Rethinking Continual Anomaly Detection on the Edge: Benchmarking Under Realistic Industrial Conditions

Chad Weatherly, Sen Lin

University of Houston, Houston, TX, USA

Abstract. Continual anomaly detection (CAD) addresses the need for industrial inspection systems to adapt to evolving production conditions, yet existing methods share three critical gaps: unrealistic evaluation, no systematic comparison, and no consideration of edge deployment constraints. We introduce a unified benchmark combining discrete-task evaluation on structural and logical anomalies, a novel continuous drift protocol, the first head-to-head comparison of all published CAD methods, and computational efficiency profiling on edge hardware. Our results reveal that existing CAD methods do not consistently outperform traditional approaches with simple experience replay. Thus motivated, we propose DINOSaur, a training-free method combining a frozen DINOv3 backbone with spatially-indexed coreset memory and neighborhood-restricted anomaly scoring. DINOSaur achieves zero forgetting by construction, outperforms all evaluated methods on every protocol except geometric drift (where all methods collapse to chance), and runs at sub-100 ms inference on an NVIDIA Jetson Orin Nano, with on-device adaptation to new tasks in under 30 seconds.

Keywords: Continual Learning · Anomaly Detection · Industrial Inspection · Benchmark · Edge Deployment

1 Introduction

The rise of Industry 4.0 has brought increased automation to manufacturing, including the adoption of AI-driven quality control. In production environments, poor quality control can be extremely costly, and missed defects can result in warranty claims, product recalls, and significant reputational damage [11]. Anomaly detection (AD), which identifies defective products using models trained primarily on normal samples, has emerged as a natural fit for automated visual inspection [7] in industrial environments.

However, real-world manufacturing processes are never truly static. Environmental conditions fluctuate, tools wear and degrade, material batches vary, and production parameters drift over time. These factors cause the distribution of normal product appearances to shift continuously, posing a fundamental challenge for AD systems trained under the assumption of a fixed data distribution. When the normal distribution changes, a previously deployed AD model

may either miss genuine defects or flag normal variations as anomalous, both of which carry significant economic consequences. Continual learning (CL) offers a framework that enables models to adapt to new data while retaining knowledge of previously learned distributions [31]. Several recent works have proposed methods for continual anomaly detection (CAD), combining AD architectures with CL strategies to maintain performance as data evolves across sequential tasks [18, 22, 30].

Despite this progress, we identify three critical gaps in the current CAD literature:

(1) Computational feasibility ignored. Industrial AD systems typically run on edge devices with limited compute, often requiring multiple inference passes per second to keep pace with production throughput. Recent CAD methods employ large-scale architectures built on vision-language models (VLMs), diffusion models, or large vision transformers, with no analysis of whether these models could feasibly be deployed in the constrained environments [13,19,32,39]. While recent work studies efficient anomaly detection on edge hardware [3, 4], it targets static or single-backbone replay settings. Our feasibility concern is with dedicated CAD methods, whose added machinery for sequential learning and task identification has not been examined for edge deployment.

(2) Unrealistic and narrow evaluation. Existing CAD benchmarks treat each product categories as distinct tasks, creating sharp and discrete boundaries (*e.g.*, *bottle* to *cable*). In reality, industrial drift is gradual—the same product changes appearance over time due to process variations. No existing benchmark captures this continuous drift scenario. Moreover, evaluation is limited to *structural* anomalies (surface defects), ignoring *logical* anomalies (missing components, incorrect arrangements) [6]—a significant unsolved challenge that no CAD method has been evaluated on.

(3) Lack of systematic comparison. The three published CAD methods, i.e., DNE [18], IUF [30], and UCAD [22], were each evaluated independently against different baselines and under different experimental conditions. The three methods have never been comprehensively evaluated against one another, making it impossible to understand their relative strengths and weaknesses, let alone their feasibility in real-world scenarios.

To address these gaps, this paper makes the following contributions:

- We introduce a **unified evaluation framework** for CAD that, for the first time, combines a head-to-head comparison of all published CAD methods, a novel *continuous drift protocol*, evaluation on *logical anomalies*, and computational efficiency profiling on edge hardware.
- Our evaluation reveals that **dedicated CAD methods do not consistently outperform** traditional baselines with simple experience replay. Motivated by this finding, we propose **DINOSaur**[1], a training-free method that outperforms all evaluated approaches at a fraction of the computational cost, enabling practical edge deployment.

[1] Code and data: `https://github.com/Continue-Edge-AI-Lab/Rethinking-Continual-AD`

2 Related Work

Anomaly Detection in Industrial Settings. Industrial anomaly detection (IAD) methods broadly fall into two paradigms. *Distribution-based methods* [10, 24,26] model the distribution of normal-sample features and flag out-of-distribution test samples as anomalous, using techniques such as nearest-neighbor scoring, multivariate Gaussian fitting, or learned normality scores. *Reconstruction-based methods* [34, 35] train encoder-decoder networks on normal images and use high reconstruction error as an anomaly signal. Unified or multi-class AD [34] handles multiple categories simultaneously but assumes access to all data at once, unlike CAD's sequential constraint.

Large-Scale Models for Anomaly Detection. Recent work has applied large foundation models to AD, including LLM-based reasoning [13], vision-language prompting [39], in-context learning [40], and diffusion models [19, 32]. While these achieve strong benchmark results, they require billions of parameters and GPU-class hardware with inference latencies measured in seconds per image—fundamentally incompatible with edge deployment where throughput of several images per second is required. Moreover, these large models are not immune to catastrophic forgetting: adapting vision-language models such as CLIP to sequential tasks causes significant degradation in zero-shot transfer ability [38], and multimodal LLMs similarly fail to retain their pre-trained visual understanding after fine-tuning [37]. This dual challenge—computational infeasibility *and* susceptibility to forgetting—further motivates our focus on lightweight, training-free approaches for edge-deployed CAD.

Continual Learning. CL addresses the problem of learning from a sequence of tasks $\{D_1, D_2, \ldots, D_T\}$, subject to the constraint that when training on task t, only dataset D_t is accessible (data from $D_1, \ldots, D_{t-1}$ is unavailable). Each task is a unique set of data, $D_t = \{x_t, y_t\}$ for $t \in [1, 2, \ldots, T]$. Under the CL framework, we aim to build a model that can continuously learn new tasks with minimum *Catastrophic Forgetting* of the knowledge learned from previous tasks [31].

Existing CL strategies fall into three broad families. *Regularization-based methods* add penalty terms to the loss function that restrict weight updates in order to preserve the important knowledge for previous tasks. Elastic Weight Consolidation (EWC) [16] uses Fisher information to identify important parameters, Synaptic Intelligence (SI) [36] tracks parameter importance online, and Memory Aware Synapses (MAS) [1] computes importance based on output sensitivity. *Architecture-based methods* allocate dedicated parameters or sub-networks for different tasks, as in Progressive Neural Networks [27]. *Memory-based methods* store information about previous tasks and modify the new task learning to leverage this information, which can be further divided into *rehearsal-based methods* and *gradient projection based methods*. The former stores raw exemplars or feature representations from previous tasks and replay them during training on new tasks, such as Experience Replay [25] and Dark Experience Replay [9]. The latter [20, 21, 28] stores gradient information of previous tasks in this memory and uses this to modify the gradient direction for the new task. Empirically, memory-based methods consistently achieve the strongest performance across

CL benchmarks [31], as they most directly address forgetting by maintaining access to past information.

