IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

EagleEye: Industrial Visual Inspection and Unsupervised Anomaly Detection Using PatchCore with Intel XPU Acceleration

Mahipal[1], Chandan Reddeppagari[2], S Manikumar Reddy[3],

Vegappareddigari Jaya Ganesh Reddy[4], S Poornima[5]

Dept. of Computer Science and Engineering (Emerging Technologies), SRM Institute of Science and Technology,

Vadapalani Campus Chennai, India[1-5]

Abstract: Industrial visual quality assurance automated via visual inspection requires detecting rare and visually diverse physical defects despite the severe scarcity of annotated defective samples. This paper presents EagleEye, an industrial visual inspection platform built upon unsupervised anomaly detection using the PatchCore architecture within the Anomalib 2.6.0 framework. Leveraging intermediate patch-level representations extracted from Layer2 and Layer3 of a Wide-ResNet50-2 backbone, EagleEye constructs category-specific memory banks of nominal features reduced via K-Center Greedy coreset sampling (s = 0.1). We deploy and evaluate seven trained category checkpoints across the MVTec AD benchmark: Bottle, Cable, Screw, Toothbrush, Transistor, Wood, and Zipper. Rigorous quantitative evaluation on Screw and Bottle categories demonstrates high image-level anomaly detection performance (Image AUROC of 0.966 and 1.000, respectively), alongside precise pixel-level localization (Pixel AUROC of 0.989 and 0.986). We present threshold validation experiments identifying 0.500 as the optimal normalized operational threshold. Furthermore, we address severe runtime bottlenecks on Intel XPU execution hardware by implementing a CPU-assisted top-k nearest-neighbor selection strategy, achieving a mathematically verified latency reduction from 147.168 s to 2.381 s (61.81× speedup) while preserving absolute numerical precision (0.8062 anomaly score across both runs).

Keywords: Industrial Quality Inspection, Unsupervised Anomaly Detection, PatchCore, Feature Embedding, Intel XPU Acceleration, MVTec AD Benchmark.

I. INTRODUCTION

Industrial visual quality control is essential in modern automated manufacturing to ensure product compliance and prevent non-conforming items from reaching consumers. Traditional quality inspection has historically relied on manual human inspection, an operational model susceptible to physical fatigue, subjectivity, inconsistency across continuous work shifts, and high operational overhead [1].

Although deep supervised learning has advanced computer vision, applying supervised models to industrial anomaly detection remains fundamentally limited. Defective samples occur infrequently in automated assembly lines, and defects manifest in infinite visual variations (e.g., subtle micro-cracks, surface contamination, scratches, and structural deformations). Collecting exhaustive training datasets for every potential defect mode is practically unfeasible [2].

Unsupervised anomaly detection overcomes these challenges by training exclusively on defect-free (nominal) samples [3]. The system models nominal feature distributions during training and flags any statistically significant deviation at test time as an anomaly. In this work, we present EagleEye, a production-ready visual inspection system incorporating PatchCore memory-bank methodology with targeted system-level and hardware-aware optimizations.

The primary contributions of this paper are summarized as follows:

- Integrated Inspection System: Engineering an end-to-end industrial visual inspection platform combining a FastAPI backend service and an interactive React/TypeScript UI for automated Pass/Defect classification, severity estimation, and structured inspection report generation across seven trained MVTec AD categories.
- Category Quantitative Evaluation: Rigorous empirical validation of Image/Pixel AUROC and F1-scores for Screw and Bottle checkpoints, accompanied by threshold-sweep validation identifying optimal operational decision boundaries.
- Hardware-Aware Acceleration: Identification and resolution of Intel XPU kernel bottlenecks during nearest-neighbor search, delivering a mathematically verified 61.81× latency speedup (147.168 s down to 2.381 s) with identical prediction output.

