RETROFIT: Continual Learning with Bounded Forgetting for Security Applications

Yiling He*†, Junchi Lei*‡, Hongyu She‡,
Shuo Shao‡, Xinran Zheng†, Yiping Liu§, Zhan Qin‡ and Lorenzo Cavallaro†
†*University College London* ‡*Zhejiang University* §*Alibaba Group*
**Co-first authors. Email:* †*heyilinge0@gmail.com*, ‡*junchilei@zju.edu.cn*,
‡*hongyushe@zju.edu.cn*, ‡*shaoshuo_ss@zju.edu.cn*, †*xinran.zheng.23@ucl.ac.uk*,
§*ypinglinkle@gmail.com*, ‡*qinzhan@zju.edu.cn*, †*l.cavallaro@ucl.ac.uk*

***Abstract*—Modern security analytics are increasingly powered by deep learning models, but their performance often degrades as threat landscapes evolve and data representations shift. While continual learning (CL) offers a promising paradigm to maintain model effectiveness, many approaches rely on full retraining or data replay, which are infeasible in data-sensitive environments. Moreover, existing methods remain inadequate for security-critical scenarios, facing two coupled challenges in knowledge transfer: preserving prior knowledge without old data and integrating new knowledge with minimal interference.**

We propose RETROFIT, a *data retrospective-free* CL method that achieves *bounded forgetting* for effective knowledge transfer. Our key idea is to consolidate previously trained and newly fine-tuned models, serving as teachers of old and new knowledge, through parameter-level merging that eliminates the need for historical data. To mitigate interference, we apply low-rank and sparse updates that confine parameter changes to independent subspaces, while a knowledge arbitration adaptively balances the teacher contributions guided by model confidence. 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 security analytics, driving advances in applications such as malware detection [1], [2], vulnerability discovery [3], [4], and binary analysis [5], [6]. However, their effectiveness is constantly challenged by the field's dynamic nature, which creates two primary forms of data shift: *temporal drift* and *representation shift*. The former arises from the ongoing evolution of adversary techniques and attack vectors [7], while the latter stems from the inherent diversity of software ecosystems and data formats [8]. Such variability can destabilize learned decision boundaries, resulting in critical alarm failures or even catastrophic system compromises [9].

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 [10]. In security analytics, this paradigm is already well-recognized for addressing temporal drift in malware detection, where models are periodically updated through sequential retraining [11] or replay-based strategies [12]. 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 [13] 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-handling policies, privacy regulations, and storage limits render dominant strategies such as full retraining or data replay infeasible [14], [15], forcing reliance on data-free adaptation. Yet, existing methods remain inadequate, as the constraint amplifies two coupled challenges to effective model adaptation, both critical for security deployment. First, models struggle to preserve prior knowledge without old data, leading to *catastrophic forgetting* [16]. 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 [17].

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

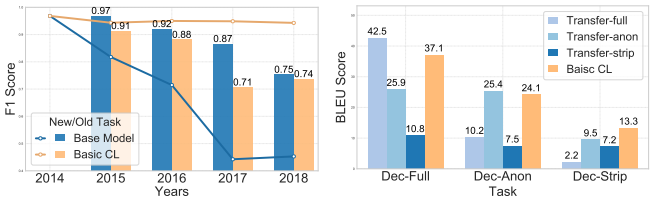
RETROFIT replaces inaccessible historical data with the previous model as a proxy for past knowledge, satisfying the data constraints inherent to security applications. Knowledge accumulation is then formulated as model consolidation, where a newly updated model learns new information and is subsequently integrated with the previous one via parameter-level merging. To realize bounded forgetting in this process, RETROFIT employs two complementary mechanisms: 1) *structurally*, it confines parameter updates to task-relevant subspaces via low-rank decomposition, and 2) *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 in security analytics. For temporal drift, we consider Android malware detection [18], a long-standing benchmark for CL where models must adapt to evolving threats without forgetting earlier ones. For representation shift, we study binary analysis [17], a more recent LLM-driven summarization task in which models are sequentially trained on decompiled code of increasing obfuscation levels to assess cross-representation generalization. The two datasets comprise 259K labeled apps across five years and 215K decompiled function–summary pairs spanning multiple levels.

Across both applications, we demonstrate the clear superiority of RETROFIT over five representative CL baselines [19], [20], [21], [22], [23]. In malware detection, RETROFIT markedly mitigates forgetting while maintaining adaptability—achieving an average retention score improvement of 30.29% and even surpassing the oracle upper bound (full retraining [24]) 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 [25], 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, which addresses both challenges through bounded forgetting without access to historical data. RETROFIT enables data-free knowledge accumulation by consolidating previously trained and newly adapted models through parameter-level merging, where bounded forgetting is enforced through controlled low-rank and sparse model 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 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 effective knowledge accumulation and cross-representation robustness remain 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) [26] 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 [10]: 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 fine-tuned on each new dataset. However, this approach typically suffers from *catastrophic forgetting* [27], 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 based on their core mitigation strategy:

- *Replay-based*: These methods explicitly preserve prior knowledge by storing data from previous tasks and interleaving it with new samples during training. Representative approaches include Experience Replay (ER) [28] and iCaRL [29], 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 parameters important to earlier knowledge. *Weight-constraint* methods often require preserving and leveraging detailed information from the old model's training process. They adopt common heuristics including the Fisher information (EWC [30]), path-integral contri-

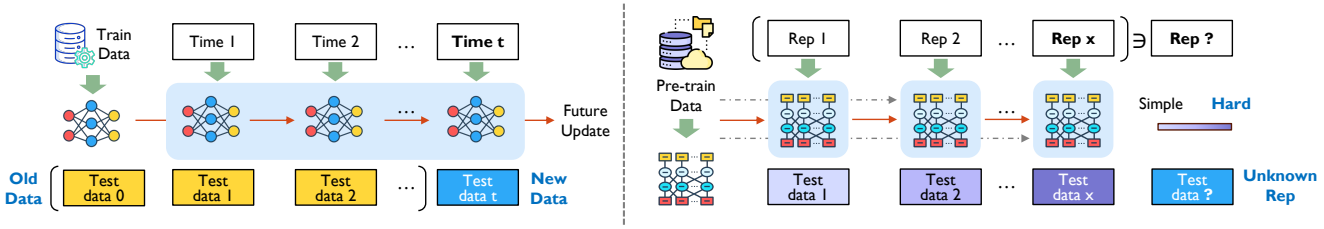


Figure 2. Continual learning for addressing temporal shift (left) and representation shift (right) in security applications.

butions (SI [31]), and output sensitivity (MAS [32]). In contrast, knowledge *distillation-based* methods preserve the functional knowledge of the old model. For example, the classical approach LwF [21] treats the old model's logits as a distillation target, while subsequent works improve by distilling feature-level representations (PODNet [22]) or adding contrastive learning (Co^{2}L [23]).

- *Architecture-based*: These methods mitigate forgetting by modifying the network structure, such as by isolating parameters for different tasks (PackNet [33], HAT [34]) or dynamically expanding the network (DEN [35]). 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 [20], have adapted this paradigm to domain-incremental and introduces lightweight adapters.

