

USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

RETROFIT: Continual Learning with Controlled Forgetting for Binarym Security Detection and Analysis

Yiling He, *University College London;* Junchi Lei, Hongyu She, and Shuo Shao, *Zhejiang University;* Xinran Zheng, *University College London;* Yiping Liu, *Alibaba Group;* Zhan Qin, *Zhejiang University;* Lorenzo Cavallaro, *University College London*

https://www.usenix.org/conference/usenixsecurity26/presentation/he-yiling

This paper is included in the Proceedings of the 35th USENIX Security Symposium.

August 12–14, 2026 • Baltimore, MD, USA

ISBN 978-1-939133-58-8

Open access to the Proceedings of the 35th USENIX Security Symposium is sponsored by



جامعة الملك عبدالله للعلوم والتقنية

King Abdullah University of Science and Technology

ARTIFACT EVALUATED
USENIX
AVAILABLE

RETROFIT: Continual Learning with Controlled Forgetting for Binary Security Detection and Analysis

Yiling He[1,⋆]✉, Junchi Lei[2,⋆], Hongyu She[2],
Shuo Shao[2], Xinran Zheng[1], Yiping Liu[3], Zhan Qin[2], Lorenzo Cavallaro[1]
[1] *University College London, London, United Kingdom*
[2] *Zhejiang University, Hangzhou, China*
[3] *Alibaba Group, Beijing, China*

Abstract

Binary security has increasingly relied on deep learning to reason about malware behavior and program semantics. However, the performance often degrades as threat landscapes evolve and code representations shift. While continual learning (CL) offers a natural solution through sequential updates, most existing approaches rely on data replay or unconstrained updates, limiting their applicability and effectiveness in data-sensitive security environments. We propose RETROFIT, which regulates knowledge retention and adaptation with controlled forgetting at each update, without requiring historical data. Our key idea is to consolidate previously trained and newly fine-tuned models, serving as teachers of legacy and emergent knowledge through retrospective-free parameter merging. Forgetting control is achieved by 1) constraining parameter changes to low-rank and sparse subspaces for approximate orthogonality, and 2) employing a confidence-guided arbitration to dynamically aggregate knowledge from both teachers.

Our evaluation on two representative applications demonstrates that RETROFIT consistently mitigates forgetting while maintaining adaptability. In malware detection under temporal drift, it substantially improves the retention score, from 20.2% to 38.6% over CL baselines, and exceeds the oracle upper bound on new data. In binary summarization across decompilation levels, where analyzing stripped binaries is especially challenging, RETROFIT achieves over 2× the BLEU score of transfer learning used in prior work and surpasses all baselines in cross-representation generalization.

1 Introduction

Deep learning models have become integral to modern binary security systems, driving advances in applications such as malware detection [38] and binary analysis [28, 47]. However, their effectiveness is constantly challenged by the non-stationarity from two dominant sources: *temporal drift*, as malware families and attack techniques evolve over time [60], and *representation shift*, as binaries undergo compilation, optimization, and symbol stripping [7]. Such variability can destabilize learned decision boundaries, resulting in critical alarm failures or even catastrophic system compromises [22].

Continual learning (CL) offers a natural paradigm for adapting models to this evolving data, aiming to accumulate knowledge across a sequence of domains rather than training a static model on a fixed dataset [51]. In security analytics, this paradigm is already well-recognized for addressing temporal drift in malware detection, where models are periodically updated through sequential retraining [10] or replay-based strategies [39]. We further identify that representation shift, where data differ across abstraction levels (e.g., increasingly stripped binaries), poses an analogous challenge. While prior work often employs transfer learning [62] for adaptation between single domains, we argue that CL's cumulative knowledge provides a distinct advantage. As illustrated in Figure 1b, even a basic CL approach enables stronger forward transfer to each subsequent task than independent single-task fine-tuning.

Nevertheless, applying CL in security-critical scenarios is highly constrained. Strict data-retention and sharing regulations for malware samples and proprietary binaries, together with progressively arriving large-scale data, make dominant strategies such as full retraining or data replay infeasible [27,40]. This setting fundamentally amplifies two coupled challenges to effective model adaptation. First, models struggle to preserve prior knowledge without old data, leading to *catastrophic forgetting* [18]. We find this issue to be surprisingly severe in malware detection: as validated in Section 5.2, adapting to new threats causes striking forgetting of older yet still-relevant malware, with the F1-score of all existing methods dropping below 0.8 within three years. Second, models fail to build a *cumulative understanding* to integrate new knowledge. As a result, they lack robust cross-representation generalization: experience gained from simpler data cannot be leveraged to enhance performance on more complex, data-scarce tasks like analyzing stripped binaries [2].

To address these challenges, we observe that both retention loss and generalization failure stem from the same root

⋆ Co-first authors. ✉ Corresponding author: heyilinge0@gmail.com

cause: uncontrolled model updates that overwrite previously learned information. We therefore propose RETROFIT, a data Retrospective-Free CL method that achieves effective Information Transfer through controlled forgetting.

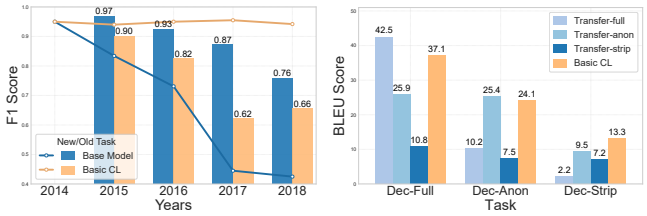
RETROFIT replaces inaccessible historical data with the previous model as a proxy for past knowledge, satisfying the data constraints inherent to security applications. Knowledge accumulation is then formulated as model consolidation, where a newly updated model learns new information and is subsequently integrated with the previous one via parameter-level merging. To realize controlled forgetting in this process, RETROFIT employs two complementary mechanisms. *Structurally*, it confines parameter updates to task-relevant subspaces via low-rank decomposition. *Functionally*, it optimizes a sparse attribution mask that adaptively balances knowledge contributions guided by model confidence.

We evaluate RETROFIT on two applications, each exemplifying a distinct form of non-stationarity. For temporal drift, we consider Android malware detection [25], a long-standing benchmark for CL where models must adapt to evolving threats without forgetting earlier ones. For representation shift, we study binary analysis [2], a more recent LLM-driven summarization task in which models are sequentially trained on decompiled Linux binaries of increasing obfuscation levels to assess cross-representation generalization. The two datasets comprise 259K labeled apps collected across five years and 215K decompiled function–documentation pairs spanning multiple optimization conditions, respectively.

Across both applications, we demonstrate the clear superiority of RETROFIT over five representative CL baselines [8, 13, 29, 33, 41]. In malware detection, RETROFIT markedly mitigates forgetting while maintaining adaptability—achieving an average retention score improvement of 33.3% and even surpassing the oracle upper bound (full retraining [56]) on new data. In binary summarization, RETROFIT consistently outperforms all baselines across abstraction levels and achieves more than twice the BLEU score of the prior transfer-learning approach on the critical stripped-binary task. We further show that alternative strategies that preserve individual models, such as ensemble learning and model merging [58], fail on the critical task of adapting to new malware and are significantly outperformed by CL methods in the summarization task.

Contributions. This paper makes three main contributions:

- We formalize CL under data-sensitive security constraints to address temporal and representational shifts. We identify two coupled challenges, i.e., retention loss and generalization failure, and trace both to uncontrolled model updates that fail to effectively integrate prior knowledge.
- We present RETROFIT to address both challenges through controlled forgetting without access to historical data. It enables data-free knowledge accumulation by consolidating previously trained and newly adapted models through parameter-level merging, where controlled forgetting is enforced through adaptive low-rank and sparse updates.
- We evaluate RETROFIT on two representative applications: malware detection under temporal drift and binary analysis under representation shift. RETROFIT consistently outperforms existing CL methods on previous domains while maintaining or surpassing state-of-the-art on newly encountered, security-critical ones.



(a) Malware Detection (b) Binary Analysis

Figure 1: Examples of the benefits and insufficiencies of CL in binary security applications. (a) Malware detection: continual fine-tuning mitigates temporal degradation compared to a static model trained on early data, but suffers from severe catastrophic forgetting; (b) Binary analysis: continual fine-tuning across abstraction levels outperforms transfer learning baselines in the most security-critical task, but cross-representation robustness remains challenging.

2 Background and Motivation

2.1 Continual Learning (CL)

In many real-world scenarios, data distributions evolve over time, requiring models to update continuously as new data arrive. Continual learning (CL) [11] seeks to adapt effectively to new distributions while maintaining performance on previously learned ones. The problems are typically categorized into three main scenarios based on how tasks are defined and evaluated [51]: task-incremental (task identity known at inference), class-incremental (no explicit task ID, expanding label space), and domain-incremental (shared label space, changing input distribution). Our work addresses the *domain-incremental* setting, which is highly relevant to real-world security applications, as it directly models the challenges of temporal drift and representation shift (defined in Section 2.2.1).

A common naive baseline for this setting is *continual fine-tuning (CFT)*, where the model is sequentially full fine-tuned on each new dataset. However, this approach typically suffers from *catastrophic forgetting* [17], where updates on new data overwrite representations learned from previous distributions. Many existing CL methods are thus designed to mitigate forgetting and are typically grouped into three main categories:

Replay-based. These methods explicitly preserve prior knowledge by storing data from previous tasks and inter-

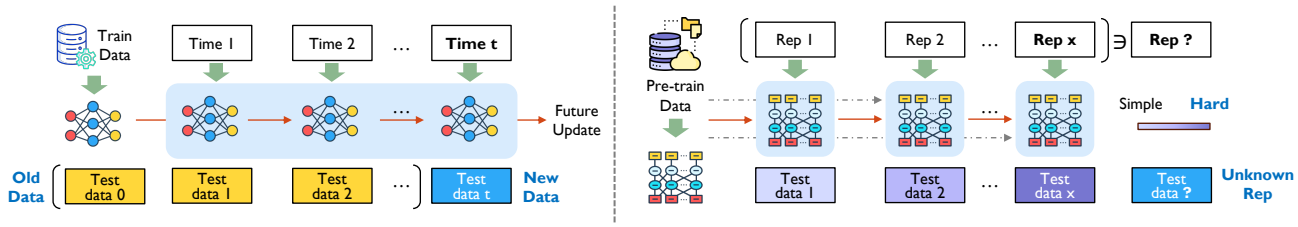


Figure 2: Continual learning for addressing temporal shift (left) and representation shift (right) in security applications.

leaving it with new samples during training. Representative approaches include Experience Replay (ER) [44] and iCaRL [42], both of which adopt selective sample replay. While ER relies on simple random sampling of past data, iCaRL designs a more structured strategy by retaining samples whose feature representations are closest to the class mean in the embedding space.

Regularization-based. This category constrains model updates by preserving dynamics of earlier knowledge. *Weight-constraint* methods often require preserving detailed information from the old model's training process. They adopt common heuristics including the Fisher information (EWC [26]), path-integral contributions (SI [64]), and output sensitivity (MAS [3]). In contrast, knowledge *distillation-based* methods preserve the functional knowledge of the old model. For example, the classical approach LwF [29] treats the old model's logits as a distillation target, while subsequent works improve by distilling feature-level representations (PODNet [13]) or adding contrastive learning (Co^{2}L [8]).

Architecture-based. These methods mitigate forgetting by modifying the network structure, such as by isolating parameters for different tasks (PackNet [32], HAT [46]) or dynamically expanding the network (DEN [63]). Since most approaches rely on task labels at inference time to know which mask or sub-network to use, they are restricted to the task-incremental setting. Some recent works, like ResAdapt [41], have adapted this paradigm to domain-incremental and introduces lightweight adapters.

While these paradigms show effectiveness in controlled benchmarks from computer vision and natural language processing [49,53], they often rely on idealized assumptions such as stable task boundaries or accessible replay buffers.

2.2 CL for Binary Security Applications

2.2.1 Why CL is needed

Deep learning-based binary security systems operate in dynamic environments, where the underlying data and representations continuously evolve [12, 37]. Figure 2 illustrates two fundamental types of this non-stationarity and the corresponding CL settings. **a) Temporal shift:** the underlying data distribution changes as new behaviors, threats, or software versions emerge. In this setting, the model is sequentially trained on datasets from time 1 to t, and evaluated on both previous test sets ($\text{test}_0, \ldots, \text{test}_{t-1}$) and the current one ($\text{test}_t$). This scenario highlights the need for *temporal robustness*, i.e., maintaining detection capability as data evolve. **b) Representation shift:** the similar semantics varies across abstraction levels, e.g., different stages of code transformation or recovery. Here, a pretrained model is continually trained across representations ($\text{rep}_1, \ldots, \text{rep}_x$), where each step introduces a more complex data form. Evaluation is performed on an incoming test set with an unknown representation, often dominated by the most degraded but security-critical form (e.g., rep_x).