Continual Anomaly Detection. To date, three published methods have specifically addressed the problem of CAD in realistic industrial settings. **DNE** [18] stores per-task embedding statistics from a Vision Transformer (ViT) backbone and scores anomalies via Mahalanobis distance at image-level only. **IUF** [30] uses a ViT encoder-decoder with regularized latent space separation and constrained gradient updates. **UCAD** [22] combines contrastive learning with SAM [15] for texture decomposition and a task-specific knowledge bank. Detailed architectural descriptions of each method are provided in the supplementary material.

Several benchmarks for CAD have emerged very recently, but don't compare CAD methods directly. Bugarin *et al.* [8] introduce a pixel-level CAD benchmark on MVTec-AD, evaluating off-the-shelf AD methods (PatchCore, CFA, STFPM, EfficientAD, FastFlow, among others) wrapped with standard replay or memory-bank strategies. Pezzè *et al.* [23] similarly benchmark existing AD methods under replay, contributing a compressed-replay strategy that reduces memory via super-resolution but with no new detection architecture. Continual-MEGA [17] offers a large-scale benchmark and evaluates zero-shot generalization; its evaluated methods are CAD methods and CLIP-based VLMs, though it proposes a CLIP-based baseline with mixture-of-expert adapters specifically to mitigate forgetting in large models. A common thread across all three is reliance on generic CL strategies, predominantly replay, applied to conventional AD backbones, without methods designed around the specific constraints of continual anomaly detection (e.g., non-stationary normality, unsupervised, or edge deployment). Closest to our edge focus, Barusco *et al.* reduce the cost of static AD on edge hardware [3] and study continual VAD on the edge with compressed replay [4], but both apply generic replay to a single static backbone and do not compare dedicated CAD methods, continuous drift, or logical anomalies. Our benchmark differs in several further respects: we are the first to include a *continuous drift* protocol (rather than only discrete category shifts), the first to evaluate CAD on *logical anomalies* via MVTec-LOCO, and the first to profile *computational efficiency on actual edge hardware.*

3 A Unified Benchmark for Continual Anomaly Detection

3.1 Motivation

The central motivation for our benchmark is the disconnect between how CAD methods are currently evaluated and how industrial AD systems actually operate. In a real manufacturing environment, an inspection system monitors the same product over time. The product's visual appearance changes gradually as environmental conditions shift (temperature, humidity, lighting), tools degrade, material batches change, or production processes drift. These changes create a continuous evolution of the normal data distribution—not the sharp, categorical shifts simulated by existing IAD datasets. For example, a factory inspecting magnetic tiles for surface defects will experience gradual sensor degradation, seasonal

lighting shifts, and material batch variation—all of which alter what "normal" looks like, yet the system must continue detecting genuine defects throughout. This is fundamentally different from a system that must suddenly transition from inspecting bottles to inspecting cables.

Beyond the distribution drift problem, existing CAD evaluations are also narrow in scope. Industrial defects are not limited to surface-level structural anomalies; logical anomalies, such as missing components and incorrect arrangements, are equally prevalent and require fundamentally different detection strategies, yet no published CAD method has been evaluated on them.

Our benchmark therefore evaluates CAD methods along three axes corresponding to the identified gaps: (1) *evaluation realism and scope*, covering both the standard discrete task protocol and a novel continuous drift protocol alongside logical anomaly evaluation via MVTec-LOCO; (2) *systematic comparison* of all existing CAD methods against each other and against traditional AD baselines; and (3) *computational efficiency profiling* for edge deployment feasibility.

3.2 Discrete Task Protocol

For our discrete task protocol, we used the standard datasets for both AD and CAD [33]:

MVTec-AD. MVTec-AD [7] contains 15 object categories spanning textures and objects, each treated as a separate task learned sequentially. Only normal samples are used during training (unsupervised setting), reflecting the practical reality that anomalous samples are scarce in manufacturing. This protocol serves as our baseline for direct comparison with existing methods.

MVTec-LOCO. MVTec-LOCO [6] contains 5 categories that include both structural and *logical anomalies*—defects requiring global scene understanding. The sequential protocol mirrors MVTec-AD, with each category as a task.

3.3 Continuous Drift Protocol

To simulate realistic industrial data drift, we design a protocol based on the Magnetic Tile Defects (MTD) dataset [14]. MTD contains a single product category (magnetic tiles) with multiple defect types (blowhole, crack, break, fray, uneven), making it ideal for studying within-product distribution shift rather than between-product category changes.

Drift Simulation via Progressive Augmentation. We construct a sequence of $T = 10$ tasks from the MTD dataset by applying image augmentations of progressively increasing intensity. Let $\mathcal{A}(\mathbf{x}, \alpha)$ denote an augmentation function applied to image $\mathbf{x}$ with intensity α. For task $t \in \{1, \ldots, T\}$, the intensity is greater for each successive task:

$$\alpha_t > \alpha_{t-1} \; \forall \; t > 1. \tag{1}$$

The data for each task is:

$$D_t = \{\mathcal{A}(\mathbf{x}, \alpha_t) \mid \mathbf{x} \in D_{\text{MTD}}\}, \tag{2}$$

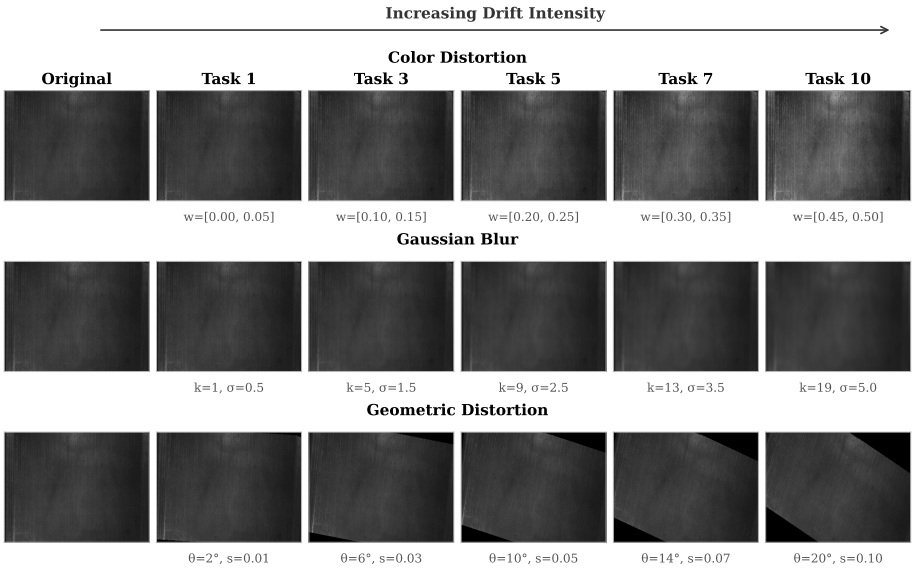

Fig. 1: Progressive augmentation on MTD across tasks for each drift type. *Top:* Color distortion (brightness, contrast, saturation). *Middle:* Gaussian blur (increasing kernel size and σ). *Bottom:* Geometric distortion (rotation, translation, scale, shear). Intensity increases from left (Task 1, minimal distortion) to right (Task 10, maximum distortion). Defect masks are unchanged under blur and color augmentation, and are transformed identically to the image under geometric augmentation.

where D_{MTD} is the original MTD dataset. As t increases, images undergo more severe distortion, simulating gradual environmental or process drift.

