Journal of Computer and Communications, 2026, 14(2), 106-129
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

A YOLOv8-Based Network for Surface Defect Detection in Strip Steel

Jiacheng Jiang[1], Yuebin Su[1*], Jing Song[2]

[1]College of Mathematics and Statistics, Sichuan University of Science and Engineering, Zigong, China
[2]Information Technology Research Institute, Southwest Jiaotong University, Chengdu, China
Email: *dawnfading@outlook.com, *mathsyb@suse.edu.cn, jesen811206@126.com

How to cite this paper: Jiang, J.C., Su, Y.B. and Song, J. (2026) A YOLOv8-Based Network for Surface Defect Detection in Strip Steel. *Journal of Computer and Communications*, **14**, 106-129.
https://doi.org/10.4236/jcc.2026.142006

Received: January 29, 2026
Accepted: February 11, 2026
Published: February 14, 2026

Abstract

Effectively detecting subtle surface defects in strip steel is vital for industrial quality assurance; however, most existing approaches fail to strike an optimal balance between accuracy and efficiency, especially under real-time processing constraints. To address these limitations, we propose SGS-YOLO, a lightweight detection network built upon an enhanced YOLOv8 architecture. In this work, we offer three core contributions through our proposed method. First, we incorporate the star operation into YOLOv8's backbone and C2f modules to reduce the model's parameters and computational cost. Second, we introduce a Group-Sharing Detection Head (GS-Detect), leveraging Group Normalization and shared convolutions to jointly enhance accuracy and inference efficiency. Third, we propose a novel Self-Concatenated Feature Enhancement (SCFE) mechanism to enrich feature diversity and gradient propagation. Notably, SCFE itself is parameter-free, while the reported overall parameter/FLOPs increase arises from subsequent layers that receive widened inputs after concatenation. Compared with YOLOv8n, SGS-YOLO improves mAP@50 by 2.9%, respectively, on the NEU-DET dataset. With 1.49 M parameters and an inference speed of 130.22 FPS, SGS-YOLO achieves a favorable accuracy-efficiency trade-off. Compared to other mainstream models, SGS-YOLO demonstrates strong detection accuracy while maintaining a lightweight architecture, which satisfies the quasi-real-time requirement in multiscale defect detection task.

Keywords

Lightweight Network, Starnet, Self-Concatenated Feature Enhancement, Surface Defect Detection, YOLOv8

1. Introduction

Strip steel, known for its high strength, excellent ductility, and versatile formabil-

ity, is widely used in industrial production, construction, and various other sectors [1]. Common surface defects such as scratches, inclusions and edge cracks, which can compromise both performance and overall product quality [2] [3]. Additionally, such defects can cause more severe issues like belt breakage and production losses [4]. Therefore, rapid and accurate defect detection is essential to improving quality and efficiency in steel manufacturing.

Historically, surface defect detection in strip steel relied predominantly on manual visual inspection, wherein trained inspectors identified anomalies based on experience and subjective judgment. This approach, however, was limited by low efficiency, high labor cost, and poor reproducibility, making it ill-suited for modern, high-throughput production lines [5]. To address these limitations, early automation efforts employed traditional machine learning methods that typically involved handcrafted feature extraction followed by classification algorithms such as Support Vector Machines (SVM) and Decision Trees [6]. Although these methods improved detection accuracy to some extent, their dependency on human-defined features restricted their performance in terms of adaptability, localization precision, and generalization [7].

In recent years, the emergence of deep learning has revolutionized defect detection, particularly within the steel manufacturing industry. Object detection models built on convolutional neural networks (CNNs) have significantly improved both detection accuracy and automation efficiency. These models generally fall into two categories: two-stage and single-stage detectors. Two-stage approaches, such as Faster R-CNN [8] and Mask R-CNN [9], first generate region proposals followed by object classification and bounding box refinement. While offering superior accuracy, they are computationally expensive and lack real-time capability. In contrast, single-stage models like YOLO [10]-[12], SSD [13], and EfficientDet [14] directly perform classification and localization in a single forward pass, enabling faster inference and making them more suitable for real-time industrial deployment.

Among single-stage models, the YOLO family has gained particular prominence due to its impressive inference speed and end-to-end trainability. Its rapid image processing capabilities make it an attractive solution for surface defect detection in fast-paced industrial environments where latency is critical.

Despite these advances, achieving a satisfactory balance between detection accuracy, computational efficiency, and model robustness remains a challenge. YOLO-based models, while fast, often suffer performance degradation when applied to complex and highly variable surface defect patterns. Factors such as diverse defect morphologies, background interference, and limited annotated data can significantly impair detection precision and recall. These limitations underscore the need for enhanced architectures tailored to the unique demands of strip steel surface defect detection.

To this end, we propose a lightweight detection network based on an improved YOLOv8 framework [15]. This superior balance of precision and computational

cost underscores the model's potential for effective real-time deployment in high-demand industrial environments. The main contributions of this work are summarized as follows:

1) **StarNet Backbone and C2f Module Reconstruction**: We reconstruct the backbone and C2f modules of the baseline YOLOv8 architecture using the lightweight StarNet module. This redesign substantially reduces model parameters while enhancing the network's multi-scale feature extraction capability.

2) **Group-Sharing Detection Head (GS-Detect)**: We introduce a lightweight detection head, termed GS-Detect, which integrates Group Normalization and shared convolutional layers to reduce cross-scale head redundancy and improve normalization-aware feature processing.

3) **Self-Concatenated Feature Enhancement (SCFE)**: We propose SCFE, a novel intra-layer feature fusion strategy that concatenates a layer's output with its own duplicated representation, effectively doubling the channel dimension. Unlike conventional cross-layer or multi-scale fusion methods, SCFE enriches feature diversity and expressiveness without introducing additional parameters. Furthermore, it facilitates implicit multi-path gradient propagation, thereby enhancing optimization stability and convergence speed. While SCFE introduces no learnable parameters itself, it widens the downstream inputs and thus may slightly increase the subsequent-layer weights and FLOPs at the network level.

2. Related Work

2.1. YOLOv8

This study builds upon the YOLOv8n model. YOLOv8 further improves both accuracy and flexibility through coordinated optimizations in the backbone, neck, and head. Specifically, in the backbone, YOLOv8 adopts the C2f module, which incorporates the ELAN-style design from YOLOv7 [12]. This design enhances feature extraction and gradient flow while maintaining a lightweight architecture. In addition, YOLOv8 employs a hybrid FPN-PAN neck to effectively aggregate multi-scale semantic and spatial cues, which is crucial for detecting defects ranging from microscopic pinholes to extensive scratches. The detection head follows an anchor-free paradigm, simplifying the pipeline and improving inference efficiency [16], which is particularly beneficial for small-object detection. The head component leverages enhanced feature computation to provide confidence scores and location information for each target. Consequently, the accuracy of both classification and regression tasks is improved, thus supporting precise object detection and classification [17].

With its anchor-free formulation and efficient multi-scale feature aggregation, YOLOv8 has become a competitive baseline for steel-related detection tasks.

2.2. Lightweight Real-Time Detectors for Strip Steel Defect Detection

Strip steel surface defect detection is challenging due to severe scale variation, sub-

tle texture contrast, irregular morphologies, and specular reflections [18] [19], which jointly demand strong multi-scale representation and robust localization under strict real-time constraints [20]. While earlier defect inspection systems often favored multi-stage pipelines for accuracy [21], their computational overhead and latency typically limit throughput in online production lines.

From the perspective of lightweight real-time deployment, SDD-YOLO [22] emphasizes compact modeling and generalization for strip steel surface defects under runtime constraints, while AEDN-YOLO [23] improves steel defect detection by enhancing feature extraction and multi-scale processing within a one-stage pipeline. For hot-rolled strip scenarios, TBD-YOLO [24] further adapts feature fusion and detection components to better handle scale diversity and complex backgrounds. In parallel, context- and receptive-field-oriented designs strengthen multi-scale cues via explicit context modeling: MSC-DNet [25] introduces efficient multi-scale context modules for strip steel defects, and Trident-LK Net [26] employs a lightweight trident structure with large kernels to improve multi-scale defect perception. Moreover, data/learning-regime adaptation has been explored for limited annotations: YOLO-SD [27] leverages stable-diffusion-driven simulation and feature fusion to boost few-shot industrial defect detection performance.

Beyond steel defect detection, target detection in other domains also provides methodological references for meeting strict latency constraints. MobileSAM-Track [28] enables efficient per-frame localization and mask prediction by decomposing semi-supervised video object segmentation into lightweight tracking and prompt-based segmentation, highlighting the value of modular designs for real-time deployment. Meanwhile, TGC-YOLOv5 [29] improves detection in complex scenes by incorporating Transformer-based encoding and attention mechanisms to strengthen feature interactions. Although these studies are not tailored to steel defect detection, they underscore the importance of jointly considering efficiency and representation.

