Memory-Bounded Continuation of Greedy Sampling for Continual Anomaly Detection

Yoon Gyo Jung*[1]
jung.yoo@northeastern.edu
Jaewoo Park*†[2]
park.jaewoo@aivexvision.ai
Kuan-Chuan Peng[3]
kpeng@merl.com
Seongdeok Bang[2]
bang.seongdeok@aivexvision.ai
Octavia Camps‡[1]
o.camps@northeastern.edu

[1] Electrical and Computer Engineering Department
Northeastern University
Boston, MA, USA

[2] AIVEX Inc.
Seoul, South Korea

[3] Mitsubishi Electric Research Laboratories (MERL)
Cambridge, MA, USA

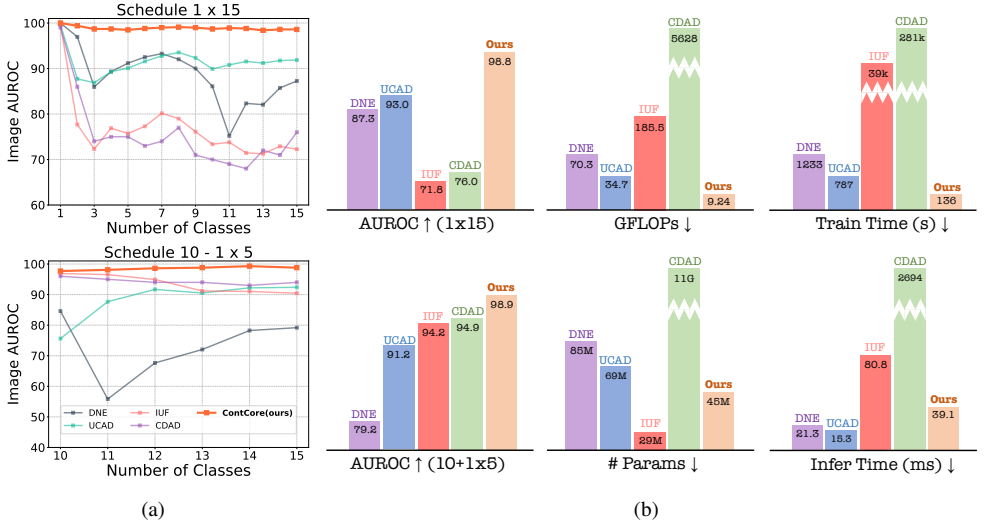
Abstract

Greedy sampling produces a compact yet representative summary of normal data, which is essential for reliable anomaly detection that relies on measuring distance from normality. For continual anomaly detection where tasks arrive sequentially, extending greedy sampling is straightforward with unbounded memory through coreset accumulation. However, practical deployment requires fixed memory where the coreset size remains constant regardless of task count. We observe that *continued greedy sampling*, which iteratively applies greedy selection over previously greedy-sampled sets, effectively preserves representativeness under strict memory limits. Despite discarding data at each step to satisfy the memory constraint, coreset quality degrades gracefully rather than catastrophically, enabling reliable anomaly detection across the tasks. We provide theoretical justification by showing that resulting *greedy-continued coreset* approximates the oracle coreset within a bounded gap. We instantiate this principle in ContCore, which constructs a greedy-continued coreset through greedy expansion on new task features followed by greedy consolidation to enforce the memory budget. Unlike neural methods susceptible to catastrophic forgetting or naive coreset accumulation requiring unbounded memory, ContCore maintains fixed memory with theoretical guarantees. Empirically, ContCore achieves state-of-the-art performance across 11 task schedules on MVTecAD and VisA, and extends effectively to online continual AD settings where prior methods degrade significantly. Code: https://github.com/jungyg/ContCore.

1 Introduction

Solving unsupervised anomaly detection is vital in various fields such as manufacturing, cybersecurity [1], and medical applications [7, 11] due to its productivity and cost reduction.

* equal contribution † project lead ‡ corresponding author



(a) (b)

Figure 1: (a) Comparison of task average image-AUROC for CAD baselines and ContCore on 1×15 and $10 - 1 \times 5$ task schedules on MVTecAD, which demonstrates the effect of catastrophic forgetting in continual learning. (b) Efficiency comparison of CAD methods.

However, discovering the decision boundary between normal and unbounded anomalies makes it challenging. Early models [4, 8, 21, 26, 33, 35, 36] achieve high performance by training a separate model for each class. However, they require costly annotations, long training time, and large model sizes. Multi-class anomaly detection (AD) methods [23, 31] address these issues by training a single model on multiple classes. This provides faster training and eliminates the need for class labeling. However, they fail in dynamic environments, where new samples and tasks are added, as they suffer from catastrophic forgetting [17].

The key challenge in Continual Anomaly Detection (CAD) is preventing catastrophic forgetting and holding the legacy information while detecting newly introduced anomalies under the unsupervised setup (*i.e.*, without class labels and given only normal training images). A naive approach is applying either regularization-based [2, 17, 34] or replay buffer-based [3, 25, 28] continual learning methods to network-based multi-class AD [13, 23, 31]. While regularization-based methods [2, 17, 34] preserve past information by regularizing related parameters from past tasks, they struggle to balance stability and plasticity of the parameters. Moreover, they are highly sensitive to the hyperparameters that controls the trade-off between old and new tasks. Replay-based methods [3, 25, 28] store data from previous tasks in a replay-buffer and replay along with the novel tasks. However, this approach faces scalability issues, as memory usage grows proportionally with the number of tasks. In addition, its performance heavily depends on the quality of the replay samples.

Coreset-based AD [8, 26] builds a coreset memory composed of patch features extracted from a given task and directly uses it for inference. This coreset memory differs from the memory of replay buffer-based methods as it is not used for re-training and the objective is coverage of normal features. These methods can solve continual tasks by adding newly extracted features into the coreset memory without suffering from catastrophic forgetting. However, like replay-based methods, they also struggle with memory overflow as more tasks are added.

To mitigate these issues, dedicated solutions for CAD [18, 19, 20, 27] have been proposed. Memory-based methods such as DNE [18] and UCAD [20] handle multiple steps of simple tasks well, but struggle on complex ones (*e.g.*, 10-class task in Fig. 1(a) bottom). Conversely, parameter regularization-based methods such as CDAD [19] and IUF [27] use iterative SVD to project the gradients of new tasks into the null space of parameters to minimize changes to previous task information, but still suffer from forgetting with a large number of tasks (Fig. 1(a) top).

A natural question arises: can we design a bounded coreset that preserves the representational quality of an unbounded one? Greedy sampling selects maximally separated points, producing a compact yet representative summary of normal data [26]. Since anomaly detection relies on measuring distance from normality, a representative coreset is essential for reliable detection. Extending this property to the continual setting under fixed memory is non-trivial. We observe that *continued greedy sampling*, which iteratively applies greedy selection over previously greedy-sampled sets, effectively preserves representativeness under strict memory limits. We call the resulting set a *greedy-continued coreset* and provide theoretical justification by showing it approximates the oracle coreset within a bounded gap.

