frontiers in Electronics

TYPE Original Research
PUBLISHED 12 March 2026
DOI 10.3389/felec.2026.1750707

Check for updates

OPEN ACCESS

EDITED BY
Yiqi Liu,
South China University of Technology,
China

REVIEWED BY
Hongxu Li,
Wuxi University, China
Navneet Singh,
Bennett University, India
Doaa A. Altantawy,
Mansoura University, Egypt

*CORRESPONDENCE
Md. Saiful Islam,
saiful05eee@cuet.ac.bd
Hezerul bin Abdul Karim,
hezerul@mmu.edu.my

RECEIVED 20 November 2025
REVISED 08 January 2026
ACCEPTED 26 January 2026
PUBLISHED 12 March 2026

CITATION
Khatun MR, Farid FA, Dhar S, Islam MS, Uddin J and Abdul Karim Hb (2026) CBAM-enhanced lightweight CNN for wafer map defect classification. *Front. Electron.* 7:1750707. doi: 10.3389/felec.2026.1750707

COPYRIGHT
© 2026 Khatun, Farid, Dhar, Islam, Uddin and Abdul Karim. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.

CBAM-enhanced lightweight CNN for wafer map defect classification

Mst. Rokeya Khatun[1], Fahmid Al Farid[2], Sharith Dhar[3], Md. Saiful Islam[1]*, Jia Uddin[4] and Hezerul bin Abdul Karim[2]*

[1]Department of Electronics and Telecommunication Engineering, Chittagong University of Engineering and Technology, Chattogram, Bangladesh, [2]Centre for Image and Vision Computing (CIVC), Centre of Excellence for Artificial Intelligence, Faculty of Artificial Intelligence and Engineering (FAIE), Multimedia University, Cyberjaya, Selangor, Malaysia, [3]Department of Electrical and Electronic Engineering, Premier University, Chattogram, Bangladesh, [4]Department of AI and Big Data, Endicott College, Woosong University, Daejeon, Republic of Korea

Automated interpretation of wafer maps is central to manufacturing quality monitoring. Identifying rare defects with less detailed wafer maps is a challenging task. Moreover, class imbalance, heavyweight backbones, and limited model transparency are constraints for the real-world deployment of defective wafer identification. However, a nine-class wafer-map classifier is required that maintains high accuracy under tight parameter and compute budgets and provides decision-level interpretability, despite long-tailed class distributions. To address this issue, a compact convolutional network is presented for wafer-map classification on standardized low-resolution inputs. The architecture uses two convolution–pooling stages, followed by a modified convolutional block attention module (CBAM). Channel attention is realized via a shared multilayer perceptron with batch normalization for stable reweighting, while spatial attention uses a multi-scale gate to emphasize ring-like, edge-localized, and streak patterns. A compact dense head with softmax produces nine class probabilities, with a total footprint of approximately 0.15M parameters. Class imbalance is mitigated through a training-only convolutional autoencoder that generates minority samples via latent feature variation, together with a controlled reduction in the dominant `None` class. Validation and test sets remain unchanged. A fixed-seed protocol ensures reproducibility, and performance is evaluated using accuracy and macro-averaged precision, recall, and F1. On a balanced benchmark derived from the WM-811K dataset, the model achieves 99.88% test accuracy with near-ceiling macro-F1 while using a small fraction of the parameters required by transfer learning and transformer baselines and consistently outperforming conventional convolutional neural network (CNN) backbones. Post-training interpretability analyses with Grad-CAM, integrated gradients (IG), and occlusion show alignment between salient regions and physically meaningful defect morphology. Ablation studies indicate complementary gains from latent feature augmentation and attention mechanisms, while robustness checks with input noise and reduced training support show graceful degradation. The resulting pipeline is accurate, lightweight, and transparent, making it suitable for inline screening scenarios.

KEYWORDS
convolutional block attention module–convolutional neural network, lightweight architecture, wafer defect, WM-811K, XAI

1 Introduction

1.1 Background

Detection of defects in silicon wafers is very crucial in the semiconductor industry as defective wafers result in the manufacturing of low-quality electronic chips. For this reason, precise anomaly detection in the wafer is becoming a major concern in the fabrication process. There are several defective wafer patterns; it is becoming increasingly difficult to identify subtle defect patterns such as "Scratch" in the presence of a noisy background. Consequently, semiconductor production management increasingly depends on automatic interpretation of wafer maps, where spatially structured defect patterns encode actionable process information for root cause analysis and control (Wu et al., 2014; Theodosiou et al., 2023; Hsu et al., 2020). Automated wafer-map analytics have, therefore, become a core capability alongside traditional inspection, with modern studies reporting that defect-pattern classification directly supports excursion detection and similarity retrieval across lots (Wu et al., 2014; Hsu et al. 2020). However, a major problem is that publicly used benchmark datasets, such as WM-811K, have low-resolution wafer maps. Traditional detection methods struggle in identifying defects in these wafers. WM-811K contains on the order of 8×10^5 wafer maps partitioned into nine canonical classes (Center, Donut, Edge-Loc, Edge-Ring, Loc, Random, Scratch, Near-Full, and None); only a subset is labeled, and the distribution is highly long-tailed, with the background None class dominating and genuinely defective wafers being relatively scarce (MathWorks, 2025; Tsai and Wang 2025; Jeong et al., 2023). Across several summaries, approximately 55k maps carry explicit defect labels, while a large fraction remain unlabeled or marked as None, underscoring the rarity of positive examples and the risk that accuracy metrics alone may mask poor minority-class recall (MathWorks, 2025; Tsai and Wang, 2025; Jeong et al., 2023). In addition, the representation is typically a low-resolution binary or categorical grid; many implementations standardize maps to an 26×26 grid with integer codes for background/normal/defect, which accentuates undersampling of thin rings or scratches and complicates separability unless models are encouraged to focus on morphology (MathWorks, 2025; Tsai and Wang, 2025; Ward, 2022).

These properties situate wafer-map classification squarely within imbalanced learning, where long-tailed distributions, minority-class sparsity, and label uncertainty degrade generalization if not addressed by balancing, augmentation, or tailored objectives (Theodosiou et al., 2023; Chen et al., 2024). Contemporary surveys highlight that even strong deep models can be biased toward majority classes unless the training signal is reweighted or augmented; this observation aligns with reports specific to wafer maps that emphasize skewed class priors and scarcity of certain patterns (Khalil and Islam, 2022; Tsai and Wang, 2025; Jeong et al., 2023; Singh 2023; Heydari and Mahmoud, 2025; Ehsanul Haque et al., 2025).

A second practical constraint is computational efficiency. Many high-accuracy classifiers rely on deep backbones or heavy attention modules whose memory footprint and latency are ill-suited to inline inspection on fab equipment. As a result, there is persistent interest in lightweight architectures and compression techniques that preserve accuracy under strict compute and parameter budgets, aligning with broader edge-AI design goals (Singh, 2023; Shin and Yoo, 2023; Liu et al., 2024; Khalil et al., 2019). Within the wafer-map domain, both efficient backbones and attention mechanisms have been explored. Nevertheless, the need to balance accuracy with runtime and footprint for deployment scenarios is repeatedly noted (Shin and Yoo, 2023; Chen et al., 2022). Finally, interpretability is required for operational trust: black-box predictions hinder corrective action and auditability. Explainable AI (XAI) techniques, such as Grad-CAM, integrated gradients (IG), and occlusion, provide complementary evidence at class-level, pixel-level, and causal (score-drop) perspectives and have become standard tools for assessing whether learned features align with physically meaningful wafer morphology (Selvaraju et al., 2017; Sundararajan et al., 2017; Zeiler and Fergus, 2014). In manufacturing-oriented studies, qualitative maps are often paired with quantitative checks to support reliable decision-making; calibration itself has been shown to matter for high-stakes applications (Guo et al., 2017).

In this study, we address rare-defect recognition on low-resolution wafer maps under long-tailed priors, subject to strict compute and memory budgets, while providing transparent, decision-faithful explanations suitable for deployment. We use WM-811K in a single-label, nine-class regime, standardized to $26 \times 26 \times 3$; the results are reported on a stratified 70/10/20 split with fixed seed 2019, CPU-only execution, and no augmentation on validation and test sets.

1.2 Contribution

The constraints posed by long-tailed class priors and low spatial resolution were first addressed at the data level. Wafer morphology was preserved by admitting only native 26×26 maps (no resizing or padding) and converting each map to a three-channel, pixel-wise one-hot tensor so that discrete die semantics were retained. Label integrity was enforced by removing null labels and mapping the remainder to a fixed nine-class taxonomy (MathWorks, 2025; Tsai and Wang, 2025; Jeong et al., 2023). Minority-class scarcity was counteracted during training using a lightweight convolutional autoencoder that generated class-consistent variants through small latent feature variation (Bao et al., 2024; Yu et al., 2023; Wei et al., 2023; Hu et al., 2021), while the dominant None class was reduced by uniform downsampling to prevent bias. Validation and test partitions were kept augmentation-free, ensuring that reported generalization reflected performance on real data.

Computational and memory budgets were met with a compact convolutional neural network (CNN) into which attention was integrated in a parameter-efficient manner. Two shallow convolution–pooling blocks formed the backbone, after which a modified convolutional block attention module (CBAM) was applied: channel re-weighting was realized with a shared two-layer MLP stabilized by batch normalization and sigmoid gating, and spatial focus was obtained by a multi-scale branch (3×3, 5×5, and 7×7), whose outputs were fused by averaging and complemented with a residual path to preserve information flow. Placing attention after pooling increased the effective receptive field at minimal overhead, enabling reliable discrimination of thin rings, edge bands, and localized anomalies while maintaining a sub-

megabyte footprint suitable for CPU-only execution (Woo et al., 2018; Shin and Yoo, 2023; Liu et al., 2024; Wang et al., 2018).

Model transparency and reliability were addressed by integrating complementary attribution tools—Grad-CAM, IG, and occlusion sensitivity—so that defect-relevant regions could be inspected and compared with expected wafer morphology (Selvaraju et al., 2017; Sundararajan et al., 2017; Zeiler and Fergus, 2014). Beyond headline accuracy, calibration quality and robustness were evaluated; low expected calibration error and stable behavior under additive Gaussian noise and reduced training support were observed (Guo et al., 2017). Ablation and stress studies attributed measurable gains to the imbalance-aware augmentation and attention modules (with the multi-scale spatial branch providing the largest share within CBAM). Taken together, these elements constitute an accurate, computationally efficient, and interpretable pipeline that directly targets the intertwined challenges of class imbalance, model footprint, and decision auditability.

Taken together, the contribution is an accurate, computationally efficient, and transparent solution that directly addresses the intertwined challenges of imbalance, model footprint, and interpretability highlighted in prior work (Theodosiou et al., 2023; Wu et al. 2014; Hsu et al., 2020; MathWorks, 2025; Tsai and Wang, 2025; Jeong et al., 2023; Chen et al., 2024; Singh, 2023; Heydari and Mahmoud, 2025; Shin and Yoo, 2023; Liu et al., 2024; Wang et al., 2018; Woo et al., 2018; Selvaraju et al., 2017; Sundararajan et al., 2017; Zeiler and Fergus, 2014, Guo et al., 2017).

1.3 Related works

Semiconductor defect prediction has been the subject of sustained study, evolving from traditional statistics and classical machine learning (ML) to deep learning (DL), attention-augmented CNNs, and, more recently, transformers. Early lines of work demonstrated that spatially structured wafer-map patterns are informative for root-cause analysis and retrieval across lots, revealing practical limits when models face low-resolution grids, long-tailed class distributions, and deployment constraints (Wu et al., 2014; Theodosiou et al., 2023; Hsu et al., 2020; MathWorks, 2025; Tsai and Wang, 2025; Jeong et al., 2023; Singh, 2023; Heydari and Mahmoud, 2025; Ehsanul Haque et al., 2025). In what follows, related efforts are grouped by methodology, emphasizing strengths and limitations relevant to the present problem setting. Table 1 summarizes key works on wafer-defect classification in chronological order (2015–2025), listing each study's approach, dataset context (primarily WM-811K/factory maps), and reported limitations alongside the year.

