Advances in Science and Technology Research Journal

*Advances in Science and Technology Research Journal*, 2026, 20(3), 29–46
https://doi.org/10.12913/22998624/212538
ISSN 2299-8624, License CC-BY 4.0

Received: 2025.09.10
Accepted: 2025.11.13
Published: 2026.02.01

# A comparative assessment of you only look once nano architectures for high-speed and accurate steel surface inspection

Fakhra Aftab[1], Muhammad Babar Ali[1], Muhammad Maaz Khan[1], Aaliyan Mansoor[1], Mohib Ud Din[1], Majida Kazmi[1,2*]

[1] Computer & Information Systems Engineering Department, NED University of Engineering and Technology, Main University Road, Karachi, Pakistan
[2] Neurocomputation Lab, National Centre of Artificial Intelligence, NED University of Engineering and Technology, Main University Road, Karachi, Pakistan
* Corresponding author's e-mail: majidakazmi@neduet.edu.pk

## ABSTRACT

Automated detection of surface defects in steel is critical for ensuring product quality and operational efficiency in modern manufacturing. This study presents a unified framework for real-time steel surface inspection using five lightweight YOLO (you only look once) nano architectures: YOLOv5n, YOLOv8n, YOLOv11n, YOLOv12n, and YOLOv13n. Unlike prior studies that focused on optimizing individual models, this work conducts a cross-generation comparative analysis, introducing the first evaluation of YOLOv13 for steel surface defect detection. The proposed framework integrates systematic dataset preparation, model training on benchmark and custom industrial datasets, and detailed performance assessment under stringent IoU thresholds and real-time inference conditions. Experimental results reveal that inference performance depends more on architectural efficiency, hardware utilization, and software-level optimization than on model size alone. Basic augmentation techniques, such as flipping and rotation, further enhance the detection of small and hard-to-capture defects. Among all models, YOLOv13n achieves the fastest inference of 303.03 FPS with competitive accuracy, demonstrating exceptional suitability for real-time, edge-based industrial deployments. The findings provide valuable empirical insights for selecting efficient architectures in automated visual inspection systems.

**Keywords:** machine vision, steel surface defects, light-weight YOLO variants, architectural differences, model performances.

## INTRODUCTION

Steel surfaces, such as steel rods and hot-rolled steel strips, are integral to industries like automotive, aerospace, manufacturing, and construction. However, it was shown that these surfaces are susceptible to defects, including scratches, inclusions, cracks, and pitted surfaces, which compromise quality, safety, and operational efficiency [1, 2]. Ensuring the quality of steel products is an important requirement of modern industrial manufacturing. Traditional defect detection methods are manual, labour-intensive and prone to inconsistency [3], encouraging a shift toward automated defect detection methods. Advancements in deep learning, especially convolutional neural networks, have led to the YOLO (you only look once) model family, one of the most popular for object identification. Recent work applying YOLO architectures for welding surface inspection has highlighted the potential of these models for defect detection and automation [4]. However, using the latest YOLO models to find defects in steel surfaces is still an active field of research.

Multiple studies adopted and enhanced YOLOv5, v8, and v11 models for steel surface defect detection, each proposing unique improvements aiming to enhance accuracy and speed. Cheng et al. [1] proposed EC-YOLO, an improved version of the YOLO-V5-based model, using an efficient channel attention bottleneck (EB) module for better feature extraction, and Context Transformation Networks block for enhanced feature fusion and context modeling, for detecting small and elongated defects. They used publicly available datasets; GC10-DET, NEU-DET and SLD-DET, and achieved mean Average Precision (mAP) values of 71%, 83% and 87.5%, respectively, on the improved model, but with FPS of 91.74 on NEU-DET and 64.10 on GC10-DET, it also increased the parameters of the model by around 116.81 MB, slowing down the detection efficiency. Wang et al. [2] proposed a lightweight RDB-YOLO model for detecting steel surface defects based on YOLOv5n. Using Receptive Field Enhancement (RFE) to compensate for accuracy loss due to noise, deformable convolution network (DCN) to detect irregularly shaped defects, and Bi-level Routing Attention (BRA) to improve detection of small targets. Trained on the benchmark dataset, NEU-DET, and attained an mAP@0.5 score of 0.821, an improvement of 0.51, but with an FPS of 60.4 FPS on an NVIDIA RTX 3090. Yang et al. [3] introduced a lightweight spatial attention module to extract effective information and designed a new feature fusion network, Amsf, and added the FAM attention module to strengthen the capability to detect on a lightweight surface anomaly detection algorithm leveraging the modified YOLOv5 architecture. Experiments were conducted using the NEU-DET dataset, which gave an average detection accuracy of 81.97%. Yu et al. [5] customized a YOLO-v5 model, LCG-YOLO, for real-time anomaly detection in metallic parts. The LSandGlass (LSG) module removes the low-resolution feature layer to reduce loss of critical details. The experiment was executed on a self-made dataset, on which the mAP improved to 0.955, and an increase of 21 fps, but it cost them an increase of 3.7% in parameter size, compared to the original YOLO-v5. Ma et al. [6] developed a lightweight approach for steel surface anomaly detection to reduce computational complexity while improving the accuracy, based on an improved YOLOv8 model. Model training and evaluation were conducted on the NEU-DET dataset containing 1800 images equally distributed among six defect classes. The model used GhostNet as a backbone to reduce complexity, MPCA was added to enhance feature extraction, and replaced IoU loss with SIoU for improved bounding box accuracy. The model was trained and evaluated on the system with Intel i9-9900k CPU and NVIDIA RTX 2080Ti GPU, using PyTorch 2.1.1. The improved YOLOv8n variant achieved an mAP@0.5 of 78.6%, outperforming YOLOv8n baseline (77.4%) and YOLOv8s (77.9%). The system also achieved an FPS of 171.5, which is the highest among compared models and a parameter count of as low as 2.04 M and 4.5 MB. Liu et al. [7] introduced a modified version of YOLOv8, SLF-YOLO, for steel surface anomaly detection. They modified the model by including an SC_C2f module to improve feature representation, Light-SSF_Neck Structure to improve multi-scale feature extraction and FIMetal-IoU Loss Function to enhance the recognition of small defect features. The SLF-YOLO model achieved the mAP@0.5 of 80.0% compared to the YOLOv8's 75.9% on the NEU-DET (1800 images with 6 classes) dataset. On the AL10-DET (4652 images with 10 classes) dataset, SLF-YOLO achieved the mAP@0.5 of 86.8%, and reduced the parameter count to 9.65 M compared to YOLOv8's 11.12 M parameters. NVIDIA RTX 3090 GPU was used to train the model, with CUDA 11.8 and cuDNN 7.0. Huang et al. [8] introduced DM-YOLOv11, using a Dynamic Weight Reparameterization Module, MPCA, and the Wise-IoUv2 loss function to enhance regression robustness. The experiments were carried out on benchmark datasets; NEU-DET and GC10-DET, achieving an mAP score of 0.806 and 0.714, respectively and reduced the model size to 10.8 M parameters. Sun et al. [9] proposed MCH-YOLOv12, an enhanced lightweight version of YOLOv12 that integrates a MultiScaleGhost module for improved multi-scale feature extraction, a Spatial-Channel Collaborative Gated Linear Unit (SCCGLU) for refined attention to defect details, and a Hybrid Detection Head combining anchor-based and anchor-free strategies to better handle diverse defect sizes and shapes. The improved model achieved higher accuracy with reduced complexity, reaching 89.2 mAP (6.8 M parameters) on the NEU-DET dataset and 95.0 mAP (7.0 M parameters) on the APDDD dataset, outperforming the baseline YOLOv12. Zhao et al. [10] introduced RT-DETR, a real-time Transformer-based detector that eliminates Non-Maximum Suppression (NMS) through an efficient hybrid encoder and Uncertainty-Minimal Query Selection, achieving 53.1% AP at 108 FPS (ResNet-50) and 54.3% AP at 74 FPS (ResNet-101) on the COCO dataset, outperforming YOLOv8-L. Similarly, Xu et al. [11] proposed PP-YOLOE, an anchor-free detector with the CSPRepResStage backbone, Task Alignment Learning (TAL), and an Efficient Task-aligned Head (ET-head), attaining 51.4 mAP at 78.1 FPS on the COCO dataset, improving both accuracy (+1.9 AP) and speed (+13.35%) over PP-YOLOv2.

YOLOv13, which is presented in [13], and YOLOv12 are recent models, while there are some studies utilizing YOLOv12 for steel surface defect detection, there are none utilizing YOLOv13. Most of the studies in the literature emphasize optimizing the original model(s) and assessing their performance across different scenarios like IFEM-YOLOv13 proposed by Feng et al. [14] is a modified version of YOLOv13 for underwater target detection, achieving a 96.8% mAP@0.5 on the j-EDI (Underwater Litter Image) dataset. YOLOv13-Cone-Lite proposed by Wang et al. [15] is a modified version of YOLOv13 for real-time detection of traffic cones for autonomous formula racing cars, achieving an mAP@50 of 92.9% with a model size of 8.7 MB.