We implement this principle through ContCore, which constructs the greedy-continued coreset via two steps: greedy expansion samples features from the new task that are maximally distant from the current coreset, and greedy consolidation enforces the memory constraint while preserving representativeness.

Our contributions are:

- We observe that continued greedy sampling effectively preserves representativeness under strict memory limits, enabling continual AD without unbounded memory growth. We provide theoretical justification by showing the resulting greedy-continued coreset approximates the oracle coreset within a bounded gap.
- We instantiate this principle in ContCore, which constructs a *greedy-continued coreset* that maintains representativeness of normal data across task transitions. ContCore achieves state-of-the-art performance across 11 task schedules on MVTecAD and VisA.
- We validate the approach in online continual AD settings, demonstrating robustness under stricter conditions where prior methods degrade significantly.

2 Related works

Anomaly detection (AD) Single-class AD methods [4, 8, 9, 21, 26, 33, 36] and multi-class [13, 14, 23, 31, 35] mainly use reconstruction [4, 9, 13, 14, 23, 31, 33, 35, 36] or memory-based methods [8, 26]. Reconstruction-based methods aim to reconstruct only normal images and measure anomaly scores based on the difference between the input and reconstruction. Moreover, multi-class methods aim to address the "identity shortcut" issue, which is caused by the decoder processing more various features, even similar to other classes' anomalies, as more classes are shown. However, these recent works lack any explicit mechanisms designed for a continual learning setup, suffering from catastrophic forgetting.

Continual learning The main challenge in continual learning is avoiding catastrophic forgetting, where a model's performance on past tasks degrades upon learning new ones. Two method types have been proposed to address this: regularization-based and replay-based methods. Regularization-based methods [2, 17, 34] use a loss function to prevent important

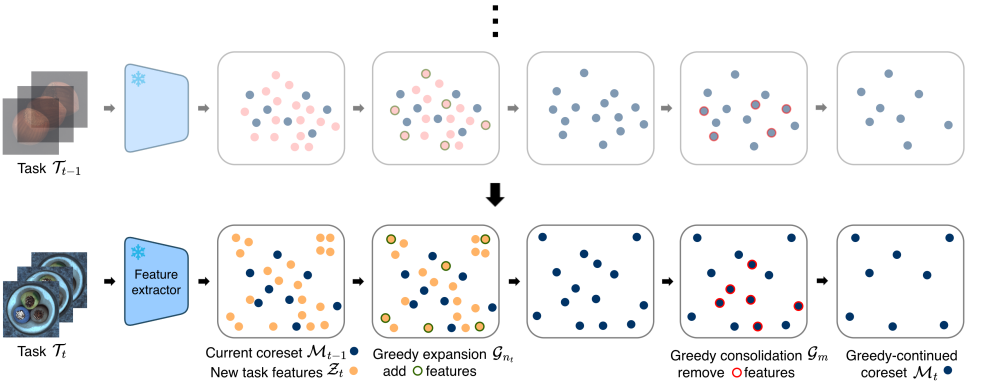


Figure 2: Memory-bounded continuation of greedy sampling via ContCore. Greedy expansion selects maximally distant features from new task features relative to the current coreset (green circles). Greedy consolidation enforces the memory constraint by selecting maximally separated features from the combined set (red circles), producing a greedy-continued coreset that preserves coverage across tasks.

parameters from old tasks from diluting, but often struggle to balance stability and plasticity. Replay-based methods [25], inspired by biological mechanisms [28], store a subset of past data in a memory buffer to interleave with new data during training. Though effective, they scale poorly as memory grows with task count, and their performance heavily depends on replay-sample quality [3, 24].

Continual anomaly detection (CAD) DNE [18] models each task features with the mean vector and covariance matrix of the assumed Gaussian distributions for fast, lightweight training and inference, but struggles with complex, multi-class tasks and lacks pixel-level AD ability. UCAD [20] uses a memory bank with key-prompt modules, continual prompt modulation, and Segment Anything Model (SAM)-based [16] pseudo-label contrastive learning. Its performance heavily relies on SAM's segmentation quality, which is used for training the prompts, where wrong segmentations would affect the representations of the prompt. Both DNE and UCAD struggle with multi-class tasks.

IUF [27] uses object-aware self-attention and semantic compression loss to preserve key semantics across tasks, but requires class labels, limiting it to supervised settings, unlike most multi-class AD methods [23, 31], which assume that the class information is not given. CDAD [19] employs a diffusion model with the anomaly-masked network, which enhances the conditioning mechanism, and memory-efficient gradient projection via SVD. However, it is computationally heavy with slow training and inference, as shown in Fig. 1(b). Both IUF and CDAD suffer from catastrophic forgetting when facing long-term continual task schedules.

3 Preliminaries

Greedy sampling is a subset-selection operation $\mathcal{G}$ that, given a set $\mathcal{Z}$ of latent embedding vectors as an input, returns a subset with maximized distances as follows:

$$\mathcal{G}_n(\mathcal{Z}) = \{z_i^* \in \mathcal{Z} : z_i^* = \underset{z \in \mathcal{Z}}{\arg\max} d(z, \mathcal{Z}_{i-1}^*)\} \tag{1}$$

for $2 \leq i \leq n$, where n is the sampling size and $\mathcal{Z}^*_{i-1} = \{z^*_1, \ldots, z^*_{i-1}\}$, where the set distance between a vector z and the set $\mathcal{Z}^*_{i-1}$, d, is defined as the minimum Euclidean distance to the vectors in the set. The initial point z^*_1 is chosen randomly or as one whose distance summation to all points is maximal. To simplify notations, we set $\mathcal{G}_n(\mathcal{Z}) = \mathcal{Z}$ if the data size is not greater than the sample size n (*i.e.*, $|\mathcal{Z}| \leq n$).

Continual anomaly detection Unsupervised CAD consists of a sequence $(\mathcal{T}_1, \ldots, \mathcal{T}_T)$ of T tasks, where *the number of tasks T is unknown*. Each task T_t is paired with its training data $\mathcal{D}_t$, which has only normal samples without task or class labels. At the t-th task, the model trainer has access to the model from the previous $(t-1)$-th task, but has no direct access to the data from the previous tasks.

4 Method

We implement ContCore, a memory-based anomaly detector that constructs a greedy-continued coreset via two steps: greedy expansion and greedy consolidation. Greedy expansion samples features from the new task that are maximally distant from the current coreset, and greedy consolidation enforces the memory constraint while preserving representativeness. The greedy-continued coreset is the key to continual AD: it maintains a representative summary of all observed normal data under fixed memory. The fact that the base set B is arbitrary enables greedy sampling to generalize across tasks by setting B as the current coreset.

4.1 Greedy sampling on a base set

The original greedy sampling $\mathcal{G}_n$ starts from an empty set $\emptyset$, and inductively collects the points that have maximal distance to the set collected at hand. The notion of greedy sampling can be extended by a base set B, where the greedy sampling starts from a non-empty base set B:

$$\mathcal{G}_n(\mathcal{Z}; B) = \{z^*_i \in \mathcal{Z} : z^*_i = \arg\max_{z \in \mathcal{Z}} d(z, \mathcal{Z}^*_{i-1} \cup B)\} \tag{2}$$

for $2 \leq i \leq n$. The initial point z^*_1 is given as $z^*_1 = \arg\max_{z \in \mathcal{Z}} d(z, B)$.

