machine learning & knowledge extraction

Article

Continual Semi-Supervised Malware Detection

Matthew Chin [†] and Roberto Corizzo [*,†]

Department of Computer Science, American University, Washington, DC 20016, USA; mc1038a@american.edu
* Correspondence: rcorizzo@american.edu
† These authors contributed equally to this work.

Abstract: Detecting malware has become extremely important with the increasing exposure of computational systems and mobile devices to online services. However, the rapidly evolving nature of malicious software makes this task particularly challenging. Despite the significant number of machine learning works for malware detection proposed in the last few years, limited interest has been devoted to continual learning approaches, which could allow models to showcase effective performance in challenging and dynamic scenarios while being computationally efficient. Moreover, most of the research works proposed thus far adopt a fully supervised setting, which relies on fully labelled data and appears to be impractical in a rapidly evolving malware landscape. In this paper, we address malware detection from a continual semi-supervised one-class learning perspective, which only requires normal/benign data and empowers models with a greater degree of flexibility, allowing them to detect multiple malware types with different morphology. Specifically, we assess the effectiveness of two replay strategies on anomaly detection models and analyze their performance in continual learning scenarios with three popular malware detection datasets (CIC-AndMal2017, CIC-MalMem-2022, and CIC-Evasive-PDFMal2022). Our evaluation shows that replay-based strategies can achieve competitive performance in terms of continual ROC-AUC with respect to the considered baselines and bring new perspectives and insights on this topic.

Keywords: continual learning; malware detection; semi-supervised learning; one-class learning; anomaly detection

Citation: Chin, M.; Corizzo, R. Continual Semi-Supervised Malware Detection. *Mach. Learn. Knowl. Extr.* **2024**, *6*, 2829–2854. https://doi.org/10.3390/make6040135

Academic Editor: Andreas Holzinger

Received: 30 October 2024
Revised: 4 December 2024
Accepted: 5 December 2024
Published: 10 December 2024

1. Introduction

Malware infections are on the rise, especially with the pervasiveness of IoT and mobile devices exposed to online services. A malware intends to alter, disrupt, or destroy software elements that are essential to the proper functioning of a computer system or device. It can steal sensitive information, gain persistent access to victim networks, as well as encrypt and send system information to the threat actor's command and control (C&C) centers.

Modern malware is coded using sophisticated techniques, which may make conventional security mechanisms (such as firewalls and anti-viruses) ineffective. For this reason, machine learning-based approaches for malware detection appear particularly attractive. However, one central issue pertains to the adequacy of existing solutions in real-world scenarios. While machine learning and deep learning have shown a promising detection performance [1–3], they may provide a limited robustness in a continuous scenario where malware morphology evolves over time, due to their reliance on labelled malware data, as well as their demanding computational requirements.

In this context, continual learning (CL) could provide a significant advance to machine learning-based malware detection, allowing models to effectively perform in challenging and evolving scenarios with limited computational requirements. The goal is to continuously adapt models to maintain and improve performance across all tasks encountered in a lifelong learning setup. Success is measured by the model's ability to balance plasticity (learning new tasks) and stability (retaining old tasks) using metrics such as accuracy on a sequence of tasks, forgetting rate, and forward/backward transfer. Learning strategies are task-driven and can be broadly categorized as regularization-based, replay-based, and architectural-based.

One of the most popular strategies is experience replay (ER), which maintains and revisits a subset of past data during training on new tasks. ER is also commonly adopted in reinforcement learning (RL) [4] and Q-learning [5] to support agents in learning from a mix of past experiences and approximate the value function or policy better [6,7]. In the context of CL, however, ER focuses on mitigating catastrophic forgetting, i.e., the tendency of neural networks to overwrite knowledge of previously learned tasks when learning new ones [8]. While RL and CL share the overlapping goal of adapting to new information over time, they are structurally different. In fact, RL focuses on training an agent that interacts with an environment to maximize cumulative rewards over time [9]. The agent learns policies for sequential decision-making tasks, often in stochastic or dynamic settings. On the other hand, CL focuses on training models on a sequence of tasks without forgetting knowledge of previously learned tasks. A structured breakdown of the differences for ER in CL and RL learning settings is presented in Table 1.

Table 1. Comparison of replay in reinforcement learning and continual learning settings.

	Reinforcement Learning (RL)	**Continual Learning (CL)**
Goal	Improve convergence	Mitigate catastrophic forgetting
Stored Data	Transitions or episodes	Task data or summaries
Sampling	Random or prioritized	Task or diversity-aware
Constraints	Less constrained by memory	Highly memory-constrained
Focus	Single task, dynamic exploration	Multi-task knowledge retention

Despite the high potential of CL for malware detection, this has been scarcely explored so far. In this paper, our focus is investigating malware detection from a continual, semi-supervised, one-class learning perspective. One-class learning allows us to train models using exclusively normal/benign data. This learning setting can empower models with a greater degree of flexibility and robustness when compared to a fully supervised setting. Specifically, (i) it overcomes the limitation of requiring labelled data for different malware classes, and (ii) it allows models to detect novel malware types, possibly unknown at training time. Moreover, in our work, a continual learning perspective is added to this setting, where models are challenged to incorporate emerging normal tasks/concepts while retaining knowledge of previously observed concepts and while being exposed to new malware instances. We assess the detection performance and the forgetting of a diverse set of anomaly detection models in two devised continual learning scenarios, adopting continual learning strategies. Our contribution is three-fold. Specifically, we do the following:

(i) Formulate a semi-supervised one-class continual malware detection workflow, which is essential for real-world applications but has received limited attention in previous research.
(ii) Devise two model-agnostic experience replay strategies that support anomaly detection models popular in conventional learning settings.
(iii) Conduct a comprehensive empirical evaluation and analysis involving seven diverse models, three real-world malware detection datasets, and two continual learning scenarios, assessing the effectiveness of experience replay strategies compared to proposed lower and upper-bound baselines.

2. Background

2.1. Malware Detection

Malware detection has become an urgent need in recent years. Operating systems are subject to run on a wide variety of devices, including desktops, servers, routers, security cameras, drones, etc., which exacerbates the difficulty of this task [10].

Machine learning-based approaches for malware detection have become popular in recent years. An LSTM-based approach to detect five malware families, i.e., Trojan, rootkit, backdoor, virus, and worm, was proposed by [11]. Deep recurrent neural networks have been explored in the context of Android malware detection [12]. A multi-head squeeze-and-excitation residual network to detect malware in Android apps has been devised [3], analyzing manifest file permissions, API calls, and hardware features. Specularly, a Chameleon–Hunter algorithm has been proposed [13] to address mobile malware in iOS devices and detect UI-based illicit activity threats.

Some approaches specialize in ransomware detection, spanning from conventional machine learning to deep learning models [14]. Recently, multi-classifier network-based [1], concept-drift-aware approaches [15], and multi-view feature intelligence [2] have been proposed. However, a significant pitfall of existing approaches stands in their fully supervised nature. Semi-supervised learning in this context can provide more flexibility, allowing models to generalize to unforeseeable classes of attacks that are not described by known patterns or profiles.

The authors in [16] propose a semi-supervised approach to identify malicious traffic by leveraging multimodal traffic characteristics, where two independent neural networks are adopted to learn sequence and topological features from the traffic. The model is trained using a joint strategy that minimizes both the reconstruction error from the autoencoder and the classification loss, allowing it to effectively utilize limited labeled data alongside a large amount of unlabeled data. A semi-supervised technique to detect Android malware from Android permissions and Application Programmer Interface (API) call logs is proposed in [17]. The ML technique is incorporated into an Android application to scan the installed applications and detect the corresponding levels of maliciousness with success. The work in [18] introduces a Semi-Supervised Vulnerability Detection (SSVD) mtehod that leverages the information gain of model parameters as the certainty of the correctness of pseudo-labels and prioritizes high-certainty pseudo-labeled code snippets as training data. The proposed approach incorporates a triplet loss to maximize the separation between vulnerable and non-vulnerable code snippets to better propagate labels from labeled code snippets to nearby unlabeled snippets.

An ensemble semi-supervised classification algorithm named Random Forest of Tensors (RFoT) is proposed in [19]. RFoT leverages tensor decomposition to extract intricate latent patterns from the data and combines multidimensional analysis with clustering to capture sample groupings within latent components, aiding in distinguishing between malware and benign-ware. On a similar thread, the authors in [20] propose a hierarchical semi-supervised algorithm (HNMFk) based on non-negative matrix factorization with automatic model selection. The method exploits the hierarchical structure of the malware data together with a semi-supervised setup, which enables the classification malware families under conditions of extreme class imbalance.

Despite the advantages provided by semi-supervised learning over fully supervised approaches, continual learning has rarely been considered for malware detection, resulting in a substantial gap. In fact, the unknown signatures of modern malware require models to be proactive rather than reactive. As a result, investigating models that can evolve and deal with dynamic scenarios where adaptation and knowledge retention are simultaneously considered is of paramount importance for future research. With this present paper, we attempt to fill this gap at the intersection of continual learning and semi-supervised malware detection, providing a new perspective on this problem.

2.2. Continual Learning

Also known as lifelong learning, it focuses on a continuous process in which a series of different tasks (or concepts) are presented to a machine learning method over time [8], challenging its adaptation and knowledge retention capabilities. In some cases, continual learning strategies draw inspiration from diverse disciplines, including biology and neuroscience [8]. Most common continual learning settings tackle image classification problems in task-incremental, class-incremental, and domain-incremental scenarios [21]. Emerging types of scenarios include online learning [22] and recurring tasks [23]. In anomaly detection, an emerging benchmark for continual learning with naturally occurring changes over time is proposed in [24].

Continual learning strategies in the literature can be loosely grouped into three main categories:

Regularization-based: These strategies introduce constraints on weight updates during the training process of neural network models. This goal could be realized by preventing or limiting updates to weights learned in previously observed tasks [25] or by freezing early layers in the model architecture to mitigate forgetting previous tasks while updating late layers to incorporate new tasks [26]. Another option is to use specialized losses to prevent drastic changes in already learned weights, as in EWC [27] and LWF [28].

Dynamic architectures: They adaptively adjust the model architecture during the learning process by expanding the network with new neurons or layers as they encounter new tasks [29,30]. Another option is to perform dynamic adaptation with pruning capabilities to keep model capacity under control by removing insignificant weights, as in PackNet [31] and WSN [32], as well as by introducing quantization capabilities [33]. Dynamic architecture strategies consider a neural network model as built upon independent sub-networks, each specialized in addressing a different task. Some approaches are also referred to as forget-free [33], i.e., not subject to forgetting. However, this behavior holds only under certain assumptions, such as the availability of task labels and unbounded capacity.

Replay-based: They focus on collecting a memory that represents a summarized version of data from previous tasks, which can be used for model updates. This goal can be achieved by selecting and storing salient samples in a buffer and replaying them during model updates [8].

Several experience replay strategies have been recently proposed in the literature. Examples include Retrospective Adversarial Replay (RAR) [34], which perturbs a buffered sample towards its nearest neighbors drawn from the current task in a latent representation space. Replaying such samples was shown to refine the boundary between previous and current tasks.

A brain-inspired variant of replay is proposed in [35], where hidden representations are replayed based on the neural network's own, context-modulated feedback connections.

Adaptive-experience replay (AdaER) [36] introduces a contextually-cued memory recall (C-CMR) strategy, which selectively replays memories that are most conflicting with the current input data in terms of both data and task. It also incorporates an entropy-balanced reservoir sampling (E-BRS) strategy to enhance the performance of the memory buffer by maximizing information entropy.

Uncertainty-Aware Sampling (UAS) [37] employs model and data uncertainties to select samples that are stable to the model and have low noise for rehearsal. The method leverages a dual Convolutional Neural Network (CNN) and Bayesian Neural Network (BNN) to continuously learn and consolidate knowledge.

The authors in [38] propose a mixup-based training approach to mitigate representation shifts by incorporating asymmetric mixup training into the replay method. The method selectively targets the old data stored in the memory buffer, deliberately excluding classes from the newly incoming data and enabling the model to learn new data while preserving the representation of the old data.

In general, experience replay approaches adopt selection strategies, which are devised to identify the most relevant samples for every task, with the intention to keep a compact replay buffer size and limit resource utilization [39]. Variants of these approaches have also been explored in distributed and collaborative learning settings [40].

Alternatively to collecting samples, artificial samples can be obtained through generative models [41], resulting in a lower memory footprint.

Despite the growing interest in continual learning, proposed strategies are mostly focused on training neural-based models in a supervised image classification setting. A few works explored continual learning from an anomaly detection perspective [42], although not strictly focused on malware detection. An exploration on continual learning for malware detection was recently conducted [43]. Albeit limited to supervised approaches, important takeaways of this study include an emphasis on continual learning methods as a more memory-efficient alternative to conventional machine learning and a thorough analysis that shows that the detection performance of available methods is still unsatisfactory.