Augmentation Types. We run three separate experimental tracks, each simulating a different category of real-world drift:

- **Color distortion**: simulates lighting changes and material batch variation via progressive hue, saturation, brightness, and contrast shifts.
- **Gaussian blur**: simulates sensor degradation and focus drift via increasing kernel size and standard deviation.
- **Geometric distortion**: simulates orientation changes and camera misalignment via progressive perspective transforms, rotations, and warping.

Each augmentation type maps to documented sources of variation in industrial environments [12], providing interpretable and physically motivated drift scenarios rather than arbitrary distribution shifts. We empirically evaluated the intensity ranges to ensure that the magnetic tile objects were still visible. The exact parameter values for all 10 intensity levels of each drift type are provided in the supplementary material and illustrated in Figure 1.

3.4 Evaluation Metrics

Image-level anomaly scores are obtained by taking the maximum patch-level score. We evaluate all methods using the following metrics:

Image-level AUROC (Area Under the ROC Curve) measures threshold-independent discrimination between normal and anomalous images. We report the final AUROC averaged across all tasks after training on the last task.

Image-level Accuracy provides a threshold-dependent measure of classification performance on both normal and anomalous samples.

Recall (Sensitivity) specifically measures the fraction of anomalous samples correctly identified, which is critical in industrial settings where missed defects (false negatives) carry higher costs than false positives. Accuracy and recall use a fixed threshold (97.5th percentile of training scores); details on threshold selection are in the supplementary material. Since AUROC is threshold-independent, it remains our primary metric.

Pixel-level AUROC measures localization of the defect region, complementing image-level detection. In distribution-based methods like DINOSaur and PatchCore, pixel-level anomaly maps are derived directly from the same patch-level scores used for image-level prediction, where the image score is the maximum over patch scores. We therefore expect the two to be tightly coupled, and we verify this in Sec. 5.3: across the protocols where pixel scoring is meaningful (excluding geometric drift, where transform padding dominates) they correlate at r=0.945 and preserve the ranking of the top methods, confirming that they are corollaries. Image-level AUROC remains our primary metric, as it matches the binary pass/fail decision common in industrial inspection, while pixel-level AUROC shows that our conclusions are not an artifact of metric choice.

Forgetting Measure. To quantify catastrophic forgetting, we adopt the standard Forgetting Measure (FM) [31]. Let $a_{t,j}$ denote the performance on task j after the model has been trained on task t. The forgetting for task j after training on the final task T is:

$$f_j = \max_{t \in \{1,\ldots,T-1\}} a_{t,j} - a_{T,j}, \tag{3}$$

and the average Forgetting Measure is:

$$\text{FM} = \frac{1}{T-1} \sum_{j=1}^{T-1} f_j. \tag{4}$$

A positive FM indicates performance degradation (catastrophic forgetting), while an FM of zero indicates zero forgetting of previous tasks.

Computational Efficiency. We report model size (total parameters), inference latency (ms/image), and peak memory usage, which directly determine edge deployment feasibility on hardware such as NVIDIA Jetson and Raspberry Pi.

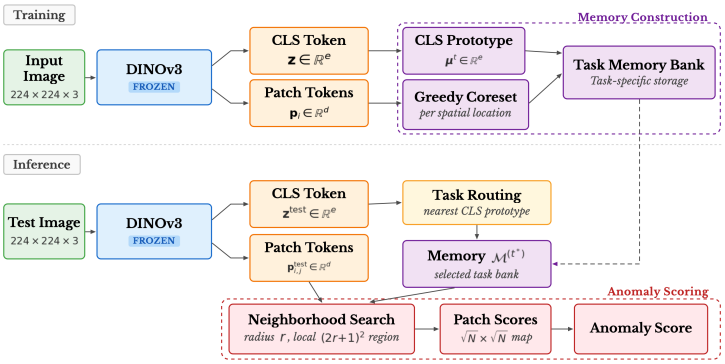

Fig. 2: Overview of the DINOSaur architecture. *Top:* During training, a frozen DINOv3 ViT-S/16 extracts CLS and patch tokens, which are stored in a task-specific memory bank via CLS prototyping and greedy coreset selection. *Bottom:* During inference, the test image's CLS token is compared to stored prototypes for task routing, and patch tokens are scored against the selected memory bank using neighborhood-restricted nearest-neighbor search.

4 DINOSaur: DINO Spatial Anomaly Unsupervised Recognition

4.1 Motivation

Existing CAD methods employ increasingly complex architectures—UCAD incorporates SAM [15], IUF trains a full ViT encoder-decoder, and DNE requires task-specific distribution estimation—yet, as we demonstrate in Sec. 5, this complexity does not consistently translate into superior performance. Memory-based CL mechanisms consistently achieve the strongest results [31], and in industrial settings long-term storage is rarely a constraint even on edge devices. We therefore hypothesize that a powerful frozen feature extractor paired with a non-parametric memory bank is sufficient: it yields competitive performance with zero forgetting by construction, since no model weights are updated. Thus motivated, we propose DINOSaur: a minimal baseline combining a frozen feature extractor with spatially-structured memory and neighborhood-based anomaly scoring, designed for edge deployment. An overview is shown in Figure 2.

4.2 Feature Extraction with DINOv3

DINOSaur uses a frozen DINOv3 ViT-S/16 [29] as its feature backbone—the smallest variant in the DINOv3 family, with only 21M parameters—to demonstrate that strong CAD performance does not require large backbones. Larger variants (ViT-B/16, ViT-L/16) would likely improve detection quality at the cost of higher inference latency and memory usage; we deliberately select the smallest backbone to establish a lower bound representative of edge deployment constraints. We also chose to use the ViT version, as the CLS Token is necessary

for task identification and routing. DINOv3 is a self-supervised vision transformer trained with student-teacher distillation and Gram anchoring, producing both a global CLS token and spatially distinctive patch-level features, considered current state of the art in producing embedding vectors with hierarchical conceptual understanding.