While these paradigms have shown effectiveness in controlled benchmarks from computer vision and natural language processing [36], [37], they often rely on idealized assumptions such as stable task boundaries or accessible replay buffers. These conditions are rarely satisfied in dynamic security-critical environments, which we discuss next.

2.2. CL for Security Applications

2.2.1. Why CL is needed. Deep learning has achieved remarkable success in security applications, such as threat detection and binary analysis. However, these systems operate in dynamic environments, where the underlying data and representations continuously evolve [38], [39]. 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 domains or 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 [18], [40]: 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 [17]. 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 continual learning can provide a unified mechanism for sustaining performance across temporal and representational evolution in security-critical environments, which we provide more details in Section 4.

2.2.2. Security-Specific Constraints on CL. 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. Security data are sensitive, proprietary, and often governed by strict sharing and retention policies [41], [42]. Datasets collected by different vendors or research teams cannot be exchanged due to confidentiality or regulatory restrictions, and historical samples are frequently deleted or inaccessible after use. As a result, models must be updated solely on newly available data, which defines the *data retrospective-free* nature of continual adaptation in real security workflows.
- Cross-domain performance requirements. Security models must remain effective as task boundaries are ambiguous at test time [39]. 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

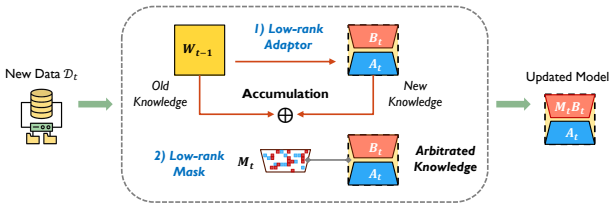


Figure 3. Design insights for accumulating old and new knowledge at each continual learning stage. Knowledge accumulation is achieved through parameter merging between the previous and newly adapted models, while interference control is ensured by applying masked low-rank updates.

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 [30] and MAS [32] still depend on previous task data to compute importance metrics, while SI [31] 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 [11], 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 remains difficult. In code summarization, 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 the need 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 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 [43], 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 [44] 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.

Reduced Interference & Bounded Forgetting. The combination of structural control and arbitration ensures that each merge step introduces only bounded drift on past knowledge. Formally, we define the effective update energy at round t as $E_t := \|M_t \odot B_t\|_F$, which measures the magnitude (via 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

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

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

In the following, we analyze each component of the bound in detail. The first term in Equation 1, which scales with the rank ratio r/d_{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

We first analyze how the structural design of our update mechanism mitigates task conflict during continual learning.

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 $A_t \in \mathbb{R}^{d_{\text{in}} \times r} \sim \mathcal{N}(0, 1/d_{\text{in}})$.

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

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

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

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

// Calibrated true-class confidence

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

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

// Keep and Gain arbitration

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

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

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

// Supervision & mask sparsity

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

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

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

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

The key idea is that task-specific knowledge is not dispersed across the full parameter space, but instead localized into a low-rank subspace governed by the projection matrices B_t. 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: 1) low-rank constraints shrink the capacity for interference by limiting updates to a subspace of dimension $r \ll d_{\text{in}}$, and 2) freezing the random matrix A provides a near-isometric embedding that preserves geometric stability across tasks.

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, only 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), which will be 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. Consequently, 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. For clarity, the main text uses the representative layer weight W, while a full model-level analysis is provided in Appendix A.

Effect of Low Rank. Let $D = d_{\text{in}}$ for brevity. Constraining updates to rank r effectively reduces the dimensionality of the subspace in which interference may occur from D to r. Since $r \ll D$, the expected overlap between new updates and old gradients is proportionally diminished.

Proposition 1 (Low-rank reduces expected interference by r/D)**.** *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 $\Delta W_t = A_t B_t$,*

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

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

Effect of Freezing A. The random initialization of matrix A, specifically with a variance of $1/D$ (Line 3), is critical for normalizing the projection and ensuring isotropy. The scaling factor precisely calibrates the strength of the random matrix so that it approximately preserves distances and inner products during projection. Once frozen (Line 5), A induces a near-isometry from the coefficient space, where the trainable B_t matrices live, into the full model space. This property, as formalized in Proposition 2, preserves the geometric relationships between task solutions.

Proposition 2 (Frozen random A as near-Isometry)**.** *Let $A \sim \mathcal{N}(0, 1/D)$. For any $U, V \in \mathbb{R}^{r \times d_{out}}$,*

$$\langle AU, AV \rangle = \langle U, V \rangle \pm O(\varepsilon)\|U\|_F \|V\|_F, \tag{4}$$

with high probability, where $\varepsilon \to 0$ as $D \gg r \log d_{out}$.

Hence, by applying the proposition with $U = B_i$ and $V = B_j$, it is guaranteed that training and merging only B_t is a valid approach. The cross-task interactions ($\langle B_i, B_j \rangle$) remain stable and directly analyzable within the coefficient space, as they are a high-fidelity proxy for the interactions $\langle AB_i, AB_j \rangle$ in the full model space.

Remarks. By restricting updates and merges exclusively to B_t, the model localizes all task-specific knowledge to a controlled low-rank subspace. The capacity of interference is bounded by the rank ratio r/D, while the frozen random

projection A guarantees geometric stability across tasks. These design choices reduce the likelihood of destructive interference, stabilize cross-task interactions, and prepare the ground for confidence-driven arbitration to further regulate overlap and preserve functional consistency.

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 already restricts interference to the coefficient space $\{B_t\}$, 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 directly correspond to the residual terms in the forgetting bound (Equation 1): 1) a confidence-driven knowledge transfer that preserves predictions in high-confidence regions through distillation-based teacher guidance and controls 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. The objective of this module is to transfer "keep" knowledge from the old model and "gain" knowledge from the new model solely using data from the current domain. At each step, arbitration determines which model should dominate the merged knowledge and proceeds through four components: calibrated confidence estimation, teacher selection, sample reweighting, and distillation to the student.

Arbitration Intuition. For a training or validation sample (x, y) from 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 $p_t = f_{\text{new}}(x)$. Since only data from the current domain are accessible, the old model serves as the sole proxy for past knowledge. Arbitration therefore *prioritizes the old model* to maintain stability. When the old model is confident on the true class, the sample likely lies within regions representing well-learned prior knowledge. Enforcing consistency with p_o in these cases prevents overwriting reliable representations and bounds forgetting. Only when the old model is uncertain does arbitration shift toward the new model p_t to strengthen knowledge acquisition. This confidence-driven asymmetry provides a principled means to balance stability and plasticity without relying on historical data.

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), for instance, $\text{Quantile}(\{p(y_i)\}_{(x_i,y_i)\in\mathcal{D}_t^{\text{valid}}}, \alpha)$, which corresponds to the α-percentile of true-class confidences [45]. A prediction is then regarded as confident if $c > \tau$, which provides a class-calibrated decision criterion. This optional calibration improves consistency across classes while maintaining generality; its implementation is not the focus of this study and one can follow standard practices from prior work [40], [46]. 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) [47] 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:

$$T^\star = \begin{cases} p_o, & c_o \geq \tau_o, \\ p_t, & c_t \geq \tau_t, \\ 0, & \text{otherwise}, \end{cases} \qquad w^\star \equiv 1. \tag{6}$$

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:

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

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. While arbitration governs functional alignment between teachers and the student, sparse mask learning regulates structural alignment at the parameter level. A mask M_t is learned over the low-rank adapter B_t to control parameter overlap across tasks. To achieve selective and interpretable merging, we impose a Sparse-

Group-Lasso regularization that promotes both element-wise and structured sparsity:

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

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 for each task and by isolating task-specific subspaces in the shared parameter space.