In our work, we argue that continual semi-supervised learning could provide a significant advantage in malware detection settings.Table 2 compares different learning approaches considering the key properties considered in our study and highlighting the lack of a continual learning perspective in popular supervised and semi-supervised malware detection approaches.

An overview of our semi-supervised one-class continual malware detection workflow is graphically presented in Figure 1. Selected results in Figure 2 allow us to highlight this potential: learning new concepts while mitigating forgetting of previously learned concepts can lead to more robust models that exploit general knowledge to improve their performance on all concepts. A more detailed discussion on the adopted continual learning strategies for malware detection as well as the evaluation approaches used in our work is provided in the following sections.

Table 2. Comparison of surveyed research works on supervised, semi-supervised, and continual learning, based on three key aspects considered in our study. The first two groups are strictly focused on malware detection. The third group includes continual learning works focused on different downstream tasks such as image classification, reinforcement learning, and anomaly detection in different domains.

Approaches	Learn from Known Malware Patterns	Generalize to Unseen Malware types	Provide Adaptation and Knowledge Retention
MFMCNS [1]	X		
Qiu et al. [2]	X		
Zhu et al. [3]	X		
Deepflow [11]	X		
HaddadPajouh et al. [12]	X		
Lee et al. [13]	X		
Beaman et al. [14]	X		
FeSA [15]	X		
Liu et al. [16]	X	X	
Memon et al. [17]	X	X	
Yu et al. [18]	X	X	
Eren et al. [19]	X	X	
Eren et al. [20]	X	X	
Razavian et al. [25]			X
EWC [27]			X
LWF [28]			X
Diethe et al. [29]			X
Mignone et al. [30]			X
Packnet [31]			X
WSN [32]			X
Ada-Q-Packnet [33]			X
RAR [34]			X
Van De Ven et al. [35]			X
AdaER [36]			X
UAS [37]			X
MixER [38]			X
Buzzega et al. [39]			X
Faber et al. [40]			X
Shin et al. [41]			X
Rahman et al. [43]	X		X
Proposed	X	X	X

3. Methodology

3.1. Experience Replay Strategies

Our proposed continual malware detection workflow consists of alternate model training and evaluation phases. In the experimental scenario, training concepts (In this work, the terms *task* and *concept* are interchangeable. We refer to a *concept* as a self-consistent behavior of the normal class in an anomaly detection setting, as opposed to the conventional term *task* used in a classification setting.) $C_i = 1, 2, \ldots, N$ are presented to the model, which is updated accordingly using a replay strategy. In the evaluation phase, one-class models perform inference (anomaly detection) for all evaluation concepts $E_i = 1, 2, \ldots, k$, and their performance is evaluated.

In our work, we adopt commonly used anomaly detection models in a semi-supervised one-class learning setting that does not require knowledge of anomalous patterns for the model training phase. Therefore, models are trained on background data (normal) and extract anomaly scores on newly observed data. A known drawback of model updates in sequential learning is that models become prone to forgetting, which implies a decrease in the model performance for previously learned concepts as new knowledge is acquired. As a result, the challenge presented by a continual learning scenario is to incorporate new concepts in the model while preserving knowledge acquired from previously observed concepts. In our work, we leverage experience replay to consider stability and plasticity in our base models simultaneously. The replay buffer allows storing a summarized version of all concepts observed so far and leverages this knowledge to prevent forgetting while updating the model. Using a subset of data points allows for small storage requirements for the replay data. Notably, replay is model-agnostic, supporting any machine learning-based anomaly detection model.

For each training concept C_i, we store its summarized version R_i as part of the replay buffer. The complete experience replay buffer R can be defined as

$$R = \{R_i \,;\, \forall i \in \{1, 2, \ldots, k\}\}, \tag{1}$$

where k is the number of training concepts observed in the scenario. As a result, the replay buffer contains a subset of samples from each concept and allows us to maintain their representation throughout the whole scenario. Thus, replay-based model training takes into account all concepts observed so far. It is worth noting that data samples in the replay buffer are selected from the original data in each concept without replacement. As a result, each sample can be only included once in the replay buffer.

The replay buffer has a limited size controlled by budget B, which depends on the available storage resources. This budget is typically relatively small in comparison to observed data. In our work, the budget for each concept C_i is equal for all concepts in the scenario based on the total budget B. We devise two variants of experience replay given a concept C_i:

Random: selects data points at random from C_i according to budget B. This strategy gives equal weight to all experiences, regardless of their distance from the centroid of each concept. It favors the diversity of selected data samples, but it may increase the risk of concept overlap and noise as more concepts are observed by anomaly detection models, potentially impacting their ability to discriminate between normal data and anomalies.

Selective: selects data points closest to the centroid of C_i according to budget B. It is based on the assumption that experiences that lie closest to the centroid in each concept are the most relevant and should be preserved. It can potentially aid anomaly detection models in defining a restricted concept boundary that does not overlap with other concepts but may be subject to underfitting and limited exploitation of diversity.

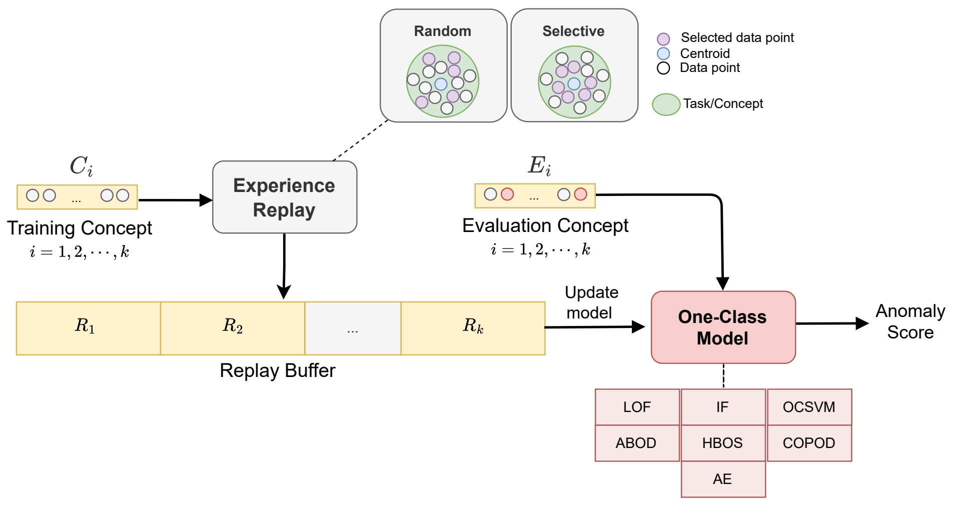


Figure 1. Semi-supervised one-class continual malware detection workflow consisting of model training and evaluation phases. Training concepts C_i contain exclusively normal data, whereas evaluation concepts E_i contain normal and anomalous data points. The experience replay component updates the replay buffer R each time a new training concept C_i is presented, according to the budget B. Two strategies are used: *Random* and *Selective*.

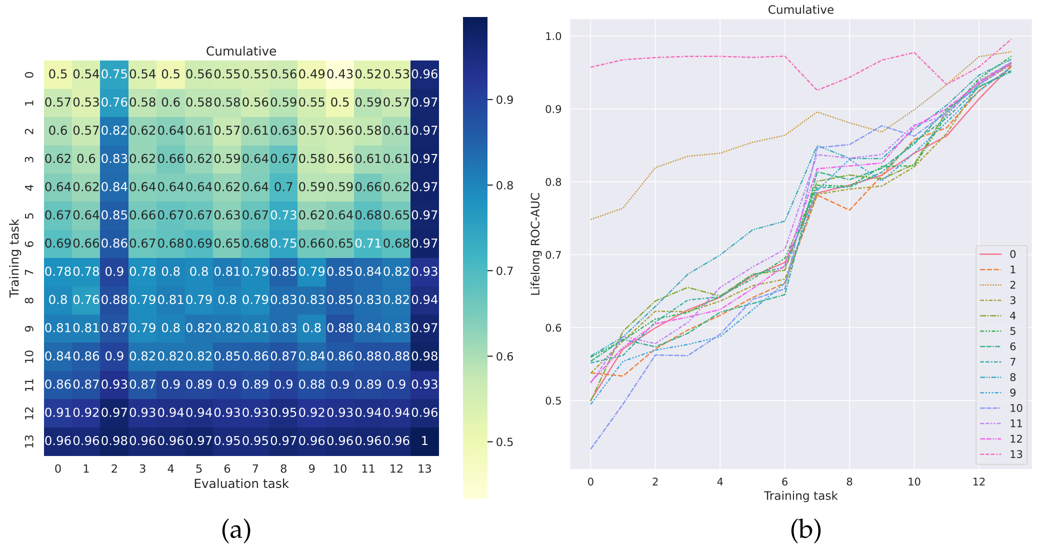


Figure 2. Example of model evaluation in continual malware detection: concept-level ROC-AUC as a heatmap (**a**) and as a line plot showing performance over time (**b**)—CIC-MalMem-2022 dataset—Strategy: Cumulative—Scenario: A—Model: LOF. The results show that learning a new concept without forgetting previous concepts leads to a more comprehensive and robust model, which results in a performance improvement on all other concepts.

Once the replay buffer is formed, the entire buffer (all samples) is used for model training independently from the chosen strategy. Intuitively, the proposed strategies allow models to adapt during the learning scenario by processing new data and adjusting the replay buffer to include a summary of past and new data. Performing training and model updates using data collected in this buffer has the effect of incorporating new relevant information without forgetting past knowledge, which is one of the most desired behaviors in continual learning.

3.2. Scenarios

To define continual learning scenarios, we exploit the characterization defined in [44] for continual/lifelong anomaly detection. The procedure leverages clustering functions to create normal and anomaly concepts (multiple self-consistent sets of data points) based on normal and anomaly data, respectively. In general, a concept corresponds to a new distribution, change in a performed activity, or a new state of the environment. For instance, in network traffic data, the entire normal/benign class can be thought of as a set of concepts describing legitimate traffic: web browsing, file transfer, and video streaming. Similarly, in our malware detection context, a normal concept can be regarded as a set of benign files with similar characteristics, where each file is described by a feature vector, and each concept is characterized by similar vectors, i.e., vectors with numeric feature values in a specific range.

Scenarios are built based upon the selection of the number of desired concepts k for normal (**N**) and anomaly (**A**) data from a given dataset and functions ϕ and γ to create multiple self-consistent sets of data points (normal and anomaly concepts, respectively) having common characteristics based on the entire classes.

The scenario creation procedure is described in more detail in Algorithm 1 [44]. First, we create concepts for the normal class through a concept creation function ϕ (Line 1). The function can consider any aspect (or feature values) that can delineate the boundaries of one concept. Second, we create concepts for the anomaly class through the anomalous concepts creation function γ (Line 2). Third, for each normal concept C_{N_i}, we select a corresponding anomaly concept C_{A_j} using a function λ (Line 3–5). The combination of C_{N_i} and C_{A_j} is a concept added to the continual learning scenario (Line 6). Each time a concept is built, the selected anomaly concept C_{A_j} is removed from the set of available anomaly concepts C_A (Line 7). The algorithm returns the resulting scenario as a sequence of concepts (Line 9), each of which may need to be separated into training and evaluation data depending on the learning settings, e.g., unsupervised or semi-supervised.

In this work, we leverage the k-Means clustering algorithm to implement ϕ and γ. The assignment function λ matches each normal concept with an anomaly concept, leading to a combined concept, which allows for model training (normal data) and evaluation (normal and anomaly data). We leverage two continual learning scenario types based on different choices of λ:

A: clustered anomaly concepts assigned to the closest normal concept.
C: clustered anomaly concepts assigned randomly to normal concepts.

In regards to experiments and the relationship between datasets, scenarios, and corresponding model performance, it is worth noting that the ordering of data samples in the replay buffer does not affect experiments, since samples used for model updates have all equal weight, and models have no notion of time to consider them differently. A different ordering of concepts in the learning scenario may, however, lead to a different model performance on single concepts. Emerging studies on curriculum learning in continual learning scenarios recently studied this phenomenon [45–47].

Algorithm 1: Scenario creation protocol [44]

```
Input: $c$ – Number of desired concepts
Input: $\mathbf{N}, \mathbf{A}$ – Normal/anomaly data
Input: $\phi$ – Concepts creation function for normal data
Input: $\gamma$ – Concepts creation function for anomalies
Input: $\lambda$ – Assignment function
1 $C_N \leftarrow \phi(\mathbf{N}, c)$                  // Create concepts $\{C_{N_0}, C_{N_1}, \dots, C_{N_c}\}$
2 $C_A \leftarrow \gamma(\mathbf{A}, c)$                  // Create concepts $\{C_{A_0}, C_{A_1}, \dots, C_{A_c}\}$
3 $T \leftarrow \emptyset$                                  // Result scenario
4 for $C_{N_i} \in C_N$ do
5     $j \leftarrow \lambda(C_A, C_{N_i})$                 // Match anomaly-normal concepts
6     $T \leftarrow T \cup (C_{N_i}, C_{A_j})$             // Add concepts to scenario
7     $C_A \leftarrow C_A - C_{A_j}$                      // Remove used anomaly concept
8 end
9 return $T$
```

4. Experiments

All experiments are run on a machine with an Intel Core i7-9750H CPU and 32 GB of RAM. The code and processed dataset of this paper are available at https://github.com/rcorizzo/cl-malware (accessed on 29 October 2024).