1.3.1 Machine learning-based approaches

Classical ML pipelines such as decision trees, random forests (RFs), and support vector machines (SVMs) typically operate on hand-crafted morphology features (radial or edge profiles, connected components, and texture descriptors) extracted from wafer maps or on structured process-control data. Such systems can be interpretable and efficient and can retrieve or distinguish canonical patterns to a useful degree in early studies (Wu et al., 2014; Hsu et al., 2020). However, performance depends strongly on feature design and invariances (rotation/flip), and generalization degrades under the long-tailed priors characteristic of public corpora such as WM-811K, where the None class dominates and rare defects are underrepresented (MathWorks, 2025; Tsai and Wang, 2025; Jeong et al., 2023). Reports consistently indicate that accuracy can mask low-minority recall without explicit rebalancing, motivating the transition to representation learning (Theodosiou et al., 2023; Tsai and Wang, 2025; Jeong et al., 2023; Singh, 2023; Heydari and Mahmoud, 2025).

TABLE 1 Training configuration. Fixed schedule used for all reported runs (main model, baselines, and ablations).

Item	Setting
Input tensor	26 × 26 × 3 (pixel-level one-hot)
Classes	9 (center, donut, edge-loc, edge-ring, loc, random, scratch, near-full, none)
Optimizer/LR	RMSprop/0.001
Batch size/ Epochs	128/30
Loss/Output	Categorical cross-entropy/9-way softmax
Train/Test split	70%/10%/20% (stratified), `random_state = 2019`
Device/Platform	CPU (Google Colab free tier)

Same configuration applied across all experiments; no resizing or normalization required

1.3.2 Deep learning-based approaches (CNNs and hybrids)

CNNs improved recognition by learning hierarchical features directly from wafer maps, lifting accuracy over classical baselines on WM-811K-like datasets (Tsai and Wang, 2025; Jeong et al., 2023; Chen et al., 2022). Encoder–decoder and autoencoder hybrids have been adopted both for representation learning and augmentation: convolutional autoencoders and latent perturbation schemes enlarge minority-class support and have reported consistent macro-F1 gains under imbalance (Yu et al., 2023; Bao et al., 2024). Semi-supervised and contrastive formulations further reduce label dependence and improve robustness when annotations are scarce (Wei et al., 2023; Hu et al., 2021). Nevertheless, many high-performing CNNs rely on deep backbones with sizable memory and latency footprints, and several works highlight the need to balance accuracy with runtime for inline deployment (Singh, 2023; Heydari and Mahmoud, 2025; Shin and Yoo, 2023; Liu et al., 2024; Wang et al., 2018). Capsule-style models (WaferCaps/WaferCap) have also been explored to encode part–whole relations and to support open-set rejection; these approaches report improvements in minority-class F1 and robustness to unknown patterns but at nontrivial compute cost (Mishra et al., 2024; Abd Al Rahman et al., 2021).

1.3.3 Attention and lightweight CNNs

Attention mechanisms have been introduced to sharpen feature selectivity under tight budgets. The CBAM re-weights channels and spatial locations with modest overhead and has been shown to offer favorable accuracy–efficiency trade-offs; recent

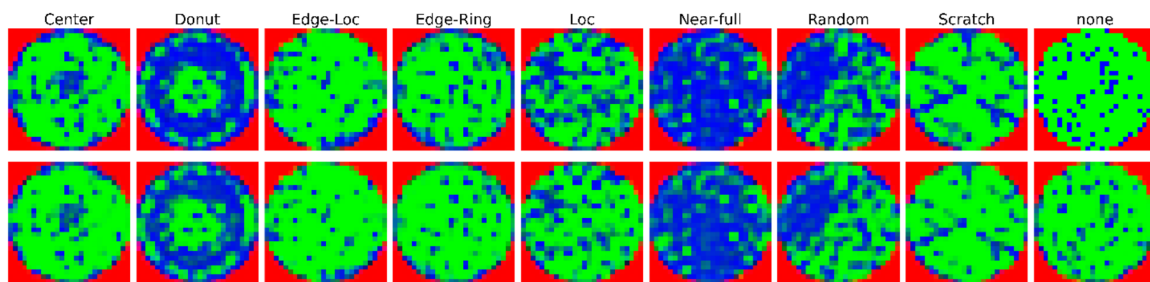

FIGURE 1
Sample wafer defect maps illustrating original wafers (top row) and their augmented counterparts (bottom row).

wafer-map studies adopt CBAM to emphasize ring/edge morphologies without large parameter growth (Woo et al., 2018; Shin and Yoo, 2023). Complementary trends in lightweight design—SqueezeNet, MobileNetV2, ShuffleNet/ShuffleNetV2, EfficientNet, and related families—provide practical blueprints for meeting edge-AI constraints on parameters and MACs while preserving accuracy (Singh, 2023; Heydari and Mahmoud, 2025; Liu et al., 2024; Wang et al., 2018). Within wafer-map classification, efficient backbones coupled with selective attention have been repeatedly advocated as a pathway to real-time deployment on fab hardware (Shin and Yoo, 2023; Chen et al., 2022; Woo et al., 2018).

1.3.4 Transformers and hybrid vision models

Transformer-based approaches have been explored to capture global context and long-range dependencies that may be missed by the limited receptive fields of CNNs. Vision transformer (ViT) variants and hybrid designs have reported performance gains on complex or mixed-type wafer patterns and when used as learned augmenters, particularly in scenarios where multi-scale cues are important (Fan and Chiu, 2024; Zhang et al., 2025). At the same time, transformer models can be data-hungry and compute-intensive; studies emphasize the continued relevance of compact CNNs or CNN–transformer hybrids when latency, memory, and power budgets are strict (Singh, 2023; Heydari and Mahmoud, 2025; Shin and Yoo, 2023; Liu et al., 2024; Wang et al., 2018; Fan and Chiu, 2024; Zhang et al., 2025).

1.3.5 Explainability and calibration for deployment

For manufacturing use, XAI has become essential. Grad-CAM, IG, and occlusion/meaningful perturbations provide complementary class-level, pixel-level, and causal (score-drop) evidence and are now standard for verifying that decisions align with physically meaningful wafer morphology (Selvaraju et al., 2017; Sundararajan et al., 2017; Zeiler and Fergus, 2014). Quantitative faithfulness tests (ROAR) and probability calibration analyses (temperature scaling) are increasingly paired with qualitative maps to ensure reliable, auditable behavior in high-stakes settings (Guo et al., 2017).

Across methodologies, the literature converges on three pain points that motivate the present work: (i) severe dataset imbalance with rare yet operationally critical defects; (ii) the need for lightweight but highly discriminative models that meet fab-floor latency and memory budgets; and (iii) the requirement for transparent predictions supported by XAI and proper calibration. Prior studies contribute key components—imbalance handling, efficient backbones, attention, transformers, augmentation, and explainability—but an integrated solution that jointly addresses all three constraints remains the practical target advanced here.

2 Dataset and preprocessing

This section describes the WM-811K wafer-map dataset and the preprocessing pipeline used to obtain uniform 26×26 inputs, followed by the class balancing strategy adopted for training under severe label imbalance.

2.1 Dataset (WM-811K) description

WM-811K is a large-scale wafer-map dataset comprising approximately 811k samples collected from multiple fabrication tools and lots. Among these, approximately 55k wafers are reliably labeled into nine defect classes: Center, Donut, Edge-Loc, Edge-Ring, Loc, Random, Scratch, Near-Full, and `None`. Each wafer map is represented as a discrete die grid, where each die takes one of three states: background/no-die, pass, or fail.

Representative examples of the eight patterned defect classes are shown in Figure 1. To ensure consistency and avoid interpolation artifacts, only wafers with native 26×26 resolution were retained and samples with ambiguous or conflicting labels were removed, resulting in 14,366 labeled wafer maps, denoted D_{26}.

2.2 Preprocessing and class balancing

All preprocessing steps are applied to the labeled subset unless stated otherwise. Wafer maps were first restricted to a fixed 26×26 die grid to enable consistent receptive-field design and lightweight modeling. Let D_0 denote the labeled corpus and $D_{26} \subset D_0$ the retained set, as shown in Equation 1:

$$D_{26} = \{ X \in D_0 \mid X \in \{0, 1, 2\}^{26 \times 26} \}, \tag{1}$$

10.3389/felec.2026.1750707

TABLE 2 Per-class accuracy, precision, and recall for baseline and proposed models evaluated on 26 × 26 × 3 wafer maps (nine classes).

Model	Measure	Center	Donut	Edge-loc	Edge-ring	Loc	Scratch	Near-full	Random	None
ShuffleNet-v2-CNN	Accuracy	0.952	0.965	0.976	0.958	0.965	0.984	0.985	0.969	0.968
	Precision	0.938	0.951	0.959	0.945	0.938	0.972	0.961	0.968	0.954
	Recall	0.949	0.962	0.966	0.951	0.958	0.977	0.963	0.975	0.962
ResNet50	Accuracy	0.77	0.90	0.88	0.95	0.59	0.70	0.95	0.87	0.89
	Precision	0.79	0.86	0.86	1.00	0.65	0.70	0.90	0.89	0.91
	Recall	0.75	0.95	0.90	0.90	0.55	0.70	1.00	0.85	0.88
EfficientNetB0	Accuracy	0.90	0.90	0.87	1.00	0.82	0.68	0.92	0.85	0.92
	Precision	0.90	0.90	0.90	0.94	1.00	0.62	0.90	0.83	0.94
	Recall	0.90	0.90	0.90	1.00	0.70	0.75	0.95	1.00	0.90
VGG16	Accuracy	0.89	0.90	0.77	0.95	0.63	0.56	0.95	0.83	0.88
	Precision	0.94	0.86	0.74	1.00	0.62	0.58	0.90	0.94	0.92
	Recall	0.85	0.95	0.81	0.90	0.65	0.55	1.00	0.75	0.83
DC-net	Accuracy	0.99	0.978	0.965	0.944	0.998	0.98	0.958	0.934	0.991
	Precision	0.994	0.978	0.972	0.939	0.988	0.934	0.965	0.957	0.997
	Recall	0.983	0.982	0.967	0.942	0.967	0.983	0.952	0.929	0.995
Proposed CBAM–CNN	Accuracy	0.99	1.00	0.99	1.00	0.98	0.99	1.00	1.00	0.99
	Precision	0.99	1.00	1.00	0.99	0.99	1.00	1.00	1.00	0.99
	Recall	1.00	1.00	0.98	1.00	0.96	0.99	1.00	1.00	0.98

where values $\{0, 1, 2\}$ correspond to background/no-die, pass, and fail states, respectively.

Each retained map X was converted into a three-channel one-hot tensor $T \in \{0, 1\}^{26\times26\times3}$ using Equation 2:

$$T_{(i,j,c)} = \mathbf{1}\left[X_{(i,j)} = c \right], \quad c \in \{0, 1, 2\}. \tag{2}$$

No smoothing or intensity normalization was applied to preserve die-level discreteness. Samples with missing labels, conflicting annotations, or ambiguous multi-pattern tags were removed, yielding a cleaned set $D_{\text{clean}} \subset D_{26}$.

Class imbalance was addressed using a two-pronged strategy applied *only* to the training split: (i) learned augmentation for minority classes and (ii) controlled downsampling of the dominant `None` class. A lightweight convolutional encoder–decoder (≈40.5k parameters) was trained on the training data to learn a compact latent representation of wafer maps. Synthetic samples were generated by encoding a real training sample, perturbing the latent code with Gaussian noise, and decoding back to image space, followed by channel-wise $\arg\max$ projection to obtain valid one-hot maps. Generated samples were visually inspected to ensure preservation of physically meaningful defect structures (e.g., ring continuity, edge localization, and clustered failures), and malformed outputs were discarded. For each minority class, approximately 2,000 valid synthetic samples were retained.

To limit majority-class bias, the `None` class was downsampled from 13,489 to 2,489 samples (81.6% reduction, ~ 5.4 ×), yielding a class scale comparable to the augmented minority classes. This reduction substantially improved gradient balance and training stability during optimization. Validation and test sets were kept entirely real and were never used for fitting augmentation models or tuning parameters. Table 2 summarizes the per-class sample counts after filtering and balancing for the training split.

3 Proposed methodology

The methodology targets reliable classification of discrete wafer-map imagery with a compact model that is easy to deploy and explain. Inputs are standardized to a 26 × 26 die grid and one-hot, three-channel representation (background/no-die, pass, and fail). The training subset is balanced using learned augmentation for minority classes and downsampling of the dominant "None" class, while validation and test sets remain real-only to prevent data leakage. A lightweight backbone—two convolutional stages with interleaved attention—feeds a small classifier head [`Flatten` → `Dense(128, ReLU)` → `Dense(9, softmax)`], yielding a total complexity of roughly 153k parameters (≈ 0.6 MB). Post-hoc XAI (e.g., Grad-CAM and IG) is used strictly for interpretability and diagnostics; explanation maps are not used as inputs or training signals.

