Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.077838

REVIEW

Applying Deep Learning to Defect Detection in Steel Manufacturing

Duane G. Noé[1], Ku-Chin Lin[2], Chang-Lin Chuang[3] and Yung-Tsung Cheng[3,*]

[1]Graduate School of Electronic Engineering, Kun Shan University, Tainan, Taiwan
[2]Department of Mechanical Engineering, Kun Shan University, Tainan, Taiwan
[3]Department of Electric Vehicles and Intelligent Electronics Engineering, Kun Shan University, Tainan, Taiwan

*Corresponding Author: Yung-Tsung Cheng. Email: 520.yong.cong@gmail.com

Received: 17 December 2025; Accepted: 30 March 2026; Published: 08 May 2026

ABSTRACT: Steel manufacturing requires high-throughput and high-reliability surface inspection to minimize safety risks, scrap rates, and downstream quality reductions. Conventional rule-based vision and manual inspection are often impeded in real production environments by variable illumination, complex textures, subtle defect morphology, and stringent latency constraints imposed by production-line operation. Deep learning (DL) has become a dominant paradigm for the detection and classification of defects when inspecting steel, but many previous studies have performed broad architectural overviews without explicitly connecting model and pipeline choices to deployment-critical factors such as processing speed, hardware availability, annotation cost, and robustness during domain shift. This review synthesizes ten representative case studies on DL-based defect inspection in steel manufacturing and closely related industrial settings, spanning classification, object detection, and segmentation workflows. To enable structured comparisons, we harmonized practical considerations across the ten studies, including task formulation, backbone design, training strategy (e.g., transfer learning), data augmentation, reported throughput, and commonly used performance indicators such as accuracy, precision/recall, mean average precision, and processing speed. Due to the heterogeneity in datasets, metrics, and hardware configurations across studies, we further introduce a transparent, review-oriented figure of merit as a heuristic summary of reported benefit–cost parameters (performance accuracy and processing speed vs. training burden and model complexity), while explicitly addressing the limitations associated with missing values and avoiding claims of statistically definitive ranking. Based on recurring patterns across the selected studies, we propose a conceptual hybrid framework blueprint—derived from the reviewed literature rather than newly experimentally validated results—that integrates transfer learning, real-time detection, and end-to-end learning principles as an engineering template for practical deployment. We conclude by providing actionable guidance for applications and outline future directions in label-efficient learning, cross-domain robustness, and standardized benchmarking and reporting to improve the reproducibility and industrial relevance of our approach.

KEYWORDS: Deep learning (DL); neural networks (NNs); dataset; object detection; architecture

1 Introduction

Advances during 2023–2025 in inspection methods for detecting industrial defects have focused on data-efficient learning and deployment feasibility. Beyond conventional supervised pipelines, self-supervised and weakly/semisupervised strategies have been actively explored with the aim of mitigating annotation scarcity and improving transferability during domain shift [1–4], while edge-oriented optimization and lightweight design have become central for addressing real-time manufacturing constraints. This review

integrates recent progress on label-efficient learning, cross-domain robustness, and industrial deployment considerations to provide an updated, practice-oriented synthesis for steel-defect inspection [5–7].

Steel is a fundamental material used in numerous industries, and its integrity directly affects the safety and value of downstream products. Defects such as cracks, dents, scratches, and inclusions can severely undermine the durability of steel, leading to potential failures, safety hazards, and increased production costs.

This review article synthesizes deep learning (DL)-based defect inspection for steel manufacturing while focusing on deployment-related trade-offs such as performance accuracy vs. throughput, annotation burden, and robustness during domain shift. Domain generalizability and cross-domain adaptation have become central concerns due to the potential for changes in texture, illumination, and surface condition to invalidate laboratory-tuned models on production lines [4]. Accordingly, we do not claim to have produced a new experimentally validated model; instead, our approach was designed to guide practical implementations by providing (i) a transparent, selection-criteria–driven synthesis of representative studies, (ii) a harmonized, deployment-related comparison template (including a conservative missing-data policy), and (iii) a conceptual hybrid framework blueprint derived from recurring patterns in the literature.

Traditional defect detection (DD) methods such as manual inspection and basic automation are often labor-intensive, insufficiently precise, and prone to human error [8–10]. Recent studies of steel-surface inspection have increasingly focused on the effects of deployment constraints—throughput, annotation cost, and robustness during domain shift—on accuracy, motivating the need for an updated synthesis grounded in practical manufacturing settings [5–7,11–13]. These limitations have led to increasing interest in applying DL techniques to DD. Convolutional neural networks (CNNs) have performed particularly well in DD due to their ability to transform raw image data into layered patterns that highlight surface anomalies. CNN models such as You Only Look Once (YOLO) [14,15], VGG16 [16], MobileNet [17], and Residual Network (ResNet) have been used to detect defects in steel. YOLO is known for its real-time performance in industrial settings [14,15], and MobileNet exhibits high efficiency and low computational cost [17]. ResNet introduced skip connections to address vanishing gradients, and it performs well in complex defect scenarios, although it can be computationally expensive [18].

Research Questions and Scope

To ensure that this review was not simply a descriptive aggregation of case studies, a question-driven synthesis guided by the following research questions (RQs) was performed to address key knowledge gaps in DL-based steel-defect inspection:

RQ1: What tasks (classification, detection, and segmentation), defect taxonomies, datasets, and benchmarking practices are most commonly adopted in steel-defect inspection, and what reporting heterogeneity restricts cross-study comparability?

RQ2: What model families and pipeline components appear across representative studies, and what design rationales are reported to support steel-defect inspection under industrial conditions, including trade-offs in accuracy and deployment metrics such as throughput and latency, model size and complexity, and training and annotation burden?

RQ3: What strategies are reported to mitigate annotation scarcity and domain shift (e.g., transfer learning, weakly/semisupervised and self-supervised learning, and domain adaptation and generalization), and what evidence is available regarding robustness in deployment contexts related to steel?

RQ4: Based on cross-study patterns, what deployment-related blueprint and reporting recommendations can be identified to guide practical system design and future benchmarking for steel-defect inspection?

These RQs drove the review process in this study, with RQ1 and RQ2 informing the literature synthesis and case-study harmonization, RQ3 guiding the analysis of robustness strategies, and RQ4 underpinning the conceptual framework blueprint and actionable guidance.

This review focused on steel manufacturing and closely related industrial metal-surface-inspection problems with clear transferability to steel. We synthesized reported evidence from representative peer-reviewed studies and interpreted numerical results within the reported evaluation setting in each study (datasets, metrics, and hardware), rather than as statistically pooled comparisons.

2 Contextualization

Neural networks (NNs) and DL have greatly improved the efficacy and precision of DD. NNs are foundational to DL, and they have evolved since their introduction during the late 1980s to become the backbone of modern DD methodologies. DL models [19–23] are characterized by multilayer NN architectures and have become powerful tools for DD thanks to the proliferation of annotated training data, advances in computing infrastructures, and refinements in network architectures and training strategies. Seminal contributions in deep representation learning [24–26] have further accelerated the adoption of deep NNs in computer vision and related inspection tasks.

This study performed a comprehensive literature review of the application of DL models in DD that included a detailed examination of the problems encountered during defect inspections, and categorizing them based on various characteristics such as defect type, methodology, domain of application, imaging modality, classification, and data availability. Numerous quantitative analyses have been conducted into the use of DL in DD, but qualitative findings remain scarce. This section presents the background context for the question-driven synthesis defined in Research Questions and Scope Section. We briefly summarize (i) representative DL model families used in industrial inspection applications, (ii) commonly used datasets and benchmarking practices, and (iii) typical defect taxonomies and deployment constraints that influence practical design choices in steel inspection. Building on this context, this review synthesizes reported evidence obtained from ten representative case studies selected by applying the inclusion and exclusion criteria described in Section 5.1 and a harmonized extraction template (Section 5.2), and uses recurring cross-study patterns to develop a conceptual blueprint and reporting recommendations (Section 7), without claiming that a novel experimentally validated inspection system has been developed [27].

2.1 Problem Statement

DL models are distinguished by their multilayer NN structure, which is critical for DD because it allows DL models to surpass the accuracy and efficient-detection capabilities of the human eye [28]. Fig. 1 illustrates an artificial NN with multiple hidden layers that can support complex pattern recognition, thus enabling the detection of subtle features that conventional inspection methods are unable to detect. This study adopted a qualitative approach to examine how DL is applied to the detection of material and product defects.

DD problems were categorized on the basis of various characteristics [8,9,29–31] (Table 1).

We adopted the following classifications in this study:

- Defect type: surface (e.g., scratches), structural (e.g., fractures), or functional (e.g., safety impact).
- Inspection methodology: automated [computer vision or machine learning (ML)] or manual.
- Domain of application: manufacturing or infrastructure engineering (among others).
- Imaging modality: X-ray, ultrasound, or infrared (among others).
- Output type: quality or safety impact (severity), or category (classification).
- Data availability: supervised or unsupervised learning.

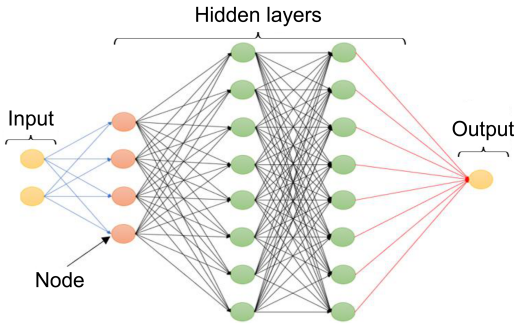

Figure 1: Schematic of an artificial NN used for DD.

Table 1: Some types of defects considered in this study.

Classification	Defect Type
Type	• Surface defects: scratches, dents, cracks, or stains • Subsurface defects: internal cracks or other structural weaknesses • Dimensional deviations: incorrect size or shape
Nature	• Localized defects: problems within specific small areas • Distributed defects: problems across relatively large areas
Domain of application	• Manufacturing: quality control and inspection of production lines for various products • Infrastructure: monitoring structural integrity in buildings or infrastructure

These classifications were used to provide the background context on how DL is applied to DD across materials and products, which formed the foundation for the question-driven synthesis later in this review.

2.2 Justification

The objective of this review was to provide deployment-related qualitative insights into how DL has been applied to DD in industrial contexts, specifically when inspecting steel [6]. Given the heterogeneity of datasets, defect taxonomies, metrics, and hardware in the literature, we synthesized reported evidence from ten representative peer-reviewed case studies selected by applying predefined criteria (Section 5.1). Rather than statistically pooling results across studies, we harmonized reported attributes using a consistent extraction methodology (Section 5.2) to support transparent cross-study comparisons and to identify recurring engineering trade-offs and evidence gaps.

2.3 Study Objective

The objective of this review was to synthesize recent DL and NN approaches for steel DD with an explicit emphasis on deployment-related constraints such as latency and throughput, hardware considerations, annotation burden, and robustness during domain shift. Specifically, we (i) identified representative model families and pipeline components found in studies of steel-defect inspection, (ii) summarized their structural and functional characteristics together with the design rationales stated in the literature, (iii) harmonized the extracted evidence using a deployment-oriented comparison template with a conservative missing-data policy, and (iv) used recurring cross-study patterns to develop a conceptual hybrid framework blueprint and reporting recommendations to support practical implementation and future benchmarking.

Guided by RQ1–RQ4 (Research Questions and Scope Section), this review article focuses on advancements that occurred during 2023–2025, while referencing earlier foundational studies when necessary. RQ1 and RQ2 directed the selection and harmonization of representative studies and the structured comparisons of reported performance and deployment metrics within each study's stated evaluation setting, RQ3 guided the synthesis of label-efficient learning and domain-shift mitigation strategies, and RQ4 underscored the blueprint and actionable guidance for practical deployment and standardized reporting. The next sections review representative DL-based studies of defect inspection and organize the synthesis based on foundational concepts, datasets, and defect taxonomies.