Finally, the mask is optimized jointly with arbitration through the composite objective:

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

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). Given that A_t acts as a near-isometry, the interference between two task updates ΔW_s and ΔW_t can be approximated by their coefficient overlap $\langle M_s \odot B_s,\ M_t \odot B_t \rangle$. By minimizing shared support, the learned masks explicitly control this overlap, thereby reducing cross-domain interference and bounding forgetting.

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

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

complementing the earlier forgetting bound, where $C > 0$ is a model-specific constant reflecting sensitivity to drift, and E_t is specifically induced by the conditioned merging in $\mathcal{L}_{\text{merge}}$ and the ground-truth-anchored supervised loss $\mathcal{L}_{\text{sup}}$. This dual perspective establishes a bounded yet adaptive learning dynamic, ensuring stability under confident predictions and guaranteed plasticity on new data.

4. RETROFIT Deployment

We demonstrate the deployment of RETROFIT under two representative CL scenarios in security: temporal drift in malware detection and representation shift in binary analysis. These examples capture realistic challenges in real-world pipelines (Figure 2), and each instantiating a different mode of the confidence-driven knowledge arbitration.

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 and label noise is minimal, 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 Kullback–Leibler 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 representation shifts occur gradually across decompilation levels and the code summaries used for supervision are often noisy (e.g., extracted from comments), the *soft arbitration* mode is adopted to enable confidence-weighted blending and resilient alignment. For the sequence-to-sequence architecture used in code summarization and analysis, it is performed in latent space via a mean squared error: $\|p_s - T^\star\|_2^2$, where p_s, p_o, and p_t denote pooled latent representations of the student, old, and new models, and $T^\star$ is the confidence-weighted latent interpolation.

We use these deployment setups for evaluation in the next section and provide specific 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 [40], 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 Drebin features [48] and employ a multilayer perceptron (MLP) classifier as the backbone model [49].

For binary analysis, we leverage the CAPYBARA [17] 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*, *decompiled-anonymized*, and

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

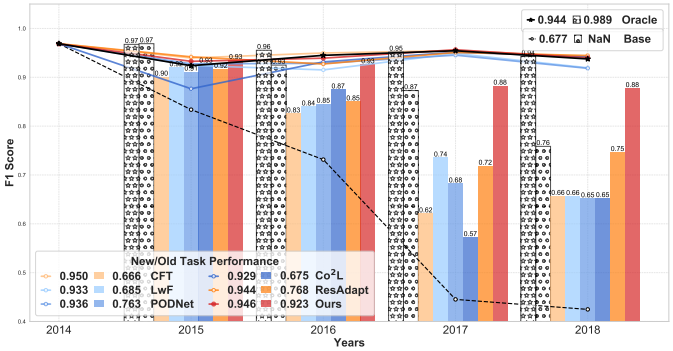


Figure 4. Cumulative CL comparison in malware detection.

TABLE 1. YEAR-WISE CL COMPARISON IN MALWARE DETECTION: F1-SCORE EVALUATED ON OLD AND NEW TEST DATASET.

	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 [21]	0.920	0.924	0.840	0.915	**0.735**	0.949	0.656	0.919
PODNet [22]	0.911	0.927	0.845	0.929	0.682	0.945	0.651	0.918
Co²L [23]	**0.927**	0.877	**0.875**	0.932	0.572	0.951	0.652	0.942
ResAdapt [20]	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**

decompiled-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 accompanying BinT5 model design, which fine-tunes an expert model for each subset via transfer learning, we employ the same CodeT5 [50] backbone to ensure consistency and comparability across abstraction levels.

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** [21]: a logit-distillation method that regularizes the new model's training using the previous model's predictions as soft targets. 3) **PODNet** [22]: 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** [23]: a contrastive learning method that designs an instance-relationship distillation loss that preserves pairwise similarity structures from the previous model. 5) **ResAdapt** [20]: 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.

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.

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. 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* [43] directly averages parameters across models; *Task Arithmetic* [51] transfers task-specific knowledge via linear operations on task vectors; *TIES-Merging* [52] resolves conflicting parameters through sign-based majority rules; and *AdaMerging* [53] learns adaptive layer-wise merge weights. Implementation details are provided in Appendix B.3, and their comparative results across both applications is 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 continual learning baselines. In this scenario, the model is incrementally fine-tuned on samples from each year to emulate real-world temporal evolution. The model must continually adapt to new, evolving threats without forgetting how to identify older, still-relevant malware samples—thus jointly evaluating *adaptability* and *retention* across time.

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 [38], 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: $\mathrm{PTR}_t = \frac{1}{t-1}\sum_{s=1}^{t-1}\frac{a_{t,s}}{a^{\max}_{\leq t,s}}$, where $a_{t,s}$ is the score on the s-th test set after learning up to time t, and $a^{\max}_{\leq t,s}$ denotes the highest score ever achieved on that set. PTR is consistently reported at the final time step to summarize forgetting accumulated over the entire sequence in this paper. 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 4, 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 $1.3 \sim 14.5\%$) thanks to their knowledge-regularization losses, yet this comes at the cost of adaptability (with AUT values decrease by about 1.8% on average). Among these, PODNet performs best overall, maintaining the most balanced trade-off between old and new tasks, likely owing to its control of intermediate feature representations. 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 across years.

Overall, RETROFIT achieves the best balance between retention and adaptability. It improves the PTR by 30.29% 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 full-data-training model on most yearly evaluations and in the cumulative AUT metric. This suggests that bounded forgetting not only mitigates catastrophic forgetting but also enhances stability for long-term generalization, a phenomenon consistent with prior observations that full retraining is not always the optimal upper bound [24].

5.3. Effectiveness on Representation Shift

We next evaluate RETROFIT on representation shift using the binary summarization benchmark and compare it against the CL baselines. In this setting, the model is sequentially fine-tuned on increasingly obfuscated representations, simulating the progressive loss of program information caused by symbol removal and compiler optimizations. This evaluation examines whether the model can accumulate transferable knowledge that sustains performance not only on the final and most security-critical representation (stripped binaries) but also across all intermediate levels.

Evaluation Metrics. We use the BLEU score [54] 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 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 [55]) 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 other semantic similarity metrics in Appendix C for complementarity 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 5, 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, the naive CFT baseline performs best among all CL baselines, whereas 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 sencod-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. Indeed, all other baselines underperform the naive CFT, with XRep scores lower by $0.04 \sim 4.17$ and $\text{BLEU}_{\text{strip}}$ lower by $0.28 \sim 2.16$, indicating their limited ability to generalize knowledge across obfuscated code representations.