Figure 2 visualizes the end-to-end workflow: raw wafer maps → preprocessing stack (label mapping, 26 × 26 filtering, missing/

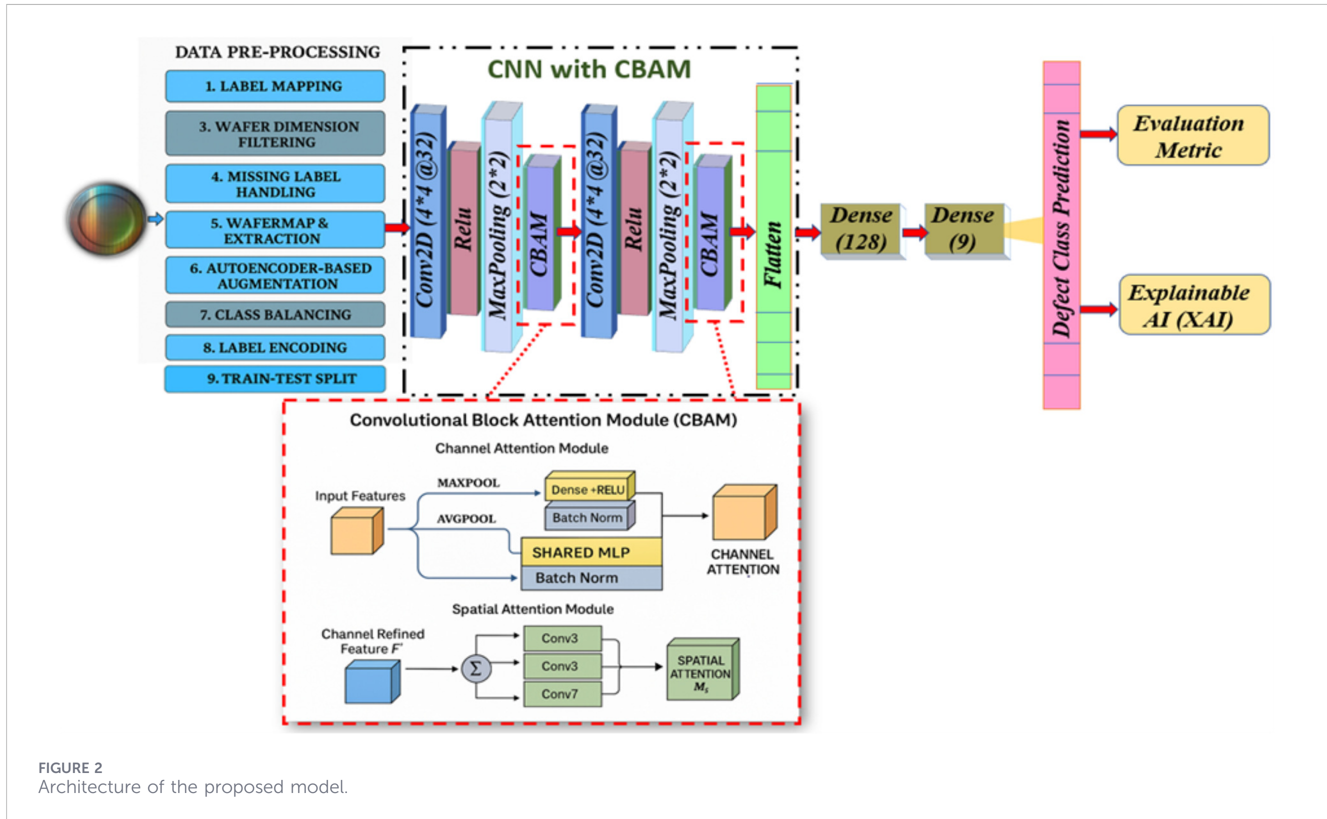

FIGURE 2
Architecture of the proposed model.

ambiguous label handling, wafer extraction and one-hot encoding, training-only balancing, and stratified splitting) → CNN + modified CBAM. In the network block, each Conv–ReLU–MaxPool stage is followed by a modified CBAM: channel attention aggregates global max/avg descriptors and passes them through a shared MLP with BatchNorm, while spatial attention applies multi-scale convolutions (3 × 3, 5 × 5, and 7 × 7) to capture fine scratches, local blobs, and broad rings. The attention-refined features are flattened and classified into nine defect types; the rightmost branch of the figure indicates evaluation metrics and *post-hoc* explanation tools applied after training.

Given the fixed 26 × 26 input resolution, two convolution–pooling stages are sufficient to achieve an effective receptive field that covers most of the wafer surface, making deeper backbones redundant while increasing computational cost. Standard CBAM was, therefore, adapted with stabilization and multi-scale spatial gating to better suit low-resolution, morphology-driven wafer patterns.

3.1 Design constraints and deployment considerations

The proposed architecture was designed under explicit deployment-oriented constraints derived from the characteristics of the WM-811K dataset and practical wafer-screening requirements. First, wafer maps were standardized to their native 26 × 26 resolution to avoid interpolation artifacts and preserve die-level topology, resulting in an input tensor of size 26 × 26 × 3. Second, to support real-time or near-real-time inspection on resource-constrained platforms, the model size was restricted to below 0.2M parameters (approximately 0.6 MB in FP32), yielding a final architecture with 0.15M parameters and ~0.02 GFLOPs per inference.

Heavy backbones (deep CNNs or transformers with > 10M parameters) were, therefore, avoided due to their higher memory footprint and latency, which are disproportionate for low-resolution inputs. Attention mechanisms were introduced only after spatial downsampling stages (11× 11 and 4× 4 feature maps) to enhance defect saliency with minimal computational overhead. Finally, class imbalance was addressed at the data level via training-only augmentation and controlled downsampling, ensuring robustness without increasing inference-time cost. The quantitative design constraints are summarized in Table 3.

3.2 CNN backbone architecture

In Figure 3, the left-to-right flow illustrates the two Conv–ReLU–Pool blocks, with CBAM blocks applied immediately after each pooling operation. The representation after the second attention gate is flattened and fed to the two fully connected layers; the right margin lists the nine target classes of the softmax. A legend maps colors to layer types (convolution, pooling, CBAM, flatten, fully connected, and softmax).

The classifier maps a preprocessed wafer tensor $T \in \{0, 1\}^{26\times26\times3}$ (background/no-die, pass, and fail) to a nine-class probability vector. The backbone is deliberately compact: two Conv–ReLU–MaxPool stages with valid padding—the first uses a 4 × 4 kernel with 32 output channels, and the second, a 3 × 3 kernel with 64 output channels. Each pooling operation is immediately followed by a CBAM

TABLE 3 Quantitative design constraints guiding the proposed model.

Constraint	Value
Input resolution	26 × 26 × 3
Parameter budget	≤ 0.2 M
Final parameter count	0.15 M
Model size (FP32)	~0.6 MB
Computational cost	~0.02 GFLOPs
Inference setting	CPU/edge-compatible
Attention placement	After pooling (11× 11, 4× 4)

attention gate. This placement allows the network to emphasize defect-relevant responses—annular rings, edge bands, local blobs, and thin scratches—immediately after spatial downsampling, where selectivity is most beneficial at low cost.

The local feature extractor at each stage is the discrete 2D convolution (implemented as cross-correlation in modern libraries), which aggregates a $k \times k$ neighborhood across all input channels to produce each output channel, as presented in Equation 3:

$$\left[X_{(l-1)} * W^{(l)}\right]_{(i,j,c)} = \sum_{u=0}^{k-1} \sum_{v=0}^{k-1} \sum_{\gamma=1}^{C_{(l-1)}} W^{(l)}_{(u,v,\gamma,c)} X_{(l-1,\, i+u-p_t,\, j+v-p_l,\, \gamma)}. \quad (3)$$

Immediately after the linear filtering, a per-channel bias is applied, and the result is passed through a rectifier to obtain sparse, nonnegative activations that are well matched to discrete wafer imagery (Equation 4):

$$F_l = \phi\left(X_{(l-1)} * W^{(l)} + b^{(l)}\right), \quad \phi(x) = \max(0, x). \quad (4)$$

Because convolutions use stride $s = 1$ with "same" padding, the spatial size is preserved across the convolution step; the relationship between input and output dimensions is therefore presented in Equations 5, 6:

$$H_l = \left\lfloor \frac{H_{(l-1)} + p_t + p_b - k}{s} \right\rfloor + 1, \quad (5)$$

$$W_l = \left\lfloor \frac{W_{(l-1)} + p_l + p_r - k}{s} \right\rfloor + 1, \quad (6)$$

which reduces to $H_l = H_{(l-1)}$ and $W_l = W_{(l-1)}$, respectively, under the standard padding choice $p_t + p_b = p_l + p_r = k - 1$ (possibly asymmetric for even k).

After each convolution, spatial resolution is reduced to build invariance and contain compute; this is accomplished by a 2×2 max-pool operator with stride 2 that halves height and width while preserving channel count (Equation 7):

$$P_l(i, j, c) = \max_{0 \le u,v < 2} F_l(2i + u, 2j + v, c), \quad H_l^{\text{pool}} = \lfloor H_l/2 \rfloor, \quad W_l^{\text{pool}} = \lfloor W_l/2 \rfloor. \quad (7)$$

Chaining preservation under convolution, together with downsampling under pooling, produces the exact size progression used later by the classifier head; across the two stages, the tensor evolves as follows:

$$26 \times 26 \times 3 \xrightarrow[\text{conv1}]{} 23 \times 23 \times 32 \xrightarrow[\text{pool1}]{} 11 \times 11 \times 32 \xrightarrow[\text{conv2}]{} 9 \times 9 \times 64 \xrightarrow[\text{pool2}]{} 4 \times 4 \times 64.$$

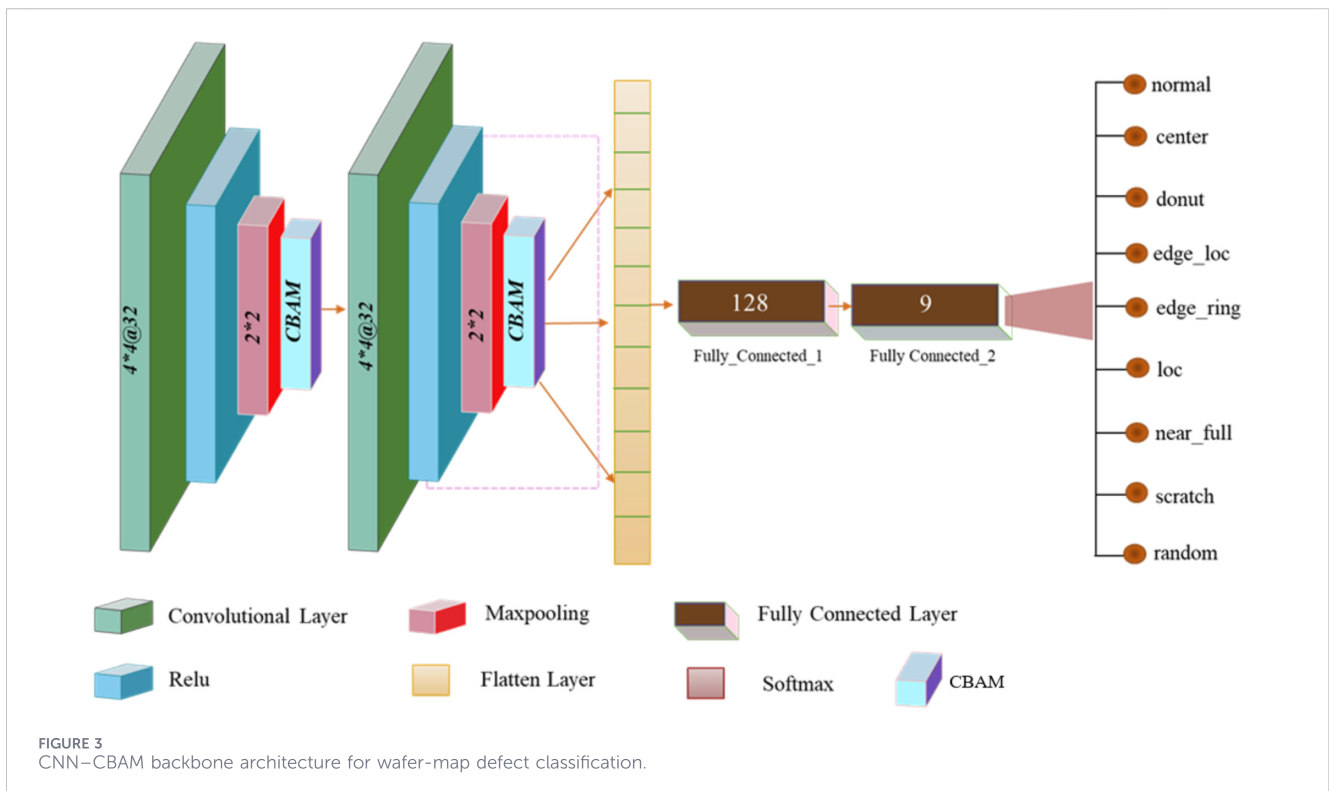

FIGURE 3
CNN–CBAM backbone architecture for wafer-map defect classification.