Although existing lightweight networks achieve a reasonable trade-off between accuracy and efficiency, further improving performance without introducing redundant parameters remains challenging. To address this challenge, we propose SGS-YOLO, a lightweight object detection network, aiming to jointly reduce cross-scale redundancy, stabilize normalization under lightweight settings, and enhance feature expressiveness without incurring additional computational overhead. Details are provided in Sec. 3.

2.3. StarNet Algorithm Introduction

StarNet, recently introduced by Ma *et al.* [30], proposes a novel *star operation* to efficiently model second-order feature interactions via element-wise multiplicative transformations. Unlike traditional convolutions that rely on increased channel dimensions and nonlinear activations to enhance representational power, the star operation implicitly projects input features into a high-dimensional nonlinear space without explicitly expanding dimensionality. Formally, given an input fea-

ture vector $X \in \mathbb{R}^d$ and weight matrices $W_1, W_2 \in \mathbb{R}^{d\times d}$, the star operation is defined as

$$Y = \left(W_1^\top X\right) \odot \left(W_2^\top X\right), \tag{1}$$

where $\odot$ denotes element-wise multiplication. Each element of the output Y_k can be expanded as

$$Y_k = \sum_{i=1}^{d}\sum_{j=1}^{d} w_1^{ki} x_i \cdot w_2^{kj} x_j, \tag{2}$$

which implicitly encodes $\mathcal{O}\left(d^2\right)$ second-order interactions, akin to polynomial kernel mappings. This enables compact yet expressive feature representations.

Stacking multiple star blocks—each comprising depth-wise convolutions, parallel linear projections, element-wise multiplications, batch normalization, and nonlinear activations—allows StarNet to hierarchically aggregate high-order interactions, significantly enhancing both feature expressiveness and network efficiency. This forms the foundation for its integration into the YOLOv8 backbone and C2f modules.

In this work, we embed the star operation into the backbone and C2f modules of YOLOv8 to promote high-order feature interaction and fusion. Although our enhanced model maintains a comparable mAP to the original, it achieves approximately 33% reductions in both parameter count and FLOPs, demonstrating its suitability for deployment in latency-sensitive or resource-constrained environments.

Different from the approach adopted in [3], which primarily employs StarNet as a backbone replacement for generic object detection, our work focuses on a module-level integration of the star operation into both the YOLOv8 backbone and its C2f fusion units. This design choice is motivated by the fine-grained textures and irregular morphologies of strip steel defects, where high-order feature interactions are particularly beneficial. Moreover, the proposed StarNet integration is co-designed with our GS-Detect head and lightweight feature enhancement strategy, thereby reducing computational cost and improving inference efficiency.

3. Methodology

To advance the state of the art in steel surface defect detection, we propose a series of architectural enhancements to the YOLOv8 framework, aiming to simultaneously improve representational capacity and computational efficiency. The resulting architecture, termed SGS-YOLO, is designed to address the challenges of identifying subtle, irregular, and low-contrast defects in complex industrial settings. As shown in **Figure 1**, the framework integrates three key innovations:

First, we enhance the YOLOv8 backbone and C2f modules by incorporating the *star operation* and star blocks derived from StarNet, enabling high-order feature interactions critical for capturing subtle and irregular surface defects.

Second, we introduce the GS-Detect, a lightweight and unified detection head that replaces scale-specific convolutions with a shared group-normalized convo-

lutional block. This design encourages cross-scale semantic consistency, reduces parameter redundancy, and enhances training stability, particularly in resource-constrained environments.

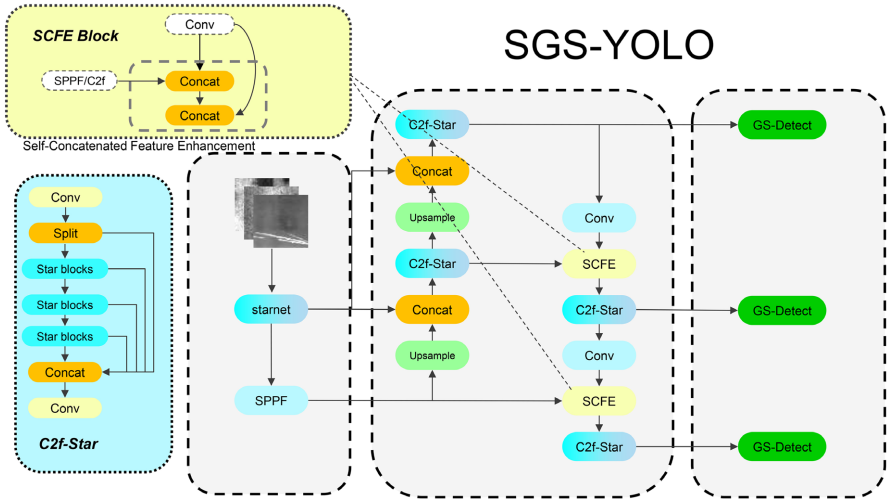

Figure 1. Architecture overview of SGS-YOLO.

Third, we design the Self-Concatenated Feature Enhancement (SCFE) module, a novel, parameter-free strategy that improves feature diversity by concatenating a feature map with itself before feeding it into downstream modules. Our analysis reveals that SCFE benefits from implicit regularization, feature reuse, and optimization advantages, achieving higher accuracy with reduced parameter and FLOPs overhead. As shown in Table 1, the locations of these three modules in the network are specified.

The following subsections will elaborate on the technical details and rationale behind each of these components.

Table 1. Module placement of StarNet, C2f-Star, SCFE, and GS-Detect in SGS-YOLO.

Module	Location	Pyramid level(s)
StarNet-s050	Backbone(replace YOLOv8 backbone)	*P*3/*P*4/*P*5 (stride 8/16/32)
C2f-Star	Neck fusion blocks (replace YOLOv8 C2f block)	*P*3, *P*4, *P*5
SCFE	before C2f-Star	*P*4 and *P*5
GS-Detect	Detection head(replace YOLOv8 head)	inputs: (*P*3, *P*4, *P*5)

3.1. Integration of StarNet in YOLOv8 Backbone and C2f Module

Building upon the principles introduced by StarNet, we integrate the star operation into YOLOv8's backbone and C2f module to enhance feature extraction and fusion capabilities, crucial for accurate steel defect detection. As illustrated in Figure 1, in the backbone, we replace the original YOLOv8 backbone with starnet_s050 as a unified feature extractor, followed by the original SPPF mod-

ule. In the neck, we replace the post-concatenation C2f fusion blocks in the YOLOv8 PAN/FPN with Star-enhanced C2f. C2f_Star is applied at three pyramid levels: P3/8 with 256 channels, P4/16 with 512 channels, and P5/32 with 1024 channels.

The star operation efficiently captures high-order feature interactions through element-wise multiplicative transformations, which allows for richer representations of complex defect patterns without explicitly increasing the network's width.

For the backbone, the star operation is employed to replace traditional convolutional layers, enabling the model to learn more discriminative and fine-grained feature representations. This enhancement allows YOLOv8 to better capture subtle textures and irregular shapes characteristic of steel surface defects, improving its ability to detect small and complex defects that are typically challenging for conventional CNN-based architectures.

In the C2f module, integrating the star operation facilitates a more sophisticated feature fusion mechanism. This enables the module to learn more complex interactions between multi-scale features, improving its ability to distinguish subtle variations between different defect types. The star operation enhances the capacity of YOLOv8's feature fusion process, allowing the model to better handle the variability and complexity inherent in steel surface defects.

The integration of the star operation into both the backbone and C2f modules significantly improves YOLOv8's capability to model complex defect patterns while preserving its lightweight nature. This design offers a theoretically grounded and practically effective solution for real-time, high-precision surface defect detection in resource-constrained industrial settings.

3.2. GS-Detect: Lightweight Shared Convolutional Detection Head

We propose a new lightweight detection head structure, abbreviated as GS-Detect, to replace the conventional detection head. Typically, the YOLO series employs independent detection heads at different feature levels, with each comprising three branches, to manage the same target at different scales. However, these branches operate independently, thus resulting in inefficient parameter utilization and an increased risk of model overfitting. Features detected by different feature layers should exhibit similar relative scales. This has been demonstrated by the FCOS [31]. To address the issue of low parameter utilization, this study adopted the concept of shared convolutional structures and introduced them into the detection head, thus enabling different branches to share convolutional layers.