The extended greedy sampling differs from the original in that a point being collected is compared to both the sampled set $\{z^*_1, \ldots, z^*_{i-1}\}$ and the base set B. Hence, under this definition, the original greedy sampling is sampling with the empty base set, *i.e.*, $\mathcal{G}_n(\mathcal{Z})$ is equal to $\mathcal{G}_n(\mathcal{Z}; \emptyset)$. Note that the base set B can be any set of vectors of the same dimension. Depending on the choice of the base set, however, the final sampled output can differ.

4.2 Greedy-Continued Coreset for CAD

A naive application of greedy sampling in the continual learning scenario would result in a sampled set with size proportional to the number of tasks T. In this case, the memory and computation cost grow unboundedly, making it impractical for many tasks. Therefore, we propose ContCore, which constructs a greedy-continued coreset via memory-bounded continuation of greedy sampling as follows:

Given a set $\mathcal{Z}_t$ of patch embedding features from training set $\mathcal{D}_t$ of the t-th task, ContCore at t-th task is performed by

$$\mathcal{M}_t = \mathcal{G}_m(\mathcal{G}_{n_t}(\mathcal{Z}_t; \mathcal{M}_{t-1}) \cup \mathcal{M}_{t-1}), \tag{3}$$

where the memory $\mathcal{M}_t$ is inductively computed starting with $\mathcal{M}_0 = \emptyset$, subscript n_t on greedy expansion is the sampling size at t-th task and subscript m on greedy consolidation is the maximum coreset size. We denote the overall ContCore operation in Eq. (3) by $\mathcal{M}_t = \mathcal{G}_{m,n_t}(\mathcal{Z}_t;\, \mathcal{M}_{t-1})$.

Thus, ContCore in Eq. (3) consists of greedy expansion $\mathcal{G}_{n_t}(\cdot;\, \mathcal{M}_{t-1})$ and greedy consolidation $\mathcal{G}_m(\cdot)$. Fig. 2 shows that greedy expansion selects features from the new task that are maximally distant from the current coreset, while greedy consolidation enforces the memory constraint by selecting maximally separated features from the combined set. This two-step process constructs a greedy-continued coreset at each task transition. The constrained optimization criterion in Eq. (8) in Sec. 4.6 further clarifies the structure-preserving nature of ContCore.

Why greedy-continued coresets enable continual AD Greedy sampling selects maximally separated points, producing a compact yet representative summary of normal data [26]. This representativeness is essential for effective AD, since anomaly detection relies on measuring distance from normality. The central question for continual AD is whether representativeness can be maintained as tasks accumulate under bounded memory. We observe that continued greedy sampling effectively preserves representativeness across task transitions. Theorem 4.1 provides theoretical justification, showing the representational gap to the oracle coreset remains bounded.

4.3 Implementation in practice

Improving efficiency by approximation. The greedy consolidation step inevitably increases computation time. To mitigate this, we approximate greedy sampling using nearest neighbors. The key insight is that after greedy expansion, the combined features are already mostly representative. Thus, instead of strict costly greedy sampling, faster nearest neighbor can sufficiently measure distinctiveness.

Upon this intuition, during greedy consolidation, a fraction q of the samples are selected by nearest neighbors instead of full greedy sampling. Specifically, we compute the nearest neighbor distance d_i for each embedding z_i in the set $\widehat{\mathcal{M}}_t$ from greedy expansion and the current coreset;

$$d_i = d(z_i,\, \widehat{\mathcal{M}}_t \setminus \{z_i\}) \tag{4}$$

with $\widehat{\mathcal{M}}_t = \mathcal{G}_{n_t}(\mathcal{Z}_t;\, \mathcal{M}_{t-1}) \cup \mathcal{M}_{t-1}$. Then, by sorting d_i in descending order, we select the first fraction q of the embeddings with highest nearest distances.

4.4 Inference

To detect anomalies in the inference stage of the t-th task, we compute the anomaly score based on the nearest distance as shown in PatchCore [26]. In particular, given a test sample, we extract its patch embeddings $z^{(h,w)}$ for the (h,w)-th patch and compute the anomaly score of (h,w)-th patch by

$$a^{(h,\,w)} = d(z^{(h,\,w)},\, \mathcal{M}_t). \tag{5}$$

If the anomaly score for a particular patch is higher than a threshold, then that region is considered to contain an anomaly. If the maximum of anomaly scores across all patches is greater than a threshold, then the given test sample is considered to be an anomaly.

4.5 Theoretical analysis

We provide theoretical justification for our observation. The representational gap between the greedy-continued coreset and the oracle coreset (greedy sampling applied to all task data) remains bounded.

Theorem 4.1. *Suppose* $H(\mathcal{O}, \cup_{t=1}^{T} \mathcal{Z}_t) \leq \varepsilon_o$, *and assume* $H(S_t, \mathcal{Z}_t) \leq \varepsilon_t$, *and* $H(\mathcal{M}_t, S_t \cup \mathcal{M}_{t-1}) \leq \widehat{\varepsilon}_t$ *for all* $t = 1, \ldots, T$ *with* $S_1 = \mathcal{M}_1$, $\mathcal{M}_0 = \emptyset$, *and* $\widehat{\varepsilon}_0 = 0$, *where* H *is the Hausdorff distance,* $\mathcal{O} = \mathcal{G}_{n_0}(\cup_{t=1}^{T} \mathcal{Z}_t)$, $S_t = \mathcal{G}_{n_t}(\mathcal{Z}_t; \mathcal{M}_{t-1})$, *and* $\mathcal{M}_t = \mathcal{G}_{m_t}(S_t \cup \mathcal{M}_{t-1})$. *Then, we have*

$$H(\mathcal{O}, \mathcal{M}_T) \leq \varepsilon_o + \max_{1 \leq k \leq T} \left(\varepsilon_k + \sum_{j=k}^{T} \widehat{\varepsilon}_j \right) \tag{6}$$

for sufficiently large $n_o \leq |\cup_{t=1}^{T} \mathcal{Z}_t|$, $n_t \leq |\mathcal{Z}_t|$, *and* $m_t \leq |S_t \cup \mathcal{M}_{t-1}|$.

The theorem establishes that the greedy-continued coreset $\mathcal{M}_T$ incurs only bounded error relative to the oracle coreset $\mathcal{O}$. The error accumulates per-task but does not compound catastrophically, supporting our observation that continued greedy sampling preserves representativeness under memory constraints. With sufficient per-task sampling size n_t, the greedy-continued coreset maintains quality. We prove Theorem 4.1 in Sec. 7 in the supplement.