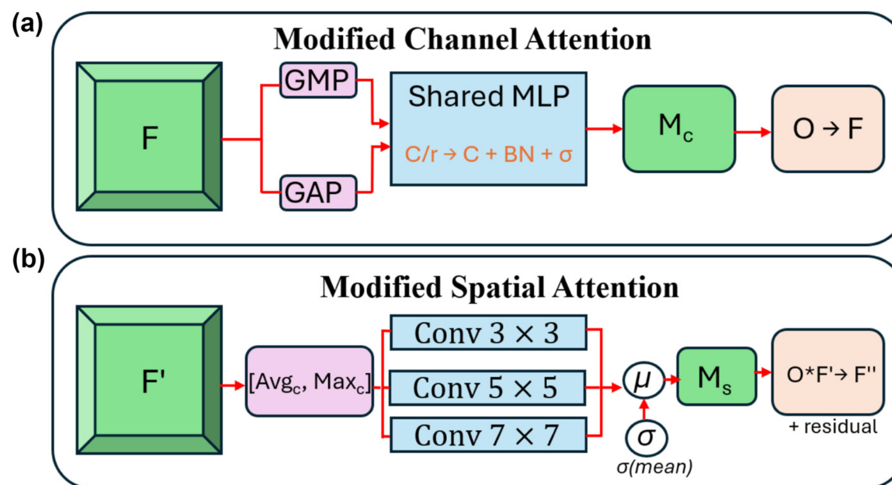

FIGURE 4
Modified CBAM for wafer-map classification: **(a)** channel attention; **(b)** spatial attention.

Beyond raw sizes, it is useful to quantify how much of the input each deeper activation "sees." Starting from a single input cell, a $k \times k$ convolution with stride 1 enlarges the receptive field by $k - 1$, and a 2×2 pooling with stride 2 doubles it. Therefore, the effective receptive field after the full stack can be obtained using the standard recursion (Equation 8):

$$r_l = r_{(l-1)} + (k_l - 1) j_{(l-1)}, \quad j_l = j_{(l-1)} s_l, \quad \text{with} \quad r_0 = j_0 = 1. \tag{8}$$

The progression is as follows:

$$\begin{aligned} \text{conv1 } (k = 4, s = 1) &\Rightarrow r_1 = 4, \ j_1 = 1; \\ \text{pool1 } (k = 2, s = 2) &\Rightarrow r_2 = 5, \ j_2 = 2; \\ \text{conv2 } (k = 3, s = 1) &\Rightarrow r_3 = 9, \ j_3 = 2; \\ \text{pool2 } (k = 2, s = 2) &\Rightarrow r_4 = 11, \ j_4 = 4. \end{aligned}$$

Thus, each activation in the final 4×4 map summarizes an 11×11 input neighborhood. Hence, each unit in the final feature map summarizes approximately a 22×22 region of the 26×26 wafer—sufficient to capture ring thickness, edge belts, and long scratches while still retaining local detail.

Finally, immediately after each pooling (where attention is cheapest and most impactful), features are refined by a channel-then-spatial gating sequence following the CBAM paradigm; the gates rescale channels globally and locations uniformly across channels via broadcast element-wise products (Equation 9):

$$X' = M_c(P_l) \odot P_l, \quad X_l = M_s(X') \odot X'. \tag{9}$$

This ordering first prioritizes channel groups that encode defect-relevant motifs (rings, edge bands, and scratches) and then concentrates responses on the wafer regions where those motifs manifest. Concrete instantiations of M_c and M_s (common vs. modified) are specified in the attention subsection that follows.

3.3 Modified CBAM

Attention is inserted after each pooling stage, where the spatial grid is small (11×11 after the first pool and 4×4 after the second) and features already encode the multi-die context. Standard CBAM is retargeted to the 26×26 wafer regime by (i) a stabilized, parameter-efficient channel gate and (ii) a multi-scale spatial gate. A lightweight residual skip is also added to guard against over-suppression. The goal is to improve defect saliency (rings, edge belts, local clusters, and thin scratches) without materially increasing model size.

Figure 4 shows the modified CBAM used in the classifier. The top panel applies channel attention: global max/avg pooling of F feeds a shared MLP ($C/r \rightarrow C$, BatchNorm, and σ) to produce a channel mask M_c that reweights F to F'. The bottom panel applies spatial attention: the squeezed maps $[\text{Avg}_c, \text{Max}_c]$ pass through parallel 3×3, 5×5, and 7×7 convolutions and are mean-fused and squashed (σ) to yield M_s, which gates F' to $\tilde{F}$; an optional residual skip is shown. $C = 32$ after the first pooling layer and $C = 64$ after the second; the reduction ratio r follows the implementation.

3.3.1 Channel attention-shared MLP with BatchNorm

Given a feature tensor $F \in \mathbb{R}^{H\times W\times C}$ (here, $C = 32$), CBAM first computes global per-channel descriptors by average and max pooling over spatial locations (Woo et al., 2018) (Equation 10):

$$s_{\text{avg}} = \text{GAP}(F) \in \mathbb{R}^C, \quad s_{\text{max}} = \text{GMP}(F) \in \mathbb{R}^C. \tag{10}$$

Both descriptors are processed by the same two-layer MLP with the reduction ratio r (tuned for low-resolution feature maps; $r = 8$), and batch normalization is applied after each affine transformation:

TABLE 4 Per-class counts post-filtering and post-balancing (training split; $n = 19,707$), shown as n (%).

Class	After 26 × 26 Filter	(%)	Final post-aug	(%)
Center	90	0.63	2160	10.96
Donut	1	0.01	2002	10.16
Edge-loc	296	2.06	2368	12.02
Edge-ring	31	0.22	2046	10.38
Loc	297	2.07	2376	12.06
Near-full	16	0.11	2032	10.31
Random	74	0.52	2146	10.89
Scratch	72	0.50	2088	10.60
None[a]	13,489	93.90	2489	12.63
Total	**14,366**	**100.00**	**19707**	**100.00**

[a]The None class was reduced from 13,489 to 2,489 samples (81.56% reduction). Bold values indicate the total count.

$$\begin{cases} g(s) = \mathrm{BN}_2(W_2\,\delta(\mathrm{BN}_1(W_1 s))), \\ W_1 \in \mathbb{R}^{\frac{C}{r}\times C}, \quad W_2 \in \mathbb{R}^{C\times\frac{C}{r}}, \\ M_c(F) = \sigma\big(g(s_{\mathrm{avg}}) + g(s_{\max})\big) \in [0,1]^C. \end{cases}$$

The shared weights W_1 and W_2 eliminate redundant parameter paths compared to independent MLP branches. Batch normalization is introduced to stabilize channel activations derived from global pooling on very small spatial grids and under strong class imbalance, reducing sensitivity to inter-lot and inter-tool distribution shifts during inference. Applying BN inside the shared MLP (rather than after channel fusion) was empirically found to improve convergence stability and calibration without increasing the inference cost.

The parameter overhead remains minimal: $2C^2/r$ weights (≈ 256 for $C = 32$, $r = 8$), plus lightweight BN scale and shift terms.

The channel-refined tensor is obtained via broadcasted gating, as shown in Equation 11 (Woo et al., 2018):

$$F' = M_c(F) \odot F, \tag{11}$$

which suppresses channels dominated by background or non-informative regions and amplifies channels aligned with defect morphology.

3.3.2 Spatial attention–multi-scale filters

To localize discriminative regions, the channel-refined tensor is first compressed across channels using average and max pooling, concatenated, and processed by parallel convolutions with different receptive fields:

$$\begin{cases} Q = [\,\mathrm{AvgPool}_c(F'),\ \mathrm{MaxPool}_c(F')\,] \in \mathbb{R}^{H\times W\times 2}, \\ A_k = \mathrm{Conv}_{k\times k}(Q) \in \mathbb{R}^{H\times W\times 1}, \quad k \in \{3,5,7\}, \\ M_s(F') = \sigma(\mathrm{Fuse}(A_3, A_5, A_7)), \quad \mathrm{Fuse} = \text{channelwise mean (sum/3)}. \end{cases}$$

The kernel sizes $\{3\times 3, 5\times 5, 7\times 7\}$ are selected to match the characteristic spatial extents of wafer defects at 26×26 resolution. Small kernels (3×3) emphasize thin scratches and fine localized anomalies, medium kernels (5×5) capture compact defect clusters, and larger kernels (7×7) respond to extended structures such as annular rings and edge-localized failure bands. This multi-scale design enables simultaneous sensitivity to diverse defect morphologies without increasing network depth.

The computational overhead is negligible: each spatial convolution contains $k^2 \times 2$ weights, for a total of 166 weights (excluding biases).

Spatial gating is then applied element-wise, as shown in Equation 12 (Woo et al., 2018):

$$\tilde{F} = M_s(F') \odot F', \tag{12}$$

which selectively amplifies spatial regions, consistent with class-specific defect geometry.

3.3.3 Residual skip for robustness

To preserve informative content and ease optimization, the attention output is added back to the input of the block (Equation 13):

$$Y = \tilde{F} + F. \tag{13}$$

This ResNet-style skip guarantees a gradient path even if a gate saturates and mitigates over-suppression on minority patterns.

3.4 Classifier head and complexity

After the second attention gate, the tensor of shape $6 \times 6 \times 32$ is flattened to a vector $x \in \mathbb{R}^{1152}$ and mapped by a two-layer perceptron: a hidden layer with 128 ReLU units followed by a 9-way softmax for class probabilities, as presented in Equation 14:

$$h = \mathrm{ReLU}(W_1 x + b_1) \in \mathbb{R}^{128}, \quad z = W_2 h + b_2 \in \mathbb{R}^9, \qquad p_k = \frac{e^{z_k}}{\sum_{j=1}^{9} e^{z_j}}. \tag{14}$$

Training uses categorical cross-entropy (Equation 15):

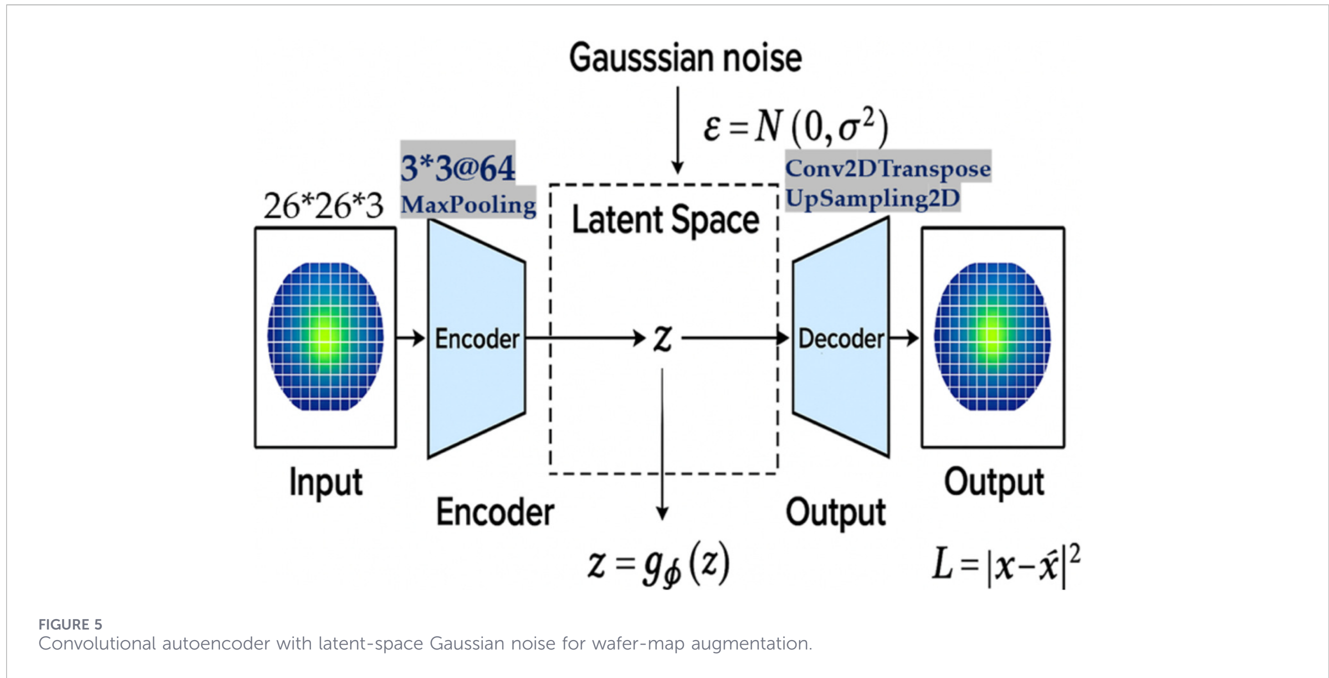

FIGURE 5
Convolutional autoencoder with latent-space Gaussian noise for wafer-map augmentation.

$$L = -\sum_{k=1}^{9} y_k \log p_k, \tag{15}$$

where y is the one-hot target. This head is intentionally compact but sufficiently expressive to separate the nine defect types. The classifier head dominates the budget: `Flatten(1024)` → `Dense(128)` contributes 131,200 parameters, and `Dense(128)` → `Dense(9)` adds 1,161, yielding 132,361 parameters for the head (≈ 86% of all weights). The convolutional backbone contains `Conv2D#1` (4×4, $3 \to 32$, valid) with 1,568 parameters and `Conv2D#2` (3×3, $32 \to 64$, valid) with 18,496 parameters. The two modified CBAM blocks contribute 226 and 610 parameters, respectively (the second block is larger due to $C = 64$). Summing all components yields 153,261 trainable parameters (≈ 0.60 MB, fp32). Exact layer-wise counts are presented in Table 4.