Given an input image $\mathbf{x} \in \mathbb{R}^{H \times W \times 3}$, the frozen DINOv3 encoder Φ produces a set of patch-level feature vectors:

$$\{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_N\} = \Phi(\mathbf{x}), \quad \mathbf{p}_i \in \mathbb{R}^d, \tag{5}$$

where $N = (H/s) \times (W/s)$ is the number of patches determined by the patch size s, and d is the feature dimension. We extract features from the final layer of the transformer, as these features contain descriptive, hierarchical and semantic understanding of the image patch.

Using a frozen backbone eliminates weight updates—the primary source of catastrophic forgetting—while DINOv3's self-supervised features provide strong semantic representations without task-specific adaptation.

4.3 Spatially-Indexed Memory Bank

Rather than aggregating all patch features into a single flat memory bank, DINOSaur maintains a novel *spatially-indexed* memory where each patch location has its own independent coreset. Given D training images from the current task, we collect, for each spatial location (i, j) in the $\sqrt{N} \times \sqrt{N}$ patch grid, the set of all feature vectors observed at that position across the task training set:

$$\mathcal{P}_{i,j} = \{\mathbf{p}_{i,j}^{(k)} \mid k = 1, \ldots, D\}, \quad |\mathcal{P}_{i,j}| = D. \tag{6}$$

We then apply greedy coreset selection [26] independently at each location, iteratively choosing the point farthest from the current coreset, ensuring we have a representative set of location-specific feature vectors while also minimizing our memory bank size. This yields a per-location coreset $\mathcal{C}_{i,j}$ of size $M = \max(20, \lfloor D \cdot \rho \rfloor)$, where ρ is the coreset ratio, a hyperparameter in the range $(0, 1]$. The full memory bank for a task is the spatially-structured tensor $\mathcal{M} \in \mathbb{R}^{\sqrt{N} \times \sqrt{N} \times M \times e}$, where e is the embedding dimension of the feature vectors. This structure preserves the spatial layout of the image in the memory bank, which is essential for the neighborhood-based scoring described below.

4.4 Neighborhood-Restricted Anomaly Scoring

During inference, a test image must first be routed to the appropriate task-specific memory bank (described in Sec. 4.5). Given the selected memory bank $\mathcal{M}$, we compute the anomaly score for each test patch by comparing it only to memory features within a local spatial neighborhood. Given a test patch at

location (i, j) with feature vector $\mathbf{p}_{i,j}^{\text{test}}$, we define the neighborhood memory as the union of patch coresets within a radius r:

$$\mathcal{N}_{i,j}^{r} = \bigcup_{\substack{i' \in [i-r, i+r] \\ j' \in [j-r, j+r]}} \mathcal{C}_{i',j'}. \tag{7}$$

The patch-level anomaly score is then:

$$s_{i,j} = \min_{\mathbf{c} \in \mathcal{N}_{i,j}^{r}} \|\mathbf{p}_{i,j}^{\text{test}} - \mathbf{c}\|_2, \tag{8}$$

and the image-level score is the maximum over all patch locations:

$$S(\mathbf{x}_{\text{test}}) = \max_{(i,j)} s_{i,j}. \tag{9}$$

This neighborhood restriction enforces *spatial consistency*—anomalies are detected based on what *should* appear at a given location—and reduces computation from $N \cdot M$ to at most $(2r+1)^2 \cdot M$ comparisons per patch, a $4\times$ reduction for our default $r = 3$ on a 14×14 grid.

4.5 Continual Learning via Task-Specific Memory

DINOSaur handles continual learning through two complementary mechanisms: **task-specific memory banks** and **CLS token prototypes** for unsupervised task identification.

1) Task-specific memory. Rather than accumulating all features into a single growing bank, DINOSaur maintains a separate, spatially-indexed memory bank $\mathcal{M}^{(t)}$ for each task t. When a new task arrives, its normal training images are processed through the frozen encoder, and a new per-location coreset is constructed and stored independently for that task, giving us a smaller memory footprint by using an optimal set of embedding vectors. Since the feature extractor Φ is never updated, the number of tasks able to be processed is limited mainly by memory and storage constraints, not model architecture.

2) CLS token prototypes. DINOv3 also produces a global CLS token $\mathbf{z} \in \mathbb{R}^e$ per image. During training on task t, we compute the mean CLS token $\boldsymbol{\mu}^t$ across all training images as a compact task prototype. At test time, unsupervised task routing selects the closest prototype:

$$t^* = \arg\min_t \|\mathbf{z}^{\text{test}} - \boldsymbol{\mu}^t\|_2, \tag{10}$$

and the selected task's memory bank $\mathcal{M}^{(t^*)}$ is used for scoring. This requires no task labels at inference—only a single distance computation per stored task.

5 Experiments

5.1 Experimental Setup

Evaluated Methods. We evaluate the following methods:

- **CAD methods**: DNE [18], IUF [30], and UCAD [22]—the three published continual anomaly detection methods.
- **Traditional AD baselines with replay**: PatchCore [26] and EfficientAD [5]—augmented with experience replay [25]. Replay budget and augmentation strategy details are in the supplementary material.
- **DINOSaur** (ours)—the proposed baseline described in Sec. 4.

For all methods, we use official implementations where available and faithfully reimplement otherwise, matching reported hyperparameters. All methods are trained for 300 epochs per task.

Training Protocol. All experiments use the *unsupervised* setting, where only normal samples are available during training, reflecting real industrial scenarios. Models are trained sequentially on each task without access to previous task data (the standard CL constraint), except for the generous replay buffer provided to PatchCore and EfficientAD. Task orderings for all protocols are provided in the supplementary material. Training is conducted on a single NVIDIA RTX 4090. Edge inference profiling is performed on an NVIDIA Jetson Orin Nano and a Raspberry Pi 5, as detailed in Sec. 5.4.

5.2 Main Results

Table 1 presents the image-level AUROC, Accuracy, Recall, and Forgetting Measure (FM) for all evaluated methods on MVTec-AD and MVTec-LOCO. Key findings:

- DINOSaur and PatchCore with replay dominate on both protocols; dedicated CAD methods fall short.
- DNE and IUF collapse to predicting "normal" (zero recall on MVTec-AD), revealing fundamental architectural limitations in the unsupervised setting.
- Among dedicated CAD methods, only UCAD demonstrates meaningful learning, yet lags PatchCore by over 12 percentage points in AUROC on MVTec-AD.
- DINOSaur and UCAD achieve zero forgetting; PatchCore's modest forgetting shows that replay alone mitigates catastrophic forgetting.