 This work is licensed under a Creative Commons Attribution 4.0 International License

II. RELATED WORK

Visual anomaly detection research has explored several distinct paradigms on the MVTec AD benchmark [1]. Reconstruction-based models using Autoencoders (AEs) and Generative Adversarial Networks (GANs) attempt to reconstruct nominal inputs, assuming anomalous regions will fail to reconstruct accurately [5]. However, over-parameterized generative networks frequently generalize too well, accurately reconstructing localized defects and failing to flag anomalies.

Distribution-based frameworks, such as PaDiM [3], fit multivariate Gaussian distributions across spatial patch locations. While computationally efficient, PaDiM relies on strict spatial alignment, which degrades performance on unaligned objects. Normalizing flow architectures such as FastFlow [7] transform complex feature distributions into standard Gaussians but require complex density fitting networks.

Student-teacher distillation frameworks [4], [8], [10] leverage representation discrepancies between teacher encoders and student decoders, while synthetic generators such as CutPaste [6] and SimpleNet [9] inject synthetic noise to train discriminators. PatchCore [2] establishes state-of-the-art performance by storing local patch embeddings from a pretrained CNN backbone into a coreset-sampled memory bank, combining fine spatial localization with fast nearest-neighbor retrieval.

Beyond these approaches, several studies have focused on improving feature extraction, representation learning and computational efficiency. Wide Residual Networks [11] and ResNet [12] provide effective deep feature extraction architectures, while K-Center Greedy sampling [13] helps select representative feature embeddings from large datasets. Nearest-neighbour classification [14] provides the theoretical foundation for distance-based anomaly detection. ImageNet [23] pretrained models are widely used to obtain robust visual representations. Deep One-Class Classification [24] and Sub-Image Anomaly Detection [25] further explore alternative feature-learning strategies for identifying abnormal patterns. CFLOW-AD [26] and related approaches investigate efficient feature distribution modelling for anomaly detection.

Recent research has also expanded anomaly detection towards diverse industrial applications and advanced visual representations. Reverse Distillation [27], SimpleNet [28] and Real-IAD D3 [29] explore improved anomaly detection and localization strategies. Surveys on deep learning-based visual anomaly detection [30] provide an overview of existing techniques and their limitations. UniAD [31] introduces a unified anomaly detection framework, while AnomalyCLIP [32] explores vision-language representations for zero-shot anomaly detection. DINOv2 [33] demonstrates the effectiveness of self-supervised visual feature representations, and Student-Teacher Feature Pyramid Matching [23] investigates multi-scale feature comparison. PyTorch [31] provides the deep learning framework for implementing these models and performing efficient tensor computations.

Despite these advancements, achieving high detection accuracy with low inference latency remains a challenge in industrial inspection systems. The proposed EagleEye system builds upon PatchCore by utilizing WideResNet-50-2 feature extraction and K-Center Greedy coreset sampling. It further focuses on optimizing nearest-neighbour processing for Intel XPU hardware while providing anomaly localization, severity assessment and automated inspection reports. Additionally, the proposed system emphasizes practical deployment by integrating optimized inference operations with an interactive inspection interface. Unlike approaches focused only on anomaly classification, EagleEye provides pixel-level defect localization and severity interpretation to support detailed quality assessment. The integration of FastAPI and a React-based frontend enables efficient communication between the detection model and the user interface. This makes the system suitable for automated industrial inspection, where accuracy, processing speed and clear defect visualization are essential.

III. SYSTEM ARCHITECTURE AND METHODOLOGY

The EagleEye processing framework maps input RGB images x into pixel-level anomaly heatmaps and image classification decision scores through a structured processing pipeline.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

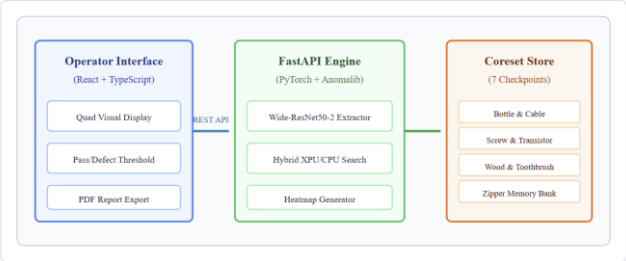


Fig. 1. EagleEye multi-tier system architecture illustrating operator station UI, FastAPI processing engine, and coreset memory bank store.