Our critical review of literature suggests that researchers have introduced several notable improvements to YOLO-based architectures for steel surface defect detection; however, most of these efforts have concentrated on optimizing individual models rather than performing systematic, cross-generation evaluations. Despite their promising efficiency, the most recent lightweight architectures, such as YOLOv12 and YOLOv13, are largely unexplored within the domain of quality inspection. Moreover, existing studies rarely integrate standardized benchmark datasets with real industrial data. The majority of existing research has focused on increasing detection accuracy, with limited attention given to the collective influence of architectural design, hardware, and software factors on real-time inference performance. Furthermore, previous works largely failed to empirically demonstrate that inference speed and model size are not directly proportional. Our research study addresses these gaps and demonstrates that baseline models can also achieve competitive detection accuracies.

In this paper, we present a comprehensive comparative study of various YOLO nano variants for the automated detection and localization of surface defects on steel products. The models are trained and evaluated on publicly available benchmark datasets and a custom industrial dataset collected from a local manufacturing facility, respectively, thereby addressing both standard and real-world inspection scenarios. Although the experiments were validated on a custom dataset, it was carefully designed to reflect real production conditions. This ensures reliable model performance within its intended deployment context. While such specialization limits broader generalization, the YOLO-nano architectures offer strong transfer learning capabilities. With additional fine-tuning or re-training on small samples from new environments within the steel domain, the very same models can be successfully generalized.

We systematically analyze each model to identify their respective strengths and limitations. The insights derived from this study are intended to guide researchers and industry practitioners in selecting and optimizing deep learning models for automated steel surface inspection scenarios. The major contributions of this paper are listed below:

1. Developed a comprehensive framework for automated quality inspection, encompassing systematic dataset preparation, the training of various YOLO-based models and their validation for precise defect detection and localization.
2. Evaluated the framework by comparing multiple lightweight YOLO variants, emphasizing precision under stringent IoU thresholds and real-time performance in terms of FPS to ensure an optimal balance between accuracy and efficiency.
3. Established that inference speed is predominantly determined by architectural efficiency, hardware-level utilization, and software optimizations, highlighting the limited correlation between model size and real-time performance.

The methodology adopted as part of this research work is systematic and iterative, ensuring the achievement of the project's objectives. Figure 1 shows the machine vision pipeline with the following steps:

- Data collection: Using two publicly available datasets and a custom dataset collected from a manufacturing facility.
- Data preprocessing: Combining datasets to achieve best possible results, augmenting datasets to make the trained models robust, splitting the dataset into subsets for training, testing and validation of the models.
- Model training and development: Training the nano variants five chosen YOLO model architectures on the datasets.

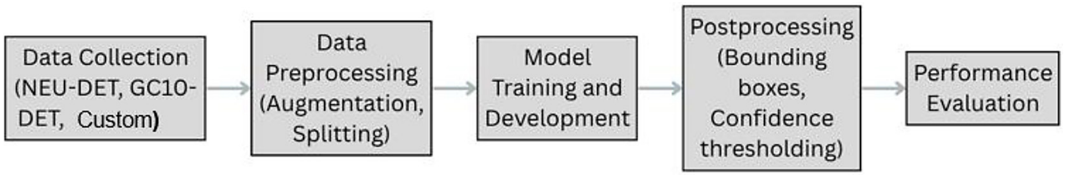
Data Collection (NEU-DET, GC10-DET, Custom)
Data Preprocessing (Augmentation, Splitting)
Model Training and Development
Postprocessing (Bounding boxes, Confidence thresholding)
Performance Evaluation

**Figure 1.** Proposed machine vision pipeline

- Postprocessing: Drawing bounding boxes around the defects and only allowing results above a specified confidence threshold.
- Performance evaluation: Evaluate and compare the trained models based on various performance metrics.

## DATASET PREPARATION

### Dataset overview

#### *NEU-DET (NorthEastern University Dataset)*

It is a dataset available at [16] and further discussed in [17], it is a collection of grayscale images featuring surface defects on hot-rolled steel strips categorized into six major defect types: scratch, patch, rolled in scale, pitted surface, inclusion, and crazing as shown in Figure 2, these defects represent the most frequent and impactful surface anomalies encountered in industrial hot-rolled steel production. Specifically, scratches and rolled-in scales commonly result from friction and oxide formation on steel surfaces, inclusions originate during the steelmaking process, while pitted surfaces and patches arise from irregular rolling conditions. These categories collectively encompass the majority of defect types that significantly affect product quality and production efficiency, making them the most relevant for industrial inspection applications. The dataset contains 300 grayscale images for each defect type, totaling 1800 images.

### GC10-DET dataset

It is a dataset available at [18] and further discussed in [19], it is a collection of images of steel surface defects on hot-rolled steel strips, categorized into ten commonly occurring steel surface defects: punching hole, welding line, crescent gap, water spot, oil spot, silk spot, inclusion, rolled pit, crease, and waist folding as shown in Figure 3. The dataset consists of 2,300 images, with some images containing multiple defects. There are 3,563 labelled objects. Each of these objects represents an individual defect annotated with a bounding box and its corresponding class label. These annotated instances serve as the ground truth for model training and evaluation, enabling the network to learn precise localization and classification of defects on steel surfaces.

#### *Custom dataset*

It is a dataset collected from a local manufacturing facility consisting of images of surface defects on steel piston rods used in vehicle shock absorbers, it is a binary class dataset with two classes: defected and non-defected. Figure 4 highlights the real-world defect types included in the dataset: dent, material pit, process pit, line scratch, plating crack, and roughness, where each defect is zoomed in on and circled in red. In this work, 3445 images of only dent images were taken into account.

### Data preprocessing

#### *Combining datasets*

We combined the two publicly available datasets, NEU-DET and GC10-DET, to create a dataset comprising 13 defect types, namely: crease, crescent gap, inclusion, oil spot, patches, pitted surface, punching hole, rolled in scale, scratches, silk spot, waist folding, water spot, and welding line. This was done to create a dataset with improved diversity, resulting in better usability. All the images were resized to 416×416 to maintain a balance between defect visibility and computational efficiency. This resolution offers an optimal trade-off by preserving sufficient detail for detecting small surface defects while keeping the computational cost manageable. Moreover, the resizing followed standard aspect ratio adjustments to ensure consistency with YOLO input requirements. Larger image sizes significantly increase processing time and memory consumption, whereas smaller resolutions tend to blur fine defect details,

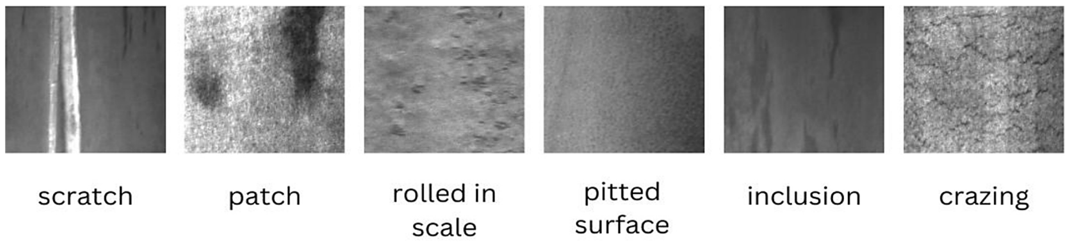

**Figure 2.** NEU-DET dataset six defect types

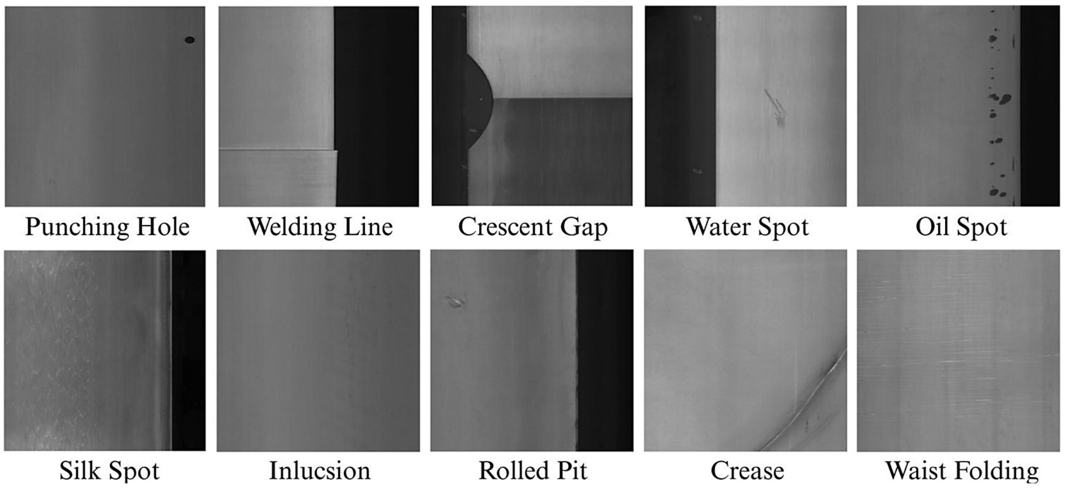

**Figure 3.** GC10-DET dataset defect types

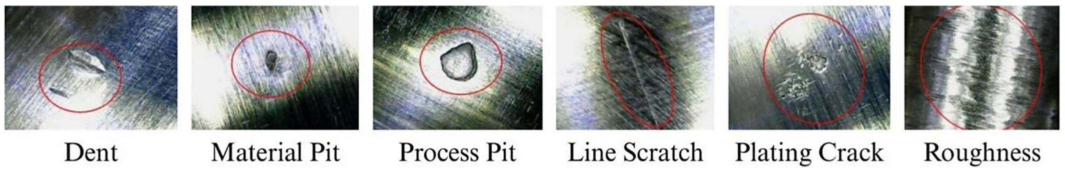

**Figure 4.** Custom dataset defect types

reducing detection accuracy. Therefore, the 416 × 416 dimension was selected as the most practical choice for both accuracy and efficiency.