To exemplify these scenarios, this paper examines two representative applications, as shown in Figure 1. The first, *malware detection*, represents the well-established motivation for continual learning [6, 25]: a model trained on early-year samples experiences sharp accuracy degradation on later data, whereas CFT maintains stable detection across years. The second, *binary summarization*, reveals a less-explored benefit that our findings highlight in recent LLM-based code understanding [2]. CFT across abstraction levels of decompiled binaries leads to improvements over individually tuned models on the rep_x, demonstrating effective knowledge accumulation. Together, these examples show how CL can provide a unified mechanism for sustaining performance across temporal and representational evolution in security-critical environments.

2.2.2 Security-Specific Constraints on CL

Binary security applications introduce unique constraints that make conventional CL paradigms difficult to apply directly. The constraints arise from two complementary perspectives:
Retrospective-free data constraints. Due to the sensitivity of malware samples and proprietary binaries, historical data is often subject to strict sharing, access, and reuse restrictions [43]. In practice, they may be maintained by different parties or under evolving access policies across training cycles, and earlier representations are frequently unavailable when one party encounters obfuscated or stripped inputs. Moreover, large-scale security systems (e.g., cloud-based malware detection) continuously ingest massive data streams [15], rendering long-term dataset storage and repeated retraining impractical. These factors necessitate retrospective-free continual adaptation.

Cross-domain performance requirements. Binary security models must remain effective as task boundaries are ambigu-

ous at test time [12]. In malware detection, the notion of 'old' and 'new' data naturally coexist: novel malware families continually emerge with unseen behaviors, while legacy variants persist in the wild. In binary analysis, decompiled programs may originate from different abstraction levels (e.g., anonymized or stripped) without metadata indicating their form. Such conditions demand cross-domain robustness: the capacity to preserve prior knowledge while generalizing to new, security-critical data distributions.

2.2.3 Research Gaps and Challenges

The constraints above shape the applicability and effectiveness of CL in security. Due to limited data accessibility, many conventional CL approaches are impractical. Replay-based strategies directly conflict with data confidentiality and retention policies, as they require storing or sampling from historical datasets. Similarly, regularization-based methods, particularly those relying on weight constraints, often assume the availability of shared validation data or explicitly defined task boundaries to balance stability and plasticity. For example, EWC [26] and MAS [3] still depend on previous task data to compute importance metrics, while SI [64] requires access to historical training trajectories—an even less feasible assumption in real-world security workflows.

Even within the feasible space, achieving effective cross-domain adaptation remains challenging. Without access to previous or nonexchangeable data, models easily overwrite learned representations, leading to *catastrophic forgetting*. This problem is especially evident in time-evolving malware detection, where CFT on newer yearly data, as used in prior work [10], rapidly degrades the detection of earlier malware. Our evaluation shows that, while such models maintain strong performance on recent threats, their F1-score on older samples drops below 0.7 after three years of adaptation. At the same time, transferring knowledge to harder domains is difficult. In binary analysis, continual adaptation from decompiled normal binaries to stripped ones still struggles to recover high-level semantics lost through symbol removal. These observations reveal two key gaps: the need to preserve prior knowledge without data replay and to accumulate domain-invariant representations that generalize across evolving inputs.

3 Our RETROFIT Method

3.1 Insights and Overview

Design Insights. As illustrated in Figure 3, our method is motivated by two complementary principles that together enable retrospective-free continual learning. First, in the absence of old data, the previous model serves as a proxy for past knowledge, and merging its parameters with new updates provides a natural path to knowledge accumulation. Second, this merging must be controlled to limit interference, ensuring that

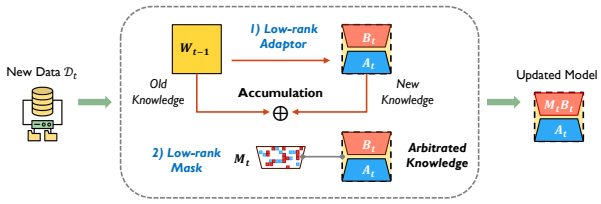


Figure 3: Design insights for accumulating old and new knowledge at each continual learning stage. Knowledge accumulation is achieved through parameter merging between the previous and newly adapted models, while interference control is ensured by applying masked low-rank updates.

new knowledge is integrated without degrading previously acquired functionality.

Accumulating Knowledge via Merging. To preserve past knowledge without data replay, we draw inspiration from model merging [54], where the parameters of multiple simultaneously trained models are combined to achieve strong multi-task performance. We adapt this idea to continual learning by incrementally merging each new update with the existing model, treating the previous model as a proxy for past data. This consolidation enables knowledge from evolving distributions to be integrated sequentially without access to earlier samples. Let $W \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ denote a layer weight of the base model. Formally, after T rounds of adaptation, the accumulated weight is $W_T = W_0 + \sum_{t=1}^{T} \Delta W_t$.

From Full Parameters to Localized Updates. To make merging more controlled, we adapt the LoRA [21] perspective by restricting each update to a low-rank decomposition of the weight matrix. The original form of a LoRA update is expressed as $\Delta W = AB, A \in \mathbb{R}^{d_{\text{in}} \times r}, B \in \mathbb{R}^{r \times d_{\text{out}}}, r \ll d_{\text{in}}$, where A is the down-projection and B is the up-projection matrix. We adopt this setup for continual learning: new knowledge is localized into a compact subspace rather than distributed across the full parameter space, thereby reducing the chance of interference. To further refine this localization, the update is constrained on B and a real-valued mask M is introduced to selectively filter its components. Thus, for round t, the resulting update is $\Delta W_t = A_t(M_t \odot B_t)$.

Method Overview. Algorithm 1 summarizes the overall procedure. Building on the insights of localized low-rank updates and cumulative merging, each task-specific adaptation is restricted to the projection matrix B_t (Lines 3–7), and the resulting masked update is incorporated into the running model (Line 21). Arbitration (Lines 8–20) then complements this foundation by regulating how new knowledge is integrated during merging: the learned masks limit parameter overlap across tasks, while confidence-guided transfer ensures that reliable past predictions are preserved.

Controlled Forgetting. The combination of structural control and arbitration introduces a soft bound on update-induced forgetting. Formally, let the effective update energy at round

Algorithm 1: Continual Learning with Low-Rank Updates and Adaptive Keep & Gain Arbitration

Input: Base model θ_{base} with weights in $\mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$, dataset stream $\{\mathcal{D}_t^{\text{train}}, \mathcal{D}_t^{\text{valid}}\}_{t=1}^{N}$, where $\mathcal{D}_t = \{(x_i, y_i)\}$ with $x_i \in \mathbb{R}^{d_{\text{in}}}$ and $y_i \in \mathcal{Y}$

Output: Merged model f_{merge} after N tasks

1 **Init:** $\theta_{\text{prev}} \leftarrow \theta_{\text{base}}$.

2 **for** $t = 1$ **to** N **do**

/* (1) Low-rank adaptation on current task */

3 Draw $A_t \in \mathbb{R}^{d_{\text{in}} \times r}$ with i.i.d. entries of zero mean and finite variance $\sigma^2 > 0$.

4 Init $B_t \in \mathbb{R}^{r \times d_{\text{out}}} \leftarrow 0$.

5 **Update** B_t by minimizing $\text{ModelLoss}(\mathcal{D}_t^{\text{train}})$ validated on $\mathcal{D}_t^{\text{valid}}$. ▷ freeze θ_{prev} and A_t

6 Define teacher models: $f_{\text{old}} \leftarrow \theta_{\text{prev}}$, $f_{\text{new}} \leftarrow \theta_{\text{prev}} + \text{scaled}\ (A_t B_t)$.

/* (2) Learn mask for knowledge merging */

7 Init $M_t \in \mathbb{R}^{r \times d_{\text{out}}} \propto B_t$. ▷ mask on B_t

8 **repeat**

9 Sample $(x, y) \sim \mathcal{D}_t^{\text{train}} \cup \mathcal{D}_t^{\text{valid}}$.

// Model predictions

10 $p_o = f_{\text{old}}(x)$, $p_t = f_{\text{new}}(x)$;

11 $p_s = (\theta_{\text{prev}} + \text{scaled}\ (A_t(M_t \odot B_t)))(x)$.

// Calibrated true-class confidence

12 $\mathcal{C}_o = (c_o, \tau_o) \leftarrow \text{Conf}(p_o, y; f_{\text{old}})$;

13 $\mathcal{C}_t = (c_t, \tau_t) \leftarrow \text{Conf}(p_t, y; f_{\text{new}})$.

// Keep and Gain arbitration

14 $T^\star \leftarrow \text{Teach}(p_o, p_t; \mathcal{C}_o, \mathcal{C}_t)$,

15 $w^\star \leftarrow \text{SampleWeight}(\mathcal{C}_o, \mathcal{C}_t)$;

16 $\mathcal{L}_{\text{merge}} \leftarrow \mathbb{E}\left[w^\star \cdot \text{Arbitrate}(p_s \,\|\, T^\star)\right]$.

// Supervision & mask sparsity

17 $\mathcal{L}_{\text{sup}} = \text{ModelLoss}(p_s, y)$;

18 $\mathcal{R}(M_t) = \text{SparseGroupLasso}(M_t)$.

19 **Update** M_t by minimizing total loss $\mathcal{L} \leftarrow \mathcal{L}_{\text{merge}} + \eta \mathcal{L}_{\text{sup}} + \mu \mathcal{R}(M_t)$.

20 **until** *convergence*

/* (3) Merge into the running model */

21 $\theta_{\text{prev}} \leftarrow \theta_{\text{prev}} + \text{scaled}\ (A_t(M_t \odot B_t))$. ▷ current merged

22 **Return** $f_{\text{merge}} \leftarrow \theta_{\text{prev}}$.

t be $E_t := \|M_t \odot B_t\|_F$, which measures the Frobenius norm of the masked low-rank expansion actually merged. If A is an ε-approximate isometry, then the average prediction drift introduced by the t-th merge satisfies

$$\mathbb{E}_{\mathcal{D}_{\le t-1}}\left[\|p_t - p_{t-1}\|_2\right] \le O\left(\sqrt{\frac{r}{d_{\text{in}}}} \sum_{s=1}^{t} E_s + \rho \sum_{s=1}^{t} E_s + \sqrt{\delta} + \varepsilon\right), \tag{1}$$

where $\mathcal{D}_{\le t-1}$ denotes the union of past distributions and p is the predictive probability distribution of the model.

In the following, we explain different components of this forgetting bound. The first term, which scales with $\sqrt{r/d_{\text{in}}}$, reflects structural interference control through low-rank adaptation and is analyzed in Section 3.2. The latter two terms, involving mask overlap ρ and arbitration tightness δ, arise from the arbitration stage and are examined in Section 3.3.

3.2 Low-rank Model Adaptation

This section analyzes how the structural design of our update mechanism mitigates task conflict during continual learning. To make this concrete, we first describe the update–merge formulation, showing how only B_t is trained and integrated into the model while the random projection A remains fixed. We then establish two theoretical properties (proofs in Appendix A) to explain the reduced knowledge interference.

Update–Merge on B_t. In our design, all task-specific knowledge is concentrated in the low-rank coefficients B_t, while both the base model θ_{prev} and the projection matrix A remain fixed. For each round t, the expansion matrix B_t is optimized to minimize the model loss on new data (Line 5). The mask M_t is then initialized from the drift captured in B_t (Line 7) and refined subsequently during arbitration to regulate how the update is merged. Let θ denote the parameters of the base model, and consider a representative weight matrix $W \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ within θ, such as a projection in an MLP or Transformer block. The merged weight after T updates is

$$W_T = W_0 + \sum_{t=1}^{T} A_t(M_t \odot B_t). \tag{2}$$

Specifically, the low-rank update for task t follows our mask-based LoRA form, where only B_t (for adaptation) and M_t (for arbitration) are trainable. This setup ensures that interference across tasks arises solely from overlaps among the matrices $\{B_t\}$, and thus inter-task variability is entirely confined to the $r \times d_{\text{out}}$ coefficient space.

Effect of Low Rank. By projecting the update onto an r-dimensional manifold, the 'surface area' for potential interference with previous knowledge is reduced from d_{in} to r.

Proposition 1 (Low-rank reduces expected interference)**.** *Let $g_{old} = \nabla_\theta \mathcal{L}_{old}(\theta_{prev})$ denote the gradient of a past task at the current model. For any old-task gradient g_{old} and update $\Delta W_t = A_t B_t$,*

$$\mathbb{E}\left[\frac{\langle g_{old}, \Delta W_t \rangle}{\|g_{old}\|_F \|\Delta W_t\|_F}\right] \le O\left(\sqrt{\frac{r}{d_{in}}}\right). \tag{3}$$

Thus, the expected cosine similarity between new updates and old-task gradients scales with $\sqrt{r/d_{\text{in}}}$, making destructive interference less likely as long as $r \ll d_{\text{in}}$.

Effect of Freezing A. The matrix A is randomly initialized with zero mean and positive variance (Line 3) and remains frozen throughout training (Line 5). By fixing A, we leverage a high-dimensional phenomenon where random directions are naturally perpendicular to one another, resulting in approximate orthogonality in high-dimensional regimes.