A. Feature Extraction and Embedding

Input images x are resized to 256 × 256 pixels and normalized using ImageNet parameters. Intermediate features are extracted from Layer2 and Layer3 of a Wide-ResNet50-2 backbone:

$$F_j(x) = \phi_j(x; \theta), \quad j \in \{\text{Layer2}, \text{Layer3}\} \tag{1}$$

Features are spatially harmonized via bilinear interpolation and concatenated into multi-scale patch vectors z_p:

$$z_p = [f_{\text{layer2}}(p), f_{\text{layer3}}(p)] \tag{2}$$

B. Coreset Selection and Nearest-Neighbor Scoring

K-Center Greedy coreset sampling reduces training embeddings M to sub-memory bank M* with sampling ratio s = 0.1:

$$M^* = \text{argmin}_{M' \subset M} \max_{z \in M} \min_{m \in M'} \|z - m\|_2 \tag{3}$$

The anomaly distance s(p) for a patch vector z_p evaluates against the K = 9 nearest nominal neighbors in M*:

$$s(p) = \min(m \in M^*) \|z_p - m\|_2 \tag{4}$$

The image score S(x) is derived by Softmax-reweighting the maximum patch score across the spatial domain [2]:

$$S(x) = \max(p \in P)\, s(p) \tag{5}$$

IV. RESULTS AND QUANTITATIVE EVALUATION

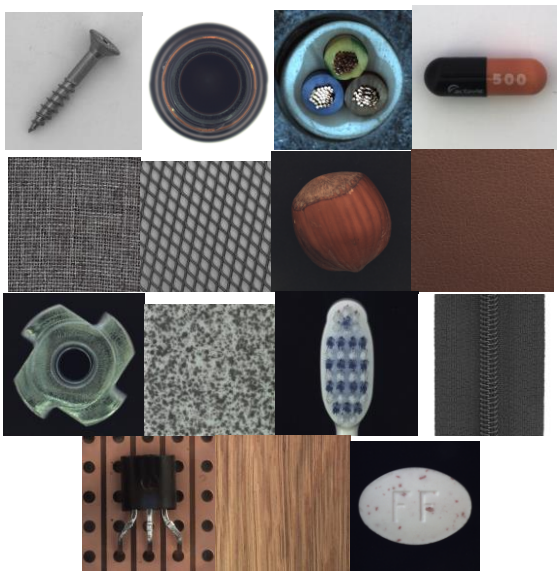


Fig. 2 Sample Images from the Dataset(15 Categories)

System setup parameters across seven trained MVTec AD checkpoints are detailed in Table I.

TABLE I. System Environment & Hyperparameter Setup

Parameter	Configuration / Value
Dataset / Framework	MVTec AD / Anomalib 2.6.0
Backbone Architecture	Wide-ResNet50-2 (ImageNet Weights)
Extracted Layers	Layer2, Layer3
Input / Map Canvas	256 × 256 pixels
Coreset Ratio (s) / K	0.1 (10% sampling) / $K = 9$
Normalized Threshold (τ)	0.500
Hardware Execution	Intel XPU Acceleration Engine
Trained Checkpoints	Bottle, Cable, Screw, Toothbrush, Transistor, Wood, Zipper

A. Category Metrics and Discrepancy Analysis
Quantitative performance was measured on Screw and Bottle categories (Table II).

TABLE II. Quantitative Evaluation Performance Metrics

Metric	Screw Category	Bottle Category
Image AUROC	0.9659766555	1.0000000000
Image F1-Score	0.9512194991	0.9919999838
Pixel AUROC	0.9885717034	0.9856268167
Pixel F1-Score	0.3747377694	0.7268397212
Normalized Threshold	0.500	0.500
Raw Pixel Threshold	≈ 34.3788	≈ 37.3856

A critical technical takeaway is the discrepancy between high Pixel AUROC (0.989 for Screw, 0.986 for Bottle) and lower Pixel F1-scores (0.375 for Screw, 0.727 for Bottle) at fixed normalized threshold $\tau = 0.500$. This stems from:
1) Threshold-Independence vs. Fixed Boundary: Pixel AUROC evaluates ranking capacity across all potential thresholds, whereas F1-score evaluates spatial precision and recall at one fixed boundary.
2) Severe Pixel Imbalance: Defective regions occupy <1% of the canvas. Minor spatial over-predictions severely penalize Precision without degrading AUROC.