### *Data annotation*

Annotation plays a vital role in the dataset preparation process required for effective model training. In this study, images were manually annotated using bounding boxes to precisely mark defect locations, and all annotations were validated by domain experts to ensure spatial accuracy and correct class labeling. The Roboflow tool was used for efficient and consistent image annotations. Each image has its own.txt file with a single line for each bounding box, the annotation format for YOLO models is as follows: *class_id center_x center_y width height.*

The data.yaml file includes the configuration information that the model uses to find the images and link class names to class ids during training, testing, or validation.

### *Data augmentation*

The model's performance is greatly impacted by the variety and size of the training data [20], so the data has been augmented by creating modified copies of the images as shown in Figure 5, the original image without any augmentation goes through the data augmentation process using different augmentation techniques to create three different modified copies from one image. The following augmentation techniques have been utilized:

- Flip: randomly flip an image vertically(in the up/down direction) or horizontally(in the left/

right direction), helps the model be insensitive to the orientation of the subject.
- Rotation: Applies a small, random rotation to the image within a specific range of -10° to +10° (includes counter-clockwise and clockwise rotations).

After performing data augmentation the original dataset of 3445 images was increased to 5857 images.

### *Splitting the dataset*

We split the dataset into three subsets:
- Training set: 82% of the images, 4824 were used to train the machine learning model.
- Validation set: 12% of the images, 689 were used to fine-tune the hyperparameters and monitor the model and prevent overfitting.
- Testing set: Remaining 6%, 344 were reserved to evaluate the final model's accuracy and performance.

## MODEL SELECTION, TRAINING AND EVALUATION

The choice of multiple YOLO versions enables a comprehensive comparative evaluation, as each iteration introduces new architectural improvements such as better backbone designs, advanced feature aggregation, or more effective training strategies. Evaluating these models on the same datasets allows us to systematically analyze the impact.

In this present work, five contemporary and widely adopted YOLO architectures including YOLOv5n, YOLOv8n, YOLOv11n, YOLOv12n and YOLOv13n were selected. These models are particularly suitable for industrial inspection scenarios and deployment on devices with limited computational resources, as they provide an effective trade-off between detection accuracy, inference speed, and computational cost. The nano variants of YOLO architectures are designed with significantly fewer parameters, making them ideal for hardware-constrained environments. As demonstrated in [21], the YOLOv8n model achieves faster inference compared to its small counterpart (YOLOv8s) with only a minor reduction in accuracy. Furthermore, a recent comparative study involving YOLOv8n, YOLOv10n, and YOLOv12n confirmed that these lightweight variants are well-suited for industrial applications, offering high throughput and real-time performance while maintaining low resource consumption [22].

### Model architecture

Although all the models come with their own distinct innovations in design, they all are made up of a common structure of three main components, three basic components, namely backbone, neck, and head as depicted in Figure 6. Visual characteristics are extracted from the raw input image by the backbone, forming a hierarchy of low-level (textures, edges) and high-level (objects, shapes) features, to make up the foundation of the model's understanding. Earlier versions of YOLO used DarkNet as the backbone while newer versions utilize CSPNet or convolutional blocks designed for optimized deployment with improved computational efficiency.

To improve the identification of objects of different sizes, the neck combines these characteristics over several scales. Semantic (high-level) and spatial (low-level) information flow is enhanced by the usage of Feature Pyramid Networks (FPN) and Path Aggregation Networks (PAN). increasing the model's accuracy in identifying both huge, clearly defined flaws and tiny, subtle ones.

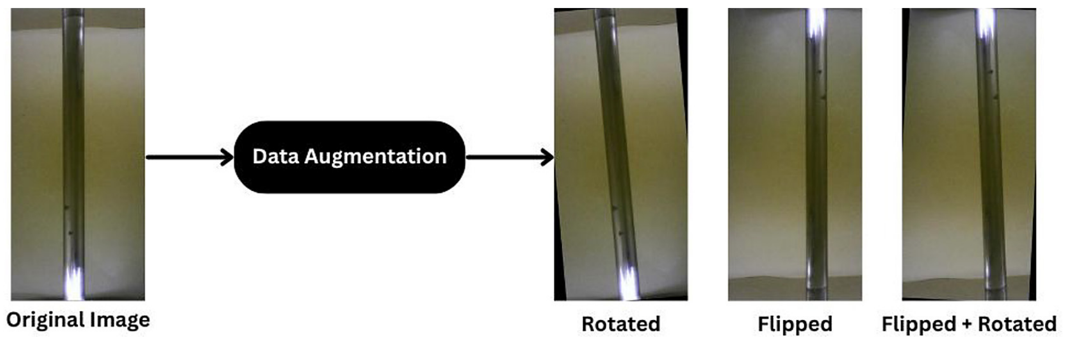

**Figure 5.** Example results of data augmentation

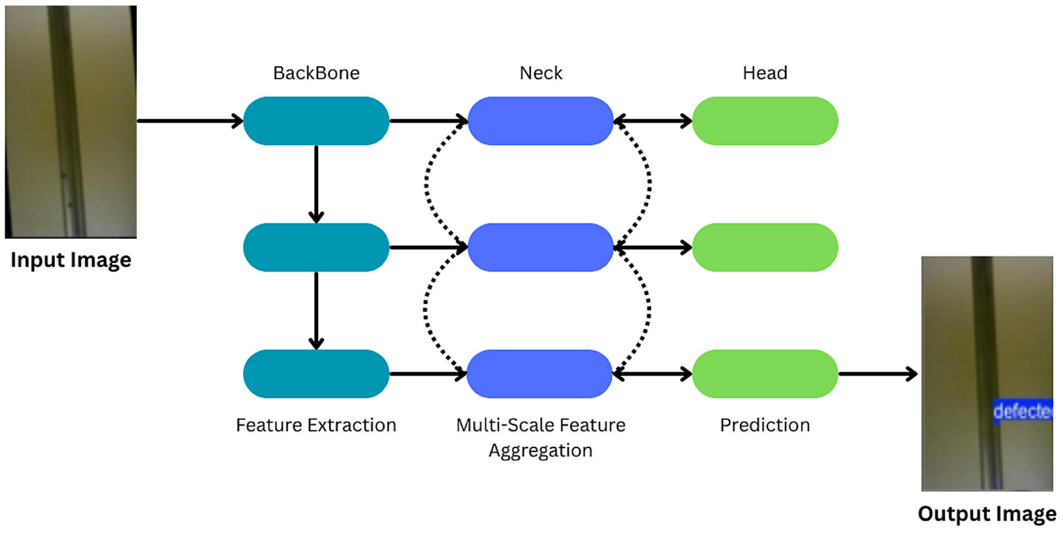

**Figure 6.** Common structure of the YOLO architecture

The head ultimately produces predictions that include bounding box coordinates, class labels, and confidence scores [23]. Depending on the YOLO version, the head may use anchor-based detection (as in YOLOv5) or an anchor-free approach (as introduced in YOLOv8 and beyond). The head is crucial in producing fast detections allowing the model to maintain real time performance.

The architectural evolution of YOLO models has introduced several enhancements within this structure, to improve efficiency and detection accuracy. One of the early major improvements were the Cross Stage Partial (CSP) connections which separate feature maps into two paths to be merged later on in the model's pipeline; this has resulted in a reduction in computational complexity and accuracy.

The Efficient Layer Aggregation Network (ELAN) added in later versions such as YOLOv7, improves the organization of convolutional blocks for better gradient propagation across deep layers. Allowing the network of the model to be deeper and wider without the problem of vanishing gradients, thereby enhancing feature learning.

The Bidirectional Feature Pyramid Network (BiFPN) was introduced by Tan et al. [24] to improve the fusion of features at various scales; it differs from the standard FPN as it includes weighted connections and bi-directional pathways, allowing the model to adaptively learn the significance of different feature levels. This is advantageous in detecting defects of various sizes from dents to small scratches.

Attention mechanisms have also been introduced in newer versions of YOLO, such as Squeeze and Excitation (SE) or Convolutional Block Attention Module (CBAM) which consists of a Channel Attention Module (CAM) and a Spatial Attention Module (SAM), both modules allow the estimation of attention weights for the precise refinement of feature maps to ensure more attention towards more relevant spatial regions and channels, and less focus on less important information. Further improvements have been made in the activation layers with enhanced activation functions such as Sigmoid Linear Unit (SiLU) or LeakyReLU (Rectified Linear Unit) which improves upon the standard ReLU function offering smoother gradients. Recent YOLO models contain decoupled detection heads, wherein classification and localization tasks are executed in separate branches which have significantly improved detection accuracy, speed, and deployment efficiency across different variants as discussed in [25]. Table 1 presents a comprehensive overview of the YOLO model evolution, outlining the core innovations and architectural enhancements introduced in each version:

## Hyperparameter tuning and model training

All the models were trained under the following consistent hyperparameters to guarantee unbiased comparison:

- Epochs: All models were trained for 300 epochs with patience set at 50, i.e., the training stops earlier if the mAP is the same for 50 epochs.

