 21(31): 20-25 (in Chinese). DOI: 10.14004/j.cnki.ckt.2025.1565.

[13] Hussain M. YOLO-v1 to YOLO-v8, the rise of YOLO and its complementary nature toward digital manufacturing and industrial defect detection[J]. Machines, 2023, 11(7): 677.

[14] Han K, Wang Y, Tian Q, et al. Ghostnet: More features from cheap operations[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020: 1580-1589.