Proposition 2 (Approximate orthogonality with frozen random A)**.** *Let $A_s, A_t \in \mathbb{R}^{d_{in} \times r}$ be independent random matrices*

for distinct tasks $s \neq t$, whose entries are i.i.d. with zero mean and positive variance. Assume the entry distribution is sub-Gaussian. Let $\widetilde{B}_s := B_s \odot M_s$ and $\widetilde{B}_t := B_t \odot M_t$, and define $\Delta_s := A_s\widetilde{B}_s$ and $\Delta_t := A_t\widetilde{B}_t$. If $r = o(\sqrt{d_{in}})$, then as $d_{in} \to \infty$,

$$\frac{\langle \Delta_s, \Delta_t \rangle_F}{\|\Delta_s\|_F \, \|\Delta_t\|_F} \xrightarrow{\text{p}} 0. \tag{4}$$

Hence, given task-specific updates Δ_s and Δ_t, freezing the random projections A ensures that task directions are asymptotically orthogonal. As the cross-task interaction (cosine similarity) becomes negligible in high-dimensional regimes, multiple updates can be merged into a single model without significant interference.

Remarks. By restricting updates and merges exclusively to B_t, the model localizes all task-specific knowledge to a controlled subspace. The interference is bounded by $\sqrt{r/d_{\text{in}}}$, while the frozen random projection A induces approximately orthogonal updates across tasks.

Note that our analysis is conducted at the *parameter level*. Although the full model may be nonlinear, adaptation is still realized through structured perturbations to trainable modules, e.g., linear projections and feed-forward matrices in Transformers. Under standard local linearization assumptions, their effect is governed by their geometry in parameter space, making our analysis meaningful beyond purely linear models (see Appendix A.3). The result should therefore be understood as a *local controlled-forgetting guarantee*, not a strict global guarantee in highly non-convex function space.

3.3 Confidence-driven Knowledge Arbitration

We next analyze how the arbitration stage complements low-rank adaptation by regulating both functional preservation and parameter overlap. While structural design restricts interference to a subspace, naive merging can still introduce conflicts when updates overlap or when old-task predictions are altered. To address this, arbitration introduces two key mechanisms that correspond to the residual terms in Equation 1: 1) a confidence-driven knowledge distillation that preserves predictions in high-confidence regions of teachers to control functional tightness δ, and 2) sparse mask learning with group regularization, which reduces overlap between updates and controls the mask overlap ratio ρ.

Confidence-guided Arbitration. For a training sample (x, y) from the current task, let $p(\cdot) = f(x)$ denote some model output distribution. The student prediction p_s is trained to reconcile the outputs of two teachers (Lines 10–11): the previous model $p_o = f_{\text{old}}(x)$ and the adapted model $p_t = f_{\text{new}}(x)$.

As shown in Figure 4, new adaptation may derive a decision boundary that overfits to current-domain data, causing legacy knowledge encoded by the old model to be forgotten, as highlighted legacy samples become misclassified in the left panel. Since only data from the current domain are accessible, the old model serves as the sole proxy for past

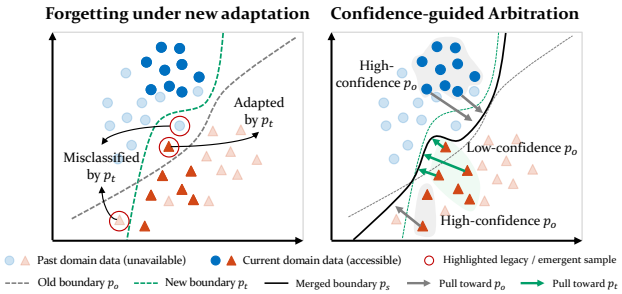


Figure 4: Intuitive examples of how confidence-guided arbitration pulls the decision boundary to preserve legacy knowledge while adapting to emergent knowledge.

knowledge. Arbitration therefore *prioritizes the old model* when its prediction is reliable to maintain stability, as shown in the right panel. When the old model is confident on the true class, the sample likely lies within regions where p_o has a large decision margin and encodes reliable legacy knowledge. Enforcing consistency with p_o in these cases prevents overwriting reliable representations, thereby correctly classifying the highlighted legacy samples. Only when the old model is less certain does arbitration shift toward the new model p_t to strengthen knowledge acquisition, which keeps the highlighted emergent sample correctly classified.

To this end, we implement arbitration with four components: calibrated confidence estimation, teacher selection, sample reweighting, and distillation to the student. The true-class confidence c is computed as the model's softmax probability for the ground-truth class y. To improve cross-class comparability, an optional conformal percentile threshold $\tau \in [0, 1]$ can also be computed based on validation data confidences (Lines 12–13). We follow standard practices from prior work [6,50] for this optional calibration, and a prediction is then regarded as confident if $c > \tau$. The resulting confidence pairs $\mathcal{C}_o = (c_o, \tau_o)$ and $\mathcal{C}_t = (c_t, \tau_t)$ are then used to determine which teacher to follow for each sample.

Formally, the arbitration loss guided by confidence to enforce consistency with different models (Lines 14–15) is

$$\mathcal{L}_{\text{merge}} = \mathbb{E}[w^\star \cdot \textsc{Arbitrate}(p_s \,\|\, T^\star)], \tag{5}$$

where $T^\star$ denotes the arbitration target derived from teacher predictions $\{p_o, p_t\}$, $w^\star$ is a sample-wise weight proportional to teacher reliability, and $\textsc{Arbitrate}(\cdot)$ measures the divergence (e.g., KL or MSE) between the student prediction and the selected or blended teacher outputs.

Arbitration Modes. We instantiate $T^\star$ and $w^\star$ under two confidence-based modes.

1) *Hard arbitration.* A discrete confidence rule selectively activates each teacher based on its own reliability:

$$T^\star = p_o \mathbf{1}[c_o \geq \tau_o] + p_t \mathbf{1}[c_o < \tau_o, c_t \geq \tau_t], \quad w^\star = 1. \tag{6}$$

Only confident and correct teachers contribute to supervision, while low-confidence predictions are excluded from the merge loss. This selective strategy enforces reliability at each

step: the old model preserves established knowledge when trustworthy, and the new model contributes only when it confidently improves on unseen or shifted concepts. This mode is typically favored when precise control over forgetting is required or when label noise is minimal.

2) *Soft arbitration.* A continuous gate produces a convex combination of teachers using a smooth old-priority gate:

$$g = \sigma(c_o - \tau_o) \in [0,1], \qquad T^\star = g\, p_o + (1-g)\, p_t. \quad (7)$$

The gate g increases monotonically with old-model confidence, yielding a smooth transition from the old to the new teacher as confidence decreases. An optional weighting term $w^\star$ can further modulate each sample's contribution based on confidence or disagreement, for example: $w^\star \propto \max(c_o, c_t)^\beta \big(1 + \lambda \|p_o - p_t\|_1\big)$, where β and λ control the emphasis on teacher reliability and disagreement, respectively. This formulation allows both teachers to contribute proportionally, avoiding abrupt supervision shifts, and is advantageous when representations evolve progressively, such as in feature-level or generative tasks.

The arbitration loss is integrated into the overall objective (Line 19), along with supervision and sparsity regularization, driving a sparse mask learning process for knowledge accumulation, which we introduce next.

Sparse Mask Learning. A mask M_t is learned over the low-rank adapter B_t to control parameter overlap across tasks. We impose a Sparse-Group-Lasso regularization that promotes both element-wise and structured sparsity:

$$\mathcal{R}(M_t) = \|M_t\|_1 + \lambda \sum_b \|M_{t,b}\|_2, \quad (8)$$

where λ balances fine-grained and group-level sparsity. The first term enforces element-wise pruning for compactness, while the second term encourages group-level disjointness, with each group b corresponding to a rank column in the LoRA adapter B_t. This structured sparsity enhances interpretability by identifying which low-rank components are activated and isolating task-specific subspaces. Finally, the mask is optimized jointly with arbitration through:

$$\mathcal{L} = \mathcal{L}_{\text{merge}} + \eta\, \mathcal{L}_{\text{sup}} + \mu\, \mathcal{R}(M_t), \quad (9)$$

where $\mathcal{L}_{\text{sup}}$ denotes the supervised loss (e.g., cross-entropy) on the current task, which anchors the optimization to ground-truth labels to avoid drifting toward purely self-consistent representations. The optimization proceeds until convergence, after which the resulting masked adapter $A_t(M_t \odot B_t)$ is merged into the running model θ_{prev} (Line 21).

Remarks. Confidence-driven knowledge arbitration clamps functional drift where the old model is reliable and constrains structural interference via the sparse mask, providing safeguards for stable and interpretable *knowledge retention*. Meanwhile, it guarantees plasticity on new data by releasing these constraints when the old model is uncertain, enabling effective adaptation to novel concepts. Formally, the expected predictive drift on the current distribution $\mathcal{D}_t$ satisfies

$$\mathbb{E}_{\mathcal{D}_t}[\|p_t - p_{t-1}\|_2] \geq C(1-\rho)E_t, \quad (10)$$

where $C > 0$ is a model-specific constant reflecting sensitivity to drift, complementing the earlier soft forgetting bound to illustrate control over *knowledge adaptation*.

4 RETROFIT Deployment

We demonstrate the deployment of RETROFIT under two representative CL scenarios in binary security.

Temporal Drift (Malware Detection Example). This scenario reflects the continual evolution of malicious behaviors over time, where each temporal segment introduces new families or variants unseen in earlier datasets. Older samples remain critical for preserving established family patterns and recurring code reuse, while newer samples are the most security-critical, often embedding novel evasion strategies. Therefore, RETROFIT ensures backward-compatible detection while maintaining adaptability to emerging threats. Because malware datasets are typically well-annotated, the *hard arbitration* mode is adopted, enforcing discrete teacher selection for precise control over forgetting. Formally, the arbitration in Equation 5 operates on probability distributions via a KL divergence: $\sum_i p_s(i) \log \frac{p_s(i)}{T^\star(i)}$, where p_s denotes the student's output logits and $T^\star$ is that from the selected teacher.

Representation Shift (Binary Analysis Example). In binary analysis, executables appear at different abstraction levels, from normally compiled to stripped versions. While high-level representations provide richer semantics, stripped binaries are the most security-critical, often being the only artifacts available during incident response. RETROFIT sustains robustness across such heterogeneous representations by transferring structural knowledge between them. Because the model is generative and the summary ground truth is often noisy (e.g., extracted from comments), the *soft arbitration* mode is adopted to enable confidence-weighted blending. It is performed in latent space of the model via a mean squared error: $\|p_s - T^\star\|_2^2$, where p_s denotes pooled latent representations of the student model, and $T^\star$ is the confidence-weighted latent interpolation between that of teacher models.

We use these deployment setups for evaluation and provide more implementation details of our method in Appendix B.1.

5 Evaluation

5.1 Experimental Setup

Dataset and Model. For the malware detection task, we adopt the dataset from Transcendent [6], which spans five years (from 2014 to 2018) and mitigates common sources of temporal and spatial bias. It contains 232,848 benign and 26,387 malicious Android applications collected from AndroZoo.[1] Following prior work, we extract the widely used

[1] AndroZoo: https://androzoo.uni.lu

Drebin features [4] and employ a multilayer perceptron (MLP) classifier as the backbone model [16]. For binary analysis, we leverage the CAPYBARA [2] dataset, which provides function–documentation pairs extracted from decompiled binaries. The decompiled code is organized into three subsets to represent progressively reduced levels of semantic information that emulate real-world obfuscation (*decompiled-*): *full*, *anonymized*, and *stripped*.[2] The dataset contains 215K aligned and documented decompiled functions, while only 14K documented samples are available for the stripped subset, underscoring the difficulty of recovering meaningful semantics from heavily stripped binaries. Following the dataset's accompanying BinT5 model design, we employ the same CodeT5 [52] backbone to ensure consistency. Within each temporal split and abstraction subset, data are partitioned with a consistent 8:2 train/test split throughout all CL experiments.

CL Baselines. We compare RETROFIT with five representative continual learning methods, encompassing the primary strategies compatible with our challenging replay-free and task-agnostic setting. 1) **CFT**: the naive sequential training baseline that fine-tunes the model on each new task without any mechanism to mitigate forgetting. 2) **LwF** [29]: a logit-distillation method that regularizes the new model's training using the previous model's predictions as soft targets. 3) **PODNet** [13]: a feature-distillation approach that constrains both the output and intermediate feature representations of the new model to remain consistent with those of the previous one. 4) **Co^{2}L** [8]: a contrastive learning method that designs an instance-relationship distillation loss that preserves pairwise similarity structures from the previous model. 5) **ResAdapt** [41]: an architecture-based method that retains the base weight and expands the network with residual adapters for new tasks. The distillation-based baselines (2∼4) may share similar concepts to our knowledge arbitration but differ fundamentally in mechanism: these methods regularize new model training, whereas RETROFIT performs model consolidation through mask-based merging.

For reference, two additional models are included as anchors. 1) *Base*: a static model trained only on the first-year data for malware detection or the stripped-binary subset for binary analysis. 2) *Oracle*: a joint training model using all available data at each period for malware detection or across all representation levels for binary analysis. Implementation details of these baselines are provided in Appendix B.2, with comparative results reported in Section 5.2 for malware detection and Section 5.3 for binary analysis.