**Table 1.** YOLO models' architectural modifications

| YOLO variant | Architectural modifications | Key strengths |
|---|---|---|
| YOLOv5 | CSPDarknet backbone, PANet, SWISH activation | Balanced accuracy, speed, and deployment efficiency with minimal parameters |
| YOLOv8 | CSP+ELAN, BiFPN-style neck, anchor-free head | Improved accuracy with high throughput |
| YOLOv11 | R-ELAN backbone, PANet neck, lightweight modules | Fast convergence, lightweight, and offers a balanced trade-off of speed and accuracy |
| YOLOv12 | R-ELAN+Attention backbone, BiFPN neck, decoupled head | Small model footprint with competitive mAP |
| YOLOV13 | Light hybrid CNN, multiscale adaptive fusion | Exceptional inference speed while maintaining high mAP |

- Batch Size: 32 images per batch, balancing GPU memory utilization and training speed.
- All the rest of the hyperparameters were kept the same as the default.

## Performance evaluation indicators

The following performance metrics were evaluated to assess the trained models:

- mAP: Evaluates the overall detection accuracy by averaging precision across all classes as represented in Equation 1 [26].

$$mAP = \frac{1}{n}\sum_{k=1}^{n} AP_k \tag{1}$$

where: $AP_k$ – average precision of one class, $n$ – the number of classes.

- Precision: Indicates the proportion of correctly detected defects among all predicted defects. A high precision (close to 1) means that most of the predictions are correct with very few misclassifications. Mathematically, it is presented in Equation 2 [26].

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

- Recall: Measures the ratio of actual problems accurately identified by the model. A high recall value means the model rarely misses true detections as shown in Equation 3 [26].

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

where: $TP$ – the number of true positives,
$FN$ – the number of false negatives,
$FP$ – the number of false positives.

- Intersection over union (IoU): IoU thresholds are used to measure how well the model is able to localize an object as discussed in [26]. It is the ratio of the predicted bounding boxes' intersection area to the ground truth (annotated) bounding boxes' union area as shown in Equation 4. This quantifies the intersection between the expected and the actual bounding boxes, a perfect prediction would yield an IoU of 1, in practice a prediction is considered if its IoU exceeds a set threshold.

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union} \tag{4}$$

- Frames per second (FPS): It is also a key performance metric and a focus point of this study, it measures how many images a model can process and generate predictions for in one second. It evaluates the model's inference speed and is essential for real-time applications. such as industrial inspection. Calculated as shown in Equation 5 [5], a higher FPS indicates faster processing speeds and better efficiency.

$$FPS = \frac{Number\ of\ Images\ Processed}{Total\ Inference\ Time} \tag{5}$$

## Experimental setup

All models were trained using kaggle, with 2 T4 GPUs each with 15 GiB (Gibibyte) memory, 29 GiB of RAM. All YOLO models from YOLOv5 onwards are developed on the Pytorch framework with CUDA acceleration enabled for efficient GPU-based training and inference. Data preprocessing, augmentation, and post-processing routines were managed using Roboflow and the Albumentations and OpenCV libraries.

## RESULTS AND DISCUSSION

In this section, the results of the conducted experiments are discussed in detail, with emphasis on understanding the trade-offs between accuracy, speed, and computational cost for different YOLO architectures under industrial inspection conditions.

## Yolov5 results

Figure 7 shows the training evolution of the performance metrics of the YOLOv5n model, with separate graphs for the evolution of the following metrics: precision, recall, mAP@0.5, and mAP@0.5:0.95. The values of the metrics are on the y-axis and the training epochs are on the x-axis. The graph shows a sharp increase in mAP@0.5, precision and recall in the initial epochs. Precision and recall stabilize above 0.99 indicating reliable performance, mAP@0.5 nearly reaches 0.1, while mAP@05:0.95 shows consistent improvement and stabilises over 0.66.

Figure 8 shows the confusion matrix of the YOLOv5n model trained on the custom dataset, the x-axis represents the true-labels and the y-axis represents the predicted labels generated by the model. The diagonal elements (top left to bottom right) are the correctly classified input images and off-diagonal elements represent the misclassified input images. Both the classes in the matrix indicate an mAP@0.5 of 100. The darker shade of the deep blue color denotes higher correctness in the model's detections.

Figure 9 shows the inference results of the YOLOv5n model on the custom dataset. Where 4 different input images have been processed, in each image all the detected defects are highlighted

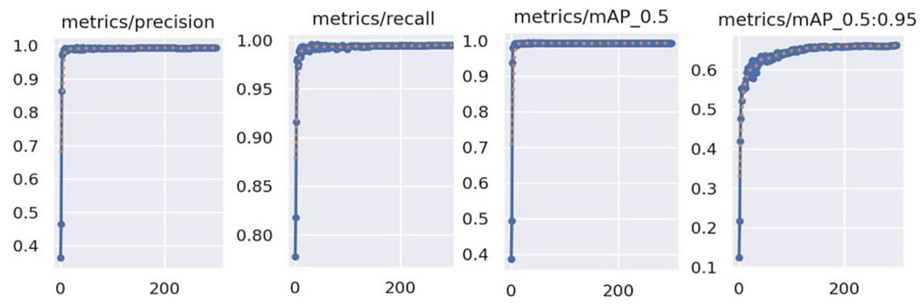

**Figure 7.** Training evolution of YOLOv5n on a custom dataset

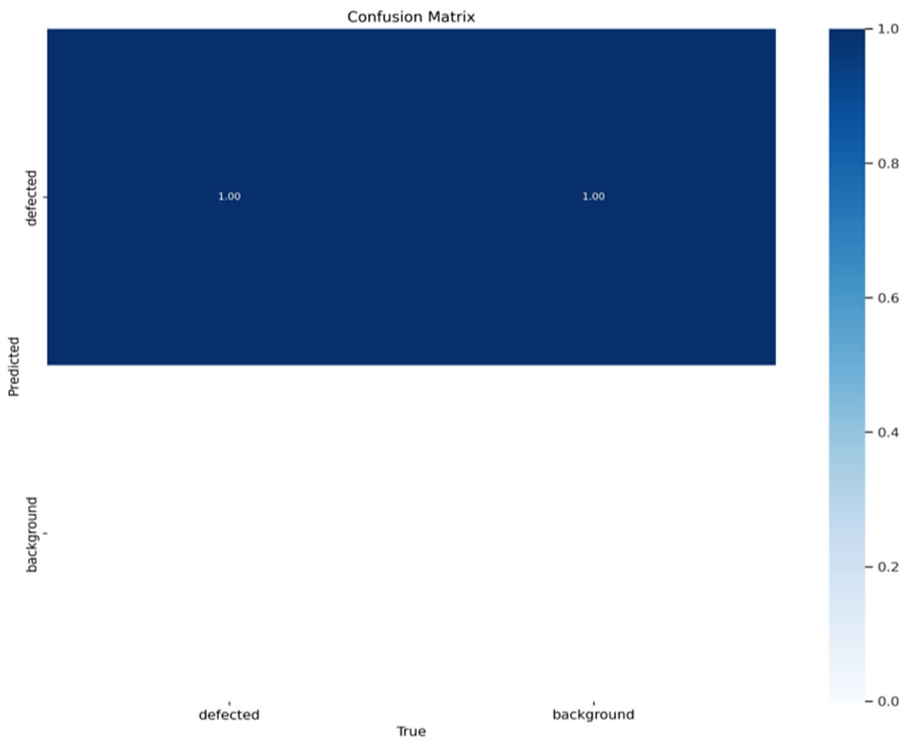

**Figure 8.** Confusion matrix of YOLOv5n

with bounding boxes and labels, along with their respective confidence scores.

## Yolov8 results

Figure 10 shows the training evolution of the YOLOv8n model's performance metrics with separate graphs for the evolution of the following metrics: precision, recall, mAP@0.5, and mAP@0.5:0.95. The values of the metrics are on the y-axis and the training epochs are on the x-axis. The graphs show a sharp increase in precision, recall, and mAP@0.5 in the initial epochs, and rapid stabilization above 0.99, indicating reliable performance, whereas mAP@0.5:0.9 rises gradually above 0.6.

Figure 11 shows the confusion matrix of the YOLOv8n model trained on the custom dataset, the x-axis represents the true-labels and the y-axis represents the predicted labels generated by the model. The diagonal elements (top left to bottom right) are the correctly classified input images and off-diagonal elements represent the misclassified input images. Both the classes in the matrix indicate an mAP@0.5 of 100. The darker shade of the deep blue color denotes higher correctness in the model's detections.