4.6 ContCore as Meta Learning

Given a set Z, *greedy sampling* samples instances by

$$z^*_{(n+1)} = \arg\max_{z \in Z} d(z, Z^*_n) \tag{7}$$

where d is the distance between a vector z and the set of vectors $Z^*_n = \{z^*_{(k)}\}_{k=1}^{n}$ constructed recursively. As the set distance between a vector z and the set $\mathcal{Z}^*_n$, d is defined as the minimum Euclidean distance to the vectors in the set. The initial point z_1 is chosen randomly or as one whose distance summation to all points is maximal.

Given a set Z and an arbitrary base set B, *greedy-continued coreset sampling* samples instances by

$$z^*_{(n+1)} = \arg\max_{z \in \widehat{Z}_m} d(z, Z^*_n)$$
$$\text{subject to } \widehat{Z}_m = \{\widehat{z}_{(j)}\}_{j=1}^{m}$$

where $\widehat{Z}_m = \{\widehat{z}_{(j)}\}_{j=1}^{m}$ is constructed recursively by

$$\widehat{z}_{(j+1)} = \arg\max_{z \in Z} d(z, \widehat{Z}_j \cup B). \tag{8}$$

with an arbitrary base set B. This constrained optimization criterion clearly indicates the meta-learning nature of ContCore as it applies greedy sampling on samples that are sampled by the greedy sampling.

Method	1 × 15 with 15 Steps		3 × 5 with 5 Steps		10 – 1 × 5 with 6 Steps		10 – 5 with 2 Steps		14 – 1 with 2 Steps	
	AUROC (↑)	FM (↓)	AUROC (↑)	FM (↓)	AUROC (↑)	FM (↓)	AUROC (↑)	FM (↓)	AUROC (↑)	FM (↓)
UniAD NeurIPS'22 [51]	77.4 / -	22.9 / -	81.3 / 88.7	7.4 / 10.6	76.6 / 82.3	21.1 / 17.3	86.7 / 91.5	14.9 / 10.6	85.7 / 89.6	18.3 / 13.3
UniAD + EWC PNAS'17 [27]	- / -	- / -	79.6 / 89.0	9.5 / 10.1	89.6 / 93.8	5.4 / 3.6	90.5 / 93.6	7.3 / 4.2	92.8 / 95.4	4.1 / 1.9
UniAD + SI PMLR'17 [54]	- / -	- / -	81.9 / 88.5	7.0 / 10.8	77.2 / 81.6	20.2 / 18.2	84.1 / 88.3	20.2 / 17.0	85.7 / 89.5	18.4 / 13.4
UniAD + MAS ECCV'18 [1]	- / -	- / -	81.5 / 89.0	7.2 / 10.2	77.9 / 82.0	19.5 / 17.7	86.8 / 91.0	14.9 / 11.6	85.8 / 89.6	18.1 / 13.3
UniAD + LVT CVPR'22 [31]	- / -	- / -	80.4 / 88.6	8.6 / 10.6	78.2 / 88.3	19.1 / 16.1	87.1 / 90.6	14.1 / 12.3	80.4 / 86.0	29.1 / 20.6
DNE ACMMM'22 [53]	87.3 / -	5.3 / -	84.3 / -	1.75 / -	79.2 / -	**0.4** / -	82.8 / -	2.1 / -	79.2 / -	0.1 / -
UCAD AAAI'24 [40]	93.0 / 83.3	1.0 / **-0.6**	84.8 / 90.1	10.3 / 9.2	91.2 / 94.0	6.3 / 3.0	88.7 / 93.1	5.2 / 3.7	93.8 / 95.7	1.8 / 0.3
IUF ECCV'24 [24]	71.8 / 79.2	5.1 / 6.6	84.2 / 91.1	10.0 / 8.4	94.2 / 95.1	3.2 / 1.0	92.2 / 94.4	9.3 / 4.3	96.0 / 96.3	1.0 / 0.6
CDAD CVPR'25 [33]	76.0 / 89.8	10.2 / 5.7	89.1 / 92.2	3.7 / 4.7	94.9 / 95.7	1.0 / 0.7	94.2 / 95.3	2.1 / 2.4	96.4 / 96.6	**-0.6** / **-0.04**
ContCore (ours)	**98.8 / 97.4**	**0.1** / 0.2	**98.2 / 96.9**	**0.6 / 0.5**	**98.9 / 96.4**	0.5 / **0.5**	**97.7 / 96.6**	**0.5 / 0.5**	**98.3 / 97.3**	0.0 / 0.1

Table 1: Task average AUROC (↑) and forgetting measurement (FM) (↓) on MVTecAD dataset with 5 different task schedules. Each cell shows the image-level/pixel-level AUROC. The best and second best results are marked in **bold** and underlined, respectively.

Method	1 × 12 with 12 Steps		8 – 1 × 4 with 5 Stepss		8 – 4 with 2 Steps		11 – 1 with 2 Steps	
	AUROC (↑)	FM (↓)	AUROC (↑)	FM (↓)	AUROC (↑)	FM (↓)	AUROC (↑)	FM (↓)
UniAD NeurIPS'22 [51]	63.9 / -	29.7 / -	72.2 / 90.8	16.6 / 9.2	78.1 / 94.0	14.7 / 8.4	75.0 / 792.1	22.4 / 11.4
UniAD + EWC PNAS'17 [27]	- / -	- / -	72.3 / 92.3	16.5 / 7.3	80.5 / 95.4	10.0 / 5.3	78.7 / 95.4	14.9 / 4.8
UniAD + SI PMLR'17 [54]	- / -	- / -	69.8 / 88.5	19.8 / 12.0	69.8 / 88.5	9.2 / 8.3	78.1 / 92.0	16.9 / 11.5
UniAD + MAS ECCV'18 [1]	- / -	- / -	72.1 / 90.6	16.7 / 9.4	72.1 / 90.6	14.1 / 8.4	75.4 / 91.8	21.5 / 11.9
UniAD + LVT CVPR'22 [31]	- / -	- / -	70.8 / 94.4	18.3 / 8.4	70.8 / 91.4	13.4 / 8.1	77.5 / 92.3	17.3 / 10.9
DNE (ViT-B/16) ACMMM'22 [53]	55.2 / -	11.4 / -	58.6 / -	10.2 / -	64.1 / -	6.1 / -	68.8 / -	1.9 / -
UCAD (ViT+SAM) AAAI'24 [40]	78.1 / 76.9	**0.0 / 0.6**	78.8 / 93.5	10.4 / 5.4	79.9 / 94.2	8.9 / 4.8	85.9 / 94.5	2.7 / 0.6
IUF (EfficientNet) ECCV'24 [24]	64.7 / 83.6	3.9 / 5.0	79.8 / 95.0	9.8 / 6.8	80.1 / 95.4	9.8 / 6.8	87.3 / **97.6**	2.4 / 1.8
CDAD (SDv1.5) CVPR'25 [33]	78.5 / **97.3**	11.1 / 6.0	83.4 / 95.8	3.8 / 1.5	85.3 / **97.1**	1.4 / **0.3**	88.3 / 97.2	**-1.3** / **-0.1**
ContCore (ours) (WRN50)	**93.7** / 97.2	0.7 / **0.6**	**96.3 / 97.2**	**0.7 / 0.5**	**93.8** / 96.5	**1.2** / 0.8	**96.1** / 97.7	0.4 / **-0.0**

Table 2: Task average AUROC (↑) and forgetting measurement (FM) (↓) on VisA dataset with 4 different task schedules. Each cell shows the image-level/pixel-level AUROC. The best and second best results are marked in **bold** and underlined, respectively.