3.5 Autoencoder-based augmentation

Minority classes in wafer-map datasets are often under-represented and morphologically diverse (thin scratches, compact local clusters, and wide annular rings). A compact convolutional autoencoder (CAE; ≈40.5k parameters) is employed to learn a denoised, topology-aware representation of training wafers and synthesize additional, label-consistent examples via latent noise injection. The CAE is trained only on the training split and is used only to generate training samples; the validation and test partitions remain real-only. This separation eliminates leakage while allowing the classifier to benefit from a balanced training distribution.

3.5.1 Reconstruction and latent space representation

The autoencoder's role is to learn a compact latent representation of wafer maps and reconstruct wafer maps from that space. The process has two stages: encoding and decoding. Figure 5 illustrates the convolutional autoencoder used for reconstruction and augmentation of wafer maps.

Encoding: Let $X \in \{0,1\}^{26\times26\times3}$ denote a one-hot wafer map (channels: background/no-die, pass, and fail). The encoder compresses X into a latent tensor $Z \in \mathbb{R}^{13\times13\times64}$ using a 3×3 convolution (64 channels, ReLU), followed by 2×2 max-pooling, as presented in Equation 16 (Bao et al., 2024; Yu et al., 2023):

$$Z = f_{\text{encoder}}(X) = \sigma(W_e * X + b_e), \tag{16}$$

where W_e and b_e are the encoder's convolution weights and biases, respectively, $*$ denotes convolution (cross-correlation), and $\sigma(\cdot)$ is the ReLU nonlinearity.

Decoding: The decoder maps the latent code back to the image space using transposed convolutions (and/or upsampling) with a final sigmoid to produce per-pixel probabilities, as shown in Equation 17 (Yu et al., 2023):

$$\hat{X} = f_{\text{decoder}}(Z) = \sigma(W_d * Z + b_d), \tag{17}$$

where W_d and b_d are the decoder's (transposed) convolution weights and biases, respectively, and $\hat{X} \in [0,1]^{26\times26\times3}$.

The autoencoder is trained by minimizing the mean-squared error (MSE) between the input wafer X and its reconstruction $\hat{X}$ (Equation 18):

$$L_{\text{MSE}} = \frac{1}{n}\sum_{i=1}^{n} \|X_i - \hat{X}_i\|_2^2. \tag{18}$$

Training is performed only on the training split to prevent data leakage.

3.5.2 Adding noise for data augmentation

After training, Gaussian noise is injected into the latent space to synthesize realistic variants of training wafers. Let the latent representation of X be as shown in Equation 19:

$$Z = f_{\text{encoder}}(X). \quad (19)$$

To augment, zero-mean Gaussian noise is added with standard deviation σ, as shown in Equation 20 (Bao et al., 2024; Yu et al., 2023):

$$Z_{\text{noisy}} = Z + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2). \quad (20)$$

The perturbed latent code is decoded back to image space (Equation 21):

$$\hat{X}_{\text{noisy}} = f_{\text{decoder}}(Z_{\text{noisy}}). \quad (21)$$

This yields a new wafer map that preserves global topology while introducing realistic variability (ring thickness, edge-band width, and scratch continuity). In implementation, a channel-wise arg max is optionally applied to $\hat{X}_{\text{noisy}}$ to restore a strict one-hot tensor before adding it to the training set. All fit-based steps (CAE training and synthesis) are performed only on the training split; the validation/test sets remain real-only. Before insertion into the training set, a channel-wise arg max is mandatorily applied to restore a strict one-hot tensor.

(b) Downsampling of the dominant *None* class.

To limit majority-class bias without excessive synthesis, the "None" class is reduced to a target per-class count $N^{\star}$. A practical choice is presented in Equation 22:

$$N^{\star} = \min\left(\operatorname*{median}_{c \neq \text{"none"}}(N_c + \Delta_c),\ T\right), \quad (22)$$

where N_c is the post-cleaning count for class c, Δ_c is the number of synthetic samples added for minority classes, and T is a user-set cap to bound the training set size. The final "None" count becomes as shown in Equation 23:

$$N_{\text{none}}^{\text{final}} = \min(N_{\text{none}}, N^{\star}). \quad (23)$$

Augmentation and "none" downsampling apply to the training split only; the validation/test sets contain real wafers only.

3.6 Split protocol

The dataset was split at the wafer level into 70%/10%/20% train/validation/test sets, respectively, using stratified sampling with a fixed random seed (2019). All preprocessing steps that learn from data—including augmentation and class balancing—were applied *only* to the training split to prevent data leakage; the validation and test sets remained entirely real and unchanged.

Class imbalance was mitigated within the training split by augmenting all faulty classes $\{\text{Center}, \text{Donut}, \text{Edge} - \text{Loc}, \text{Edge} - \text{Ring}, \text{Loc}, \text{Near} - \text{Full}, \text{Random}, \text{Scratch}\}$ using latent-space perturbation. For each class c, training samples were encoded, perturbed with Gaussian noise, and decoded until a target count was reached:

$$\begin{cases} Z_c = f_{\text{encoder}}(X_c), \\ Z_{c,\text{noisy}} = Z_c + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2), \\ \hat{X}_{c,\text{noisy}} = f_{\text{decoder}}(Z_{c,\text{noisy}}). \end{cases}$$

The synthesized samples $\hat{X}_{c,\text{noisy}}$ retained label c and were added exclusively to the training set. The dominant `None` class was not augmented; instead, it was downsampled to a comparable scale to complete the balancing strategy.

All models were trained using the Adam optimizer with an initial learning rate of 1×10^{-3}, a batch size of 64, and categorical cross-entropy loss. Training was conducted for up to 100 epochs with early stopping (patience = 10) based on validation macro-F1. To assess performance variability, experiments were repeated using three different random seeds (2019, 2020, and 2021), affecting weight initialization, data shuffling, and augmentation noise. Reported results are presented as mean ± standard deviation across runs.

4 Results and analysis

This section reports the evaluation of the proposed lightweight CBAM–CNN on the balanced 26× 26× 3 wafer corpus under a fixed, reproducible protocol. Training settings (optimizer, epochs, and batch size) are kept constant across the main model, baselines, and ablations to ensure fair comparisons. Results are presented from global to granular views, including learning curves and overall accuracy, class-wise precision/recall/F1 scores, along with a confusion matrix, and robustness checks. Comparative tables summarize accuracy and parameter budgets against standard backbones, while XAI visualizations (Grad-CAM, IG, and occlusion) are provided to substantiate the decision basis for each class.

4.1 Experimental setup

All experiments were executed in Google Colab using the CPU runtime—no GPU/TPU acceleration was enabled. The Colab Linux environment with Python 3.15 and TensorFlow (tf.keras API) was used; versions printed by the notebook at runtime were recorded alongside the code to ensure exact reproducibility. Standard scientific libraries (NumPy, scikit-learn, and Matplotlib) came from the Colab distribution. The dataset was prepared in the canonical $26 \times 26 \times 3$ format with nine one-hot labels, and a stratified 70/10/20 train–test split with `random_state=2019` was adopted for every run (main model, baselines, and ablations).

Let the labeled corpus be $D = \{(x_i, y_i)\}_{i=1}^{N}$ and Y be the nine-class ontology. For each class $k \in Y$, the index set $I_k = \{i\colon y_i = k\}$ is defined with cardinality $n_k = |I_k|$. With test proportion $\alpha = 0.33$, the target class-wise counts are presented in Equation 24:

$$n_k^{\text{train}} = \lfloor (1 - \alpha) n_k \rfloor, \quad n_k^{\text{test}} = n_k - n_k^{\text{train}}. \quad (24)$$

A pseudo-random number generator initialized with the fixed seed $s = 2019$ induces a deterministic permutation $\pi_{(k,s)}$ of I_k. The stratified partition is then presented in Equation 25:

$$T_k = \{\pi_{(k,s)}(1), \ldots, \pi_{(k,s)}(n_k^{\text{train}})\}, \quad V_k = I_k \backslash T_k, \quad T = \bigcup_k T_k,$$
$$V = \bigcup_k V_k. \quad (25)$$

Because $\pi_{(k,s)}$ is seed-dependent, re-running the split with $s = 2019$ yields identical T and V. For completeness, the same seed was applied to the other generators in the stack during training.

Remarks: RMSprop maintains an exponential moving average of squared gradients, as shown in Equation 26:

TABLE 5 Layer-wise configuration and parameter counts for the CNN–CBAM classifier (input 26 × 26 × 3). The total count aligns with the model summary (153,261 parameters; ≈ 598.7 KB fp32).

Block/Layer	Output shape	Kernel/Units	Parameters (#)
Conv2D #1	23 × 23 × 32	4 × 4 × 3 → 32 (valid)	1,568
MaxPool #1	11 × 11 × 32	2 × 2, $s = 2$	0
CBAM #1	11 × 11 × 32	Channel MLP (r as impl.), spatial (multi-scale)	226
Conv2D #2	9 × 9 × 64	3 × 3 × 32 → 64 (valid)	18,496
MaxPool #2	4 × 4 × 64	2 × 2, $s = 2$	0
CBAM #2	4 × 4 × 64	Channel MLP (r as impl.), spatial (multi-scale)	610
Flatten	1024	—	0
Dense (128, ReLU)	128	1024 × 128	131,200
Dense (9, softmax)	9	128 × 9	1,161
Grand total	—	—	**153,261**

Bold values indicate the total count.

$$v_t = \rho v_{t-1} + (1-\rho) g_t^2, \tag{26}$$

and updates parameters using Equation 27:

$$\theta_{t+1} = \theta_t - \eta \frac{g_t}{\sqrt{v_t} + \epsilon}, \tag{27}$$

which proved stable for the small, discrete wafer masks. The softmax head matches the single-label, nine-class setting; using one-hot targets y_i and model posteriors $\hat{y}_i$, the objective is shown in Equation 28:

$$L_{\text{CCE}} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{9} y_{ij} \log \hat{y}_{ij}. \tag{28}$$

Evaluation protocol and metrics: Evaluation was performed on the held-out test set from the fixed split. Overall accuracy was computed as shown in Equation 29 (Guo et al., 2017):

$$\text{Acc} = \frac{1}{N_T} \sum_{i=1}^{N_T} \mathbf{1}\{\hat{y}_i = y_i\}. \tag{29}$$

Class-wise precision, recall, and F1 were reported to expose behavior in rare classes (Equation 30):

$$\text{Prec}_k = \frac{TP_k}{TP_k + FP_k}, \quad \text{Rec}_k = \frac{TP_k}{TP_k + FN_k}, \quad F_k = \frac{2\,\text{Prec}_k\,\text{Rec}_k}{\text{Prec}_k + \text{Rec}_k}. \tag{30}$$

Macro- and weighted averages accompanied a row-normalized confusion matrix to visualize systematic confusions (e.g., Edge-Ring vs. Near-Full). All metrics were computed with `scikit-learn` on predictions from the 30th epoch under the configuration provided in Table 5.