Despite the use of shared convolutional layers, normalization remains essential because feature statistics vary across pyramid levels. Applying batch normalization in a shared detection head can lead to biased running statistics and accumulated errors in the moving averages, which in turn destabilizes training. We utilize Group Normalization because it enhances training stability, particularly with small or variable batch sizes, and improves feature consistency across scales, which is crucial for robust defect detection. In our head, GN is used in both the scale-specific

projection modules and the shared refinement block, while the final prediction layers are linear convolutions. This projection reduces dimensionality and ensures consistent feature representation across scales.

The GS-Detect structure is illustrated in Figure 2. GS-Detect employs a two-stage convolutional strategy. The first stage involves an initial scale-specific channel projection, followed by a cross-scale shared feature extraction stage, which significantly improves feature interaction and reduces model complexity.

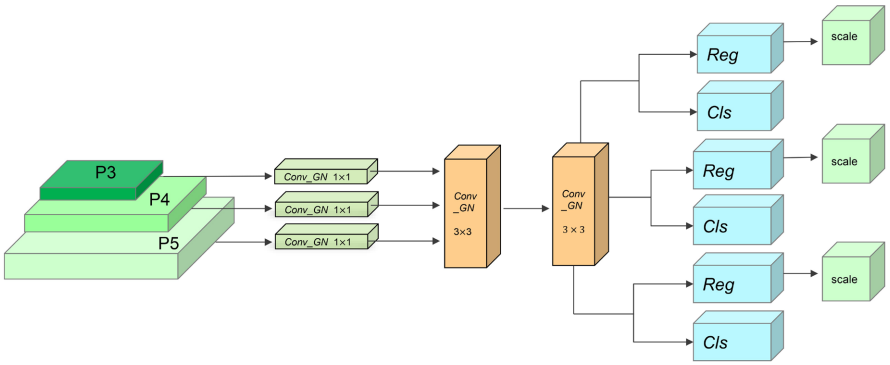

Figure 2. Architecture of the proposed GS-Detect module. Scale-wise 1 × 1 Group Normalized convolutions are first applied, followed by a shared two-layer 3 × 3 Group Normalized convolution block.

In the first stage, scale-wise 1 × 1 Group Normalized convolutions (Conv_GN) [32] are applied to each input feature map, projecting features from all scales into a unified channel space. Formally, given multi-level FPN features $\left\{X^{(l)}\right\}_{l=1}^{L}$ with $X^{(l)} \in \mathbb{R}^{B\times C_l\times H_l\times W_l}$, we project each level to a unified hidden dimension H_c via a level-specific 1 × 1 Conv-GN:

$$Z^{(l)} = \phi\left(\mathrm{GN}\left(\mathrm{Conv}_{1\times 1}^{(l)}\left(X^{(l)}\right)\right)\right). \tag{3}$$

In the second stage, the aligned feature maps from all scales are passed through a shared convolutional block consisting of two sequential 3 × 3 Group Normalized convolutions. The shared weights across scales ensure that the feature extraction process is consistent and efficient, facilitating rich cross-scale feature interactions while maintaining computational efficiency:

$$F^{(l)} = \Psi\left(Z^{(l)};\theta\right),\ \forall l, \tag{4}$$

where Ψ is instantiated as two shared 3 × 3 Conv-GN layers.

Following the shared convolutional block, the refined features are bifurcated into two separate branches: one for bounding box regression and the other for classification. The regression branch uses distribution focal loss (DFL) to predict bounding box parameters, augmented with learnable per-scale scaling factors to adaptively calibrate predictions at different scales:

$$\hat{R}^{(l)} = \alpha_l \odot R^{(l)}. \tag{5}$$

The classification branch outputs class confidence scores.

Following the ablation results reported in the original GN study [32], we adopt 16 channels per group, which was shown to achieve the best accuracy. With $H_c = 256$, choosing $G = 16$ keeps 16 channels per group, offering a practical trade-off between overly coarse normalization (small G) and overly fine normalization (large G), while preserving sufficient within-group cross-channel interactions and stable statistics.

In Ultralytics YOLOv8n, the detection head instantiates level-specific regression/classification towers $\left\{\text{cv2}^{(l)}, \text{cv3}^{(l)}\right\}_{l=1}^{L}$, so the head parameters and FLOPs grow approximately linearly with L. By contrast, GS-Detect keeps only the level-wise channel projections $\sum_{l=1}^{L} \mathcal{O}\left(C_l H_c\right)$, while sharing the refinement and prediction modules across all levels, yielding $\#\theta_{\text{GS}} = \sum_{l=1}^{L} \mathcal{O}\left(C_l H_c\right) + \mathcal{O}\left(P_{\text{shared}}\right)$ versus $\#\theta_{\text{YOLOv8}} = \sum_{l=1}^{L} \mathcal{O}\left(C_l H_c\right) + L \cdot \mathcal{O}\left(P_{\text{tower}}\right)$, and analogously lower head FLOPs (same $\sum_l H_l W_l$ dependence but fewer replicated tower operations).

3.3. Self-Concatenated Feature Enhancement (SCFE)

3.3.1. Motivation and Structural Design

To enhance the feature representation without incurring significant architectural complexity, we introduce the Self-Concatenated Feature Enhancement (SCFE) module. This design duplicates the input feature map F as $[F, F]$, forming a structurally redundant representation that is then processed by downstream modules. While the concatenation itself introduces no new parameters, it increases the input dimension for the subsequent layer. As shown in the SCFE module in **Figure 1**, we implement SCFE as a parameter-free self-concatenation operator and place it immediately after the bottom-up fusion on (P4) and (P5). Formally, given a fused feature map (F), SCFE produces along the channel dimension. The resulting enhanced features are then reorganized by subsequent C2f_Star blocks before being fed into the detection head.

3.3.2. Theoretical Analysis: Intra-Block Feature Reuse and Implicit Ensembling

The performance gain achieved by SCFE over a conventionally widened baseline, despite having fewer parameters, can be attributed to its structural enforcement of two synergistic principles: intra-block feature reuse and implicit ensembling.

First, SCFE facilitates dense information access by duplicating the input feature map F as $[F, F]$, enabling downstream modules, such as the C2f block with multiple internal transformations, to repeatedly extract features from the same source. This mirrors the feature reuse philosophy of DenseNet [33], albeit applied at a finer, intra-block granularity. Whereas DenseNet promotes global reuse across layers, SCFE enables local reuse within a block, allowing multiple transformations to jointly exploit shared information and extract complementary patterns. This increases representational diversity without increasing the number of independ-

ent input features.

Second, SCFE induces an implicit ensemble effect through its dual-branch computation. Let $F \in \mathbb{R}^{C\times H\times W}$ be the input. SCFE applies a channel-wise self-concatenation to form $\tilde{F} = [F, F] \in \mathbb{R}^{2C\times H\times W}$, which is then fed into a block with two learnable filters W_1 and W_2:

$$Y = \mathrm{Conv}(W_1, \tilde{F}) + \mathrm{Conv}(W_2, \tilde{F}), \quad \tilde{F} = [F, F]. \tag{6}$$

We split each kernel along the input-channel dimension as

$$W_1 = [W_{1a}, W_{1b}], \quad W_2 = [W_{2a}, W_{2b}], \tag{7}$$

where $W_{\cdot a}$ and $W_{\cdot b}$ each take C input channels. By the linearity of convolution w.r.t. input channels, we have

$$\begin{aligned} \mathrm{Conv}(W_1, [F, F]) &= \mathrm{Conv}(W_{1a}, F) + \mathrm{Conv}(W_{1b}, F), \\ \mathrm{Conv}(W_2, [F, F]) &= \mathrm{Conv}(W_{2a}, F) + \mathrm{Conv}(W_{2b}, F). \end{aligned} \tag{8}$$

Substituting Equation (8) into Equation (6) yields the equivalent effective-kernel form:

$$\begin{aligned} Y &= \mathrm{Conv}(W_1', F) + \mathrm{Conv}(W_2', F), \\ W_1' &= W_{1a} + W_{1b}, \quad W_2' = W_{2a} + W_{2b}. \end{aligned} \tag{9}$$

Due to independent initialization and the stochastic nature of training, the two convolutional paths are driven to learn complementary filters. This deterministic ensemble effect is structurally analogous to the multi-branch strategy used in Shake-Shake regularization [34], but is realized here without the need for explicit stochastic weighting or architectural complexity.