4.1. Datasets

The datasets adopted in our study were selected to satisfy goals of realism, scale, and diversity. First, datasets are representative and significant examples in real-world malware detection, as they are curated by the Canadian Institute for Cybersecurity (CIC), a comprehensive multidisciplinary training, research and development, and entrepreneurial unit that draws on the expertise of researchers in social sciences, business, computer science, engineering, law, and science. Second, they are large-scale, as they contain a large number of samples and malware types. Third, they cover diverse malware types spanning from Android applications to malware memory analysis and infections in PDF files.

CIC-MalMem-2022 [48]: This dataset is designed to test obfuscated malware detection methods through memory. The dataset was created to represent real-world malware types, including Spyware, Ransomware, and Trojan Horse. Debug mode was used for the memory dump process to reproduce typical use cases of users with applications in execution when malware attacks occur. The dataset contains a total of 58,596 instances (29,298 benign, 29,298 malicious).

CIC-Evasive-PDFMal2022 [49]: PDF is the most widely used document format due to its portability and reliability. Unfortunately, its popularity and advanced features have allowed attackers to misuse them to deliver a malicious payload. Authors collected a large number of malicious files from Contagio and VirusTotal and extracted 32 representative features, including 12 general and 25 structural. After processing, this dataset consists of 10,025 instances (4468 benign, 5557 malicious).

CIC-AndMal-2017 [50]: Android malware detection is becoming increasingly challenging for cybersecurity experts, due to the large number of variants released every day. This dataset includes 10,854 samples (4354 malware and 6500 benign) from several sources. Benign data are collected from the Google Play market. A total of 5000 of the collected apps (426 malware and 5065 benign) were installed on real devices. Malware samples are classified into four categories: Adware, Ransomware, Scareware, and SMS Malware.

4.2. Strategies

In addition to the proposed replay-based strategies, we adopt the following baselines in our experiments:

Naive: Models are updated each time a new training concept is available, without exploiting any smart continual learning strategy to provide both adaptation and knowledge retention. This strategy has the goal of simulating lower-bound performance provided by incremental learning, which is expected to gradually or catastrophically forget knowledge of previously presented concepts.

Cumulative: It accumulates all data from concepts observed so far, and it updates the model accordingly. It can be thought of as an upper bound adoptable when unlimited memory resources are available. Despite being unrealistic, it is useful to compare performances against resource-aware learning strategies.

In our experiments, we aim to compare the performance of two variants of experience replay (ER) with naive and cumulative.

4.3. Anomaly Detection Models

We adopt standard anomaly detection learning methods that are often used in unsupervised and semi-supervised settings. We selected seven methods, spanning a large diversity of modeling approaches:

LOF (Local Outlier Factor) [51]: simple but effective anomaly detection method based on a nearest-neighbor-based approach. LOF measures the deviation in the local density of data points with respect to their neighbors and compares the local density of data points with the average local density of nearest neighbors, considering it as an anomaly score.

Considering an object O_1, local reachability densities are compared with those of its neighbors as

$$\mathrm{LOF}_k(\mathrm{O}_1) := \frac{\Sigma_{\mathrm{O}_2 \in \mathrm{N}_k(\mathrm{O}_1)} \frac{\mathrm{lr}_k(\mathrm{O}_2)}{\mathrm{rr}_k(\mathrm{O}_1)}}{|\mathrm{N}_k(\mathrm{O}_1)|} = \frac{\Sigma_{\mathrm{O}_2 \in \mathrm{N}_k(\mathrm{O}_1)} \mathrm{lr}_k(\mathrm{O}_2)}{|\mathrm{N}_k(\mathrm{O}_1)| \cdot \mathrm{lr}_k(\mathrm{O}_1)}, \tag{2}$$

which denotes the average local reachability density of neighbors divided by the local reachability density of the object. If the score is less than 1, the object has a higher density than its neighbors, indicating it is an inlier. A score greater than 1 suggests a lower density than its neighbors, classifying the object as an outlier.

LOF measures the deviation in the local density of data objects with respect to their neighbors. The prediction score returned by LOF for a particular data object is obtained as the ratio between its local density and the average local density of the nearest neighbors.

Considering the k-distance k_D as the distance of object O_1 from its k-th nearest neighbors, the notion of reachability distance can be defined as

$$RD_k(\mathrm{O}_1, \mathrm{O}_2) = \max\{k_D(\mathrm{O}_2), \mathrm{d}(\mathrm{O}_1, \mathrm{O}_2)\}.$$

Based on this definition, objects that belong to the k nearest neighbors of O_2 are considered to be equally distant. The local reachability density of an object O_1 defined as $\mathrm{lr}_k(\mathrm{O}_2)$ is the inverse of the average reachability distance of the object O_1 from its neighbors.

A value of approximately 1 indicates that the object is similar to its neighbors (and thus not an outlier). A value below 1 indicates a denser region (i.e., an inlier), whereas values significantly larger than 1 indicate outliers. A value lower than 1 expresses a higher density than neighbors (inlier), while a value greater than 1 denotes a lower density than neighbors (outlier).

IF (Isolation Forests) [52]: The tree and ensemble-based modeling capabilities of IF showed great potential in anomaly detection works. The method computes an isolation score for every data object. The average distance from the tree's root to the leaf associated with the data object (corresponding to the number of splits required to reach the object) is used to predict the anomaly score. Considering that more pronounced variations in values equal shorter paths in the tree, Isolation Forest uses this information to distinguish an abnormal data object from the rest. The anomaly score is defined as

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}}, \tag{3}$$

where $h(x)$ is the path length of the object x, $c(n)$ is the average path length of an unsuccessful search in the Binary Search Tree, and n is the number of external nodes. An anomaly score close to 1 indicates that an object has a high chance of being an anomaly, while scores that are smaller than 0.5 are indicative of a regular (or non-anomaly) data object. *OCSVM* (One-Class Support Vector Machines) [53]: A well-recognized hyperplane-based method in anomaly detection is due to its ability to detect out-of-distribution data points considering their distance from the decision boundary. It conceptually operates in a similar way to Support Vector Machines, which identify a hyperplane to separate data instances from two classes. However, the one-class learning counterpart uses a hyperplane to encompass all of the background data instances (human essays). Solving the OCSVM optimization problem corresponds to solving the dual quadratic programming problem:

$$\min_{\alpha} \frac{1}{2} \sum_{ij} \alpha_i \alpha_j K(x_i, x_j)$$

subject to the constraints $0 \leq \alpha_i \leq \frac{1}{\nu l}$ and $\sum_i \alpha_i = 1$, where α_i is the weight for the instance i, vectors with non-zero weights are defined as *support vectors* and determine the optimal hyperplane, ν is a parameter that represents a trade-off between the distance of the hyperplane from the origin and the number of instances covered by the hyperplane, l is the number of instances in training data, and $K(x_i, x_j)$ is the kernel function. Leveraging the kernel function to project input vectors into a feature space allows for nonlinear decision boundaries. Specifically, a feature map can be defined as

$$\phi : X \to \mathbb{R}^N,$$

where ϕ maps training vectors from the input space X to a dimensional feature space, and the kernel function is defined as

$$K(x, y) = \langle \phi(x), \phi(y) \rangle$$

The adoption of kernel values $K(x, y)$ allows one to avoid the explicit computation of feature vectors, with a great improvement in computational efficiency. Common kernels include Linear, Radial Basis Function (RBF), Polynomial, and Sigmoid.

Following the training phase, the OCSVM-learned hyperplane can categorize a new data instance (essay) as regular/normal (human) or different/anomaly (AI-generated) with regard to the training data distribution, based on its geometric location within the decision boundary.

COPOD (Copula-based Outlier-Detection) [54]: Copulas are defined as multi-variate cumulative distribution functions with a uniform marginal probability distribution for each feature. Copula-based anomaly detection predicts the degree of "extremeness" of data samples based on tail probabilities. COPOD is known as a robust parameterless one-class learning method.

HBOS (Histogram-based Outlier Score) [55]: a statistical approach that assumes independence across features. It generates a histogram for each feature and multiplies the inverse height of the bins, assessing the density of all features. Even though feature relationships are ignored (i.e., the method assumes feature independence), this simplification allows the method to converge quickly. HBOS builds histograms in two different modalities: *(i)* static bin sizes and a preset bin width, and *(ii)* dynamic bins with a close-to-equal number of bins. This approach, similar in nature to the Naive Bayes algorithm, provides a quick and effective way to identify anomalies.

ABOD (Angle–Base Outlier Detection) [56] computes the variance of weighted cosine scores between data points and their neighbors and considers them as the anomaly score. It is known for efficiently identifying outliers in high-dimensional datasets and for being robust to false positives.

Auto-Encoder (AE) [57]: Auto-encoders compress data into a simplified latent representation and then reconstruct the original data from it. Auto-encoders present a mirrored structure, i.e., the number of layers in the decoding phase matches the encoding phase. The final output layer has the same size as the input layer and reconstructs the input data. A common approach is to train an auto-encoder on normal data and then detect anomalies by identifying data that the model cannot reconstruct well.

Auto-encoders consist of a model architecture with multiple hidden layers, where the output of the i-th hidden layer is the i-th encoding level of the input data. Typically, the model presents a mirrored architecture, where the number of layers for the decoding stage corresponds to the number of layers defined for the encoding stage. Finally, the last layer of the auto-encoder has the same size as the input layer and returns the reconstructed input representation by means of the decoded stage.

After training, the model is used to reconstruct unseen data objects. When a high reconstruction error is observed for a new data object, this is assumed to belong to a different class than that observed during training (normal/benign), i.e., it is classified as malicious. Otherwise, it is recognized to belong to the training data distribution, i.e., it is classified as benign.

4.4. Model Evaluation

We devise an evaluation protocol to assess model performance for any one-class learning model across all concepts in our continual learning scenarios. We initialize a matrix R to store anomaly detection results for specific tasks/concepts. The protocol iterates over training concepts and trains/updates the model accordingly. Following up, models are evaluated on all testing concepts, i.e., previous, current, and future concepts. In the resulting matrix R, entries $R_{i,j}$ contain the ROC–AUC metric of the model on concept j after learning concept i. The resulting matrix R can then be used to compute continual learning metrics directly, such as backward and forward transfer, which allow us to assess model behavior more precisely than standard performance metrics, taking into account model performance on previous, current, and future concepts:

Continual ROC–AUC ($\mathsf{ROC-AUC}$): Inspired by [58], we propose a variant of ROC-AUC that assesses each model's performance on all concepts after learning every new concept:

$$\mathsf{ROC-AUC} = \frac{\sum_{i\geq j}^{N} R_{i,j}}{\frac{N(N+1)}{2}} \tag{4}$$

This metric is calculated considering previously learned concepts, including the current concept (corresponding to averaging over $\frac{N(N+1)}{2}$ lower triangular entries). One major advantage of ROC–AUC over threshold-dependent metrics such as F1–Score is that it evaluates the model's performance in a more detailed manner, considering all possible detection thresholds. However, ROC–AUC may be swapped with other metrics of choice in our protocol.

Backward transfer for ROC–AUC (BWT): It measures the impact of learning new concepts on the model's performance of all previously learned concepts. Negative BWT indicates that the model presents a degree of forgetting (a strongly negative BWT value is also known as catastrophic forgetting). Positive BWT signals that learning new concepts benefits models' performance on previously observed concepts:

$$\mathsf{BWT} = \frac{\sum_{i=2}^{N} \sum_{j=1}^{i-1} R_{i,j} - R_{j,j}}{\frac{N(N-1)}{2}} \tag{5}$$

Forward transfer for ROC–AUC (FWT): It measures the impact of learning each concept on the model's performance on future concepts, i.e., zero-shot model performance on future concepts:

$$\mathsf{FWT} = \frac{\sum_{i<j}^{N} R_{i,j}}{\frac{N(N-1)}{2}} \tag{6}$$

4.5. Results and Discussion

In this subsection, we analyze the extracted results. The main research question we aim to address is whether the adoption of experience replay is beneficial for the performance of one-class learning models in continual malware detection scenarios. We recall that a replay strategy is in charge of selecting and maintaining a buffer of data samples from previously observed concepts, and it is limited in size by a budget, which represents a subset of available data points for concepts. As the model is provided with a new task (concept), the buffer is updated in order to capture knowledge of all concepts presented in the scenario so far. This behavior should, in principle, equip models with both adaptation and knowledge retention capabilities [39]. As a result, the model should be able to provide a satisfactory performance on all concepts, without a significant degree of forgetting for any of them.