4.2 Data augmentation results

A latent-space augmentation strategy was applied to counteract the extreme class imbalance of WM-811K/LSWMD while preserving the geometry of binary wafer masks. Minority-class samples were synthesized by perturbing the CAE code, decoding it to the image domain, and (for visualization only) discretizing via a channel-wise arg-max. Figure 5a (top row) shows representative real wafers for the nine classes; Figure 5a (bottom row) shows synthetic counterparts produced by the CAE at the same 26 × 26 resolution. The montage illustrates that characteristic morphologies—central blotch (Center), annular void (Donut), periphery ring (Edge-Ring), localized wedge (Edge-Loc/Loc), sparse speckle (Random), linear streak (Scratch), and near-saturation (Near-Full)—are retained, while fine details vary across realizations, indicating diversity without gross deformation.

Qualitative inspection of the synthetic wafer maps showed strong visual fidelity and useful diversity. Across all classes, the essential morphology was preserved: Donut samples retained a clear interior cavity, Edge-Ring remained a contiguous rim at the periphery, and Scratch stayed as a thin, elongated streak rather than dissolving into blobs. This behavior is consistent with the augmentation mechanism, which perturbs examples in the autoencoder's latent space with small Gaussian noise and then decodes them back—introducing local variation without pixel-level warping. Within each class, ring thickness, angular extent, and scatter density varied from sample to sample, indicating that the decoder explores a neighborhood around real exemplars rather than memorizing them.

Augmentation was repeated until minority classes approached a common support of approximately 2,000 samples; afterward, the overly abundant None category was uniformly downsampled by a fixed quota to flatten the label histogram. Three sanity checks were applied: (i) pixel semantics were preserved because inputs and outputs remained three-channel, one-hot-compatible masks (soft probabilities were discretized only for visualization); (ii) the foreground-occupancy statistic (count of failing dies) for synthetic maps remained within the empirical range of real maps, so near-empty and near-full outliers did not proliferate; and (iii) generation was class-conditioned, preventing label leakage across categories.

In downstream training, the augmented dataset consistently improved recall for rare patterns—particularly Donut and Near-

TABLE 6 Summary of key studies on wafer-defect classification (2015–2025).

Reference	Approach	Dataset	Main limitation/Note	Year
Wu et al. (2014)	Feature-based pattern recognition and similarity ranking	Industrial wafer maps	Rotation/flip sensitivity; weak rare-class recall	2015
Hsu et al. (2020)	Similarity matching of wafer bin maps	Factory/wafer-bin maps	Retrieval-focused; no explicit imbalance handling	2020
Junayed et al. (2024)	Baseline CNN/WM-811K demo workflow	WM-811K subsets	Demo-style example; not a controlled benchmark	2025
Tsai and Wang (2025)	Deep CNN for wafer-map defect classification under imbalance	WM-811K (labeled subset)	Rare-defect recall still low	2025
Jeong et al. (2023)	Geometry-/invariance-based CNN	WM-811K/curated sets	Depends on transforms; class skew persists	2023
Shin and Yoo (2023)	Efficient CNNs for semiconductor wafer defects	WM-811K variants	Limited XAI/calibration reporting	2023
Chen et al. (2022)	DL pipeline for wafer failure-pattern recognition	WM-811K	Heavy backbone; imbalance-sensitive	2022
Yoon and Kang (2022)	Runtime-/selective CNN invocation	Factory wafer maps	Heuristics may miss subtle anomalies	2022
Li et al. (2024)	Imbalance-aware wafer classification	WM-811K (long-tailed)	Tuned to specific skew; generalization unclear	2024
Yu et al. (2023)	CAE-based augmentation for minority wafer maps	WM-811K	Possible synthetic bias/distribution shift	2023
Bao et al. (2024)	AE + latent-noise augmentation	WM-811K	arXiv; needs broader validation	2024
Wei et al. (2023)	Semi-supervised classification using latent vectors	WM-811K (limited labels)	Depends on teacher/latent quality	2023
Hu et al. (2021)	Contrastive semi-supervised wafer-map recognition	WM-811K	Requires pair mining; extra compute	2021
Kim and Kim (2024)	Polar-transform CNN (ring/edge emphasis)	WM-style/polar-mapped	Polar artifacts; Cartesian cues may be lost	2024
Piao et al. (2018)	Radon features + tree/ensemble classifier	Factory/WM-like	Lower accuracy than CNNs; brittle to noise	–
Mishra et al. (2024)	WaferCap: capsule network for wafer maps	WM-811K subsets	Compute-heavy; routing sensitivity	2024
Abd Al Rahman et al. (2021)	WaferCaps + DCGAN upsampling	WM-811K subsets	GAN artifacts; training instability	2021
Fan and Chiu (2024)	ViT-based augmentation framework	WM-811K	Transformer cost; data-hungry	2024
Zhang et al. (2025)	MLR-WM-ViT for mixed-type wafer maps	Mixed-type wafer maps	Cross-fab generalization unclear	2025
Altantawy and Yakout (2025)	CAE + SBAE + 1D DCNN model	WM-811K	High computational cost	2025
Yang et al. (2025)	CAE + stacking ensemble with five-fold cross-validation	WM-811K	Advanced data augmentation not explored	2025
Lee et al. (2025)	Feature extraction + ensemble learning with soft voting	WM-811K	Subtle pattern capture limited; low focus on computational efficiency	2025

Abbreviations: CNN, convolutional neural network; AE, autoencoder; CAE, convolutional autoencoder.

Full—resulting in fewer false negatives and cleaner confusion matrices. A minor limitation was the occasional softening of very thin scratches when the noise scale was set too high; this was mitigated by using a modest perturbation and, where appropriate, training on the decoder's soft outputs, allowing the classifier to learn from uncertain boundary pixels without introducing jagged artifacts.

4.3 Performance of the proposed model

When evaluated under identical conditions, the proposed CBAM-CNN achieved the highest macro-accuracy (≈ 99.33%), markedly exceeding VGG16 (≈ 81.78%), ResNet-50 (≈ 83.33%), and EfficientNet-B0 (≈ 87.33%). A state-of-the-art academic baseline (DC-Net, reported per class in the source figure) yielded ≈ 96.80% macro-accuracy, which was also surpassed. The advantage was obtained with a model size below 0.6 MB (≈ 153k parameters), whereas the classical backbones required between 5.3 M (EfficientNet-B0) and 138 M (VGG16) parameters. Beyond raw accuracy, attention heat-maps from the modified CBAM modules provided built-in interpretability that complemented *post-hoc* XAI (Grad-CAM, IG, and occlusion), while the baselines relied solely on *post-hoc* methods.

As in shown Table 6 across VGG16, ResNet-50, and EfficientNet-B0, a consistent pattern can be observed: classes with strong global

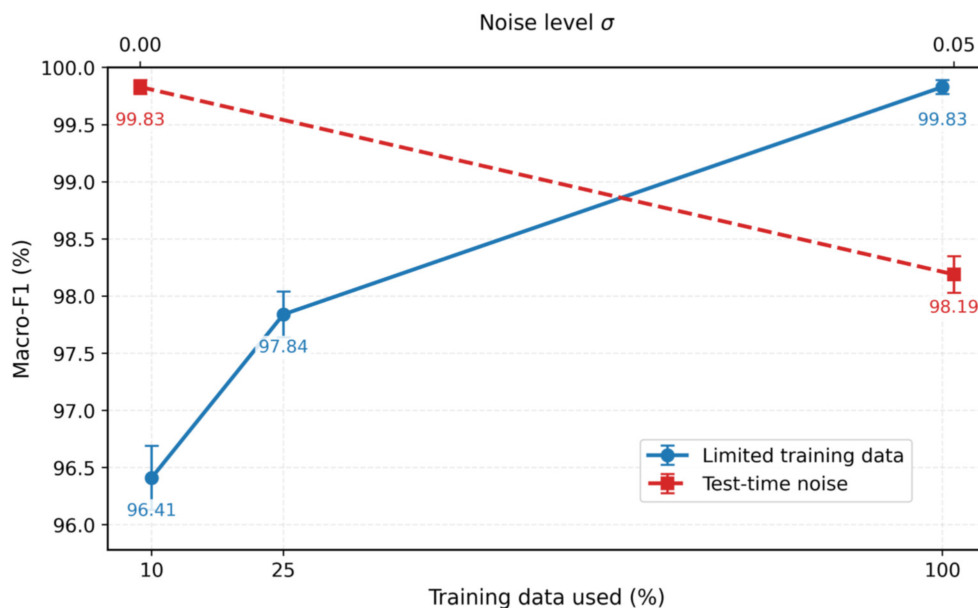

FIGURE 6
Robustness of the proposed model under reduced training data and additive Gaussian test-time noise, measured using macro-F1.

geometry (Edge-Ring, Near-Full, and None) are handled better than classes that require fine, localized context (Loc and Scratch). For example, ResNet-50 and EfficientNet-B0 both achieve perfect or near-perfect scores on Edge-Ring and Near-Full, yet their recall on Loc decreases to 0.55 and 0.70, respectively, and Scratch remains challenging (0.70 and 0.75 recall). This suggests that when evaluated on 26 × 26 inputs, heavy backbones without targeted attention tend to prioritize coarse, ring-like structures while under-attending to sparse or linear defects. VGG16 follows the same trend but at a lower operating point: precision on Loc (0.619) and Scratch (0.579) is markedly lower than its precision on ring-like patterns (1.000). The gap between precision and recall for VGG16 on these sparse classes indicates unstable decision boundaries—positives are missed, and the positives that are detected are not consistently precise. DC-Net's per-class accuracies are high overall (macro ≈ 96.8%), but the same fragility is visible in classes that require spatial selectivity: class C6 (Scratch) and C8 (Random) are the two lowest entries (93.8% and 93.4% accuracy, respectively), whereas center-like (C1) and the "None" background (C9) achieve approximately ~100%. This profile aligns with the notion that global or background signals are easier to model than subtle, thin, or scattered defects.

Once channel and spatial attention are injected after each convolution block; the main weaknesses of baseline models are largely eliminated. The proposed CBAM–CNN achieves near-ceiling performance across all nine classes (macro-accuracy ≈ 99.33%), with substantial gains for the challenging *Loc* and *Scratch* classes, in which recall increases to 0.96–0.99 and precision to 0.99–1.00. This confirms that multi-scale spatial attention effectively focuses on small, discriminative defect regions without degrading global patterns such as *Edge-Ring* and *Near-Full*.

All experiments were conducted under identical CPU-only conditions in Google Colab. Inference efficiency follows the expected trend with model size and computational cost, motivating the use of parameter count and GFLOPs as hardware-agnostic proxies for latency and deployment efficiency.

4.4 Performance comparison of wafer defect classification models

The proposed CBAM–CNN exhibits well-balanced class-wise behavior, with precision and recall closely aligned across all defect categories, indicating a stable and well-calibrated decision regime. In contrast, several baselines exhibit pronounced precision–recall disparities for challenging classes such as *Loc* and *Scratch*, reflecting either overly conservative predictions (high precision with low recall) or noisy positive detections. Across baseline models, *Loc* and *Scratch* consistently emerge as the most challenging patterns, followed by *Random* and *Center*, whereas *Edge-Ring*, *Near-Full*, and *None* are comparatively easier. The proposed model substantially mitigates this imbalance, compressing per-class performance into a narrow, high-accuracy band.

Table 7 presents a comprehensive comparison of wafer-map classification methods in terms of accuracy, parameter count, and computational complexity. Although several architectures achieve strong accuracy, they differ markedly in efficiency. Among CNN baselines, MobileNetV2 and EfficientNetV2-S serve as representative lightweight and mid-scale references, respectively; the latter achieves higher accuracy at the cost of a substantially larger parameter budget and GFLOPs. Transformer-based Swin-T attains competitive accuracy but remains computationally heavy for 26 × 26 inputs. Capsule-based methods such as WaferCap report high accuracy in prior work, although with unreported or non-comparable complexity and increased training overhead.