In contrast, RETROFIT achieves clear and consistent improvements on both metrics. It raises the BLEU on stripped binaries to 15.05, substantially outperforming prior transfer-learning methods used in existing research (7.20, $+109.0\%$) and even exceeding the oracle joint training model (13.26, $+13.5\%$). At the same time, RETROFIT maintains strong performance across all decompilation levels, achieving an average XRep score of 25.81 and an improvement of 3.9% over the best baseline. These results confirm that bounded forgetting effectively stabilizes representation learning, enabling cumulative knowledge transfer from easier domains to more obfuscated, data-scarce representations without sacrificing overall generalization.

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 [43]	**0.859**	0.857	0.891	0.777	0.891	0.773	25.06	14.17	**10.32**	16.52
Task Arithmetic [51]	0.747	0.833	0.906	**0.912**	0.898	0.853	**32.40**	**17.61**	8.71	**19.57**
Ties-Merging [52]	0.791	0.835	0.897	0.853	0.889	0.875	32.38	16.55	7.84	18.92
AdaMerging [53]	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).

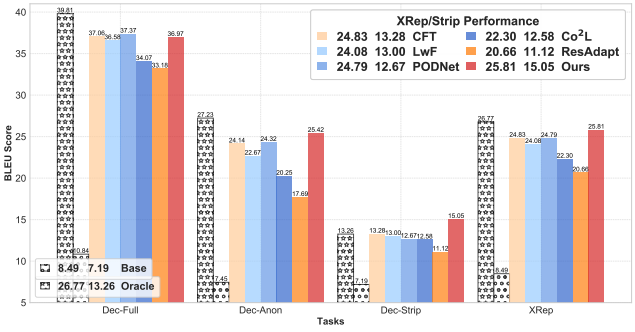


Figure 5. Comparison with existing CL methods in binary analysis.

5.4. Comparison to Model-Preserving 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 is complete. We evaluate them across both applications, where the first involves aggregating malware classifiers trained from different years, and the second aggregates summarization models fine-tuned on distinct representation levels.

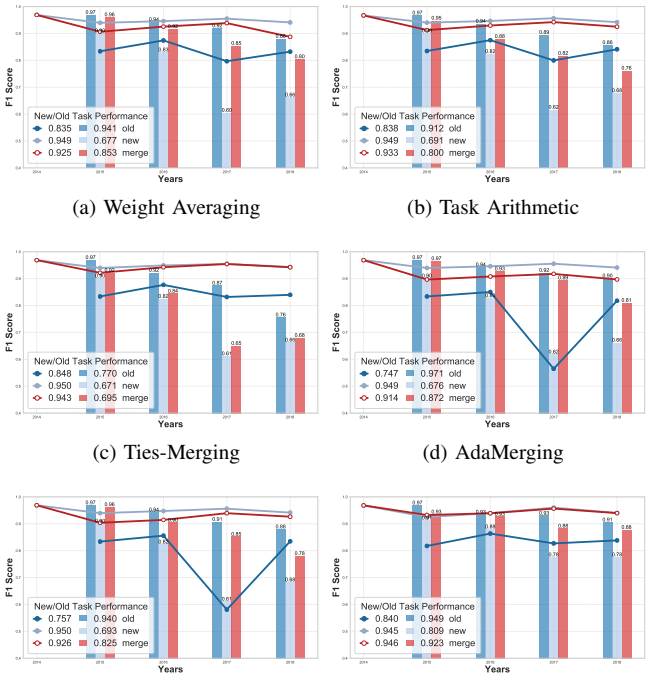
Evaluation Setup. We adopt the previously established metrics for each application. For malware detection, model-preserving methods are evaluated through a single aggregation step at the final year (2018). Each resulting model, including RETROFIT, is then tested on all earlier years' test sets as well 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 consolidated model, which is evaluated across all 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 4 and Figure 5, none of the model-preserving 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 [24]. 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. To further understand the contribution of each component in RETROFIT, we conduct an ablation study on the malware detection task. Its clear temporal structure and well-separated metrics for retention and adaptation provide an interpretable testbed for analyzing our design. The study validates the contributions of our adaptive



(a) Weight Averaging (b) Task Arithmetic

(c) Ties-Merging (d) AdaMerging

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

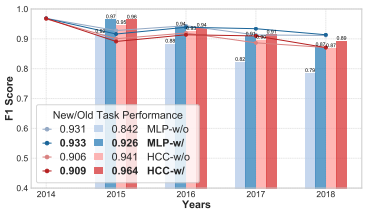
Figure 6. Ablation of our adaptive merging strategy and low-rank constraint.

merging strategy and the low-rank structural constraint, with results presented in Figure 6.

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 and theoretically bounded forgetting.

We further observe that in all these baselines, the newly adapted models (each fine-tuned from the previously merged model) exhibit sharp forgetting after every update, indicating that successive merging erodes prior knowledge and amplifies errors over time. In contrast, RETROFIT slightly improves AUT over the newly adapted model, demonstrating that bounded forgetting also facilitates forward transfer by promoting more domain-invariant representations.



		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	0.897	0.887
	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 7. Improvements in retention and adaptation performance when RETROFIT is integrated into active learning with two classifier architectures.

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.

5.5.2. Combination with Active Leaning. In malware detection, recent research increasingly adopts *active learning* to support model adaptation under limited human-labeling budgets. These approaches typically focus on detecting and labeling new drift samples and then combining them with existing training data for retraining, yet most rely on CFT as the adaptation mechanism. Among them, HCC [11] represents a state-of-the-art framework that designs a classifier with hierarchical contrastive learning to intrinsically detect drift during deployment.

It is important to note that this paradigm operates under assumptions that violate the data constraints in our paper. Both the *cold-start* and *warm-start* adaptation modes described in HCC presume access to previously collected data for retraining once new samples are labeled. Specifically, cold-start retrains a new model from scratch using all accumulated data, whereas warm-start continues from the existing model weights. In particular, the warm-start mode is explicitly shown to yield better performance. To evaluate the compatibility of RETROFIT with such paradigms, we integrate it into HCC's 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

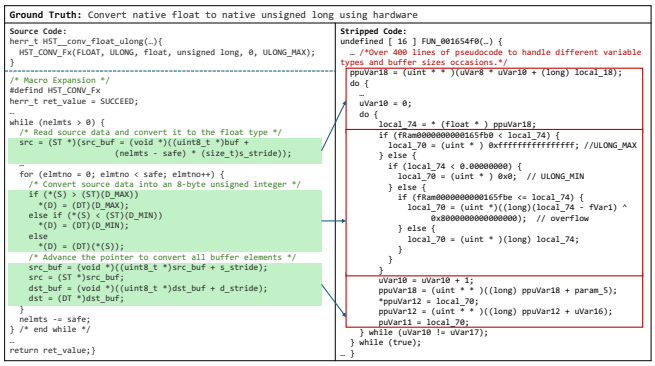


Figure 8. An example of how the function decompilation from a stripped binary significantly expands the source code. RETROFIT achieves an exact match with BLEU of 100, while baselines fail (CFT: 4.92, Oracle: 20.16).

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) and adapting it with RETROFIT for direct comparison.