Results in Tables 3–5 show the performance achieved by all models on all datasets (the best results for each strategy in both scenarios are marked in bold—all reported ER configurations are with a budget of 0.15, i.e., 15% of data for each concept is stored in the replay buffer). We compare replay strategies in terms of anomaly detection performance (ROC-AUC), backward transfer (BWT), and forward transfer (FWT) with respect to the naive (lower bound) and cumulative (upper bound) strategies. Results show that the Replay strategy (ER) brings various degrees of improvement. In some cases, the improvement in ROC-AUC over naive is non-existent, as seen in Table 3 (Scenario: A) for ER (Random) with IF as the base model, i.e., the ROC-AUC is the same (0.511). In many other cases, the improvement is significant. For example, focusing on ABOD, we can observe in Table 5 that ER (Random) increases ROC-AUC from 0.613 to 0.674 (Scenario: A) and from 0.669 to 0.73 (Scenario: C). There are also cases where the improvement is remarkable. This is the case for LOF in Table 3 (from 0.621 to 0.662 in Scenario: A and from 0.586 to 0.654 in Scenario: C) and for ABOD in Table 4 (from 0.591 to 0.864 in Scenario: A and from 0.559 to 0.821 in Scenario: C).

Our results emphasize that some anomaly detection models (LOF, ABOD) are more robust than others in terms of detection performance and resistance to forgetting. One explanation for this phenomenon is that LOF and ABOD are more suitable for datasets with complex and irregular distributions, as they can capture local density variations effectively. In contrast, IF and OCSVM tend to struggle when the data distribution is highly irregular or presents overlapping clusters and are more sensitive to hyperparameter tuning. ABOD and LOF also tend to be more effective at detecting local anomalies or clusters of anomalies, compared to IF and OCSVM.

Table 3. Results: CIC-MalMem-2022 dataset: forward transfer (FWT), backward transfer (BWT), and ROC-AUC (AUC) with all models and strategies, including two experience replay (ER) variants in two scenarios (A,C). For all metrics, higher values are indicative of a better performance: zero-shot model capabilities on future concepts (FWT), models' ability to avoid forgetting and improve on previously learned concepts (BWT), and anomaly detection performance (AUC). Values in bold highlight the best model for each strategy according to AUC.

Scenario: A												
Strategy		**OCSVM**			**IF**			**LOF**			**COPOD**	
	FWT	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**
Naive	0.398	0.004	0.458	0.46	0.003	0.511	0.561	0.049	**0.621**	0.364	0.006	0.44
Cumulative	0.395	0.0	0.455	0.462	0.0	0.51	0.669	0.122	**0.783**	0.383	0.006	0.469
ER (Random)	0.4	0.001	0.459	0.461	0.001	0.511	0.594	0.059	**0.659**	0.367	0.01	0.447
ER (Selective)	0.39	−0.005	0.459	0.44	−0.005	0.485	0.586	0.072	**0.662**	0.359	0.008	0.438
Strategy		**HBOS**			**ABOD**			**AE**				
	FWT	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**			
Naive	0.455	0.001	0.511	0.624	−0.04	0.586	0.586	0.0	0.569			
Cumulative	0.455	−0.001	0.511	0.315	−0.049	0.289	0.587	−0.001	0.569			
ER (Random)	0.454	0.001	0.511	0.549	−0.2	0.419	0.588	0.0	0.571			
ER (Selective)	0.444	−0.004	0.498	0.591	−0.053	0.546	0.588	−0.0	0.571			
Scenario: C												
Strategy		**OCSVM**			**IF**			**LOF**			**COPOD**	
	FWT	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**
Naive	0.383	0.001	0.472	0.498	0.003	0.485	0.61	0.025	0.586	0.368	0.001	0.444
Cumulative	0.381	0.001	0.475	0.495	0.002	0.482	0.698	0.16	**0.822**	0.405	0.004	0.455
ER (Random)	0.381	0.001	0.474	0.496	0.001	0.483	0.632	0.061	**0.648**	0.382	0.002	0.45
ER (Selective)	0.409	−0.008	0.444	0.465	−0.011	0.45	0.63	0.064	**0.654**	0.384	−0.005	0.435
Strategy		**HBOS**			**ABOD**			**AE**				
	FWT	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**			
Naive	0.495	−0.001	0.478	0.629	−0.009	0.591	0.562	0.0	**0.592**			
Cumulative	0.495	−0.0	0.478	0.323	−0.071	0.283	0.562	−0.0	0.591			
ER (Random)	0.495	−0.001	0.479	0.512	−0.265	0.286	0.564	0.001	0.595			
ER (Selective)	0.474	−0.012	0.458	0.59	−0.029	0.553	0.566	0.0	0.594			

Table 4. Results: CIC-Evasive-PDFMal2022 dataset: forward transfer (FWT), backward transfer (BWT), and ROC-AUC with all models and strategies, including two experience replay (ER) variants in two scenarios (A,C). For all metrics, higher values are indicative of a better performance: zero-shot model capabilities on future concepts (FWT), models' ability to avoid forgetting and improve on previously learned concepts (BWT), and anomaly detection performance (AUC). Values in bold highlight the best model for each strategy according to AUC.

Scenario: A

Strategy		**OCSVM**			**IF**			**LOF**			**COPOD**	
	FWT	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**
Naive	0.187	−0.356	0.508	0.566	−0.138	0.739	0.205	−0.401	0.484	0.505	−0.204	0.559
Cumulative	0.18	−0.154	0.41	0.549	−0.061	0.699	0.173	−0.014	0.733	0.515	0.054	0.698
ER (Random)	0.186	−0.266	0.383	0.564	−0.111	0.704	0.184	−0.022	0.712	0.51	−0.022	0.642
ER (Selective)	0.187	−0.178	0.507	0.565	−0.107	0.721	0.184	−0.123	0.66	0.511	−0.013	0.644

Strategy		**HBOS**			**ABOD**			**AE**	
	FWT	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**
Naive	0.56	−0.077	**0.784**	0.249	−0.512	0.591	0.27	−0.17	0.507
Cumulative	0.558	0.006	0.771	0.214	−0.016	**0.909**	0.315	0.043	0.573
ER (Random)	0.557	−0.002	0.776	0.221	−0.084	**0.864**	0.376	−0.049	0.54
ER (Selective)	0.558	−0.008	0.779	0.232	−0.172	**0.814**	0.315	−0.028	0.578

Scenario: C

Strategy		**OCSVM**			**IF**			**LOF**			**COPOD**	
	FWT	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**
Naive	0.37	−0.649	0.534	0.575	−0.374	0.703	0.439	−0.541	0.492	0.574	−0.139	0.596
Cumulative	0.301	−0.199	0.399	0.651	−0.024	0.7	0.392	0.002	0.749	0.606	−0.003	0.671
ER (Random)	0.373	−0.419	0.538	0.652	−0.138	0.728	0.48	0.067	0.781	0.594	−0.07	0.631
ER (Selective)	0.375	−0.383	0.565	0.664	−0.138	0.733	0.405	−0.068	0.761	0.592	−0.067	0.633

Strategy		**HBOS**			**ABOD**			**AE**	
	FWT	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**
Naive	0.656	−0.273	**0.759**	0.395	−0.626	0.559	0.568	−0.396	0.528
Cumulative	0.728	−0.008	0.775	0.411	−0.031	**0.85**	0.397	−0.013	0.581
ER (Random)	0.708	−0.063	0.773	0.488	−0.09	**0.821**	0.471	−0.27	0.541
ER (Selective)	0.706	−0.075	0.753	0.422	−0.155	**0.794**	0.481	−0.169	0.566

Overall, we observe that at least one of the replay strategies presents a performance that is equal to or higher than naive in 34 out of 42 configurations (7 models, 3 datasets, 2 scenarios). Ablation results (see Appendix A) reveal the performance achieved with different replay budgets $(0.05, 0.1, 0.15, 0.2)$. One interesting result is that model performance does not increase dramatically with larger budgets, which represents a promising direction for lightweight replay strategies in settings with limited memory requirements. It is also worth noting that, in our experiments, model performance did not increase significantly with budget values higher than 0.2.

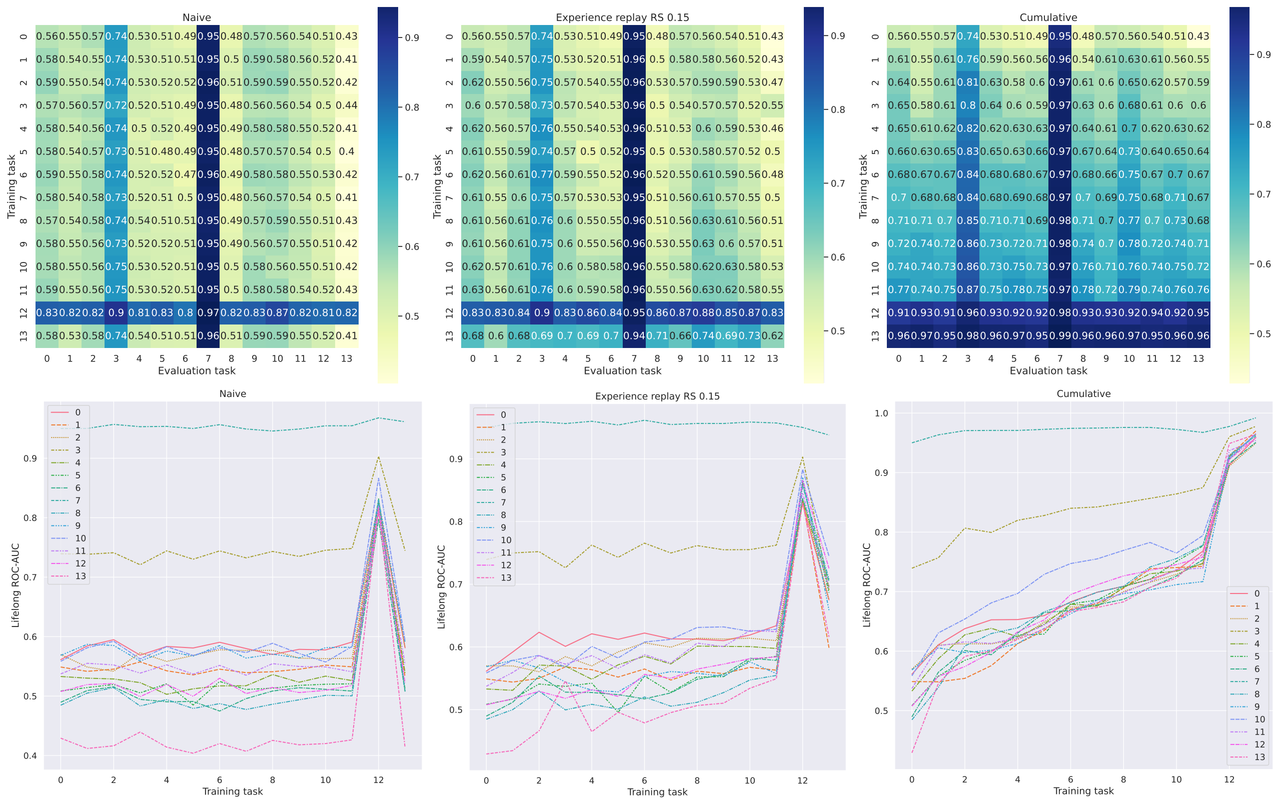
Another important consideration is that ROC-AUC generally improves symmetrically with BWT, which means that replay strategies are beneficial for both the anomaly detection performance and the ability for one-class models to reuse knowledge across different tasks. It should be noted that in many of the cases in which replay does not provide an improvement over naive, we observe that all strategies perform poorly for the specific base model, as noted by their poor scores (close to random performance). This phenomenon suggests that the poor performance may not depend on the replay strategy but on predictive models being fundamentally inadequate to deal with the complexity presented by specific datasets or continual learning scenarios.

We note that there is still a gap between results obtained with the replay strategies and the simulated cumulative strategy considered as the upper-bound. We can observe that cumulative presents the best results in 21 out of 42 cases. This result demonstrates that it is possible to achieve a higher performance on the considered malware detection datasets and scenarios and signals the need for more sophisticated continual learning strategies to address this gap.