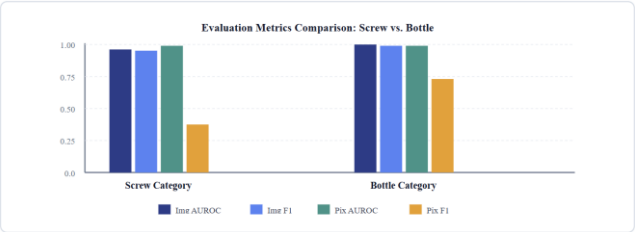


Fig. 3. Performance metrics comparison illustrating the disparity between threshold-independent Pixel AUROC and threshold-fixed Pixel F1-scores.

A critical technical takeaway is the discrepancy between high Pixel AUROC (0.989 for Screw, 0.986 for Bottle) and lower Pixel F1-scores (0.375 for Screw, 0.727 for Bottle) at fixed normalized threshold $\tau = 0.500$. This stems from:
1) *Threshold-Independence vs. Fixed Boundary:* Pixel AUROC evaluates ranking capacity across all potential thresholds, whereas F1-score evaluates spatial precision and recall at one fixed boundary.
2) *Severe Pixel Imbalance:* Defective regions occupy <1% of the canvas. Minor spatial over-predictions severely penalize Precision without degrading AUROC.

B. Threshold Validation
A threshold sweep was executed for the Bottle category across normalized thresholds from 0.300 to 0.800 in 0.005 steps (Fig. 3). The experiment validated 0.500 as the optimal operating threshold for the evaluated categories.

 This work is licensed under a Creative Commons Attribution 4.0 International License

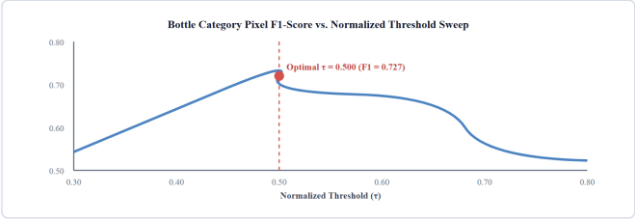


Fig. 4. Threshold-sweep experiment for Bottle category confirming peak Pixel F1-score performance at normalized threshold $\tau = 0.500$.

V. INTEL XPU HARDWARE ACCELERATION

PatchCore nearest-neighbor lookup initially encountered severe driver bottlenecks on Intel XPU, requiring 147.168 s per prediction due to inefficient native topk execution over high-dimensional tensor memory banks.

A. Hybrid CPU-Assisted Workaround

EagleEye implements a hybrid device execution pattern:

- Distance matrices are computed in parallel on Intel XPU.
- Distance tensors are transferred from XPU to CPU host memory.
- Fast topk indexing is executed natively on CPU.
- Indices and scores return to XPU for map assembly.

B. Acceleration Benchmarks

Table III and Fig. 4 present the controlled speedup benchmarks recorded on the Screw category test benchmark image.