Result Analysis. As shown in Figure 7, the integration of RETROFIT into active learning consistently improves both classifiers under evaluation. For the MLP, RETROFIT increases AUT by 0.2% and PTR by 10.0%, while for the more advanced HCC classifier, it yields additional gains of 0.3% in AUT and 2.4% in PTR.

Under active learning, 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%). Incorporating RETROFIT restores adaptability and further strengthens retention (PTR +10.0%), confirming that RETROFIT preserves its core advantages even when the data-access assumptions differ from our 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 further enhances HCC's stability and adaptability, raising both metrics simultaneously. A closer year-wise analysis reveals that RETROFIT consistently 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 bounded forgetting ultimately yields superior long-term performance.

5.5.3. Summarization Quality Analysis. While automated metrics like BLEU provide a quantitative benchmark, they often fail to capture the semantic quality or security-specific utility. To overcome this limitation, we conduct a qualitative manual analysis. This analysis has two primary goals: 1) to understand why the automated scores for stripped binaries are universally low, and 2) to demonstrate how RETROFIT provides more interpretable and useful summaries.

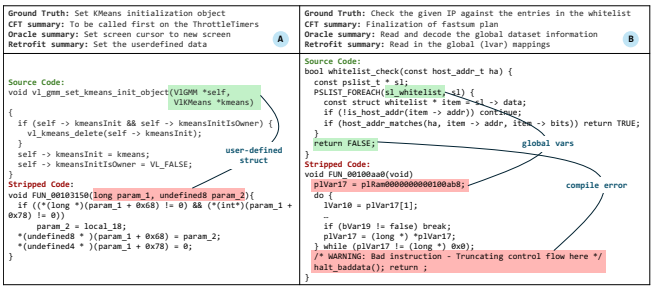


Figure 9. Summary comparison for challenging stripped binaries, where the BLEU of all methods is lower than 20.

Exact Match on Complex Logic. We provide an example in Figure 8 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 9 presents two challenging cases where all methods (including RETROFIT) achieve BLEU scores below 20. We identify two factors that explain the universally low scores:

- *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 are missing after stripping.
- *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 again fabricates irrelevant logic, seemingly driven by superficial lexical cues, whereas Oracle overgeneralizes, mistaking conditional whitelist checking for dataset processing.

6. Discussion

Applicable Domain. As demonstrated in Section 5.5.2, RETROFIT is compatible with different model architectures for malware detection and can likewise be extended to emerging pretrained feature extractors like DexBERT [56]. In binary analysis, similar arbitration mechanisms can be applied to other LLMs [57]. 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 [39] and vulnerability discovery [4]. More broadly, RETROFIT offers a general solution for privacy-sensitive applications where models must adapt across domains without access to historical data, e.g., in medical informatics where continual updates to patient data must comply with HIPAA regulatory.

Computational Cost. RETROFIT is designed for continual adaptation with minimal computational overhead. It requires no data buffer, maintains a constant model size, and preserves a single forward pass identical to that of the base model for inference. The main cost arises from training the task-specific low-rank matrices B_t and sparse masks M_t, both of which scale linearly with the adapter rank r. During arbitration, computing confidence scores and merging loss involves only forward passes through the two teacher models, adding negligible runtime overhead. Compared with the simplest CL baseline CFT, 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 $r/\min(d_{\text{in}}, d_{\text{out}})$.

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 [12] demonstrates that distribution-aware replay enhances malware classification by accounting for heterogeneity both across and within families. Designing and incorporating similar replay mechanisms across applications into RETROFIT could strengthen retention under moderate data-availability conditions, at the cost of additional storage and training overhead.

7. Related Work

Representation Learning for Robustness. Representation learning methods such as contrastive learning [11], [58] and invariant risk minimization [59], [60] aim to extract domain-invariant features that remain stable under distributional shifts. For example, hierarchical contrastive learning employed in HCC [11] 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 [61] and structure-aware encoders that integrate control-flow information [62]. While sharing the goal of preserving performance amid changing conditions, they operate in static settings; CL instead adapts representations dynamically as new data arrive, making the two approaches naturally complementary.

MTL and Model Merging. Multi-task learning (MTL) [63] 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 [64], [65]. To mitigate task interference, TIES-Merging [52] applies a majority vote on the sign for each parameter, whereas AdaMerging [53] 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 Leaning. 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 [40], [66]. In security analytics, such mechanisms have been widely used to detect emerging malware families or new attack behaviors [67], [68]. 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 [11], [61]. As demonstrated in Section 5.5.2, our method serves as an adaptation extension that seamlessly integrates into this active learning workflow when data-access constraints are relaxed.

Code Analysis for Security. Recent advances in LLMs have opened new directions for automated code understanding in security applications [69]. A key research question is how well these models comprehend low-readability code [70], [71], especially that found in obfuscated binaries. This exploration is often hampered by the scarcity of ground-truth data, which forces a reliance on proxy metrics for tasks like malware analysis [72]. Our work applies RETROFIT to an LLM-based binary summarization task, using a recent dataset compiled from open-source (non-malware) projects that provides ground-truth summaries from source code comments [17]. While this benchmark is a valuable tool, we also note its remaining flaws (as discussed in Section 5.5.3), highlighting the need for further dataset refinement to study CL in broader code analysis applications.

8. Conclusion

This paper tackles continual learning in data-sensitive 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 bounded forgetting. Empirical results

across two representative security 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.

Our open-source code will be available at https://github.com/XdlSecZJU/RETROFIT-CL and models be available at https://huggingface.co/RETROFIT-CL.

References

[1] X. Qian, X. Zheng, Y. He, S. Yang, and L. Cavallaro, "Lamd: Context-driven android malware detection and classification with llms," in *2025 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2025, pp. 126–136.

[2] Y. He, Y. Liu, L. Wu, Z. Yang, K. Ren, and Z. Qin, "Msdroid: Identifying malicious snippets for android malware detection," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 3, pp. 2025–2039, 2023.

[3] J. He, L. Beurer-Kellner, and M. Vechev, "On distribution shift in learning-based bug detectors," in *International conference on machine learning*. PMLR, 2022, pp. 8559–8580.

[4] P. Liu, J. Liu, L. Fu, K. Lu, Y. Xia, X. Zhang, W. Chen, H. Weng, S. Ji, and W. Wang, "Exploring {ChatGPT's} capabilities on vulnerability management," in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 811–828.

[5] M. Lau, I. SECK, A. P. Meliopoulos, W. Lee, and E. Ndiaye, "Revisiting non-separable binary classification and its applications in anomaly detection," *Transactions on Machine Learning Research*, 2024. [Online]. Available: https://openreview.net/forum?id=zOJ846BXhl

[6] Z. Su, X. Xu, Z. Huang, K. Zhang, and X. Zhang, "Source code foundation models are transferable binary analysis knowledge bases," *Advances in Neural Information Processing Systems*, vol. 37, pp. 112 624–112 655, 2024.

[7] L. Yang, A. Ciptadi, I. Laziuk, A. Ahmadzadeh, and G. Wang, "Bodmas: An open dataset for learning based temporal analysis of pe malware," in *2021 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2021, pp. 78–84.

[8] Y. Cao, R. Zhang, R. Liang, and K. Chen, "Evaluating the effectiveness of decompilers," in *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2024, pp. 491–502.