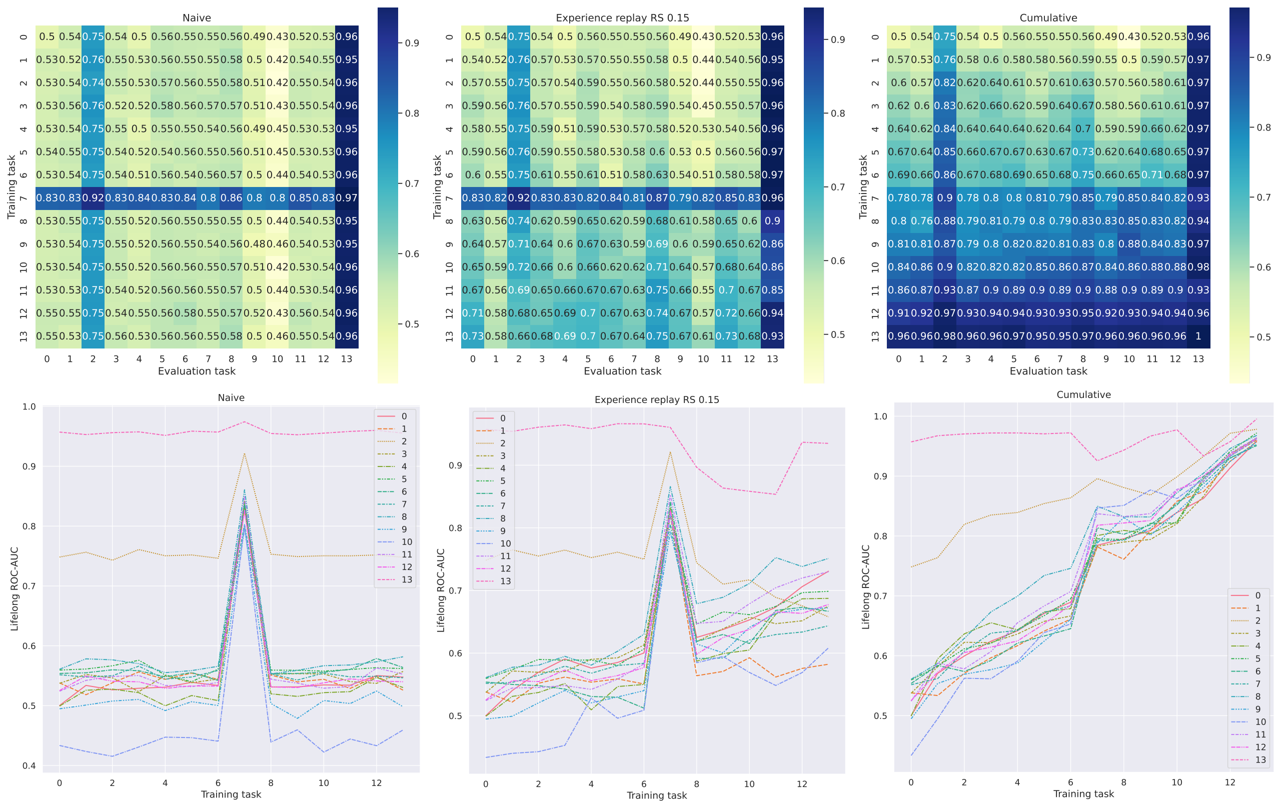
Table 5. Results: CIC-AndMal2017 dataset: forward transfer (FWT), backward transfer (BWT), and ROC-AUC with all models and strategies, including two experience replay (ER) variants in two scenarios (A,C). For all metrics, higher values are indicative of a better performance: zero-shot model capabilities on future concepts (FWT), models' ability to avoid forgetting and improve on previously learned concepts (BWT), and anomaly detection performance (AUC). Values in bold highlight the best model for each strategy according to AUC.

Scenario: A												
Strategy		**OCSVM**			**IF**			**LOF**			**COPOD**	
	FWT	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**
Naive	0.668	−0.107	0.635	0.616	−0.09	0.615	0.652	−0.076	0.65	0.508	−0.084	0.462
Cumulative	0.606	−0.082	0.655	0.604	−0.024	0.658	0.579	−0.003	0.673	0.526	−0.037	0.586
ER (Random)	0.608	−0.105	0.648	0.594	−0.036	0.673	0.589	−0.102	0.636	0.524	−0.1	0.51
ER (Selective)	0.664	−0.114	0.641	0.602	−0.063	0.645	0.637	−0.1	0.629	0.526	−0.096	0.518
Strategy		**HBOS**			**ABOD**			**AE**				
	FWT	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**			
Naive	0.532	−0.063	0.612	0.663	−0.118	0.613	0.66	−0.002	**0.66**			
Cumulative	0.55	−0.005	0.644	0.595	0.001	**0.679**	0.61	−0.008	0.645			
ER (Random)	0.539	−0.026	0.628	0.6	0.003	**0.674**	0.594	−0.013	0.65			
ER (Selective)	0.533	−0.025	0.631	0.68	−0.029	**0.668**	0.622	−0.002	0.652			
Scenario: C												
Strategy		**OCSVM**			**IF**			**LOF**			**COPOD**	
	FWT	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**
Naive	0.595	−0.343	0.67	0.583	−0.306	0.654	0.584	−0.317	**0.671**	0.42	−0.18	0.536
Cumulative	0.606	−0.109	0.69	0.612	−0.051	0.671	0.535	−0.071	**0.758**	0.471	−0.044	0.58
ER (Random)	0.591	−0.224	0.692	0.592	−0.151	0.678	0.573	−0.184	**0.729**	0.444	−0.13	0.56
ER (Selective)	0.602	−0.234	0.701	0.595	−0.154	0.678	0.553	−0.197	0.726	0.444	−0.122	0.563
Strategy		**HBOS**			**ABOD**			**AE**				
	FWT	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**			
Naive	0.55	−0.141	0.622	0.571	−0.347	0.669	0.622	−0.092	0.64			
Cumulative	0.569	−0.029	0.632	0.566	−0.082	0.727	0.624	−0.037	0.659			
ER (Random)	0.557	−0.083	0.628	0.586	−0.135	0.726	0.615	−0.036	0.63			
ER (Selective)	0.554	−0.086	0.624	0.581	−0.169	**0.73**	0.616	−0.058	0.647			

The heatmaps shown in Figures 3–5 allow us to zoom into the performance of different replay strategies on specific tasks/concepts. Considering the CIC-MalMem-2022 dataset presented in Figure 3, we observe that, for the A scenario, the ER strategy manages to preserve knowledge of multiple concepts (3, 10–12), especially in the second half of the scenario, as visible by the darker colored areas. Training the model on concept 12 shows a significant spike in model performance on all concepts, whereas the performance on concept 7 is preserved throughout the entire scenario. However, its performance is still far from the upper bound achieved by cumulative. In the C scenario, ER appears less robust on concept 7 across the entire scenario, although training the model on task 7 appears to boost the performance on all concepts. The performance on concept 13 is preserved throughout the entire scenario. Shifting the focus to the CIC-Evasive-PDFMal2022 dataset in Figure 4, we observe a robust performance for ER in the A scenario, as shown by the dark-colored areas in the lower diagonal part of the heatmap. This performance is quite close to that achieved by the cumulative strategy. However, the performance for ER appears weaker on concept 2 in the C scenario. On the CIC-AndMal2017 dataset, results in Figure 5 show a generally satisfactory performance for ER across the scenario. The model appears relatively robust to forgetting, and its performance is significantly better than naive. One interesting case is that of concept 3, where ER achieves 0.93 after the model is trained on concept 0, possibly due to a high concept similarity. On this dataset, it can also be noted that the difference in performance between the ER and cumulative strategies is minimal, which represents a successful case for ER.



Scenario: A



Scenario: C

Figure 3. Continual ROC-AUC performance (CIC-MalMem-2022 dataset) with different strategies (naive: left; ER—Best-performing variant: center; cumulative: right) on single tasks/concepts after learning each task in two scenarios (A, C) with the best-performing one-class model (LOF).

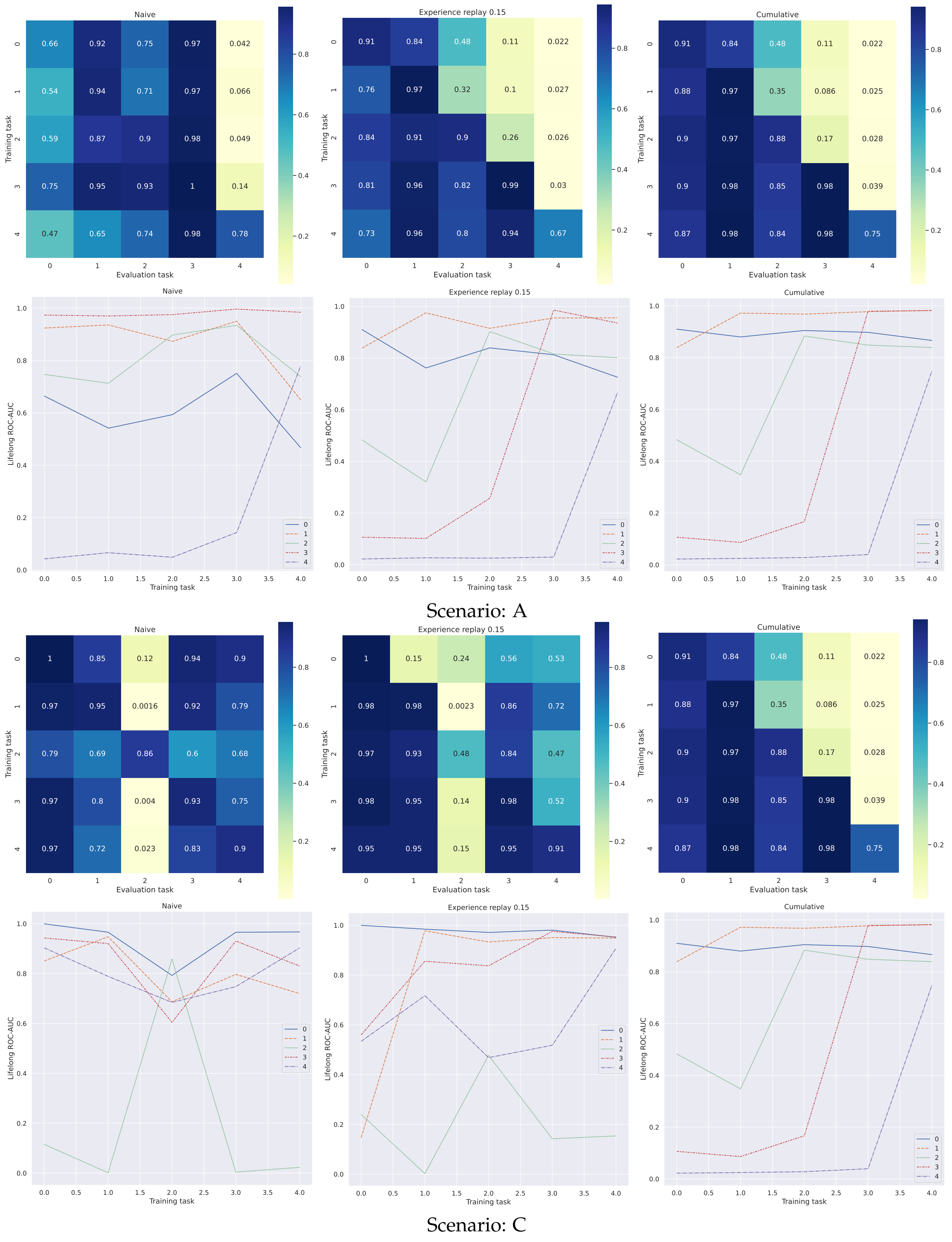


Figure 4. Continual ROC-AUC performance (CIC-Evasive-PDFMal2022 dataset) with different strategies (naive: left; ER—Best-performing variant: center; cumulative: right) on single tasks/concepts after learning each task in two scenarios (A, C) with the best-performing one-class model (ABOD).