3 Literature Review

This section synthesizes the selected literature from three aspects that directly influence deployment outcomes in steel inspection: (i) core NN and DL concepts and architectural evolution, (ii) datasets and benchmarking practices, and (iii) defect taxonomies commonly reported in industrial settings. These components collectively motivated the model choices and their subsequent evaluations in this study. These foundational concepts, datasets, and defect taxonomies (addressing RQ1) form the basis for the methodological extraction in Section 5 and the case analyses in Section 6, where we harmonize heterogeneous evidence to reveal deployment trade-offs.

3.1 NNs and DL

NNs have played a pivotal role in DD, particularly in recognition tasks (Fig. 2). The use of DL has increased markedly since 2006 due to advances in speech recognition, the increased availability of annotated training datasets, improvements in parallel computing infrastructures, and refinements in training methodologies and network architectures. Contributions from Hinton's research group [24,25] further increased the adoption of DL, especially in labeled image analysis, in which major advances have been achieved through optimized techniques. CNNs [21,24,25,32–38] utilize feature mapping, filtering, and pooling operations to convert input images into hierarchical feature representations. This conversion process enables the automated recognition of patterns in high-dimensional data and facilitates joint optimization across multiple tasks. CNNs are widely employed across various computer vision applications, including image classification, image retrieval, facial recognition, and video analysis [32,34,36–38]. In object detection, CNNs support both region-based approaches and regression- or classification-based methods [32,37,39–41].

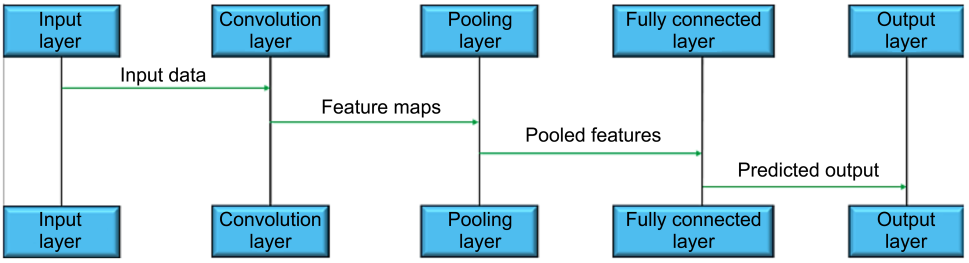

Figure 2: Example architecture of an artificial NN.

Region-based convolutional neural networks (R-CNNs) [41,42] employ a region-based detection pipeline (Fig. 3) that can achieve high localization accuracy, but their multistage design often incurs a substantial computational overhead and training complexity, which has motivated the development of more-streamlined alternatives [42].

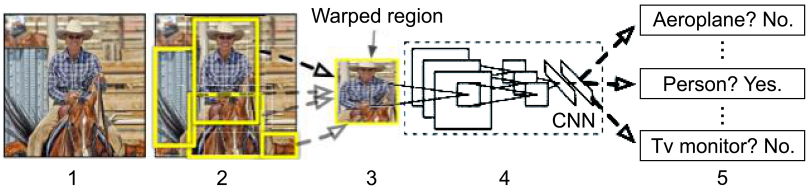

Figure 3: Workflow of a region-based convolutional neural network [41].

SPP-Net (Spatial Pyramid Pooling Network) [41,42] improves the proposal-based detection efficiency (Fig. 4) by pooling multiscale features on shared feature maps, which reduces the sensitivity to input size and warping during region processing.

YOLOv5 [43–45] is a single-stage detector that is widely adopted for industrial-defect inspection because it offers a desirable speed–accuracy trade-off and deployment flexibility (Fig. 5). However, its realized robustness and localization accuracy remain sensitive to training strategy and operating constraints.

YOLOv4 [46,47] enables real-time detection (Fig. 6) by improving feature extraction and multiscale fusion, but the effective gains in steel inspection depend on the available computational resources and the use of careful tuning according to the evaluation setting.

VGGNet [16] provides a deep yet conceptually simple CNN backbone (Fig. 7) that is effective for feature extraction and classification. However, its relatively high parameter count and computational costs make it less attractive for latency- or edge-constrained inspection tasks when compression is not used.

MobileNet variants [19,48] focus on resource-efficient processing (Fig. 8) to enable edge deployment in a lightweight convolutional design, typically trading off representational ability for throughput and potentially reducing robustness for subtle defects or during domain shift.

YOLOv8n (Fig. 9) represents a lightweight single-stage detector configuration that prioritizes parameter efficiency to achieve real-time defect localization when latency and hardware constraints dominate deployment decisions [49].

We also note that Ultralytics released the newer generation YOLOv11 model in September 2024 with the aim of improving accuracy–speed–efficiency trade-offs across diverse vision tasks. Recent studies of defect inspections have begun to evaluate YOLOv11 in edge-oriented settings, highlighting the continuing trend toward real-time feasibility with hardware-specific reporting [50].

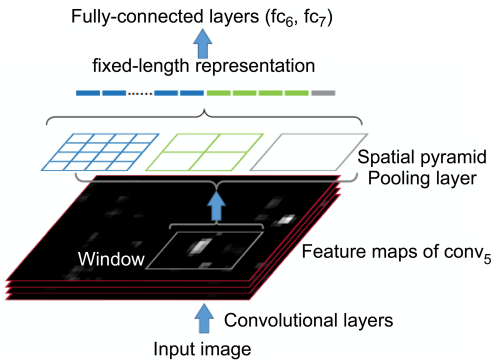

Figure 4: Architecture of the SPP-Net model [41].

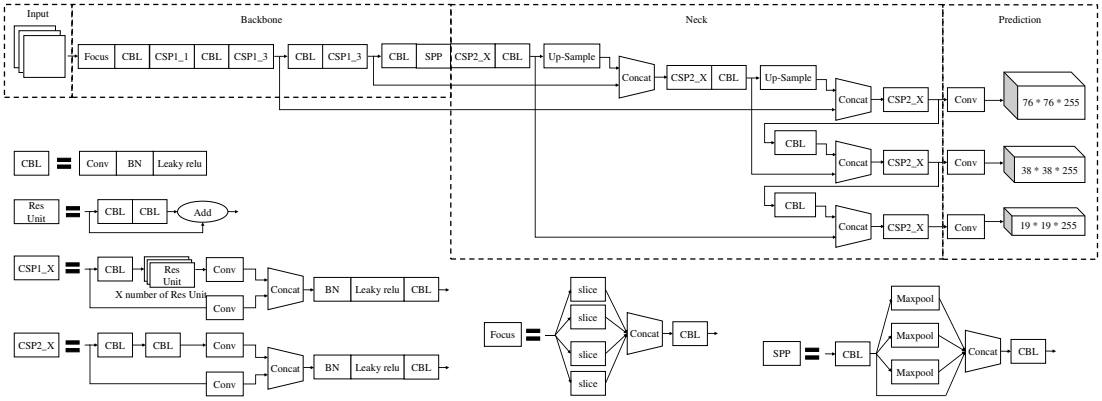

Figure 5: Architecture of the YOLOv5 model [44].

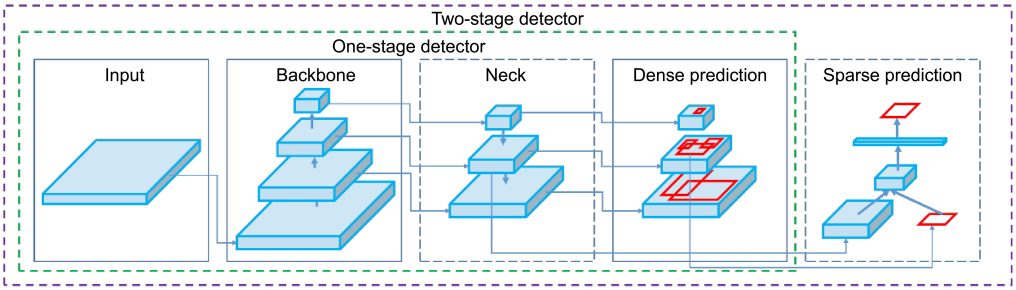

Figure 6: Architecture of the YOLOv4 model [43].

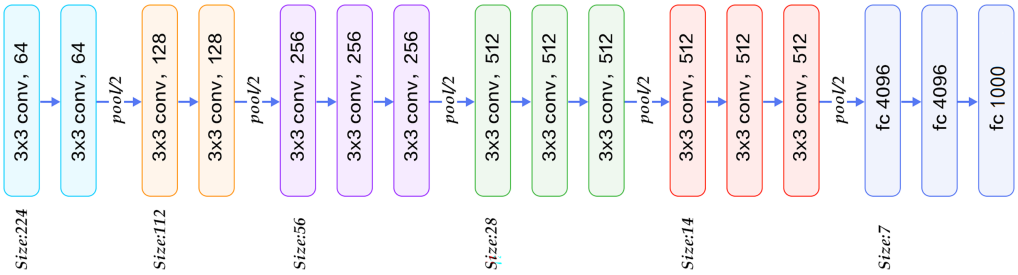

Figure 7: Architecture of the VGG16 model [16].

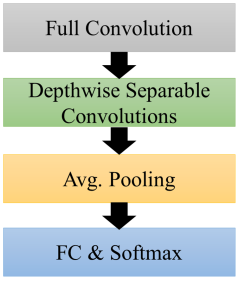

Figure 8: Architecture of the MobileNetV1 model [17].

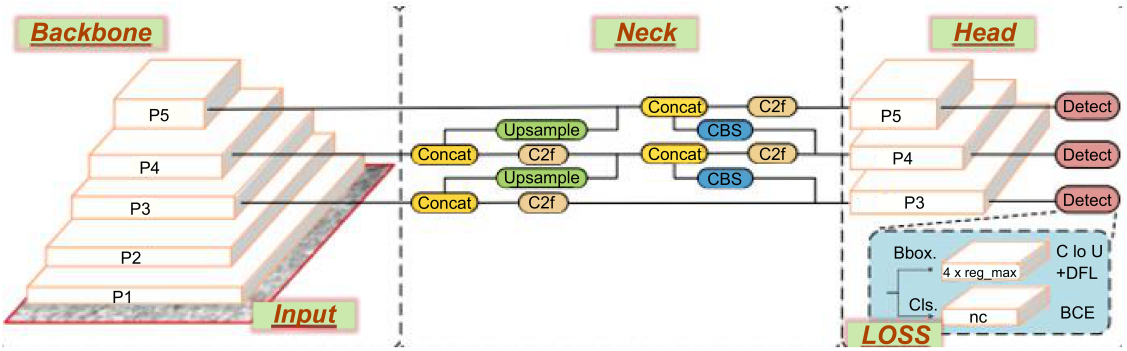

Figure 9: Architecture of the YOLOv8n model [49].

NNs (especially CNNs) have transformed DD by enabling faster and more-accurate analyses. Advances in datasets, computing, and architecture design have driven the success of models such as R-CNNs, YOLO, VGG, and MobileNet in DD. Despite the effectiveness of these models, recent reviews [51,52] argued that continued research is needed to realize their full potential in industrial and scientific applications.

3.2 Datasets

The reviewed studies commonly used publicly available benchmarking datasets such as Severstal Steel Defect Detection (SSDD), NEU/NEU-DET, DAGM, and Kolektor Surface-Defect Dataset (KolektorSDD) to evaluate methodologies for steel-defect inspection [53,54]. In parallel, newer datasets continue to expand defect taxonomies and support more realistic evaluations, including CSDD for casting-surface defect detection and segmentation [6], Deep Metallic Surface Defect Detection [55], and X-SDD [56], which together facilitate cross-dataset validation and stress testing under production-related variability.

The SSDD dataset [53] supports accurate surface-defect identification in steel manufacturing. It contains 18,074 images that include 11,408 unlabeled and 19,958 labeled objects across 4 defect classes. This dataset facilitates object localization and classification by providing pixel-level annotations that enable instance segmentation. The dataset is divided into 12,568 training images and 5506 validation images (Fig. 10).