SCFE induces a deterministic two-path parameterization, yielding the gradient superposition

$$\nabla_F \mathcal{L} = \mathrm{Conv}^{\top}(W_1', G_Y) + \mathrm{Conv}^{\top}(W_2', G_Y), \tag{10}$$

where the backward signal is explicitly fused from two complementary routes. Although the forward form can be algebraically collapsed, Equation (10) reveals that the *training* process is not equivalent: the two branches, initialized independently and optimized under stochasticity, are naturally encouraged to specialize and provide mutually compensating gradients.

This dual-route accumulation effectively smooths the update direction (reducing variance and directional bias), improves conditioning, and enhances training stability. Therefore, the gradient derivation provides a principled support for our qualitative claim of *implicit ensembling*: SCFE behaves like a deterministic ensemble of two coupled learners, achieving stronger generalization without explicit stochastic mixing or additional architectural overhead.

In summary, SCFE harnesses the strengths of two well-established principles: local feature reuse for efficient representation learning, and deterministic multi-path processing for ensemble-like generalization. These mechanisms operate jointly to yield richer, more robust features, explaining the module's empirical advantage over heavier but less structurally optimized baselines.

3.3.3. Ablation Study

To rigorously evaluate the effectiveness of SCFE, we consider the baseline without SCFE that keeps the original channel width C, and further construct a capacity-enhanced variant termed TCE (Two-times Channel Expansion). In TCE, we explicitly widen the input channels of two selected C2f_Star blocks from C to $2C$ via standard learnable convolutional expansion, thereby emulating a conventional capacity scaling strategy with increased parameters and computation.

Functionally, TCE and SCFE share the same purpose: to enhance the expressiveness of the downstream C2f_Star blocks by increasing input dimensionality. However, unlike SCFE's parameter-free duplication mechanism, TCE relies on explicit structural expansion and a corresponding increase in learnable weights.

Importantly, aside from this localized difference, all other architectural components, training settings, and hyperparameters are held constant across both variants. This ensures a controlled and fair comparison focused solely on the efficacy of the feature enhancement strategy. The comparison results are reported in Table 2. Experimental results demonstrate that SCFE outperforms TCE, despite having fewer parameters and lower computational cost.

Table 2. Ablation comparison between TCE and SCFE.

Model	Params (M)	GFLOPs	Precision (%)	mAP@50 (%)	FPS
without SCFE	1.37	4.5	73.57	76.88	137.45
TCE	1.74	5.4	75.87	77.98	154.56
SCFE (Ours)	1.49	4.7	78.10	78.52	128.53

4. Experiments and Results

To evaluate the effectiveness of the proposed detection architecture, which integrates the StarNet backbone, Group-Sharing detection head (GS-Detect), and Self-Concatenated Feature Enhancement (SCFE) module, we conduct a comprehensive series of experiments. These include controlled ablation studies, comparisons with state-of-the-art (SOTA) methods, and cross-dataset generalization assessments.

4.1. Dataset

We evaluate our method on the publicly available benchmarks: NEU-DET. A widely used benchmark for evaluating surface defect detection on rolled steel strips. As illustrated in Figure 3, it contains six common defect categories: cracks (Cr), inclusions (In), patches (Pa), pitted surface (PS), rolled-in scale (RS), and scratches (Sc). The dataset comprises 1800 grayscale images, each with a resolution of 200 × 200 pixels. The dataset is divided into training, validation and testing sets in an 8:1:1 ratio.

All images are fed into the network at 640 × 640. For inputs with varying aspect ratios, we apply the standard letterbox strategy in YOLOv8: images are isotropically scaled to fit the target size and then symmetrically padded to 640 × 640.

Ground-truth boxes are scaled and shifted with the same transformations. This prevents geometric distortion from anisotropic resizing and maintains consistent spatial geometry across datasets.

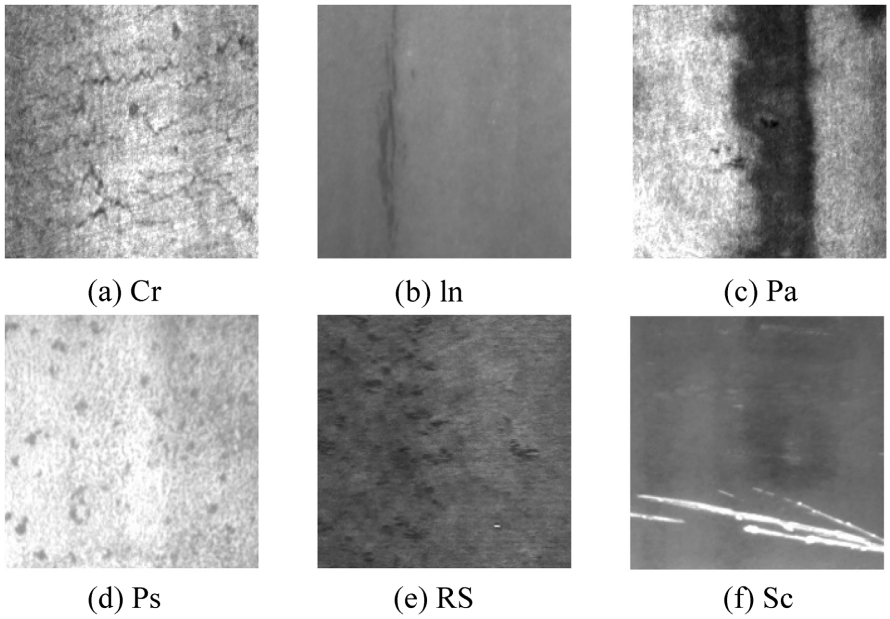

Figure 3. Visualization of the detection results of NEU-DET dataset, belonging to the categories: (a) Crazing; (b) Inclusion; (c) Patches; (d) Pitted_surface; (e) Rolled-in_scales; (f) Scratches.

4.2. Implementation Details

All experiments were conducted on a workstation running Ubuntu 22.04.2 LTS, equipped with an NVIDIA RTX 4090 GPU (24 GB VRAM). The models were implemented using PyTorch 2.3.0 with CUDA 12.1. We adopt YOLOv8n from the Ultralytics framework [15] as the baseline due to its lightweight architecture and favorable trade-off between accuracy and efficiency.

During training, all input images were resized to 640 × 640. We used stochastic gradient descent (SGD) with an initial learning rate of 0.01, momentum of 0.937, and a weight decay of 5×10^{-4}. The model was trained for 300 epochs with a batch size of 64 and 8 data loader workers. All experiments are conducted with a fixed random seed (seed = 0) and repeated five times to verify result stability.

To enhance robustness and generalization, we adopted a diverse data augmentation pipeline comprising photometric distortions (HSV jitter), geometric transformations (random flipping, translation, scaling), and composite techniques such as mosaic, RandAugment, and random erasing. Detailed settings are summarized in Table 3.

Table 3. Data augmentation configurations.

Category	Augmentation Type (Value)
Photometric	HSV jitter: $h = 0.015$, $s = 0.7$, $v = 0.4$

Continued

Geometric	Flip (horizontal: 0.5), translation: 0.1 Scale: 0.5
Composite	Mosaic: 1.0
Auto-augment	RandAugment
Erasing	Random erasing: 0.4

4.3. Evaluation Metrics

To provide an intuitive assessment of the classification performance of our defect detection model, we begin by presenting the confusion matrix, as shown in Figure 4. This matrix offers a comprehensive visualization of the model's predictions across all defect categories. The diagonal elements correspond to correctly classified instances, while off-diagonal elements indicate misclassifications—often between visually similar defect types. Such qualitative analysis reveals class-wise confusion patterns and complements the quantitative performance metrics presented in subsequent sections.

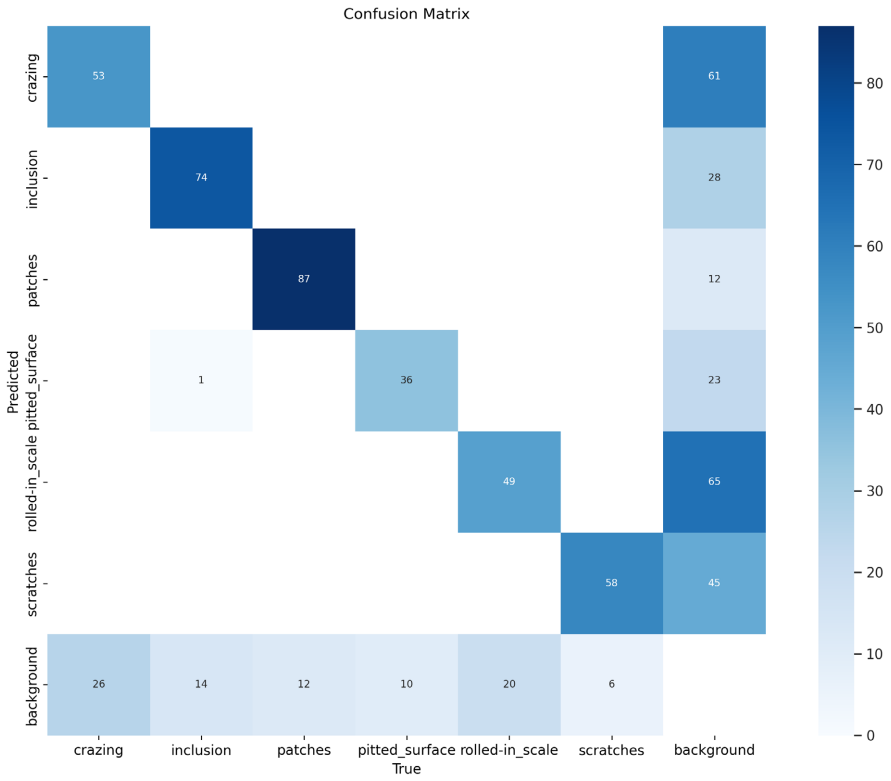

Figure 4. Confusion matrix of the proposed model on the test set.