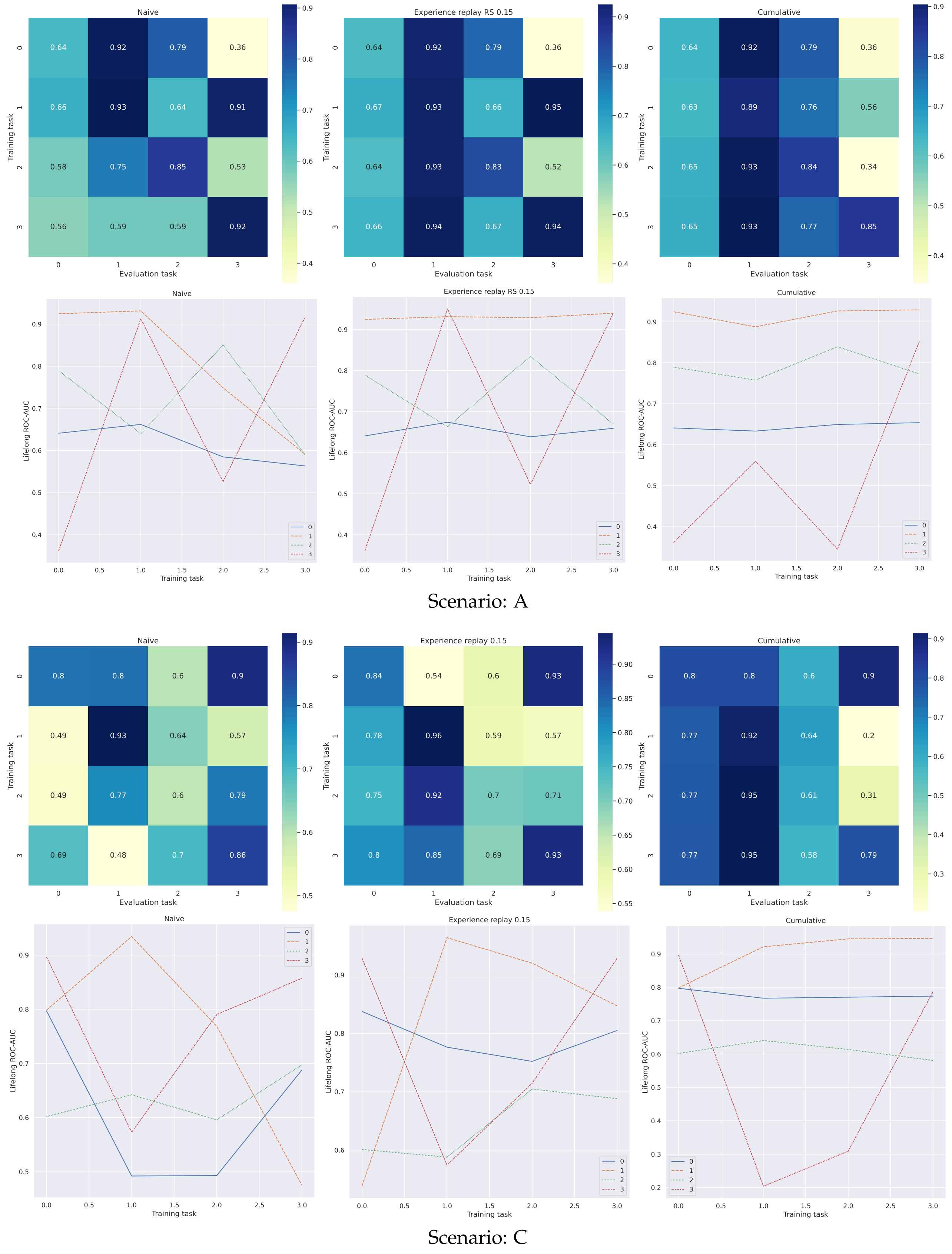


Figure 5. Continual ROC-AUC performance (CIC-AndMal2017 dataset) with different strategies (naive: left; ER—Best-performing variant: center; cumulative: right) on single tasks/concepts after learning each task in two scenarios (A, C) with the best-performing one-class models (ABOD and LOF, respectively).

5. Conclusions

In this work, we investigate a semi-supervised one-class continual malware detection workflow, leveraging two model-agnostic experience replay strategies. In addition to adopting a conventional replay strategy that selects data samples at random from each concept, we proposed a selective replay strategy, which attempts to select the most relevant data by prioritizing samples that are closer to the centroid of each concept. Our extensive evaluation involves seven diverse anomaly detection models, three real-world malware detection datasets, and two continual learning scenarios. Our results show that anomaly detection models combined with replay strategies are in some cases effective in providing a proper stability–plasticity trade-off in the presence of complex malware detection scenarios. In our experiments, replay strategies are capable of improving the performance of a naive incremental learning strategy in 34 out of 42 of the considered settings, both in terms of continual ROC-AUC and backward transfer. Our results suggest that the selective replay strategy achieves a similar performance to conventional replay, with slight improvements in some configurations. Moreover, the experiments emphasize that some models (LOF, ABOD) are more robust than others in terms of detection performance and limited forgetting.

Overall, we advocate continual learning as a suitable learning framework for malware detection, with the potential to provide reactive model capabilities in evolving scenarios with limited computational requirements. However, we underscore that further research is required to design more sophisticated strategies in order to fill the existing gap between replay strategies and simulated upper-bound performance, as emphasized by our results with the cumulative strategy. In future work, we aim to explore continual malware detection in task-free/task-agnostic scenarios where task boundaries are not known. Moreover, we aim to investigate new continual learning strategies specifically tailored for malware detection.

Author Contributions: Conceptualization, R.C.; methodology, R.C. and M.C.; software, R.C. and M.C.; validation, R.C.; investigation, R.C. and M.C.; resources, R.C.; data curation, M.C.; writing—original draft preparation, R.C.; writing—review and editing, M.C.; visualization, R.C. and M.C.; supervision, R.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The code and the processed datasets of this paper are available at https://github.com/rcorizzo/cl-malware accessed on 29 October 2024.

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

Appendix A. Model Hyperparameters

For experience replay (ER), the budget B adopted for results reported in the main paper is 20% of the data samples available in each concept.

For reproducibility, Table A1 shows hyperparameter configurations for all models considered in our experiments. Using ROC-AUC as an evaluation metric allows us to overcome the burden of optimizing hyperparameters that influence different scales for the anomaly score since no decision function based on a threshold is used for the predictions. For all methods, we follow the recommended guidelines or default values provided in the PyOD [59] documentation or recommended in reference papers of the methods, where suitable.

Table A1. Hyperparameters for all the anomaly detection models considered in our experiments. All models are trained and evaluated five times using all hyperparameter values in the sets shown in the table, and the final results are averaged.

LOF	leaf_size = 20, n_neighbors = 5
IF	n_estimators = 85
OCSVM	kernel = rbf, gamma = 8, shrinking = True
COPOD	parameterless

Appendix A.1. Qualitative Analysis

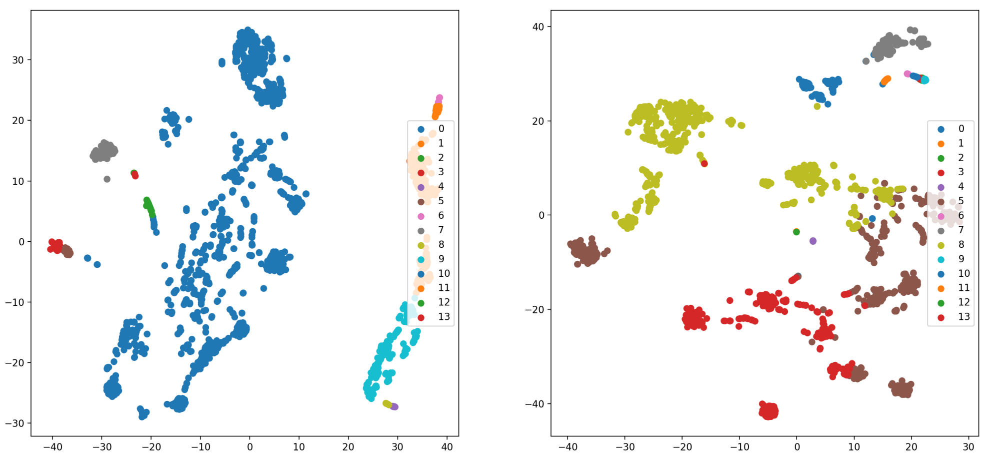


Figure A1. Visualization of extracted concepts via t-SNE: CIC-MalMem-2022 dataset. Normal class (**left**) and anomaly class (**right**).

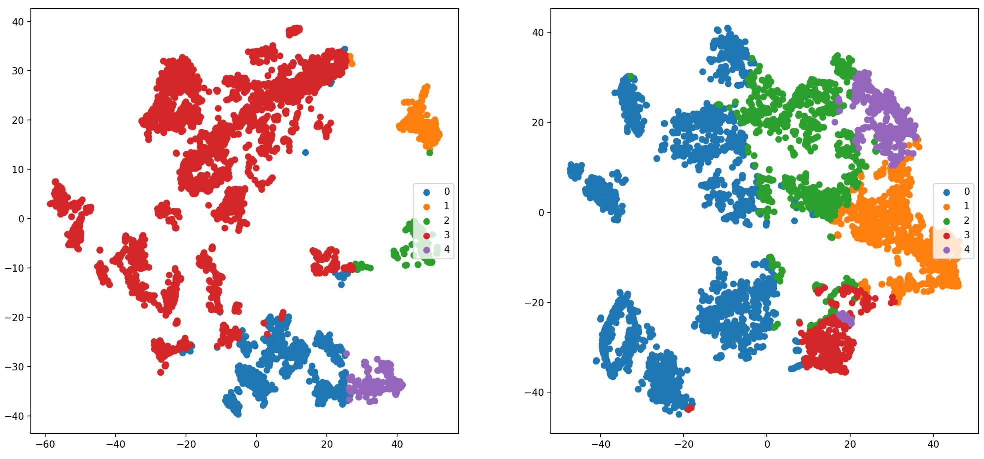


Figure A2. Visualization of extracted concepts via t-SNE: CIC-Evasive-PDFMal2022 dataset. Normal class (**left**) and anomaly class (**right**).

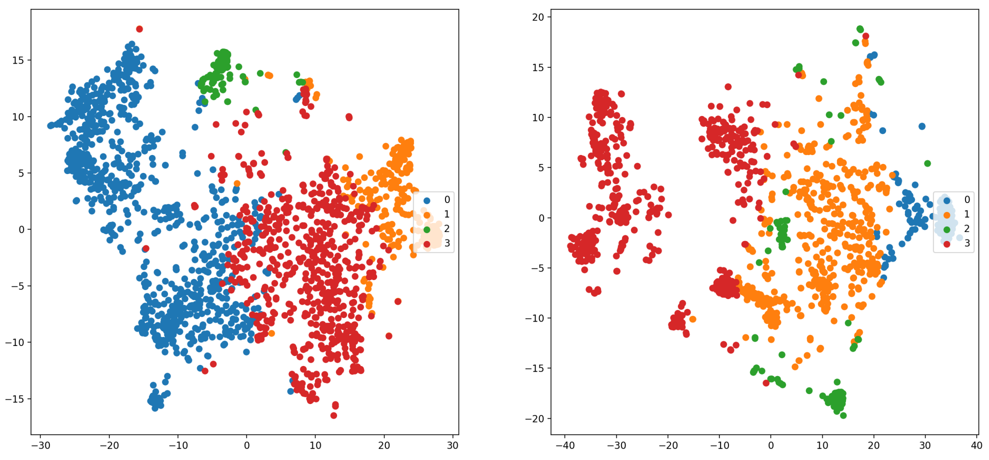


Figure A3. Visualization of extracted concepts via t-SNE: CIC-AndMal-2017 dataset. Normal class (**left**) and anomaly class (**right**).

Appendix A.2. Ablation Experiments

Table A2. Results: CIC-MalMem-2022 dataset: forward transfer (FWT), backward transfer (BWT), and ROC-AUC with all models and strategies including two experience replay (ER) variants in two scenarios (A,C) with different budget rates (B = 0.05, 0.1, 0.15, 0.2).

Scenario: A

Strategy	**OCSVM**			**IF**			**LOF**			**COPOD**		
	FWT	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**
ER (Random) 0.05	0.395	0.002	0.454	0.462	0.006	0.512	0.575	0.054	0.639	0.365	0.007	0.442
ER (Random) 0.1	0.398	0.001	0.456	0.461	−0.001	0.51	0.584	0.059	0.647	0.366	0.008	0.444
ER (Random) 0.15	0.4	0.001	0.459	0.461	0.001	0.511	0.594	0.059	0.659	0.367	0.01	0.447
ER (Random) 0.2	0.397	0.001	0.456	0.462	−0.006	0.511	0.601	0.064	0.669	0.368	0.011	0.448
ER (Selective) 0.05	0.389	0.001	0.453	0.452	0.001	0.499	0.566	0.058	0.633	0.361	0.005	0.436
ER (Selective) 0.1	0.387	−0.003	0.458	0.446	0.001	0.497	0.573	0.059	0.643	0.359	0.007	0.437
ER (Selective) 0.15	0.39	−0.005	0.459	0.44	−0.005	0.485	0.586	0.072	0.662	0.359	0.008	0.438
ER (Selective) 0.2	0.393	−0.007	0.458	0.434	−0.007	0.479	0.602	0.076	0.673	0.36	0.006	0.439

Strategy	**HBOS**			**ABOD**			**AE**		
	FWT	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**
ER (Random) 0.05	0.454	0.001	0.511	0.612	−0.098	0.536	0.587	0.001	0.569
ER (Random) 0.1	0.455	0.0	0.511	0.587	−0.156	0.476	0.588	0.001	0.57
ER (Random) 0.15	0.454	0.001	0.511	0.549	−0.2	0.419	0.588	0.0	0.571
ER (Random) 0.2	0.454	0.001	0.511	0.511	−0.223	0.372	0.588	−0.001	0.57
ER (Selective) 0.05	0.449	−0.001	0.504	0.616	−0.042	0.576	0.588	0.001	0.569
ER (Selective) 0.1	0.445	−0.002	0.501	0.604	−0.049	0.558	0.588	−0.0	0.57
ER (Selective) 0.15	0.444	−0.004	0.498	0.591	−0.053	0.546	0.588	−0.0	0.571
ER (Selective) 0.2	0.445	−0.008	0.493	0.577	−0.049	0.534	0.588	0.001	0.571