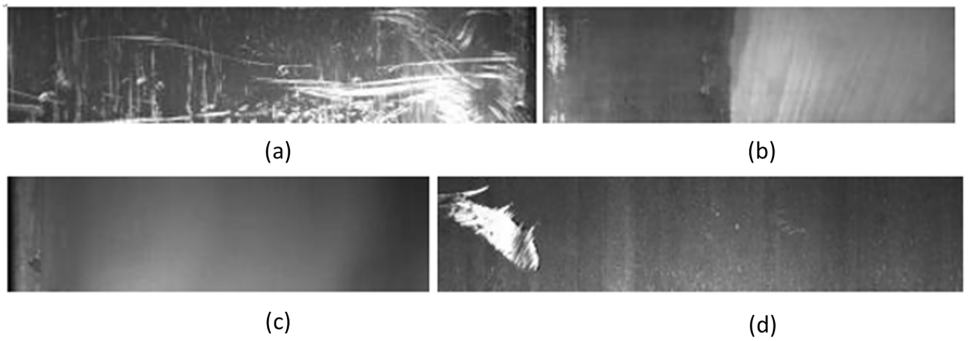

Figure 10: Sample images from the SSDD dataset containing defects of classes 1 (**a**), 2 (**b**), 3 (**c**) and 4 (**d**).

Northeastern University (Boston, Massachusetts, USA) has developed two vital resources for defect recognition: the NEU surface database and NEU-DET [57]. The NEU database serves as a crucial benchmarking tool in defect-recognition research, by comprising 1800 grayscale images showcasing 6 types of defects (Figs. 11 and 12), and also 1200 images that include detailed annotations of 4 types of defects.

DAGM is a benchmarking dataset for DD on textured surfaces. It consists of ten subsets, each containing grayscale PNG images labeled as defective or nondefective. Each subset consists of 1000 images for training and 2000 images for evaluation (Fig. 13), as shown in Fig. 13, each defect sample is presented as a paired visualization, with the original grayscale image in the upper subplot and the corresponding defect annotation in the lower subplot, enabling direct comparison between the raw surface pattern and the labeled defect region; however, the defects are poorly annotated. DAGM is designed to test real-time inspection challenges, since the dataset presents varied textures and defects to facilitate assessments of algorithm robustness.

The KolektorSDD [58] was established to address the shortage of publicly available datasets containing real images with annotated surface defects of motor/generator commutators. The surface of each commutator was captured in eight nonoverlapping images. The dataset contains 400 images of 50 defective commutators. The defect of each commutator is visible in only 1 of its 8 images, and so the dataset contains 50 images that depict defects (Fig. 14).

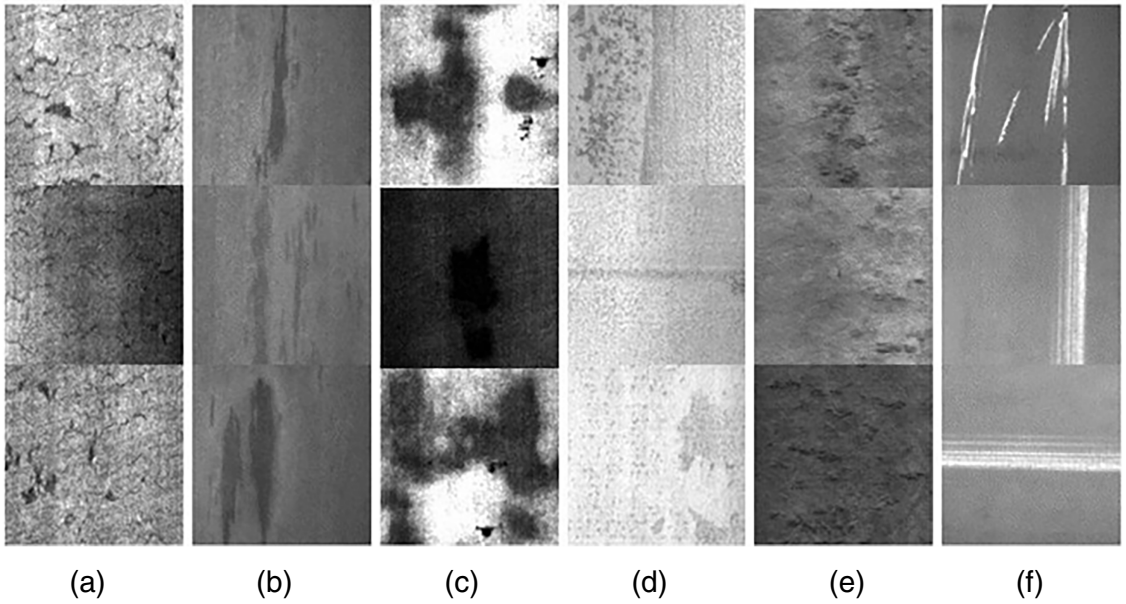

Figure 11: Representative grayscale images from the NEU dataset for six typical defect categories: (**a**) crazing, (**b**) inclusions, (**c**) patches, (**d**) pitting, (**e**) rolled-in scale, and (**f**) scratches.

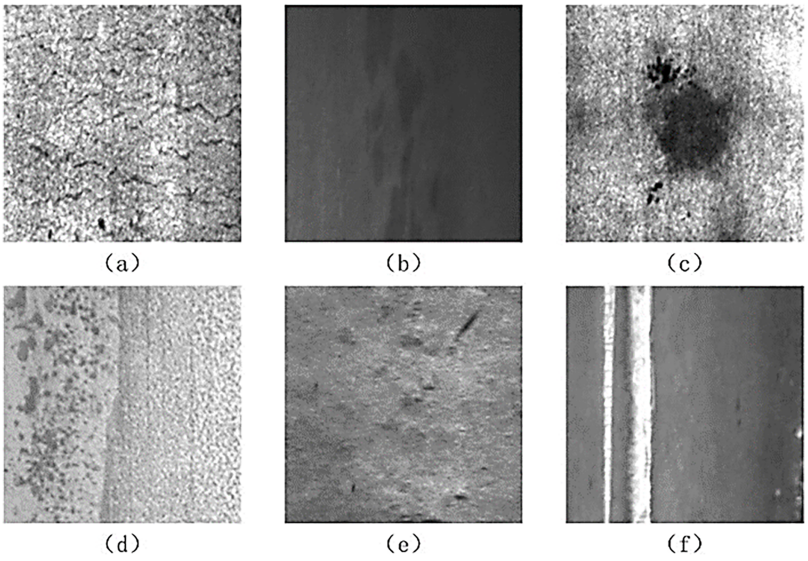

Figure 12: Examples of defect types in the NEU-DET dataset: (**a**) crazing, (**b**) inclusions, (**c**) patches; (**d**) pitting, (**e**) rolled-in scale, and (**f**) scratches.

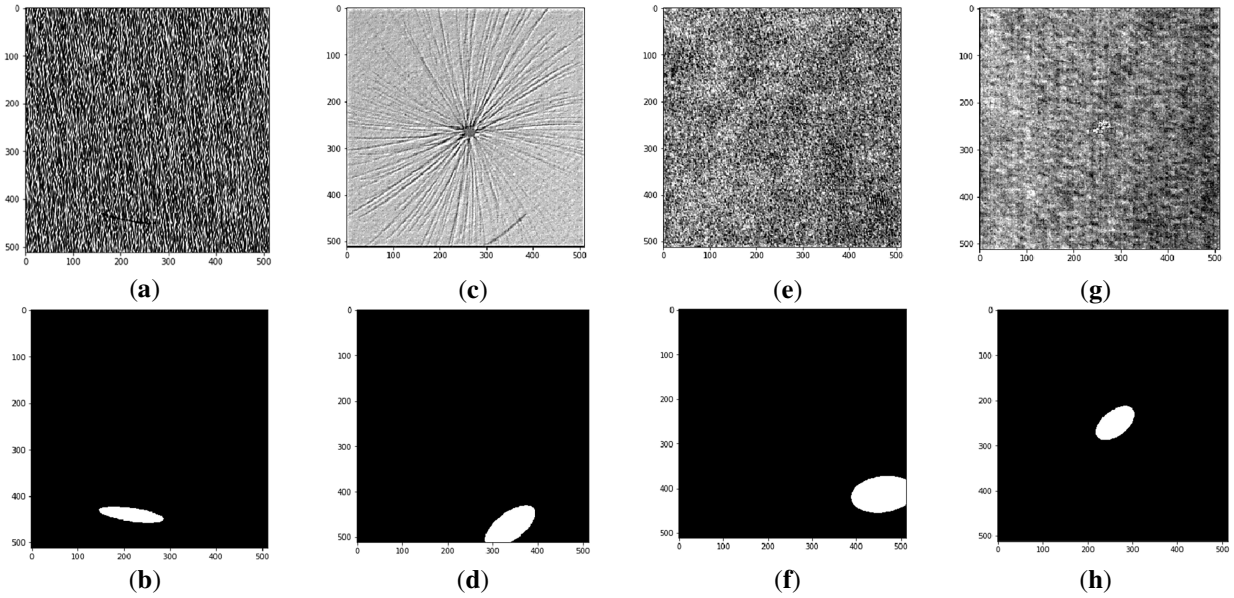

Figure 13: Representative grayscale defect samples from the DAGM benchmark dataset. (**a**,**c**,**e**,**g**) show the original defect images; (**b**,**d**,**f**,**h**) show the corresponding annotated images indicating the defect regions. Each upper–lower pair corresponds to the same sample, enabling direct comparison between the raw surface appearance and the labeled defect area.

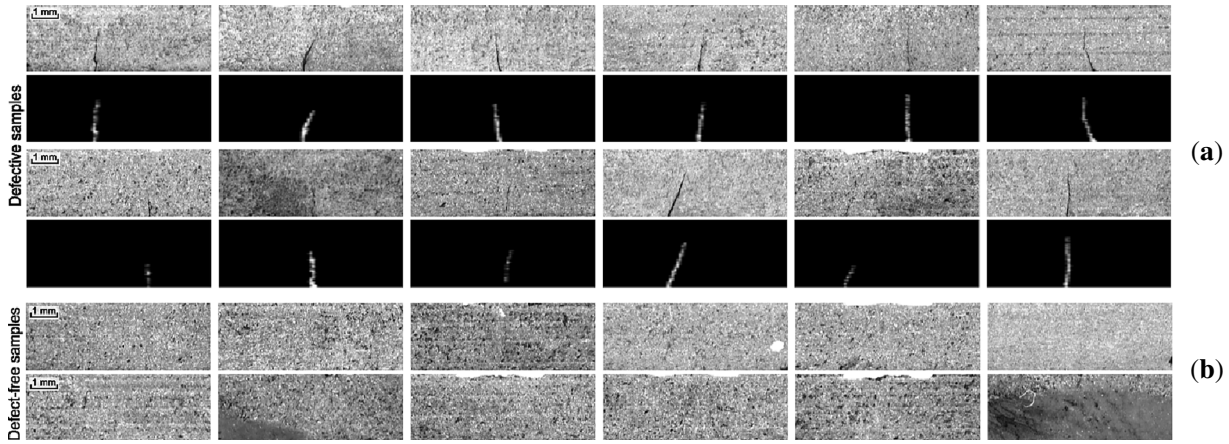

Figure 14: Surface images from the KolektorSDD: (**a**) defective samples and (**b**) defective-free sample.

The next section summarizes how defects are typically defined and categorized when inspecting steel, which directly influences labeling practices, task formulation, and metric interpretation.

3.3 Types of Defects

Accurate defect identification and categorization are crucial in industrial manufacturing for ensuring safety, quality, and efficiency. DD is crucial to ensuring that defective products are not sold and also to reduce waste from the erroneous disposal of usable material. Defect categorization allows manufacturers to address recurring problems and refine production systems. Case studies can revealed the role of ML in distinguishing

similar defects and evaluating defect severity. We used the aforementioned datasets to examine the following types of defects:

1. SSDD dataset: steel surface defects, such as scratches.
2. NEU-DET: steel surface defects, such as crazing and inclusions.
3. GIMBDIC: aluminum defects, such as mottling and orange peel.
4. Customized X-ray dataset: steel surface defects, specifically welding defects.