Figure 12 shows the inference results of the YOLOv8n model on the custom dataset. Where 4 different input images have been processed, in each image all the detected defects are highlighted with bounding boxes and labels, along with their respective confidence scores.

## Yolov11 results

Figure 13 shows the training evolution of the performance metrics of the YOLOv11n model with separate graphs for the evolution of the following metrics: precision, recall, mAP@0.5, and mAP@0.5:0.95. The values of the metrics are on the y-axis and the training epochs are on the x-axis. The model's mAP@0.5 converges gradually to achieve near perfect performance. Precision and recall reach above 0.99 demonstrating very strong detections with minimal ambiguities. mAP@0.5:0.95 rises steadily indicating the model can handle stricter IoU thresholds.

Figure 14 shows the confusion matrix of the YOLOv11n model trained on the custom dataset, the x-axis represents the true-labels and the y-axis represents the predicted labels generated by the model. The diagonal elements (top left to bottom right) are the correctly classified input images and off-diagonal elements represent the misclassified input images. Both the classes in the matrix indicate an mAP@0.5 of 100. The darker shade of the deep blue color denotes higher correctness in the model's detections.

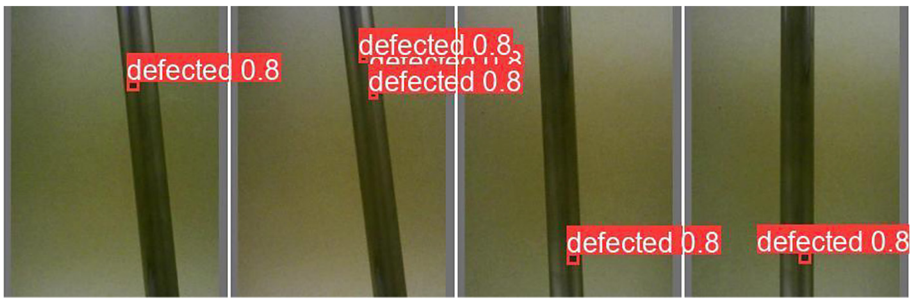

**Figure 9.** Inference images of YOLOv5n on a custom dataset

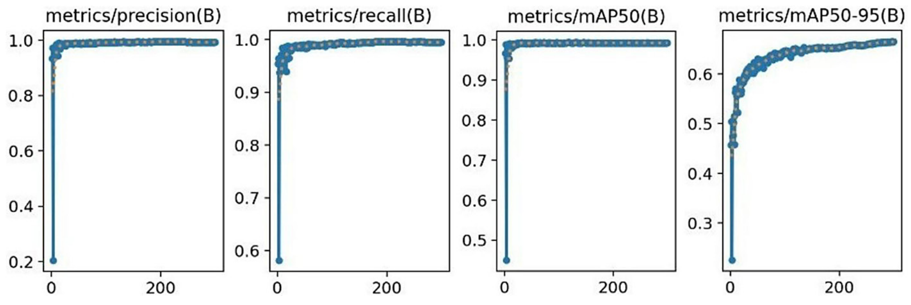

**Figure 10.** Training evolution of YOLOv8n on a custom dataset

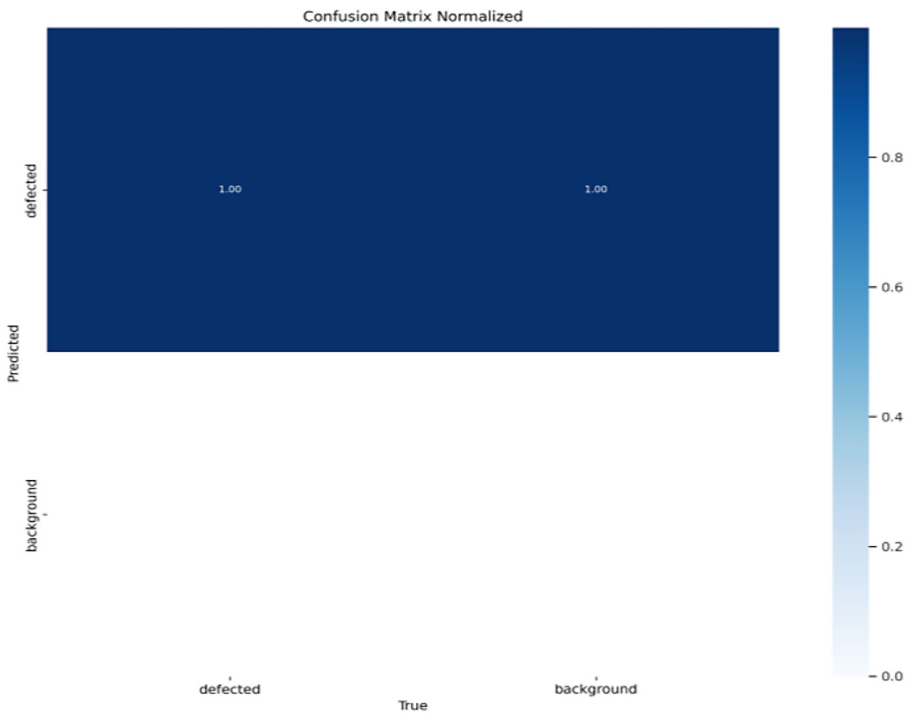

**Figure 11.** Confusion Matrix of YOLOv8n

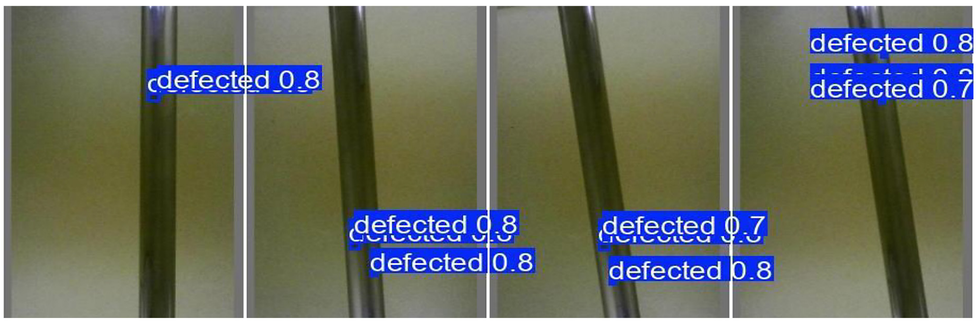

**Figure 12.** Inference images of YOLOv8n on a custom dataset

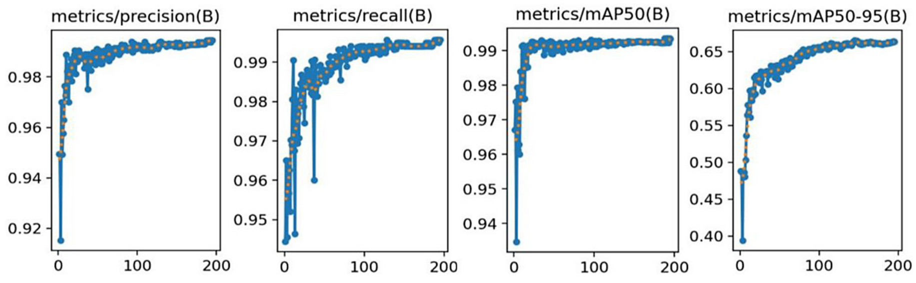

**Figure 13.** Training evolution of YOLOv11n on a custom dataset

Figure 15 shows the inference results of the YOLOv11n model on the custom dataset. Where 4 different input images have been processed, in each image all the detected defects are highlighted with bounding boxes and labels, along with their respective confidence scores.

## Yolov12 results

Figure 16 shows the training evolution of the performance metrics of the YOLOv12n model with separate graphs for the evolution of the following metrics: precision, recall, mAP@0.5, and