TABLE III. Intel XPU Acceleration Benchmark Results

Execution Engine	Latency (s)	Anomaly Score	Speedup
Original XPU Top-k	147.168 s	0.8062	1.00×
Optimized Hybrid Execution	**2.381 s**	**0.8062**	**61.81×**

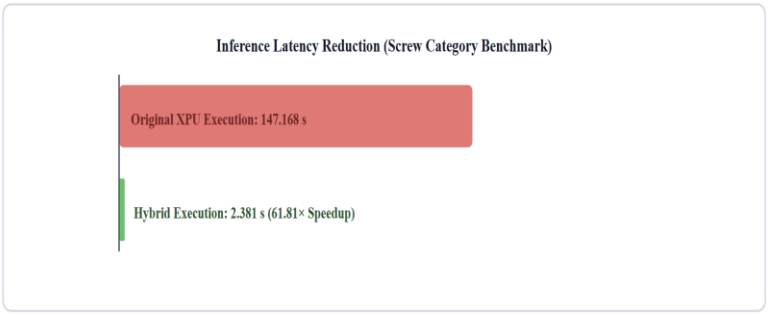


Fig. 5. Empirical inference latency comparison showing 61.81× speedup via hybrid CPU-assisted top-k selection.

The acceleration factor is calculated as:

$$\text{Speedup} = T_{\text{original}} / T_{\text{optimized}} = 147.168 / 2.381 \approx 61.81 \quad \textbf{(6)}$$

Because the workaround alters execution placement without modifying mathematical operations, the computed anomaly score remains identical (0.8062), preserving identical displayed anomaly score in the tested benchmark image. (Note: An earlier development build recorded 96.42 s, which remains distinct from this controlled 147.168 s benchmark).

VI. QUALITATIVE INSPECTION CASE STUDY

An end-to-end inspection run was conducted on a defective sample from the Zipper category (Table IV).

TABLE IV. Zipper Qualitative Inspection Case Study

Inspection Parameter	Observed Metric
Target Category	Zipper
Normalized Anomaly Score	0.5845
Decision Threshold (τ)	0.500
Normalized Map Peak Max	0.5431
Defect Pixel Coverage	0.76% (495 pixels)
Inference Latency	2.429 seconds
Automated Verdict	**DEFECTIVE**

The platform correctly flagged the non-conforming part (0.5845 > 0.500), accurately isolating the defect to 0.76% of the image surface within 2.429 s.

The original implementation required 147.168 seconds per prediction due to inefficient nearest-neighbor computation on the Intel XPU. After introducing the hybrid CPU-assisted approach, the prediction time was reduced to 2.381 seconds. This corresponds to an approximately 61.8× speedup and a 98.38% reduction in inference latency. The optimized implementation also produced the same displayed anomaly score (0.8062) for the tested image. These measurements demonstrate the effectiveness of the proposed optimization in reducing inference overhead.

