L2U: LEARNING WHAT TO UNLEARN FOR CONTINUALLY ADAPTING CYBERSECURITY THREAT DETECTORS

Anonymous authors
Paper under double-blind review

ABSTRACT

Machine learning models deployed on evolving data streams must continually adapt as the underlying data distribution changes over time. This challenge is particularly critical in cybersecurity, where shifts can occur naturally as malware, intrusion techniques, and adversarial behaviors evolve, or adversarially through poisoned samples designed to manipulate the model and enable future evasion. For example, benign network behavior observed in an earlier period may later resemble emerging malicious behavior, meanwhile an adversary may deliberately inject poisoned samples to influence how future attacks are classified. Continual Learning (CL) primarily focuses on incorporating new knowledge while preserving prior knowledge; however, indiscriminately retaining historical information can preserve redundant, obsolete, or harmful information that hinders future adaptation. We argue that effective adaptation should instead enable models to *adaptively unlearn*: learning what historical information to retain and what to remove as the data evolves. In this work, we propose a learning-to-unlearn (L2U) framework that learns which historical samples to unlearn while preserving useful prior knowledge. At each distribution shift, our mechanism evaluates historical data against the current data to determine what should be retained or unlearned, then performs unlearning as part of the adaptation process. We demonstrate the benefits of unlearning as part of the continual adaptation process in both non-adversarial settings, where it removes redundant or obsolete information, and adversarial settings, where it removes poisoned historical samples. Experiments across multiple continual adaptation scenarios show that L2U consistently outperforms CL baselines on new data while preserving performance on previously learned data.

1 INTRODUCTION

Machine learning models are increasingly deployed in environments where the data they encounter changes over time. In cybersecurity, for example, attackers constantly evolve malware, intrusion methods, and exploitation techniques, causing models trained on historical data to gradually become outdated. As a result, maintaining model performance requires the ability to continually adapt to new data. Several approaches have been proposed to address this problem, including few-shot adaptation (Liu et al., 2024), online learning (Gama et al., 2014), active learning (Žliobaitė et al., 2013), and Continual Learning (CL) (Wang et al., 2024), (Gu et al., 2024). Among these, CL is particularly appealing because it enables models to incrementally acquire new knowledge without retraining from scratch, adapting to new data while retaining some of what was previously learned. CL is most useful in scenarios where the underlying data distribution evolves over time and the cost of full retraining is prohibitive. For example, this includes settings where new classes or attack types emerge, where data arrives as a stream rather than a fixed batch, or where labeled data from earlier tasks becomes inaccessible or unreliable, whether due to privacy constraints, storage limits, or poisoning detected after the fact. In such settings, a CL-based model can be updated incrementally as new tasks arrive, preserving performance on prior tasks while remaining responsive to the current data distribution.

However, continual adaptation requires balancing two competing objectives: (i) retaining useful historical knowledge and (ii) remaining sufficiently plastic to learn from the current distribution. Prior CL work has largely emphasized the former, particularly *catastrophic forgetting*, where adaptation

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

to new data degrades previously acquired knowledge. Consequently, replay, regularization, and architectural approaches are commonly designed to preserve historical information (Wang et al., 2021; 2026). Yet, preservation is not always beneficial: as distributions evolve, historical information may become obsolete, conflicting, or, especially in security settings, *even harmful to adaptation*. This trade-off becomes particularly important under limited model capacity, where continually accommodating new distributions while preserving prior knowledge can progressively constrain plasticity (Chakraborty & Raghavan, 2025; Bonnet et al., 2025). Thus, effective continual adaptation requires not only deciding *what to retain*, but also *what should be forgotten*.

We ask a simple question: *while learning new information, should a model also forget information that is no longer useful?* A second challenge naturally follows: *what* should be forgotten? Rather than relying on handcrafted heuristics, we ask whether a model can learn this decision automatically. When distributions change abruptly, or even triggered by adversaries, the model has to adapt its own unlearning process and cannot rely only on heuristics, underscoring adaptive unlearning.

We formulate this adaptive selective forgetting as a meta-adaptation problem. Given features describing historical training samples, a forgetting "strategy" (Salimans et al., 2017) predicts which samples should be unlearned before adapting to a new data distribution. After adaptation, the model is evaluated to assess whether the selected forget set improved adaptation, and this performance is used as a fitness signal to refine the sample-selection strategy across distribution shifts. Over time, the strategy learns which historical samples are worth retaining and which can be safely discarded to maximize future adaptability. In the security context, selective forgetting also provides a practical security benefit. In long-running machine learning systems, poisoned or malicious samples may accumulate over time and influence future model updates. A learned forgetting strategy offers a natural mechanism for removing such harmful samples before they negatively affect subsequent adaptations. Our main contributions are summarized as follows:

- We demonstrate that ***selective unlearning*** during continual adaptation improves performance on the current task while preserving performance on previously learned tasks, showing that targeted forgetting can facilitate adaptation without exacerbating catastrophic forgetting.
- For ***natural or non-adversarial drift*** in security data, we introduce HEURISTIC-L2U, a distribution-aware, rule-based selection strategy that identifies redundant or obsolete historical samples to unlearn before adapting to the current data distribution.
- For ***targeted or adversarial drift***, we formulate a *long-range poisoning attack* in which poisoned historical samples persist across the adaptation process and degrade performance on future tasks. As a defense, we extend HEURISTIC-L2U to identify and selectively unlearn poisoned historical samples with minimal supervision.
- We make the unlearning selection strategy *learnable* through META-L2U, a meta-learning framework that learns which historical samples to unlearn based on past and current data distributions.

2 RELATED WORK

CL approaches that mitigate catastrophic forgetting come from a range of well known directions; **Replay-Based:** MADAR, MalCL, GCR (Rahman et al., 2025; Park et al., 2025; Tiwari et al., 2022; Park et al., 2025), **Regularization-Based:** SI, Progress and Compress (Zenke et al., 2017; Schwarz et al., 2018), and **Architecture-Based:** DEN, Packnet (Yoon et al., 2017; Mallya & Lazebnik, 2018). And although these approaches mitigate catastrophic forgetting, they do not erase it (Wang et al., 2024; Chen et al., 2023b), this encourages the idea of including forgetting as part of the learning process itself. DEDUCE (Wang et al., 2026) detects negative transfer from outdated knowledge via a transferability bound and gradient conflict analysis, removing it through a Local and Global Unlearning Module. However, its forgetting decisions target only task-level interference between clean tasks, not poisoned, stale, or redundant samples at the individual level. SSF (Zhang et al., 2025) similarly discards outdated replay samples upon detecting concept drift, improving IDS adaptability, but its forgetting criterion is likewise tied to natural drift and collapses against adversarial poisoning. Machine Unlearning is intended at first to target specific information for a model to forget to comply with GDPR (Bourtoule et al., 2021; Xu et al., 2024) and verify that its influence has been removed (Arslan & Aydemir, 2026; Hui et al., 2023). However, now approaches exist to balance forget quality and utility, speed and scalability and interpretation (Kurmanji et al., 2023; Tarun et al., 2023; Nguyen et al., 2022), however many may not consider if it's possible to learn what information is best to unlearn. The purpose of Meta learning is learning to learn or learning to generalize (Gharoun et al.,

2024). This is not trivial and requires more investment than normally training a model (Finn et al., 2017; Hospedales et al., 2021). However, at a minimal cost of added training, you get a model that is able to generalize very easily (Finn et al., 2017; Snell et al., 2017).

3 METHODOLOGY

3.1 PRELIMINARIES

Continual Learning: We consider a CL setting with sequential tasks $\{D_1, \ldots, D_T\}$. At task t, the classifier adapts to D_t while retaining knowledge from previous tasks. Experience replay (ER) mitigates catastrophic forgetting by maintaining a bounded replay buffer R_t of previously observed samples used during subsequent adaptation.

Distribution-Aware Replay: Following MADAR (Rahman et al., 2025), we use Isolation Forest (IF) to rank samples from anomalous to prototypical. Given replay budget B and F observed classes, each class receives $b = \max(1, \lfloor B/F \rfloor)$ samples. For class f, let $\pi = \text{argsort}(\text{IF}(f))$; the replay set $\mathcal{B}^{(f)} = \text{interleave}(\pi_{1:\lfloor b/2 \rfloor}, \pi_{n-\lceil b/2 \rceil+1:n})$ therefore preserves both anomalous and prototypical samples under a fixed total budget.

Machine Unlearning: Given a forget set U_t, machine unlearning reduces the influence of selected samples while preserving behavior on the retain set $E_t = D_t \setminus U_t$. Approaches range from retraining on E_t to approximate procedures that remove the influence of U_t without full retraining. We instantiate these alternatives within L2U in Sec. 4.

3.2 PROBLEM STATEMENT

Our goal is to develop an unlearning framework that improve continual learning components specialized in minimizing catastrophic forgetting, by learning *which* samples to forget and *how* to forget them, targeting samples that are stale, redundant, or poisoned. In doing so, we aim to maximize classification accuracy across previous and future tasks, or groups of data. Given a batch of data D_t at task t, our goal is to identify the optimal samples x per class f for inclusion in the forget set U_t and replay buffer R_t, such that after t tasks the classifier C_t performs well across all previous tasks' test samples, and is capable of good future adaptation. Our framework learns what samples to unlearn and how to unlearn them using characteristics directly derived from the data distribution, through (i) HEURISTIC-L2U and (ii) META-L2U. We evaluate our framework on class incremental and poisoned domain-incremental settings.

4 LEARNING TO UNLEARN (L2U)

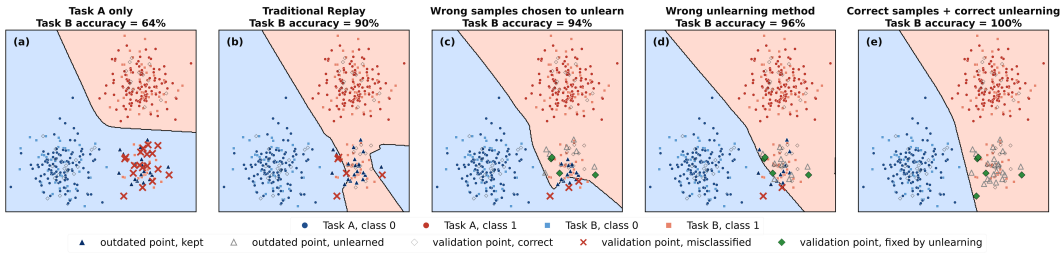


Figure 1: Motivation for learning to unlearn under distribution shift. Traditional replay retains outdated samples, while effective adaptation requires identifying *which* samples to unlearn and *how* to unlearn them.

Motivation: As distributions evolve, the decision boundary learned from earlier data (Fig. 1a) must adapt to the new distribution. Traditional replay improves adaptation but retains outdated samples that can conflict with the new boundary, and may also store redundant or poisoned samples to its detriment(b). Effective adaptation requires identifying the right information to discard (c) and how to remove its effect (d), enabling a better-adapted boundary while retaining useful knowledge (e).

Our L2U framework improves continual adaptation (Fig. 1b) with a *forget phase* that determines *which* samples to forget and *how* to remove their effect. We address both natural and adversarial distribution drift, introducing heuristic and learned mechanisms for selecting samples to unlearn. Fig. 3 provides an overview of this L2U framework. Each task consists of four stages: (1) distribution-aware continual adaptation, (2) forget-set selection, (3) application of an unlearning strategy, and (4) post-unlearning evaluation. We consider two approaches for selecting the forget set: HEURISTIC-L2U, which uses heuristic or lightly supervised selection, and META-L2U, which makes the selection strategy learnable through meta-learning.

Adversarial Drift: Long-Range Poisoning Threat Model: Beyond natural drift, any domain may also encounter adversarial drift: deliberate changes to the training data. Our threat model builds on prior work showing that model-update events can be exploited by delayed-effect poisoning attacks (Di et al., 2023).To show our framework retains performance under poisoned data, we test it on samples altered to appear as a different class. These mutated samples shift the learned decision boundary, creating "pockets" of benign regions within malicious regions and vice versa (Fig. 2).

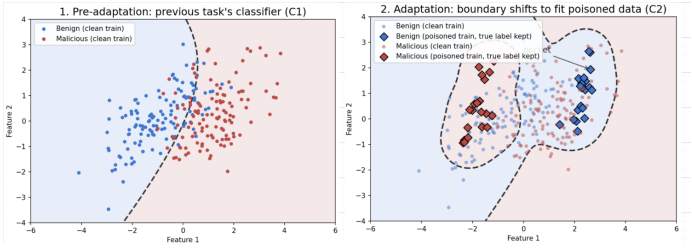


Figure 2: Long-range poisoning threat model. Poisoned training samples distort the decision boundary, creating *pockets* that can later be exploited through small inference-time perturbations.

At inference time, current or previously observed samples can then be slightly perturbed to fall into these pockets and evade the classifier. This assumes a threat model in which the adversary is not limited to a single static snapshot of the classifier, but can infer *when* an organization updates or retrains its system. One way to do this is by observing shifts in prediction confidence or decision boundaries across queries made before and after an adaptation cycle (Salem et al., 2020; Hui et al., 2023).

Under this model, the adversary times their perturbations to coincide with a detected or anticipated retraining event, formulating test-time perturbations that exploit the newly updated decision boundary. We refer to this delayed effect as a *long-range poisoning attack*; attack details are provided in the Appendix G. Since perturbed attack samples must remain valid, we use verified perturbation methods (Hore et al., 2025), **restricting** this evaluation to network datasets rather than malware.

4.1 DISTRIBUTION-AWARE CONTINUAL ADAPTATION

Task Adaptation: Let C_{t-1} denote the classifier obtained after task $t-1$. When the current task's data D_t arrives, we adapt C_{t-1} using D_t combined with the replay buffer data $R_t - 1$ retained from previous tasks, producing the adapted classifier C_t. This constitutes the standard continual adaptation step before selective unlearning is performed.