Scenario: C

Strategy	**OCSVM**			**IF**			**LOF**			**COPOD**		
	FWT	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**
ER (Random) 0.05	0.384	0.002	0.476	0.497	0.001	0.486	0.624	0.042	0.616	0.37	0.001	0.446
ER (Random) 0.1	0.38	0.001	0.472	0.495	0.004	0.485	0.63	0.05	0.634	0.374	0.003	0.448
ER (Random) 0.15	0.381	0.001	0.474	0.496	0.001	0.483	0.632	0.061	0.648	0.382	0.002	0.45
ER (Random) 0.2	0.382	0.001	0.476	0.495	0.002	0.483	0.639	0.072	0.669	0.387	0.004	0.451
ER (Selective) 0.05	0.403	−0.013	0.453	0.485	−0.009	0.466	0.615	0.045	0.607	0.367	−0.004	0.437
ER (Selective) 0.1	0.406	−0.009	0.448	0.472	−0.011	0.46	0.621	0.05	0.623	0.374	−0.004	0.436
ER (Selective) 0.15	0.409	−0.008	0.444	0.465	−0.011	0.45	0.63	0.064	0.654	0.384	−0.005	0.435
ER (Selective) 0.2	0.411	−0.01	0.441	0.464	−0.01	0.447	0.641	0.072	0.667	0.386	−0.006	0.434

Strategy	**HBOS**			**ABOD**			**AE**		
	FWT	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**
ER (Random) 0.05	0.495	−0.001	0.479	0.59	−0.145	0.455	0.562	0.001	0.591
ER (Random) 0.1	0.496	−0.001	0.479	0.538	−0.25	0.321	0.573	0.001	0.593
ER (Random) 0.15	0.495	−0.001	0.479	0.512	−0.265	0.286	0.564	0.001	0.595
ER (Random) 0.2	0.495	−0.001	0.479	0.498	−0.247	0.294	0.564	0.0	0.593
ER (Selective) 0.05	0.489	−0.006	0.469	0.617	−0.014	0.582	0.566	0.002	0.593
ER (Selective) 0.1	0.481	−0.01	0.462	0.605	−0.021	0.568	0.564	0.001	0.595
ER (Selective) 0.15	0.474	−0.012	0.458	0.59	−0.029	0.553	0.566	0.0	0.594
ER (Selective) 0.2	0.473	−0.014	0.455	0.574	−0.029	0.542	0.564	0.001	0.593

Table A3. Results: CIC-Evasive-PDFMal2022 dataset: forward transfer (FWT), backward transfer (BWT), and ROC-AUC with all models and strategies including two experience replay (ER) variants in two scenarios (A,C) with different budget rates (B = 0.05, 0.1, 0.15, 0.2).

Scenario: A

Strategy		**OCSVM**			**IF**			**LOF**			**COPOD**	
	FWT	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**
ER (Random) 0.05	0.187	−0.279	0.399	0.555	−0.086	0.698	0.2	−0.263	0.557	0.507	−0.089	0.613
ER (Random) 0.1	0.191	−0.282	0.379	0.558	−0.13	0.7	0.186	−0.079	0.674	0.508	−0.051	0.628
ER (Random) 0.15	0.186	−0.266	0.383	0.564	−0.111	0.704	0.184	−0.022	0.712	0.51	−0.022	0.642
ER (Random) 0.2	0.187	−0.257	0.388	0.563	−0.106	0.689	0.184	−0.033	0.702	0.51	−0.008	0.652
ER (Selective) 0.05	0.187	−0.196	0.518	0.544	−0.095	0.711	0.199	−0.303	0.537	0.508	−0.09	0.61
ER (Selective) 0.1	0.186	−0.179	0.513	0.557	−0.084	0.721	0.196	−0.163	0.624	0.51	−0.037	0.63
ER (Selective) 0.15	0.187	−0.178	0.507	0.565	−0.107	0.721	0.184	−0.123	0.66	0.511	−0.013	0.644
ER (Selective) 0.2	0.185	−0.18	0.503	0.564	−0.052	0.731	0.184	−0.115	0.668	0.511	0.002	0.653
Strategy		**HBOS**			**ABOD**			**AE**				
	FWT	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**			
ER (Random) 0.05	0.561	−0.026	0.778	0.239	−0.122	0.842	0.427	−0.004	0.617			
ER (Random) 0.1	0.56	−0.01	0.776	0.216	−0.102	0.853	0.346	−0.017	0.615			
ER (Random) 0.15	0.557	−0.002	0.776	0.221	−0.084	0.864	0.376	−0.049	0.54			
ER (Random) 0.2	0.553	−0.002	0.773	0.221	−0.071	0.875	0.371	−0.016	0.551			
ER (Selective) 0.05	0.557	−0.022	0.782	0.203	−0.306	0.718	0.432	−0.165	0.513			
ER (Selective) 0.1	0.559	−0.017	0.778	0.237	−0.2	0.794	0.382	−0.144	0.517			
ER (Selective) 0.15	0.558	−0.008	0.779	0.232	−0.172	0.814	0.315	−0.028	0.578			
ER (Selective) 0.2	0.555	−0.009	0.775	0.204	−0.145	0.832	0.451	−0.069	0.569			

Scenario: C

Strategy		**OCSVM**			**IF**			**LOF**			**COPOD**	
	FWT	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**
ER (Random) 0.05	0.383	−0.471	0.561	0.626	−0.245	0.722	0.552	−0.025	0.666	0.579	−0.109	0.611
ER (Random) 0.1	0.378	−0.429	0.551	0.657	−0.18	0.734	0.551	0.075	0.754	0.582	−0.083	0.625
ER (Random) 0.15	0.373	−0.419	0.538	0.652	−0.138	0.728	0.48	0.067	0.781	0.594	−0.07	0.631
ER (Random) 0.2	0.372	−0.411	0.525	0.629	−0.09	0.715	0.482	0.04	0.761	0.596	−0.059	0.637
ER (Selective) 0.05	0.372	−0.464	0.582	0.635	−0.239	0.717	0.502	−0.155	0.636	0.574	−0.108	0.611
Strategy		**HBOS**			**ABOD**			**AE**				
	FWT	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**			
ER (Selective) 0.1	0.374	−0.398	0.574	0.641	−0.161	0.709	0.414	−0.05	0.754	0.58	−0.082	0.624
ER (Selective) 0.15	0.375	−0.383	0.565	0.664	−0.138	0.733	0.405	−0.068	0.761	0.592	−0.067	0.633
ER (Selective) 0.2	0.376	−0.368	0.552	0.663	−0.132	0.719	0.389	0.022	0.767	0.595	−0.055	0.638
ER (Random) 0.05	0.7	−0.121	0.772	0.483	−0.144	0.792	0.396	−0.254	0.516			
ER (Random) 0.1	0.704	−0.074	0.775	0.489	−0.103	0.818	0.419	−0.176	0.535			
ER (Random) 0.15	0.708	−0.063	0.773	0.488	−0.09	0.821	0.471	−0.27	0.541			
ER (Random) 0.2	0.708	−0.051	0.77	0.487	−0.077	0.821	0.454	−0.182	0.536			
ER (Selective) 0.05	0.698	−0.125	0.755	0.38	−0.264	0.728	0.352	−0.081	0.577			
ER (Selective) 0.1	0.703	−0.096	0.746	0.411	−0.162	0.791	0.428	0.141	0.507			
ER (Selective) 0.15	0.706	−0.075	0.753	0.422	−0.155	0.794	0.481	−0.169	0.566			
ER (Selective) 0.2	0.706	−0.058	0.758	0.417	−0.155	0.794	0.495	−0.186	0.535			

Table A4. Results: CIC-AndMal2017 dataset: forward transfer (FWT), backward transfer (BWT), and ROC-AUC with all models and strategies including two experience replay (ER) variants in two scenarios (A,C) with different budget rates (B = 0.05, 0.1, 0.15, 0.2).

Scenario: A

Strategy		**OCSVM**			**IF**			**LOF**			**COPOD**	
	FWT	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**
ER (Random) 0.05	0.62	−0.103	0.644	0.619	−0.055	0.644	0.616	−0.091	0.642	0.518	−0.095	0.484
ER (Random) 0.1	0.606	−0.102	0.648	0.599	−0.054	0.656	0.596	−0.129	0.622	0.522	−0.098	0.5
ER (Random) 0.15	0.608	−0.105	0.648	0.594	−0.036	0.673	0.589	−0.102	0.636	0.524	−0.1	0.51
ER (Random) 0.2	0.604	−0.099	0.646	0.609	−0.027	0.665	0.581	−0.089	0.643	0.526	−0.098	0.52
ER (Selective) 0.05	0.666	−0.123	0.635	0.596	−0.063	0.639	0.654	−0.122	0.62	0.52	−0.097	0.488
ER (Selective) 0.1	0.664	−0.12	0.638	0.597	−0.042	0.662	0.648	−0.135	0.61	0.524	−0.098	0.506
ER (Selective) 0.15	0.664	−0.114	0.641	0.602	−0.063	0.645	0.637	−0.1	0.629	0.526	−0.096	0.518
ER (Selective) 0.2	0.662	−0.107	0.645	0.594	−0.015	0.678	0.631	−0.089	0.635	0.528	−0.091	0.528

Strategy		**HBOS**			**ABOD**			**AE**	
	FWT	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**
ER (Random) 0.05	0.535	−0.04	0.622	0.616	−0.003	0.673	0.584	−0.005	0.642
ER (Random) 0.1	0.538	−0.031	0.626	0.591	0.005	0.675	0.612	−0.006	0.649
ER (Random) 0.15	0.539	−0.026	0.628	0.6	0.003	0.674	0.594	−0.013	0.65
ER (Random) 0.2	0.542	−0.018	0.631	0.602	0.006	0.682	0.588	−0.009	0.641
ER (Selective) 0.05	0.532	−0.04	0.623	0.675	−0.057	0.652	0.603	0.012	0.651
ER (Selective) 0.1	0.532	−0.032	0.628	0.68	−0.03	0.668	0.598	0.001	0.654
ER (Selective) 0.15	0.533	−0.025	0.631	0.68	−0.029	0.668	0.622	−0.002	0.652
ER (Selective) 0.2	0.533	−0.02	0.633	0.68	−0.031	0.666	0.667	0.001	0.653

Scenario: C

Strategy		**OCSVM**			**IF**			**LOF**			**COPOD**	
	FWT	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**
ER (Random) 0.05	0.6	−0.262	0.683	0.587	−0.236	0.673	0.6	−0.214	0.677	0.431	−0.159	0.545
ER (Random) 0.1	0.591	−0.242	0.686	0.579	−0.167	0.68	0.585	−0.201	0.707	0.438	−0.143	0.553
ER (Random) 0.15	0.591	−0.224	0.692	0.592	−0.151	0.678	0.573	−0.184	0.729	0.444	−0.13	0.56
ER (Random) 0.2	0.595	−0.211	0.69	0.575	−0.144	0.672	0.569	−0.158	0.751	0.449	−0.119	0.565
ER (Selective) 0.05	0.602	−0.275	0.687	0.572	−0.214	0.688	0.562	−0.262	0.694	0.431	−0.157	0.546
ER (Selective) 0.1	0.601	−0.254	0.696	0.604	−0.185	0.672	0.552	−0.227	0.714	0.438	−0.138	0.556
ER (Selective) 0.15	0.602	−0.234	0.701	0.595	−0.154	0.678	0.553	−0.197	0.726	0.444	−0.122	0.563
ER (Selective) 0.2	0.601	−0.22	0.703	0.593	−0.138	0.671	0.552	−0.176	0.739	0.449	−0.111	0.568

Strategy		**HBOS**			**ABOD**			**AE**	
	FWT	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**	**FWT**	**BWT**	**AUC**
ER (Random) 0.05	0.555	−0.108	0.63	0.593	−0.18	0.707	0.626	−0.036	0.635
ER (Random) 0.1	0.555	−0.092	0.63	0.594	−0.15	0.721	0.613	−0.035	0.629
ER (Random) 0.15	0.557	−0.083	0.628	0.586	−0.135	0.726	0.615	−0.036	0.63
ER (Random) 0.2	0.559	−0.076	0.629	0.59	−0.131	0.727	0.619	−0.036	0.628
ER (Selective) 0.05	0.552	−0.107	0.621	0.576	−0.211	0.717	0.613	−0.055	0.646
ER (Selective) 0.1	0.553	−0.095	0.626	0.581	−0.179	0.729	0.614	−0.061	0.641
ER (Selective) 0.15	0.554	−0.086	0.624	0.581	−0.169	0.73	0.616	−0.058	0.647
ER (Selective) 0.2	0.557	−0.076	0.627	0.583	−0.156	0.737	0.625	−0.045	0.644

References

1. Almashhadani, A.O.; Carlin, D.; Kaiiali, M.; Sezer, S. MFMCNS: A multi-feature and multi-classifier network-based system for ransomworm detection. *Comput. Secur.* **2022**, *121*, 102860. [CrossRef]
2. Qiu, J.; Han, Q.L.; Luo, W.; Pan, L.; Nepal, S.; Zhang, J.; Xiang, Y. Cyber code intelligence for android malware detection. *IEEE Trans. Cybern.* **2022**, *53*, 617–627. [CrossRef] [PubMed]
3. Zhu, H.j.; Gu, W.; Wang, L.m.; Xu, Z.c.; Sheng, V.S. Android malware detection based on multi-head squeeze-and-excitation residual network. *Expert Syst. Appl.* **2023**, *212*, 118705. [CrossRef]