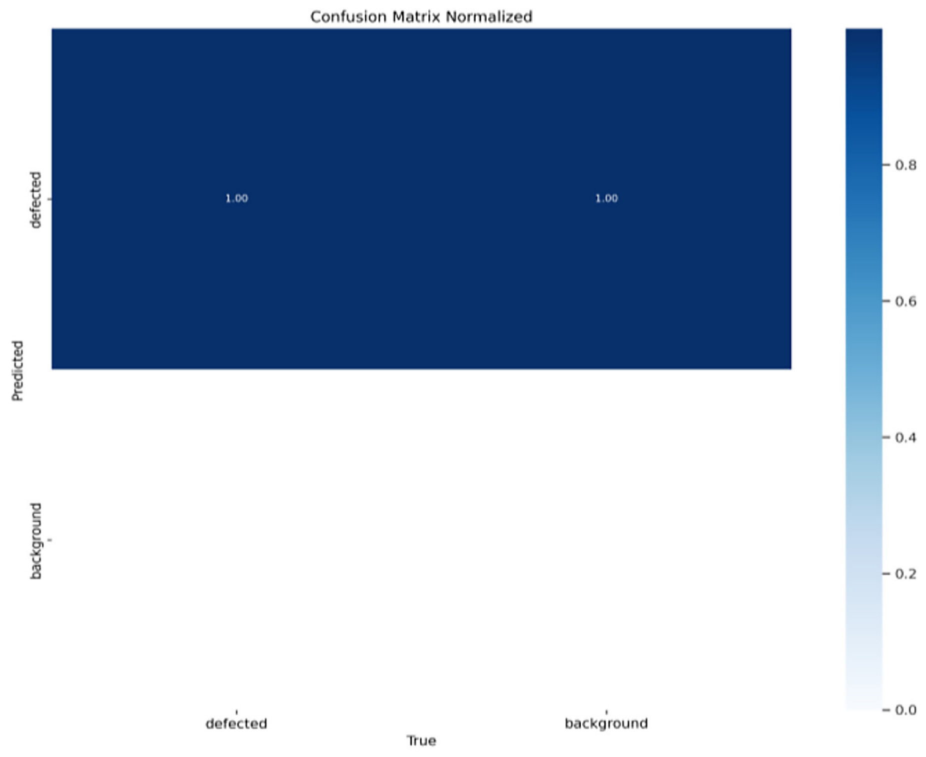

**Figure 14.** Confusion Matrix of YOLOv11n

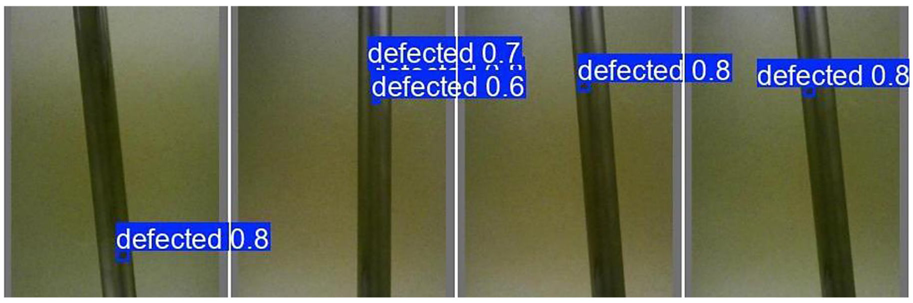

**Figure 15.** Inference images of YOLOv11n on a custom dataset

mAP@0.5:0.95. The values of the metrics are on the y-axis and the training epochs are on the x-axis, the graphs display rapid and stable conversion of prevision, recall and mAP@0.5. All the metrics consistently remain over 0.99 indicating strong and reliable detection capabilities. The mAP@0.5:0.95 improves steadily, demonstrating robust performance across various IoU thresholds.

Figure 17 shows the confusion matrix of the YOLOv12n model trained on the custom dataset, the x-axis represents the true-labels and the y-axis represents the predicted labels generated by the model. The diagonal elements (top left to bottom right) are the correctly classified input images and off-diagonal elements represent the misclassified input images. Both the classes in the matrix indicate an mAP@0.5 of 100. The darker shade of the deep blue color denotes higher correctness in the model's detections.

Figure 18 shows the inference results of the YOLOv12n model on the custom dataset. Where 4 different input images have been processed, in each image all the detected defects are highlighted with bounding boxes and labels, along with their respective confidence scores.

## Yolov13 results

Figure 19 shows the training evolution of the performance metrics of the YOLOv13n model with separate graphs for the evolution of the following metrics: precision, recall, mAP@0.5, and mAP@0.5:0.95. The values of the metrics are on the y-axis and the training epochs are on

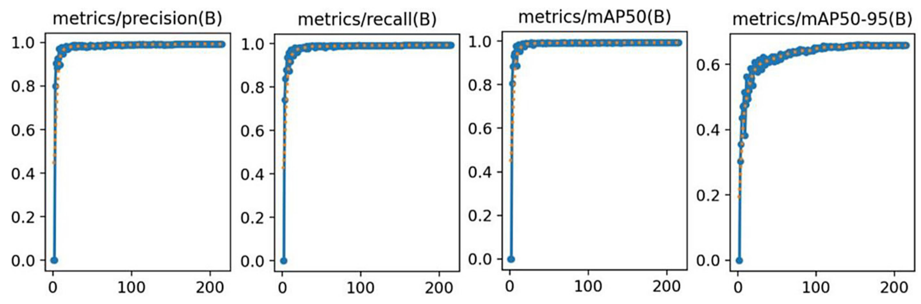

**Figure 16.** Training evolution of YOLOv12n on a custom dataset

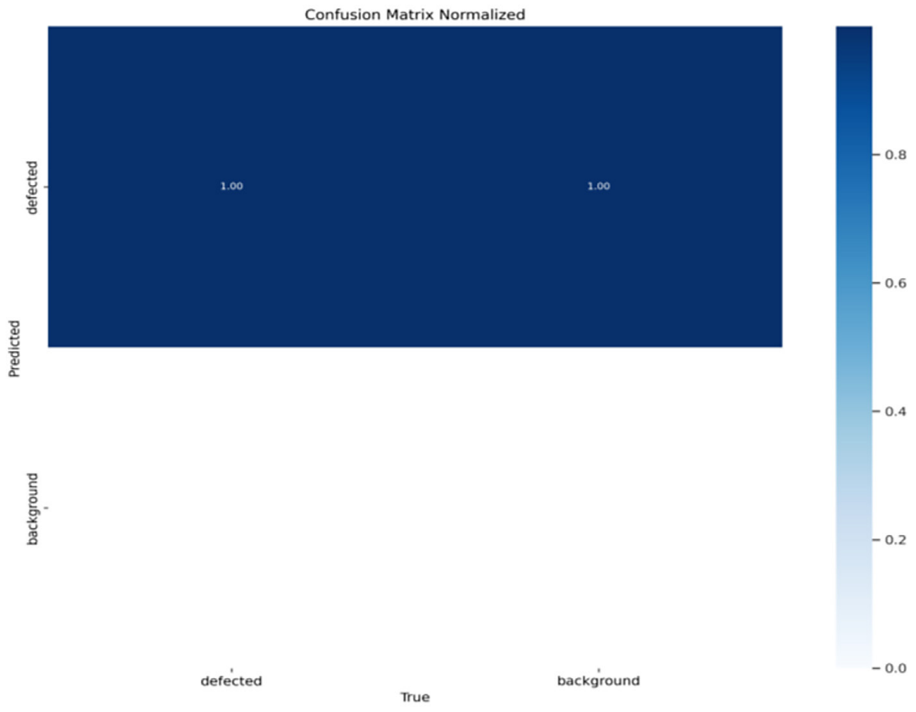

**Figure 17.** Confusion Matrix of YOLOv12n

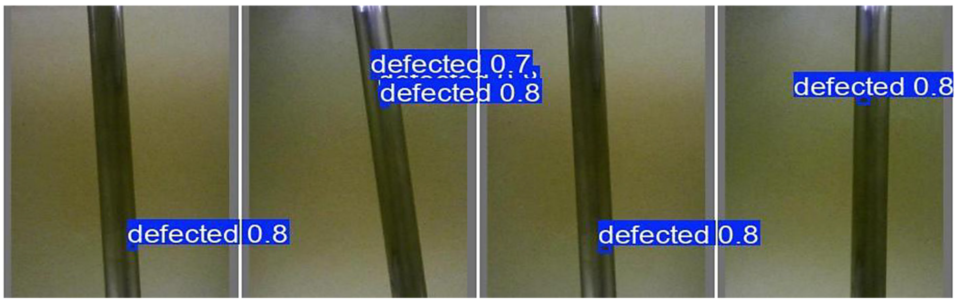

**Figure 18.** Inference images of YOLOv12n on a custom dataset

the x-axis, illustrating rapid and stable conversion of mAP@0.5, precision and recall in the early epochs. Precision and recall reach above 0.98 indicating strong defect detection capability, mAP@0.5 quickly approaches 1.0 reflecting high accuracy. While mAP@0.5:0.95 does not stabilize as rapidly but does improve steadily demonstrating effective detection even at stricter IoU thresholds.

Figure 20 shows the confusion matrix of the YOLOv13n model trained on the custom dataset, the x-axis represents the true-labels and the y-axis represents the predicted labels generated by the

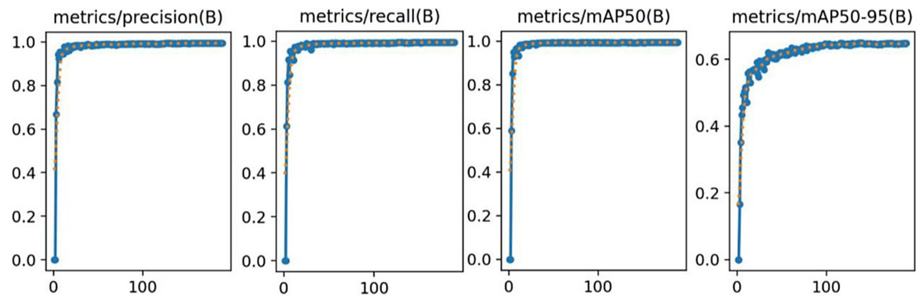

**Figure 19.** Training evolution of YOLOv13n on a custom dataset

model. The diagonal elements (top left to bottom right) are the correctly classified input images and off-diagonal elements represent the misclassified input images. Both the classes in the matrix indicate an mAP@0.5 of 100. The darker shade of the deep blue color denotes higher correctness in the model's detections.

Figure 21 shows the inference results of the YOLOv13n model on the custom dataset. Where 4 different input images have been processed, in each image all the detected defects are highlighted with bounding boxes and labels, along with their respective confidence scores.