[9] IBM, "2025 threat intelligence index," IBM Institute for Business Value, Tech. Rep., Oct 2025. [Online]. Available: https://www.ibm.com/thought-leadership/institute-business-value/en-us/report/2025-threat-intelligence-index

[10] L. Wang, X. Zhang, H. Su, and J. Zhu, "A comprehensive survey of continual learning: Theory, method and application," *IEEE transactions on pattern analysis and machine intelligence*, vol. 46, no. 8, pp. 5362–5383, 2024.

[11] Y. Chen, Z. Ding, and D. Wagner, "Continuous learning for android malware detection," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 1127–1144.

[12] M. S. Rahman, S. Coull, Q. Yu, and M. 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.

[13] W. Ying, Y. Zhang, J. Huang, and Q. Yang, "Transfer learning via learning to transfer," in *International conference on machine learning*. PMLR, 2018, pp. 5085–5094.

[14] S. Lai, X. Yuan, A. Sakzad, M. Salehi, J. K. Liu, and D. Liu, "Enabling efficient privacy-assured outlier detection over encrypted incremental data sets," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 2651–2662, 2019.

[15] M. Raynal and C. Troncoso, "On the conflict between robustness and learning in collaborative machine learning," in *2025 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2025, pp. 2171–2189.

[16] T. L. Hayes, K. Kafle, R. Shrestha, M. Acharya, and C. Kanan, "Remind your neural network to prevent catastrophic forgetting," in *European conference on computer vision*. Springer, 2020, pp. 466–483.

[17] A. Al-Kaswan, T. Ahmed, M. Izadi, A. A. Sawant, P. Devanbu, and A. 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)*. IEEE, 2023, pp. 260–271.

[18] R. Jordaney, K. Sharad, S. K. Dash, Z. Wang, D. Papini, I. Nouretdinov, and L. Cavallaro, "Transcend: Detecting concept drift in malware classification models," in *26th USENIX security symposium (USENIX security 17)*, 2017, pp. 625–642.

[19] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*. Elsevier, 1989, vol. 24, pp. 109–165.

[20] S.-A. Rebuffi, H. Bilen, and A. Vedaldi, "Learning multiple visual domains with residual adapters," *Advances in neural information processing systems*, vol. 30, 2017.

[21] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.

[22] A. Douillard, M. Cord, C. Ollion, T. Robert, and E. Valle, "Podnet: Pooled outputs distillation for small-tasks incremental learning," in *European Conference on Computer Vision*. Springer, 2020, pp. 86–102.

[23] H. Cha, J. Lee, and J. Shin, "Co2l: Contrastive continual learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 9516–9525.

[24] Z. Wu, H. Tran, H. Pirsiavash, and S. 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)*. IEEE, 2023, pp. 1–5.

[25] E. Yang, L. Shen, G. Guo, X. Wang, X. Cao, J. Zhang, and D. Tao, "Model merging in llms, mllms, and beyond: Methods, theories, applications and opportunities," *arXiv preprint arXiv:2408.07666*, 2024.

[26] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, "A continual learning survey: Defying forgetting in classification tasks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 7, pp. 3366–3385, 2021.

[27] Z. Guo, A. Kumar, and R. Tourani, "Persistent backdoor attacks in continual learning," in *34th USENIX Security Symposium (USENIX Security 25)*, 2025, pp. 6379–6397.

[28] D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, and G. Wayne, "Experience replay for continual learning," *Advances in neural information processing systems*, vol. 32, 2019.

[29] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 2001–2010.

[30] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[31] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *International conference on machine learning*. PMLR, 2017, pp. 3987–3995.

[32] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, "Memory aware synapses: Learning what (not) to forget," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 139–154.

[33] A. Mallya and S. Lazebnik, "Packnet: Adding multiple tasks to a single network by iterative pruning," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 7765–7773.

[34] J. Serra, D. Suris, M. Miron, and A. Karatzoglou, "Overcoming catastrophic forgetting with hard attention to the task," in *International conference on machine learning*. PMLR, 2018, pp. 4548–4557.

[35] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, "Lifelong learning with dynamically expandable networks," in *6th International Conference on Learning Representations, ICLR 2018*, 2018.

[36] S. Woo, S. Debnath, R. Hu, X. Chen, Z. Liu, I. S. Kweon, and S. Xie, "Convnext v2: Co-designing and scaling convnets with masked autoencoders," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 16 133–16 142.

[37] M. Tan and Q. Le, "Efficientnetv2: Smaller models and faster training," in *International conference on machine learning*. PMLR, 2021, pp. 10 096–10 106.

[38] F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder, and L. Cavallaro, "Tesseract: Eliminating experimental bias in malware classification across space and time," in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 729–746.

[39] S. H. Ding, B. C. Fung, and P. 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)*. IEEE, 2019, pp. 472–489.

[40] F. Barbero, F. Pendlebury, F. Pierazzi, and L. Cavallaro, "Transcending transcend: Revisiting malware classification in the presence of concept drift," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 805–823.

[41] P. Ren, C. Zuo, X. Liu, W. Diao, Q. Zhao, and S. 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)*. IEEE Computer Society, 2023, pp. 468–480.

[42] S. Shao, Y. Li, H. Yao, Y. He, Z. Qin, and K. Ren, "Explanation as a watermark: Towards harmless and multi-bit model ownership verification via watermarking feature attribution," in *Network and Distributed System Security Symposium (NDSS)*, 2025.

[43] M. Wortsman, G. Ilharco, S. Y. Gadre, R. Roelofs, R. Gontijo-Lopes, A. S. Morcos, H. Namkoong, A. Farhadi, Y. Carmon, S. Kornblith *et al.*, "Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time," in *International conference on machine learning*. PMLR, 2022, pp. 23 965–23 998.

[44] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen *et al.*, "Lora: Low-rank adaptation of large language models." *ICLR*, vol. 1, no. 2, p. 3, 2022.

[45] J. Vaicenavicius, D. Widmann, C. Andersson, F. Lindsten, J. Roll, and T. Schön, "Evaluating model calibration in classification," in *The 22nd international conference on artificial intelligence and statistics*. PMLR, 2019, pp. 3459–3467.

[46] Y. Virk, P. Devanbu, and T. Ahmed, "Calibration of large language models on code summarization," *Proceedings of the ACM on Software Engineering*, vol. 2, no. FSE, pp. 2944–2964, 2025.

[47] T. Kim, J. Oh, N. Y. Kim, S. Cho, and S.-Y. Yun, "Comparing kullback-leibler divergence and mean squared error loss in knowledge distillation," in *30th International Joint Conference on Artificial Intelligence (IJCAI-21)*. IJCAI, 2021, pp. 2628–2635.

[48] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens, "Drebin: Effective and explainable detection of android malware in your pocket." in *Ndss*, vol. 14, no. 1, 2014, pp. 23–26.

[49] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial examples for malware detection," in *European symposium on research in computer security*. Springer, 2017, pp. 62–79.

[50] Y. Wang, W. Wang, S. Joty, and S. C. 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*, 2021, pp. 8696–8708.