While DINOSaur and PatchCore both occupy the top tier, the two methods differ in fundamental ways. PatchCore aggregates all patch features into a single flat memory bank and scores test patches via global nearest-neighbor search, relying on experience replay (100 samples per previous task) to mitigate forgetting when new tasks arrive. DINOSaur, by contrast, requires no replay, and its spatially-indexed memory with neighborhood-restricted scoring (Sec. 4.4) enforces spatial consistency that a flat bank cannot. This architectural difference translates directly to edge performance: DINOSaur is 23% faster than PatchCore on the Jetson (91.6 vs. 119 ms; Tab. 4) while achieving higher AUROC on every protocol. Moreover, DINOSaur's performance is tunable via the coreset ratio ρ:

Table 1: Discrete-Task Results: Image-level AUROC (↑), Accuracy (↑), Recall (↑), and Forgetting Measure (↓) on MVTec-AD and MVTec-LOCO, averaged across all tasks. Methods marked with * are augmented with experience replay. Best results per column are in **bold**.

	MVTec-AD				MVTec-LOCO			
Method	AUROC	Acc	Recall	FM	AUROC	Acc	Recall	FM
DNE [18]	0.500	0.277	0.000	0.012	0.500	0.368	0.000	**−0.022**
IUF [30]	0.508	0.300	0.074	**−0.066**	0.572	0.436	0.234	0.002
UCAD [22]	0.798	0.635	0.552	0.000	0.629	0.537	0.389	0.000
PatchCore* [26]	0.925	0.808	0.804	0.023	0.743	0.655	0.521	0.038
EfficientAD* [5]	0.529	0.310	0.071	0.051	0.485	0.384	0.053	0.025
DINOSaur (ours)	**0.968**	**0.829**	**0.991**	0.000	**0.768**	**0.709**	**0.845**	0.000

on smaller or less variable tasks, reducing ρ below the default 10% would further decrease storage and latency with only modest AUROC loss (Tab. 5), giving practitioners an explicit efficiency–accuracy knob. We note that our PatchCore* baseline is mechanistically equivalent to PatchcoreCL [2]: both maintain persistent feature access without weight updates and use task-structured retrieval. Beyond the spatial indexing and neighborhood-restricted scoring above, DINOSaur further differs from both through explicit CLS-prototype task routing.

The continuous drift results (Tab. 2) reveal distinct patterns across drift types. On color and blur drift, DINOSaur and PatchCore clearly separate from dedicated CAD methods, confirming that strong features matter more than specialized CL mechanisms under continuous shift. Notably, continuous drift is not uniformly easier than discrete task shifts: DINOSaur and PatchCore achieve comparable or lower AUROC on MTD color and blur drift than on MVTec-AD, while the weaker CAD methods (DNE, IUF) show similar near-chance performance regardless of protocol, suggesting their failures are fundamental architectural limitations rather than protocol-specific artifacts.

Among drift types, geometric distortion is uniquely challenging—all methods collapse near chance. This is because geometric transformations fundamentally alter the spatial relationships between patch features, disrupting the very structure that distribution-based methods rely on. Color and blur distortions, by contrast, modify appearance while preserving spatial structure, making them far more amenable to patch-based detection. This distinction has practical implications: industrial scenarios involving mechanical deformation or perspective changes (*e.g.*, conveyor belt misalignment, camera drift) may require fundamentally different representations than those sufficient for photometric variations. DINOSaur maintains near-zero forgetting across all drift types ($\text{FM} \leq 0.013$), and on color drift PatchCore actually exhibits negative FM (-0.043), indicating backward transfer—continued training on shifted data *improves* detection on earlier tasks.

Table 2: Continuous Drift Results: Image-level AUROC (↑), Accuracy (↑), Recall (↑), and Forgetting Measure (↓) on MTD-Color, MTD-Geo, and MTD-Blur, averaged across all tasks. Methods marked with * are augmented with experience replay. Best results per column are in **bold**.

	MTD-Color				MTD-Geo				MTD-Blur			
Method	AUROC	Acc	Recall	FM	AUROC	Acc	Recall	FM	AUROC	Acc	Recall	FM
DNE [18]	0.520	0.529	0.308	0.021	0.480	0.517	**0.342**	0.032	0.555	0.573	0.062	−0.003
IUF [30]	0.521	0.532	0.294	−0.008	**0.525**	**0.588**	0.048	0.002	0.516	0.567	0.042	0.001
UCAD [22]	0.566	0.591	0.141	0.030	0.480	0.548	0.107	0.007	0.582	0.586	0.137	0.003
PatchCore* [26]	0.836	0.710	0.530	**−0.043**	0.509	0.564	0.042	**−0.008**	0.832	0.751	**0.685**	**−0.022**
EfficientAD* [5]	0.565	0.573	0.080	−0.007	0.510	0.567	0.050	−0.004	0.566	0.564	0.051	−0.010
DINOSaur (ours)	**0.928**	**0.836**	**0.861**	0.000	0.519	0.567	0.050	0.013	**0.891**	**0.810**	0.676	−0.002

Table 3: Image- vs. pixel-level AUROC (↑), averaged across all tasks. Image-level numbers are reproduced from Tabs. 1 and 2. DNE has no spatial map ("—"). Under geometric drift, pixel-level scoring is unreliable because the affine padding dominates the pooled metric, so we omit it ("—"). Best pixel-level result per protocol is in **bold**.

	MVTec-AD		MVTec-LOCO		MTD-Color		MTD-Geo		MTD-Blur	
Method	img	pix	img	pix	img	pix	img	pix	img	pix
DNE [18]	0.500	—	0.500	—	0.520	—	0.480	—	0.555	—
IUF [30]	0.508	0.525	0.572	0.606	0.521	0.557	0.525	—	0.516	0.585
UCAD [22]	0.798	0.822	0.629	0.651	0.566	0.564	0.480	—	0.582	0.537
PatchCore* [26]	0.925	0.945	0.743	0.742	0.836	0.770	0.509	—	0.832	0.730
EfficientAD* [5]	0.529	0.402	0.485	0.429	0.565	0.599	0.510	—	0.566	0.636
DINOSaur (ours)	**0.968**	**0.947**	**0.768**	**0.810**	**0.928**	**0.875**	0.519	—	**0.891**	**0.857**

5.3 Pixel-level Evaluation

Table 3 reports image- and pixel-level AUROC side by side. Across the protocols where pixel scoring is meaningful (MVTec-AD, MVTec-LOCO, MTD-color, MTD-blur), the two correlate at r=0.945 and preserve the ranking of the top methods, confirming that pixel-level performance is largely a corollary of image-level detection. DINOSaur retains its pixel-level lead on all four. We omit geometric pixel-AUROC, where the affine padding dominates the pooled metric. DNE has no spatial map, so its pixel entries are undefined.

5.4 Computational Efficiency

Tab. 4 compares the computational requirements of all evaluated methods on two representative edge platforms. Industrial inspection systems typically require processing multiple images per second on hardware with limited GPU memory; a method that cannot meet these constraints is not a viable CAD solution regardless of its detection metrics.