The diversity of defect morphologies and imaging conditions reported in these datasets mean that the selected inspection targets will influence the demands of representation learning. These defect categories and visual characteristics have resulted in different priorities being focused on when designing networks, such as localization sensitivity, multiscale feature extraction, and robustness to texture and illumination variations. In the next section we review the CNN model families that are most commonly adopted for the steel DD and steel classification.

4 CNN Models

CNNs are among the most widely used NNs in DL and are particularly suited for image- and video-related tasks. Their main strength lies in learning spatial hierarchies and automatically extracting features from raw image data. CNNs consist of the following distinct layers, each of which has a different purpose:

1. Convolutional layers apply small kernels across the input volume to create feature maps that are used to identify patterns such as edges and textures.
2. Pooling layers reduce the spatial dimensions of the feature maps, which reduces both overfitting and computational costs.
3. Activation functions allow the network to learn complex, nonlinear representations. ReLU is the most-well-known activation function, which maps all negative inputs to 0 and maps all positive inputs to themselves [59].
4. Fully connected layers connect each node from one layer to every node in the next layer. They flatten two-dimensional maps into one-dimensional feature vectors that are used for classification.

The key elements of CNNs are as follows:

1. Kernels (filters) are small matrices used to perform convolution across the input to extract features.
2. Feature maps capture spatial patterns and are generated by convolving kernels.
3. Downsampling aids in abstraction and ensuring translation invariance.
4. The ReLU activation function improves the training efficiency by promoting sparse activation and mitigating the problem of vanishing gradients [59].

The individual components of a CNN make specific contributions to the performance of the network architecture as follows:

- Convolutional layers detect local features by sliding filters over the input.
- Pooling layers compress the data and highlight desired features.
- Fully connected layers consolidate information for output predictions.
- ReLU facilitates fast and efficient training while enabling deeper networks.

Having outlined the key CNN building blocks and representative model families (contributing to RQ2), the next section describes our review protocol and the harmonized extraction template used to compare heterogeneous studies based on deployment-related criteria.

5 Methodology

This review adopted a transparent, question-driven protocol to synthesize reported evidence from representative peer-reviewed studies on DL and NN approaches for steel DD. Rather than conducting new experiments or statistically pooling results across heterogeneous settings, we performed a structured qualitative synthesis of selected case studies, and here we summarize the reported methods, evaluation setups, and deployment factors to support practical interpretations and future benchmarking.

In line with RQ1–RQ4, we selected ten representative case studies and extracted harmonized attributes using a consistent template. Matrix tables were then used to compare and cross-check key attributes across studies (e.g., task formulation, datasets, model families, training strategies, metrics, and deployment reporting) and to identify recurring engineering trade-offs and evidence gaps. The case-study synthesis was performed using the following steps:

1. Candidate records were identified through database searches and screened using the predefined inclusion and exclusion criteria (addressing RQ1).
2. Ten representative case studies were selected to cover the dominant task paradigms relevant to steel inspection (classification, object detection, and segmentation).
3. For each case study, we extracted a consistent set of methodological and reporting attributes using a harmonized template (addressing RQ2).
4. Harmonized matrix tables were constructed to enable structured cross-study comparisons and to explicitly mark missing items.
5. Reported performance and deployment metrics were summarized within each study's stated evaluation setting, and cross-study patterns and gaps were identified to inform the conceptual blueprint and reporting recommendations (addressing RQ3 and RQ4).

This methodology ensured a coherent integration of evidence, facilitating the case-by-case analysis and cross-study synthesis as reported in Section 6.

5.1 Case-Study Identification and Selection Criteria

To increase the methodological rigor and transparency, we adopted an explicit identification and selection protocol tailored to defect inspection in steel manufacturing. Candidate studies were identified using searches of major scholarly databases (e.g., IEEE Xplore, Web of Science, Scopus, and Google Scholar) with combinations of keywords such as "steel surface defect," "industrial inspection," "deep learning," "object detection," "segmentation," "YOLO," and "transfer learning." The searches focused on advancements from 2023 to 2025 (while also allowing earlier foundational studies when necessary) and was followed by title/Abstract screening and full-text eligibility assessment.

A study was included in this review if it satisfied all of the following criteria:

1. Scope relevance: Inspection of defects on steel products or closely related industrial metal-surface inspection problems with clear transferability to steel manufacturing.
2. DL-centered pipeline: DL was the primary mechanism used for classification, detection, and segmentation.
3. Methodological extractability: Sufficient information provided to identify the model family (and key design choices), training strategy (e.g., transfer learning and augmentation), and evaluation setup.
4. Reportable outcomes: Reporting of at least one accuracy performance metric [e.g., accuracy/mAP (mean average precision)/IoU] and/or deployment-relevant information (e.g., processing speed and hardware).
5. Scholarly reliability: Published in a peer-reviewed journal or reputable conference series.

A study was excluded if it satisfied any of the following criteria:

1. Not related to industrial-defect inspection (e.g., unrelated medical or consumer tasks).
2. Insufficient details provided for methodological extraction (e.g., missing model specification or evaluation description).
3. Was a duplicate or substantially overlapping version of another study (in which case the most-complete report was retained).
4. Was purely conceptual without any reported evaluation outcomes or engineering-relevant discussion.

This protocol resulted in the selection of ten representative case studies that collectively cover the dominant task paradigms in industrial inspection—classification, object detection, and segmentation—while being relevant to the constraints specific to steel manufacturing.

5.2 Data Extraction and Harmonization

To enable structured comparisons, we applied a consistent extraction template to all of the included studies, and report the resulting attributes in harmonized tables. For each included case study, we extracted the (i) inspection target and defect taxonomy, (ii) dataset type and scale, (iii) task formulation (classification, detection, and segmentation), (iv) model family and key architectural choices, (v) training strategy (transfer learning, augmentation, and epochs), (vi) deployment reporting (processing speed, latency, and hardware), and (vii) evaluation metrics [accuracy indicators (ACs), precision/recall, mAP, and/or IoU/Dice].

5.3 Handling Cross-Study Heterogeneity and Missing Metrics

Because datasets, metrics, and hardware settings can vary substantially across studies, we adopted a conservative missing-data policy and explicitly annotated unavailable items to avoid overinterpretation. We did not treat reported numeric results as statistically comparable across studies, nor do we perform meta-analytic pooling. Instead, we used harmonized fields to support deployment-related qualitative comparisons and to identify recurring engineering trade-offs (e.g., accuracy vs. throughput, annotation cost vs. robustness, and lightweight vs. heavyweight models).

To maintain transparency, missing metrics are explicitly marked as "N/A" (not available) in the tables, and we did not impute missing values. Any heuristic aggregation used for structured discussion [e.g., figure of merit (FoM) in Section 6.3] was computed only when all required inputs were reported for the source study.

6 Data Analysis

Following the protocol and harmonization rules described in Section 5, and guided by RQ1–RQ3, we analyzed the selected studies individually and then consolidated cross-study patterns using matrix tables and the FoM heuristic. Each included study was reviewed in full, and its reported information was structured into a case-level matrix table containing fields such as inspection target/description, dataset, task type, model family, evaluation metrics (ACs), and deployment reporting (e.g., throughput and hardware). In total, ten case-level tables (one per case study) were constructed to support structured cross-study comparisons.

These matrix tables facilitated deployment-related synthesis across the heterogeneous study settings. We further incorporated the FoM heuristic [18,52] as a transparent summary of reported benefit–cost parameters, where the cost proxies included the number of training epochs and model complexity (MCo) [11,15,60], and the benefit proxies included accuracy metrics [61] and throughput [11,15]. Because all of these fields were not reported for several studies, the FoM was computed only when all of these parameters were available—the missing items are explicitly marked as "N/A" to avoid overinterpretation.

The above-described matrix-based comparison and the FoM framework formed the backbone of our structured, question-driven evidence synthesis. Together they improved the deployment-related interpretation of reported DD strategies across the heterogeneous study settings, while maintaining transparency through the explicit handling of cross-study heterogeneity and missing metrics.

6.1 Analysis of DD Performance

6.1.1 Case Study 1

Case Study 1 [47] investigated an improved real-time detector that combines a lightweight design with bidirectional multiscale features to improve the sensitivity to defects of varying sizes without an excessive computational cost. The authors emphasized the practicality of their approach in the presence of manufacturing constraints, where accuracy gains must be balanced against throughput and deployment feasibility.

Trained on the GIMBDIC dataset, which was augmented to 9600 images, the M2-BL-YOLOv4 model achieved an mAP of 93.5% and a processing speed of 52.99 fps (frames per second) on 2560 × 1960-pixel images. On the NEU-DET dataset, the model achieved an mAP of 84.3%, compared with values of 82.8% and 74.9% for YOLOv5s and YOLOv7, respectively, under the same experimental settings, and it achieved a particularly high mAP of 97% for pitted-surface defects. Compared with YOLOv4-tiny and Ghost-YOLOv4, M2-BL-YOLOv4 was reported to provide a favorable speed–accuracy balance, especially for detecting small defects. Overall the reported results [47] suggest that the method is suitable for industrial scenarios requiring real-time inspections, while the cross-dataset performance should be interpreted while considering differences in the respective evaluation protocols and hardware settings.

6.1.2 Case Study 2

Case Study 2 [17] classified steel defects by using transfer learning to individually apply four CNN models that had been pretrained on the ImageNet database: VGG16, MobileNet, DenseNet121 [62], and ResNet101. The study focused on MobileNet because of its high computational efficiency associated with the implementation of depthwise separable convolutions, making it ideal for real-time tasks and addressing strategies for achieving annotation efficiency and robustness (in accordance with RQ3).

The authors applied their approach to the SSDD dataset (12,568 grayscale images of 256 × 480 pixels) and the NEU dataset (1800 images of 32 × 32 pixels across 6 classes) using an 80%–20% training–validation split. Training was conducted on Google Colab using Keras (TensorFlow backend) and a Tesla K80 GPU over 50 epochs with a batch size of 32, and the authors applied fine-tuning to adapt ImageNet-pretrained weights to the target datasets.

MobileNet reportedly outperformed VGG16, DenseNet121, and ResNet101, achieving accuracies of 80.41% and 96.94% for the SSDD and NEU datasets, respectively. Deeper models have a greater feature complexity and so require longer training times. The utilization of GPU acceleration made convergence faster, with model depth directly impacting the training duration. MobileNet's lightweight design delivered high accuracy at a relatively low computational cost, making it highly suitable for industrial applications with low computational capacities. The findings of that study indicate that steel DD can be effectively achieved using streamlined architectures and that compact models can be paired with accessible platforms such as Google Colab to accelerate deployment in real-world settings.

6.1.3 Case Study 3

Case Study 3 [63] applied a region-based detection approach to localizing steel defects that prioritized the localization accuracy, which generally has a higher computation cost and operational complexity than

using streamlined single-stage detectors. The study employed an adaptive genetic algorithm (AGA) to optimize the model parameters (e.g., kernel size, pooling dimensions, and node counts), and used heat maps from the final convolutional layer to estimate the probability of a specific type of defect (dents) and to guide bounding-box placement. The study performed training on 8017 images with a 70%–30% training–validation split, and achieved an accuracy of 98.5%, which was higher than that of 88.7% for a standard R-CNN under the same experimental settings. The sensitivity was 97.9% and the specificity was 99.2%, with a mean absolute localization error of 13.7 pixels (standard deviation = 10.7 pixels). These values were obtained after 50 AGA generations, which the authors considered would make the approach feasible for real-time DD, although they also acknowledged the associated computational complexity. Overall the reported results [63] suggest that the approach is applicable to dent localization in similar conditions, although the generalizability to other components and production environments would depend on the specific dataset, protocol, and platform setting.

6.1.4 Case Study 4

Case Study 4 [15] used an improved VGG16 architecture to detect surface defects in steel components. The inadequate accuracy and consistency of traditional methods has prompted a shift toward DL-based solutions. VGG16 was selected for its depth and strong classification ability, and its architecture was modified to improve feature extraction. The adopted dataset comprised images of 400 defect samples collected using an industrial camera, with each sample containing 100 examples. The presence of class imbalance meant that data augmentation was applied to improve the generalizability and reduce overfitting. Training was conducted over 50 epochs on a computer with an Intel i7-4790 CPU and Nvidia GTX 1060 GPU by using the TensorFlow backend.