[51] G. Ilharco, M. T. Ribeiro, M. Wortsman, S. Gururangan, L. Schmidt, H. Hajishirzi, and A. Farhadi, "Editing models with task arithmetic," *arXiv preprint arXiv:2212.04089*, 2022.

[52] P. Yadav, D. Tam, L. Choshen, C. A. Raffel, and M. Bansal, "Ties-merging: Resolving interference when merging models," *Advances in Neural Information Processing Systems*, vol. 36, pp. 7093–7115, 2023.

[53] E. Yang, Z. Wang, L. Shen, S. Liu, G. Guo, X. Wang, and D. Tao, "Adamerging: Adaptive model merging for multi-task learning," *arXiv preprint arXiv:2310.02575*, 2023.

[54] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[55] S. Banerjee and A. 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*, 2005, pp. 65–72.

[56] T. Sun, K. Allix, K. Kim, X. Zhou, D. Kim, D. Lo, T. F. Bissyandé, and J. Klein, "Dexbert: Effective, task-agnostic and fine-grained representation learning of android bytecode," *IEEE Transactions on Software Engineering*, vol. 49, no. 10, pp. 4691–4706, 2023.

[57] B. Roziere, J. Gehring, F. Gloeckle, S. Sootla, I. Gat, X. E. Tan, Y. Adi, J. Liu, R. Sauvestre, T. Remez *et al.*, "Code llama: Open foundation models for code," *arXiv preprint arXiv:2308.12950*, 2023.

[58] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PmLR, 2020, pp. 1597–1607.

[59] K. Ahuja, K. Shanmugam, K. Varshney, and A. Dhurandhar, "Invariant risk minimization games," in *International Conference on Machine Learning*. PMLR, 2020, pp. 145–155.

[60] X. Zheng, S. Yang, E. C. Ngai, S. Jana, and L. Cavallaro, "Learning temporal invariance in android malware detectors," *arXiv preprint arXiv:2502.05098*, 2025.

[61] X. Zhang, Y. Zhang, M. Zhong, D. Ding, Y. Cao, Y. Zhang, M. Zhang, and M. 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*, 2020, pp. 757–770.

[62] K. Pei, W. Li, Q. Jin, S. Liu, S. Geng, L. Cavallaro, J. Yang, and S. Jana, "Exploiting code symmetries for learning program semantics," in *Proceedings of the 41st International Conference on Machine Learning*, 2024, pp. 40 092–40 113.

[63] Y. Zhang and Q. Yang, "A survey on multi-task learning," *IEEE transactions on knowledge and data engineering*, vol. 34, no. 12, pp. 5586–5609, 2021.

[64] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert, "Cross-stitch networks for multi-task learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3994–4003.

[65] C. Wu, T. Wang, Y. Ge, Z. Lu, R. Zhou, Y. Shan, and P. Luo, "π-tuning: Transferring multimodal foundation models with optimal multi-task interpolation," in *International Conference on Machine Learning*. PMLR, 2023, pp. 37 713–37 727.

[66] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE transactions on knowledge and data engineering*, vol. 31, no. 12, pp. 2346–2363, 2018.

[67] Y. He, J. Lei, Z. Qin, K. Ren, and C. Chen, "Combating concept drift with explanatory detection and adaptation for android malware classification," *arXiv preprint arXiv:2405.04095*, 2024.

[68] L. Yang, W. Guo, Q. Hao, A. Ciptadi, A. Ahmadzadeh, X. Xing, and G. Wang, "{CADE}: Detecting and explaining concept drift samples for security applications," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 2327–2344.

[69] X. Zheng, X. Qian, Y. He, S. Yang, and L. Cavallaro, "Beyond classification: Evaluating llms for fine-grained automatic malware behavior auditing," *arXiv preprint arXiv:2509.14335*, 2025.

[70] C. Fang, N. Miao, S. Srivastav, J. Liu, R. Zhang, R. Fang, Asmita, R. Tsang, N. Nazari, H. Wang, and H. Homayoun, "Large language models for code analysis: Do LLMs really do their job?" in *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association, Aug. 2024, pp. 829–846.

[71] X. Jin, J. Larson, W. Yang, and Z. Lin, "Binary code summarization: Benchmarking chatgpt/gpt-4 and other large language models," *arXiv preprint arXiv:2312.09601*, 2023.

[72] Y. He, H. She, X. Qian, X. Zheng, Z. Chen, Z. Qin, and L. Cavallaro, "On benchmarking code llms for android malware analysis," in *Proceedings of the 34th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2025, pp. 153–160.

Appendix A. Model-level Bound and Interference

While the main text expresses the update rule using a single weight matrix W for clarity, the complete model consists of layer parameters $\theta = \{W^{(\ell)} \in \mathbb{R}^{d_{\text{in}}^{(\ell)} \times d_{\text{out}}^{(\ell)}} \mid \ell = 1, \ldots, L\}$, each following the same masked low-rank adaptation mechanism. We therefore generalize the interference and forgetting analysis to the model level.

Notation. During CL, the layer-wise update at round t is $\Delta W_t^{(\ell)} = A^{(\ell)}\big(M_t^{(\ell)} \odot B_t^{(\ell)}\big)$, where $A^{(\ell)} \in \mathbb{R}^{d_{\text{in}}^{(\ell)} \times r^{(\ell)}}$ is a frozen random projection, $B_t^{(\ell)}$ is a trainable expansion matrix, and $M_t^{(\ell)}$ is a learnable mask regulating parameter merging. The update energy of layer ℓ is $E_t^{(\ell)} := \|M_t^{(\ell)} \odot B_t^{(\ell)}\|_F$, and that of model-level is $E_t := \sum_{\ell=1}^{L} E_t^{(\ell)}$.

Proposition 3 (Model-level expected interference reduced by low-rank)**.** *For sets U, V, define the Frobenius inner product and norm as $\langle U, V\rangle = \sum_{\ell=1}^{L} \langle U^{(\ell)}, V^{(\ell)}\rangle_F$, and $\|U\|^2 = \sum_{\ell=1}^{L} \|U^{(\ell)}\|_F^2$. Let $\Delta\theta_t = \{\Delta W_t^{(\ell)}\}_{\ell=1}^{L}$. Then the expected cosine similarity between them satisfies*

$$\mathbb{E}\left[\frac{\langle g_{old},\, \Delta\theta_t\rangle}{\|g_{old}\|\,\|\Delta\theta_t\|}\right] \;\le\; O\left(\sqrt{\sum_{\ell=1}^{L} \alpha_\ell \,\frac{r^{(\ell)}}{d_{in}^{(\ell)}}}\right), \tag{11}$$

where

$$\alpha_\ell \;=\; \frac{\|\Delta W_t^{(\ell)}\|_F^2}{\sum_{j=1}^{L} \|\Delta W_t^{(j)}\|_F^2}, \qquad \alpha_\ell \ge 0,\; \sum_{\ell} \alpha_\ell = 1 \tag{12}$$

are the normalized layer energy weights. Hence, the expected interference between old-task gradients and new updates decreases with the square root of the rank ratio.