Among the dedicated CAD methods, IUF exceeds the Jetson's GPU memory entirely and requires 3.7 seconds per image on the Raspberry Pi. UCAD's reliance on SAM [15] results in nearly 1 second per image on both devices. DNE is the most efficient method overall—fastest on the Raspberry Pi (560 ms, 1.8

Table 4: Computational efficiency on two edge platforms (NVIDIA Jetson Orin Nano, Raspberry Pi 5). Latency is mean ± std over 30 single-image runs. Storage is measured on-device per task; "—" marks weight-update methods with no task-indexed storage (DINOSaur storage shown for MVTec-AD). IUF exceeded the Jetson's memory (OOM). FPS = frames per second; * = experience replay. Best latency per platform in **bold**.

			Jetson Orin Nano		Raspberry Pi 5	
Method	Params (M)	Storage/Task (MB)	Inf. (ms)	FPS	Inf. (ms)	FPS
DNE [18]	86.6	2.3	60.7 ± 4.9	16.5	$\mathbf{560.0 \pm 6.1}$	**1.8**
IUF [30]	631.1	—	OOM	—	3668.6 ± 69.6	0.3
UCAD [22]	36.0	1.2	956.5 ± 7.3	1.0	1058.0 ± 15.4	0.9
PatchCore* [26]	9.3	5.0	119.0 ± 2.4	8.4	1074.5 ± 7.9	0.9
EfficientAD* [5]	8.1	—	$\mathbf{34.7 \pm 0.1}$	**28.8**	1318.2 ± 4.3	0.8
DINOSaur (ours)	21.6	7.1	91.6 ± 9.6	10.9	630.6 ± 10.2	1.6

FPS) and competitive on the Jetson (60.7 ms)—but achieves near-chance AUROC, making its speed advantage moot in practice.

Among methods with meaningful detection, DINOSaur achieves sub-100 ms inference on the Jetson (91.6 ms, 10.9 FPS), faster than PatchCore (119 ms). EfficientAD is faster (34.7 ms) but collapses in detection under continual learning. On the CPU-only Raspberry Pi 5, DINOSaur (630.6 ms) is second-fastest after DNE. Per-task storage (7.1 MB on MVTec-AD) scales with ρ and training set size; details are in the supplementary material.

On-Device Training Feasibility. Training cost matters as much as inference for edge deployment. Gradient-based methods (DNE, IUF, UCAD, EfficientAD) require forward and backward passes over hundreds of epochs per task, with memory far exceeding inference. DINOSaur instead adapts to a new task with a single forward pass per image plus coreset selection, enabling autonomous on-device adaptation without cloud connectivity.

5.5 Analysis and Discussion

On the four protocols with meaningful signal, no dedicated CAD method achieves the top AUROC; DINOSaur leads all four, followed by PatchCore with replay. Under geometric drift every method, dedicated or not, sits at chance. This raises a fundamental question: are the performance gains reported in prior CAD papers genuine algorithmic advances, or artifacts of experimental protocol? We identify the *unsupervised* setting as a key factor. All of our experiments train on normal samples only, which is substantially harder than the supervised setting used in some prior evaluations of DNE and IUF, which use synthetic anomaly generation. The near-chance performance of these methods suggests that their architectures may rely on anomaly examples during training to learn meaningful decision boundaries—a luxury that real industrial deployments rarely afford. When this crutch is removed, dedicated CL mechanisms provide no measurable benefit over simple replay, and the added architectural complexity yields no return on investment for edge-constrained deployment.

Task Ordering Invariance. Because DINOSaur's backbone is frozen and each task receives an independent memory bank, its final performance is *completely invariant* to task ordering—a structural guarantee no other evaluated method provides, since all others either update weights or replay across tasks. This eliminates a confound in CL evaluation and simplifies deployment. Extended analysis is in the supplementary material.

DINOSaur Hyperparameter Sensitivity. DINOSaur has two hyperparameters: the coreset ratio ρ and the neighborhood radius r. We perform a grid search over $\rho \in \{1\%, 2.5\%, 5\%, 10\%, 20\%\}$ and $r \in \{0, 1, 2, 3, 4\}$, reporting the average AUROC across all five protocols in Tab. 5. Performance increases monotonically with ρ but with diminishing returns (1.5% gain from 10% to 20%), while r has a smaller but consistent effect. We select ρ=10% and r=3, balancing detection quality against edge efficiency (sub-100 ms inference on the Jetson). The full per-dataset breakdown is in the supplementary material.

Table 5: Hyperparameter sensitivity: average image-level AUROC across all five benchmark protocols for each coreset ratio (ρ) and neighborhood radius (r) combination. The selected configuration (ρ=10%, r=3) is **bolded**.

	Neighborhood radius r				
Coreset ratio ρ	0	1	2	3	4
1%	0.771	0.773	0.773	0.780	0.778
2.5%	0.765	0.770	0.775	0.776	0.782
5%	0.771	0.775	0.777	0.781	0.785
10%	0.787	0.790	0.794	**0.801**	0.798
20%	0.803	0.816	0.819	0.816	0.822

6 Conclusion

We introduced a unified CAD benchmark addressing three gaps in the literature, i.e., unrealistic evaluation, lack of systematic comparison, and neglect of edge constraints, and showed that existing CAD methods do not consistently outperform traditional AD baselines with simple experience replay. Motivated by this finding, we proposed DINOSaur, which combines a frozen DINOv3 backbone with spatially-indexed coreset memory and task identification/routing to achieve state-of-the-art performance with zero forgetting by construction across four of the five benchmark protocols (every method collapses to chance under geometric drift). Its training-free design enables on-device adaptation in under 30 seconds on an NVIDIA Jetson Orin Nano, making it practical for autonomous industrial deployment.

Future Work. Three directions remain open. First, geometric drift collapses every method to chance; addressing it will likely require geometrically equivariant feature representations beyond what current frozen backbones provide. Second, DINOSaur's neighborhood scoring is inherently local, so logical anomalies that require holistic scene understanding would benefit from added global reasoning alongside the spatial memory. Finally, extending the benchmark to online streaming without explicit task boundaries would bring evaluation closer to real industrial deployment.