Aggregation Baselines. As an alternative to continual learning, aggregation-based methods maintain multiple expert models and combine their knowledge at prediction or parameter level instead of performing sequential adaptation.

[2]Subset names in its original paper: *Decompiled*, *Demi-Stripped*, and *Stripped*. We adopt clearer names to improve interpretability. Specifically, the anonymized variant (originally Demi-Stripped) removes all identifiers from the decompiled code and replaces them with generic placeholders.

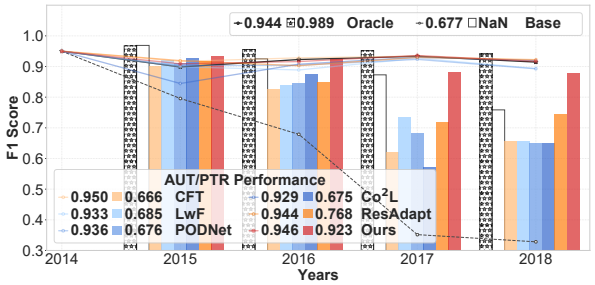


Figure 5: Cumulative CL comparison in malware detection.

Table 1: Year-wise CL comparison in malware detection: per-step F1-scores on old and new test data (bar and line values).

	2015		2016		2017		2018	
	Old	New	Old	New	Old	New	Old	New
Base	0.969	0.834	0.925	0.731	0.873	0.445	0.759	0.425
Oracle	0.967	0.924	0.955	0.945	0.952	0.954	0.943	0.938
CFT	0.901	0.940	0.825	**0.950**	0.621	**0.955**	0.656	0.942
LwF [29]	0.920	0.924	0.840	0.915	**0.735**	0.949	0.656	0.919
PODNet [13]	0.911	0.927	0.845	0.929	0.682	0.945	0.651	0.918
Co^{2}L [8]	**0.927**	0.877	**0.875**	0.932	0.572	0.951	0.652	0.942
ResAdapt [41]	0.916	**0.942**	0.850	0.927	0.718	0.951	**0.746**	**0.945**
RETROFIT	**0.935**	**0.933**	**0.926**	**0.939**	**0.882**	**0.956**	**0.877**	**0.940**

We include two ensemble approaches: *hard voting*, which selects the majority prediction among experts, and *soft averaging*, which aggregates their probability outputs. Beyond ensembles, four representative model-merging techniques are considered. *Weight Averaging* [54] directly averages parameters across models; *Task Arithmetic* [23] transfers task-specific knowledge via linear operations on task vectors; *TIES-Merging* [57] resolves conflicting parameters through sign-based majority rules; and *AdaMerging* [59] learns adaptive layer-wise merge weights. Implementation details are provided in Appendix B.3, and their comparative results across both applications are presented in Section 5.4.

5.2 Effectiveness on Temporal Drift

We first evaluate RETROFIT on temporal drift using the long-standing benchmark of Android malware detection, comparing its performance against representative CL baselines.

Evaluation Metrics. Following standard practice, we report the F1-scores on the latest test data (new threats) and all previously encountered test sets (old threats). To quantify overall adaptability across time, we use the AUT metric [37], defined as: $\mathrm{AUT}(N,f) = \frac{1}{N-1}\sum_{k=1}^{N-1}\frac{f(k+1)+f(k)}{2}$, where N is the number of evaluation slots, and $f(k)$ denotes the F1-score at time k. A perfect classifier with no temporal degradation achieves an AUT of 1. To assess retention of previously acquired knowledge, we define the Past-Task Retention (PTR) score: $\mathrm{PTR}_t = \frac{1}{t-1}\sum_{s=1}^{t-1}\frac{a_{t,s}}{a^{\max}_{\le t,s}}$, where $a_{t,s}$ is the F1-score on the s-th test set after learning up to time t, and $a^{\max}_{\le t,s}$ denotes the highest F1-score ever achieved on that set. PTR is consistently

reported at the final time step to summarize forgetting accumulated over the entire sequence. Note that PTR is computed only for continual models that undergo sequential updates across years and not applicable to the static base model, which serves as a fixed reference for absolute performance.

Result Analysis. We present the cumulative performance of RETROFIT and all CL baselines across years in Figure 5, with detailed per-year results on both historical (old) and current (new) test sets shown in Table 1. All CL baselines exhibit reasonable adaptability to newly emerging threats but experience substantial forgetting of earlier malware samples. The naive CFT baseline achieves strong new-year performance but exhibits the worst retention, with its PTR dropping to only 0.666 after the four-year sequential updates. In contrast, RETROFIT improves the retention score by 38.6% over CFT while preserving comparable adaptation to new data. Distillation-based methods such as LwF, PODNet, and Co^{2}L yield moderate improvements in retention (exceeding CFT by $1.3 \sim 2.9\%$) thanks to their knowledge-regularization losses, yet this comes at the cost of adaptability (with AUT values decreasing by about 1.8% on average). Among these, PODNet favors adaptation, whereas LwF favors retention, but their overall performance remains similar, suggesting that regularization alone is insufficient to balance. The architecture-based method ResAdapt achieves the highest retention among the baselines, but at the cost of network expansion, which increases training cost and model size; RETROFIT still surpasses it by 20.2% in PTR while retaining competitive AUT.

Overall, RETROFIT achieves the best balance between retention and adaptability. It improves the PTR by 33.3% across all baselines on average while maintaining or exceeding their AUTs. Being conceptually aligned with distillation-based methods during knowledge arbitration, RETROFIT attains its clear advantage through low-rank model consolidation rather than sample-level knowledge transfer, resulting in superior stability in preserving old knowledge each year. Notably, RETROFIT even outperforms the oracle in most yearly new-test evaluations and the cumulative AUT metric. This suggests that controlled forgetting not only mitigates catastrophic forgetting but also enhances specialization. This phenomenon also exists in other applications, for which we provide a thorougher discussion in Section 6.

5.3 Effectiveness on Representation Shift

We next compare RETROFIT against CL baselines on representation shift using the binary summarization benchmark.

Evaluation Metrics. We use the BLEU score [35] as the primary metric to evaluate the similarity between generated and reference summaries. Given a candidate summary, BLEU is defined as: $\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$, where p_n is the n-gram precision, w_n is the corresponding weight (usually uniform, $1/N$), and BP is the brevity penalty to discourage overly short outputs. We report BLEU scores for all three

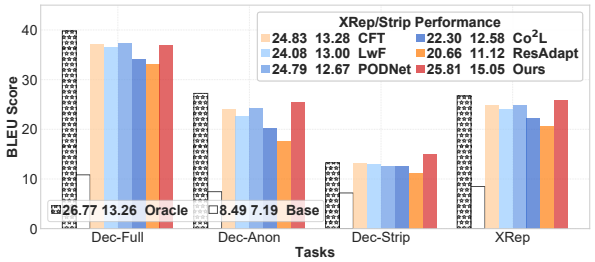


Figure 6: Comparison with CL baselines in binary analysis.

representations to capture both local and cumulative learning effects. Among them, the stripped binaries represent the most security-critical setting and serve as the primary target for continual adaptation. To measure overall generalization across representations, we take the average to define a Cross-Representation (XRep) metric as: $\text{XRep}(R, f) = \frac{1}{R}\sum_{r=1}^{R} f(r)$, where $f(r)$ denotes a performance measure (e.g., BLEU, EM, METEOR [5]) evaluated on representation level r, and R is the total number of representation levels. This metric is a direct analogy to the AUT metric used in previous year-evolving temporal task; here, it measures the model's average performance across representations rather than time. For completeness, we report complementary semantic similarity metrics in the supplementary open-source artifacts repository (which show trends consistent with BLEU) and include a small-scale manual analysis in Section 5.5.3.

Result Analysis. We summarize the performance of all CL baselines and RETROFIT in Figure 6, where models are sequentially trained across decompilation levels to evaluate cross-representation generalization (XRep) and the security-critical stripped binaries ($\text{BLEU}_{\text{strip}}$). Unlike the temporal drift setting, all other baselines underperform the naive CFT, with XRep scores lower by $0.04 \sim 4.17$ and $\text{BLEU}_{\text{strip}}$ lower by $0.28 \sim 2.16$. The architecture-based ResAdapt exhibits the weakest performance on both metrics. This degradation likely arises from its task-specific expansion strategy: while isolating parameters helps mitigate forgetting in disjoint tasks, it disrupts the cumulative transfer needed to generalize across related abstraction levels. Among the distillation-based methods, PODNet attains the highest XRep, which is 0.71 higher than the second-best LwF, but transfers knowledge less effectively to the most obfuscated binaries, dropping by 0.33 BLEU compared to LwF. This trend suggests that existing CL methods, particularly those designed for incremental classification, are less effective in this scenario, as they fail to maintain consistent representation alignment.

In contrast, RETROFIT achieves clear and consistent improvements on both metrics. It raises the BLEU on stripped binaries to 15.05, substantially outperforming transfer-learning methods used in existing research (7.20, +109.0%) and even exceeding the oracle (13.26, +13.5%). At the same time, RETROFIT maintains strong performance across all decompilation levels, achieving an average XRep score of 25.81

Table 2: Comparison with model-preserving baselines across malware detection and binary analysis.

	Malware Detection						Binary Analysis			
	2014	2015	2016	2017	PTR	2018	Full	Anon	Strip	XRep
Ensemble Voting	0.827	0.860	0.904	0.846	**0.905**	**0.876**	22.60	11.01	8.39	14.00
Ensemble Averaging	0.829	**0.860**	0.904	0.843	0.905	0.875	27.52	14.83	8.05	16.80
Weight Averaging [54]	**0.859**	0.857	0.891	0.777	0.891	0.773	25.06	14.17	**10.32**	16.52
Task Arithmetic [23]	0.747	0.833	0.906	**0.912**	0.898	0.853	**32.40**	**17.61**	8.71	**19.57**
Ties-Merging [57]	0.791	0.835	0.897	0.853	0.889	0.875	32.38	16.55	7.84	18.92
AdaMerging [59]	0.751	0.839	**0.908**	0.909	0.895	0.852	24.77	14.01	6.62	15.13
RETROFIT (Ours)	**0.867**	**0.854**	**0.894**	**0.890**	**0.923**	**0.940**	**36.97**	**25.42**	**15.05**	**25.81**

* Rows 1–2 are ensemble-based baselines, while Rows 3–6 list model-merging approaches. For each metric (column), the best-performing value among the six baselines is highlighted in bold; results of our method are bolded throughout for ease of comparison. Some entries may appear numerically identical due to rounding, yet only the true maximum is highlighted.
* Columns 2–7 present results for malware detection, and Columns 8–11 for binary analysis. Within each application, ▢ marks the most security-critical scenario (new threats in 2018 or highly stripped binaries), whereas ▢ indicates generalization performance (retention over earlier years or cross-representation across decompilation levels).

and an improvement of 3.9% over the best baseline. These results confirm that controlled forgetting effectively stabilizes representation learning, enabling cumulative knowledge transfer from easier domains to more obfuscated, data-scarce representations without sacrificing overall generalization.

5.4 Comparison to Aggregation Baselines

We also compare RETROFIT against alternative strategies that preserve individual models instead of sequential adaptation. These approaches aggregate model outputs (ensemble) or parameters (merging) after the training on all domains. We evaluate them across both applications, i.e., aggregating malware classifiers trained from different years and summarization models fine-tuned on distinct representation levels.

Evaluation Setup. We adopt the previously established metrics for each application. For malware detection, model-preserving methods are evaluated through a single aggregation step at the final year (2018). Each resulting model, including RETROFIT, is then tested on all earlier years' test sets as well as the final-year data. Since these approaches do not involve sequential updates, AUT is not applicable, and PTR is instead computed by normalizing each year's performance against the best score achieved by individually trained single-year models. For binary analysis, these methods are similarly adopted by fine-tuning an individual model from CodeT5 for each abstraction level. These models are then aggregated into a final model, evaluated across representation levels.

Result Analysis. We present the overall results in Table 2. In malware detection, we observe an interesting pattern in the model-preserving baselines. Although unit models are trained concurrently, the performance of their aggregated versions is not uniform across all tasks. Instead, they exhibit a distinct performance boost on the middle years (e.g., 2016 and 2017) compared to the edge years. This fluctuation likely arises from partial overlap in malware families or shared behavioral patterns between adjacent years, which causes over-optimization for central tasks during model aggregation. Consequently, this non-uniformity leads to two critical failures. First, these baselines show a severe inability to detect new malware, where the best F1-score on 2018 data is only 0.876, which RETROFIT surpasses by 7.3%. Second, their performance variance across earlier years is substantially higher, with standard deviations $1.7 \sim 3.9$ times greater than that of RETROFIT, dragging their PTR scores 2.7% lower on average.