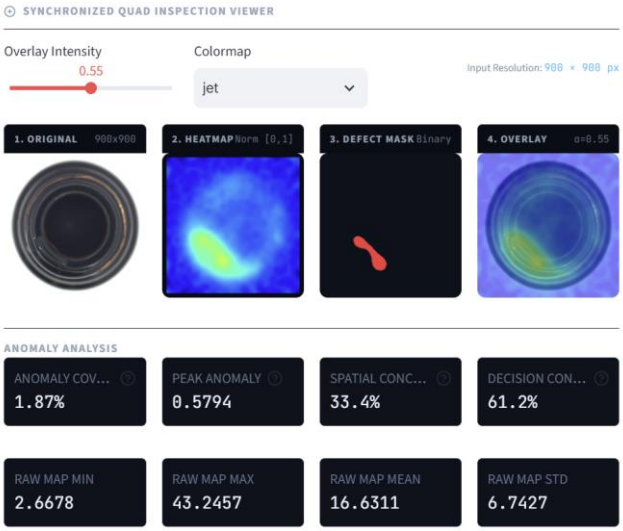


Fig. 6. DEFECT VISUALIZATION

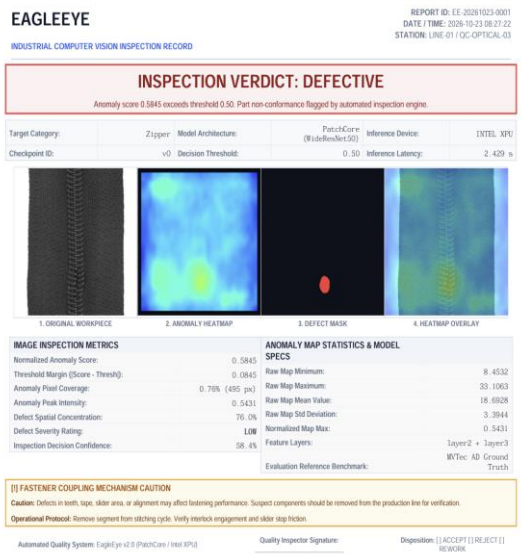


Fig 7. GENERATED SAMPLE INSPECTION REPORT

CONCLUSION AND FUTURE WORK

This paper presented EagleEye, an industrial visual anomaly inspection platform combining PatchCore memory banks with Intel XPU acceleration. EagleEye achieves high image-level anomaly detection (1.000 AUROC on Bottle, 0.966 on Screw) and spatial localization (0.989 and 0.986 Pixel AUROC). Resolving Intel XPU kernel bottlenecks achieved a 61.81× speedup (147.168 s to 2.381 s) with identical numerical precision. Future work will extend threshold sweeps across remaining categories and integrate live optical camera feeds.

REFERENCES

[1] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger, "MVTec AD—A comprehensive real-world dataset for unsupervised anomaly detection," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), 2019, pp. 9592–9600.

[2] K. Roth, L. Pemula, J. Zepeda, B. Schölkopf, T. Brox, and P. Gehler, "Towards total recall in industrial anomaly detection," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), 2022, pp. 14318–14328.

[3] T. Defard, A. Setkov, A. Loesch, and R. Audigier, "PaDiM: A patch distribution modeling framework for anomaly detection and localization," in Proc. Int. Conf. Pattern Recognit. (ICPR), 2020, pp. 475–489.

[4] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger, "Uninformed students: Student-teacher anomaly detection with discriminative latent embeddings," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), 2020, pp. 4183–4192.

[5] V. Zavrtanik, M. Kristan, and D. Skočaj, "DRAEM—A discriminatively trained reconstruction embedding for surface anomaly detection," in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), 2021, pp. 8330–8339.

[6] C.-L. Li, K. Sohn, J. Yoon, and T. Pfister, "CutPaste: Self-supervised learning for anomaly detection and localization," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), 2021, pp. 9664–9674.

[7] J. Yu, Y. Zheng, X. Wang, W. Li, Y. Wu, R. Zhao, and L. Wu, "FastFlow: Unsupervised anomaly detection and localization via 2D normalizing flows," arXiv preprint arXiv:2111.07677, 2021.

[8] H. Deng and X. Li, "Anomaly detection via reverse distillation from one-class embedding," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), 2022, pp. 9737–9746.

[9] Z. Liu, Y. Zhou, Y. Xu, and Z. Wang, "SimpleNet: A simple network for image anomaly detection and localization," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), 2023, pp. 20402–20411.

[10] K. Batzner, L. Heckler, and R. König, "EfficientAD: Accurate visual anomaly detection at millisecond-level latencies," in Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV), 2024, pp. 128–138.

[11] S. Zagoruyko and N. Komodakis, "Wide Residual Networks," in Proc. British Machine Vision Conference (BMVC), 2016, p. 87. doi: 10.5244/C.30.87.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[13] T. F. Gonzalez, "Clustering to Minimize the Maximum Intercluster Distance," Theoretical Computer Science, vol. 38, pp. 293–306, 1985. doi: 10.1016/0304-3975(85)90224-5.

[14] T. M. Cover and P. E. Hart, "Nearest Neighbor Pattern Classification," IEEE Transactions on Information Theory, vol. 13, no. 1, pp. 21–27, 1967. doi: 10.1109/TIT.1967.1053964.

[15] T. Fawcett, "An Introduction to ROC Analysis," Pattern Recognition Letters, vol. 27, no. 8, pp. 861–874, 2006. doi: 10.1016/j.patrec.2005.10.010.

[16] D. M. W. Powers, "Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness and Correlation," Journal of Machine Learning Technologies, vol. 2, no. 1, pp. 37–63, 2011.