## Effects of data augmentation on model performance

In this study, basic augmentation techniques such as flipping and rotation were employed to expand the number of training samples and to enhance model's generalizability. To further investigate this aspect, an ablation experiment was conducted on the YOLOv13n model using the same dataset without augmentation. The results revealed a noticeable decline in detection performance, confirming that the applied augmentation techniques positively impacted model learning and enhanced the detection of small and hard-to-capture defects. Specifically, precision improved by 1.33%, recall by 4.13%, mAP@50 by 1.52%, and mAP@50–95 by 0.63% as shown in Table 2 compared to the non-augmented dataset, demonstrating that even simple augmentations such as flipping and rotation contributed meaningfully to the overall detection performance. Figure 22 shows the comparison of the mAP@50 of the YOLOv13n trained on the original dataset

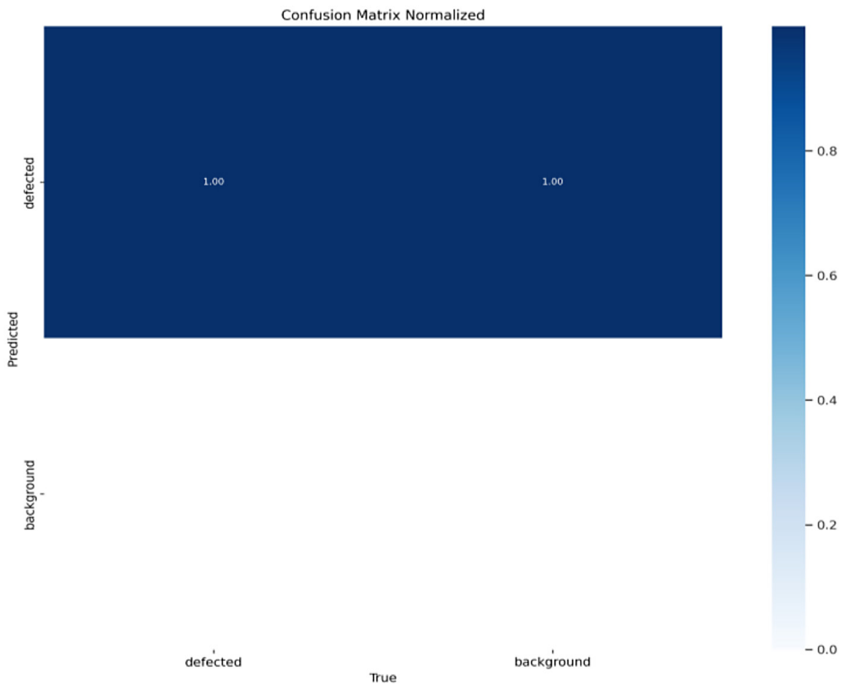

**Figure 20.** Confusion Matrix of YOLOv13n

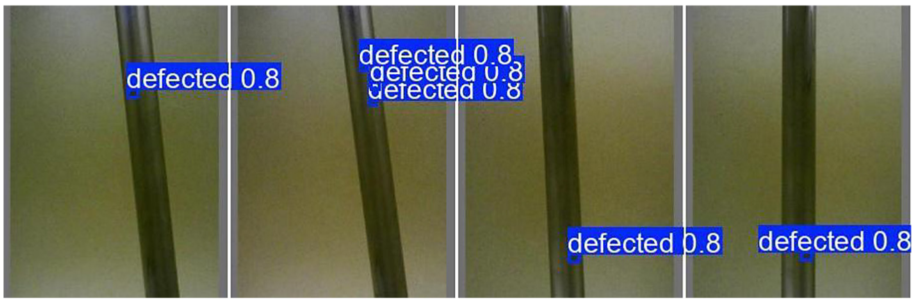

**Figure 21.** Inference images of YOLOv13n on a custom dataset

plotted with a blue solid line and the augmented dataset plotted with a red dashed line.

## Comparative analysis of results

The performance metrics of all the evaluated models are summarized in Table 3, which includes the number of epochs, training time, parameter count, and final model size. The results reveal a distinct trade-off between accuracy, inference speed, and computational cost. While YOLOv8n achieves the highest mAP of 0.9978 and leading precision–recall scores, this comes at the expense of greater model complexity and longer inference time. YOLOv11n and YOLOv13n perform comparably in accuracy, particularly under stricter IoU thresholds, yet deliver significantly improved inference speeds. YOLOv5n remains competitive despite its smaller size, demonstrating suitability for highly resource-constrained scenarios, whereas YOLOv12n, although slightly lower in mAP, still maintains balanced precision and recall values. Interestingly, the results indicate that smaller model size does not necessarily translate to faster inference. Although YOLOv5n has the smallest footprint i.e. 3.61 MB, it does not achieve the highest frame rate. Instead, YOLOv13n with a moderately larger size delivers the fastest inference of 303.03 FPS, surpassing even the much larger YOLOv8n (11.7 MB, 175.44 FPS). This observation confirms that inference performance is governed more by architectural efficiency than by model size alone.

The superior runtime efficiency of YOLOv13n arises from its refined architecture, incorporating HyperACE and FullPAD modules that minimize redundant computation and enable adaptive multi-scale feature fusion. Combined with depthwise separable convolutions and a lightweight backbone, these design enhancements substantially reduce FLOPs while preserving detection accuracy. As reported in [13], these architectural refinements allow YOLOv13n to achieve higher throughput without sacrificing precision. From a cost–benefit perspective, YOLOv13n offers the most favorable balance by providing near-top accuracy, the highest inference speed, and efficient resource usage. Its performance profile makes it highly suitable for real-time, edge-level deployment in industrial steel inspection systems, where high frame rates and timely detection directly influence production safety and quality assurance.

From the empirical analysis, it can be concluded that while inference speed may appear to be primarily dependent on hardware performance, in practice it is a complex relationship of model architecture, hardware utilization, and software-level optimizations. The architectural efficiency of a model determines how effectively its layers and parameters can be executed in parallel. The models with depthwise separable convolutions or reduced feature maps typically achieve lower latency because they minimize redundant computations and memory accesses. The YOLO family demonstrates this through progressive architectural refinements such as improved feature fusion and optimized convolutional modules that directly impact throughput without significantly increasing model complexity. Hardware utilization is another critical factor influencing inference speed. Deployment platforms with higher computational resources (e.g., GPUs, FPGAs, or AI accelerators) can process multiple operations concurrently, provided the model structure allows for efficient parallel execution. Memory bandwidth, cache utilization, and the precision format (e.g., FP32 vs. INT8) also play important roles in determining real-time performance. As reported in [27], deploying a smaller YOLOv3 variant on an edge device increased FPS by approximately 7.6×, affirming the significance of optimizing model and hardware compatibility

**Table 2.** Performance metrics of YOLOv13n trained on the original and augmented datasets

| Metric | Original dataset | Augmented dataset | Change (%) |
|---|---|---|---|
| mAP@50 | 0.9825 | 0.9957 | 1.3200 |
| Precision | 0.9545 | 0.9957 | 4.1290 |
| Recall | 0.9790 | 0.9942 | 1.5190 |
| mAP@50-95 | 0.6436 | 0.6400 | 0.6330 |

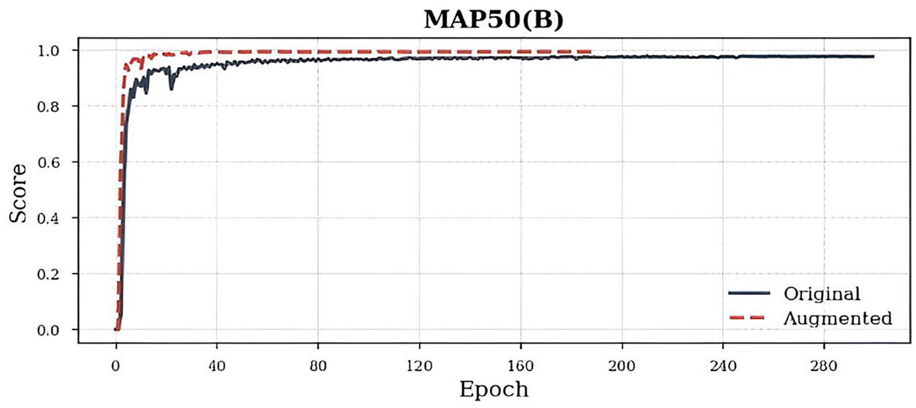

**Figure 22.** Comparison of mAP@50 of YOLOv13n trained on the original non-augmented and augmented datasets

rather than relying solely on raw hardware capability. At the software level, optimizations such as pruning, quantization, kernel fusion, and parameter sharing further enhance inference efficiency by reducing model size, memory footprint, and computational overhead while maintaining accuracy. Therefore, inference speed is not merely a function of hardware performance; it is the cumulative outcome of architectural design choices, hardware-aware deployment strategies, and targeted software optimizations. Collectively, these factors define the real-world applicability of deep learning models in industrial inspection systems, especially when deployed on embedded or edge computing platforms, as also emphasized in [28].