Unlike the temporal drift scenario, the representation gaps between abstraction levels are more pronounced in binary analysis. As a result, model-merging approaches generally outperform ensemble-based ones, suggesting that direct parameter consolidation better mitigates discrepancies across code abstractions than output-level aggregation. Nevertheless, RETROFIT surpasses all model-preserving baselines by a substantial margin, improving BLEU by 45.8% on stripped binaries and 31.9% in XRep compared to the best-performing baselines, i.e., Weight Averaging and Task Arithmetic, respectively. Compared to prior results in Figure 5 and Figure 6, none of the aggregation baselines surpass existing CL methods, despite their 'unfair' advantage of concurrent access to all task models. This contrast likely stems from the non-IID and highly evolving nature of security data, where static aggregation fails to reconcile domain shifts that violate the assumptions on which these methods rely [56]. Overall, these results reaffirm a key distinction between preservation and accumulation: while model-preserving approaches retain fragmented expertise across tasks, RETROFIT consolidates knowledge effectively through controlled continual adaptation.

5.5 Ablation Analysis and Case Studies

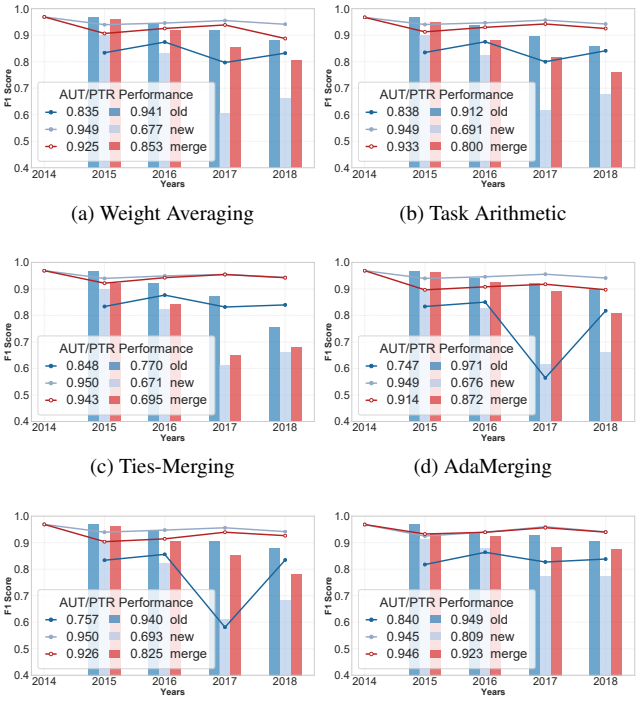
5.5.1 Ablation Analysis

We conduct an ablation study on the malware detection task to validate the contributions of two main components in RETROFIT, with results presented in Figure 7.

Adaptive Merging Strategy. First, to validate the effectiveness of our adaptive merging strategy, we replace it with the four prominent model merging techniques, applying them in the same continual, data-free setting (i.e., merging the old model $f_{\text{old}}^{t=n} \leftarrow f_{\text{merge}}^{t=n-1}$ with the new model $f_{\text{new}}^{t=n}$). The adapted baseline results are presented in the first two rows, where we report the performance of both constituent models and their merged outcome at each year, allowing us to analyze

how different merging strategies balance knowledge. First, Weight Averaging and Task Arithmetic show moderate degradation in both AUT and PTR, confirming that simple linear fusion cannot simultaneously balance old and new knowledge. TIES-Merging maintains adaptability (AUT −0.32%) but its rigid parameter alignment causes it to closely mimic the new model, severely compromising retention (PTR −24.70%). AdaMerging achieves the best retention among baselines, showing that adaptive parameter scaling helps preserve shared representations. However, its AUT decreases the most and its PTR remains 5.53% lower than RETROFIT, indicating that RETROFIT's low-rank, confidence-guided consolidation yields more stable forgetting control.

Low-Rank Constraint. Second, we investigate the role of the low-rank structural constraint. We create a variant that removes the low-rank decomposition. In this setup, the new model is fine-tuned on the new data within the parameter space $\mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$, and our adaptive sparse mask is also learned in that space to merge it with the old model. Results in the last row of the figure confirm that full parameter space merging violates our interference constraints. It induces severe parameter drift, breaking the adaptive mask's ability to consolidate and causing a significant performance drop in both AUT (−2.11%) and PTR (−10.62%). Even so, arbitration alone maintains balanced performance, ranking mid-tier among the four adapted baselines, where two surpass it in AUT and the other two do so in PTR. Compared to AdaMerging, which is adaptive in learning layer-wise coefficients through entropy minimization, our arbitration employs confidence-guided consolidation that pulls logits toward new model when old model's predictions are uncertain, thereby achieving higher AUT but lower PTR, a trade-off for protecting new, security-critical knowledge.

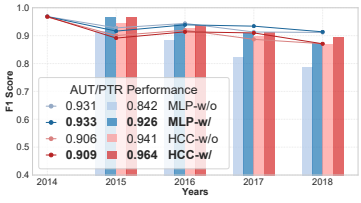


(a) Weight Averaging (b) Task Arithmetic

(c) Ties-Merging (d) AdaMerging

(e) w/o Low-rank (f) RETROFIT

Figure 7: Ablation analysis of RETROFIT.



		MLP		HCC	
		Old	New	Old	New
2015	w/o	0.917	0.928	0.947	0.900
	w/	0.966	0.916	0.965	0.891
	Δ	↑5.33%	↓1.26%	↑1.92%	↓1.01%
2016	w/o	0.884	0.944	0.926	0.920
	w/	0.941	0.939	0.938	0.914
	Δ	↑6.45%	↓0.54%	↑1.20%	↓0.64%
2017	w/o	0.823	0.913	0.897	0.887
	w/	0.908	0.934	0.915	0.910
	Δ	↑10.3%	↑2.29%	↑1.91%	↑2.61%
2018	w/o	0.786	0.911	0.869	0.871
	w/	0.872	0.914	0.893	0.871
	Δ	↑11.1%	↑0.25%	↑2.84%	0.00%

Figure 8: Improvements in retention & adaptation when combining RETROFIT with active learning for different classifiers.

5.5.2 Combination with Active Learning

In malware detection, recent research adopts *active learning* to support model adaptation under limited human-labeling budgets. These approaches typically focus on detecting and labeling new drift samples, using them for retraining or fine-tuning, during which most methods still rely on CFT. Among them, HCC [10] represents a state-of-the-art framework that designs a classifier with hierarchical contrastive learning to intrinsically detect drift during deployment. To evaluate our compatibility with such paradigms, we integrate RETROFIT into HCC's best-performing warm-start procedure. In each iteration, informative samples are selectively labeled under a fixed budget of 5000, after which model adaptation is performed using RETROFIT to assess the improvements over the original adaptation scheme. For completeness, we also include the standard MLP classifier used in our previous experiments, placing it under the same active learning protocol (where drift samples are selected by HCC's pseudo loss sampler) and adapting it with RETROFIT for direct comparison.

Result Analysis. As indicated in Figure 8, the integration of RETROFIT into active learning consistently improves both classifiers. For the MLP classifier, RETROFIT increases AUT by 0.2% and PTR by 10.0%, while for the more advanced HCC, it yields additional gains of 0.3% in AUT and 2.4% in PTR. Higher retention is observed for the MLP classifier compared to the earlier continual fine-tuning results (PTR 0.842 vs. 0.666), although adaptability slightly declines due to the limited labeling budget (AUT −2.0%). RETROFIT restores its adaptability and further strengthens retention, confirming that preservation of our core advantages even when the data-access assumptions differ from retrospective-free setting. The HCC classifier, benefiting from hierarchical contrastive representations, remains the most robust baseline across years, achieving a strong retention score (PTR 0.941) but at the expense of adaptability (AUT lower than MLP by 2.7%). Nevertheless, incorporating RETROFIT raises both metrics simultaneously. A closer year-wise analysis reveals that RETROFIT consis-

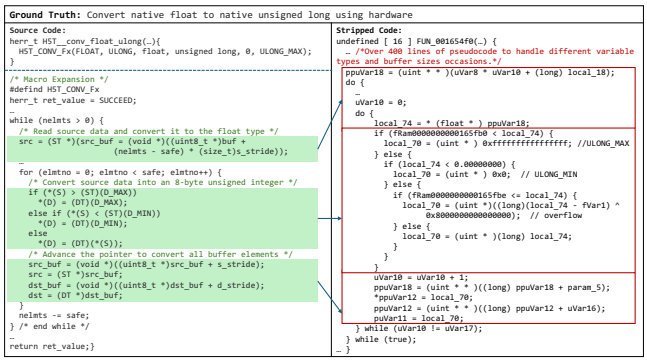


Figure 9: An example of how the function decompilation from a stripped binary significantly expands the source code. RETROFIT achieves an exact match with BLEU of 100, while baselines fail (CFT: 4.92, Oracle: 20.16).

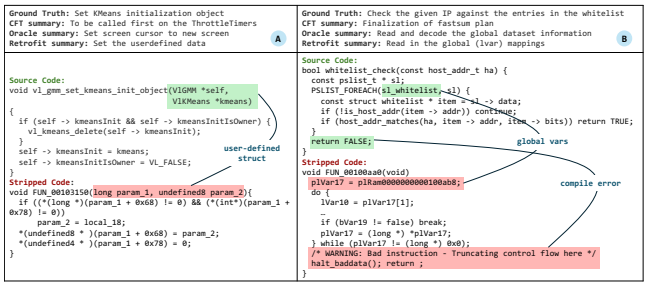


Figure 10: Summary comparison for challenging stripped binaries, where the BLEU of all methods is lower than 20.

tently improves performance on old tasks, particularly in later years, where accumulated knowledge becomes more critical. While minor fluctuations occur in early-year new-test results, the strengthened forward transfer achieved through controlled forgetting ultimately yields superior long-term performance.

5.5.3 Summarization Quality Analysis

While automated metrics like BLEU provide quantitative results, we conduct a qualitative manual analysis, with two primary goals: 1) to understand why the automated scores for stripped binaries are universally low; 2) to demonstrate how RETROFIT provides more interpretable and useful summaries.

Exact Match on Complex Logic. We provide an example in Figure 9 to show the difficulty of summarizing stripped binaries. The original source function (left) already contains intricate logical operations, while its stripped counterpart (right) further obscures semantic intent. Compiler preprocessing and optimizations, such as macro expansion and loop unrolling, inflate a single source statement into more than 400 lines of decompiled pseudocode. RETROFIT achieves a BLEU of 100 in this example, exactly matching the ground truth. It correctly identifies both the operation ('float to unsigned long' conversion) and its implementation detail ('using hardware'), inferred from the three highlighted code patterns. By contrast, the best CL baseline only reaches 4.92 BLEU, misidentifying the source type and the operational detail. The oracle model also performs poorly, summarizing the function as 'Convert native unsigned integer to native float...', inverting the conversion and failing to capture the correct semantics. This case underscores RETROFIT's superior capability to recover precise functional intent from heavily obfuscated binaries.

Semantic Robustness on Limited information. Figure 10 presents two challenging cases where all methods achieve BLEU scores below 20. We identify two factors that explain the universally low scores. 1) *Abstract ground truth*: reference summaries often describe high-level intent unavailable in stripped code, e.g., the metadata of parameter `VlKMeans` in Case A and variable `sl_whitelist` in Case B; 2) *Decompilation artifacts*: the decompiler itself introduces errors, e.g., in Case B, the decompiler fails to propagate a boolean return value, causing truncated control flow. Despite the low scores, RETROFIT produces the most faithful interpretation of the observable logic, whereas baselines often produce irrelevant or hallucinated descriptions. In Case A, its summary accurately captures the data initialization behavior, while both CFT and Oracle hallucinate semantics, misidentifying `param_1` as unrelated entities (e.g., a *ThrottleTimer* or mouse coordinates). In Case B, RETROFIT likewise offers the most plausible interpretation, correctly recognizing the global-variable access pattern in the partial control flow. By contrast, CFT fabricates irrelevant logic, seemingly driven by superficial lexical cues, whereas Oracle overgeneralizes, mistaking conditional whitelist checking for dataset processing.

5.6 Computational and Robustness Analysis

We next analyze RETROFIT's computational efficiency and practical robustness under varying update capacity (e.g., rank and sparsity) and task horizon, supported by empirical statistics from the malware detection application.

Computational Cost. We evaluate the practical overhead of RETROFIT by comparing its per-epoch training time and GPU memory usage with CFT and ResAdapt. CFT represents the most efficient baseline, while ResAdapt represents architecture-based CL methods with network expansion. As shown in Table 3, RETROFIT achieves the lowest GPU memory usage. Its per-epoch runtime is slightly higher than CFT, but remains substantially lower than ResAdapt. Unlike architecture-based methods, RETROFIT maintains a constant model size and preserves the same single forward path as the base model at inference. Compared with the simplest CFT baseline, which trains all parameters at a cost of $O(d_{\text{in}}d_{\text{out}})$ per task, RETROFIT reduces the training complexity by approximately a factor of $r/\min(d_{\text{in}}, d_{\text{out}})$. This primarily lowers update-related memory, including gradients and optimizer states. The modest runtime overhead over CFT mainly comes from mask optimization, whose cost scales linearly with the rank r. Arbitration adds only negligible overhead, i.e., requir-

Table 3: Training time and GPU memory usage comparison.