References

1. Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., Tuytelaars, T.: Memory Aware Synapses: Learning What (not) to Forget. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) Computer Vision – ECCV 2018, vol. 11207, pp. 144–161. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-030-01219-9_9, https://link.springer.com/10.1007/978-3-030-01219-9_9, series Title: Lecture Notes in Computer Science
2. Barusco, M., Borsatti, F., Beda, N., Pezze, D.D., Susto, G.A.: Towards continual visual anomaly detection in the medical domain. In: Proceedings of the International Workshop on Personalized Incremental Learning in Medicine (PILM) (2025). https://doi.org/10.1145/3746259.3760434
3. Barusco, M., Borsatti, F., Pezze, D.D., Paissan, F., Farella, E., Susto, G.A.: PaSTe: Improving the efficiency of visual anomaly detection at the edge. In: 2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). IEEE (2025), https://openaccess.thecvf.com/content/CVPR2025W/VAND/html/Barusco_PaSTe_Improving_the_Efficiency_of_Visual_Anomaly_Detection_at_the_CVPRW_2025_paper.html
4. Barusco, M., D'Antoni, L., Borsatti, F., Pezze, D.D., Susto, G.A.: Memory Efficient Continual Learning for Edge-Based Visual Anomaly Detection. IFAC-PapersOnLine **59**(26), 85–90 (Jan 2025). https://doi.org/10.1016/j.ifacol.2025.12.015, https://www.sciencedirect.com/science/article/pii/S2405896325026916
5. Batzner, K., Heckler, L., König, R.: EfficientAD: Accurate Visual Anomaly Detection at Millisecond-Level Latencies. In: 2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). pp. 127–137. IEEE (2024). https://doi.org/10.1109/WACV57701.2024.00020, https://openaccess.thecvf.com/content/WACV2024/html/Batzner_EfficientAD_Accurate_Visual_Anomaly_Detection_at_Millisecond-Level_Latencies_WACV_2024_paper.html
6. Bergmann, P., Batzner, K., Fauser, M., Sattlegger, D., Steger, C.: Beyond Dents and Scratches: Logical Constraints in Unsupervised Anomaly Detection and Localization. International Journal of Computer Vision **130**(4), 947–969 (Apr 2022). https://doi.org/10.1007/s11263-022-01578-9, https://doi.org/10.1007/s11263-022-01578-9
7. Bergmann, P., Fauser, M., Sattlegger, D., Steger, C.: MVTec AD — A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 9584–9592. IEEE, Long Beach, CA, USA (Jun 2019). https://doi.org/10.1109/CVPR.2019.00982, https://ieeexplore.ieee.org/document/8954181/
8. Bugarin, N., Bugaric, J., Barusco, M., Pezze, D.D., Susto, G.A.: Unveiling the Anomalies in an Ever-Changing World: A Benchmark for Pixel-Level Anomaly Detection in Continual Learning. In: 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 4065–4074. IEEE, Seattle, WA, USA (Jun 2024). https://doi.org/10.1109/CVPRW63382.2024.00410, https://ieeexplore.ieee.org/document/10678644/
9. Buzzega, P., Boschini, M., Porrello, A., Abati, D., CALDERARA, S.: Dark Experience for General Continual Learning: a Strong, Simple Baseline. In: Advances in Neural Information Processing Systems. vol. 33, pp. 15920–15930. Curran Associates, Inc. (2020), https://proceedings.neurips.cc/paper/2020/hash/b704ea2c39778f07c617f6b7ce480e9e-Abstract.html

10. Defard, T., Setkov, A., Loesch, A., Audigier, R.: PaDiM: A Patch Distribution Modeling Framework for Anomaly Detection and Localization. In: Del Bimbo, A., Cucchiara, R., Sclaroff, S., Farinella, G.M., Mei, T., Bertini, M., Escalante, H.J., Vezzani, R. (eds.) Pattern Recognition. ICPR International Workshops and Challenges. pp. 475–489. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-68799-1_35
11. Dimitriou, E., Georgiou, A.: Automatic Inspection Systems Cut Quality Control Costs by 60%. National Journal of Quality, Innovation, and Business Excellence **2**(1), 23–33 (Mar 2025), https://theeducationjournals.com/index.php/NJQIBE/article/view/147
12. Faber, K., Corizzo, R., Sniezynski, B., Japkowicz, N.: Lifelong Continual Learning for Anomaly Detection: New Challenges, Perspectives, and Insights. IEEE Access **12**, 41364–41380 (2024). https://doi.org/10.1109/ACCESS.2024.3377690, https://ieeexplore.ieee.org/document/10473036/?arnumber=10473036, conference Name: IEEE Access
13. Gu, Z., Zhu, B., Zhu, G., Chen, Y., Tang, M., Wang, J.: AnomalyGPT: Detecting Industrial Anomalies Using Large Vision-Language Models. Proceedings of the AAAI Conference on Artificial Intelligence **38**(3), 1932–1940 (Mar 2024). https://doi.org/10.1609/aaai.v38i3.27963, https://ojs.aaai.org/index.php/AAAI/article/view/27963
14. Huang, Y., Qiu, C., Yuan, K.: Surface defect saliency of magnetic tile. The Visual Computer **36**(1), 85–96 (Jan 2020). https://doi.org/10.1007/s00371-018-1588-5, https://doi.org/10.1007/s00371-018-1588-5
15. Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.Y., Dollar, P., Girshick, R.: Segment Anything. In: 2023 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 4015–4026. IEEE (2023), https://openaccess.thecvf.com/content/ICCV2023/html/Kirillov_Segment_Anything_ICCV_2023_paper.html
16. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., Hadsell, R.: Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences **114**(13), 3521–3526 (Mar 2017). https://doi.org/10.1073/pnas.1611835114, https://pnas.org/doi/full/10.1073/pnas.1611835114
17. Lee, G., Oh, Y., Jang, G., Lee, S., Song, J., Cha, S., Yoo, Y.: Continual-MEGA: A Large-scale Benchmark for Generalizable Continual Anomaly Detection (Jun 2025). https://doi.org/10.48550/arXiv.2506.00956, http://arxiv.org/abs/2506.00956, arXiv:2506.00956 [cs] version: 1
18. Li, W., Zhan, J., Wang, J., Xia, B., Gao, B.B., Liu, J., Wang, C., Zheng, F.: Towards Continual Adaptation in Industrial Anomaly Detection. In: Proceedings of the 30th ACM International Conference on Multimedia. pp. 2871–2880. MM '22, Association for Computing Machinery, New York, NY, USA (Oct 2022). https://doi.org/10.1145/3503161.3548232, https://doi.org/10.1145/3503161.3548232
19. Li, X., Tan, X., Chen, Z., Zhang, Z., Zhang, R., Guo, R., Jiang, G., Chen, Y., Qu, Y., Ma, L., Xie, Y.: One-for-More: Continual Diffusion Model for Anomaly Detection. In: 2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4766–4775. IEEE (2025), https://openaccess.thecvf.com/content/CVPR2025/html/Li_One-for-More_Continual_Diffusion_Model_for_Anomaly_Detection_CVPR_2025_paper.html