That study evaluated the model using precision, recall, average precision, and mAP under specific experimental settings. The accuracy across all defect classes was higher for the improved VGG16 model than for a standard VGG16 model. However, the higher MCo—especially its Conv4 module with 5.9 million parameters—resulted in a high computational cost and slow responsiveness. To address these weaknesses, architectural improvements inspired by ResNet and Inception were introduced, including residual connections and multiscale feature extraction. These modifications improved the training convergence and feature learning, making the model promising for industrial DD despite its computational complexity.

6.1.5 Case Study 5

Case Study 5 [64] used a YOLOv5-based single-stage detector to identify defects in steel-pipe welds from raw X-ray images, with the goal of supporting efficient end-to-end localization and classification. The study converted a dataset of 3408 images to over 30,000 images by applying transformations such as rotation, flipping, color adjustment, noise addition, and motion deblurring in order to increase the defect diversity. The study applied an 80%–20% training–validation split, and evaluated the outcomes using the precision, recall, $F1$ score, and mAP@0.5. Using a hardware configuration comprising an Intel Core i7-4710MQ CPU and GTX950M GPU with YOLOv5 achieved an mAP@0.5 of 98.7%, precision and recall values higher than 95%, and an average processing time of 0.12 s per image. Compared with the Faster R-CNN, the study found that YOLOv5 was both faster and more accurate under the same settings (i.e., precision of 97.8% vs. 95.5%). Overall the reported results [64] suggest that a YOLOv5-style pipeline is a practical choice for inspecting welds for defects when throughput and integration simplicity are prioritized. Nevertheless, the reported performance should be interpreted while considering the dataset characteristics, augmentation policy, and the experimental and hardware settings.

6.1.6 Case Study 6

Case Study 6 [19] adopted DL for DD on industrial surfaces using a mixed-supervision strategy that combined weakly and fully supervised learning in order to reduce reliance on detailed annotations. The study developed an integrated model with a segmentation component for pixel-level analysis and a classification component for image-level labeling, with these components being trained jointly to accommodate both coarse and detailed labels. The method further incorporated training stabilizers (e.g., gradient-flow control between tasks) and a distance-transform–based emphasis on pixels near defect centers to mitigate ambiguity from inaccurate annotations. When evaluated using the DAGM, KolektorSDD, KolektorSDD2, and SSDD datasets, the study found an average precision of above 90% when there were only predominantly weak labels, and that that this performance improved substantially when detailed annotations were available for only 5% of the training samples. The authors also reported a throughput of 57 fps on DAGM and 36 fps on the SSDD dataset for their hardware and software setup. Overall the reported results [19] suggest that mixed supervision can reduce the annotation burden while maintaining a competitive performance in the tested setting. However, deployment suitability and real-time feasibility still need to be assessed for a target production platform in the presence of specific operational constraints.

6.1.7 Case Study 7

Case Study 7 [61] designed an advanced DL model called the multistage feature transformer (MSFT)-YOLO for surface DD on hot-rolled steel strips. This model is based on the YOLOv5 architecture and incorporates transformer-based modules for capturing global semantic features and a BiFPN for multiscale feature fusion. These improvements increase both the accuracy and robustness, especially for images with large-scale variance, complex backgrounds, and visual similarities between defect types. MSFT-YOLO can perform real-time detection and uses data augmentation techniques such as MixUp, CutMix, and Mosaic to improve generalizability.

This model uses stochastic gradient descent with momentum and weight decay, which increased the learning rate for the first 3 epochs before shifting to cosine annealing across 200 epochs. When trained on the NEU-DET dataset, which includes 1800 grayscale images across 6 defect categories, MSFT-YOLO achieved an mAP of 0.757, which was 7.5% higher than that of the baseline YOLOv5 model, and processed images at 30.6 fps. Although it was slower than YOLOv5, MSFT-YOLO maintained a high processing speed and detection accuracy. Despite the added complexity of MSFT-YOLO due to the inclusion of transformer-based modules and BiFPN, this model is highly effective for industrial DD. Its applicability in real-time monitoring tasks can be improved through further dataset expansion and model compression to achieve a better balance between performance and efficiency.

6.1.8 Case Study 8

Case Study 8 [1] applied an improved YOLOv5-based model to steel-surface DD with the aim of improving sensitivity across defect scales while maintaining practical processing efficiency. Rather than focusing on block-level architectural details, the key design aim was to improve multiscale feature representation and attention to defect-prone regions in order to improve localization in challenging images [65]. The study trained the model on the NEU-DET dataset (1800 grayscale images of 640 × 640 pixels) for 100 epochs using the Adam optimizer. For the test setup that included an Nvidia RTX 3080 Ti GPU, the model achieved an mAP@0.5 of 72%, and more than 190 images could be processed per second. The study found that despite the overall complexity of the model being similar to that of YOLOv5-m, it achieved better detection performance than YOLOv5-s, YOLOv5-m, and YOLOv7-tiny under the same experimental settings. Overall the reported

results [1] suggested the feasibility of high-throughput inspection within the constraints of the utilized dataset and hardware configuration. However, the robustness of the approach to overlapping and ambiguous defects and also its generalizability may require additional benchmarking in production-specific conditions.

6.1.9 Case Study 9

Case Study 9 [66] developed a refined DL model based on YOLOv5—called global and directional context perception (GDCP)-YOLO—to increase the accuracy and speed of DD in steel-surface inspection. This model integrates the following three key modules to achieve real-time processing and robustness against complex backgrounds and variations in the defect scale: DCNV2, CA_C2f_Faster, and ghost convolution modules. The DCNV2 module uses deformable convolutions to adapt its receptive fields, which improves its ability to detect defects with irregular shapes and sizes. The CA_C2f_Faster module improves the precision of feature extraction through channel attention and efficient convolution strategies, while the ghost convolution module reduces computational redundancy to enable faster processing. When tested on the NEU-DET dataset using a platform with an Intel Xeon Platinum 8358P CPU and an Nvidia RTX 3060 GPU, GDCP-YOLO achieved an mAP@0.5 value of 75.7% and processed images at 384.6 fps, confirming its real-time capability [66]. The model exhibited high efficiency and stability throughout a training schedule of 200 epochs, cosine annealing, and warm-up phases. Despite incorporating advanced modules, GDCP-YOLO is lightweight and has a relatively low parameter count, making it ideal for deployment in resource-constrained industrial environments. The adaptive spatial attention, multiscale feature extraction, and optimized training framework of GDCP-YOLO facilitate robust DD even in noisy and cluttered conditions, indicating that it is a high-performance DD solution for use in modern steel manufacturing.

6.1.10 Case Study 10

Case Study 10 [67] used an improved YOLOv5s architecture to detect small and complex defects on metal surfaces. This model incorporates CSPlayer and GAMAttention modules for increasing the accuracy and robustness in challenging conditions. The CSPlayer module improves feature representation through the use of adjustable channel weights, depthwise and pointwise convolutions, and channel attention. Moreover, the GAMAttention module improves the detection performance by combining channel and spatial attention to suppress background noise and focus on critical regions. The model was trained on an augmented GC10-DET dataset using 3608 training images, and an evaluation using 900 validation images indicated that the proposed configuration outperformed the Faster R-CNN and several YOLO variants. It achieved mAP@0.5 and mAP@0.5:0.95 values of 82.8% and 55.5%, respectively, outperforming YOLOv5s by 1.4% and 1.7%. The real-time throughput was 79.4 fps on an Nvidia RTX 3060 GPU. A comparison of attention mechanisms indicated that GAMAttention delivered the best performance for small targets such as water spots and punched holes. The joint use of CSPlayer and GAMAttention modules yielded favorable precision, speed, and efficiency. Although the performance of this lightweight model was not ideal for relatively large defects such as rolled pits, it has strong industrial applicability by offering a balanced solution for DD in scenarios involving resource constraints and requiring real-time processing.

6.2 Summary of Case Studies

Table 2 summarizes the computational complexity, accuracy, and processing speeds of the developed models and adopted datasets for the ten included case studies. The table summarizes the diverse range of strategies used to achieve rapid and precise DD in complex conditions.

Table 2: Summary of the ten case studies.

Case Study	Study Description	Model Accuracy for Specific Dataset	Processing Speed (fps)	NN Model	Number of Epochs	Computational Complexity
1	M2-BL-YOLOv4 model that contains a MobileNetV2 backbone and uses BiFPN-Lite for feature fusion	GIMBDIC: 93.5% NEU-DET: 84.3%	54.99	YOLOv4	300	LW
2	MobileNet model with depthwise separable convolutions used for efficient feature extraction with fewer parameters	SSDD: 80.41% NEU-DET: 96.94%	N/A	MobileNet	50	LW
3	R-CNN model with an AGA for dent detection; the model uses convolutional layers and heat maps for defect localization	Vehicle Fender Img.: 98.5%	N/A	R-CNN	50	HW
4	Improved VGG16 architecture for detecting surface defects on steel components	Images taken by the researchers: 77%	N/A	VGG	50	HW
5	YOLOv5 model, which is a fast, single-stage detection model that uses advanced techniques for real-time, high-accuracy defect identification	X-ray images: 98.7%	~8	YOLOv5x	633	LW
6	Two-subnetwork model combining segmentation and classification for DD with mixed supervision	DAGM: 90% KolektorSDD: 93.4% KolektorSDD2: 77.3% SSDD: 90.3%	57 N/A	VGG16	70 for DAGM, 50 for others	HW
7	A transformer-improved YOLOv5 model integrating multiscale feature fusion and contextual learning	NEU-DET: 75.7%	30.6	YOLOv5	200	LW
8	An improved multiscale YOLOv5 network with spatial attention for achieving real-time, accurate DD across various scales on steel surfaces	NEU-DET: 72%	192.3	YOLOv5	100	LW
9	Efficient, real-time GDCP-YOLO network combining deformable convolutions, spatial attention, and LW convolution for industrial applications	NEU-DET: 75.7%	384.6	YOLOv5	300	LW

(Continued)

Table 2 (continued)

Case Study	Study Description	Model Accuracy for Specific Dataset	Processing Speed (fps)	NN Model	Number of Epochs	Computational Complexity
10	YOLOv5s model improved by the addition of CSPlayer and GAMAttention modules for increasing small-target detection accuracy, processing speed, and computational efficiency	GC10-DET: 82.8%	79.4	YOLOv5	300	LW

Note: LW, lightweight; HW, heavyweight.

DL has revolutionized DD in the steel industry, with YOLOv5, VGG16, and R-CNNs being the most widely applied NN architectures. YOLOv5 is favored for its real-time speed and efficiency since its single-stage pipeline enables the rapid detection and classification of defects, making it ideal for industrial applications. Variants including MSFT-YOLO and GDCP-YOLO incorporate improvements such as BiFPN and GAMAttention that allow them to detect defects across complex backgrounds and varying scales. These additions improve the spatial resolution and so improve the detection of small defects such as oil spots and scratches.

The deep 16-layer structure of the VGG16 model endows it with high classification precision. Despite being computationally complex, its use of residual connections and multiscale feature extraction improves the performance and reduces overheads. VGG16 excels in identifying fine-grained features, particularly in applications requiring a high classification accuracy rather than real-time processing.

R-CNNs are well-suited for defect localization, by extracting candidate regions from images and then classifying them to identify defect positions. Although their multistage design reduces the processing speed, improvements such as AGAs can increase their efficiency and enable them to maintain high precision in mapping defects, making R-CNNs suitable for tasks requiring exact defect positioning.

While YOLOv5, VGG16, and R-CNNs all use convolutional layers for feature extraction, they exhibit different strengths. Specifically, YOLOv5 is fast, VGG16 can achieve high accuracy, and R-CNNs provide high-resolution defect localization. Thus, choosing between these architectures depends on the task requirements, and so these architectures make different contributions to DD in the steel industry. To further summarize the reported benefit–cost parameters in heterogeneous reporting conditions, we next introduce a transparent, review-oriented FoM together with a strict missing-data policy [65].