Method	Time (s/epoch)	Max alloc / reserved (MB)
CFT	1.07 ± 0.01	181.13 / 226.00
ResAdapt	1.67 ± 0.09	316.20 / 350.00
RETROFIT	1.23 ± 0.03	150.19 / 170.00

Table 4: Sensitivity analysis of r and μ. The reference setting ($r = 512$, $\mu = 2 \times 10^{-6}$) is used in the main experiments.

Metric	Ref.	Varying r			Varying μ		
		256	768	1024	10^{-6}	4×10^{-6}	8×10^{-6}
AUT	0.946	0.943	0.947	0.944	0.944	0.945	0.945
PTR	0.923	0.864	0.923	0.928	0.921	0.920	0.918

ing two additional forward passes through the teacher models.

Hyperparameter Sensitivity. The main capacity hyperparameter of RETROFIT is the low-rank dimension r, which controls the size of the update subspace; another important hyperparameter is the sparsity-balancing coefficient μ, which shapes how concentrated the merged update becomes. As shown in Table 4, RETROFIT remains stable across a broad range of both parameters. We do not expect monotonic behavior in r: smaller ranks reduce potential interference, but can also limit the capacity needed to learn transferable updates; larger ranks provide more plasticity, while masking and arbitration control interference. Varying μ introduces only minor performance changes, suggesting that it mainly affects the convergence path toward an effective sparsity pattern rather than the final stability and plasticity trade-off. In practice, we recommend choosing a moderate rank to balance adaptation capacity and interference control, and setting μ within a stable range rather than tuning it aggressively.

Robustness to Update Frequency. To study scalability as the number of tasks increases, we evaluate a month-wise active-learning setting with 48 sequential updates and a labeling budget of 500 samples per month, complementing the year-wise setting in Section 5.5.2. As shown in Figure 11, RETROFIT consistently improves both AUT and PTR across two models. For MLP, RETROFIT improves AUT by 1.6% and PTR by 10.0%; for HCC, it improves AUT by 2.7% and PTR by 1.5%. Notably, finer-grained update cycles with less tuning data amplify interference, underscoring the need for our explicit control. Compared with the year-wise setting, the relative AUT gains are larger on both models, while the PTR gains remain similar, suggesting that RETROFIT scales to longer horizons.

6 Discussion

Oracle Upper Bound. Oracle remains the strongest overall reference, but access to all accumulated data does not eliminate interference. Under temporal or representation shift, joint training must fit heterogeneous past and current distributions with a single uniform objective, which can favor intermediate distributions while diluting specialization to the newest one.

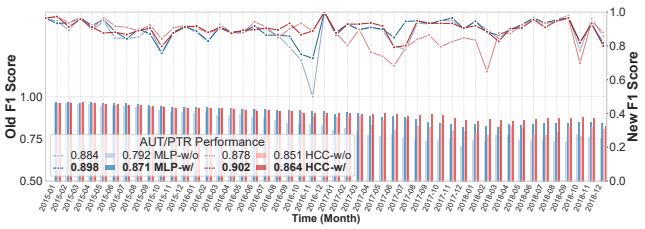


Figure 11: Month-wise active learning-based malware adaptation w/ and w/o RETROFIT, where the task number is 48.

For example, this pattern is visible in Table 1, where Oracle performs better on new test data in the two middle years. Prior work has similarly observed that full training is not always the upper bound [56]. Naive CFT can exceed Oracle on new data, but only with severe forgetting. Thus, RETROFIT's advantage on the new distribution is better understood as a regularization effect: it preserves the specialization benefit of sequential adaptation while controlling forgetting.

Applicable Domain. As shown in Section 5.5.2, RETROFIT is compatible with different model architectures for malware detection. Likewise, it can be extended to emerging pretrained feature extractors like DexBERT [48]. In binary analysis, similar arbitration mechanisms can be applied to other LLMs [45]. Future work may further explore RETROFIT under other forms of representation shift arising in security-critical contexts, e.g., binaries compiled with varying optimization levels, architectures, or platforms, and extend it naturally to related downstream tasks including binary similarity detection [12] and vulnerability discovery [30]. More broadly, RETROFIT offers a general solution for continual model adaptation in data-sensitive security applications.

Combining with Replay-based CL. While RETROFIT is designed for data-restricted environments, it can be complemented with replay-based CL strategies when data retention policies are more permissive. In such settings, a small memory buffer provides representative samples from past tasks, improving coverage of underrepresented or highly heterogeneous behaviors. For instance, MADAR [39] demonstrates that distribution-aware replay enhances malware classification by accounting for heterogeneity both across and within families. Similar replay mechanisms can be adapted for different applications, and incorporating them into RETROFIT has the potential to strengthen retention, but at the cost of additional storage and training overhead.

7 Related Work

Representation Learning for Robustness. Representation learning methods such as contrastive learning [9, 10] and invariant risk minimization [1] aim to extract invariant features that remain stable under distributional shifts. For example, hierarchical contrastive learning employed in HCC [10] enhances malware detection by separating benign and malicious

samples while capturing intra-family semantics. In security analytics, related efforts refine API representations through knowledge graphs [65] and structure-aware encoders that integrate control-flow information [36]. While sharing the goal of preserving performance amid changing conditions, they operate in static settings; instead, CL adapts representations dynamically as new datasets become available, making the two approaches naturally complementary.

MTL and Model Merging. Multi-task learning (MTL) [66] learns shared representations across related tasks but requires simultaneous access to all task data. To bypass this constraint, model merging has emerged as an alternative to execute MTL, combining independently trained models post hoc rather than through joint training on all tasks [34, 55]. To mitigate task interference, TIES-Merging [57] applies a majority vote on the sign for each parameter, whereas AdaMerging [59] learns adaptive merge weights at the task or layer level. We integrate merging into CL settings and employ low-rank and adaptive sparse mask learning to regulate conflicts between old and new models. Our ablation study in Section 5.5.1 further confirms reduced interference when replacing our design with existing merging methods.

Drift Detection and Active Learning. Drift detection aims to identify samples that deviate significantly from the training distribution, often by monitoring prediction uncertainty or changes in the embedding distribution over time [6, 31]. In security analytics, such mechanisms have been widely used to detect emerging malware families or new attack behaviors [19, 61]. Active learning complements this process by selecting those detected samples for human labeling, thus reducing annotation costs while guiding model retraining with an expanded dataset [10, 65]. As demonstrated in Section 5.5.2, our method serves as an adaptation extension that seamlessly integrates into this online workflow.

Code Analysis for Security. Recent advances in LLMs have opened new directions for automated code understanding in security applications. A key research question is how well these models comprehend low-readability code [14, 24]. This exploration is often hampered by the scarcity of ground-truth data, which forces reliance on proxy metrics [20]. Our work applies RETROFIT to binary summarization, using a recent dataset compiled from open-source (non-malware) projects that provides ground-truth summaries from source code comments [2]. While this benchmark is valuable, we also note its remaining flaws (discussed in Section 5.5.3), highlighting the need for further dataset refinement in binary analysis.

8 Conclusion

This paper tackles continual learning in data-sensitive binary security domains, where models must evolve without access to prior data. We identified catastrophic forgetting and distributional generalization failure as key barriers to long-term adaptability under such constraints. To address these, we propose RETROFIT, which formulates knowledge accumulation as model consolidation and employs low-rank structural constraints with confidence-guided sparse regulation to achieve controlled forgetting. Empirical results across two representative applications, i.e., malware detection and binary analysis, demonstrate that RETROFIT sustains strong retention and effective adaptation across model architectures and data distributions. These results establish RETROFIT as a practical and effective foundation for trustworthy continual learning.

Acknowledgments

We would like to thank the anonymous reviewers and our shepherd for their insightful comments and valuable guidance to help improve this paper. This work is supported in part by Google ASPIRE, Google Academic Research Awards, the "Pioneer" and "Leading Goose" R&D Program of Zhejiang (Grant No. 2024C01169), and the National Natural Science Foundation of China under Grants 62441238 and U2441240.

Ethical Considerations

This work focuses on continual learning for defensive binary security applications, including malware detection and binary analysis. The main stakeholders are *security analysts*, *incident-response teams*, *software vendors*, and *end users*, all of whom may benefit from more robust model adaptation under distribution shift. By reducing adaptation failures as threats evolve or program representations degrade, such systems can improve detection reliability, analysis quality, and operational continuity.

The main operational harms arise from misclassifications. *False negatives* may allow malicious binaries to evade detection, delaying triage, remediation, or vulnerability discovery, harming end users and reducing trust in software vendors and security providers. *False positives* may generate unnecessary alerts, waste analyst effort, disrupt vendor release workflows, or block legitimate software for end users. Publication also carries limited *dual-use risk*, since advances in binary analysis may be studied by adversarial users.

This work is explicitly scoped to *defensive use* and does not introduce new capabilities for malware creation, obfuscation, or evasion. RETROFIT is intended to *reduce adaptation failures under drift*, but not to replace human judgment in high-stakes settings. In addition, the retrospective-free design avoids storing or replaying historical binaries, which *reduces dependence on long-term data access* to sensitive malware samples and proprietary software artifacts.

We judged that the expected benefits of improving robustness in defensive malware detection and binary analysis outweigh the remaining risks, particularly because the work can

directly benefit stakeholders through improved binary analysis reliability while reducing dependence on sensitive data.

Open Science

We make the artifacts available at Zenodo and GitHub. The shared artifacts consist of a complete implementation of our proposed continual learning framework RETROFIT, along with all reproducible workflows necessary to adapt, merge, and evaluate models in the two applications described in the paper. A detailed README describes the code structure and usage. Due to licensing and redistribution constraints, the two datasets and pretrained BinT5 models are not included in the repository; instead, we provide instructions for obtaining them from their original sources and release our processed APK features and fine-tuned BinT5 checkpoints.

References

[1] Kartik Ahuja, Karthikeyan Shanmugam, Kush Varshney, and Amit Dhurandhar. Invariant risk minimization games. In *International Conference on Machine Learning*, pages 145–155. PMLR, 2020.

[2] Ali Al-Kaswan, Toufique Ahmed, Maliheh Izadi, Anand Ashok Sawant, Premkumar Devanbu, and Arie van Deursen. Extending source code pre-trained language models to summarise decompiled binaries. In *2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 260–271. IEEE, 2023.

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

[4] Daniel Arp, Michael Spreitzenbarth, Malte Hübner, Hugo Gascon, and Konrad Rieck. Drebin: Effective and explainable detection of android malware in your pocket. In *Network and Distributed System Security Symposium (NDSS)*, volume 14, pages 23–26, 2014.

[5] Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72, 2005.

[6] Federico Barbero, Feargus Pendlebury, Fabio Pierazzi, and Lorenzo Cavallaro. Transcending transcend: Revisiting malware classification in the presence of concept drift. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 805–823. IEEE, 2022.

[7] Ying Cao, Runze Zhang, Ruigang Liang, and Kai Chen. Evaluating the effectiveness of decompilers. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 491–502, 2024.

[8] Hyuntak Cha, Jaeho Lee, and Jinwoo Shin. Co2l: Contrastive continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9516–9525, October 2021.

[9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PmLR, 2020.

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

[11] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021.

[12] Steven HH Ding, Benjamin CM Fung, and Philippe Charland. Asm2vec: Boosting static representation robustness for binary clone search against code obfuscation and compiler optimization. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 472–489. IEEE, 2019.

[13] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *European Conference on Computer Vision*, pages 86–102. Springer, 2020.

[14] Chongzhou Fang, Ning Miao, Shaurya Srivastav, Jialin Liu, Ruoyu Zhang, Ruijie Fang, Asmita, Ryan Tsang, Najmeh Nazari, Han Wang, and Houman Homayoun. Large language models for code analysis: Do LLMs really do their job? In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 829–846, Philadelphia, PA, August 2024. USENIX Association.

[15] Mingzhe Gao, Ligeng Chen, Yiling He, Yuhang Chen, Lingyun Ying, and Wang Yang. Distilling benign knowledge with fine-grained ast fragments for precise real-world web shell detection. In *2025 IEEE/ACM 33rd International Symposium on Quality of Service (IWQoS)*, pages 1–10, 2025.

[16] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. Adversarial examples for malware detection. In *European symposium on research in computer security*, pages 62–79. Springer, 2017.

[17] Zhen Guo, Abhinav Kumar, and Reza Tourani. Persistent backdoor attacks in continual learning. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 6379–6397, 2025.

[18] Tyler L Hayes, Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan. Remind your neural network to prevent catastrophic forgetting. In *European conference on computer vision*, pages 466–483. Springer, 2020.

[19] Yiling He, Junchi Lei, Zhan Qin, Kui Ren, and Chun Chen. Combating concept drift with explanatory detection and adaptation for android malware classification. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pages 978–992, 2025.

[20] Yiling He, Hongyu She, Xingzhi Qian, Xinran Zheng, Zhuo Chen, Zhan Qin, and Lorenzo Cavallaro. On benchmarking code llms for android malware analysis. In *Proceedings of the 34th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 153–160, 2025.