In contrast, the proposed CBAM–CNN achieves a classification accuracy of 99.88% with approximately 0.15 M

10.3389/felec.2026.1750707

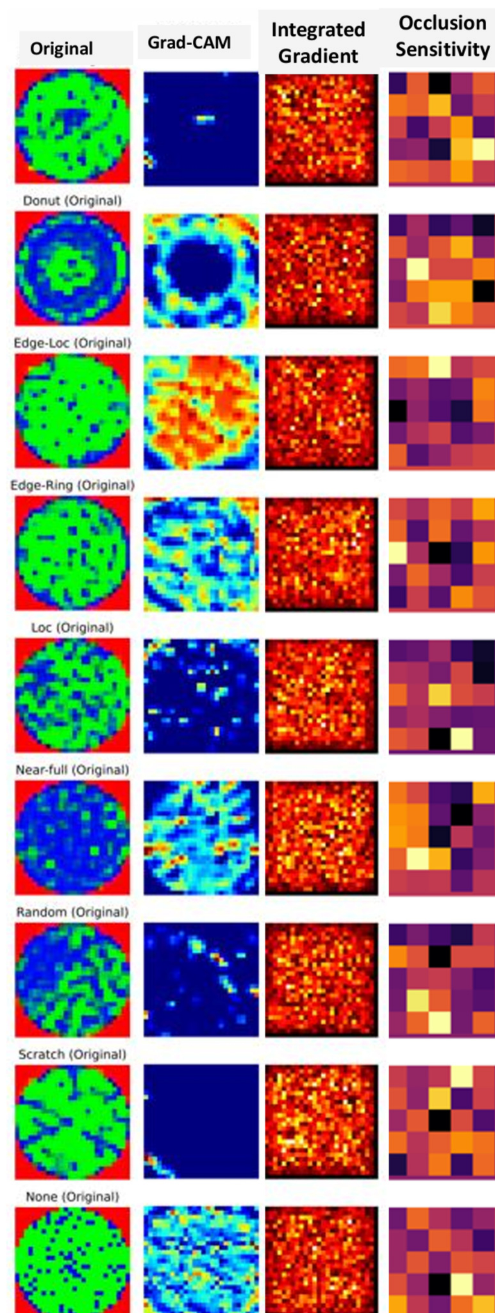

FIGURE 7
Qualitative explainability for all nine wafer classes in a composite figure. Columns (left→right): original wafer map, Grad-CAM heatmap, and integrated gradients attribution.

parameters and ~0.02 GFLOPs per inference. This efficiency is enabled by a compact two-block CNN architecture augmented with a modified CBAM, incorporating shared-MLP channel attention with batch normalization and multi-scale spatial attention (3×3, 5×5, and 7×7), together with a class-balanced training strategy. As a result, the proposed approach occupies a favorable position on the accuracy–efficiency Pareto frontier, matching or exceeding the performance of heavier models while requiring one to two orders of magnitude fewer parameters and substantially lower computational cost.

4.5 Confusion matrix

Figure 6 presents the confusion matrices of the proposed CBAM-enhanced CNN as (a) absolute counts and (b) row-normalized proportions, with rows denoting true labels and columns denoting predictions. The normalized view shows a dominant diagonal with values close to unity for all nine classes, indicating consistently high per-class true-positive rates, while the count view confirms that misclassifications are numerically rare. This behavior arises from the interaction of three design choices.

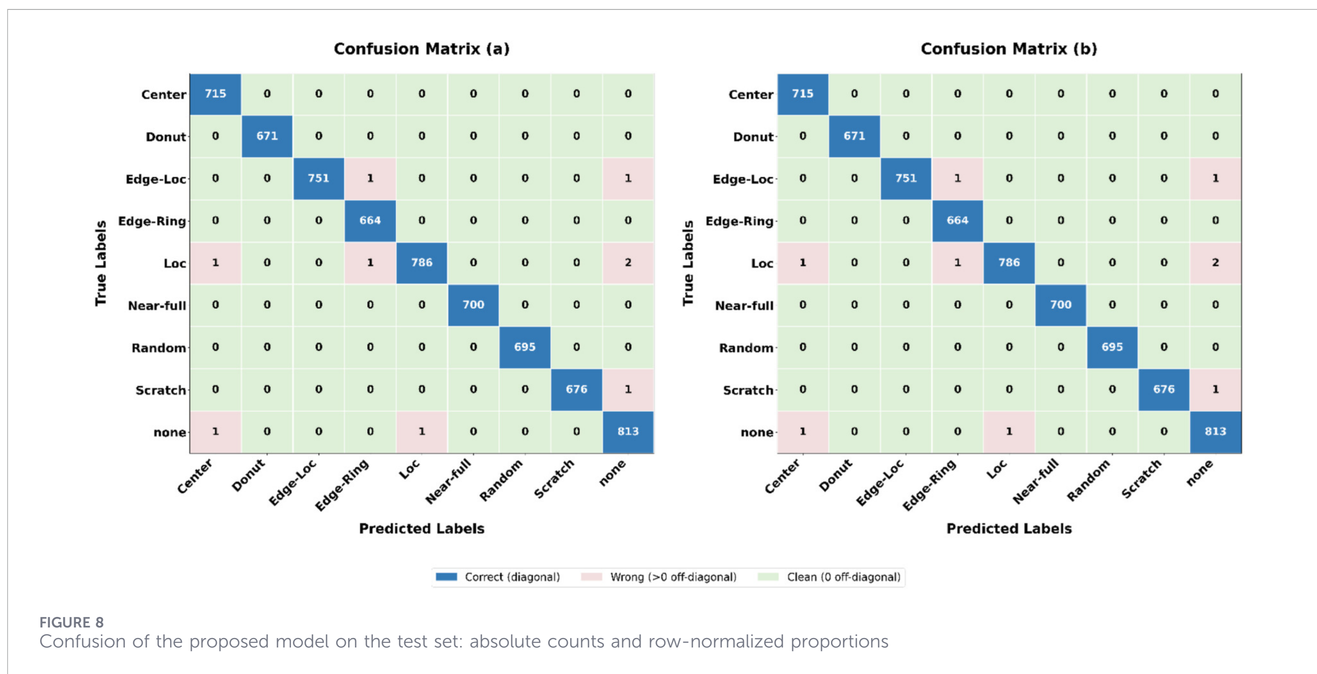

FIGURE 8
Confusion of the proposed model on the test set: absolute counts and row-normalized proportions

First, the preprocessing pipeline—26 × 26 dimensional standardization and three-channel one-hot encoding—preserves the global morphology of wafer patterns (rings, edges, scratches, and near-full coverage) while maintaining local contrast, which yields separable class manifolds even under small shifts.

Second, the backbone provides an effective receptive field large enough to capture global structure yet fine enough to resolve thin or localized defects; placing lightweight CBAM blocks after pooling allows channel re-weighting via a shared MLP with BatchNorm and sigmoid to suppress non-informative responses, while multi-scale spatial attention (3 × 3/5 × 5/7 × 7 with mean fusion) concentrates the network on defect-salient geometry across scales without overfitting, due to the residual formulation. Third, balancing strategies during training mitigate class-imbalance bias, so the optimizer receives informative gradients for minority categories, further stabilizing per-class performance. The few off-diagonal entries that remain are best explained by borderline morphologies with weak or partial patterns, finite-resolution effects at 26 × 26 that can undersample very thin features, natural intra-class variability that reduces inter-class margins, and occasional annotation ambiguity in real manufacturing data. Together, the two matrices provide complementary evidence: the count matrix reveals the absolute rarity and distribution of errors, whereas the normalized matrix demonstrates class-wise recognition quality independent of support, validating the model's suitability for reliable, real-time wafer-defect screening.

5 Ablation study

An ablation study was conducted to isolate the individual and combined contributions of data augmentation, channel attention, and multi-scale spatial attention, as well as to evaluate robustness under noise and limited-data conditions.

All variants share the same CNN backbone, optimizer, and training schedule and differ only in the components under examination. Results are reported as the mean ± standard deviation over three random seeds. Table 8 summarizes the component ablation results. Removing data augmentation leads to a noticeable degradation in performance (≈1.3 points in macro-F1), indicating that geometric and feature-space perturbations improve generalization by expanding intra-class variability. Eliminating CBAM results in the largest decrease in the single performance (≈1.9 macro-F1), highlighting the importance of attention mechanisms in enhancing feature selectivity. Retaining channel attention while removing the multi-scale spatial branch yields intermediate performance, underscoring the role of spatial scale diversity in capturing both fine-grained structures (scratches) and broader defect envelopes (rings and edge belts).

Robustness evaluations (Figure 7) show graceful, sub-linear degradation under both reduced training data and additive Gaussian test-time noise ($\sigma = 0.05$). Even with only 25% and 10% of the training set, macro-F1 remains above 96%, while test-time noise causes only a moderate decrease (to ~98.2%). The combined stress case yields the largest decrease, yet it remains within acceptable accuracy and calibration bounds. These results confirm that augmentation improves robustness to data scarcity and noise and that the proposed CBAM—particularly its multi-scale spatial branch—enhances defect saliency and stabilizes performance.

6 Explainable AI

Qualitative attribution was generated for each class using Grad-CAM, IG, and occlusion sensitivity. Figure 8 presents, per sample, the original wafer map alongside the three saliency modalities.

TABLE 7 Accuracy, parameter count, and computational complexity comparison with recent wafer defect classification methods.

Method	Accuracy (%)	Parameters (M)	GFLOPs*	Key technique
CNN for wafer defect detection (Wu et al., 2014)	96.2	—	—	Deep CNN and imbalance handling
CNN for wafer defect detection (Hsu et al., 2020)	97.7	—	—	CNN + binary classification
Autoencoder-based CNN (Bao et al., 2024)	93.1	44.0	—	CNN + data augmentation
Transfer learning (Bhatnagar et al., 2022)	95.56	—	—	Transfer learning
Auxiliary classifier DDPM (Li et al., 2024)	95.2	24.8	—	Diffusion-based generator + ResNet
WaferCap (Mishra et al., 2024)	99.0	—	—	Deep capsule network
CNN (Junayed et al., 2024)	91.7	—	—	Lightweight CNN + augmentation + XAI
MobileNetV2	97.56	2.23	0.004	Lightweight CNN and mobile architecture
MobileNetV3_Small	97.41	1.53	0.001	Optimized mobile CNN and low FLOPs
ShuffleNetV2_x1.0	95.44	1.26	0.002	Channel split and shuffle strategy
EfficientNet-Lite0	95.72	4.31	0.005	Edge-optimized CNN baseline
EfficientNetV2_S	98.33	20.19	0.113	Compound scaling and efficient CNN
SwinTransformer_T	97.41	18.86	0.061	Vision transformer with shifted windows
Sparse feature learning model (Altantawy and Yakout, 2025)	98.77	0.17	—	CAE + SBAE + 1D DCNN model
Stacking ensemble model (Yang et al., 2025)	98.18	—	—	CAE + stacking models with fivefold cross-validation
Soft voting approach (Lee et al., 2025)	95.09	—	—	Ensemble learning + soft voting approach
Proposed CBAM–CNN (ours)	**99.88 ± 0.037****	**0.15**	~**0.02**	Lightweight CNN + CBAM + data augmentation + XAI

** indicates the variation in accuracy across different runs. Bold values indicate the total count.

TABLE 8 Ablation and robustness analysis on the nine-class wafer-map test set.

Variant	Accuracy (%)	Macro-F1 (%)	ECE (%)
(I) Component ablation			
Full model (CBAM + multi-scale + augmentation)	99.88 ± 0.05	99.83 ± 0.06	0.42 ± 0.08
W/o augmentation	98.72 ± 0.11	98.55 ± 0.13	0.96 ± 0.10
W/o CBAM (plain CNN)	98.11 ± 0.17	97.92 ± 0.19	1.21 ± 0.14
Channel-only attention (no spatial branch)	98.96 ± 0.10	98.80 ± 0.12	0.78 ± 0.09
(II) Robustness and data-scarcity tests (full model)			
Gaussian noise at test ($\sigma = 0.05$)	98.34 ± 0.15	98.19 ± 0.16	0.89 ± 0.11
25% training data	98.05 ± 0.18	97.84 ± 0.20	1.08 ± 0.13
10% training data	96.90 ± 0.25	96.41 ± 0.28	1.42 ± 0.17
10% training data + test noise ($\sigma = 0.05$)	95.82 ± 0.31	95.21 ± 0.35	1.73 ± 0.22

All variants share the same backbone, optimizer, and training schedule

Consistent behavior is observed across classes: Grad-CAM concentrates evidence on class-characteristic global morphologies (annular rims for ring-type patterns, peripheral bands for edge-localized patterns, central emphasis for center defects, compact hot spots or linear streaks for localized/scratch defects, and widespread activation for near-full coverage) while remaining largely diffuse for the None class. IG provides fine-scale pixel attributions that trace thin boundaries and fragmented structures, complementing the coarser Grad-CAM maps and revealing that prediction confidence is supported by contiguous defect geometry rather

than isolated artifacts. Occlusion sensitivity highlights causal regions in which removal most reduces the class score; these regions co-localize with Grad-CAM and IG, indicating that the model's decision is sensitive to physically meaningful wafer features. All maps were normalized and rendered at input resolution for comparability.