In this work, we evaluate the performance of the proposed defect detection model using widely adopted metrics for object detection, including precision (P), recall (R), average precision (AP), and mean average precision (mAP). These metrics comprehensively assess the model's capability to accurately detect defects

while maintaining robustness.

Precision measures the proportion of correctly identified positive samples among all predicted positives, defined as

$$P = \frac{TP}{TP + FP} \tag{11}$$

where TP and FP denote true positives and false positives, respectively.

Recall quantifies the proportion of correctly identified positive samples among all actual positives:

$$R = \frac{TP}{TP + FN} \tag{12}$$

where FN is the number of false negatives.

Average precision (AP) for each defect class is computed by integrating precision over the entire range of recall values, reflecting the detection performance across all operating points:

$$\mathrm{AP} = \int_0^1 p(r)\mathrm{d}r \tag{13}$$

where $p(r)$ is precision as a function of recall r.

The mean average precision (mAP) aggregates AP values across all n classes to provide a single overall performance metric:

$$\mathrm{mAP} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{AP}(i) \tag{14}$$

In addition to detection accuracy, we assess the computational efficiency of the model through the following metrics:

- FLOPs (Floating Point Operations): Indicates the number of operations required for a single forward pass, serving as a proxy for computational complexity.
- Number of Parameters: Reflects the total count of learnable parameters. Fewer parameters typically correspond to lower memory and storage requirements.
- Frames per Second (FPS): FPS is reported as end-to-end throughput, including preprocessing, model forward inference, and postprocessing, which represents the model's inference speed. Higher FPS values signify better real-time processing capabilities, which are essential for practical deployment.

Together, these metrics provide a holistic evaluation of the model's effectiveness and efficiency, verifying its suitability for real-time deployment in resource-constrained industrial environments.

4.4. Visualization Experiments and Analysis

To thoroughly assess the effectiveness of the proposed SGS-YOLO architecture, we analyze both training dynamics and qualitative detection performance on the NEU-DET dataset. The training process spans 300 epochs, during which we monitor key performance indicators. Experimental results are presented through precision-recall (P-R) curves and F1-confidence analyses, illustrating the advantages of SGS-YOLO in terms of detection accuracy and confidence calibration.

4.4.1. Precision-Recall Evaluation on the NEU-DET Dataset

We compute the precision, recall, and mean average precision (mAP) for all six categories of surface defects in strip steel, using both the baseline YOLOv8n and our SGS-YOLO method. The corresponding P-R curves are shown in Figure 5. As evident from the results, SGS-YOLO consistently improves upon the baseline in terms of precision and recall, leading to a higher overall mAP of 0.786 compared to 0.756 for YOLOv8n.

These improvements validate the superior detection capability of SGS-YOLO, particularly in scenarios where high recall and precision are essential. This demonstrates its applicability for real-world industrial defect detection tasks, where robustness and accuracy are critical.

4.4.2. F1-Confidence Analysis on the NEU-DET Dataset

To evaluate the confidence calibration of model predictions, we compare the F1-confidence curves of SGS-YOLO and YOLOv8n, as illustrated in Figure 6. SGS-YOLO achieves a higher peak F1 score of 0.73 at a confidence threshold of 0.494, surpassing YOLOv8n's 0.71 peak at a threshold of 0.310.

This enhancement indicates not only a better precision-recall trade-off but also a more reliable confidence distribution. SGS-YOLO maintains higher F1 scores across a broader confidence threshold range, reflecting improved robustness to threshold sensitivity. Furthermore, the elevated optimal confidence threshold implies a reduced tendency toward overconfident predictions and a lower false positive rate—an essential trait in high-stakes industrial inspection environments.

These findings affirm that SGS-YOLO provides more stable, trustworthy, and better-calibrated predictions, supporting its suitability for deployment in real-time surface defect detection systems.

4.4.3. Enhanced Localization and Coverage in Qualitative Results

To further substantiate the effectiveness of the proposed SGS-YOLO architecture from a qualitative standpoint, we visualize representative detection outcomes under diverse and challenging scenarios. As illustrated in Figure 7, two distinct

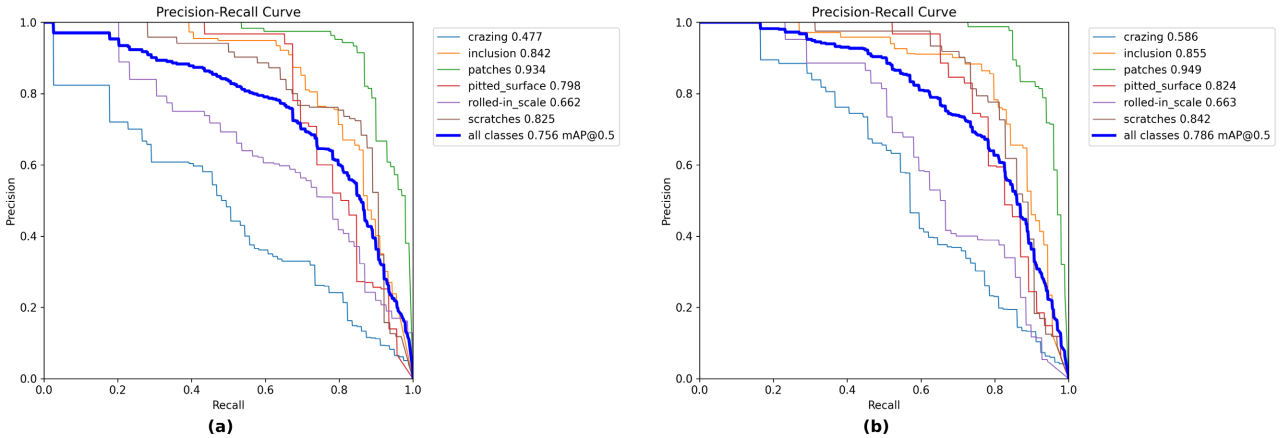

Figure 5. Precision-Recall curves for various defect types. (a) YOLOv8n and (b) SGS-YOLO.

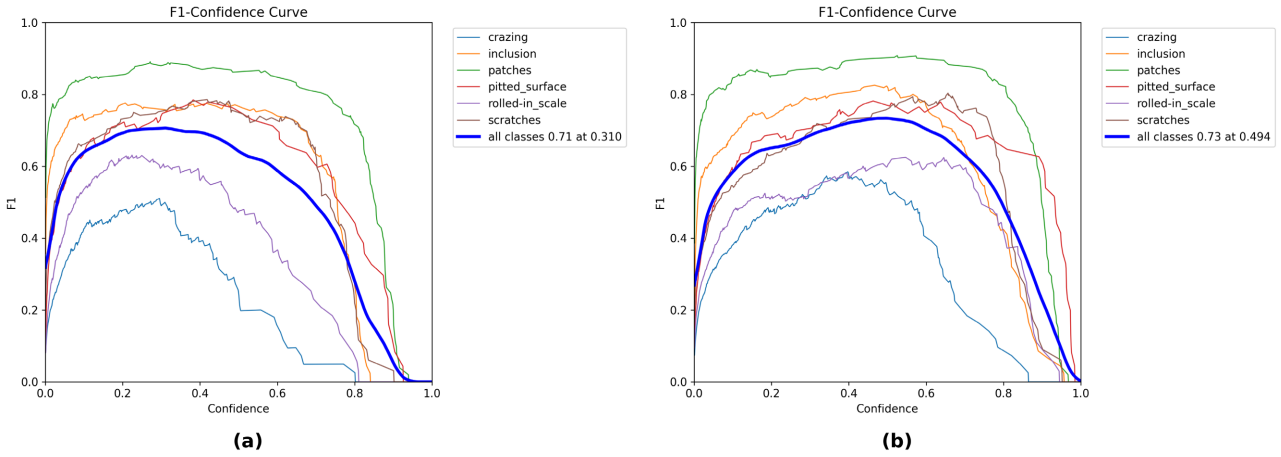

Figure 6. F1-confidence curves for various defect types. (a) YOLOv8n and (b) SGS-YOLO.