5 Experiments

We conduct experiments to show the robustness of ContCore. First, we explain the setup of our experiments: dataset descriptions, continual task schedules, evaluation metrics, and hardware resources specs. The following section compares ContCore with the SOTA baselines using 11 different task schedulings. Finally, we conduct extensive ablation studies by analyzing the components of ContCore, present empirical values which support our theoretical bound, compare with PatchCore under the oracle setup, test ContCore on the online setup comparing with the SOTA baselines, and present the results of ContCore on Real-IAD [29] in the supplement.

5.1 Experimental setup

Datasets and task scheduling We conduct most of the experiments with the two widely used AD datasets: MVTecAD [5] and VisA [37]. For the continual task setup, we follow schedules introduced in IUF and UCAD. We denote a sequence of tasks with – and repeated task with × where each number represent the number of classes for each task. For example, 10 – 1 × 5 starts with a task of 10 classes followed by 5 tasks with 1 class each. IUF schedules MVTecAD with 14 – 1 (2 steps), 10 – 5 (2 steps), 3×5 (5 steps), and 10 – 1×5 (6 steps) and VisA with 11 – 1 (2 steps), 8 – 4 (2 steps), 8 – 1×4 (5 steps) continual tasks, and UCAD arranges MVTecAD and VisA into 1×15 (15 steps) and 1×12 (12 steps). Moreover, CDAD proposes cross-dataset setup which trains the whole MVTecAD dataset, followed by the

Method	MVTec→VisA		VisA→MVTec	
	AUROC (↑)	FM (↓)	AUROC (↑)	FM (↓)
IUF ECCV'24 [27]	82.4 / 92.5	11.1 / 4.9	74.3 / 89.7	16.7 / 5.5
CDAD CVPR'25 [19]	90.0 / 94.9	4.7 / **1.8**	84.2 / 94.1	6.9 / 3.5
ContCore (ours)	**94.3 / 96.6**	**0.6** / 0.3	**94.0 / 96.7**	**2.1 / 0.7**

Table 3: AUROC and FM results on cross-dataset CAD.

whole VisA and vice versa. The order of tasks are sorted by class names in the ascending alphabetical order.

These schedulings are designed to test the model's adaptability in various ways. 1×15 and 1×12 tests single-class incremental adaptability and verifies the resilience against catastrophic forgetting over a large number of tasks. 14 – 1 and 11 – 1 tests adding a single new class to a heavily pre-trained, multi-class base model mimicking the scenario where a deployed system receives a minor update. 10 – 5 and 8 – 4 simulates adding a large batch of new classes to the base model, simulating a major facility upgrade or bulk category addition in a single time. Meanwhile, 10 – 1 × 5 and 8 – 1 × 4 evaluates whether the complex base model can continuously adapt to a steady, ongoing stream of individual new tasks over time. Finally, 3 – 3 × 4 simulates multi-class incremental setup similar to the 1×15 schedule.

Models, hyperparameters, and hardware setup We follow the setup of PatchCore [26], where patch features are extracted from layer 2 and 3 of the WideResNet50 [32] backbone with sampling rate $p = 0.01$, coreset size $m = 20,000$ for MVTecAD, $m = 40,000$ for VisA. We use a single RTX 3090 GPU for the experiments.

Metrics We report the task average performance [22] after the final task T is trained. Specifically, the task average performance R_T is computed by

$$R_T = \frac{1}{|T|}\sum_{t=1}^{T} R_{T,t}, \tag{9}$$

where $R_{s,t}$ indicates the performance of the task t at step s, and T denotes the total number of steps.

We also report the forgetting measurement (FM) [15] metric, which measures the amount of information the model has lost throughout learning additional tasks. FM is computed by averaging the difference between the best performance during the previous tasks and the last performance after all tasks are trained, as shown below:

$$\mathrm{FM} = \frac{1}{T-1}\sum_{t=1}^{T-1} \max_{s\in\{1,\ldots,T-1\}} (R_{s,t} - R_{T,t}). \tag{10}$$

We report the task average performance R_T and FM of image-level Area Under the Receiver Operator Curve (AUROC) and pixel-level AUROC in Tab. 1 and 2. In addition, we also report the image-level Average Precision (AP), pixel-level Area Under the Per-Region-Overlap (AUPRO) in Tab. 11, 12, 13, and 14 in the supplement.

5.2 Comparison with the SOTA baselines

We compare ContCore with UniAD, UniAD applied with continual learning methods (EWC, SI, MAS, and LVT), and existing CAD methods (DNE, UCAD, IUF, and CDAD). As shown

Method	GFLOPS (↓)	Parameters (↓)	Test time (ms) (↓)	Train time (s) (↓)
DNE ACMMM'22 [18]	70.3	85.8 M	21.3	1233
UCAD AAAI'24 [20]	33.8	68.9 M	**15.3**	786
IUF ECCV'24 [27]	185.5	**28.5 M**	80.8	39192
CDAD CVPR'25 [19]	5628.0	11.0 G	2694.0	281434
ContCore (ours)	**9.24**	45.3M	39.1	**754**

Table 4: Comparing the efficiency of ContCore and the SOTA baselines in terms of computation, model size, inference speed, and training time.

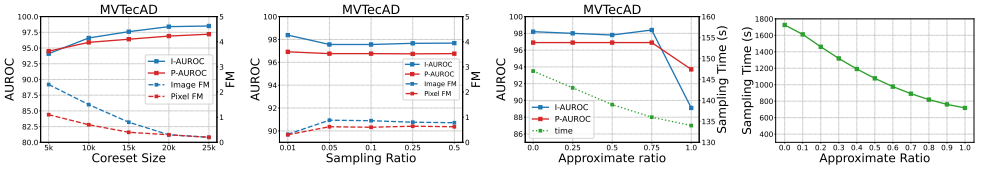
Task Schedule	Img / Pixel AUROC		Image AP / AUPRO	
	Performance (↑)	FM (↓)	Performance (↑)	FM (↓)
1 × 30	90.2 / 97.5	1.7 / 0.6	85.1 / 96.6	1.7 / 0.6
5 × 6	90.2 / 97.7	1.4 / 0.3	85 / 96.8	1.6 / 0.3
10 × 3	90.3 / 97.6	0.9 / 0.1	85 / 96.7	1.1 / 0.1
10–20	89.9 / 97.5	-0.3 / -0.1	84.6 / 96.6	-0.1 / -0.1
20–10	90.4 / 97.3	-0.2 / -0.3	85 / 96.4	-0.4 / -0.4
25–1×5	89.6 / 96.9	0.4 / 0.1	83 / 95.8	0.3 / 0.1
25–5	89.8 / 96.9	-0.2 / -0.4	83.8 / 95.8	-0.3 / -0.4
Task Schedule	Img / Pixel AUROC		Image AP / AUPRO	
	Performance (↑)	FM (↓)	Performance (↑)	FM (↓)
UniAD-Oracle	83.0 / 97.3	-	80.9 / 86.7	-
PatchCore-Oneclass	89.4 / -	-	-	-