Distribution-Aware Replay: After adaptation, the replay buffer is updated using the distribution-aware replay mechanism described in Sec. 3.1. Briefly, samples are ranked per class using IF scores in the classifier's latent representation, and the fixed replay budget is divided between prototypical and anomalous samples. We denote the resulting replay buffer at task t as R_t. We maintain a fixed replay budget B throughout the task sequence for a fair evaluation: changes in performance can therefore be attributed to selective unlearning rather than to changes in the amount of historical information available to the classifier.

Pre-Unlearning Evaluation: We evaluate C_t before the forget phase to establish its performance after standard continual adaptation. At this point, the current task provides 3 inputs to the subsequent forget-set selection stage: the current data D_t, the adapted classifier C_t, and the replay buffer R_t.

We adopt this CL framework as our baseline adaptation method because it performs strongly on data with unique, ever-changing distributions across multiple forms of incremental learning. (Rahman et al., 2025) build on distribution-aware replay to continually learn on cybersecurity data, outperforming their benchmark frameworks. Distribution-aware replay is especially valuable in settings where distributions can shift both gradually and suddenly (Lemke et al., 2024; Asadi et al., 2026).

4.2 FORGET-SET SELECTION: WHAT TO UNLEARN

At each task, L2U constructs a forget set U_t containing samples whose influence should be removed from the classifier. We investigate two approaches to this selection problem. HEURISTIC-L2U specifies the selection criterion explicitly using distribution-aware heuristics or limited supervision, whereas META-L2U learns the selection criterion from experience across tasks.

4.2.1 HEURISTIC-L2U: HEURISTIC SELECTION

HEURISTIC-L2U provides the following selectors corresponding to the two forms of drift considered in this work:

Donut Selection: Similar to (Xia et al., 2022), which selects samples closest to the median distance from the class centroid, we target samples that lie between the most anomalous and most prototypical per class. We use the same Isolation Forest ranking that underlies distribution-aware replay, but apply it to the complementary question of what should be forgotten. For a candidate set of n samples, let π denote the ordering induced by the Isolation Forest score and let k denote the forget budget. Donut selects the middle region of the ordering $U_t = \pi_{j:j+k}$, $j = \left\lfloor \frac{n}{2} \right\rfloor - \left\lfloor \frac{k}{2} \right\rfloor$. The most anomalous and most prototypical samples are retained, while samples lying between these two extremes are candidates for forgetting. We visualize this forget set selection method as the middle band of the data distribution, hence the name 'donut'. Under the long-range poisoning threat model, the *defensive objective* of L2U extends beyond removing outdated or redundant information: it must also identify and unlearn poisoned information before it can facilitate future evasion.

Informed Selection: We use a lightweight classifier to learn which samples should be forgotten. Under adversarial drift, we provide a small number of annotated clean and poisoned samples. Since successfully poisoned samples tend to exhibit similar characteristics, this limited supervision allows the selector to identify additional suspicious samples. Samples predicted for removal form U_t.

4.2.2 META-L2U: META-LEARNED SELECTION

The effectiveness of HEURISTIC-L2U depends on specifying an appropriate selection rule for the type of drift being encountered. We therefore make the selection strategy itself *learnable*. META-L2U replaces the fixed selection criterion with a meta-learned policy that determines which samples should be retained or forgotten based on their relationship to the current and historical data distributions.

Distribution-Aware Features: For each candidate sample x, we construct a feature representation $\phi_t(x)$ describing its relationship to the data observed at task t. These features include distribution-aware signals such as the sample's Isolation Forest score and its relationship to the reference or replay distribution. Unlike HEURISTIC-L2U, no single feature directly determines whether a sample is forgotten; instead, they provide evidence to the learned selection policy.

Learnable Selection Strategy: The selection policy g_ϕ maps the distribution-aware representation of a candidate sample to a decision indicating whether that sample should be remembered or forgotten, where $z_x = g_\phi\big(\phi_t(x)\big)$ and $z_x \in \{\text{REMEMBER}, \text{FORGET}\}$. The samples assigned to FORGET constitute the task-specific forget set, $U_t = \{x : g_\phi(\phi_t(x)) = \text{FORGET}\}$.

This formulation allows the same selection mechanism to operate under both natural and adversarial drift rather than requiring a manually specified selection rule for each setting.

Meta-Training: META-L2U replaces the fixed selection rule with a learned scorer g_ϕ. Each candidate sample is represented by seven features capturing *family position* (raw and latent isolation scores, and centroid distance), *model confidence* (cross-entropy loss, predictive entropy, and top-two class margin), and *context* (log family size). The scorer g_ϕ is a lightweight neural network that ranks samples from the same candidate pool used by *donut selection*, with the top $\rho\%$ forming the forget set U_t. We optimize the scorer using gradient-free Evolution Strategies (ES) (Salimans et al., 2017), which perturbs the scorers weights in $pop = 8$ directions per generation. Each perturbation runs on 1 full inner-loop which is an entire run on our CL setup Appendix B.2. A singular mean acc comes back from each and is ranked so that the weights are moved towards the best one. We run ES for 5 generations, then freeze the learned scorer and evaluate it over three random seeds. Under adversarial drift, g_ϕ is additionally warm-started using a small annotated set of clean and poisoned samples before selection. Implementation details and annotation-budget sensitivity are provided in Appendix B.6 and Appendix A, respectively.

4.3 Applying the Forgetting Strategy: How to Unlearn

We use three unlearning strategies: approximate unlearning which aims to maximize the classifier's entropy on a sample (Chundawat et al., 2023), (Golatkar et al., 2020), row dropping (Bourtoule et al., 2021) and label flipping (Chen et al., 2023a). However, the overall methodology is agnostic to the specific unlearning approach. Given a forget set U_t, we apply one of these three strategies while keeping U_t and R_t fixed, where $E_t = D_t \setminus U_t$ is the retain set: **Approximate Unlearning:** Each sample in U_t is treated in a manner that pushes the classifier toward maximal uncertainty, either by distilling knowledge from an incompetent teacher or by training on alternating random labels, rather than by exactly reversing the parameter updates it induced. More details are provided in the appendix (B.1). **Dropped Rows:** Samples in U_t are excluded, and the classifier is further trained on $E_t \cup R_t$. **Flip Label:** Each $(x_u, y_u) \in U_t$ is assigned the opposite label $y'_u = 1 - y_u$, and the classifier is further trained on the relabeled forget set, E_t, and R_t. The resulting post-unlearning classifier is denoted C'_t.

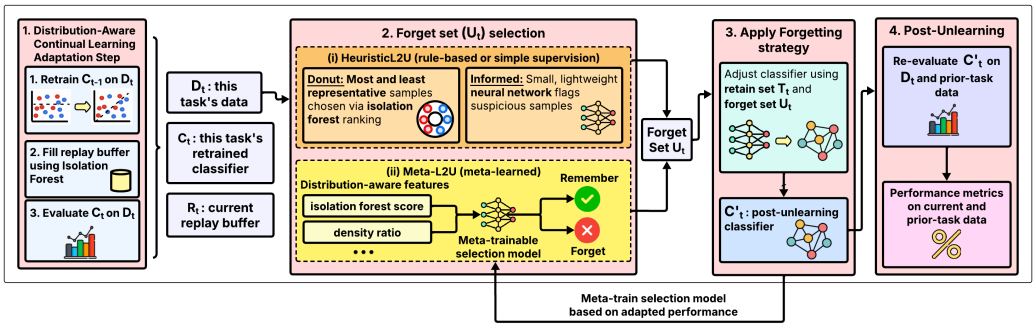


Figure 3: Overview of L2U. Heuristic-L2U selects samples for unlearning using heuristic or lightly supervised rules, while Meta-L2U learns the selection strategy from adaptation performance. The resulting model is evaluated on unseen samples of the current and prior tasks.

4.4 Post-Unlearning Evaluation

We evaluate the post-unlearning classifier C'_t on both the current task and previously observed tasks. Performance on D_t measures whether selective forgetting improves adaptation to the current distribution, while evaluation on prior-task data measures whether this adaptation comes at the expense of previously acquired knowledge. Comparing C'_t against the pre-unlearning classifier C_t therefore captures both the immediate benefit of unlearning and its effect on catastrophic forgetting. For Meta-L2U, this post-unlearning performance additionally provides the feedback used to meta-train the selection policy.

5 Experiments and Results

We evaluate L2U around three questions motivated by the two forms of data distribution drift considered in this work:

- ***RQ1:*** *What role does learned selective unlearning play in improving continual adaptation under natural distribution drift?*
- ***RQ2.1:*** *How well does learned selective unlearning mitigate adversarial distribution drift caused by long-range poisoning?*
- ***RQ2.2:*** *In adversarial distribution drift, how does the choice of samples selected for unlearning affect model performance?*
- ***RQ3:*** *How does shrinking model capacity constrain the benefits selective unlearning can provide?*

Experimental Setup: We evaluate RQ1 in a class-incremental setting, where classes are introduced sequentially and task identity is unavailable at inference, using three malware corpora, EMBER 2018, EMBER 2024, and LAMDA, plus Tiny ImageNet as a non-cybersecurity domain. RQ2.1 is evaluated in a poisoned domain-incremental setting on CICIDS 2017 and Tiny ImageNet; while RQ2.2's evaluation focuses on CICIDS 2017. RQ3 is evaluated on both domain- and class-incremental settings on CICIDS 2017 and EMBER 2018, following RQ1's setup. We compare against Naive (CL

where we only adapt to the current task) and Joint training (CL where we adapt to all data seen so far at the current task) and prior CL methods including SI, A-GEM, DEDUCE, SSF, GCR, MalCL where applicable, and MADAR. All experiments use three seeds, with Balanced Mean Accuracy across tasks seen so far as our primary metric, capturing performance throughout the continual adaptation pipeline as done in (Wang et al., 2026), (Chaudhry et al., 2018), and (Rahman et al., 2025).

Detailed dataset construction, architectures, training schedules, hyperparameters, and additional results are provided in Appendix B.

Table 1: Mean accuracy (%) per dataset under natural distribution drift. MalCL is not evaluated on Tiny-IN because it is incompatible with image dataset.

Group	Method	EMBER'18	EMBER'24	LAMDA	Tiny-IN
Baselines	Joint	85.863 ± 0.156	78.313 ± 0.097	85.950 ± 0.054	48.487 ± 1.444
	Naive	11.478 ± 0.086	15.269 ± 1.542	10.804 ± 0.294	20.439 ± 0.293
Prior Work	SI	54.317 ± 1.468	34.741 ± 3.422	18.662 ± 0.753	17.058 ± 0.371
	GCR	70.765 ± 1.752	65.670 ± 0.349	54.332 ± 0.773	16.629 ± 1.617
	MalCL	74.236 ± 3.159	55.687 ± 1.703	53.280 ± 13.376	—
	MADAR	79.343 ± 0.796	70.149 ± 0.347	71.838 ± 0.590	24.092 ± 0.363
	A-GEM	73.790 ± 0.910	65.955 ± 0.223	61.699 ± 0.419	18.912 ± 0.401
	DEDUCE	77.227 ± 0.762	65.955 ± 0.364	67.079 ± 1.787	19.793 ± 0.416
L2U	HEURISTIC-L2U	80.312 ± 0.747	70.968 ± 0.217	74.160 ± 0.144	24.125 ± 0.401
	META-L2U	80.556 ± 0.723	71.184 ± 0.235	75.290 ± 1.900	26.000 ± 0.259

5.1 RQ1: WHAT ROLE DOES LEARNED SELECTIVE UNLEARNING PLAY IN IMPROVING CONTINUAL ADAPTATION UNDER NATURAL DISTRIBUTION DRIFT?

Evaluation: We first evaluate whether introducing selective forgetting into continual adaptation improves performance under naturally evolving distributions. We compare HEURISTIC-L2U against continual-learning baselines across the four Class-IL datasets. MADAR provides the direct reference for measuring the gain introduced by selective unlearning because it follows the comparable memory-constrained replay setting without the L2U forget phase.

Finding: ***Selective unlearning improves continual adaptation across all three security datasets.*** As shown in Table 1, the natural shifts in these datasets are gradual, making continued adaptation to the evolving distribution important. Consequently, more plastic CL methods that successfully incorporate the current distribution generally outperform more stable methods that preserve historical knowledge at the expense of adaptation. HEURISTIC-L2U improves upon MADAR across the security datasets, showing that selectively removing outdated information further facilitates adaptation to gradual drift. META-L2U strengthens this effect by learning the selection criterion from adaptation performance, consistently outperforming the heuristic variant across all datasets. The smaller gains on Tiny ImageNet suggest that the benefit is most pronounced when historical information becomes progressively less representative of the current distribution. The gains from HEURISTIC-L2U are also significant on every malware corpus: on EMBER 18 ($p = 0.007$), on EMBER 24 ($p = 0.013$) and +2.323 on LAMDA ($p = 0.028$), with every seed improving in every corpus. META-L2U follows suit on EMBER 18 ($p = 0.0179$), EMBER 24 ($p < 0.001$) and LAMDA ($p = 0.0016$).

5.2 RQ2.1: HOW WELL DOES LEARNED SELECTIVE UNLEARNING MITIGATE ADVERSARIAL DISTRIBUTION DRIFT CAUSED BY LONG-RANGE POISONING?