## CONCLUSIONS

Our proposed work developed a complete machine vision pipeline using five YOLO nano variants (YOLOv5n to YOLOv13n), evaluated on benchmark and industrial steel surface defect datasets. The methodology followed a structured process from problem analysis to model evaluation. YOLOv5n offered the best resource

**Table 3.** Validation performance metrics table

| Metrics | Models | | | | |
|---|---|---|---|---|---|
| | YOLOv5 nano | YOLOv8 nano | YOLOv11 nano | YOLOv12 nano | YOLOv13 nano |
| Epochs | 300 | 265 | 195 | 215 | 189 |
| Training Time (hh:mm:ss) | 03:00:28 | 03:45:00 | 02:27:46 | 03:00:26 | 03:16:41 |
| Parameters (M) | 1.9 | 3.2 | 2.6 | 2.6 | 2.5 |
| Size (Mb) | 3.61 | 11.7 | 5.2 | 5.17 | 5.14 |
| mAP@0.5 | 0.993 | 0.9977 | 0.992 | 0.9851 | 0.993 |
| mAP@05:0.95 | 0.6573 | 0.6695 | 0.6634 | 0.659 | 0.648 |
| Precision | 0.994 | 0.9978 | 0.992 | 0.9895 | 0.994 |
| Recall | 0.995 | 0.9961 | 0.994 | 0.9912 | 0.993 |
| FPS | 126.58 | 175.44 | 131.56 | 69.93 | 303.03 |

efficiency with solid real-time accuracy, while YOLOv8n achieved the highest precision, recall, and mAP@0.5 at the cost of model size and training time. YOLOv11n trained the fastest while maintaining strong accuracy, YOLOv12n provided reliable results but with slower inference, and YOLOv13n delivered high precision with the fastest inference speed of 303.03 FPS, making it particularly suitable for real-time industrial applications. Overall, the findings demonstrate that each variant presents distinct trade-offs between accuracy, speed, and resource efficiency, enabling informed model selection based on deployment needs.

This study opens pathways for broader research directions. However, this work is limited to YOLO nano variants and steel surface datasets, which may constrain broader applicability:

- Future work may extend this study to small or larger YOLO variants while also examining transformer-based lightweight architectures to further improve accuracy and efficiency.
- Model compression techniques (e.g., quantization, pruning) could be applied to enhance deployment on edge devices.
- Multi-modal inspection approaches that combine visual and infrared imaging could be explored to improve robustness across diverse industrial scenarios.

## Acknowledgements

This work acknowledges Agriauto Industries Limited for providing access to their facility for data collection and labeling.

This work was supported in part by a research grant from the Sindh Higher Education Commission (HEC) and in part by NED University Research Grant.

## REFERENCES

1. Cheng Z., Gao L., Wang Y., Deng Z., Tao Y. EC-YOLO: Effectual detection model for steel strip surface defects based on YOLO-V5. IEEE Access. 2024;12:62765–78. https://doi.org/10.1109/ACCESS.2024.3391353
2. Wang Z., Zhang S., Xie L. RDB-YOLO: A Lightweight and High-Precision Steel Surface Defects Detection Method. In: 2024 China Automation Congress (CAC). 2024;1237–42. https://doi.org/10.1109/CAC63892.2024.10865047
3. Yang K., Chen T. Lightweight Surface Defect Detection Algorithm Based on Improved YOLOv5. In: 2024 5th International Conference on Mechatronics Technology and Intelligent Manufacturing (ICMTIM). 2024;798–802. https://doi.org/10.1109/ICMTIM62047.2024.10629491
4. Rappe N.A., Kirda A.W., Yassin H., Bartoszuk M., Caesarendra W. Application of YOLOv8 in fusion welding defect detection on carbon steel for potential remote visual inspection. Adv Sci Technol Res J. 2025;19(12).
5. Yu J., Shi X., Wang W., Zheng Y. LCG-YOLO: A real-time surface defect detection method for metal components. IEEE Access. 2024;12:41436–51. https://doi.org/10.1109/ACCESS.2024.3378999
6. Ma S., Zhao X., Wan L., Zhang Y., Gao H. A lightweight algorithm for steel surface defect detection using improved YOLOv8. 2025. https://doi.org/10.21203/rs.3.rs-5933201/v1
7. Liu Y., Liu Y., Guo X., Ling X., Geng Q. Metal surface defect detection using SLF-YOLO enhanced YOLOv8 model. Scientific Reports. 2025 Apr 1;15. https://doi.org/10.1038/s41598-025-94936-9
8. Huang B., Wang M. Steel surface defect detection algorithm based on DM-YOLOv11. 2025;848. https://doi.org/10.1109/AIITA65135.2025.11048059
9. Sun Y., Yan H., Shang Z., Yang M. MCH-YOLOv12: Research on surface defect detection algorithm for aluminum profiles based on improved YOLOv12. Sensors. 2025;25(17). https://doi.org/10.3390/s25175389
10. Zhao Y., Lv W., Xu S., Wei J., Wang G., Dang Q., et al. DETRs Beat YOLOs on Real-time Object Detection [Internet]. 2024. https://arxiv.org/abs/2304.08069
11. Xu S., Wang X., Lv W., Chang Q., Cui C., Deng K., et al. PP-YOLOE: An evolved version of YOLO [Internet]. 2022. https://arxiv.org/abs/2203.16250
12. Huang X., Wang X., Lv W., Bai X., Long X., Deng K., et al. PP-YOLOv2: A Practical Object Detector [Internet]. 2021. https://arxiv.org/abs/2104.10419
13. Lei M., Li S., Wu Y,. Hu H., Zhou Y., Zheng X., et al. YOLOv13: Real-Time Object Detection with Hypergraph-Enhanced Adaptive Visual Perception. arXiv [csCV] [Internet]. 2025. http://arxiv.org/abs/2506.17733
14. Feng Z., Liu F. Balancing feature symmetry: IFEM--YOLOv13 for robust underwater object detection under degradation. Symmetry. 2025 Sep 13;17:1531. https://doi.org/10.3390/sym17091531
15. Wang Z., Hu S., Wang X., Gao Y., Zhang W., Chen Y., et al. YOLOv13-Cone-Lite: An enhanced algorithm for traffic cone detection in autonomous formula racing cars. Applied Sciences. 2025;15(17). https://doi.org/10.3390/app15179501

16. Wu Q.K. NEU-DET [Internet]. IEEE Dataport; 2024. Available from: 10.21227/j84r-f770
17. Neudet. NEU-DET Dataset. Roboflow Universe [Internet]. 2024 Jul. https://universe.roboflow.com/neudet-98ubn/neu-det-usvkq
18. Lv X., Duan F., Jiang J. jia, Fu X., Gan L. Deep metallic surface defect detection: The new benchmark and detection network. Sensors. 2020;20(6). https://doi.org/10.3390/s20061562
19. MachineLearning. GC10-DET Dataset Dataset. Roboflow Universe [Internet]. 2024 Apr. https://universe.roboflow.com/machinelearning-p49ei/gc10-det-dataset
20. Nanthini K., Sivabalaselvamani D., Chitra K., Gokul P., KavinKumar S., S. Kishore. A Survey on Data Augmentation Techniques. In: 2023 7th International Conference on Computing Methodologies and Communication (ICCMC). 2023. p. 913–20. https://doi.org/10.1109/ICCMC56507.2023.10084010
21. Rey L., Bernardos A.M., Dobrzycki A.D., Carramiñana D., Bergesio L., Besada J.A., et al. A Performance analysis of you only look once models for deployment on constrained computational edge devices in drone applications. Electronics. 2025;14(3). https://doi.org/10.3390/electronics14030638
22. Alashrafi L., Badawood R., Almagrabi H., Alrige M., Alharbi F., Almatrafi O. Benchmarking lightweight YOLO object detectors for real-time hygiene compliance monitoring. Sensors. 2025;25(19). https://doi.org/10.3390/s25196140
23. Mutabarura P, Muchuka N, Segera D. Comparative evaluation of YOLO models on an African road obstacles dataset for real-time obstacle detection. Eng Technol Appl Sci Res. 2025 Feb 2;15(1):19045–51. https://doi.org/10.48084/etasr.9135
24. Doherty J., Gardiner B., Kerr E., Siddique N. BiFPN-YOLO: One-stage object detection integrating Bi-Directional Feature Pyramid Networks. Pattern Recognition. 2025 Apr 1;160:111209. https://doi.org/10.1016/j.patcog.2024.111209
25. Ali M.L., Zhang Z. The YOLO framework: A comprehensive review of evolution, applications, and benchmarks in object detection. Computers. 2024;13(12). https://doi.org/10.3390/computers13120336
26. Kühlechner R. Object detection survey for industrial applications with focus on quality control. Production Engineering [Internet]. 2025 Aug 29. https://doi.org/10.1007/s11740-025-01369-4
27. Feng H., Mu G., Zhong S., Zhang P., Yuan T. Benchmark analysis of YOLO performance on edge intelligence devices. Cryptography. 2022;6(2). https://doi.org/10.3390/cryptography6020016
28. Wang X., Jia W. Optimizing edge AI: A Comprehensive Survey on Data, Model, and System Strategies [Internet]. 2025. https://arxiv.org/abs/2501.03265