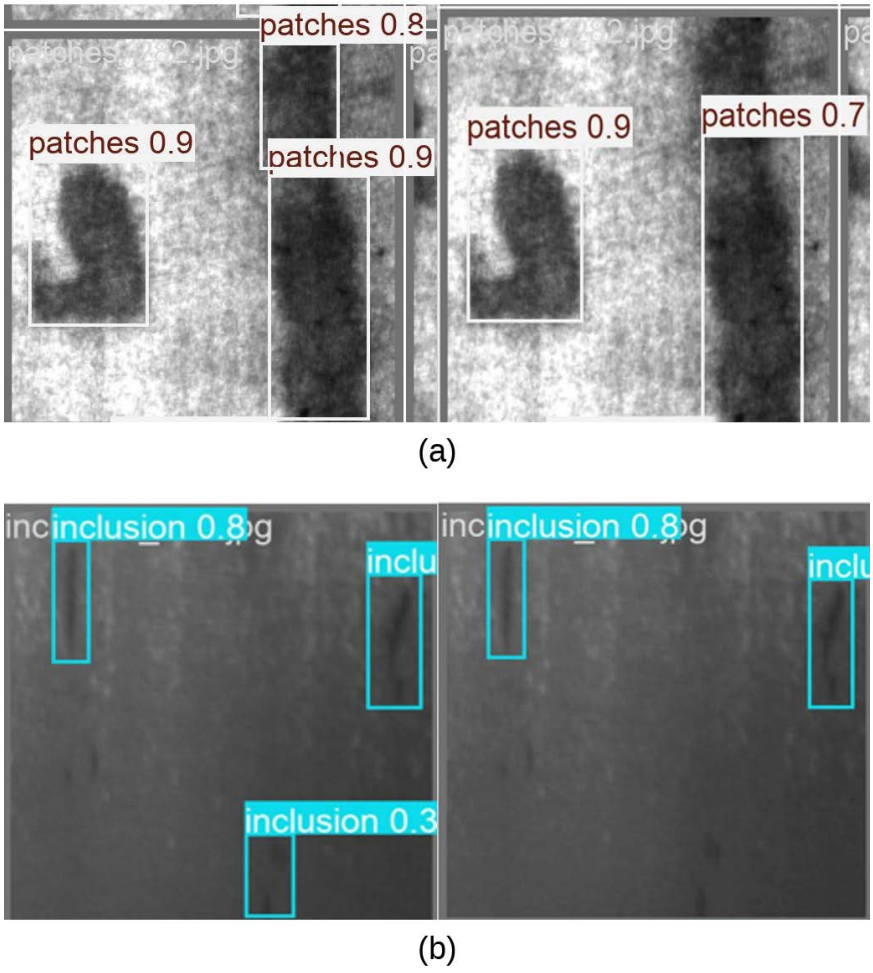

Figure 7. Qualitative comparison of detection results on NEU-DET. SGS-YOLO detects defect regions that YOLOv8n misses entirely or only partially localizes. From top to bottom: patches, inclusion.

advantages of SGS-YOLO over YOLOv8n emerge: (1) the ability to detect defect regions that are entirely missed by YOLOv8n, and (2) the ability to produce more

accurate, complete, and confidence-consistent bounding boxes in cases where both models generate predictions but YOLOv8n underperforms.

In (a), although both models produce detection results, SGS-YOLO consistently demonstrates superior spatial accuracy and confidence calibration. For example, in the patches samples, SGS-YOLO yields denser and more uniformly distributed bounding boxes that align more closely with ground-truth defect structures, whereas YOLOv8n exhibits sparse, imprecise, or offset predictions. SGS-YOLO accurately identifies multiple closely spaced defects with precise bounding box alignment, outperforming YOLOv8n's inconsistent or fragmented detections.

In (b), SGS-YOLO successfully localizes defects that are completely overlooked by YOLOv8n. For instance, in the inclusions, SGS-YOLO delineates the full spatial extent of fine-grained textures with high confidence, whereas YOLOv8n either fails to detect them or produces low-confidence predictions. SGS-YOLO captures multiple elongated and faint regions, including subtle features near image boundaries that YOLOv8n entirely misses.

Collectively, these visual results and training dynamics confirm that SGS-YOLO not only accelerates convergence and enhances training stability, but also significantly improves defect localization coverage and spatial precision—particularly in cases involving small-scale, ambiguous, or clustered anomalies. These capabilities underscore its practical value for real-world industrial surface defect detection applications.

4.5. Comparative Experiments

To comprehensively assess the performance of the proposed SGS-YOLO framework, we conduct two sets of comparative experiments on the NEU-DET dataset. The first group evaluates SGS-YOLO against various versions of the YOLO family (YOLOv8n through YOLOv12n), while the second benchmarks it against several state-of-the-art surface defect detection algorithms.

As summarized in **Table 4**, SGS-YOLO achieves a superior balance between detection accuracy and model efficiency. Specifically, it attains an mAP@50 of 78.52%, outperforming YOLOv8n (baseline) as well as subsequent variants including YOLOv9t [35], YOLOv10n [36], YOLOv11n [37] and YOLOv12n [38]. Despite its improved accuracy, SGS-YOLO maintains a compact architecture with only 1.49 million parameters and 4.7 GFLOPs, along with a high inference speed of 130.22 FPS. Compared to YOLOv8n, SGS-YOLO reduces parameter count by over 50% and compresses model size from 6.0 MB to 3.1 MB, underscoring its strong potential for real-time deployment in resource-constrained industrial inspection scenarios.

To further validate its effectiveness, SGS-YOLO is compared with a set of classical and contemporary SOTA detection frameworks, includ ing Yolo-SD [27], SDD-YOLO [22], Faster R-CNN [8], YOLO-SDS [3] and RT-DETR [39].

As reported in **Table 5**, both overall mAP and per-class detection accuracy

across six defect categories (Cr, In, Pa, Ps, Rs, Sc) are evaluated.SGS-YOLO achieves a competitive overall mAP of 78.5%, closely approaching that of Yolo-SD (82.3%) while utilizing substantially fewer parameters and computational resources. Notably, SGS-YOLO outperforms most competing models in detecting complex defect types such as cracks (Cr, 58.5%) and inclusions (In, 85.5%), demonstrating its robustness in handling challenging surface morphologies. Although the accuracy for scratches (Sc) is relatively lower (83.4%), the performance remains acceptable given the model's lightweight design. These results collectively confirm that SGS-YOLO offers an effective trade-off between detection accuracy, computational efficiency, and real-time inference—making it well-suited for practical industrial applications.

Table 4. Comparison of detection results and speed among YOLO series on the NEU-DET dataset.

Model	Params (M)	GFLOPs	Model Size (MB)	P (%)	R (%)	mAP@50 (%)	FPS
YOLOv8n	3.01	8.1	6.0	71.10 ± 0.21	70.98 ± 0.19	75.66 ± 0.22	134.23
YOLOv9t	1.97	7.6	4.5	68.91 ± 0.23	68.45 ± 0.20	75.03 ± 0.17	153.16
YOLOv10n	2.27	6.5	5.5	72.51 ± 0.16	66.03 ± 0.24	73.52 ± 0.28	165.73
YOLOv11n	2.58	6.3	5.2	73.84 ± 0.12	68.40 ± 0.26	75.64 ± 0.19	130.49
YOLOv12n	2.56	6.3	5.3	65.46 ± 0.27	68.48 ± 0.15	74.50 ± 0.23	**146.22**
Ours	**1.49**	**4.7**	**3.1**	**78.10 ± 0.24**	**70.69 ± 0.18**	**78.52 ± 0.20**	130.22

Table 5. Performance comparison of different methods on the NEU-DET dataset.