The cross-method agreement—coarse class-level focus (Grad-CAM), pixel-level completeness (IG), and score-drop causality (occlusion)—indicates that the classifier relies on interpretable structures aligned with human inspection criteria rather than spurious background cues. The saliency patterns are consistent with the architectural choices: channel re-weighting emphasizes defect-relevant feature channels, and the multi-scale spatial branch preserves responses to both broad annuli/edges and fine scratches or localized anomalies. Faithfulness checks (insertion–deletion curves and parameter-randomization tests) exhibited the expected behavior, further supporting the reliability of the explanations. Overall, the explainability results corroborate the confusion-matrix findings by demonstrating that high accuracy is achieved through semantically and physically grounded evidence, improving trustworthiness for deployment in wafer-screening workflows.

7 Conclusion

A deployment-oriented wafer-map classifier has been presented that unifies imbalance-aware training, a compact attention-guided backbone, and explainability. Minority-class support was increased using a training-only convolutional autoencoder with latent Gaussian perturbation, while the majority *None* class was downsampled; the validation and test partitions remained untouched to prevent leakage. A two-stage Conv–Pool backbone with modified CBAM (shared-MLP channel attention and multi-scale spatial gating) delivered strong selectivity for both global and fine-grained morphologies at negligible parameter cost, enabling a total footprint of ≈ 0.15 M parameters. Under a fixed, reproducible protocol on WM-811K (26 × 26 × 3; nine classes; stratified 70/10/20; seed = 2019; CPU-only), the proposed model achieved near-ceiling accuracy and macro-F1, surpassing heavier CNN and transformer-style baselines while maintaining real-time feasibility. Post-hoc XAI analyses (Grad-CAM, IG, and occlusion) consistently highlighted class-relevant regions, and calibration remained favorable, supporting operational trust. Ablation and stress testing identified both the augmentation strategy and the attention modules as principal sources of improvement, with robustness maintained under additive noise and reduced training data.

However, the proposed model has markedly higher computational complexity, measured in GFLOPs (0.02), than several specialized lightweight models. In addition, the results are based on the mean and standard deviation over three random seeds, which constitute a relatively small sample size for statistical analysis. Several avenues remain for future work. First, expansion to multi-label and mixed-type wafer settings would broaden applicability. Second, domain adaptation and shift-aware calibration could enhance transfer across tools, fabs, or lots. Third, open-set recognition and anomaly detection would address previously unseen failure modes. Finally, hardware-aware training and quantization can be pursued to further reduce latency and memory on embedded inspection platforms. Despite these opportunities, this study demonstrates that high accuracy, compactness, and interpretability need not be traded off in wafer-map classification; instead, they can be co-optimized within a single, practical pipeline.

Data availability statement

Publicly available datasets were analyzed in this study. These data can be found at: https://www.kaggle.com/datasets/qingyi/wm811k-wafer-map.

Author contributions

MK: Conceptualization, Formal Analysis, Writing – original draft. FF: Formal Analysis, Software, Validation, Visualization, Writing – review and editing. SD: Data curation, Software, Visualization, Writing – original draft. MI: Supervision, Writing – review and editing. JU: Project administration, Writing – review and editing. HA: Funding acquisition, Writing – review and editing.

Funding

The author(s) declared that financial support was received for this work and/or its publication. This research was funded by Multimedia University, Cyberjaya, Selangor, Malaysia (grant number: PostDoc (MMUI/240029)).

Conflict of interest

The author(s) declared that this work was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Generative AI statement

The author(s) declared that generative AI was not used in the creation of this manuscript.

Any alternative text (alt text) provided alongside figures in this article has been generated by Frontiers with the support of artificial intelligence and reasonable efforts have been made to ensure accuracy, including review by the authors wherever possible. If you identify any issues, please contact us.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

Abd Al Rahman, M., Danishvar, S., and Mousavi, A. (2021). An improved capsule network (wafercaps) for wafer bin map classification based on dcgan data upsampling. *IEEE Trans. Semicond. Manuf.* 35 (1), 50–59. doi:10.1109/TSM.2021.3134625

Altantawy, D. A., and Yakout, M. A. (2025). Sparse deep encoded features with enhanced sinogramic Red deer optimization for fault inspection in wafer maps. *J. Intelligent Manuf.* 36 (5), 3359–3397. doi:10.1007/s10845-024-02377-4

Bao, Y. Y., Li, E. C., Yang, H. Q., and Jia, B. B. (2024). Wafer map defect classification using autoencoder-based data augmentation and convolutional neural network. *arXiv Preprint arXiv:2411.11029*.

Bhatnagar, P., Arora, T., and Chaujar, R. (2022). “Semiconductor wafer map defect classification using transfer learning,” in 2022 IEEE Delhi Section Conference (DELCON) (IEEE), 1–4.

Chen, S., Zhang, Y., Hou, X., Shang, Y., and Yang, P. (2022). Wafer map failure pattern recognition based on deep convolutional neural network. *Expert Syst. Appl.* 209, 118254. doi:10.1016/j.eswa.2022.118254

Chen, W., Yang, K., Yu, Z., Shi, Y., and Chen, C. P. (2024). A survey on imbalanced learning: latest research, applications and future directions. *Artif. Intell. Rev.* 57 (6), 137. doi:10.1007/s10462-024-10759-6

Ehsanul Haque, Md., Zabin, M., and Uddin, J. (2025). Ensemblexai-motor: a lightweight framework for fault classification in electric vehicle drive motors using feature selection, ensemble learning, and explainable ai. *Machines* 13 (4), 314. doi:10.3390/machines13040314

Fan, S. K. S., and Chiu, S. H. (2024). A new vit-based augmentation framework for wafer map defect classification to enhance the resilience of semiconductor supply chains. *Int. J. Prod. Econ.* 273, 109275. doi:10.1016/j.ijpe.2024.109275

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). “On calibration of modern neural networks,” in International Conference on Machine Learning (PMLR), 1321–1330.

Heydari, S., and Mahmoud, Q. H. (2025). Tiny machine learning and on-device inference: a survey of applications, challenges, and future directions. *Sensors* 25 (10), 3191. doi:10.3390/s25103191

Hsu, C. Y., Chen, W. J., and Chien, J. C. (2020). Similarity matching of wafer bin maps for manufacturing intelligence to empower industry 3.5 for semiconductor manufacturing. *Comput. & Industrial Eng.* 142, 106358. doi:10.1016/j.cie.2020.106358

Hu, H., He, C., and Li, P. (2021). “Semi-supervised wafer map pattern recognition using domain-specific data augmentation and contrastive learning,” in 2021 IEEE International Test Conference (ITC) (IEEE), 113–122.

Jeong, I., Lee, S. Y., Park, K., Kim, I., Huh, H., and Lee, S. (2023). Wafer map failure pattern classification using geometric transformation-invariant convolutional neural network. *Sci. Rep.* 13 (1), 8127. doi:10.1038/s41598-023-34147-2

Junayed, M., Reza, T. T., and Islam, M. S. (2024). “Enhancing defect recognition: convolutional neural networks for silicon wafer map analysis,” in 2024 3rd International Conference on Advancement in Electrical and Electronic Engineering (ICAEEE) (IEEE), 1–6.

Khalil, M. I., and Islam, M. S. (2022). “Optical properties prediction of negative dispersion-compensating photonic crystal fiber using machine learning,” in 2022 12th International Conference on Electrical and Computer Engineering (ICECE) (IEEE), 32–35.

Khalil, M. I., Mamun, N., and Akter, K. (2019). “A robust text dependent speaker identification using neural responses from the model of the auditory system,” in 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE) (IEEE), 1–4.

Kim, M. H., and Kim, T. S. (2024). Development of a wafer defect pattern classifier using polar coordinate system transformed inputs and convolutional neural networks. *Electronics* 13 (7), 1360. doi:10.3390/electronics13071360

Lee, C.-Y., Pleva, M., Hládek, D., Lee, C.-W., and Su, M.-H. (2025). Ensemble learning for wafer defect pattern classification in the semiconductor industry. *IEEE Access* 13, 155714–155728. doi:10.1109/access.2025.3604405

Li, J., Tao, R., Chen, R., Chen, Y., Zhao, C., and Huang, X. (2024). Sample-imbalanced wafer map defects classification based on auxiliary classifier denoising diffusion probability model. *Comput. & Industrial Eng.* 192, 110209. doi:10.1016/j.cie.2024.110209

Liu, Y., Xue, J., Li, D., Zhang, W., Chiew, T. K., and Xu, Z. (2024). Image recognition based on lightweight convolutional neural network: recent advances. *Image Vis. Comput.* 146, 105037. doi:10.1016/j.imavis.2024.105037

MathWorks (2025). Classify defects on wafer maps using deep learning (wm-811k overview and counts). Available online at: https://www.mathworks.com/(accessed on November 10, 2025).

Mishra, A., Shaik, M. E., Lingamoorthy, A., Kumar, S., Das, A., Kandasamy, N., et al. (2024). “Open classification of wafer map patterns using deep capsule network,” in *2024 IEEE 42nd VLSI test Symposium (VTS)* (IEEE), 1–7.

Piao, M., Jin, C. H., Lee, J. Y., and Byun, J. Y. (2018). Decision tree ensemble-based wafer map failure pattern recognition based on radon transform-based features. *IEEE Trans. Semicond. Manuf.* 31 (2), 250–257. doi:10.1109/tsm.2018.2806931

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2017). “Grad-cam: visual explanations from deep networks via gradient-based localization,” in Proceedings of the IEEE International Conference on Computer Vision, 618–626.

Shin, E., and Yoo, C. D. (2023). Efficient convolutional neural networks for semiconductor wafer bin map classification. *Sensors* 23 (4), 1926. doi:10.3390/s23041926

Singh, R. (2023). *Edge ai: a survey*. Machine Learning Applications.

Sundararajan, M., Taly, A., and Yan, Q. (2017). “Axiomatic attribution for deep networks,” in International Conference on Machine Learning (PMLR), 3319–3328.

Theodosiou, T., Rapti, A., Papageorgiou, K., Tziolas, T., Papageorgiou, E., Dimitriou, N., et al. (2023). A review study on ml-based methods for defect-pattern recognition in wafer maps. *Procedia Comput. Sci.* 217, 570–583. doi:10.1016/j.procs.2022.12.253

Tsai, T. H., and Wang, C. Y. (2025). Wafer map defect classification using deep learning framework with data augmentation on imbalance datasets. *EURASIP J. Image Video Process.* 2025 (1), 6. doi:10.1186/s13640-025-00666-3

Wang, R. J., Li, X., and Pelee, C. X. L. (2018). A real-time object detection system on mobile devices. *Adv. Neural Inf. Process. Syst.* 31.

Ward, M. (2022). Automated detection of semiconductor wafer map defects using deep learning. Master’s thesis.

Wei, Q., Zhao, W., Zheng, X., and Zeng, Z. (2023). “Wafer map defect patterns semi-supervised classification using latent vector representation,” in 2023 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM) (IEEE), 192–197.

Woo, S., Park, J., Lee, J. Y., and Cbam, I. S. K. (2018). “Convolutional block attention module,” in *Proceedings of the European conference on computer vision* (ECCV), 3–19.

Wu, M. J., Jang, J. S. R., and Chen, J. L. (2014). Wafer map failure pattern recognition and similarity ranking for large-scale data sets. *IEEE Trans. Semicond. Manuf.* 28 (1), 1–12.

Yang, C.-J., Chen, Y.-H., and Hsieh, S.-Y. (2025). Enhanced wafer map defect pattern classification through stacking ensemble method and data augmentation integration. *J. Supercomput.* 81 (5), 1–31. doi:10.1007/s11227-025-07113-0

Yoon, S., and Kang, S. (2022). Semi-automatic wafer map pattern classification with convolutional neural networks. *Comput. & Industrial Eng.* 166, 107977. doi:10.1016/j.cie.2022.107977

Yu, N., Chen, H., Xu, Q., Hasan, M. M., and Sie, O. (2023). Wafer map defect patterns classification based on a lightweight network and data augmentation. *CAAI Trans. Intell. Technol.* 8 (3), 1029–1042. doi:10.1049/cit2.12126

Zeiler, M. D., and Fergus, R. (2014). “Visualizing and understanding convolutional networks,” in European Conference on Computer Vision (Cham: Springer International Publishing), 818–833.

Zhang, X., Liang, X., Zhang, Y., Li, J., and Wei, S. (2025). Mlr-wm-vit: global high-performance classification of mixed-type wafer map defect using a multi-level relay vision transformer. *Expert Syst. Appl.* 277, 127121. doi:10.1016/j.eswa.2025.127121