This summary connects the individual cases through recurring patterns (e.g., the dominance of YOLO in real-time tasks), informing the FoM heuristic presented in Section 6.3 and the blueprint presented in Section 7.

6.3 FoM: Justification, Computation, and Interpretation

To summarize the deployment-related trade-offs that are present across heterogeneous industrial studies, we introduce the FoM as a review-oriented heuristic rather than using a statistically definitive cross-study ranking. Because the datasets, defect taxonomies, evaluation protocols, and hardware platforms differed between the reviewed studies, directly pooling their results or claiming a universally optimal model would not be appropriate. Instead, the FoM is used to support transparent screening and discussion of benefit–cost parameters under the experimental settings for each study.

The FoM comprises benefit terms that include an AC [61] and the processing speed [11,15], where AC denotes the primary accuracy metric used in each study, such as accuracy, mAP, or IoU/Dice. The FoM cost terms include (i) the number of training epochs, which is a proxy for training effort in the reported setup and (ii) MCo [46], which is categorized into lightweight (MCo = 1) or heavyweight (MCo = 2) to reflect the burden of practical deployment.

To prevent any single numerical indicator from dominating the heuristic summary, all numeric parameters (AC, processing speed, and number of epochs) were scaled using min–max normalization [68,69], which for a metric x is

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{1}$$

For transparency, x_{min} and x_{max} were taken from the minimum and maximum values observed among the included case studies for which that metric is reported (see Table 3). MCo was not normalized because it is a binary category.

Table 3: FoM parameters for the ten studies.

Case Study	Dataset	Normalized AC	Normalized Processing Speed (fps)	Normalized Number of Epochs	MCo
1	GIMBDIC NEU-DET	0.9382 0.8348	0.1195	0.4288	1
2	SSDD NEU-DET	0.7911 0.9769	N/A	N/A	1
3	Vehicle Fender Img.	0.9944	N/A	N/A	2
4	Steel images taken by the researchers	0.7528	N/A	N/A	2
5	X-ray images	0.966	0	1.00	1
6	DAGM KolektorSDD KolektorSDD2 SSDD	0.8989 0.9371 0.7112 0.9022	0.1322 N/A N/A 0.0765	0.0343 N/A N/A 0	2
7	NEU-DET	0.7382	0.0765	0.2573	1
8	NEU-DET	0.6967	0.4894	0.088	1
9	NEU-DET	0.7382	1.00	0.4288	1
10	GC10-DET	0.8180	0.2427	0.4288	1

The FoM is defined using normalized indicators as

$$\text{FoM} = \frac{AC_{norm} + Processing\,speed_{norm}}{Number\,of\,epochs_{norm} + MCo} \tag{2}$$

This formulation reflects an engineering preference for higher reported benefits (inspection performance and throughput feasibility) relative to higher reported costs (training burden and deployment complexity). FoM is intended for coarse screening and grouping (e.g., throughput-oriented vs. accuracy-oriented candidates) and should not be treated as a standalone decision criterion.

To avoid *ad-hoc* handling, the FoM was computed only when all of the required parameters (AC, processing speed, number of epochs, and MCo) were available from the source study. If any required field was missing, the FoM is reported here as N/A (i.e., missing data) and no imputation was performed. Due to the cross-study heterogeneity in datasets, metrics, evaluation conditions, and hardware, FoM values must be interpreted strictly within each study's reported setting, and they were used to identify deployment-related patterns and reporting gaps rather than to claim statistically meaningful global superiority across studies.

The normalization process ensured that each variable contributed equally to the FoM. Some case studies were excluded from the FoM calculation because data could not be obtained for those studies. The FoM results for each case study are presented in Table 4.

Table 4: FoM values for the ten case studies.

Case Study	Dataset	Calculation	Value
1	GIMBDIC	$FoM = \frac{(0.9382 + 0.1195)}{(0.4288 + 1)}$	0.7403
	NEU-DET	$FoM = \frac{(0.8348 + 0.1195)}{(0.4288 + 1)}$	0.6679
2	SSDD	$FoM = \frac{(0.7911 + FPS)}{(0 + 1)}$	N/A (missing processing speed)
	NEU-DET	$FoM = \frac{(0.9769 + FPS)}{(0 + 1)}$	N/A (missing processing speed)
3	Vehicle Fender Img.	$FoM = \frac{(0.9944 + FPS)}{(0 + 2)}$	N/A (missing processing speed)
4	Steel images taken by the researchers	$FoM = \frac{(0.7528 + FPS)}{(0 + 2)}$	N/A (missing processing speed)
5	X-ray images	$FoM = \frac{(0.9966 + 0)}{(1 + 2)}$	0.3322
6	DAGM	$FoM = \frac{(0.8989 + 0.1322)}{(0.0343 + 1)}$	0.9969
	KolektorSDD	$FoM = \frac{(0.9371 + FPS)}{(0 + 1)}$	N/A (missing processing speed)
	KolektorSDD2	$FoM = \frac{(0.9371 + FPS)}{(0 + 1)}$	N/A (missing processing speed)
	SSDD	$FoM = \frac{(0.9022 + 0.0765)}{(0 + 1)}$	0.9787

(Continued)

Table 4 (continued)

Case Study	Dataset	Calculation	Value
7	NEU-DET	$FoM = \frac{(0.7382 + 0.0600)}{(0.2573 + 1)}$	0.6349
8	NEU-DET	$FoM = \frac{(0.6967 + 0.4894)}{(0.0858 + 1)}$	1.092
9	NEU-DET	$FoM = \frac{(0.7382 + 1)}{(0.4288 + 1)}$	1.217
10	GC10-DET	$FoM = \frac{(0.8180 + 0.2427)}{(0.4288 + 1)}$	0.7424

The normalized FoM results can be summarized as follows: Case Study 9 achieved an FoM of 1.217 on the NEU-DET dataset for a lightweight YOLOv5-based configuration, Case Study 8 achieved an FoM of 1.092, which can be used for contextual comparisons within the FoM heuristic rather than as a definitive rank across heterogeneous studies, and Case Study 5 achieved an FoM of 0.3322 (i.e., lower than the values for the lightweight models) for the only heavyweight model included in the FoM calculations. For the reported metrics available from the selected case studies and the review-oriented FoM heuristic, Case Study 9 appears to have had the most-competitive configuration; however, this observation should be interpreted cautiously due to cross-study heterogeneity and missing metrics, and it should not be treated as a statistically definitive global ranking. Overall the case synthesis and FoM heuristic provide a structured basis for making design and deployment-critical decisions, which are discussed in the following section.

The FoM thus integrates the heterogeneous metrics from Sections 3–6, highlighting engineering trade-offs that advance the understanding beyond isolated summaries.

7 Results and Discussion

Drawing from the interconnected evidence and patterns identified in the case analyses as described in Section 6, this section presents a conceptual blueprint for a hybrid framework synthesized from the reviewed studies, which is intended as an engineering template for integrating commonly reported approaches (e.g., transfer learning, real-time detection, and end-to-end learning) in the presence of real manufacturing constraints—it is not introduced as a new standalone algorithm with original experimental validation in this review.

7.1 Conceptual Hybrid Framework Blueprint for Steel-Defect Inspection

This blueprint serves as a unified, deployment-related framework that integrates the model families listed in Section 4, the evidence synthesis protocol presented in Section 5, and the reported trade-offs summarized in the case analyses and the discussion of FoM values in Section 6 in order to improve the understanding beyond isolated summaries.

The reviewed studies collectively suggest that the performance in practical industrial applications depends not only on ACs but also on robustness to texture and illumination variations, tolerance to domain shift, and the ability to meet production-line constraints. Accordingly, the blueprint combines three commonly recurring components: (i) transfer learning to reduce labeling requirements and accelerate convergence when annotation is scarce, (ii) real-time detection to support high-throughput inspection and localized defect identification, and (iii) end-to-end learning to reduce pipeline fragmentation and simplify integration across stages.

Practitioners and researchers can implement the blueprint in the presence of application-specific constraints as follows: (i) define the inspection objective and constraints, including task type (classification, detection, or segmentation), minimum detectable defect size, acceptable false-alarm tolerance, target throughput, and available hardware; (ii) characterize the data regime, including the defect taxonomy stability, illumination and texture variations, and labeling budget; (iii) select a task-appropriate baseline pipeline (e.g., one-stage detection when there is a strict latency requirement, segmentation when precise localization is required, and classification for coarse acceptance screening); (iv) choose efficient and robust labeling strategies that are consistent with the data type (e.g., transfer learning as the default, and semisupervised or self-supervised learning or domain adaptation or generalizability when label scarcity or domain shift dominates [2,3]); and (v) validate the approach in deployment-like conditions and iterate it based on observed failure modes (e.g., rare defects, long-tail classes, and cross-line drift).

The heterogeneity of datasets, metrics, and hardware settings among the reviewed studies restricted the ability to perform direct statistical comparisons. To enable reproducible benchmarking and evaluations of practical deployment, we recommend reporting the following aspects: (i) defect taxonomy and labeling policy; (ii) dataset split protocol and data leakage controls; (iii) task-appropriate metrics (e.g., precision/recall and mAP for detection, and IoU/Dice for segmentation) with a thresholding policy; (iv) latency and throughput measured for the specified hardware and software; (v) model size and complexity, and the memory footprint; and (vi) the effects of removing each blueprint component in deployment-related conditions in order to quantify its contribution.

The blueprint is offered as a synthesis of widely used and practically motivated design choices when inspecting steel for defects. However, its empirical performance and generalizability still need to be established in dedicated benchmarking experiments, which were beyond the scope of this review. Fig. 15 illustrates the blueprint at a conceptual level; accordingly, the framework should be interpreted as a design guideline that consolidates reported best practices and highlights integration points and expected trade-offs. For completeness and reproducibility, commonly used evaluation metrics are summarized in Section 7.2.

7.2 Validation Metrics

To support the reproducible interpretation of the reported results across the heterogeneous studies, this section summarizes commonly used evaluation metrics in defect inspection, addressing RQ1 regarding reporting heterogeneity and RQ4 regarding standardized benchmarking. Section 7.3 then discusses future trends and challenges in the presence of real manufacturing constraints.

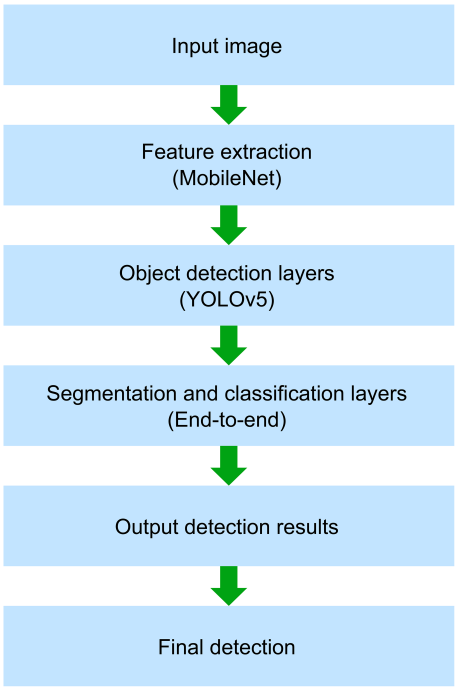

Figure 15: Flow of the conceptual hybrid framework blueprint synthesized from the reviewed literature.

Specifically, the evaluation metrics commonly used in the reviewed defect-detection studies are summarized as follows:

1. Accuracy refers to the ratio of correctly identified defects to all actual defects in the dataset [18,19,58]. This parameter is expressed as follows:

$$Accuracy = \frac{Number\,of\,Correctly\,Classified\,Samples}{Total\,Number\,of\,Samples} * 100\% \tag{3}$$

2. Precision refers to the proportion of correctly detected defects among all predictions:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{4}$$

where TP is the number of true positives and FP is the number of false positives. Recall refers to the proportion of correctly detected defects among all true positives:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{5}$$

where FN is the number of false negatives.