Evaluation: We next evaluate L2U under the long-range poisoning threat model, where poisoned samples introduced during earlier tasks distort the learned decision boundary and can later facilitate inference-time evasion. We evaluate HEURISTIC-L2U using the informed selector with each of the three unlearning strategies on CICIDS 2017 and TinyImageNet, since these support such mutation while preserving their original purpose (e.g., network packets can be subtly altered, unlike static or dynamic malware reports). To ensure that the mutated samples do not lose their malicious characteristics we follow the framework proposed by Deep PackGen's authors (Hore et al., 2025). We compare our framwork against standard CL methods. As shown in Table 2, we report the improvement of each L2U configuration over A-GEM, the strongest prior-work baseline under poisoning. We

Table 2: Performance (%) under long-range poisoning on CICIDS 2017. Prior works (Hore et al., 2025; Rivas & Piplai, 2026) say that validated perturbations are more reliable on CICIDS 2017 than on other comparable datasets.

Group	Method	Mean Acc.	Task 10	Task 1
Baselines	Joint	88.13 ± 6.7	62.20 ± 38.6	95.52 ± 5.0
	Naive	68.13 ± 2.7	91.80 ± 8.5	79.02 ± 2.4
Prior Work	SI	67.03 ± 3.2	91.80 ± 5.0	75.63 ± 8.9
	A-GEM	79.47 ± 7.3	81.47 ± 14.3	80.93 ± 5.5
	SSF	76.93 ± 1.5	18.05 ± 4.5	97.05 ± 3.3
	DEDUCE	71.50 ± 2.9	96.17 ± 0.5	74.17 ± 3.4
	MADAR	57.60 ± 4.8	50.33 ± 3.6	66.48 ± 7.0
L2U	HEURISTIC-L2U: Approximate Unlearning	84.70 ± 4.6	67.50 ± 2.8	93.97 ± 1.8
	HEURISTIC-L2U: Dropped Rows	83.00 ± 2.5	75.63 ± 29.7	92.52 ± 3.3
	HEURISTIC-L2U: Flip Label	83.43 ± 4.6	73.40 ± 14.9	85.95 ± 7.5
	META-L2U Approximate Unlearning	87.09 ± 0.38	87.70 ± 0.62	87.28 ± 1.53
	META-L2U Dropped Rows	87.75 ± 0.95	87.33 ± 0.87	87.85 ± 2.76
	META-L2U Flip Label	86.02 ± 4.23	86.36 ± 2.29	87.28 ± 4.13

additionally examine MADAR because, although it is consistently strong under non-adversarial drift (Table 1), its replay-based adaptation is particularly affected by poisoned historical samples. We find that models that perform well without unlearning in the current setting are more stable and less plastic, making them less susceptible to poisoned data and less likely to adapt to adversarial perturbations.

Finding: ***Selective unlearning reconciles adaptation to natural drift with robustness to adversarial drift.*** The results reveal a reversal from RQ1. Under gradual natural drift, plasticity is beneficial: MADAR is among the strongest prior-work methods, while highly stable methods such as SI adapt poorly. Under poisoning, however, this plasticity becomes a liability. MADAR suffers a substantial performance drop as it adapts to and replays poisoned information, whereas SI remains comparatively competitive despite performing poorly under natural drift. A-GEM and SSF exhibit a similar stability advantage under poisoning, but are less effective at adapting to natural drift. In contrast, HEURISTIC-L2U and META-L2U perform well *across both settings* by preserving the ability to adapt while selectively removing information that becomes outdated or harmful, highlighting the role of unlearning in balancing plasticity and stability when distribution shifts are either natural or adversarial.

5.3 RQ2.2: IN ADVERSARIAL DISTRIBUTION DRIFT, HOW DOES THE CHOICE OF SAMPLES SELECTED FOR UNLEARNING AFFECT MODEL PERFORMANCE?

Evaluation: We evaluate if improvements arise from *which* historical samples are selected for unlearning, rather than simply removing a fixed fraction of historical data. We compare against two selection baselines while keeping the unlearning mechanisms unchanged. *Random* uniformly selects historical samples that are not retained in the replay buffer, without considering their relevance to the current task. *Fixed Heuristic* uses donut, the distribution-based heuristic used in our natural-drift experiments, without adapting the selection criterion to the poisoning setting. For both baselines, we use a fixed selection budget of 30% of the eligible historical samples, matching the proportion of poisoned samples in the adversarial-drift experiments. Each selected set is evaluated with the same three unlearning mechanisms used by L2U: Approximate Unlearning, Dropped Rows, and Flip Label. All results are measured after completing the final task; Mean Accuracy is computed across all tasks, while Task 10 and Task 1 report performance on the current and earliest tasks, respectively.

Table 3: Effect of sample selection under long-range poisoning on CICIDS 2017. Random and Fixed Heuristic (DONUT) use a fixed 30% selection budget. Results are reported after completion of Task 10. The minimum L2U rows show the *lowest performance* achieved by each adaptive L2U selector across its unlearning variants.

Selector	Unlearning	Mean Acc.	Task 10	Task 1
Random	Approximate Unlearning	69.94 ± 2.09	74.10 ± 4.89	55.18 ± 1.58
	Dropped Rows	62.93 ± 0.75	71.02 ± 0.39	52.21 ± 2.35
	Flip Label	61.45 ± 2.76	58.39 ± 8.34	64.35 ± 4.26
Fixed Heuristic (Donut)	Approximate Unlearning	69.33 ± 1.61	75.09 ± 6.56	50.78 ± 3.16
	Dropped Rows	63.21 ± 5.06	69.58 ± 1.99	53.72 ± 7.93
	Flip Label	50.17 ± 10.77	51.25 ± 11.42	46.65 ± 9.61
Adaptive L2U Selection				
Lowest Heuristic L2U	Dropped Rows	80.57 ± 6.30	55.27 ± 30.30	85.95 ± 7.50
Lowest Meta-L2U	Flip Label	86.02 ± 4.23	86.36 ± 2.29	87.28 ± 4.13

Finding: In Table 3, the results demonstrate that *which* samples are selected for unlearning is critical under long-range poisoning. Even the worst-performing Heuristic L2U variant achieves 80.57% mean accuracy, compared with at most 69.94% for random selection and 69.33% for the fixed DONUT heuristic. The gap is even larger for Meta-L2U, whose worst-performing variant achieves 84.04% mean accuracy. ***Surprisingly, the fixed heuristic exhibits even more unstable performance than random selection.*** This supports our earlier observation that heuristics designed for natural distribution drift do not readily transfer to adversarial drift. Under natural drift, distribution-aware heuristics can identify regions of the representation space that have become redundant or obsolete. Under adversarial drift, however, test-time perturbations are not constrained to such fixed regions; an adversary can exploit vulnerable regions that emerge throughout the evolving decision boundary. Consequently, selecting samples from a predetermined region of the representation space provides little indication of which historical samples should be unlearned, underscoring the importance of *adaptive sample selection*.

5.4 RQ3: How does shrinking model capacity constrain the benefits selective unlearning can provide?

Evaluation: Our previous findings highlight the stability–plasticity trade-off in continual adaptation. Model capacity is closely tied to this trade-off: as capacity decreases, a model has fewer degrees of freedom to preserve previously learned knowledge while adapting to new distributions. Prior work (Chen et al., 2026) shows that limited capacity can restrict continual adaptation, raising an important question for L2U: *does selective unlearning remain beneficial when the underlying model has substantially less capacity?* We evaluate this by varying the classifiers from corresponding to 25%, 50%, 75%, and 100% of the default architecture's parameters. We compare L2U against MADAR, the most plastic, and DEDUCE, a more stable strategy. We report combined accuracy after the final continual-learning task. We report results using Approximate Unlearning as the representative unlearning mechanism for both Heuristic L2U and Meta-L2U, with results for the other unlearning mechanisms provided in Appendix E.3. We additionally evaluate reduced model capacity under natural drift on EMBER 2018; these results, together with complete per-mechanism results, are reported in Appendix E.3.

Finding: ***Selective unlearning remains effective even under severely constrained model capacity.*** While MADAR degrades substantially under reduced capacity, L2U maintains a large advantage even at 25% capacity. In particular, Meta-L2U remains remarkably stable across architectures, suggesting that effective historical sample selection can reduce the dependence of continual adaptation on model capacity. L2U also consistently outperforms the more conservative DEDUCE, showing that robustness need not come at the cost of adaptation. We observe the same trend under natural drift on EMBER 2018 (Appendix E.3), where L2U maintains a stable improvement over all baselines as capacity decreases. Overall, these results indicate that L2U's benefit is complementary to model capacity rather than dependent on excess capacity.

Table 4: Final-task combined accuracy (%) across model capacities under adversarial drift. L2U results use Approximate Unlearning unlearning.

Capacity	DEDUCE	MADAR	Heuristic L2U	Meta-L2U
25%	71.67 ± 5.80	48.80 ± 4.40	74.37 ± 4.10	83.19 ± 1.49
50%	67.63 ± 7.70	56.37 ± 8.80	80.63 ± 0.80	84.72 ± 2.64
75%	62.07 ± 3.40	60.80 ± 4.70	82.37 ± 6.30	85.80 ± 0.61
100%	71.50 ± 2.90	57.60 ± 4.80	84.70 ± 4.60	87.09 ± 0.38

6 Conclusion

We presented L2U, a continual adaptation framework that incorporates selective unlearning by explicitly considering both *what* information should be forgotten and *how* its effect should be removed. We studied this problem under natural distribution drift and adversarial drift induced by long-range poisoning. Our results show that selective unlearning improves continual adaptation across evolving distributions, mitigates the persistent effects of poisoned historical samples, and remains effective as model capacity decreases. These findings demonstrate that forgetting outdated or harmful information can be an important component of continual adaptation, improving upon conventional approaches that primarily focus on preserving past knowledge.

AI USE STATEMENT

In this work, we used generative AI tools for cosmetic changes, grammar checks, formatting, retrieving papers, and small code prototyping. All AI-assisted work was reviewed by the authors. We have not used generative AI tools for drafting sections of the paper, research ideation or execution, generating synthetic datasets, or for proving mathematical claims. The ideas and implementations are solely owned by the authors of the paper.

REPRODUCIBILITY STATEMENT

We provide an anonymous GitHub repository containing the implementation and experimental configurations, along with instructions for reproducing our experiments. Detailed reproduction instructions and dataset-specific details are provided in the Appendix. The per-seed results are in Appendix I.The repository's `README.md` provides additional implementation and configuration details. https://anonymous.4open.science/r/Meta-L2U-61F7/README.md

REFERENCES

Seyfullah Arslan and Firat Aydemir. Ultra: A standardized certification metric for machine unlearning. *IEEE Access*, 2026.

Zahra Asadi, Haeseung Jeon, Sohyun Han, Md Mahmuduzzaman Kamol, Se Eun Oh, and Mohammad Saidur Rahman. Freemoca: Memory-free continual learning for malicious code analysis. *arXiv preprint arXiv:2605.09664*, 2026.

Djohan Bonnet, Kellian Cottart, Tifenn Hirtzlin, Tarcisius Januel, Thomas Dalgaty, Elisa Vianello, and Damien Querlioz. Bayesian continual learning and forgetting in neural networks. *Nature Communications*, 16(1):9614, 2025.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE symposium on security and privacy (SP)*, pp. 141–159. IEEE, 2021.

Supriyo Chakraborty and Krishnan Raghavan. On understanding of the dynamics of model capacity in continual learning. *arXiv preprint arXiv:2508.08052*, 2025.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018.

Min Chen, Weizhuo Gao, Gaoyang Liu, Kai Peng, and Chen Wang. Boundary unlearning: Rapid forgetting of deep networks via shifting the decision boundary. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7766–7775. IEEE, 2023a.

Yitian Chen, Shigeng Zhang, Xuan Liu, Mingming Lu, Kai Chen, Hongye Zhu, and Xinning Chen. Parameter-efficient continual learning for enhancing plasticity without forgetting under limited model capacity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10789–10798, 2026.

Yizheng Chen, Zhoujie Ding, and David Wagner. Continuous learning for android malware detection. In *32nd USENIX Security Symposium (USENIX Security 23)*, pp. 1127–1144, 2023b.

Vikram S Chundawat, Ayush K Tarun, Murari Mandal, and Mohan Kankanhalli. Can bad teaching induce forgetting? unlearning in deep networks using an incompetent teacher. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 7210–7217, 2023.

Jimmy Z Di, Jack Douglas, Jayadev Acharya, Gautam Kamath, and Ayush Sekhari. Hidden poison: Machine unlearning enables camouflaged poisoning attacks. In *NeurIPS*, 2023.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37, 2014.

Hassan Gharoun, Fereshteh Momenifar, Fang Chen, and Amir H Gandomi. Meta-learning approaches for few-shot learning: A survey of recent advances. *ACM Computing Surveys*, 56(12):1–41, 2024.

Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9301–9309, 2020. doi: 10.1109/CVPR42600.2020.00932.

Jianyang Gu, Kai Wang, Wei Jiang, and Yang You. Summarizing stream data for memory-constrained online continual learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 12217–12225, 2024.

Soumyadeep Hore, Jalal Ghadermazi, Diwas Paudel, Ankit Shah, Tapas Das, and Nathaniel Bastian. Deep packgen: A deep reinforcement learning framework for adversarial network packet generation. *ACM Transactions on Privacy and Security*, 28(2):1–33, 2025.

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9): 5149–5169, 2021.

Tian Hui, Farhad Farokhi, and Olga Ohrimenko. Information leakage from data updates in machine learning models. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, pp. 35–41, 2023.

Meghdad Kurmanji, Peter Triantafillou, Jamie Hayes, and Eleni Triantafillou. Towards unbounded machine unlearning. *Advances in neural information processing systems*, 36:1957–1987, 2023.

Nick Lemke, Camila González, Anirban Mukhopadhyay, and Martin Mundt. Distribution-aware replay for continual mri segmentation. In *International Workshop on Personalized Incremental Learning in Medicine*, pp. 73–85. Springer, 2024.