[21] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *International Conference on Learning Representations (ICLR)*, 1(2):3, 2022.

[22] IBM. 2025 threat intelligence index. Technical report, IBM Institute for Business Value, Oct 2025. URL: https://www.ibm.com/thought-leadership/institute-business-value/en-us/report/2025-threat-intelligence-index.

[23] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

[24] Xin Jin, Jonathan Larson, Weiwei Yang, and Zhiqiang Lin. Binary code summarization: Benchmarking chatgpt/gpt-4 and other large language models. *arXiv preprint arXiv:2312.09601*, 2023.

[25] Roberto Jordaney, Kumar Sharad, Santanu K Dash, Zhi Wang, Davide Papini, Ilia Nouretdinov, and Lorenzo Cavallaro. Transcend: Detecting concept drift in malware classification models. In *26th USENIX security symposium (USENIX security 17)*, pages 625–642, 2017.

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

[27] Shangqi Lai, Xingliang Yuan, Amin Sakzad, Mahsa Salehi, Joseph K Liu, and Dongxi Liu. Enabling efficient privacy-assured outlier detection over encrypted incremental data sets. *IEEE Internet of Things Journal*, 7(4):2651–2662, 2019.

[28] Matthew Lau, ISMAILA SECK, Athanasios P Meliopoulos, Wenke Lee, and Eugene Ndiaye. Revisiting non-separable binary classification and its applications in anomaly detection. *Transactions on Machine Learning Research*, 2024.

[29] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[30] Peiyu Liu, Junming Liu, Lirong Fu, Kangjie Lu, Yifan Xia, Xuhong Zhang, Wenzhi Chen, Haiqin Weng, Shouling Ji, and Wenhai Wang. Exploring {ChatGPT's} capabilities on vulnerability management. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 811–828, 2024.

[31] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE transactions on knowledge and data engineering*, 31(12):2346–2363, 2018.

[32] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.

[33] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[34] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3994–4003, 2016.

[35] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[36] Kexin Pei, Weichen Li, Qirui Jin, Shuyang Liu, Scott Geng, Lorenzo Cavallaro, Junfeng Yang, and Suman Jana. Exploiting code symmetries for learning program semantics. In *Proceedings of the 41st International Conference on Machine Learning*, pages 40092–40113, 2024.

[37] Feargus Pendlebury, Fabio Pierazzi, Roberto Jordaney, Johannes Kinder, and Lorenzo Cavallaro. Tesseract: Eliminating experimental bias in malware classification across space and time. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 729–746, 2019.

[38] Xingzhi Qian, Xinran Zheng, Yiling He, Shuo Yang, and Lorenzo Cavallaro. Lamd: Context-driven android malware detection and classification with llms. In *2025 IEEE Security and Privacy Workshops (SPW)*, pages 126–136. IEEE, 2025.

[39] Mohammad Saidur Rahman, Scott Coull, Qi Yu, and Matthew Wright. Madar: Efficient continual learning for malware analysis with distribution-aware replay. In *Proceedings of the Conference on Applied Machine Learning in Information Security (CAMLIS)*, 2025.

[40] Mathilde Raynal and Carmela Troncoso. On the conflict between robustness and learning in collaborative machine learning. In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 2171–2189. IEEE, 2025.

[41] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. *Advances in neural information processing systems*, 30, 2017.

[42] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[43] Pengcheng Ren, Chaoshun Zuo, Xiaofeng Liu, Wenrui Diao, Qingchuan Zhao, and Shanqing Guo. Demistify: Identifying on-device machine learning models stealing and reuse vulnerabilities in mobile apps. In *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE)*, pages 468–480. IEEE Computer Society, 2023.

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

[45] Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.

[46] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International conference on machine learning*, pages 4548–4557. PMLR, 2018.

[47] Zian Su, Xiangzhe Xu, Ziyang Huang, Kaiyuan Zhang, and Xiangyu Zhang. Source code foundation models are transferable binary analysis knowledge bases. *Advances in Neural Information Processing Systems*, 37:112624–112655, 2024.

[48] Tiezhu Sun, Kevin Allix, Kisub Kim, Xin Zhou, Dongsun Kim, David Lo, Tegawendé F Bissyandé, and Jacques Klein. Dexbert: Effective, task-agnostic and fine-grained representation learning of android bytecode. *IEEE Transactions on Software Engineering*, 49(10):4691–4706, 2023.

[49] Mingxing Tan and Quoc Le. Efficientnetv2: Smaller models and faster training. In *International conference on machine learning*, pages 10096–10106. PMLR, 2021.

[50] Yuvraj Virk, Premkumar Devanbu, and Toufique Ahmed. Calibration of large language models on code summarization. *Proceedings of the ACM on Software Engineering*, 2(FSE):2944–2964, 2025.

[51] 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.

[52] Yue Wang, Weishi Wang, Shafiq Joty, and Steven CH Hoi. Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8696–8708, 2021.

[53] Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, and Saining Xie. Convnext v2: Co-designing and scaling convnets with masked autoencoders. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16133–16142, 2023.

[54] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pages 23965–23998. PMLR, 2022.

[55] Chengyue Wu, Teng Wang, Yixiao Ge, Zeyu Lu, Ruisong Zhou, Ying Shan, and Ping Luo. π-tuning: Transferring multimodal foundation models with optimal multi-task interpolation. In *International Conference on Machine Learning*, pages 37713–37727. PMLR, 2023.

[56] Zihao Wu, Huy Tran, Hamed Pirsiavash, and Soheil Kolouri. Is multi-task learning an upper bound for continual learning? In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[57] Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36:7093–7115, 2023.

[58] Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. Model merging in llms, mllms, and beyond: Methods, theories, applications, and opportunities. *ACM Computing Surveys*, 2026.

[59] Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. Adamerging: Adaptive model merging for multi-task learning. In *International Conference on Learning Representations (ICLR)*, 2024.

[60] Limin Yang, Arridhana Ciptadi, Ihar Laziuk, Ali Ahmadzadeh, and Gang Wang. Bodmas: An open dataset for learning based temporal analysis of pe malware. In *2021 IEEE Security and Privacy Workshops (SPW)*, pages 78–84. IEEE, 2021.

[61] Limin Yang, Wenbo Guo, Qingying Hao, Arridhana Ciptadi, Ali Ahmadzadeh, Xinyu Xing, and Gang Wang. {CADE}: Detecting and explaining concept drift samples for security applications. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2327–2344, 2021.

[62] Wei Ying, Yu Zhang, Junzhou Huang, and Qiang Yang. Transfer learning via learning to transfer. In *International conference on machine learning*, pages 5085–5094. PMLR, 2018.

[63] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations (ICLR)*, 2018.

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

[65] Xiaohan Zhang, Yuan Zhang, Ming Zhong, Daizong Ding, Yinzhi Cao, Yukun Zhang, Mi Zhang, and Min Yang. Enhancing state-of-the-art classifiers with api semantics to detect evolved android malware. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 757–770, 2020.

[66] Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE transactions on knowledge and data engineering*, 34(12):5586–5609, 2021.

A Proofs

We consider a high-dimensional regime where the input dimension d_{in} is large relative to the update rank r. For any task t, the random projection matrix $A_t \in \mathbb{R}^{d_{\text{in}} \times r}$ is initialized with i.i.d. entries such that $\mathbb{E}[(A_t)_{ik}] = 0$ and $\text{Var}((A_t)_{ik}) = \sigma^2 > 0$ (as determined by the chosen initializer, e.g., Gaussian or Kaiming uniform). In particular, a common normalization for random projections sets $\sigma^2 = 1/d_{\text{in}}$. Consequently, the following moment identities hold:

$$\mathbb{E}[A_t^\top A_t] = \sigma^2 d_{\text{in}} I_r, \qquad \mathbb{E}[A_t A_t^\top] = \sigma^2 r I_{d_{\text{in}}}. \tag{11}$$

Given sub-Gaussian entries of A_t, for any fixed matrix $B \in \mathbb{R}^{r \times d_{\text{out}}}$, the norm of $A_t B$ concentrates: for any $\varepsilon \in (0,1)$,

$$\|A_t B\|_F^2 \geq (1-\varepsilon)\sigma^2 d_{\text{in}} \|B\|_F^2 \tag{12}$$

holds with probability at least $1-\delta$, where the failure probability satisfies $\delta \leq 2\exp\left(-c\,\varepsilon^2 d_{\text{in}}\right)$.

A.1 Proofs of Proposition 1

Let $s < t$ index an old task and a new task, respectively, and let $g_s := \nabla_\theta \mathcal{L}_s(\theta_{\text{prev}}) \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ denote the gradient of a past task. The new-task update is $\Delta W_t := A_t B_t$, where $B_t \in \mathbb{R}^{r \times d_{\text{out}}}$ is fixed and the expectation is taken over the random initialization of $A_t \in \mathbb{R}^{d_{\text{in}} \times r}$. The goal is to bound the expected cosine similarity between g_s and ΔW_t.

Using $\langle U, V \rangle_F = \text{Tr}(U^\top V)$ and the cyclic property, we have

$$\left|\langle g_s, \Delta W_t \rangle_F\right| = \left|\langle A_t^\top g_s, B_t \rangle_F\right| \leq \|A_t^\top g_s\|_F \, \|B_t\|_F. \tag{13}$$

Normalizing by Frobenius norms yields

$$\frac{\langle g_s, \Delta W_t \rangle_F}{\|g_s\|_F \, \|\Delta W_t\|_F} \leq \frac{\|A_t^\top g_s\|_F}{\|g_s\|_F} \cdot \frac{\|B_t\|_F}{\|A_t B_t\|_F}. \tag{14}$$

Numerator Bound. Using the moment identity $\mathbb{E}[A_t A_t^\top] = \sigma^2 r I_{d_{\text{in}}}$ from (11):

$$\begin{aligned}\mathbb{E}_{A_t}\left[\|A_t^\top g_s\|_F^2\right] &= \mathbb{E}_{A_t}\left[\text{Tr}\left(g_s^\top A_t A_t^\top g_s\right)\right] = \text{Tr}\left(g_s^\top \mathbb{E}[A_t A_t^\top] g_s\right) \\ &= \text{Tr}\left(g_s^\top (\sigma^2 r I_{d_{\text{in}}}) g_s\right) = \sigma^2 r \|g_s\|_F^2. \end{aligned} \tag{15}$$

By Jensen's inequality, the expected norm is bounded by

$$\mathbb{E}_{A_t}\left[\frac{\|A_t^\top g_s\|_F}{\|g_s\|_F}\right] \leq \sqrt{\mathbb{E}_{A_t}\left[\frac{\|A_t^\top g_s\|_F^2}{\|g_s\|_F^2}\right]} = \sigma\sqrt{r}. \tag{16}$$

Denominator Bound. Applying the concentration result from (12), for a given $\varepsilon \in (0,1)$, there exists an event $\mathcal{E}$ with probability at least $1-\delta$ such that:

$$\|A_t B_t\|_F^2 \geq (1-\varepsilon)\,\sigma^2 d_{\text{in}}\,\|B_t\|_F^2. \tag{17}$$

On this event, the second term of (14) satisfies

$$\frac{\|B_t\|_F}{\|A_t B_t\|_F} \leq \frac{1}{\sigma\sqrt{(1-\varepsilon)\,d_{\text{in}}}}. \tag{18}$$

Expected Interference. Since the cosine similarity is bounded by 1 via Cauchy-Schwarz, we split the expectation over $\mathcal{E}$ and its complement $\mathcal{E}^c$. Combining (16) and (18):

$$\begin{aligned}\mathbb{E}_{A_t}\left[\frac{\langle g_s, \Delta W_t\rangle_F}{\|g_s\|_F, \|\Delta W_t\|_F}\right] &\leq \frac{\sigma\sqrt{r}}{\sigma\sqrt{(1-\varepsilon)d_{\text{in}}}} + \mathbb{P}(\mathcal{E}^c) \\ &= \frac{1}{\sqrt{1-\varepsilon}}\sqrt{\frac{r}{d_{\text{in}}}} + \delta = O\left(\sqrt{\tfrac{r}{d_{\text{in}}}}\right).\end{aligned} \tag{19}$$

A.2 Proofs of Proposition 2

Let $A_s, A_t \in \mathbb{R}^{d_{\text{in}} \times r}$ be independent random matrices for distinct tasks $s \neq t$. Let $\widetilde{B}_s := B_s \odot M_s$ and $\widetilde{B}_t := B_t \odot M_t$. The task-specific updates are defined as $\Delta_s := A_s\widetilde{B}_s$ and $\Delta_t := A_t\widetilde{B}_t$. The goal is to prove the normalized inner product of independent Δ_s and Δ_t converges to 0 in probability.