3. mAP refers to the mean of the average precision values corresponding to multiple recall thresholds; it indicates overall object detection quality and is expressed as follows:

$$\text{mAP} = \frac{\sum_{k=1}^{N} Pinter(k) \times \Delta r(k)}{N} \tag{6}$$

4. The detection time (or latency) refers to the time required by the model to analyze and process each image (or frame). When available, latency should be reported for the same measurement settings

used to determine the processing speed (hardware, input size, and pipeline scope) in order to support reproducible interpretation across studies.

5. Throughput is an efficacy metric defined as the number of input frames processed per second. The throughput complements latency by characterizing the sustained processing capacity, and it is frequently used to communicate the deployment practicality in a high-throughput production environment. Similar to latency, the throughput should be reported along with the measurement conditions—including CPU and GPU types, input resolution, batch size, and pipeline scope (including or excluding pre- and postprocessing)—to avoid comparisons across heterogeneous studies being misleading. This review treated the throughput as a deployment-related indicator that complements accuracy metrics, and is summarized in Tables 2 and 3.
6. The $F1$ score is the harmonic average of precision and recall; it is used as a balanced indicator of overall performance and is expressed as follows:

$$F1\ \text{score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{7}$$

Together these metrics provide a common vocabulary for interpreting reported results. The next section discusses emerging trends and open challenges when these methods are deployed in the presence of real manufacturing constraints.

7.3 Future Trends and Challenges

Recent progress in DL-based defect inspection has led to increases in data efficiency, cross-domain robustness, and deployment feasibility in real manufacturing environments. To mitigate the labeling cost, recent studies have explored label-efficient regimes such as active learning and self-supervised pretraining that reduce annotation requirements while preserving the detection accuracy [2,3,70]. For steel-defect inspection in particular, future research is likely to be driven by three related aspects: (i) label-efficient learning in the presence of long-tail defect distributions, (ii) generalizability during domain shift across production lines and surface conditions [4], and (iii) engineering constraints that determine whether a model can be reliably deployed at the processing speed and latency required for production-line operation.

Label-efficient few-shot, weakly/semisupervised, and self-supervised learning approaches are increasingly being investigated for use in industrial inspection applications because pixel-level annotations are costly and rare defects often follow a long-tail distribution [1–3,70]. Few-shot learning approaches are particularly relevant when new production batches introduce novel defect patterns that appear in only a small number of labeled samples [51,71,72]. Recent surveys have further summarized how limited-label settings influence evaluation protocols and practical adoption in industrial defect inspection [73,74]. A persistent bottleneck in industrial inspection is the high cost of annotation—especially for pixel-level segmentation masks—and the scarcity of rare-defect samples. Few-shot learning and meta-learning paradigms are promising for rapidly adapting to new defect types with small numbers of labeled examples, while weakly/semisupervised approaches can utilize coarse labels or partially labeled data to reduce the annotation burden. Self-supervised learning is promising for steel inspection because unlabeled line-scan imagery is abundant [2,3]; the key problem to be addressed is designing pretraining objectives that transfer to fine-grained defect cues and remain robust during domain shift [4]. Recent self-supervised studies of steel inspection have provided concrete evidence that representation learning can improve the downstream performance when labeled data are scarce, while still requiring careful validation in production-representative conditions [2,3]. In parallel, self-supervised representation learning can be used to pretrain feature extractors using large volumes of unlabeled production imagery, again improving

the downstream performance when labeled data are scarce and enabling more-robust initialization for fine-tuning. These strategies are particularly relevant to steel manufacturing since defect occurrence often follows a long-tail distribution and the surface appearance can change across different material batches and with the tooling and environmental conditions.

Training and operational data are often mismatched during production deployment due to illumination variations, texture changes, surface-finish differences, sensor upgrades, and shifts across production lines. Cross-domain deployment increasingly couples model design with edge hardware constraints, and so reporting latency and throughput is essential for ensuring reproducibility. Domain generalizability and adaptation are therefore expected to remain central research themes. Practical pipelines may benefit from explicit cross-dataset evaluations, continuous monitoring for distribution shifts, and adaptation strategies that minimize reliance on extensive target-domain labels. Robustness-oriented evaluations should therefore move beyond single-dataset reporting toward stress tests that reflect the variability in real-world industrial environments (e.g., mixed lighting conditions, motion blur, surface contamination, and subtle low-contrast defects), with clear disclosure of dataset splits and operating conditions.

Accuracy is not the only consideration in steel inspection. Deployment-related metrics—including throughput and latency measured on specified hardware, memory footprint, and stability during sustained operation—obtained in future studies need to be reported more consistently so that the results can be used to make optimal production decisions. Hardware-specific optimization (e.g., model compression, identification, quantization, and efficient processing formats) is likely to remain important, particularly for edge deployment in production lines with constrained computation and power budgets. Beyond initial deployment, maintainability becomes a dominant cost driver, since models may require periodic recalibration or retraining, robust failure detection, and traceable logging to support quality audits. This means that practical progress will depend not only on improving values of the assessment metrics but also on standardized reporting practices and reproducible benchmarking that jointly consider performance, speed, and operational reliability. These trends highlight the need for more-standardized reporting and deployment-related evaluations.

8 Conclusion

The main novel aspect of this review lies in it performing a deployment-related synthesis for steel-defect inspection: rather than introducing a new experimentally validated model, we have consolidated representative evidence across task paradigms and proposed a traceable hybrid-framework blueprint and practical reporting recommendations to improve reproducibility and industrial applicability.

This review examined DL-based defect inspection approaches that are relevant to steel manufacturing and synthesized insights from ten representative case studies spanning classification, detection, and segmentation paradigms. Our synthesis suggests that model selection in industrial inspection should be guided by deployment-related trade-offs—including processing speed and latency, hardware constraints, annotation cost, and robustness to domain shift—rather than accuracy alone.

To support structured discussions of the reported benefit–cost indicators, we introduced the transparent, review-oriented FoM as a heuristic summary of available metrics. It should be noted that FoM-based comparisons are conditional on the metrics reported for each study and should not be interpreted as providing statistically definitive rankings across heterogeneous datasets and experimental conditions. Instead, the FoM is used to highlight practical patterns and engineering considerations that may inform applied decision-making within the context of the reported setting of each study.

Based on recurring elements across the reviewed literature, we synthesized a conceptual hybrid framework blueprint that integrates transfer learning, real-time detection, and end-to-end learning as an engineering-oriented design template for steel-defect inspection systems. This blueprint is presented as a synthesis derived from previous studies, and so it does not constitute an experimentally validated new method.

This review was guided by its four RQs. In addressing RQ1, the synthesis revealed common tasks (e.g., detection and segmentation) and datasets (e.g., NEU-DET and SSDD), but also highlighted reporting heterogeneity as a key limitation for comparability. RQ2 identified recurring model families (e.g., YOLO variants) with rationales focused on speed–accuracy trade-offs. RQ3 identified mitigation strategies such as transfer learning and mixed supervision to address annotation scarcity and domain shift. Finally, RQ4 informed the blueprint and recommendations for standardized benchmarking to increase the reproducibility.

Future research should prioritize label-efficient learning (few-shot/self-supervised), cross-dataset robustness, and hardware-specific evaluations to ensure that reported benchmarks are consistent with real production-line conditions [3,4,11,12]. Future studies may also benefit from (i) label-efficient learning strategies (e.g., few-shot, self-supervised, and semisupervised learning approaches), (ii) systematic approaches to cross-domain generalizability and adaptation, and (iii) more-standardized benchmarking practices that report both performance and deployment metrics (hardware, throughput, and latency) to facilitate reproducible and meaningful comparisons between real manufacturing environments.

Acknowledgement: The authors extend their heartfelt gratitude to all the researchers and scholars whose pioneering studies are discussed and cited in this review. We are deeply indebted to them for making their invaluable work publicly available, which provided the essential foundation for this synthesis. Their dedication to advancing scientific knowledge is both recognized and greatly appreciated.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: The authors confirm their specific contributions to the paper as follows: study conception and design: Duane G. Noé, Yung-Tsung Cheng; Literature search and screening: Duane G. Noé, Ku-Chin Lin, Chang-Lin Chuang; Analysis and interpretation of results: Duane G. Noé, Ku-Chin Lin, Chang-Lin Chuang, Yung-Tsung Cheng; Draft manuscript preparation: Duane G. Noé; Manuscript review, track changes, and editing: Ku-Chin Lin, Chang-Lin Chuang, Yung-Tsung Cheng; Supervision, project management, and administrative oversight: Yung-Tsung Cheng. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: This review is based exclusively on the synthesis of previously published studies; therefore, no new primary datasets were generated or analyzed. All data supporting the findings of this review are contained within the cited articles listed in the reference section. Readers are encouraged to refer to the original publications for detailed underlying data and experimental methodologies.

Ethics Approval: Not applicable. This study is a systematic review of existing literature and did not involve any direct research on human participants or animal subjects.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Božič J, Tabernik D, Skočaj D. Mixed supervision for surface-defect detection: from weakly to fully supervised learning. Comput Ind. 2021;129(1):103459. doi:10.1016/j.compind.2021.103459.
2. Zhang H, Li S, Miao Q, Fang R, Xue S, Hu Q, et al. Surface defect detection of hot rolled steel based on multi-scale feature fusion and attention mechanism residual block. Sci Rep. 2024;14(1):7671. doi:10.1038/s41598-024-57990-3.

3. Ling L, Xu S, Wei L, Wei G, Jia J, Hong B. Fsd-detr: casting surface defect detection based on improved RT-DETR. J Real Time Image Process. 2025;22(4):135. doi:10.1007/s11554-025-01714-x.
4. Song Z, Huang X, Ji C, Zhang Y, Chao Z, Peng Y. Cross-domain fine grained strip steel defect detection method based on semi-supervised learning and Multi-head Self Attention coordination. Comput Electr Eng. 2025;121(3):109916. doi:10.1016/j.compeleceng.2024.109916.
5. Hu X, Yang J, Jiang F, Hussain A, Dashtipour K, Gogate M. Steel surface defect detection based on self-supervised contrastive representation learning with matching metric. Appl Soft Comput. 2023;145(11):110578. doi:10.1016/j.asoc.2023.110578.
6. Mao K, Wei P, Wang Y, Liu M, Wang S, Zheng N. CSDD: a benchmark dataset for casting surface defect detection and segmentation. IEEE/CAA J Autom Sin. 2025;12(5):947–60. doi:10.1109/JAS.2025.125228.
7. He Y, Li S, Wen X, Xu J. A survey on surface defect inspection based on generative models in manufacturing. Appl Sci. 2024;14(15):6774. doi:10.3390/app14156774.
8. Jain R, Kasturi R, Schunck BG. Machine vision. New York, NY, USA: McGraw-Hill; 1995.
9. Newman TS, Jain AK. A survey of automated visual inspection. Comput Vis Image Underst. 1995;61(2):231–62. doi:10.1006/cviu.1995.1017.
10. Chen F, Fu L, Zhang Y, Li J, Zhang Q, Bi S. A review of deep learning-based steel surface defect detection. Acad J Sci Technol. 2025;15(1):198–202. doi:10.54097/g36nm962.
11. Yang D, Cui Y, Yu Z, Yuan H. Deep learning based steel pipe weld defect detection. Appl Artif Intell. 2021;35(15):1237–49. doi:10.1080/08839514.2021.1975391.
12. Tang B, Chen L, Sun W, Lin ZK. Review of surface defect detection of steel products based on machine vision. IET Image Process. 2023;17(2):303–22. doi:10.1049/ipr2.12647.
13. Song K, Yan Y. A noise robust method based on completed local binary patterns for hot-rolled steel strip surface defects. Appl Surf Sci. 2013;285:858–64. doi:10.1016/j.apsusc.2013.09.002.
14. Rabuñal JR, Dorado J, Sierra AP. What is defect detection. In: Encyclopedia of artificial intelligence. Vol. 1–3. New York, NY, USA: IGI Global; 2009.
15. Recht B, Roelofs R, Schmidt L, Shankar V. Do CIFAR-10 classifiers generalize to CIFAR-10? arXiv:1806.00451. 2018.
16. Gai X, Ye P, Wang J, Wang B. Research on defect detection method for steel metal surface based on deep learning. In: Proceedings of the 2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC); 2020 Jun 12–14; Chongqing, China. p. 637–41. doi:10.1109/itoec49072.2020.9141669.
17. Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, et al. MobileNets: efficient convolutional neural networks for mobile vision applications. arXiv:1704.04861. 2017.
18. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2016 Jun 27–30; Las Vegas, NV, USA. p. 770–8. doi:10.1109/CVPR.2016.90.
19. Abu M, Amir A, Lean YH, Zahri NH, Azemi SA. The performance analysis of transfer learning for steel defect detection by using deep learning. J Phys Conf Ser. 2021;1755(1):012041. doi:10.1088/1742-6596/1755/1/012041.
20. Bozic J, Tabernik D, Skocaj D. End-to-end training of a two-stage neural network for defect detection. In: Proceedings of the 2020 25th International Conference on Pattern Recognition (ICPR); 2021 Jan 10–15; Milan, Italy. p. 5619–26. doi:10.1109/icpr48806.2021.9412092.
21. Krizhevsky A. Learning multiple layers of features from tiny images. Toronto, ON, Canada: University of Toronto; 2009 [cited 2026 Jan 1]. Available from: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.
22. Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Commun ACM. 2017;60(6):84–90. doi:10.1145/3065386.
23. Mehrish A, Majumder N, Bharadwaj R, Mihalcea R, Poria S. A review of deep learning techniques for speech processing. arXiv:2305.00359. 2023.
24. Hinton GE, Osindero S, Teh YW. A fast learning algorithm for deep belief nets. Neural Comput. 2006;18(7):1527–54. doi:10.1162/neco.2006.18.7.1527.
25. Hinton GE, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov RR. Improving neural networks by preventing co-adaptation of feature detectors. arXiv:1207.0580. 2012.