Metrics	YOLO-SD [27]	SDD-YOLO [22]	Faster R-CNN [8]	YOLO-SDS [3]	RT-DETR [39]	**Ours**
Params (M)	—	3.4	111.4	1.97	19.8	**1.49**
GFLOPs	84.4	6.4	60.5	5.3	57.0	**4.7**
mAP (%)	82.3	76.1	75.1	77.7	62.39	**78.5**
Cr (%)	54.4	58.1	38.9	47.1	28.62	**58.5**
In (%)	84.6	88.1	77.9	79.8	77.86	**85.5**
Pa (%)	93.6	94.9	85.3	95.4	85.76	94.9
Ps (%)	86.0	93.4	89.9	82.6	63.42	82.4
Rs (%)	77.5	61.7	67.5	66.9	51.85	66.4
Sc (%)	95.6	91.3	90.9	94.6	66.84	83.4

4.6. Ablation Study

To scrutinize the improvements offered by the three proposed enhancements to the network model, we undertook ablation experiments on the NEU-DET da-

taset using YOLOv8n as the baseline network. The experimental environment and parameter settings remained consistent, as summarized in Table 6. This analysis isolates the individual effects of the StarNet, the Group-Sharing Detection Head (GS-Detect), and the Self-Concatenated Feature Enhancement (SCFE) module.

Table 6. Ablation study results.

Star	GS-Detect	SCFE	Params (M)	GFLOPs	mAP@50 (%)
–	–	–	3.01	8.1	75.66
	–	–	2.01	6.1	75.18
–		–	2.36	6.5	73.67
		–	1.37	4.5	76.88
			1.49	4.7	78.52

Beginning with the YOLOv8n baseline, replacing its original backbone with the lightweight Starnet architecture results in a significant 33.3% reduction in parameter count and a 25.6% decrease in FLOPs, while incurring a negligible drop in detection accuracy. Although mAP@50 decreases by less than 0.5%, suggesting enhanced localization quality under stricter IoU thresholds. These results confirm that StarNet serves as an efficient feature extractor that preserves overall detection performance while substantially reducing computational cost.

Incorporating the GS-Detect module further enhances both precision and recall while continuing to reduce model complexity. Specifically, this variant requires only 43.4% of the original parameters and 56.1% of the FLOPs, while yielding a noticeable gain in mAP@50. although the GS-Detect module alone leads to a decline in certain performance metrics, when combined with the StarNet backbone, all evaluation indicators are consistently improved, indicating strong complementarity between the two designs. These results highlight the critical role of detection head design in improving both accuracy and efficiency.

Finally, integrating the SCFE module yields the best overall performance, with an mAP@50 of 78.52%. Although SCFE introduces a marginal increase in computational overhead, the substantial performance improvement fully justifies the trade-off. This configuration achieves the optimal balance between detection precision and computational efficiency.

For a more intuitive comparison, the evolution of mAP@50, parameter, and FLOPs across ablation stages is visualized in Figure 8, clearly demonstrating the progressive benefits introduced by each module.

Overall, the results confirm that each proposed architectural component contributes synergistically to improving the detection framework. The final model achieves a balanced trade-off between accuracy, model compactness, and inference cost, thereby making the network more comprehensive in detecting steel surface defects and improving its overall performance.

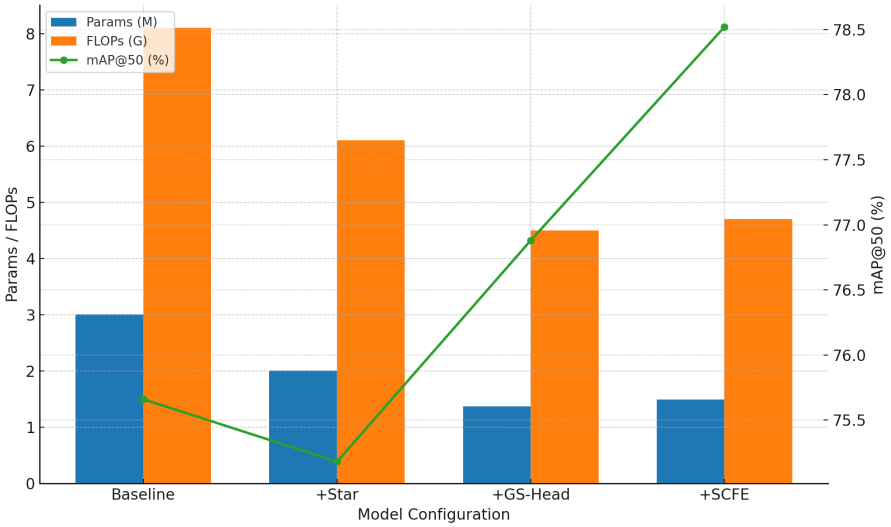

Figure 8. Impact of StarNet, GS-Detect, and SCFE modules on model parameters, FLOPs, and detection accuracy (mAP@50).

5. Conclusions

In this work, we present SGS-YOLO, a lightweight and high-performance detection framework designed for industrial-scale steel surface defect inspection. Built upon the YOLOv8n baseline, SGS-YOLO incorporates three structurally synergistic innovations: a StarNet-enhanced backbone for expressive and hierarchical feature extraction; a GS-Detect head for compact yet robust multi-scale prediction; and a parameter-free Self-Concatenated Feature Enhancement (SCFE) module for implicit feature augmentation and gradient stabilization.

Beyond quantitative gains, this work revealed a counterintuitive yet valuable phenomenon through targeted ablation. The SCFE module, which introduces structural redundancy via self-concatenation, consistently outperformed the more conventional channel expansion strategy (TCE), despite having fewer parameters and lower theoretical cost. This suggests that a well-designed architectural topology may yield greater gains than naive capacity scaling. These insights highlight the critical role of structural bias and training dynamics in modern deep network design. Therefore, SGS-YOLO and the SCFE-derived insights offer a transferable blueprint for lightweight, deployable detection systems, with clear potential for adoption and scalability across diverse industrial domains.

Future studies will focus on three main aspects. First, the reported speed is measured on a desktop GPU, and real-time performance on edge/embedded hardware has not been fully validated. We will benchmark SGS-YOLO on Jetson-class devices and optimize deployment through TensorRT acceleration and post-training/quantization-aware quantization (e.g., FP16/INT8), reporting latency, throughput, and resource footprints. Second, the detection accuracy for extremely small defects (smaller than 5 pixels) remains insufficient, which is particularly challenging under low contrast and cluttered backgrounds. Future work will investigate lightweight attention mechanisms and small-object-oriented designs (e.g., higher-

resolution/tiling inference, enhanced multi-scale feature interaction, and targeted data augmentation and supervision) to improve sensitivity to tiny defects in practical scenarios. Third, in cluttered scenes, the model can occasionally produce redundant/overlapping boxes, indicating reduced localization stability. We will explore improved label assignment strategies and stronger post-processing to suppress duplicate predictions while maintaining recall.

Funding

This work was supported by the Sichuan Science and Technology Program under Grant 2023ZHJY0009.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Wang, J., Chen, T., Xu, X., Zhao, L., Yuan, D., Du, Y., *et al.* (2024) An Improved Yolov8 Model for Strip Steel Surface Defect Detection. *Applied Sciences*, **15**, Article 52. https://doi.org/10.3390/app15010052

[2] Kou, X., Liu, S., Cheng, K. and Qian, Y. (2021) Development of a YOLO-V3-Based Model for Detecting Defects on Steel Strip Surface. *Measurement*, **182**, Article 109454. https://doi.org/10.1016/j.measurement.2021.109454

[3] Chu, Y., Yu, X. and Rong, X. (2024) A Lightweight Strip Steel Surface Defect Detection Network Based on Improved Yolov8. *Sensors*, **24**, Article 6495. https://doi.org/10.3390/s24196495

[4] Saklakoglu, N., Bolouri, A., Irizalp, S.G., Baris, F. and Elmas, A. (2021) Effects of Shot Peening and Artificial Surface Defects on Fatigue Properties of 50CrV4 Steel. *The International Journal of Advanced Manufacturing Technology*, **112**, 2961-2970. https://doi.org/10.1007/s00170-020-06532-y

[5] Zheng, X., Zheng, S., Kong, Y. and Chen, J. (2021) Recent Advances in Surface Defect Inspection of Industrial Products Using Deep Learning Techniques. *The International Journal of Advanced Manufacturing Technology*, **113**, 35-58. https://doi.org/10.1007/s00170-021-06592-8

[6] Wen, X., Shan, J., He, Y. and Song, K. (2023) Steel Surface Defect Recognition: A Survey. *Coatings*, **13**, Article 17. https://doi.org/10.3390/coatings13010017

[7] Pan, Y. and Zhang, L. (2022) Dual Attention Deep Learning Network for Automatic Steel Surface Defect Segmentation. *Computer-Aided Civil and Infrastructure Engineering*, **37**, 1468-1487. https://doi.org/10.1111/mice.12792

[8] Ren, S., He, K., Girshick, R. and Sun, J. (2017) Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **39**, 1137-1149. https://doi.org/10.1109/tpami.2016.2577031

[9] He, K., Gkioxari, G., Dollar, P. and Girshick, R. (2017) Mask R-CNN. 2017 *IEEE International Conference on Computer Vision* (*ICCV*), Venice, 22-29 October 2017, 2961-2969. https://doi.org/10.1109/iccv.2017.322

[10] Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. (2016) You Only Look Once: Unified, Real-Time Object Detection. 2016 *IEEE Conference on Computer Vision*

and Pattern Recognition (*CVPR*), Las Vegas, 27-30 June 2016, 779-788. https://doi.org/10.1109/cvpr.2016.91

[11] Li, C., Li, L., Jiang, H., Weng, K., Geng, Y., Li, L., Ke, Z., Li, Q., Cheng, M. and Nie, W. (2022) YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications. arXiv:2209.02976.