Theorem 1 (Model-level reduced interference & bounded forgetting)**.** *Assume each $A^{(\ell)}$ is an ε-approximate isometry, the learned masks have average pairwise overlap at most ρ, and arbitration enforces $\textsc{Arbitrate}(\cdot) \le \delta$ on high-confidence regions. Then the average prediction drift on past distributions is bounded with*

$$O\left((\kappa + \rho)\sum_{t=1}^{T} E_t + \sqrt{\delta} + \varepsilon\right), \tag{13}$$

where

$$\kappa \;=\; \frac{\sum_{t=1}^{T}\sum_{\ell=1}^{L}\left(\frac{r^{(\ell)}}{d_{in}^{(\ell)}}\right) E_t^{(\ell)}}{\sum_{t=1}^{T}\sum_{\ell=1}^{L} E_t^{(\ell)}} \tag{14}$$

is the effective rank ratio, reflecting structural interference capacity across layers. This bound shows that drift on past predictions remains jointly controlled by the low-rank constraint (r/d_{in}), mask overlap ρ, arbitration tightness δ, and projection error ε.

Appendix 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 the mask is initialized with a sparsity ratio. While the mask is optimized during training, true sparsity is implicitly induced as the mask values evolve. For malware detection, we initialize the mask with a ratio of 0.01, meaning that only the top 1% of entries are assigned the positive value and thus mitigate forgetting. For binary analysis, we set the ratio to 1.0, so that all entries start from the positive value, allowing for a more progressive accumulation of new knowledge. 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 $\mu = 2\times10^{-6}$, 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 $\mathcal{L} = \lambda_{\text{ce}}\,\mathcal{L}_{\text{ce}} + \lambda_{\text{kd}}\,\mathcal{L}_{\text{kd}}$. Here, $\mathcal{L}_{\text{ce}}$ measures prediction accuracy on the new task, while $\mathcal{L}_{\text{kd}}$ enforces consistency between the current (student) model and the previously

trained (teacher) model. 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 instance-wise similarity matrices between student and teacher embeddings.

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 emphasizes inter-class similarities; in our experiments we use $T = 2.0$. For malware detection, we use $\lambda_{\text{ce}} = 1.0$ and $\lambda_{\text{kd}} = 1.0$. For binary analysis, we set $\lambda_{\text{ce}} = 1.0$ and $\lambda_{\text{kd}} = 0.1$, and average the distillation loss 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 to implement PODNet but omit its local similarity classifier designed for class-incremental learning. For malware detection, MLP produces vectorized rather than spatial features. Therefore, we adopt a one-dimensional adaptation, aligning 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 loss weighting follows the same setting as LwF.

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. 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 malware detection, the adapters are injected into intermediate feature layers of the MLP backbone, with the adapter 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.

TABLE 3. BASELINE COMPARISON IN BINARY ANALYSIS (EM)

	Dec-Full	Dec-Anon	Dec-Strip	XRep
Oracle	28.01	17.57	4.31	16.63
CFT	23.34	**14.53**	4.49	**14.12**
LwF [21]	21.24	12.01	**4.87**	12.71
PODNet [22]	**23.70**	14.49	3.75	13.98
Co^{2}L [23]	19.26	10.18	3.56	11.00
ResAdapt [20]	16.22	7.15	2.25	8.54
Ensemble Voting	10.12	2.43	1.12	4.56
Ensemble Averaging	8.81	1.55	0.37	3.58
Weight Averaging [43]	5.98	2.37	0.56	2.97
Ties-Merging [52]	16.39	6.66	0.75	7.93
Task Arithmetic [51]	14.85	6.17	0.75	7.25
AdaMerging [53]	5.96	2.18	0.19	2.77
RETROFIT (Ours)	**23.09**	**15.77**	**5.62**	**14.82**

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 [53]. 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 layer-wise 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 entropy indicating less confident outputs.

Appendix C. Complementary Summarization Metric

Besides BLEU, we report three complementary metrics that are also used in the CAPYBARA benchmark [17]: *Exact Match (EM)*, *METEOR*, and *ROUGE-L*. As shown in Table 3, Table 4, and Table 5, RETROFIT consistently

TABLE 4. BASELINE COMPARISON IN BINARY ANALYSIS (METEOR).

	Dec-Full	Dec-Anon	Dec-Strip	XRep
Oracle	0.432	0.295	0.162	0.296
CFT	0.412	0.262	0.157	0.277
LwF [21]	0.414	0.252	0.157	0.274
PODNet [22]	**0.416**	**0.264**	0.154	**0.278**
Co^{2}L [23]	0.387	0.225	**0.158**	0.257
ResAdapt [20]	0.385	0.202	0.135	0.241
Ensemble Voting	0.264	0.126	0.101	0.164
Ensemble Averaging	0.333	0.176	0.089	0.200
Weight Averaging [43]	0.320	0.169	0.130	0.206
Ties-Merging [52]	0.395	0.207	0.112	0.238
Task Arithmetic [51]	0.390	0.211	0.124	0.241
AdaMerging [53]	0.322	0.171	0.071	0.188
RETROFIT (Ours)	**0.413**	**0.277**	**0.182**	**0.290**

TABLE 5. BASELINE COMPARISON IN BINARY ANALYSIS (ROUGE-L).

	Dec-Full	Dec-Anon	Dec-Strip	XRep
Oracle	0.477	0.331	0.184	0.331
CFT	0.462	0.297	**0.181**	0.313
LwF [21]	**0.468**	0.287	0.175	0.310
PODNet [22]	0.465	**0.300**	0.176	**0.314**
Co^{2}L [23]	0.440	0.262	0.179	0.294
ResAdapt [20]	0.452	0.241	0.152	0.281
Ensemble Voting	0.317	0.161	0.125	0.201
Ensemble Averaging	0.419	0.242	0.128	0.263
Weight Averaging [43]	0.406	0.226	0.160	0.264
Ties-Merging [52]	0.435	0.231	0.123	0.263
Task Arithmetic [51]	0.451	0.251	0.139	0.280
Adamerging [53]	0.405	0.230	0.109	0.248
RETROFIT (Ours)	**0.463**	**0.311**	**0.198**	**0.324**

outperforms all categories of baselines across both the most security-critical stripped binaries and the overall cross-representation averages. On stripped binaries, RETROFIT achieves substantial gains over the best baseline, improving EM by 15.4%, METEOR by 15.2%, and ROUGE-L by 9.4%. On the XRep metric, the improvements remain consistent at 4.9%, 4.3%, and 3.2% for EM, METEOR, and ROUGE-L, respectively. These results align closely with the BLEU-based findings reported in Section 5.3, confirming RETROFIT's robustness across both cumulative and security-critical performance dimensions. The best-performing baseline varies slightly across metrics, while RETROFIT remains consistently superior overall. For example, the best for stripped binary is LwF for EM, Co^{2}L for METEOR, and CFT for ROUGE-L (and BLEU in the main text). We further observe that CL baselines still tend to outperform aggregation-based ones, reinforcing the benefits of sequential knowledge integration. At the initial abstraction level (*decompiled-full*), RETROFIT remains only marginally behind the best baseline (1.1% in BLEU), with differences of merely 2.5%, 0.7%, and 1.1% in EM, METEOR, and ROUGE-L, respectively.