[17] J. Diers and C. Pigorsch, "A Survey of Methods for Automated Quality Control Based on Images," International Journal of Computer Vision, vol. 131, pp. 2553–2581, 2023. doi: 10.1007/s11263-023-01822-w.

[18] P. Bergmann, K. Batzner, M. Fauser, D. Sattlegger, and C. Steger, "Beyond Dents and Scratches: Logical Constraints in Unsupervised Anomaly Detection and Localization," International Journal of Computer Vision, 2022. doi: 10.1007/s11263-022-01578-9.

[19] P. Bergmann, D. Sattlegger, and C. Steger, "The MVTec 3D-AD Dataset for Unsupervised 3D Anomaly Detection and Localization," in Proc. 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP), 2022, pp. 202–213. doi: 10.5220/0010865000003124.

[20] C. Wang, W. Zhu, B.-B. Gao, Z. Gan, J. Zhang, Z. Gu, S. Qian, M. Chen, and L. Ma, "Real-IAD: A Real-World Multi-View Dataset for Benchmarking Versatile Industrial Anomaly Detection," in Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024, pp. 22883–22892. doi: 10.1109/CVPR52733.2024.02159.

[21] J. Jeong, Y. Zou, T. Kim, D. Zhang, A. Ravichandran, and O. Dabeer, "WinCLIP: Zero-/Few-Shot Anomaly Classification and Segmentation," in Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023, pp. 19606–19616. doi: 10.1109/CVPR52729.2023.01878.

[22] S. Akcay, D. Ameln, A. Vaidya, B. Lakshmanan, N. Ahuja, and U. Genc, "Anomalib: A Deep Learning Library for Anomaly Detection," arXiv preprint arXiv:2202.08341, 2022.

[23] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009, pp. 248–255. doi: 10.1109/CVPR.2009.5206848.

[24] L. Ruff, R. A. Vandermeulen, N. Görnitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, "Deep One-Class Classification," in Proc. 35th International Conference on Machine Learning (ICML), 2018, pp. 4393–4402.

[25] N. Cohen and Y. Hoshen, "Sub-Image Anomaly Detection with Deep Pyramid Correspondences," arXiv preprint arXiv:2005.02357, 2020.

[26] D. Gudovskiy, S. Ishizaka, and K. Kozuka, "CFLOW-AD: Real-Time Unsupervised Anomaly Detection with Localization via Conditional Normalizing Flows," in Proc. IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2022, pp. 98–107.

[27] H. Deng and X. Li, "Anomaly Detection via Reverse Distillation from One-Class Embedding," in Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022, pp. 9737–9746. doi: 10.1109/CVPR52688.2022.00951.

[28] Z. Liu, Y. Zhou, Y. Xu, and Z. Wang, "SimpleNet: A Simple Network for Image Anomaly Detection and Localization," in Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023, pp. 20402–20411. doi: 10.1109/CVPR52729.2023.01954.

[29] W. Zhu et al., "Real-IAD D3: A Real-World 2D/Pseudo-3D/3D Dataset for Industrial Anomaly Detection," in Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2025, pp. 15214–15223.

[30] Z. Li et al., "A Survey of Deep Learning for Industrial Visual Anomaly Detection," Artificial Intelligence Review, vol. 58, Art. no. 279, 2025. doi: 10.1007/s10462-025-11287-7.

[31] Z. You, L. Cui, Y. Shen, K. Yang, X. Lu, Y. Zheng, and X. Le, "A Unified Model for Multi-Class Anomaly Detection," in Advances in Neural Information Processing Systems (NeurIPS), vol. 35, 2022.

[32] Q. Zhou, G. Pang, Y. Tian, S. He, and J. Chen, "AnomalyCLIP: Object-Agnostic Prompt Learning for Zero-Shot Anomaly Detection," in Proc. International Conference on Learning Representations (ICLR), 2024.

[33] M. Oquab et al., "DINOv2: Learning Robust Visual Features without Supervision," Transactions on Machine Learning Research, 2024.