Fan Liu, Tianshu Zhang, Wenwen Dai, Chuanyi Zhang, Wenwen Cai, Xiaocong Zhou, and Delong Chen. Few-shot adaptation of multi-modal foundation models: a survey: F. liu et al. *Artificial Intelligence Review*, 57(10):268, 2024.

Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7765–7773. IEEE, 2018.

Quoc Phong Nguyen, Ryutaro Oikawa, Dinil Mon Divakaran, Mun Choon Chan, and Bryan Kian Hsiang Low. Markov chain monte carlo-based machine unlearning: Unlearning what needs to be forgotten. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, pp. 351–363, 2022.

Jimin Park, AHyun Ji, Minji Park, Mohammad Saidur Rahman, and Se Eun Oh. Malcl: Leveraging gan-based generative replay to combat catastrophic forgetting in malware classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 658–666, 2025.

Mohammad Saidur Rahman, Scott Coull, Qi Yu, and Matthew Wright. Madar: Efficient continual learning for malware analysis with distribution-aware replay. In *Conference on Applied Machine Learning for Information Security*, pp. 265–291. PMLR, 2025.

Emilia Rivas and Aritran Piplai. Contrastive adversarial agents for intentional drift induction in network intrusion detection systems. In *Proceedings of the AAAI Symposium Series*, volume 9, pp. 309–314, 2026.

Ahmed Salem, Apratim Bhattacharya, Michael Backes, Mario Fritz, and Yang Zhang. Updates-Leak: Data set inference and reconstruction attacks in online learning. In *29th USENIX Security Symposium (USENIX Security 20)*, pp. 1291–1308. USENIX Association, August 2020. ISBN 978-1-939133-17-5. URL `https://www.usenix.org/conference/usenixsecurity20/presentation/salem`.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International conference on machine learning*, pp. 4528–4537. PMLR, 2018.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

Ayush K Tarun, Vikram S Chundawat, Murari Mandal, and Mohan Kankanhalli. Fast yet effective machine unlearning. *IEEE transactions on neural networks and learning systems*, 35(9):13046–13055, 2023.

Rishabh Tiwari, Krishnateja Killamsetty, Rishabh Iyer, and Pradeep Shenoy. Gcr: Gradient coreset based replay buffer selection for continual learning. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 99–108. IEEE, 2022.

Liyuan Wang, Mingtian Zhang, Zhongfan Jia, Qian Li, Chenglong Bao, Kaisheng Ma, Jun Zhu, and Yi Zhong. Afec: Active forgetting of negative transfer in continual learning. *Advances in Neural Information Processing Systems*, 34:22379–22391, 2021.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE transactions on pattern analysis and machine intelligence*, 46(8):5362–5383, 2024.

Yiwen Wang, Diana Benavides-Prado, and Yun Sing Koh. Detect, decide, unlearn: A transfer-aware framework for continual learning. In *International Conference on Learning Representations*, volume 2026, pp. 115734–115757, 2026.

Xiaobo Xia, Jiale Liu, Jun Yu, Xu Shen, Bo Han, and Tongliang Liu. Moderate coreset: A universal method of data selection for real-world data-efficient deep learning. In *The Eleventh International Conference on Learning Representations*, 2022.

Jie Xu, Zihan Wu, Cong Wang, and Xiaohua Jia. Machine unlearning: Solutions and challenges. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 8(3):2150–2168, 2024.

Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pp. 3987–3995. Pmlr, 2017.

Xinchen Zhang, Running Zhao, Zhihan Jiang, Handi Chen, Yulong Ding, Edith CH Ngai, and Shuang-Hua Yang. Continual learning with strategic selection and forgetting for network intrusion detection. In *IEEE INFOCOM 2025-IEEE Conference on Computer Communications*, pp. 1–10. IEEE, 2025.

Indrė Žliobaitė, Albert Bifet, Bernhard Pfahringer, and Geoffrey Holmes. Active learning with drifting streaming data. *IEEE transactions on neural networks and learning systems*, 25(1):27–39, 2013.

Overview of Appendix

A Sensitivity to Sample Budget for Informed Adaptation **13**

B Extended Setup **14**

C Extended Natural-Drift Results **17**

D Adversarial Drift: Ablation **18**

E Complete Model-Capacity Results **19**

F Training attack details **20**

G Inference time attack details **20**

H Retention Test and Adaptability **21**

I Reproducibility: Per Seed Results **26**

A Sensitivity to Sample Budget for Informed Adaptation

Experiment: We evaluate the sensitivity of L2U to the number of labeled samples available to train its adaptive sample-selection classifier. Specifically, we vary the number of examples available per class (poisoned and benign) from 60 to 40, 25, and 20 samples. As discussed in our adversarial-drift experiments, adaptation is necessary because the characteristics of poisoned samples can change substantially across tasks. This experiment therefore measures how little task-specific supervision is sufficient for L2U to reliably identify samples that should be unlearned. All other experimental settings remain unchanged.

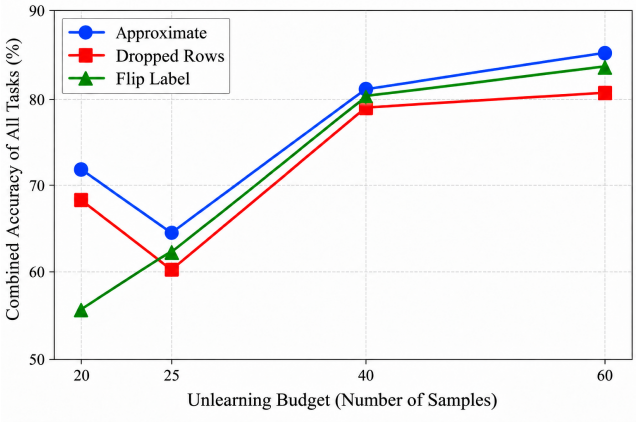


Figure 4: Sensitivity of HEURISTIC-L2U to the number of labeled samples per class (poisoned and benign) available to the adaptive sample-selection classifier. We report the combined accuracy across all tasks after completing the final task for each unlearning mechanism.

Finding: Figure 4 shows that L2U remains effective with substantially less supervision than the default 60 samples per class. With 40 samples per class, all three unlearning mechanisms retain strong combined accuracy, indicating that a relatively small labeled set is sufficient to characterize the task-specific poisoning pattern. This is consistent with the structure of our threat model: although

poisoning patterns can differ considerably across tasks, poisoned samples within a given task tend to exhibit similar characteristics, allowing the adaptive selector to learn the relevant pattern from a limited number of examples. Performance becomes less stable below 40 samples per class and degrades more noticeably at 25 samples or fewer, where the small labeled set no longer provides a sufficiently reliable characterization of the task-specific poisoning pattern. Nevertheless, the strong performance at 40 samples per class shows that L2U does not require extensive supervision to outperform competing continual-adaptation baselines.

B Extended Setup

This appendix provides the dataset construction, architectures, training protocols, hyperparameters, baseline configurations, and extended results corresponding to Sec. 5. Unless otherwise stated, all configurations are evaluated over three seeds $\{1, 2, 3\}$, and the base task is independently retrained for each seed.

B.1 Approximate Unlearning Forgetting Strategy: Details

Teacher-Student Approximate Unlearning: Here we use a teacher-student framework similar to Chundawat et al. (2023) as a means to maximize entropy. Let U_t be the forget set at task t, T_t the retained task data, R_t the replay buffer, and θ^- the pre-unlearning snapshot. Each step pairs a minibatch from U_t with one from $T_t \cup R_t$ and descends $\mathcal{L}_U = \alpha\,\mathrm{KL}(u \,\|\, p_\theta) + (1-\alpha)\big(\frac{1}{2}\mathcal{L}_{\mathrm{CE}} + \frac{1}{2}\mathcal{L}_{\mathrm{KD}}(\cdot\,;\theta^-)\big) + c\,\Omega(\theta)$, where u is uniform over the candidate classes and Ω is a Synaptic Intelligence penalty anchored at the start of the task, so the forget and retain pressures apply simultaneously rather than in separate passes. The buffer is then rebuilt from T_t alone. This yields an *approximate suppression* of U_t's influence on the parameters alongside its exact removal from replay.

Random Labeling: Following the random-label baseline of Golatkar et al. (2020), we induce uncertainty by having the classifier train against contradictory labels for each forget sample. Each $x_u \in U_t$ is duplicated under both class labels, forming $U_t^{\pm} = \{(x_u, y_{\mathrm{benign}}) : x_u \in U_t\} \cup \{(x_u, y_{\mathrm{mal}}) : x_u \in U_t\}$. We then alternate training on $U_t^{\pm}$ and $E_t \cup R_t$, suppressing the model's confidence on forgotten samples while retaining knowledge from the remaining data.

B.2 Datasets and Continual-Learning Settings

Class-Incremental Setting: For EMBER 2018, EMBER 2024, LAMDA, and Tiny ImageNet, we use a Class-IL protocol in which a base task is followed by increments containing previously unseen classes. Task identity is not available at inference. For the malware datasets, families are ranked by training frequency, with ties broken by ascending original identifier. Consequently, the first N families of a shorter schedule are identical to the corresponding families in a longer schedule.

Table 5: Datasets and Class-IL task schedules. The notation $b{+}s{\times}n$ denotes a base task containing b classes followed by n increments of s previously unseen classes.

Dataset	Schedule	Base	Step	Tasks	Classes
EMBER 2018	$50{+}5{\times}10$	50	5	11	100
EMBER 2024	$50{+}5{\times}10$	50	5	11	100
LAMDA	$30{+}5{\times}10$	30	5	11	80
Tiny ImageNet	$20{+}20{\times}9$	20	20	10	200

EMBER 2018 and EMBER 2024 use the 2381-dimensional EMBER feature representation. Features are standardized using statistics estimated only from the base task and clipped to ± 10. LAMDA contains 4561 binary features and is left unscaled. Malware families with fewer than 200 training samples are removed. EMBER 2024 is additionally capped at 10,000 samples per family to reduce its 94:1 class imbalance; the other malware corpora are uncapped.

Tiny ImageNet contains 200 classes with 500 training images per class at 64×64 resolution. Since its distributed test split is unlabeled, we use its 10,000-image validation split for evaluation. Per-channel normalization statistics are estimated using only the base task, following the same principle as the malware experiments.

Poisoned Tiny ImageNet: Classes are introduced across ten sequential tasks in a class-incremental schedule, with 20 new classes activated per task (task 0 included) until all 200 are in use. In each task after the first, 30% of the training samples of each newly introduced class are poisoned, shifted toward its nearest already-seen class by raw-pixel centroid distance. The test-time attack uses an ℓ_2 perturbation budget of 0.15 times the mean nearest-neighbor centroid distance among that task's active classes, and the replay buffer holds 4000 samples total, split evenly across every class seen so far. For the unlearning variants, the replay buffer excludes samples flagged as poisoned by the perturbation detector; the MADAR baseline's buffer is unfiltered

Domain-Incremental Setting: For CICIDS 2017, the class space (benign vs. malicious) remains fixed while the input distribution changes across ten sequential tasks. All records are pooled and ordered by timestamp; the first task contains 30% of the data, and the remainder is split across the following nine tasks in decreasing shares (9.2% down to 6.4%), each with a 20% held-out test split. In each task after the first, 30% of the training samples of each class are poisoned. The test-time attack uses an ℓ_2 perturbation budget of 0.15 times the mean benign-to-malicious centroid distance of that task, and the replay buffer holds 2000 samples per class. For the unlearning variants, the replay buffer excludes samples flagged as poisoned by the perturbation detector; the MADAR baseline's buffer is unfiltered.

B.3 Architectures

For EMBER and LAMDA, the classifier is a batch-normalized feed-forward network with four hidden layers of widths $(1024, 512, 256, 128)$. The 128-dimensional penultimate layer provides the latent representation used for distribution-aware replay selection. This results in approximately 3.14M parameters for the 2381-dimensional EMBER representation and 5.37M parameters for LAMDA's 4561-dimensional representation.

Tiny ImageNet uses ResNet-18 with a 3×3, stride-one stem and no initial max-pooling. This configuration contains approximately 11.27M parameters and avoids the aggressive early spatial reduction produced by the standard ImageNet stem on 64×64 images.

The default CICIDS classifier follows the four-hidden-layer architecture $(1024, 512, 256, 128)$. For the model-capacity experiment in Sec. 5.4, we additionally evaluate variants containing only two and one hidden layers.

B.4 Training Protocol and Hyperparameters

For the malware Class-IL experiments, the base task is trained for 30 epochs using Adam with learning rate 10^{-3}. Subsequent tasks use SGD with momentum 0.9, weight decay 10^{-6}, and gradients clipped to unit norm. We train each continual task for a fixed number of optimization iterations rather than epochs so that optimization budget does not depend on task size. Replay-based methods draw a memory batch alongside each incoming stream batch. Where knowledge distillation is used, the replay and current-task objectives are weighted using $\mathrm{rnt} = 1/(t+1)$ against a teacher frozen at the previous task boundary. Batch-normalization statistics are frozen during continual adaptation.

Tiny ImageNet trains the base task for 50 epochs using SGD with learning rate 3×10^{-2} and uses the same learning rate during continual adaptation. Table 6 summarizes the dataset-specific training configuration.

Table 6: Dataset-specific training protocol.

	EMBER'18	EMBER'24	LAMDA	Tiny-IN
Base epochs	30	30	30	50
Base LR	10^{-3}	10^{-3}	10^{-3}	3×10^{-2}
Continual iterations	2000	8000	4000	3000
Continual LR	10^{-4}	10^{-4}	10^{-4}	3×10^{-2}
Batch size	256	256	256	32
Memory size	5000	10000	2500	2000
SI strength c	1.0	100.0	100.0	1.0
Per-family cap	–	10000	–	–
Min. family samples	200	200	200	–

Distribution-aware replay fits an Isolation Forest independently for each class in latent space and selects anomalous and prototypical samples under a uniform per-class memory budget. Unless otherwise specified, unlearning uses $\alpha = 0.2$, a forget ratio of 0.10, three epochs, and learning rate 10^{-4}.