Table 5: Performance of ContCore on Real-IAD dataset.

in Tab. 1 and 2, UniAD and the continual learning variants significantly show catastrophic forgetting based on the high FM and low AUROC while the CAD baselines show improved performance with lower FM and higher AUROC. However, existing CAD methods exhibit inherent trade-offs across task schedule and task complexity. Memory-based CAD methods (DNE, UCAD) minimize long-range forgetting from long-range task schedules (MVTecAD 1×15, VisA 1×14) with the inherent memories, however, struggle with complex tasks (MVTecAD 10 – 5, VisA 8 – 4, and VisA 8 – 1×4). Regularization-based methods resolve complex tasks well (MVTecAD 10 – 5, VisA 8 – 4, and VisA 8 – 1×4) and struggle from long-range task schedule as they lack explicit mechanism to keep information of long-range tasks. In contrast, ContCore maintain stable results across every task schedules showing robust performance for both image and pixel-level metrics as well as FM. In Tab. 3, ContCore surpass the previous baseline method CDAD in the cross dataset setup with a remarkable 8.9% margin in VisA→MVTecAD image-level AUROC. Finally, Fig. 1(b) compares the efficiency in terms of computation, number of parameter, training and inference time where ContCore show significant efficiency comparing with other methods.

5.3 Large scale datasets - Real-IAD

We evaluate ContCore on Real-IAD [29], which is a large-scale dataset. We increase the coreset size $m = 250,000$ and leave the hyperparameters same. We follow the official noise-free protocol of Real-IAD with 36,465 normal train images and 114,585 test images composed of 63,256 normal and 51,329 anomalous images with 30 classes in total. We test with various task schedules, as shown in Tab. 5 ranging from simple multiple task, complex few tasks, and mix of them. We compare with UniAD-oracle model, which is equivalent as UniAD trained with multi-class setup, and PatchCore-OneClass which are reported in the Real-IAD paper.



(a) (b) (c) (d)

Figure 3: Ablation results of ContCore on MVTecAD by varying coreset size m (a), sampling ratio p (b), approximation ratio q (c), and effect of approximation ratio q with large coreset size m and sampling ratio p (d).

Despite trained with continual task schedules, ContCore outperforms both UniAD-oracle and PatchCore-OneClass models in all reported metrics. The training time took around 65,000 seconds for each model.

5.4 Ablation study

We conduct extensive ablation study on the components of ContCore. Specifically, we analyze how the coreset size m, sampling ratio p, and approximate ratio q affect ContCore. Additionally, we show how different backbones affect ContCore. Experiments are done on MVTecAD where the result metrics are averaged through all task schedules used in Tab. 1 and 2. We provide full results of the experiments in the appendix along with experiments on VisA.

Coreset size m is the most important hyperparameters of ContCore, which equivalently works as the capacity of a neural network for network training as it follows the performance-speed tradeoff. Smaller m models are faster but less performative while larger m performs better with the tradeoff of inference speed. Therefore, selecting the proper value for m is crucial. Fig. 3(a) shows how the size of the coreset impacts ContCore where performance is proportional to coreset size. However, performance increasement starts to plateu when $m \geq 20000$ which is a good performance-efficiency trade-off point.

Sampling ratio p is crucial where Theorem 4.1 explains by presenting the importance of sufficient per-task sampling size for effective greedy continuation. It also determines the number of features to sample during greedy expansion, which affects greedy consolidation. As shown in Fig 3(b), the overall performance of varying the sampling ratio from $0.01 \leq p \leq 0.5$ doesn't severely fluctuate. Despite that high p values include more features during greedy expansion, greedy consolidation sorts out the most representative features. As low p values require less time and, we select $p = 0.01$. While our experiments were conducted on MVTecAD with sufficient number of samples, higher p might be required for datasets with less samples.

Approximation ratio q reduces the overall sampling time of ContCore by approximating greedy consolidation via nearest-neighbor. In Fig 3(c), we present the overall performance of varying the q. The overall performance remains stable in the interval $0 \leq q \leq 0.75$ but drops drastically at $q = 1$ where relying on nearest-neighbor ranking alone destroys the coverage structure. The sampling time decreases as q gets larger which is more significant in results on VisA in appendix. As the effect of q is more prominent with larger coreset and higher sampling ratios(more features from greedy expansion) we conduct an additional experiment on MVTecAD with $p = 0.2$ and $m = 100000$, and vary only q to measure its effect

Backbone	MVTecAD		VisA	
	AUROC (↑)	FM (↓)	AUROC (↑)	FM (↓)
DenseNet201	96.9 / 95.4	0.8 / 0.4	92.7 / 96.4	2.2 / 0.4
DINOv2	97.9 / 93.1	0.0 / 0.0	93.7 / 95.1	0.0 / 0.0
EfficientNet-b1	96.0 / 94.4	0.0 / 0.0	89.6 / 95.8	0.1 / 0.0
WideResNet101	98.5 / 97.2	0.4 / 0.3	94.4 / 97.3	1.2 / 0.5

Table 6: Ablation of ContCore using different backbones.

T	$H(\mathcal{O}, \mathcal{M}_T)$	ε_o	$\max_{1\leq k\leq T}\left(\varepsilon_k+\sum_{j=k}^{T}\widehat{\varepsilon}_j\right)$	Right hand side of Eq. (6)
2	4.87	2.28	4.37	6.65
3	4.72	2.22	5.99	8.21
4	4.59	2.13	7.54	9.67
5	4.50	2.04	9.02	11.06
6	4.44	2.08	10.66	12.74
7	4.35	2.08	12.27	14.35
8	4.35	2.21	13.95	16.16
9	4.31	2.07	15.22	17.29
10	4.36	2.19	16.85	19.04

Table 7: Theoretical tightness of bound shown for different number of tasks $\{2 \leq T \leq 10\}$ on MVTecAD classes (tasks) with 12 images from each and the value of each of the terms.

on sampling time. Please note that this experiment focuses on the difference in sampling time based on approximate ratio of ContCore regardless of the performance. Fig 3(d) shows that the sampling time drops down to half with large p, m values with $q = 0.75$.

Different backbones We validate the effect of varying the backbone and its effect on ContCore. Tab. 6 presents that the performance is correlated with the representation quality of extracted features. ContCore consistently performs well across various backbone architectures, showing that the effectiveness of ContCore originates from the greedy-continued coreset construction and the principle holds across different feature extractors, while absolute performance scales with feature quality.