The Frobenius inner product is given by:

$$\langle \Delta_s, \Delta_t\rangle_F = \text{Tr}(\Delta_s^\top \Delta_t) = \text{Tr}(\widetilde{B}_s^\top A_s^\top A_t \widetilde{B}_t). \tag{20}$$

Let $X := A_s^\top A_t \in \mathbb{R}^{r\times r}$ represent the interaction matrix between the two random projections. By the sub-multiplicative property of the Frobenius norm,

$$\frac{|\langle \Delta_s, \Delta_t\rangle_F|}{\|\Delta_s\|_F\|\Delta_t\|_F} \leq \|X\|_F \cdot \frac{\|\widetilde{B}_s\|_F}{\|A_s\widetilde{B}_s\|_F} \cdot \frac{\|\widetilde{B}_t\|_F}{\|A_t\widetilde{B}_t\|_F}. \tag{21}$$

Interaction Matrix. For each (k,ℓ), $X_{k\ell} = \sum_{i=1}^{d_{\text{in}}} (A_s)_{ik}(A_t)_{i\ell}$ is a sum of d_{in} independent random variables with mean 0, and $\mathbb{E}[X_{k\ell}^2] = d_{\text{in}}\sigma^4$. Thus, the expected squared Frobenius norm is $\mathbb{E}\|X\|_F^2 = \sum_{k=1}^r \sum_{\ell=1}^r \mathbb{E}[X_{k\ell}^2] = r^2 d_{\text{in}}\sigma^4$. By Markov's inequality applied to $\|X\|_F^2$, it follows that

$$\|X\|_F = O_p\left(r\sigma^2\sqrt{d_{\text{in}}}\right). \tag{22}$$

Update Norms. Using the concentration result from (12), for any $\varepsilon \in (0,1)$, the following lower bounds hold with high probability as $d_{\text{in}} \to \infty$:

$$\|\Delta_s\|_F^2 \geq (1-\varepsilon)\sigma^2 d_{\text{in}}\|\widetilde{B}_s\|_F^2, \quad \|\Delta_t\|_F^2 \geq (1-\varepsilon)\sigma^2 d_{\text{in}}\|\widetilde{B}_t\|_F^2. \tag{23}$$

This implies that the norm ratios are bounded by

$$\frac{\|\widetilde{B}_s\|_F}{\|\Delta_s\|_F} \leq \frac{1}{\sigma\sqrt{(1-\varepsilon)d_{\text{in}}}}, \quad \frac{\|\widetilde{B}_t\|_F}{\|\Delta_t\|_F} \leq \frac{1}{\sigma\sqrt{(1-\varepsilon)d_{\text{in}}}}. \tag{24}$$

Orthogonality. Plugging (22) and (24) into (21) yields

$$\begin{aligned}\frac{|\langle \Delta_s, \Delta_t\rangle_F|}{\|\Delta_s\|_F\|\Delta_t\|_F} &\leq \frac{O_p(r\sigma^2\sqrt{d_{\text{in}}})}{(1-\varepsilon)\sigma^2 d_{\text{in}}} \\ &= O_p\left(\frac{r}{\sqrt{d_{\text{in}}}}\right) \xrightarrow{\text{p}} 0,\end{aligned} \tag{25}$$

Since $r = o(\sqrt{d_{\text{in}}})$ (which is satisfied if r is constant or growing slower than the square root of the input dimension), the cosine similarity converges in probability to zero.

A.3 Local Model-Level Interpretation

Let $f_\theta(x)$ denote the logit vector of the model and let $\Delta\theta_t$ be the update introduced at round t. For small updates, a first-order expansion gives

$$f_{\theta+\Delta\theta_t}(x) - f_\theta(x) = J_\theta(x)\Delta\theta_t + R_2(x), \tag{26}$$

where $J_\theta(x) = \nabla_\theta f_\theta(x)$ is the parameter Jacobian, and $R_2(x)$ is the second-order Taylor remainder, which captures the higher-order nonlinear effect of the update. If f_θ is locally L-smooth around θ, then

$$\|R_2(x)\|_2 \leq \frac{L}{2}\|\Delta\theta_t\|_2^2. \tag{27}$$

Therefore,

$$\|f_{\theta+\Delta\theta_t}(x) - f_\theta(x)\|_2 \leq \|J_\theta(x)\|_2\|\Delta\theta_t\|_2 + \frac{L}{2}\|\Delta\theta_t\|_2^2. \tag{28}$$

Taking expectation over old data $x \sim \mathcal{D}_{\leq t-1}$ gives

$$\mathbb{E}\|f_{\theta+\Delta\theta_t}(x) - f_\theta(x)\|_2 \leq G_{\text{old}}\|\Delta\theta_t\|_2 + \frac{L}{2}\|\Delta\theta_t\|_2^2, \tag{29}$$

where $G_{\text{old}} = \mathbb{E}\|J_\theta(x)\|_2$. For RETROFIT, $\Delta\theta_t$ is controlled by the masked coefficient energy $E_t = \|M_t \odot B_t\|_F$. Limiting E_t through low-rank structure and sparse masking bounds the Jacobian-projected output drift on old distributions, providing a model-level interpretation of controlled forgetting under local smoothness and small-update assumptions.

B Implementation Details

B.1 RETROFIT Implementation

Low-rank modules are applied to all linear layers in the MLP-based malware detection model. For the transformer-based CodeT5 used in binary analysis, those modules are inserted into key subcomponents, including the self-attention projections (`q`, `k`, `v`, `o`) and the feed-forward layers (`wi`, `wo`). We apply a real-valued learnable mask M_t to modulate the new-task LoRA matrices, where rank r is set to 512 in both applications. The mask is initialized with a sparsity ratio: 0.01 in malware detection for strict forgetting control, and 1.0 in binary analysis for a more progressive accumulation of new knowledge. The initialization facilitates convergence, while the effective sparsity is learned automatically. There are several hyperparameters involved in performing the knowledge arbitration. For malware detection, the sparsity-balancing parameter in Equation 8 is set to $\lambda = 1.0$, and the supervised loss coefficient and group-lasso regularization coefficient in Equation 9 to $\eta = 1.0$ and $\mu = 2 \times 10^{-6}$, respectively. For binary analysis, we use the same supervised loss coefficient $\eta = 1.0$ but adjust $\lambda = 0.1$ and $\mu = 1.0$ to accommodate the higher capacity of the transformer-based model.

B.2 CL Baselines

For distillation-based baselines, the model is fine-tuned on each new task by augmenting the standard cross-entropy objective with a knowledge-distillation term, formulated as $\mathcal{L} = \lambda_{\text{ce}}\,\mathcal{L}_{\text{ce}} + \lambda_{\text{kd}}\,\mathcal{L}_{\text{kd}}$. Different methods define $\mathcal{L}_{\text{kd}}$ in distinct ways: LwF uses softened output logits from the teacher, PODNet constrains feature representations across layers, and Co^{2}L aligns pairwise embedding similarities.

LwF: The distillation is calculated as the cross-entropy between the softened teacher and student probabilities, where are $p_t^{(i)} = \frac{\exp(z_t^{(i)}/T)}{\sum_j \exp(z_t^{(j)}/T)}$ and $p_s^{(i)} = \frac{\exp(z_s^{(i)}/T)}{\sum_j \exp(z_s^{(j)}/T)}$. The temperature $T > 1$ smooths the predictive distribution and conveys information about inter-class relationships. Through experiments over $T \in \{1.5, 2.0, 4.0\}$, we find that smaller temperatures produce overly peaked teacher distributions with limited relational transfer, whereas larger temperatures over-flatten the distribution and reduce the effectiveness of distillation; accordingly, we use $T = 2.0$. Based on experimental observations, we determine distillation weights that provide a balance between different training objectives. We select $\lambda_{\text{ce}} = 1.0$ and $\lambda_{\text{kd}} = 1.0$ for malware detection, and $\lambda_{\text{ce}} = 1.0$ and $\lambda_{\text{kd}} = 0.1$ for binary analysis, where the distillation loss is averaged over all non-`[PAD]` tokens for sequence-generation tasks.

PODNet: Its distillation term matches ℓ_2-normalized, pooled activations between the student and teacher across multiple layers and pooling scales. We follow this distillation formulation to implement PODNet but omit its local similarity classifier designed for class-incremental learning. For malware detection, we align the normalized activations from two hidden layers via chunk-wise pooling. For binary analysis, the spatial pyramid is achieved with a lightweight token-mean alignment module applied to the encoder's last four layers, and feature matching is performed through mean-squared alignment of teacher and student embeddings. The PODNet loss weight is examined in the range $\lambda_{\text{pod}} \in \{0.5, 0.1, 0.2\}$ to control the strength of representation matching. In practice, we fix $\lambda_{\text{pod}} = 0.1$, which provides sufficient regularization without over-constraining new-task adaptation.

Co^{2}L: It combines a supervised contrastive learning component with an instance-wise relation distillation module: $\lambda_{\text{con}}\mathcal{L}_{\text{supcon}} + \lambda_{\text{ird}}\mathcal{L}_{\text{IRD}}$. While $\mathcal{L}_{\text{supcon}}$ shapes the representation space by pulling together positive pairs and separating negatives, $\mathcal{L}_{\text{IRD}}$ specifically preserves the pairwise similarity patterns between student and teacher embeddings. To balance supervised learning, contrastive alignment, and instance-wise relation distillation, we examine loss weights in $\{0.5, 1.0, 2.0\}$. For malware detection, we use the penultimate-layer representations as embedding vectors for contrastive alignment. The loss weights are set to λ_{ce}=1.0, λ_{con}=1.0, and λ_{ird}=0.5. In the binary analysis experiments, the encoder representations are mean-pooled across tokens and passed through a two-layer projection head for contrastive alignment. The overall objective is weighted by λ_{ce}=0.5, λ_{con}=1.0, and λ_{ird}=0.5, respectively.

For architecture-based methods, **ResAdapt** introduces lightweight plug-in residual adapters into the backbone network while preserving the pretrained weights. Each adapter augments the representational capacity of the frozen layers through residual bottleneck projections: $\text{Adapter}(x) = x + U\big(\text{ReLU}(D(\text{LN}(x)))\big)$, where LN denotes layer normalization, D and U are down- and up-projection matrices defining the bottleneck. To avoid relying on domain labels, we continuously fine-tune the separate adapter. For tuning, we vary its bottleneck dimension $b \in \{64, 128, 256, 512\}$ to balance the new and old threats, especially for competitive new-task performance compared to other baselines. For malware detection, the adapters are injected into intermediate layers of the MLP backbone, with the bottleneck dimension set to 64. For binary analysis, the residual adapter is extended to the transformer-based CodeT5 architecture, inserting adapters after each T5BLOCK in both the encoder and decoder stacks and using a bottleneck dimension of 512.

B.3 Aggregation Baselines

The two ensemble baselines: (1) **Ensemble Voting** computes the final prediction by taking the majority label among all models' predictions, i.e., $\hat{y} = \text{mode}\big(\{\hat{y}_t\}_{t=1}^n\big)$ where $\hat{y}_t$ is the prediction from the t^{th} candidate model. (2) **Ensemble Averaging** averages logits from all models and applies a sigmoid function, expressed as $\hat{y} = \sigma\big(\frac{1}{n}\sum_{t=1}^n z_t(x)\big)$, where $z_t(x)$ denotes the logit output of the t^{th} model.

The four model merging baselines: (1) **Weight Averaging** computes the element-wise weighted mean of the parameters of n candidate models and can be formulated as $\boldsymbol{\theta}_{\text{m}} = \frac{1}{n}\sum_{t=1}^n \boldsymbol{\theta}_t$, where $\boldsymbol{\theta}_t$ denotes the parameter vector of the t-th model and $\boldsymbol{\theta}_{\text{m}}$ the merged parameters. (2) **Task Arithmetic** constructs task vectors of the n models relative to a pretrained base model and adds their scaled sum back to the base parameters, given by $\boldsymbol{\theta}_{\text{m}} = \boldsymbol{\theta}_{\text{base}} + \alpha\sum_{t=1}^n \boldsymbol{\tau}_t$, where $\boldsymbol{\tau}_t = \boldsymbol{\theta}_t - \boldsymbol{\theta}_{\text{base}}$ is the task vector of model t, α is a scaling coefficient, and $\boldsymbol{\theta}_{\text{base}}$ is the parameter vector of the base model. we set α=0.3. (3) **Ties-Merging** extends Task Arithmetic by pruning parameter entries with conflicting signs and retaining only the top-K% magnitude components before combination. For malware detection, we set K=5 and α=0.3; for binary analysis, we use K=50 and α=0.8. (4) **AdaMerging** has four variants, among which we adopt the most effective *Layer-wise AdaMerging++* in its paper [59]. The method combines K task models with the base model, formulated as $\boldsymbol{\theta}_{\text{m}}^l = \boldsymbol{\theta}_{\text{base}}^l + \sum_{k=1}^K \lambda_k^l \mathbf{T}_k^l$, where λ_k^l is the learnable coefficient for task k at layer l, and $\mathbf{T}_k^l = \boldsymbol{\theta}_k^l - \boldsymbol{\theta}_{\text{base}}^l$ denotes the corresponding task vector. The layer-specific coefficients are learned via entropy minimization on unlabeled samples, where entropy measures the uncertainty of a model's predictive distribution, with higher values indicating less confidence.