B.5 Baselines and Method-Specific Settings

Naive sequentially fine-tunes on each incoming task without a continual learning mechanism, whereas Joint retrains from scratch using all data observed so far. Neither represents a continual-learning solution; they provide reference points for sequential fine-tuning and access to all historical data, respectively.

We compare against Synaptic Intelligence (SI) Zenke et al. (2017), A-GEM (Chaudhry et al., 2018), MalCL (Park et al., 2025), DEDUCE (Wang et al., 2026), GCR in class-IL settings Tiwari et al. (2022), SSF in domain-IL settings Zhang et al. (2025), and MADAR Rahman et al. (2025). A-GEM uses reservoir memory under the corresponding replay budget. MalCL uses 50 epochs per task, $k = 3$ generated replay samples per class, and latent dimension 62. MalCL is not evaluated on Tiny ImageNet because its generator and discriminator operate over one-dimensional feature vectors and do not directly extend to the image setting.

For X-DER, we use $\gamma = 0.75$ and margin $m = 0.3$ as specified in the original work, with $\alpha = 0.3$ and $\beta = 0.2$. We use $\lambda = 0.05$ for future preparation, $\eta = 0.1$ for the past/future constraint, and contrastive temperature $\tau = 0.1$. STAR uses $\lambda = 0.01$, $\gamma = 0.001$, and one ascent step. DEDUCE uses gradient-conflict detection with tolerance 0, $\delta = 10^{-3}$, $\phi = 10^{-5}$, and $\eta = 0.99$.

B.6 Meta-L2U Details

Input: Each candidate sample is represented by seven features: three describing *family position* (raw and latent isolation scores and centroid distance), three describing *model confidence* (cross-entropy loss, predictive entropy, and top-two class margin), and one describing *context* (log family size).

Scorer: A lightweight neural scorer, NN-1, maps these features to a selection score. At the end of each task, NN-1 ranks the same candidate pool considered by *donut selection*, and the top $\rho\%$ of samples form the forget set U_t.

Feedback: We optimize NN-1 using Evolution Strategies (ES) (Salimans et al., 2017). For each candidate scorer, its selected samples are unlearned and the resulting classifier is evaluated on held-out data. This post-unlearning performance serves as the fitness signal: selections that improve adaptation receive higher fitness.

Training: ES maintains a population of 8 candidate scorers and updates NN-1 from their relative fitness. We train for 5 generations, with each generation requiring one complete CL task evaluation per candidate. The resulting scorer is then frozen and evaluated over three random seeds of our CL setup (Appendix B.2).

Cost: Meta-training therefore requires approximately $8 \times 5 = 40$ scorer evaluations, each involving a CL task run. This optimization is performed offline; at evaluation time, NN-1 requires only a forward pass to score and rank candidate samples.

Test-time adaptation in the adversarial setting: Warm start: For each new task, the selector is initialized from the parameters learned on the previous task, ϕ_t, rather than from scratch.

Calibration data: Adaptation uses a small annotated set of 10 samples per inner step, consisting of 5 clean and 5 poisoned samples. Thus, the selector requires only limited supervision from the current distribution.

Adaptation: Starting from ϕ_t, we perform a short MAML-type ES refinement with a population of 8 candidate scorers. Each candidate is evaluated for 3 inner steps using the annotated calibration samples, and its post-unlearning performance provides the fitness signal for the ES update. This process is repeated for 8 outer iterations, $\phi_t \xrightarrow[\text{3 inner steps}]{\text{8 ES iterations}} \phi'_t$.

Continual update: The adapted scorer $g_{\phi'_t}$ determines the forget set U_t for the current task. Its parameters ϕ'_t are then carried forward to warm-start the selector for task $t+1$, allowing the selection mechanism to evolve alongside the continually adapted classifier.

Cost: Each task requires 8 ES iterations with 3 inner evaluations per candidate. We use the lightweight approximate unlearning update during this refinement rather than fully retraining the classifier.

B.7 Evaluation Metrics

Mean Accuracy is the accuracy over all classes observed so far, averaged across tasks and then across seeds, and is our primary metric because it summarizes the complete adaptation trajectory. Final Accuracy measures accuracy over all observed classes after the final task. Task-0 Final measures final performance on the classes present in the initial task and therefore characterizes retention. Recent Final measures final performance on recently introduced classes and characterizes acquisition. We additionally compute Macro F1 over observed classes to account for class imbalance. The main paper reports Mean Accuracy, while additional metrics are reported here where available.

C Extended Natural-Drift Results

The tables 7, 8, 9, 10 provide the per-dataset results corresponding to RQ1. These results complement the consolidated Mean Accuracy comparison reported in Table 1.

Table 7: Extended results on EMBER 2018.

Group	Method	Mean Acc.	Task 10	Task 1
Baselines	Joint	85.863 ± 0.156	84.965	86.185
	Naive	11.478 ± 0.086	0.297	0.000
Prior Work	SI	54.317 ± 1.468	41.541	44.041
	GCR	70.765 ± 1.752	63.908	65.897
	MalCL	74.236 ± 3.159	63.422	66.574
	A-GEM	73.790 ± 0.910	72.210	76.896
	DEDUCE	77.227 ± 0.762	75.689	78.082
	MADAR	79.343 ± 0.796	76.763	80.635
L2U	Heuristic-L2U	80.312 ± 0.747	78.826	82.456
	Meta-L2U	80.556 ± 0.723	78.912	82.563

Table 8: Extended results on EMBER 2024.

Group	Method	Mean Acc.	Task 10	Task 1
Baselines	Joint	78.313 ± 0.097	75.508	80.172
	Naive	15.269 ± 1.542	0.568	0.001
Prior Work	SI	34.741 ± 3.422	8.744	9.187
	GCR	65.670 ± 0.349	58.849	65.979
	MalCL	55.687 ± 1.703	44.574	51.227
	A-GEM	65.955 ± 0.223	57.688	68.167
	DEDUCE	65.955 ± 0.364	59.827	67.254
	MADAR	70.149 ± 0.347	64.582	73.524
L2U	Heuristic-L2U	70.968 ± 0.217	66.027	74.694
	Meta-L2U	71.184 ± 0.235	66.115	74.631

Table 9: Extended results on LAMDA.

Group	Method	Mean Acc.	Task 10	Task 1
Baselines	Joint	85.950 ± 0.054	84.813	86.180
	Naive	10.804 ± 0.294	0.650	0.000
Prior Work	SI	18.662 ± 0.753	0.602	0.029
	GCR	54.332 ± 0.773	43.533	45.258
	A-GEM	61.699 ± 0.419	60.507	68.293
	DEDUCE	67.079 ± 1.787	59.436	62.594
	MADAR	71.838 ± 0.590	64.668	69.996
L2U	HEURISTIC-L2U	74.160 ± 0.144	69.881	75.576
	META-L2U	75.290 ± 1.900	68.200	73.291

Table 10: Extended results on Tiny ImageNet.

Group	Method	Mean Acc.	Task 10	Task 1
Baselines	Joint	48.487 ± 1.444	33.977	38.100
	Naive	20.439 ± 0.293	7.200	0.000
Prior Work	SI	17.058 ± 0.371	5.727	0.000
	GCR	16.629 ± 1.617	7.253	7.800
	A-GEM	18.912 ± 0.401	5.950	4.500
	DEDUCE	19.793 ± 0.416	6.710	66.833
	MADAR	24.092 ± 0.363	9.060	42.033
L2U	HEURISTIC-L2U	24.125 ± 0.401	8.973	41.667
	META-L2U	26.000 ± 0.259	8.755	41.650

D ADVERSARIAL DRIFT: ABLATION

To determine whether poisoned information retained in replay interferes with unlearning, we compare three replay configurations. *Clean* filters detected poisoned samples before they enter replay; *Dirty* allows them to remain in memory; and *Unfilled* removes detected samples post hoc without replacing them.

Table 11: Replay-buffer ablation on perturbed CICIDS 2017.

Buffer	Method	Mean Acc.	Task 10	Task 1
Clean	L2U w/o Unlearning	67.77 ± 3.4	46.07 ± 7.0	80.62 ± 2.8
	HEURISTIC-L2U: Approximate Unlearning	84.70 ± 4.6	67.50 ± 2.8	93.97 ± 1.8
	HEURISTIC-L2U: Dropped Rows	83.00 ± 2.5	75.63 ± 29.7	92.52 ± 3.3
	HEURISTIC-L2U: Flip Label	83.43 ± 4.6	73.40 ± 14.9	85.95 ± 7.5
Dirty	L2U w/o Unlearning	57.60 ± 4.8	50.33 ± 3.6	66.48 ± 7.0
	HEURISTIC-L2U: Approximate Unlearning	70.63 ± 6.2	66.83 ± 21.1	84.43 ± 3.1
	HEURISTIC-L2U: Dropped Rows	67.33 ± 3.4	54.23 ± 15.2	83.35 ± 4.3
	HEURISTIC-L2U: Flip Label	75.33 ± 4.3	92.53 ± 4.7	78.13 ± 4.3
Unfilled	L2U w/o Unlearning	64.60 ± 6.5	51.73 ± 3.9	73.70 ± 8.4
	HEURISTIC-L2U: Approximate Unlearning	85.50 ± 4.3	74.17 ± 29.9	91.30 ± 4.0
	HEURISTIC-L2U: Dropped Rows	83.70 ± 3.5	63.33 ± 11.1	92.55 ± 1.7
	HEURISTIC-L2U: Flip Label	84.10 ± 5.4	68.38 ± 23.9	90.50 ± 2.9

As seen on Table 11, across all three forgetting strategies, retaining poisoned samples in replay reduces Mean Accuracy. However, the most dramatic decrease is caused by *removing the unlearning* component. Interestingly, removing detected samples without replacement performs at least as well as maintaining the clean, filled buffer, suggesting that retaining fewer clean samples is preferable to preserving contaminated replay simply to maintain the memory budget.

E COMPLETE MODEL-CAPACITY RESULTS

E.1 REDUCED CAPACITY UNDER ADVERSARIAL DRIFT

Table 12 reports the complete reduced-capacity experiment. The main paper reports gains relative to MADAR because MADAR provides the directly comparable memory-constrained training regime.

Table 12: Complete reduced-capacity results on perturbed CICIDS 2017.

Capacity	Method	Mean Acc.	Task 10	Task 1
25%	MADAR	48.80 ± 4.4	49.12 ± 5.7	56.62 ± 6.4
	SI	68.27 ± 4.0	94.33 ± 2.4	71.73 ± 7.0
	DEDUCE	71.67 ± 5.8	93.00 ± 0.5	59.57 ± 9.6
	HEURISTIC-L2U: Approximate Unlearning	74.37 ± 4.1	93.02 ± 2.2	70.63 ± 15.1
	HEURISTIC-L2U: Dropped Rows	72.83 ± 4.0	72.42 ± 24.0	83.23 ± 6.9
	HEURISTIC-L2U: Flip Label	70.83 ± 6.3	77.35 ± 20.4	73.53 ± 12.8
	META-L2U: Approximate Unlearning	86.31 ± 0.04	87.79 ± 0.77	83.95 ± 0.35
50%	MADAR	56.37 ± 8.8	51.05 ± 2.5	57.83 ± 10.5
	SI	66.73 ± 1.7	96.18 ± 0.5	73.03 ± 4.8
	DEDUCE	67.63 ± 7.7	93.55 ± 1.3	59.25 ± 7.5
	HEURISTIC-L2U: Approximate Unlearning	80.63 ± 0.8	81.35 ± 11.5	84.05 ± 8.1
	HEURISTIC-L2U: Dropped Rows	80.10 ± 4.2	70.75 ± 25.4	90.00 ± 4.2
	HEURISTIC-L2U: Flip Label	76.17 ± 12.8	79.13 ± 19.5	76.83 ± 13.0
	META-L2U: Approximate Unlearning	82.23 ± 2.64	87.28± 1.16	82.45 ± 1.45
75%	MADAR	60.80 ± 4.7	45.73 ± 8.0	76.23 ± 4.7
	SI	71.60 ± 3.2	96.00 ± 1.1	78.92 ± 6.9
	DEDUCE	62.07 ± 3.4	94.13 ± 2.3	55.03 ± 12.6
	HEURISTIC-L2U: Approximate Unlearning	82.37 ± 6.3	69.20 ± 10.2	92.78 ± 2.0
	HEURISTIC-L2U: Dropped Rows	84.07 ± 4.2	79.85 ± 11.3	92.58 ± 1.8
	HEURISTIC-L2U: Flip Label	78.47 ± 6.0	82.68 ± 5.7	77.10 ± 9.6
	META-L2U: Approximate Unlearning	85.80 ± 0.61	86.68 ± 0.45	85.46 ± 2.14
100%	MADAR	57.60 ± 4.8	50.33 ± 3.6	66.48 ± 7.0
	SI	67.03 ± 3.2	91.80 ± 5.0	75.63 ± 8.9
	DEDUCE	71.50 ± 2.9	96.17 ± 0.5	74.17 ± 3.4
	HEURISTIC-L2U: Approximate Unlearning	84.70 ± 4.6	67.50 ± 2.8	93.97 ± 1.8
	HEURISTIC-L2U: Dropped Rows	83.00 ± 2.5	75.63 ± 29.7	92.52 ± 3.3
	HEURISTIC-L2U: Flip Label	83.43 ± 4.6	73.40 ± 14.9	85.95 ± 7.5
	META-L2U: Approximate Unlearning	87.09 ± 0.38	87.70 ± 0.62	87.28 ± 1.53

E.2 MODEL PERFORMANCE ON ADVERSARIAL NON-CYBERSECURITY DATASET

We test the robustness of our HEURISTIC-L2U pipeline on a non-cybersecurity dataset by challenging it and the same baselines with a poisoned, multiclass version of dataset, evaluating both plasticity and stability under adversarial drift.