5.5 Empirical tightness of theory

We empirically verify the inequality of Eq. (6) from Sec. 4.5 by considering the equality case where $\varepsilon_o = H(\mathcal{O}, \cup_{t=1}^{T}\mathcal{Z}_t)$, $\varepsilon_t = H(S_t, \mathcal{Z}_t)$, and $\widehat{\varepsilon}_t = H(\mathcal{M}_t, S_t \cup \mathcal{M}_{t-1})$. For this experiment, we build a toy dataset by choosing the first 10 classes of MVTecAD and sample 12 images from each class. This setup is due to the computation of the term $H(\mathcal{O}, \cup_{t=1}^{T}\mathcal{Z}_t)$ requires the pair-wise distance between every feature from the dataset and the sampled coreset, which is excessive. Then, we conduct single-class incremental setup for tasks $T \in \{2, 3, ..., 10\}$ with coreset size $m = 150 \times T$ and measure the value of each components in the inequality As shown in Tab. 7, the right hand side of Eq. (6) has higher values than $H(\mathcal{O}, \mathcal{M}_T)$ for every T.

5.6 Online learning

We evaluate ContCore under a more challenging setup, which we term "online CAD." Unlike previous CAD works that allow multiple data epochs and unrestricted batch size, this protocol introduces stricter, more realistic constraints. On top of the continual task schedules, we limit each data sample to be processed exactly once and limit the batch size to 1. We increase the sampling ratio $p = 0.1$ to acquire maximally informative patch features from single samples and leave the rest of hyperparameters the same. Tab. 8 shows that ContCore outperforms

Method	MVTecAD		VisA	
	AUROC (↑)	FM (↓)	AUROC (↑)	FM (↓)
DNE $_{\text{ACMMM'22}}$ [33]	69.1 / -	1.3 / -	65.0 / -	3.1 / -
UCAD $_{\text{AAAI'24}}$ [21]	84.2 / 76.2	**-1.1** / 0.3	74.6 / 75.8	**0.1** / 0.4
IUF $_{\text{ECCV'24}}$ [23]	71.0 / 80.5	0.4 / **0.1**	60.3 / 85.0	0.9 / 1.7
CDAD $_{\text{CVPR'25}}$ [13]	66.2 / 80.4	2.0 / **0.1**	58.9 / 87.5	0.2 / **0.3**
ContCore (ours)	**97.2 / 96.7**	1.5 / 0.9	**94.1 / 97.2**	1.3 / 0.8

Table 8: The results on online continual anomaly detection setup.

Method	MVTecAD		VisA	
	AUROC (↑)	FM (↓)	AUROC (↑)	FM (↓)
DNE $_{\text{ACMMM'22}}$ [33]	82.6 / -	1.9 / -	61.7 / -	7.4 / -
UCAD $_{\text{AAAI'24}}$ [21]	90.3 / 91.2	4.9 / 3.1	80.7 / 89.8	5.5 / 2.9
IUF $_{\text{ECCV'24}}$ [23]	87.7 / 91.2	5.7 / 4.2	78.0 / 92.9	6.5 / 5.1
CDAD $_{\text{CVPR'25}}$ [13]	90.1 / 93.9	3.3 / 2.7	83.9 / 96.9	3.8 / 1.9
PatchCore Oracle	**98.4 / 97.2**	**0.0 / 0.0**	**95.1 / 97.6**	**-0.1** / 0.0
PatchCore Oracle Online	91.2 / 90.1	**0.0 / 0.0**	77.9 / 85.2	0.0 / **-0.3**
ContCore (ours)	**98.4** / 96.9	0.3 / 0.3	95.0 / 97.2	0.8 / 0.5

Table 9: Comparison with PatchCore-Oracle.

the SOTA CAD methods under these demanding conditions with minor performance drop compared to the standard CAD protocol shown in Tab. 1 and 2.

5.7 ContCore and PatchCore

PatchCore involves a single offline greedy sampling from the whole feature set which is not inherently designed for continual learning and naively accumulating new features leads to memory overflow. To establish a comparable coreset based baseline, we propose PatchCore-Oracle where the total number of tasks is given as oracle information to uniformly divide the memory budget across all tasks. As shown in Tab. 9, ContCore performs comparably in a standard continual setting even against this advantaged baseline. However, in the online CAD setup where batch size is 1, PatchCore-Oracle's performance degrades due to the nondiversified feature set to sample at each step which shows the limitation: the lack of the mechanism refering to the current coreset for selecting new features from incoming tasks while ContCore, is explicitly designed for such dynamic updates.

Furthermore, our goal is to construct a coreset under a continual setup that approximates that of PatchCore which is sampled from all features. Therefore, we analyze the sampled coreset of ContCore, PatchCore, and a random sampling. Tab. 10 shows the results where ContCore has more overlapping samples, lower avearge minimum distance for each sample, lower sliced Wasserstein distance, and lower Hausdorff distance than random sampling, which quantitatively confirms that the coreset of ContCore is more similar to that of PatchCore-Oracle.

5.8 Qualitative analysis

We compare the segmentation qualities of CAD methods on MVTecAD with different task schedules. To visualize forgetting in the extreme case, we show the samples from the oldest task. As shown in Fig. 4, UCAD fails with complex tasks such as (14–1, 10–5) and IUF and CDAD fail for long-ranged tasks (1×14) while ContCore shows stable fine-grained localization quality. We provide more qualitative results of each task schedule in the appendix.

Sampling Method	Overlapping Samples (↑)	Average Minimum Distance (↓)	Sliced Wasserstein Distance (↓)	Hausdorff Distance (↓)
Random	356	1.17	0.0260	2.55
MGS	3933	1.04	0.0014	1.93

Table 10: Comparing of coresets of ContCore and random sampling with respect to PatchCore model.

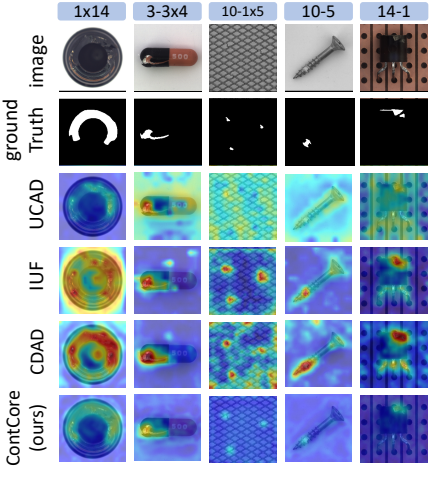


Figure 4: Anomaly segmentation qualitative comparison of ContCore and the SOTA methods by different task schedules on MVTecAD. All samples are from the first task.

6 Conclusion

Greedy sampling selects maximally separated points, producing a compact yet representative summary of normal data. Since anomaly detection relies on measuring distance from normality, a representative coreset is essential for reliable detection. The challenge for continual AD is maintaining representativeness as tasks accumulate under memory constraints. We observed that continued greedy sampling effectively preserves representativeness under strict memory limits, and provided theoretical justification showing the greedy-continued coreset approximates the oracle coreset within a bounded gap. We instantiated this principle in ContCore, which constructs a greedy-continued coreset through greedy expansion and consolidation. Unlike neural methods susceptible to catastrophic forgetting or naive coreset methods requiring unbounded memory, ContCore maintains fixed memory with theoretical guarantees. ContCore achieves state-of-the-art results across 11 continual task schedules on MVTecAD and VisA while maintaining efficient training and inference. The approach extends to online continual AD settings where prior methods fail significantly, demonstrating robustness beyond specific algorithmic choices.