20. Lin, S., Yang, L., Fan, D., Zhang, J.: Beyond Not-Forgetting: Continual Learning with Backward Knowledge Transfer. Advances in Neural Information Processing Systems **35**, 16165–16177 (Dec 2022), https://proceedings.neurips.cc/paper_files/paper/2022/hash/6728fcf94660c59c938319a6833a6073-Abstract-Conference.html
21. Lin, S., Yang, L., Fan, D., Zhang, J.: TRGP: Trust region gradient projection for continual learning. In: International Conference on Learning Representations (ICLR) (2022), https://openreview.net/forum?id=iEvAf8i6JjO
22. Liu, J., Wu, K., Nie, Q., Chen, Y., Gao, B.B., Liu, Y., Wang, J., Wang, C., Zheng, F.: Unsupervised Continual Anomaly Detection with Contrastively-Learned Prompt. Proceedings of the AAAI Conference on Artificial Intelligence **38**(4), 3639–3647 (Mar 2024). https://doi.org/10.1609/aaai.v38i4.28153, https://ojs.aaai.org/index.php/AAAI/article/view/28153
23. Pezze, D.D., Anello, E., Masiero, C., Susto, G.A.: Continual learning approaches for anomaly detection. Evolving Systems **16**(4), 111 (Sep 2025). https://doi.org/10.1007/s12530-025-09732-7, https://doi.org/10.1007/s12530-025-09732-7
24. Reiss, T., Cohen, N., Bergman, L., Hoshen, Y.: PANDA: Adapting Pretrained Features for Anomaly Detection and Segmentation. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2805–2813. IEEE, Nashville, TN, USA (Jun 2021). https://doi.org/10.1109/CVPR46437.2021.00283, https://ieeexplore.ieee.org/document/9577693/
25. Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., Wayne, G.: Experience Replay for Continual Learning. In: Advances in Neural Information Processing Systems. vol. 32. Curran Associates, Inc. (2019), https://papers.nips.cc/paper_files/paper/2019/hash/fa7cdfad1a5aaf8370ebeda47a1ff1c3-Abstract.html
26. Roth, K., Pemula, L., Zepeda, J., Scholkopf, B., Brox, T., Gehler, P.: Towards Total Recall in Industrial Anomaly Detection. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 14298–14308. IEEE, New Orleans, LA, USA (Jun 2022). https://doi.org/10.1109/CVPR52688.2022.01392, https://ieeexplore.ieee.org/document/9879738/
27. Rusu, A.A., Rabinowitz, N.C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., Hadsell, R.: Progressive Neural Networks (Oct 2016). https://doi.org/10.48550/arXiv.1606.04671, http://arxiv.org/abs/1606.04671, arXiv:1606.04671 [cs]
28. Saha, G., Garg, I., Roy, K.: Gradient projection memory for continual learning. In: International Conference on Learning Representations (ICLR) (2021), https://openreview.net/forum?id=3AOj0RCNC2
29. Siméoni, O., Vo, H.V., Seitzer, M., Baldassarre, F., Oquab, M., Jose, C., Khalidov, V., Szafraniec, M., Yi, S., Ramamonjisoa, M., Massa, F., Haziza, D., Wehrstedt, L., Wang, J., Darcet, T., Moutakanni, T., Sentana, L., Roberts, C., Vedaldi, A., Tolan, J., Brandt, J., Couprie, C., Mairal, J., Jégou, H., Labatut, P., Bojanowski, P.: DINOv3 (Aug 2025). https://doi.org/10.48550/arXiv.2508.10104, http://arxiv.org/abs/2508.10104, arXiv:2508.10104 [cs]
30. Tang, J., Lu, H., Xu, X., Wu, R., Hu, S., Zhang, T., Cheng, T.W., Ge, M., Chen, Y.C., Tsung, F.: An Incremental Unified Framework for Small Defect Inspection. In: Leonardis, A., Ricci, E., Roth, S., Russakovsky, O., Sattler, T., Varol, G. (eds.) Computer Vision – ECCV 2024. pp. 307–324. Springer, Cham (2025). https://doi.org/10.1007/978-3-031-72751-1_18
31. Wang, L., Zhang, X., Su, H., Zhu, J.: A Comprehensive Survey of Continual Learning: Theory, Method and Application. IEEE Transactions on Pattern Analysis and

Machine Intelligence **46**(8), 5362–5383 (Aug 2024). https://doi.org/10.1109/TPAMI.2024.3367329, https://ieeexplore.ieee.org/document/10444954
32. Wyatt, J., Leach, A., Schmon, S.M., Willcocks, C.G.: AnoDDPM: Anomaly Detection With Denoising Diffusion Probabilistic Models Using Simplex Noise. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 650–656. IEEE (2022), https://openaccess.thecvf.com/content/CVPR2022W/NTIRE/html/Wyatt_AnoDDPM_Anomaly_Detection_With_Denoising_Diffusion_Probabilistic_Models_Using_Simplex_CVPRW_2022_paper.html
33. Xie, G., Wang, J., Liu, J., Lyu, J., Liu, Y., Wang, C., Zheng, F., Jin, Y.: IM-IAD: Industrial Image Anomaly Detection Benchmark in Manufacturing. IEEE Transactions on Cybernetics **54**(5), 2720–2733 (May 2024). https://doi.org/10.1109/TCYB.2024.3357213, https://ieeexplore.ieee.org/document/10443076/, conference Name: IEEE Transactions on Cybernetics
34. You, Z., Cui, L., Shen, Y., Yang, K., Lu, X., Zheng, Y., Le, X.: A Unified Model for Multi-class Anomaly Detection. Advances in Neural Information Processing Systems **35**, 4571–4584 (Dec 2022), https://proceedings.neurips.cc/paper_files/paper/2022/hash/1d774c112926348c3e25ea47d87c835b-Abstract-Conference.html
35. Zavrtanik, V., Kristan, M., Skocaj, D.: DRÆM – A discriminatively trained reconstruction embedding for surface anomaly detection. In: 2021 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 8310–8319. IEEE, Montreal, QC, Canada (Oct 2021). https://doi.org/10.1109/ICCV48922.2021.00822, https://ieeexplore.ieee.org/document/9710329/
36. Zenke, F., Poole, B., Ganguli, S.: Continual Learning Through Synaptic Intelligence. In: Proceedings of the 34th International Conference on Machine Learning. pp. 3987–3995. PMLR (Jul 2017), https://proceedings.mlr.press/v70/zenke17a.html
37. Zhai, Y., Tong, S., Li, X., Cai, M., Qu, Q., Lee, Y.J., Ma, Y.: Investigating the catastrophic forgetting in multimodal large language model fine-tuning. In: Proceedings of the Conference on Parsimony and Learning (CPAL). Proceedings of Machine Learning Research, vol. 234, pp. 202–227. PMLR (2024), https://proceedings.mlr.press/v234/zhai24a.html
38. Zheng, Z., Ma, M., Wang, K., Qin, Z., Yue, X., You, Y.: Preventing Zero-Shot Transfer Degradation in Continual Learning of Vision-Language Models. In: 2023 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 19068–19079 (Oct 2023). https://doi.org/10.1109/ICCV51070.2023.01752, https://ieeexplore.ieee.org/document/10377061, iSSN: 2380-7504
39. Zhou, Q., Pang, G., Tian, Y., He, S., Chen, J.: AnomalyCLIP: Object-agnostic Prompt Learning for Zero-shot Anomaly Detection. In: International Conference on Learning Representations (ICLR) (Oct 2024), https://openreview.net/forum?id=buC4E91xZE
40. Zhu, J., Pang, G.: Toward Generalist Anomaly Detection via In-context Residual Learning with Few-shot Sample Prompts. In: 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 17826–17836. IEEE (2024), https://openaccess.thecvf.com/content/CVPR2024/html/Zhu_Toward_Generalist_Anomaly_Detection_via_In-context_Residual_Learning_with_Few-shot_CVPR_2024_paper.html