As shown in Table 13, all three HEURISTIC-L2U variants outperform every CL baseline except the clean (unpoisoned) upper bound, reaching a Mean Accuracy of 16.47–17.53% compared to 9.00% for MADAR and 5.97–6.27% for SI, AGEM, and DEDUCE—nearly double the strongest baseline and roughly three times the weakest. More importantly, HEURISTIC-L2U is the only approach besides MADAR to retain non-zero accuracy on the first task's test set after 10 tasks (12.70–13.03% vs. MADAR's 6.02%, roughly double), while SI, AGEM, and DEDUCE all collapse to 0.00% on Task 1. This indicates that these three baselines adapt aggressively enough to Task 10 (59.62–62.27% accuracy) to fully overwrite earlier knowledge, whereas HEURISTIC-L2U sacrifices some Task 10 performance (31.10–35.07%) in exchange for substantially better retention of earlier, poisoned-then-cleaned tasks—evidence that our pipeline retains useful information from old tasks while adapting to the best subset of training data and forgetting stale or poisoned samples.

Table 13: Complete results on **perturbed Tiny ImageNet.**

Group	Method	Mean Acc.	Task 10	Task 1
Baselines	Clean	60.57 ± 0.3	66.83 ± 1.2	60.10 ± 0.7
	MADAR	9.00 ± 0.2	17.27 ± 0.8	6.02 ± 0.4
	SI	5.97 ± 0.2	59.62 ± 2.5	0.00 ± 0.0
	AGEM	6.27 ± 0.1	62.27 ± 0.6	0.00 ± 0.0
	DEDUCE	6.17 ± 0.2	61.78 ± 1.4	0.00 ± 0.0
L2U	HEURISTIC-L2U: Approximate Unlearning	17.53 ± 0.7	35.07 ± 3.0	12.70 ± 1.5
	HEURISTIC-L2U: Dropped Rows	17.53 ± 0.5	34.13 ± 2.8	13.03 ± 0.8
	HEURISTIC-L2U: Flip Label	16.47 ± 0.6	31.10 ± 1.3	12.97 ± 2.6

E.3 MODEL CAPACITY UNDER NATURAL DRIFT

Table 14 reports the model-capacity analysis on EMBER 2018 under natural distribution drift. We reduce the default classifier architecture to 50% and 25% of its original capacity while keeping the continual-learning setting unchanged.

Table 14: Mean accuracy (%) on EMBER 2018 across model capacities. Full, Half, and Tiny correspond to 100%, 50%, and 25% of the default classifier capacity, respectively.

Method	100%	50%	25%
Joint	**85.863 ± 0.156**	**85.576 ± 0.199**	**84.545 ± 0.098**
Naive	11.478 ± 0.086	11.563 ± 0.058	10.575 ± 0.572
SI	54.317 ± 1.468	38.912 ± 7.357	18.525 ± 1.918
A-GEM	73.790 ± 0.910	71.760 ± 0.240	65.107 ± 0.561
DEDUCE	77.227 ± 0.762	77.756 ± 0.375	74.411 ± 3.518
MADAR	79.343 ± 0.796	78.655 ± 0.347	75.537 ± 0.595
HEURISTIC-L2U	80.312 ± 0.747	79.359 ± 0.235	76.219 ± 0.429

Selective unlearning consistently improves over its MADAR backbone at all three capacities. Although absolute performance decreases as capacity is reduced, the L2U advantage persists even at 25% capacity. This mirrors the capacity-independent behavior observed under adversarial drift, while the smaller gains on EMBER 2018 are consistent with natural drift containing obsolete or conflicting historical information rather than explicitly poisoned samples.

F TRAINING ATTACK DETAILS

At the start of task t, the attacker uses a small set of benign samples under its control to estimate a benign reference centroid μ_0. Poisoned samples are then shifted toward this benign region in feature space. Rather than arbitrarily modifying network traffic, we use DeepPackGen (Hore et al., 2025), which identifies a constrained set of network features that can be perturbed without altering the packet structure or attack payload. For a poisoning budget α, we perturb $z = \lfloor \alpha |D_t| \rfloor$ malicious samples along these feasible feature dimensions while preserving their original labels and functionality. Consequently, the model observes functional malicious samples whose measured features increasingly resemble the benign distribution. When these samples are incorporated during later adaptation, they can create *pockets* in which previously malicious regions become associated with the benign class.

G INFERENCE TIME ATTACK DETAILS

The motivation here is that adversaries, who can estimate when a model is updated with training data, can actually mutate a sample, by (i) querying a same prior to training, and (ii) ensuring that the same sample lands in a "pocket" created by distorted decision boundaries. However, as we can see in Figure 5, they can land on various regions around the decision boundary. This explains why certain tasks can disproportionately lead to bad results, as we see high variance in some tasks; hence, we measure the entire aggregate results.

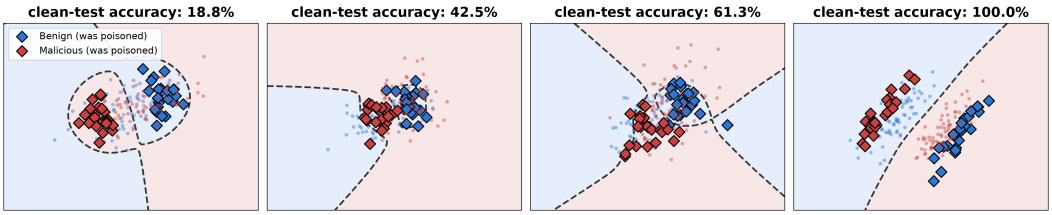


Figure 5: Illustration of the pocket-poisoning mechanism under four different random initializations, using an identical poisoning and unlearning recipe in each panel. Despite holding the recipe fixed, the specific points selected and the resulting boundary shape differ substantially by initialization. This illustrates that the perturbation stage of the pipeline can be erratic across initializations, contributing to the observed standard deviation. Some tasks' boundaries are also inherently more susceptible to this distortion than others, compounding the variance.

Here, C denotes the classifiers at tasks t and $t-1$. More specifically, C_{t-1} being the classifier at the end of the previous task, post-unlearning step, and C_t being the classifier at the current task, post adaptation (and poisoning) and pre-unlearning step. This set of successfully perturbed samples A_t represent data that has drifted from the original distribution, and is misclassified by later iterations of retrained classifiers, increasing the FPR and FNR. This dual-model requirement is deliberate. It rules out perturbations that merely exploit generic decision-boundary noise or produce inputs that any classifier would find ambiguous.

H Retention Test and Adaptability

The following graphs help us understand the retention of Task 0 families as the model continually updates and forgets. A common question may arise: that unlearning may exacerbate the problem of catastrophic forgetting, but we demonstrate that the knowledge retention of past tasks is consistent.

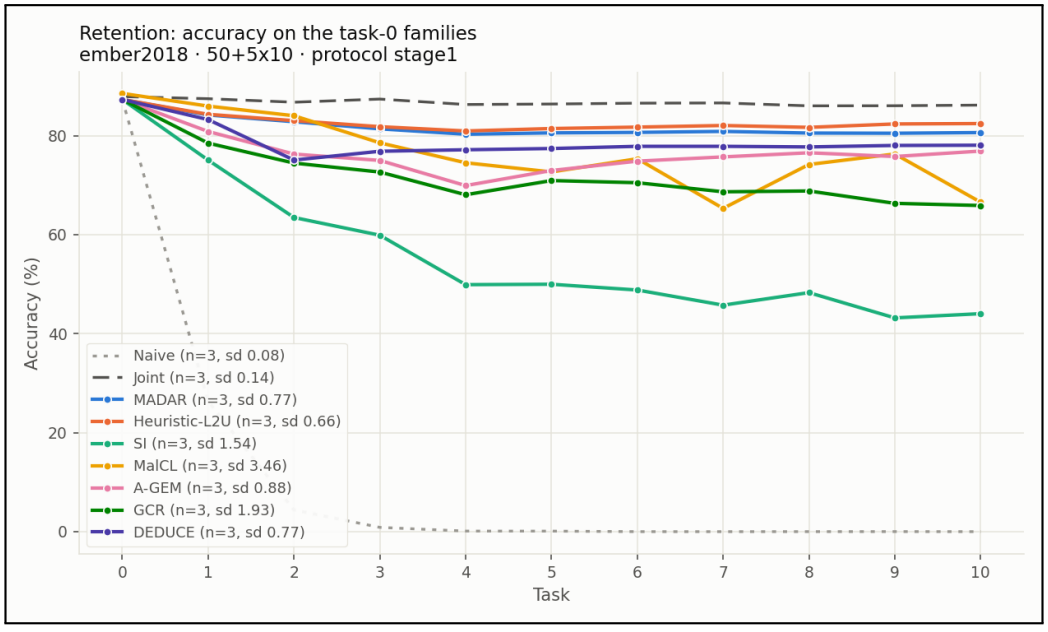


Figure 6: Mean accuracy across 3 seeds on task 0 at every task, demonstrates retention via showing how performance on task 0 deteriorates over time

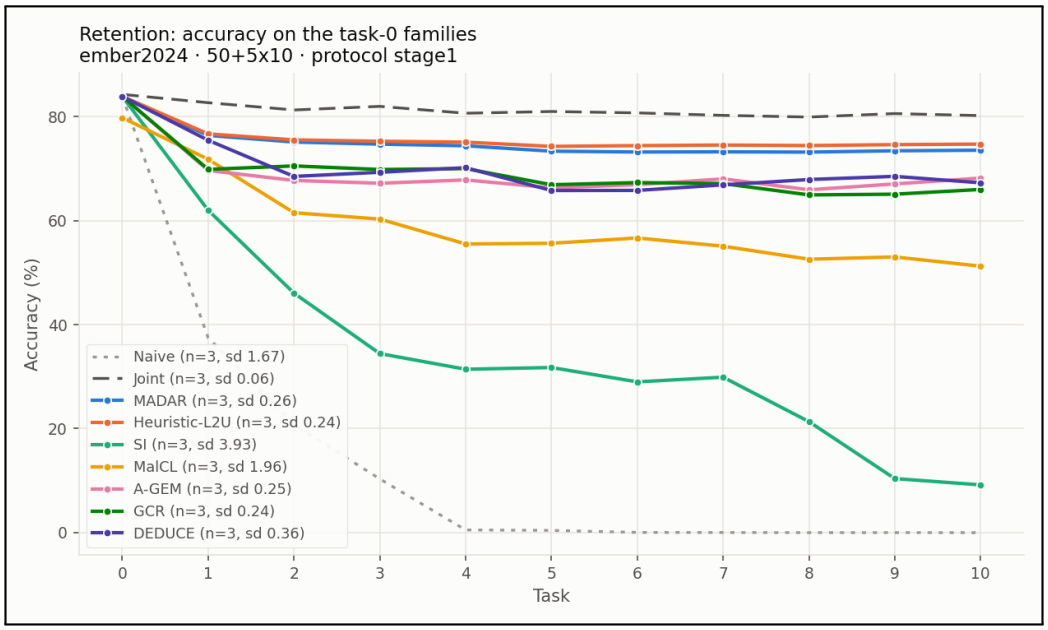


Figure 7: Mean accuracy across 3 seeds on task 0 at every task, demonstrates retention via showing how performance on task 0 deteriorates over time

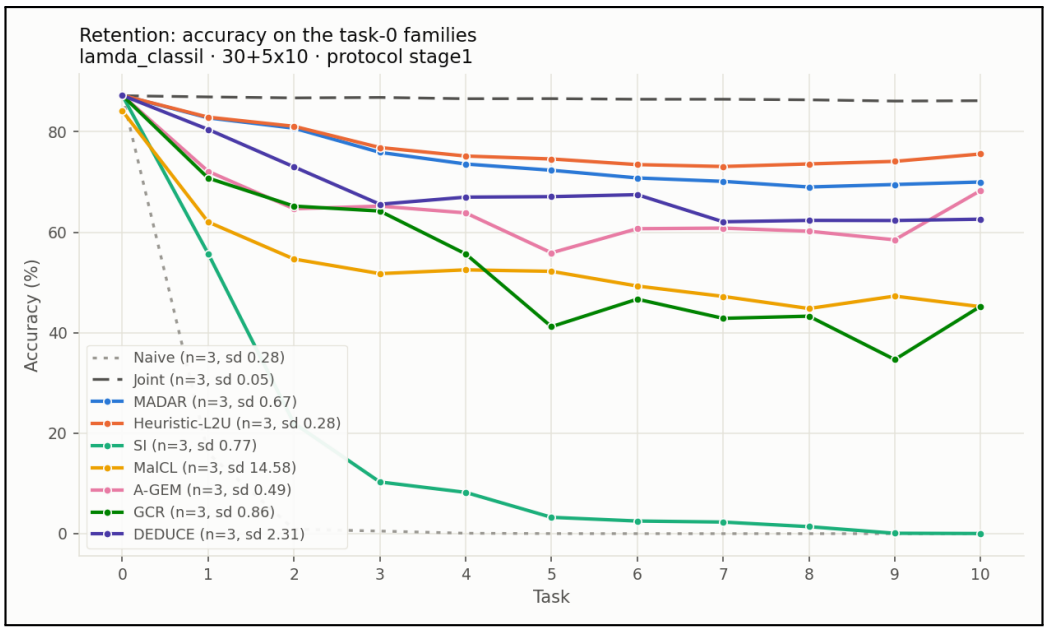


Figure 8: Mean accuracy across 3 seeds on task 0 at every task, demonstrates retention via showing how performance on task 0 deteriorates over time