Acknowledgments Yoon Gyo Jung and Octavia Camps were supported by ONR grant N00014-21-1-2431. Kuan-Chuan Peng was exclusively supported by Mitsubishi Electric Research Laboratories. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the ONR.

References

[1] Montdher Alabadi and Yuksel Celik. Anomaly detection for cyber-security based on convolution neural network: A survey. In *2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, pages 1–14. IEEE, 2020.

[2] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, pages 139–154, 2018.

[3] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *Advances in neural information processing systems*, 32, 2019.

[4] Kilian Batzner, Lars Heckler, and Rebecca König. EfficientAD: Accurate visual anomaly detection at millisecond-level latencies. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 128–138, 2024.

[5] Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger. MVTec AD–a comprehensive real-world dataset for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9592–9600, 2019.

[6] Dmitri Burago, Yuri Burago, Sergei Ivanov, et al. *A course in metric geometry*, volume 33. American Mathematical Society Providence, 2001.

[7] Yu Cai, Weiwen Zhang, Hao Chen, and Kwang-Ting Cheng. MedIAnomaly: A comparative study of anomaly detection in medical images. *Medical Image Analysis*, page 103500, 2025.

[8] Thomas Defard, Aleksandr Setkov, Angelique Loesch, and Romaric Audigier. PaDiM: A patch distribution modeling framework for anomaly detection and localization. In *International Conference on Pattern Recognition*, pages 475–489. Springer, 2021.

[9] Hanqiu Deng and Xingyu Li. Anomaly detection via reverse distillation from one-class embedding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9737–9746, 2022.

[10] Elena Deza and Michel Marie Deza. *Encyclopedia of distances*. Springer, 2009.

[11] Tharindu Fernando, Harshala Gammulle, Simon Denman, Sridha Sridharan, and Clinton Fookes. Deep learning for medical anomaly detection – A survey. *ACM Computing Surveys (CSUR)*, 54(7):1–37, 2021.

[12] Mikhael Gromov, Misha Katz, Pierre Pansu, and Stephen Semmes. *Metric structures for Riemannian and non-Riemannian spaces*, volume 152. Springer, 1999.

[13] Jia Guo, Shuai Lu, Weihang Zhang, Fang Chen, Huiqi Li, and Hongen Liao. Dinomaly: The less is more philosophy in multi-class unsupervised anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

[14] Haoyang He, Jiangning Zhang, Hongxu Chen, Xuhai Chen, Zhishan Li, Xu Chen, Yabiao Wang, Chengjie Wang, and Lei Xie. A diffusion-based framework for multi-class anomaly detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 8472–8480, 2024.

[15] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[16] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

[17] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[18] Wujin Li, Jiawei Zhan, Jinbao Wang, Bizhong Xia, Bin-Bin Gao, Jun Liu, Chengjie Wang, and Feng Zheng. Towards continual adaptation in industrial anomaly detection. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 2871–2880, 2022.

[19] Xiaofan Li, Xin Tan, Zhuo Chen, Zhizhong Zhang, Ruixin Zhang, Rizen Guo, Guanna Jiang, Yulong Chen, Yanyun Qu, Lizhuang Ma, et al. One-for-More: Continual diffusion model for anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

[20] Jiaqi Liu, Kai Wu, Qiang Nie, Ying Chen, Bin-Bin Gao, Yong Liu, Jinbao Wang, Chengjie Wang, and Feng Zheng. Unsupervised continual anomaly detection with contrastively-learned prompt. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 3639–3647, 2024.

[21] Zhikang Liu, Yiming Zhou, Yuansheng Xu, and Zilei Wang. SimpleNet: A simple network for image anomaly detection and localization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20402–20411, 2023.

[22] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

[23] Ruiying Lu, YuJie Wu, Long Tian, Dongsheng Wang, Bo Chen, Xiyang Liu, and Ruimin Hu. Hierarchical vector quantized transformer for multi-class unsupervised anomaly detection. *Advances in Neural Information Processing Systems*, 36, 2023.

[24] Lorenzo Pellegrini, Gabriele Graffieti, Vincenzo Lomonaco, and Davide Maltoni. Latent replay for real-time continual learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10203–10209. IEEE, 2020.

[25] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in neural information processing systems*, 32, 2019.

[26] Karsten Roth, Latha Pemula, Joaquin Zepeda, Bernhard Schölkopf, Thomas Brox, and Peter Gehler. Towards total recall in industrial anomaly detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14318–14328, 2022.

[27] Jiaqi Tang, Hao Lu, Xiaogang Xu, Ruizheng Wu, Sixing Hu, Tong Zhang, Tsz Wa Cheng, Ming Ge, Ying-Cong Chen, and Fugee Tsung. An incremental unified framework for small defect inspection. In *European Conference on Computer Vision*, pages 307–324. Springer, 2024.

[28] Gido M Van de Ven, Hava T Siegelmann, and Andreas S Tolias. Brain-inspired replay for continual learning with artificial neural networks. *Nature communications*, 11(1): 4069, 2020.

[29] Chengjie Wang, Wenbing Zhu, Bin-Bin Gao, Zhenye Gan, Jiangning Zhang, Zhihao Gu, Shuguang Qian, Mingang Chen, and Lizhuang Ma. Real-iad: A real-world multi-view dataset for benchmarking versatile industrial anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22883–22892, 2024.

[30] Zhen Wang, Liu Liu, Yiqun Duan, Yajing Kong, and Dacheng Tao. Continual learning with lifelong vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 171–181, 2022.

[31] Zhiyuan You, Lei Cui, Yujun Shen, Kai Yang, Xin Lu, Yu Zheng, and Xinyi Le. A unified model for multi-class anomaly detection. *Advances in Neural Information Processing Systems*, 35:4571–4584, 2022.

[32] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 87.1–87.12. BMVA Press, 2016.

[33] Vitjan Zavrtanik, Matej Kristan, and Danijel Skočaj. DRAEM – A discriminatively trained reconstruction embedding for surface anomaly detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8330–8339, 2021.

[34] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pages 3987–3995. PMLR, 2017.

[35] Ximiao Zhang, Min Xu, and Xiuzhuang Zhou. RealNet: A feature selection network with realistic synthetic anomaly for anomaly detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16699–16708, 2024.

[36] Xinyi Zhang, Naiqi Li, Jiawei Li, Tao Dai, Yong Jiang, and Shu-Tao Xia. Unsupervised surface anomaly detection with diffusion probabilistic model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6782–6791, 2023.

[37] Yang Zou, Jongheon Jeong, Latha Pemula, Dongqing Zhang, and Onkar Dabeer. SPot-the-Difference self-supervised pre-training for anomaly detection and segmentation. In *European Conference on Computer Vision*, pages 392–408. Springer, 2022.