4. Hessel, M.; Modayil, J.; Van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M.; Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32.
5. Gu, S.; Lillicrap, T.; Sutskever, I.; Levine, S. Continuous deep q-learning with model-based acceleration. In Proceedings of the International Conference on Machine Learning, PMLR, New York, NY, USA, 20–22 June 2016; pp. 2829–2838.
6. Saglam, B.; Mutlu, F.B.; Cicek, D.C.; Kozat, S.S. Actor prioritized experience replay. *J. Artif. Intell. Res.* **2023**, *78*, 639–672. [CrossRef]
7. Andrychowicz, M.; Wolski, F.; Ray, A.; Schneider, J.; Fong, R.; Welinder, P.; McGrew, B.; Tobin, J.; Pieter Abbeel, O.; Zaremba, W. Hindsight experience replay. *arXiv* **2017**, arXiv:1707.01495.
8. Parisi, G.I.; Kemker, R.; Part, J.L.; Kanan, C.; Wermter, S. Continual lifelong learning with neural networks: A review. *Neural Netw.* **2019**, *113*, 54–71. [CrossRef]
9. Shakya, A.K.; Pillai, G.; Chakrabarty, S. Reinforcement learning algorithms: A brief survey. *Expert Syst. Appl.* **2023**, *231*, 120495. [CrossRef]
10. Bensaoud, A.; Kalita, J.; Bensaoud, M. A survey of malware detection using deep learning. *Mach. Learn. Appl.* **2024**, *16*, 100546. [CrossRef]
11. Zhu, D.; Jin, H.; Yang, Y.; Wu, D.; Chen, W. DeepFlow: Deep learning-based malware detection by mining Android application for abnormal usage of sensitive data. In Proceedings of the 2017 IEEE Symposium on Computers and Communications (ISCC), Heraklion, Greece, 3–6 July 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 438–443.
12. HaddadPajouh, H.; Dehghantanha, A.; Khayami, R.; Choo, K.K.R. A deep recurrent neural network based approach for internet of things malware threat hunting. *Future Gener. Comput. Syst.* **2018**, *85*, 88–96. [CrossRef]
13. Lee, Y.; Wang, X.; Liao, X.; Wang, X. Understanding illicit UI in iOS apps through hidden UI analysis. *IEEE Trans. Dependable Secur. Comput.* **2019**, *18*, 2390–2402. [CrossRef]
14. Beaman, C.; Barkworth, A.; Akande, T.D.; Hakak, S.; Khan, M.K. Ransomware: Recent advances, analysis, challenges and future research directions. *Comput. Secur.* **2021**, *111*, 102490. [CrossRef] [PubMed]
15. Fernando, D.W.; Komninos, N. FeSA: Feature selection architecture for ransomware detection under concept drift. *Comput. Secur.* **2022**, *116*, 102659. [CrossRef]
16. Liu, M.; Yang, Q.; Wang, W.; Liu, S. Semi-Supervised Encrypted Malicious Traffic Detection Based on Multimodal Traffic Characteristics. *Sensors* **2024**, *24*, 6507. [CrossRef] [PubMed]
17. Memon, M.; Unar, A.A.; Ahmed, S.S.; Daudpoto, G.H.; Jaffari, R. Feature-based semi-supervised learning approach to android malware detection. *Eng. Proc.* **2023**, *32*, 6. [CrossRef]
18. Yu, X.; Lin, G.; Hu, X.; Keung, J.W.; Xia, X. Less is More: Unlocking Semi-Supervised Deep Learning for Vulnerability Detection. *Acm Trans. Softw. Eng. Methodol.* 2024. [CrossRef]
19. Eren, M.E.; Alexandrov, B.S.; Nicholas, C. Classifying Malware Using Tensor Decomposition. In *Malware: Handbook of Prevention and Detection*; Springer: Berlin/Heidelberg, Germany, 2024; pp. 3–36.
20. Eren, M.E.; Bhattarai, M.; Joyce, R.J.; Raff, E.; Nicholas, C.; Alexandrov, B.S. Semi-supervised classification of malware families under extreme class imbalance via hierarchical non-negative matrix factorization with automatic model selection. *ACM Trans. Priv. Secur.* **2023**, *26*, 1–27. [CrossRef]
21. Van de Ven, G.M.; Tolias, A.S. Three scenarios for continual learning. *arXiv* **2019**, arXiv:1904.07734.
22. De Lange, M.; Tuytelaars, T. Continual Prototype Evolution: Learning Online From Non-Stationary Data Streams. In Proceedings of the Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 10–17 October 2021; pp. 8250–8259.
23. Cossu, A.; Graffieti, G.; Pellegrini, L.; Maltoni, D.; Bacciu, D.; Carta, A.; Lomonaco, V. Is Class-Incremental Enough for Continual Learning? *arXiv* **2022**, arXiv:2112.02925. [CrossRef]
24. Dragoi, M.; Burceanu, E.; Haller, E.; Manolache, A.; Brad, F. AnoShift: A distribution shift benchmark for unsupervised anomaly detection. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 32854–32867.
25. Sharif Razavian, A.; Azizpour, H.; Sullivan, J.; Carlsson, S. CNN features off-the-shelf: An astounding baseline for recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Columbus, OH, USA, 23–28 June 2014; pp. 806–813.
26. Lomonaco, V.; Maltoni, D. Core50: A new dataset and benchmark for continuous object recognition. In Proceedings of the Conference on Robot Learning—PMLR, Mountain View, CA, USA, 13–15 November 2017; pp. 17–26.
27. Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A.A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al. Overcoming catastrophic forgetting in neural networks. *Proc. Natl. Acad. Sci. USA* **2017**, *114*, 3521–3526. [CrossRef]
28. Li, Z.; Hoiem, D. Learning without Forgetting. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *40*, 2935–2947. [CrossRef] [PubMed]
29. Diethe, T.; Borchert, T.; Thereska, E.; Balle, B.; Lawrence, N. Continual learning in practice. *arXiv* **2018**, arXiv:1903.05202.
30. Mignone, P.; Corizzo, R.; Ceci, M. Distributed and explainable GHSOM for anomaly detection in sensor networks. *Mach. Learn.* **2024**, *113*, 4445–4486. [CrossRef]
31. Mallya, A.; Lazebnik, S. PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning. *arXiv* **2017**, arXiv:1711.05769.

32. Kang, H.; Mina, R.J.L.; Rizky, S.; Madjid, H.; Yoon, J.; Hasegawa-Johnson, M.; Ju-Hwang, S.; Yoo, C.D. Forget-free Continual Learning with Winning Subnetworks. *ICML* **2022**, *162*, 10734–10750.
33. Pietroń, M.; Żurek, D.; Faber, K.; Corizzo, R. Ada-QPacknet–adaptive pruning with bit width reduction as an efficient continual learning method without forgetting. In Proceedings of the European Conference on Artificial Intelligence (ECAI), Krakow, Poland, 30 September–4 October 2023; pp. 1882–1889.
34. Kumari, L.; Wang, S.; Zhou, T.; Bilmes, J.A. Retrospective adversarial replay for continual learning. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 28530–28544.
35. Van de Ven, G.M.; Siegelmann, H.T.; Tolias, A.S. Brain-inspired replay for continual learning with artificial neural networks. *Nat. Commun.* **2020**, *11*, 4069. [CrossRef]
36. Li, X.; Tang, B.; Li, H. AdaER: An adaptive experience replay approach for continual lifelong learning. *Neurocomputing* **2024**, *572*, 127204. [CrossRef]
37. Wang, Q.; Ji, Z.; Pang, Y.; Zhang, Z. Uncertainty-aware enhanced dark experience replay for continual learning. *Appl. Intell.* **2024**, *54*, 7135–7150. [CrossRef]
38. Lim, W.S.; Zhou, Y.; Kim, D.W.; Lee, J. MixER: Mixup-Based Experience Replay for Online Class-Incremental Learning. *IEEE Access* **2024**, *12*, 41801–41814. [CrossRef]
39. Buzzega, P.; Boschini, M.; Porrello, A.; Calderara, S. Rethinking experience replay: A bag of tricks for continual learning. In Proceedings of the 2020 25th International Conference on Pattern Recognition (ICPR), Milan, Italy, 10–15 January 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 2180–2187.
40. Faber, K.; Sniezynski, B.; Corizzo, R. Distributed Continual Intrusion Detection: A Collaborative Replay Framework. In Proceedings of the 2023 IEEE International Conference on Big Data (BigData), Sorrento, Italy, 15–18 December 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 3255–3263.
41. Shin, H.; Lee, J.K.; Kim, J.; Kim, J. Continual Learning with Deep Generative Replay. In *Proceedings of the NeurIPS*; Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2017; Volume 30.
42. Faber, K.; Corizzo, R.; Sniezynski, B.; Japkowicz, N. VLAD: Task-agnostic VAE-based lifelong anomaly detection. *Neural Netw.* **2023**, *165*, 248–273. [CrossRef] [PubMed]
43. Rahman, M.S.; Coull, S.; Wright, M. On the Limitations of Continual Learning for Malware Classification. *arXiv* **2022**, arXiv:2208.06568.
44. Faber, K.; Corizzo, R.; Sniezynski, B.; Japkowicz, N. Lifelong Learning for Anomaly Detection: New Challenges, Perspectives, and Insights. *arXiv* **2023**, arXiv:2303.07557. [CrossRef]
45. Fayek, H.M.; Cavedon, L.; Wu, H.R. Progressive learning: A deep learning framework for continual learning. *Neural Netw.* **2020**, *128*, 345–357. [CrossRef]
46. Faber, K.; Zurek, D.; Pietron, M.; Japkowicz, N.; Vergari, A.; Corizzo, R. From MNIST to ImageNet and back: Benchmarking continual curriculum learning. *Mach. Learn.* **2024**, *113*, 8137–8164. [CrossRef]
47. Kesgin, H.T.; Amasyali, M.F. Cyclical curriculum learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, *35*, 12864–12872. [CrossRef]
48. Carrier, T. Detecting obfuscated malware using memory feature engineering. In Proceedings of the ICISSP, Virtual, 11–13 February 2022; pp. 177–188.
49. Issakhani, M.; Victor, P.; Tekeoglu, A.; Lashkari, A.H. PDF Malware Detection based on Stacking Learning. In Proceedings of the ICISSP, Virtual, 9–11 February 2022; pp. 562–570.
50. Lashkari, A.H.; Kadir, A.F.A.; Taheri, L.; Ghorbani, A.A. Toward developing a systematic approach to generate benchmark android malware datasets and classification. In Proceedings of the 2018 International Carnahan Conference on Security Technology (ICCST), Madrid, Spain, 22–25 October 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–7.
51. Breunig, M.M.; Kriegel, H.P.; Ng, R.T.; Sander, J. LOF: Identifying density-based local outliers. In Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, Dallas, TX, USA, 15–18 May 2000; pp. 93–104.
52. Liu, F.T.; Ting, K.M.; Zhou, Z.H. Isolation forest. In Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, Pisa, Italy, 15–19 December 2008; IEEE: Piscataway, NJ, USA; pp. 413–422.
53. Schölkopf, B.; Williamson, R.C.; Smola, A.J.; Shawe-Taylor, J.; Platt, J.C. Support vector method for novelty detection. In *Advances in Neural Information Processing Systems*; MIT Press: Denver, CO, USA, 2000; pp. 582–588.
54. Li, Z.; Zhao, Y.; Botta, N.; Ionescu, C.; Hu, X. COPOD: Copula-Based Outlier Detection. In Proceedings of the 2020 IEEE International Conference on Data Mining (ICDM), Sorrento, Italy, 17–20 November 2020; pp. 1118–1123. [CrossRef]
55. Goldstein, M.; Score, A.D.H.b.O. A fast Unsupervised Anomaly Detection Algorithm. In Proceedings of the KI-2012: Poster and Demo Track, Saarbrücken, Germany, 24–27 September 2012; pp. 59–63.
56. Kriegel, H.; Schubert, M.; Zimek, A. Angle-based outlier detection in high-dimensional data. In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, NV, USA, 24–27 August 2008; pp. 444–452. [CrossRef]
57. Gehring, J.; Miao, Y.; Metze, F.; Waibel, A. Extracting deep bottleneck features using stacked auto-encoders. In Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, Canada, 26–31 May 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 3377–3381.

58. Díaz-Rodríguez, N.; Lomonaco, V.; Filliat, D.; Maltoni, D. Don't forget, there is more than forgetting: New metrics for Continual Learning. *arXiv* **2018**, arXiv:1810.13166.
59. Zhao, Y.; Nasrullah, Z.; Li, Z. PyOD: A Python Toolbox for Scalable Outlier Detection. *J. Mach. Learn. Res.* **2019**, *20*, 1–7.