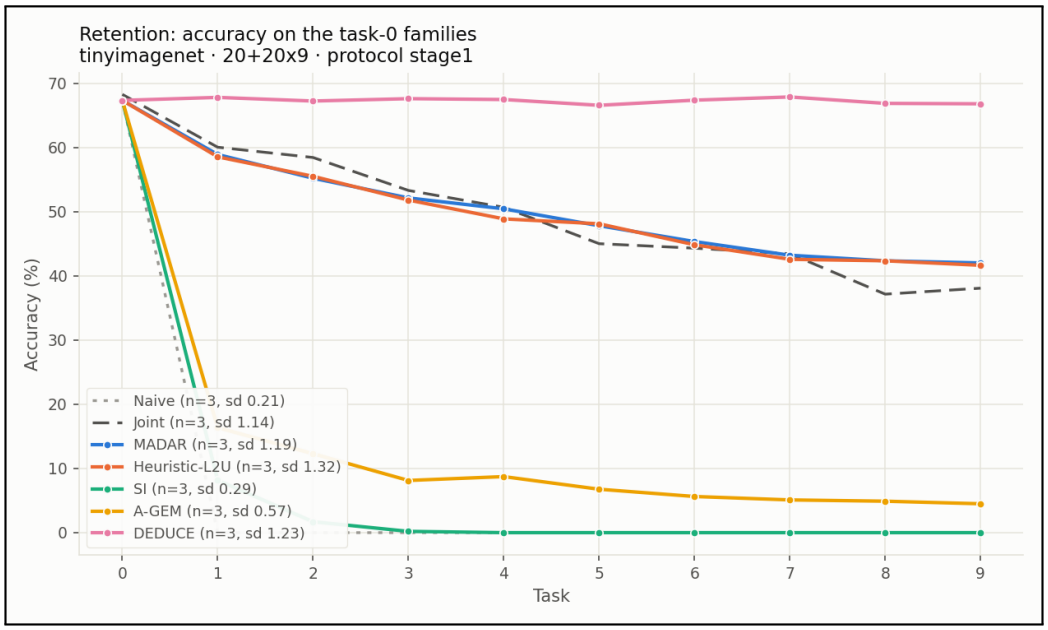


Figure 9: Mean accuracy across 3 seeds on task 0 at every task, demonstrates retention via showing how performance on task 0 deteriorates over time

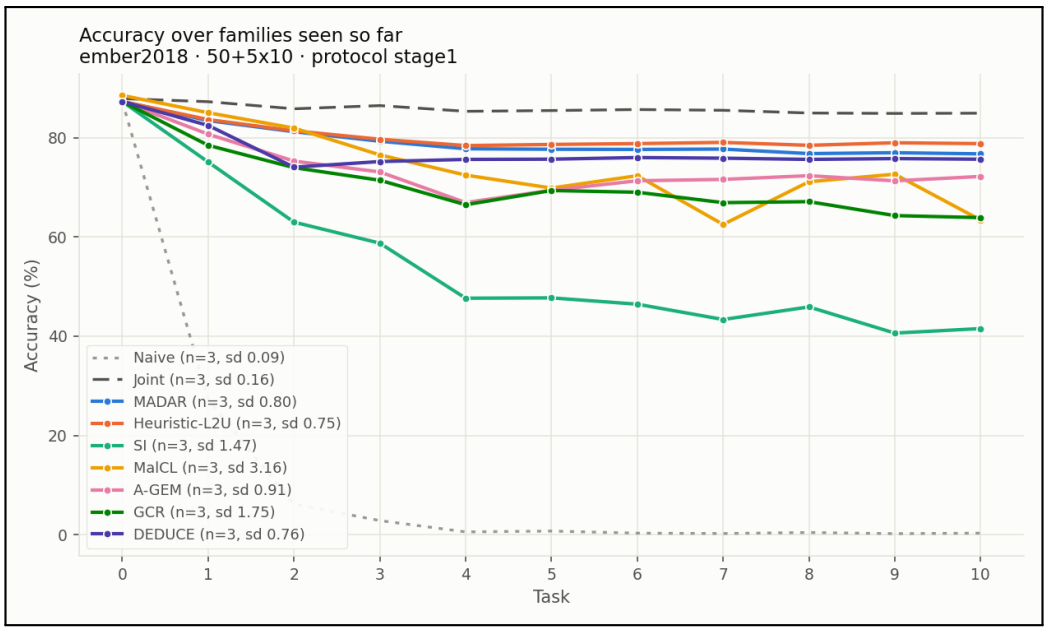


Figure 10: Mean accuracy at task t for all tasks seen so far, meaned across all tasks, meaned across 3 seeds

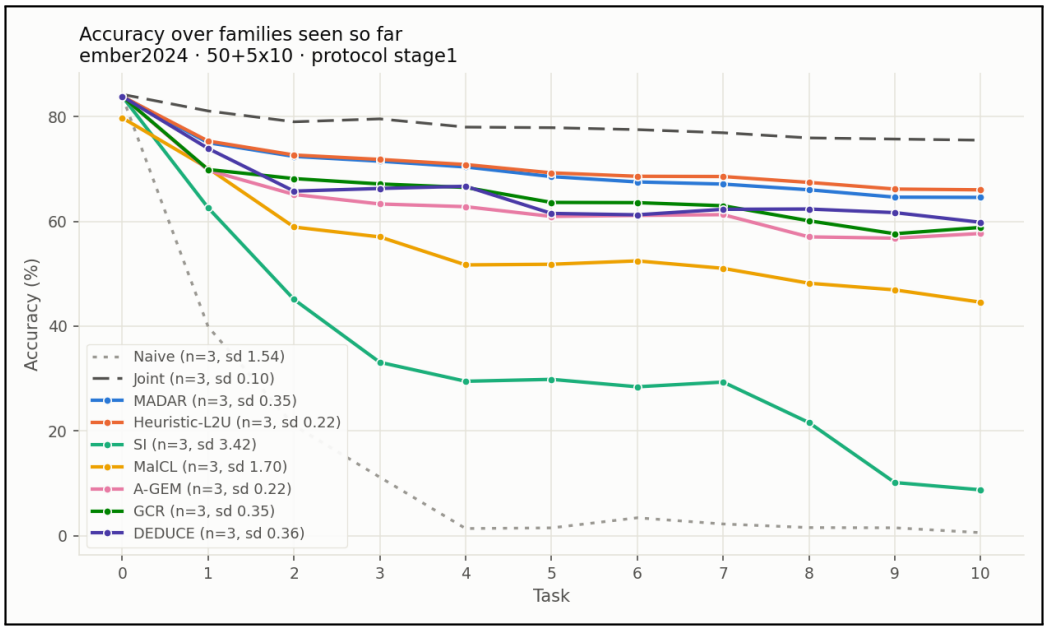


Figure 11: Mean accuracy at task t for all tasks seen so far, meaned across all tasks, meaned across 3 seeds

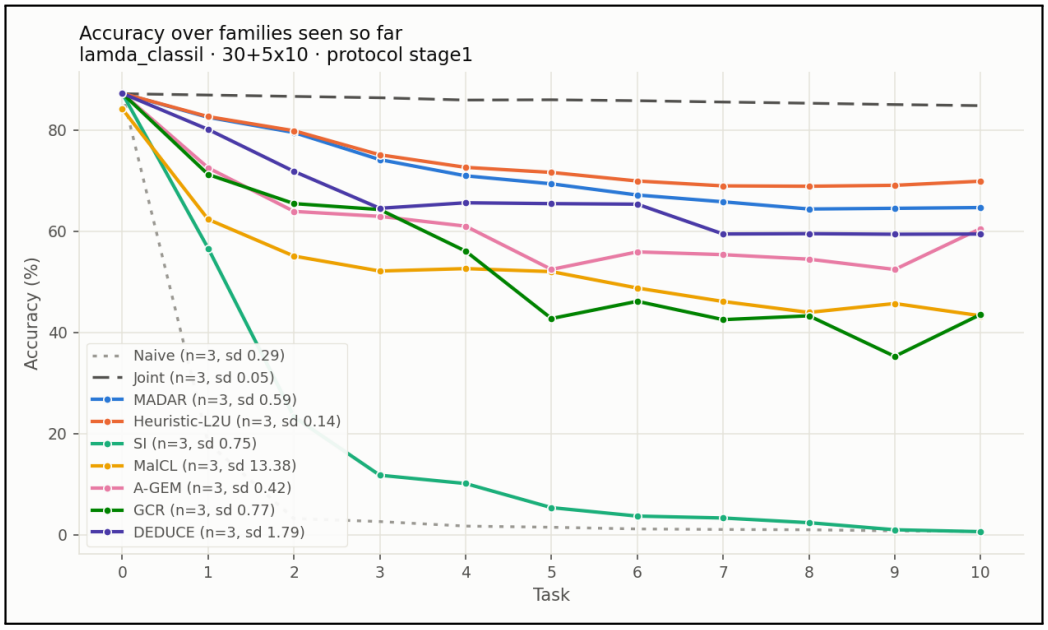


Figure 12: Mean accuracy at task t for all tasks seen so far, meaned across all tasks, meaned across 3 seeds

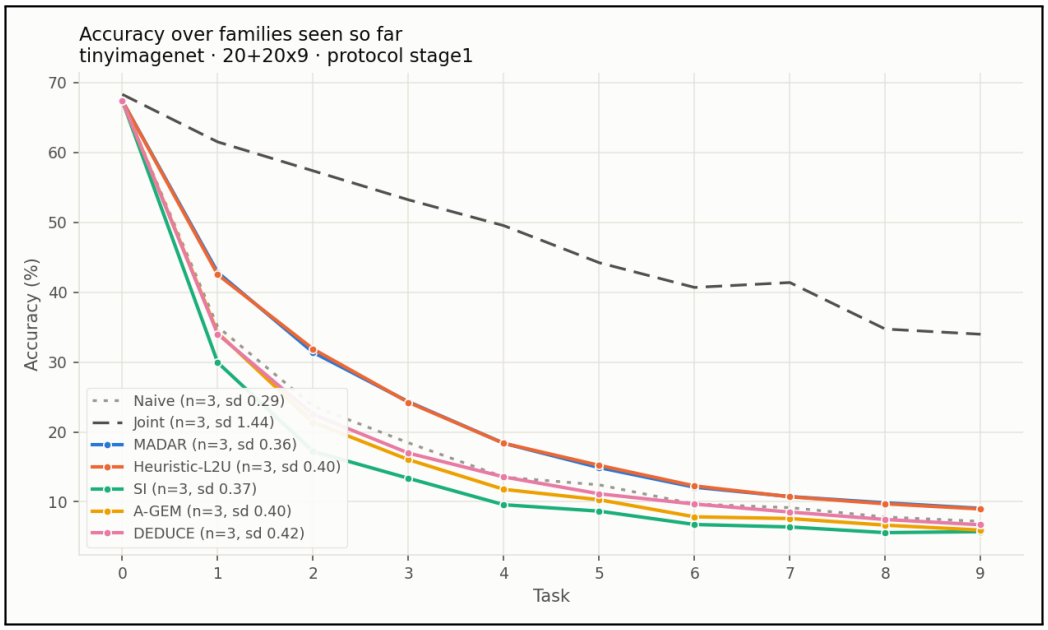


Figure 13: Mean accuracy at task t for all tasks seen so far, meaned across all tasks, meaned across 3 seeds

I REPRODUCIBILITY: PER SEED RESULTS

Table 15: Performance across three random seeds for the **Ember'18 dataset**. Mean Acc. denotes mean accuracy across all tasks; Acc. Final and Task-0 Final are the seen and task0 scopes after the final task. F1 denotes macro F1 over seen classes after the final task.

Method	Seed	Mean Acc.	Acc. Final	Task-0 Final	F1
Joint	1	86.04	84.97	86.13	64.19
	2	85.80	85.16	86.50	63.97
	3	85.75	84.76	85.93	65.54
Naive	1	11.41	0.30	0.00	0.33
	2	11.58	0.30	0.00	0.38
	3	11.45	0.29	0.00	0.34
Synaptic Intelligence	1	55.87	40.83	43.45	21.33
	2	52.95	40.17	42.54	22.79
	3	54.14	43.62	46.14	22.51
MalCL	1	76.71	69.29	73.03	36.31
	2	75.32	76.16	80.20	41.75
	3	70.68	44.81	46.50	34.38
A-GEM	1	74.53	72.57	77.29	37.41
	2	72.78	71.41	75.96	39.10
	3	74.06	72.65	77.44	38.42
DEDUCE	1	78.05	76.08	78.44	44.98
	2	76.55	75.37	77.72	47.38
	3	77.08	75.62	78.09	44.55
MADAR	1	80.04	77.93	81.60	39.81
	2	79.51	76.66	80.74	39.23
	3	78.47	75.70	79.57	37.50
HEURISTIC-L2U	1	80.56	78.55	82.25	39.98
	2	80.91	79.02	82.49	41.80
	3	79.47	78.91	82.62	39.72

Table 16: Performance across three random seeds for the **Ember'24 dataset**. Mean Acc. denotes mean accuracy across all tasks; Acc. Final and Task-0 Final are the seen and task0 scopes after the final task. F1 denotes macro F1 over seen classes after the final task.