26. Bengio Y, Courville A, Vincent P. Representation learning: a review and new perspectives. IEEE Trans Pattern Anal Mach Intell. 2013;35(8):1798–828. doi:10.1109/TPAMI.2013.50.
27. Ameri R, Hsu CC, Band SS. A systematic review of deep learning approaches for surface defect detection in industrial applications. Eng Appl Artif Intell. 2024;130(3):107717. doi:10.1016/j.engappai.2023.107717.
28. Tabernik D, Šela S, Skvarč J, Skočaj D. Segmentation-based deep-learning approach for surface-defect detection. J Intell Manuf. 2020;31(3):759–76. doi:10.1007/s10845-019-01476-x.
29. Schmidhuber J. Deep learning in neural networks: an overview. Neural Netw. 2015;61(3):85–117. doi:10.1016/j.neunet.2014.09.003.
30. Robotyshyn M, Sharkadi M, Malyar M. Surface defect detection based on deep learning approach. In: Proceedings of the II International Scientific Symposium «Intelligent Solutions» IntSol-2021; 2021 Sep 28–30; Kyiv-Uzhhorod, Ukraine. p. 32–42.
31. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556. 2014.
32. Aravindan A, Greenwood H, Varshney A. CNN for bulk material defect detection. Stanford, CA, USA: Stanford University; 2019 [cited 2026 Jan 1]. Available from: http://cs230.stanford.edu/projects_fall_2019/reports/26251851.pdf.
33. Gillis A, Craig L, Awati R. What is a convolutional neural network (CNN)? Newton, MA, USA: Enterprise AI, TechTarget [cited 2026 Jan 1]. Available from: http://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network.
34. Jia Y, Shelhamer E, Donahue J, Karayev S, Long J, Girshick R, et al. Caffe: convolutional architecture for fast feature embedding. In: Proceedings of the 22nd ACM International Conference on Multimedia; 2014 Nov 3–7; Orlando, FL, USA. p. 675–8. doi:10.1145/2647868.2654889.
35. Lin M, Chen Q, Yan S. Network in network. arXiv:1312.4400. 2013.
36. Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation. arXiv:1411.4038. 2014.
37. Redmon J, Ali F. YOLOv3: an incremental improvement. arXiv:1804.02767. 2018.
38. Vyas SM. Severstal steel defect detection. Medium. [cited 2026 Jan 1]. Available from: https://medium.com/@smv9495/severstal-steel-defect-detection-1b5966dc0512.
39. Kundu R. Yolo algorithm for object detection explained. [cited 2026 Jan 1]. Available from: www.v7labs.com/blog/yolo-object-detection.
40. Ren R, Hung T, Tan KC. A generic deep-learning-based approach for automated surface inspection. IEEE Trans Cybern. 2018;48(3):929–40. doi:10.1109/TCYB.2017.2668395.
41. Zeiler MD, Fergus R. Visualizing and understanding convolutional networks. In: Computer vision—ECCV 2014. Cham, Switzerland: Springer International Publishing; 2014. p. 818–33. doi:10.1007/978-3-319-10590-1_53.
42. Zhao ZQ, Zheng P, Xu ST, Wu X. Object detection with deep learning: a review. IEEE Trans Neural Netw Learn Syst. 2019;30(11):3212–32. doi:10.1109/TNNLS.2018.2876865.
43. Thakkar V, Tewary S, Chakraborty C. Batch normalization in convolutional neural networks—a comparative study with CIFAR-10 data. In: Proceedings of the 2018 Fifth International Conference on Emerging Applications of Information Technology (EAIT); 2018 Jan 12–13; Kolkata, India. p. 1–5. doi:10.1109/eait.2018.8470438.
44. Weimer D, Scholz-Reiter B, Shpitalni M. Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection. CIRP Ann. 2016;65(1):417–20. doi:10.1016/j.cirp.2016.04.072.
45. Thuan DC. Evolution of YOLO algorithm and YOLOv5: the state-of-the-art object detection algorithm. [cited 2026 Jan 1]. Available from: http://urn.fi/URN:NBN:fi:amk-202103042892.
46. Bochkovskiy A, Wang CY, Liao HM. YOLOv4: optimal speed and accuracy of object detection. arXiv:2004.10934. 2020.
47. Li S, Guo S, Han Z, Kou C, Huang B, Luan M. Aluminum surface defect detection method based on a lightweight YOLOv4 network. Sci Rep. 2023;13(1):11077. doi:10.1038/s41598-023-38085-x.
48. Sandler M, Howard A, Zhu M, Zhmoginov A, Chen LC. MobileNetV2: inverted residuals and linear bottlenecks. arXiv:1801.04381. 2018.

49. Yang J, Li S, Wang Z, Dong H, Wang J, Tang S. Using deep learning to detect defects in manufacturing: a comprehensive survey and current challenges. Materials. 2020;13(24):5755. doi:10.3390/ma13245755.
50. Frydrych K, Tomczak M, Jasiński J, Papanikolaou S. Steel surface defects analysis with machine vision and deep learning. Int J Adv Manuf Technol. 2025;140(7):3691–710. doi:10.1007/s00170-025-16539-y.
51. Cumbajin E, Rodrigues N, Costa P, Miragaia R, Frazão L, Costa N, et al. A systematic review on deep learning with CNNs applied to surface defect detection. J Imaging. 2023;9(10):193. doi:10.3390/jimaging9100193.
52. Ma Y, Yin J, Huang F, Li Q. Surface defect inspection of industrial products with object detection deep networks: a systematic review. Artif Intell Rev. 2024;57(12):333. doi:10.1007/s10462-024-10956-3.
53. Kaggle. Severstal: steel defect detection. [cited 2026 Jan 1]. Available from: https://kaggle.com/competitions/severstal-steel-defect-detection.
54. Bergmann P, Batzner K, Fauser M, Sattlegger D, Steger C. The MVTec anomaly detection dataset: a comprehensive real-world dataset for unsupervised anomaly detection. Int J Comput Vis. 2021;129(4):1038–59. doi:10.1007/s11263-020-01400-4.
55. Lv X, Duan F, Jiang JJ, Fu X, Gan L. Deep metallic surface defect detection: the new benchmark and detection network. Sensors. 2020;20(6):1562. doi:10.3390/s20061562.
56. Feng X, Gao X, Luo L. X-SDD: a new benchmark for hot rolled steel strip surface defects detection. Symmetry. 2021;13(4):706. doi:10.3390/sym13040706.
57. Guan S, Lei M, Lu H. A steel surface defect recognition algorithm based on improved deep learning network model using feature visualization and quality evaluation. IEEE Access. 2020;8:49885–95. doi:10.1109/ACCESS.2020.2979755.
58. Great Learning Team. AlexNet: the first CNN to win Image Net. Great Learning; 2022 [cited 2026 Jan 1]. Available from: https://www.mygreatlearning.com/blog/alexnet-the-first-cnn-to-win-image-net/.
59. Xu B, Wang N, Chen T, Li M. Empirical evaluation of rectified activations in convolutional network. arXiv:1505.00853. 2015.
60. Park SH, Tjolleng A, Chang J, Cha M, Park J, Jung K. Detecting and localizing dents on vehicle bodies using region-based convolutional neural network. Appl Sci. 2020;10(4):1250. doi:10.3390/app10041250.
61. Guo Z, Wang C, Yang G, Huang Z, Li G. MSFT-YOLO: improved YOLOv5 based on transformer for detecting defects of steel surface. Sensors. 2022;22(9):3467. doi:10.3390/s22093467.
62. Huang G, Liu Z, Van Der Maaten L, Weinberger KQ. Densely connected convolutional networks. In: Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2017 Jul 21–26; Honolulu, HI, USA. p. 2261–9. doi:10.1109/CVPR.2017.243.
63. Wang L, Liu X, Ma J, Su W, Li H. Real-time steel surface defect detection with improved multi-scale YOLO-v5. Processes. 2023;11(5):1357. doi:10.3390/pr11051357.
64. Zhou C, Lu Z, Lv Z, Meng M, Tan Y, Xia K, et al. Metal surface defect detection based on improved YOLOv5. Sci Rep. 2023;13(1):20803. doi:10.1038/s41598-023-47716-2.
65. Chen C, Lee H, Chen M. Steel surface defect detection method based on improved YOLOv9. Sci Rep. 2025;15(1):25098. doi:10.1038/s41598-025-10647-1.
66. Yuan Z, Ning H, Tang X, Yang Z. GDCP-YOLO: enhancing steel surface defect detection using lightweight machine learning approach. Electronics. 2024;13(7):1388. doi:10.3390/electronics13071388.
67. Nguyen VH, Pham VH, Cui X, Ma M, Kim H. Design and evaluation of features and classifiers for OLED panel defect recognition in machine vision. J Inf Telecommun. 2017;1(4):334–50. doi:10.1080/24751839.2017.1355717.
68. Shantal M, Othman Z, Abu Bakar A. A novel approach for data feature weighting using correlation coefficients and Min–max normalization. Symmetry. 2023;15(12):2185. doi:10.3390/sym15122185.
69. Deshpande AM, Minai AA, Kumar M. One-shot recognition of manufacturing defects in steel surfaces. Procedia Manuf. 2020;48(12):1064–71. doi:10.1016/j.promfg.2020.05.146.
70. Lv X, Duan F, Jiang JJ, Fu X, Gan L. Deep active learning for surface defect detection. Sensors. 2020;20(6):1650. doi:10.3390/s20061650.
71. Zajec P, Rožanec JM, Theodoropoulos S, Fontul M, Koehorst E, Fortuna B, et al. Few-shot learning for defect detection in manufacturing. Int J Prod Res. 2024;62(19):6979–98. doi:10.1080/00207543.2024.2316279.

72. Li H, Liu Y, Liu C, Pang H, Xu K. A few-shot steel surface defect generation method based on diffusion models. Sensors. 2025;25(10):3038. doi:10.3390/s25103038.
73. Jin Q, Chen L. A survey of surface defect detection of industrial products based on a small number of labeled data. arXiv:2203.05733. 2022.
74. Hu S, Ma X, Zhang Y, Xu W. Application of self-supervised learning in steel surface defect detection. J Mater Inf. 2025;5(4):44. doi:10.20517/jmi.2025.21.