[12] Wang, C., Bochkovskiy, A. and Liao, H.M. (2023) YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors. 2023 *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (*CVPR*), Vancouver, 17-24 June 2023, 7464-7475. https://doi.org/10.1109/cvpr52729.2023.00721

[13] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C., *et al.* (2016) SSD: Single Shot Multibox Detector. In: Leibe, B., Matas, J., Sebe, N. and Welling, M., Eds., *Lecture Notes in Computer Science*, Springer International Publishing, 21-37. https://doi.org/10.1007/978-3-319-46448-0_2

[14] Tan, M., Pang, R. and Le, Q.V. (2020) Efficientdet: Scalable and Efficient Object Detection. 2020 *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (CVPR), Seattle, 13-19 June 2020, 10781-10790. https://doi.org/10.1109/cvpr42600.2020.01079

[15] Jocher, G., Chaurasia, J., Qiu, A. and Stoken, K. (2023) Ultralytics YOLOv8. GitHub Repository.

[16] Ameli, Z., Nesheli, S.J. and Landis, E.N. (2024) Deep Learning-Based Steel Bridge Corrosion Segmentation and Condition Rating Using Mask RCNN and YOLOv8. *Infrastructures*, **9**, Article 3. https://doi.org/10.3390/infrastructures9010003

[17] Lu, Z., Chengao, Z., Lu, L., Yan, Y., Jun, W., Wei, X., *et al.* (2025) Star-YOLO: A Lightweight and Efficient Model for Weed Detection in Cotton Fields Using Advanced YOLOv8 Improvements. *Computers and Electronics in Agriculture*, **235**, Article 110306. https://doi.org/10.1016/j.compag.2025.110306

[18] He, Y., Song, K., Meng, Q. and Yan, Y. (2020) An End-to-End Steel Surface Defect Detection Approach via Fusing Multiple Hierarchical Features. *IEEE Transactions on Instrumentation and Measurement*, **69**, 1493-1504. https://doi.org/10.1109/tim.2019.2915404

[19] Jain, S., Seth, G., Paruthi, A., Soni, U. and Kumar, G. (2022) Synthetic Data Augmentation for Surface Defect Detection and Classification Using Deep Learning. *Journal of Intelligent Manufacturing*, **33**, 1007-1020. https://doi.org/10.1007/s10845-020-01710-x

[20] Liu, C. and Cheng, H. (2024) Steel Surface Defect Detection Based on YOLOv8-TLC. *Applied Sciences*, **14**, Article 9708. https://doi.org/10.3390/app14219708

[21] Li, Z., Tian, X., Liu, X., Liu, Y. and Shi, X. (2022) A Two-Stage Industrial Defect Detection Framework Based on Improved-YOLOv5 and Optimized-Inception-Resnetv2 Models. *Applied Sciences*, **12**, Article 834. https://doi.org/10.3390/app12020834

[22] Wu, Y., Chen, R., Li, Z., Ye, M. and Dai, M. (2024) SDD-YOLO: A Lightweight, High-Generalization Methodology for Real-Time Detection of Strip Surface Defects. *Metals*, **14**, Article 650. https://doi.org/10.3390/met14060650

[23] Wei, M., Chen, B., Liu, J., Yuan, N., Liu, J. and Ji, Z. (2024) AEDN-YOLO: An Efficient One-Stage Detection Network for Strip Steel Surface Defects. *Engineering Research Express*, **6**, Article 035415. https://doi.org/10.1088/2631-8695/ad681d

[24] Kong, S., Kong, Y., Chi, X., Feng, X. and Ma, L. (2025) A TBD-YOLO-Based Surface Defect Detection Method for Hot Rolled Steel Strips. *Russian Journal of Nondestructive Testing*, **61**, 137-149. https://doi.org/10.1134/s1061830924603192

[25] Liu, R., Huang, M., Gao, Z., Cao, Z. and Cao, P. (2023) MSC-DNet: An Efficient Detector with Multi-Scale Context for Defect Detection on Strip Steel Surface. *Measurement*, **209**, Article 112467. https://doi.org/10.1016/j.measurement.2023.112467

[26] Yang, S., Wang, X., Qian, X., Yu, Y. and Jin, W. (2023) Trident-LK Net: A Lightweight Trident Structure Network with Large Kernel for Muti-Scale Defect Detection. *IEEE Access*, **11**, 131073-131080. https://doi.org/10.1109/access.2023.3333918

[27] Wen, Y. and Wang, L. (2024) YOLO-SD: Simulated Feature Fusion for Few-Shot Industrial Defect Detection Based on YOLOv8 and Stable Diffusion. *International Journal of Machine Learning and Cybernetics*, **15**, 4589-4601. https://doi.org/10.1007/s13042-024-02175-7

[28] Liu, Y., Zhao, Y., Zhang, X., Wang, X., Lian, C., Li, J., *et al.* (2023) Mobilesam-Track: Lightweight One-Shot Tracking and Segmentation of Small Objects on Edge Devices. *Remote Sensing*, **15**, Article 5665. https://doi.org/10.3390/rs15245665

[29] Zhao, Y., Ju, Z., Sun, T., Dong, F., Li, J., Yang, R., *et al.* (2023) TGC-YOLOv5: An Enhanced YOLOv5 Drone Detection Model Based on Transformer, GAM & CA Attention Mechanism. *Drones*, **7**, Article 446. https://doi.org/10.3390/drones7070446

[30] Ma, X., Dai, X., Bai, Y., Wang, Y. and Fu, Y. (2024) Rewrite the Stars. 2024 *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (*CVPR*), Seattle, 16-22 June 2024, 5694-5703. https://doi.org/10.1109/cvpr52733.2024.00544

[31] Tian, Z., Shen, C., Chen, H. and He, T. (2019) FCOS: Fully Convolutional One-Stage Object Detection. 2019 *IEEE/CVF International Conference on Computer Vision* (*ICCV*), Seoul, 27 October 2019-2 November 2019, 9626-9635. https://doi.org/10.1109/iccv.2019.00972

[32] Wu, Y. and He, K. (2018) Group Normalization. In: Ferrari, V., Hebert, M., Sminchisescu, C. and Weiss, Y., Eds., *Lecture Notes in Computer Science*, Springer International Publishing, 3-19. https://doi.org/10.1007/978-3-030-01261-8_1

[33] Huang, G., Liu, Z., Van Der Maaten, L. and Weinberger, K.Q. (2017) Densely Connected Convolutional Networks. 2017 *IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*), Honolulu, 21-26 July 2017, 4700-4708. https://doi.org/10.1109/cvpr.2017.243

[34] Gastaldi, X. (2017) Shake-Shake Regularization. arXiv:1705.07485.

[35] Wang, C., Yeh, I. and Mark Liao, H. (2024) YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information. In: Leonardis, A., Ricci, E., Roth, S., Russakovsky, O., Sattler, T. and Varol, G., Eds., *Lecture Notes in Computer Science*, Springer, 1-21. https://doi.org/10.1007/978-3-031-72751-1_1

[36] Wang, A., Chen, H., Liu, L., Chen, K., Lin, Z., Han, J. and Ding, G. (2024) YOLOv10: Real-Time End-to-End Object Detection. arXiv:2405.14458.

[37] Jocher, G., Chaurasia, J., Qiu, A. and Stoken, K. (2024) Ultralytics YOLOv11. GitHub repository.

[38] Jocher, G., Chaurasia, J., Qiu, A. and Stoken, K. (2025) Ultralytics YOLOv12. GitHub repository.

[39] Zhao, Y., Lv, W., Xu, S., Wei, J., Wang, G., Dang, Q., *et al.* (2024) DETRs Beat Yolos on Real-Time Object Detection. 2024 *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (*CVPR*), Seattle, 16-22 June 2024, 16965-16974. https://doi.org/10.1109/cvpr52733.2024.01605