Method	Seed	Mean Acc.	Acc. Final	Task-0 Final	F1
Joint	1	78.22	75.92	80.46	61.84
	2	78.30	75.22	79.94	61.03
	3	78.42	75.38	80.11	61.51
Naive	1	14.35	0.44	0.00	0.78
	2	17.05	0.81	0.00	1.09
	3	14.40	0.45	0.00	0.62
Synaptic Intelligence	1	31.89	3.69	3.06	2.73
	2	38.54	11.51	12.68	4.61
	3	33.79	11.03	11.83	4.68
MalCL	1	57.51	46.27	53.29	22.18
	2	54.13	41.17	47.67	15.12
	3	55.42	46.29	52.71	18.92
A-GEM	1	63.50	57.16	67.85	37.48
	2	63.88	58.52	68.68	38.69
	3	63.48	57.39	67.97	36.59
DEDUCE	1	65.65	58.82	67.00	41.62
	2	66.36	61.56	68.96	44.62
	3	65.85	59.11	65.80	38.49
MADAR	1	70.29	64.85	73.87	43.10
	2	70.40	64.38	73.57	41.90
	3	69.75	64.52	73.13	42.37
HEURISTIC-L2U	1	71.14	66.40	75.28	44.77
	2	71.05	65.69	74.48	44.58
	3	70.72	65.98	74.33	44.41

Table 17: Performance across three random seeds for the **LAMDA dataset**. Mean Acc. denotes mean accuracy across all tasks; Acc. Final and Task-0 Final are the seen and task0 scopes after the final task. F1 denotes macro F1 over seen classes after the final task.

Method	Seed	Mean Acc.	Acc. Final	Task-0 Final	F1
Joint	1	85.91	84.92	86.30	77.45
	2	85.93	84.78	86.12	77.53
	3	86.01	84.74	86.11	78.08
Naive	1	10.98	0.67	0.00	0.38
	2	10.96	0.64	0.00	0.15
	3	10.46	0.64	0.00	0.18
Synaptic Intelligence	1	18.90	0.57	0.00	0.31
	2	19.27	0.57	0.01	0.34
	3	17.82	0.67	0.07	0.53
MalCL	1	68.49	59.29	62.75	43.29
	2	48.01	37.95	39.22	24.89
	3	43.35	32.59	33.55	25.47
A-GEM	1	61.88	60.09	68.08	25.15
	2	61.22	60.75	68.59	25.44
	3	62.00	60.67	68.20	27.14
DEDUCE	1	69.13	59.76	63.63	40.13
	2	65.88	57.47	60.43	39.73
	3	66.22	61.08	63.72	43.47
MADAR	1	72.49	65.34	70.52	41.47
	2	71.69	64.08	69.83	38.56
	3	71.33	64.58	69.63	39.85
HEURISTIC-L2U	1	74.09	70.31	75.84	44.96
	2	74.06	67.83	73.32	41.84
	3	74.33	71.51	77.57	42.43

Table 18: Performance across three random seeds for the **Tiny ImageNet dataset**. Mean Acc. denotes mean accuracy across all tasks; Acc. Final and Task-0 Final are the seen and task0 scopes after the final task. F1 denotes macro F1 over seen classes after the final task.

Method	Seed	Mean Acc.	Acc. Final	Task-0 Final
Joint	1	47.37	31.31	36.60
	2	50.12	35.78	39.50
	3	47.97	34.84	38.20
Naive	1	20.17	7.23	0.00
	2	20.39	7.12	0.00
	3	20.75	7.25	0.00
Synaptic Intelligence	1	16.64	5.19	0.00
	2	17.34	5.37	0.00
	3	17.20	6.62	0.00
A-GEM	1	18.56	5.82	3.70
	2	19.35	6.18	5.60
	3	18.83	5.85	4.20
DEDUCE	1	19.33	6.56	65.30
	2	19.91	6.78	67.60
	3	20.14	6.79	67.60
MADAR	1	23.71	8.88	40.50
	2	24.43	9.60	45.80
	3	24.13	8.70	39.80
HEURISTIC-L2U	1	23.76	8.61	40.00
	2	24.56	9.57	45.00
	3	24.05	8.74	40.00

Table 19: Performance across three random seeds for the **Perturbed CICIDS 2017 dataset for Table 2**. Mean Acc. denotes mean accuracy across all tasks. Task 10 and Task 1 denote the mean accuracy on the corresponding tasks.

Method	Seed	Mean Acc.	Task 10	Task 1
Joint	130	91.50%	92.10%	89.75%
	132	92.50%	75.90%	97.75%
	133	80.40%	18.60%	99.05%
Naive	130	65.00%	82.10%	76.40%
	131	70.00%	97.60%	81.20%
	132	69.40%	95.70%	79.45%
SI	130	68.60%	86.05%	81.75%
	131	69.20%	94.20%	79.75%
	132	63.30%	95.15%	65.40%
AGEM	130	83.20%	92.65%	85.25%
	131	71.00%	65.35%	74.75%
	132	84.20%	86.40%	82.80%
DEDUCE	130	70.80%	96.10%	77.90%
	131	69.00%	96.70%	73.50%
	132	74.70%	95.70%	71.10%
MADAR	130	54.10%	50.55%	59.00%
	131	63.10%	53.80%	73.00%
	132	55.60%	46.65%	67.45%
HEURISTIC-L2U: Dropped Rows	130	85.80%	90.05%	88.95%
	131	82.30%	41.45%	95.50%
	133	81.60%	86.05%	77.60%
HEURISTIC-L2U: Approximate Unlearning	130	87.30%	69.55%	95.90%
	131	87.40%	68.65%	93.65%
	132	79.40%	64.30%	92.35%
HEURISTIC-L2U: Flip Label	130	84.60%	90.40%	82.80%
	131	87.30%	62.90%	94.55%
	132	78.40%	66.90%	80.50%
META-L2U: Dropped Rows	150	88.16%	87.33%	87.43%
	200	86.67%	88.19%	85.32%
	134	88.43%	86.46%	90.79%
META-L2U: Approximate Unlearning	150	87.16%	87.08%	88.63%
	200	87.44%	87.69%	87.59%
	134	86.68%	88.33%	85.62%
META-L2U: Flip Label	150	87.88%	87.33%	87.43%
	200	86.67%	88.19%	85.32%
	134	88.43%	86.46%	90.79%

Table 20: Performance across three random seeds for the **Pertrubed CICIDS 2017 dataset replay buffer ablation in Table** 11. Mean Acc. denotes mean accuracy across all tasks. Task 10 and Task 1 denote the mean accuracy on the corresponding tasks.

Method	Seed	Mean Acc.	Task 10	Task 1
L2U w/o Unlearning (Clean Buffer)	130	71.70%	50.40%	79.30%
	131	66.00%	49.80%	78.75%
	132	65.60%	38.00%	83.80%
HEURISTIC-L2U: Flip Label (Clean Buffer)	130	84.60%	90.40%	82.80%
	131	87.30%	62.90%	94.55%
	132	78.40%	66.90%	80.50%
HEURISTIC-L2U: Dropped Rows (Clean Buffer)	130	85.80%	90.05%	88.95%
	131	82.30%	41.45%	95.50%
	133	81.60%	86.05%	77.60%
HEURISTIC-L2U: Approximate Unlearning(Clean Buffer)	130	87.30%	69.55%	95.90%
	131	87.40%	68.65%	93.65%
	132	79.40%	64.30%	92.35%
L2U w/o Unlearning (Dirty Buffer)	130	54.10%	50.55%	59.00%
	131	63.10%	53.80%	73.00%
	132	55.60%	46.65%	67.45%
HEURISTIC-L2U: Flip Label (Dirty Buffer)	130	70.40%	87.20%	80.30%
	131	81.20%	86.50%	85.55%
	132	78.20%	94.60%	73.15%
HEURISTIC-L2U: Dropped Rows (Dirty Buffer)	130	65.50%	66.55%	84.45%
	131	79.30%	65.65%	92.30%
	132	71.20%	37.25%	78.65%
HEURISTIC-L2U: Approximate Unlearning (Dirty Buffer)	130	70.10%	87.25%	81.85%
	131	69.50%	35.30%	84.90%
	132	77.10%	45.05%	87.80%
L2U w/o Unlearning (Unfilled Buffer)	130	69.20%	56.10%	69.00%
	131	57.20%	48.60%	68.65%
	132	67.40%	50.50%	83.45%
HEURISTIC-L2U: Flip Label (Unfilled Buffer)	130	78.60%	47.45%	87.85%
	131	84.30%	63.25%	93.60%
	132	89.40%	94.45%	90.05%
HEURISTIC-L2U: Dropped Rows (Unfilled Buffer)	130	84.30%	56.70%	94.50%
	131	86.90%	76.20%	91.15%
	132	79.90%	57.10%	92.00%
HEURISTIC-L2U: Approximate Unlearning (Unfilled Buffer)	130	80.80%	39.90%	95.95%
	131	86.40%	87.35%	89.20%
	132	89.30%	95.25%	88.75%

Table 21: Performance across three random seeds for the **Pertrubed CICIDS 2017 dataset scaled to a smaller NN from Table** 12. Mean Acc. denotes mean accuracy across all tasks. Task 10 and Task 1 denote the mean accuracy on the corresponding tasks.

Method	Seed	Mean Acc.	Task 10	Task 1
MADAR (1 Hidden Layer)	130	49.90%	48.35%	59.85%
	131	44.00%	43.80%	49.20%
	132	52.50%	55.20%	60.80%
Synaptic Intelligence (1 Hidden Layer)	130	64.20%	91.65%	65.10%
	131	68.50%	96.05%	71.10%
	132	72.10%	95.30%	79.00%
DEDUCE (1 Hidden Layer)	130	70.00%	93.45%	60.10%
	131	66.90%	92.40%	49.75%
	132	78.10%	93.15%	68.85%
HEURISTIC-L2U: Flip Label (1 Hidden Layer)	130	64.20%	55.50%	58.80%
	131	71.60%	80.55%	82.00%
	132	76.70%	96.00%	79.80%
HEURISTIC-L2U:Dropped Rows (1 Hidden Layer)	130	75.20%	76.40%	76.15%
	131	68.20%	46.65%	90.00%
	132	75.10%	94.20%	83.55%
HEURISTIC-L2U: Approximate Unlearning(1 Hidden Layer)	130	71.00%	91.80%	54.55%
	131	79.00%	91.65%	84.45%
	132	73.10%	95.60%	72.90%
MADAR (2 Hidden Layers)	130	65.20%	53.00%	67.35%
	131	56.20%	51.85%	59.65%
	132	47.70%	48.30%	46.50%
Synaptic Intelligence (2 Hidden Layers)	130	65.90%	95.60%	71.30%
	131	65.60%	96.65%	69.30%
	132	68.70%	96.30%	78.50%
DEDUCE (2 Hidden Layers)	130	65.30%	92.95%	54.10%
	131	76.20%	95.05%	67.85%
	132	61.40%	92.65%	55.80%
HEURISTIC-L2U: Flip Label (2 Hidden Layer)	130	62.50%	57.15%	61.85%
	131	88.00%	94.40%	84.20%
	132	78.00%	85.85%	84.45%
HEURISTIC-L2U: Dropped Rows (2 Hidden Layer)	130	75.30%	43.90%	92.40%
	131	83.20%	73.95%	92.50%
	132	81.80%	94.40%	85.10%
HEURISTIC-L2U: Approximate Unlearning(2 Hidden Layer)	130	81.50%	68.85%	87.95%
	131	80.30%	91.60%	74.70%
	132	80.10%	83.60%	89.50%
MADAR (3 Hidden Layers)	130	56.60%	48.05%	70.90%
	131	65.80%	52.35%	78.35%
	132	60.00%	36.80%	79.45%
Synaptic Intelligence (3 Hidden Layers)	130	72.40%	96.60%	83.10%
	131	74.30%	96.70%	82.75%
	132	68.10%	94.70%	70.90%
DEDUCE (3 Hidden Layer)	130	65.80%	96.10%	66.70%
	131	61.30%	91.60%	41.75%
	132	59.10%	94.70%	56.65%
HEURISTIC-L2U: Flip Label (3 Hidden Layer)	130	80.70%	81.70%	75.85%
	131	83.00%	77.50%	87.30%
	132	71.70%	88.85%	68.15%
HEURISTIC-L2U:Dropped Rows (3 Hidden Layer)	130	84.90%	68.55%	94.65%
	131	87.80%	79.90%	91.80%
	132	79.50%	91.10%	91.30%
HEURISTIC-L2U: Approximate Unlearning(1 Hidden Layer)	130	84.70%	58.15%	94.70%
	131	87.20%	71.15%	90.80%
	132	75.20%	78.30%	92.85%

Table 22: Performance across three random seeds for the **Pertrubed Tiny Imagenet dataset from Table** 13. Mean Acc. denotes mean accuracy across all tasks. Task 10 and Task 1 denote the mean accuracy on the corresponding tasks.

Method	Seed	Mean Acc.	Task 10	Task 1
MADAR	130	8.80%	16.55%	5.70%
	131	9.10%	18.05%	6.45%
	132	9.10%	17.20%	5.90%
Synaptic Intelligence	130	5.70%	56.70%	0.00%
	131	6.10%	61.30%	0.00%
	132	6.10%	60.85%	0.00%
AGEM	130	6.20%	61.55%	0.00%
	131	6.30%	62.45%	0.00%
	132	6.30%	62.80%	0.00%
DEDUCE	130	6.00%	60.20%	0.00%
	131	6.30%	62.80%	0.00%
	132	6.20%	62.35%	0.00%
HEURISTIC-L2U: Flip Label	130	16.70%	32.15%	15.60%
	131	15.80%	29.60%	10.40%
	132	16.90%	31.55%	12.90%
HEURISTIC-L2U:Dropped Rows	130	17.70%	37.05%	13.90%
	131	17.00%	31.40%	12.50%
	132	17.90%	33.95%	12.70%
HEURISTIC-L2U: Approximate Unlearning	130	17.00%	36.25%	12.00%
	131	17.30%	31.65%	11.70%
	132	18.30%	37.30%	14.40%