applied sciences

MDPI

Article

Geometry-Aware Risk-Guided Active Learning for Malware Concept Drift Adaptation

Kailun Wang, Kailun Cao, Yujie Liu, Yanshuai Yin, Jing Shang, Jiqiang Liu * and Jian Wang

Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University, Beijing 100044, China; wangkailun@bjtu.edu.cn (K.W.); 22221119@bjtu.edu.cn (K.C.); 23281173@bjtu.edu.cn (Y.L.); yinyanshuai@bjtu.edu.cn (Y.Y.); shangjing@bjtu.edu.cn (J.S.); wangjian@bjtu.edu.cn (J.W.)

* Correspondence: jqliu@bjtu.edu.cn

Abstract

Mobile malware evolution induces concept drift, degrading the performance of static supervised detectors. Concept Drift Adaptation with Active Learning (CDA-AL) mitigates this problem by querying informative samples for model updating, but existing methods still face two limitations. First, uncertainty-based selection measures prediction ambiguity without explicitly assessing consistency with previously learned representation structure, causing confidently misclassified drifted samples to be overlooked. Second, conventional retraining often treats labeled samples uniformly, providing insufficient emphasis on missed malware and samples near ambiguous benign-malicious boundaries. To address these limitations, we propose GRAD (Geometry-Aware Risk-Guided Active Drift Adaptation), a CDA-AL framework with three complementary modules. Geometry-Aware Drift-Risk Ranking uses residual deviations from locally estimated benign tangent spaces to prioritize potential misclassification risks. Hybrid active sample selection combines geometry-ranked risk candidates with uncertainty sampling under a fixed annotation budget. Boundary-Aware Retraining constructs Error Memory and Semantic Ambiguity Memory and applies targeted triplet constraints to correct missed malware and improve class separation. Experiments on two large-scale temporal malware datasets totaling over 460K samples show that GRAD achieves the best F1 performance on both datasets, improving F1 by 3.26% on APIGraph and 7.23% on AndroZoo, while reducing FNR by 39.15% and FPR by 72.50%, respectively. GRAD also shows no evidence of systematic catastrophic forgetting and reduces the number of long-lived malware samples by 12.5% on APIGraph.

Keywords: malware detection; concept drift adaptation; active learning

Check for updates

Received: 4 August 2026
Revised: 26 September 2026
Accepted: 28 September 2026
Published: 10 October 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Mobile security is challenged by the mismatch between the rapid evolution of malware and slower updates of detection models [1]. According to GDATA [2], a new malware sample emerges in mobile app markets roughly every 11 s. With billions of active mobile devices worldwide [3], even short detection delays can expose a large user base to significant risk [4]. These trends underscore the urgent need for adaptive detection strategies that can adapt to emerging threats in a timely manner [5,6]. Supervised machine learning is widely employed in security-critical applications such as malware analysis [7–9], where accurate and timely classification is critical for mitigating threats. However, as the distribution discrepancy between training and deployment data widens over time [10], model performance on concept-drifted malware degrades [11]. This temporal distribution shift is

commonly referred to as concept drift [6,12,13], and it undermines the reliability of static supervised detectors in deployment.

To address this critical issue, numerous Concept Drift Adaptation (CDA) methods have been developed [14–16]. Among these, Concept Drift Adaptation with Active Learning (CDA-AL) [17–19] stands out as a highly effective solution. In CDA-AL pipelines, the model actively queries human annotations for highly informative samples, enabling effective drift detection and adaptation [17,20]. These selected samples are manually annotated and added to the training set for iterative retraining. Consequently, CDA-AL improves model generalization against evolving data distributions [12,17].

Despite their empirical successes, existing CDA-AL frameworks still suffer from two fundamental limitations tailored to malware drift scenarios. **Challenge 1: Insufficient Identification of Drift-Risk Samples.** Existing drift adaptation methods commonly identify informative samples primarily according to model-output uncertainty. However, predictive confidence alone does not explicitly characterize whether a sample's prediction is consistent with the local representation geometry learned from previously observed data. Under temporal distribution shift, an evolving malicious sample may remain confidently predicted as benign, while a newly changed benign application may be incorrectly predicted as malicious. Consequently, security-critical false-negative and false-positive risk samples may not receive sufficient priority during active sample selection. **Challenge 2: Insufficient Emphasis on Difficult Samples during Drift Adaptation.** After informative samples are selected and labeled, existing adaptation methods typically incorporate them into model retraining without explicitly distinguishing which samples require greater attention. Consequently, missed malware samples may not receive sufficient corrective emphasis, while samples that are difficult to distinguish between benign and malicious classes may remain poorly separated. These difficult cases are particularly important under concept drift, as unresolved classification errors and ambiguous local boundaries can persist across successive adaptation cycles.

To address the above limitations, we propose **GRAD** (**Geometry-Aware Risk-Guided Active Drift Adaptation**), a closed-loop CDA-AL framework for evolving Android malware detection. GRAD integrates Geometry-Aware Drift-Risk Ranking, Hybrid Active Sample Selection, and Boundary-Aware Retraining within each adaptation cycle. Specifically, we redesign three key stages of the conventional CDA-AL pipeline, namely drift-risk sample identification, queried-set construction, and adaptive model retraining. In the inference stage, GRAD evaluates each incoming sample relative to the local geometry of previously observed benign samples. Based on the current predicted label, Module 1 applies two complementary ranking channels to prioritize potential false-negative and false-positive risk candidates according to their residual deviations from the estimated local benign tangent space **(Module 1, Section 4.1, addressing Challenge 1)**. In the sample selection stage, GRAD combines geometry-ranked risk candidates with pseudo-loss-based uncertainty selection. A predefined portion of the annotation budget is allocated to samples exhibiting prediction–geometry inconsistencies, while the remaining budget is used to select model-uncertain samples. This hybrid strategy constructs a retraining set that jointly covers security-critical risk candidates and informative boundary samples **(Module 2, Section 4.2, addressing Challenge 1)**. In the retraining stage, GRAD identifies false-negative hard samples from the cumulative labeled training set and ambiguous-hard samples from the newly labeled queried set. It then dynamically constructs error memory and semantic ambiguity memory to impose targeted triplet constraints during model updating. These constraints repair false-negative-prone regions and improve class separation in locally mixed benign–malicious regions **(Module 3, Section 4.3, addressing Challenge 2)**.

To validate our method's effectiveness, we conduct experiments on two large-scale real-world malware concept drift datasets [21,22]: APIGraph (300K+ samples, 7 years) and AndroZoo (100K+ samples, 3 years). We benchmark against 17 baselines including UNCER [23], CORESET [24], CADE [25], TRANS [26], HCL [17], MADCAT [27], and MALF-SCIL [28]. GRAD achieves the highest overall F1 performance on both datasets, improving F1 by 3.26% on APIGraph and 7.23% on AndroZoo compared with the second-best overall method. It further reduces the false-negative rate by 39.15% on APIGraph and the false-positive rate by 72.50% on AndroZoo. Furthermore, it exhibits robust performance across varying label budgets and neighborhood sizes, and maintains stable performance without evidence of systematic catastrophic forgetting.

The novelty of GRAD does not lie in the isolated use of active learning, local geometric modeling, or metric learning, but in their risk-oriented coupling for temporal malware drift adaptation. Specifically, GRAD forms a closed-loop process in which local geometry is used to identify security-relevant prediction risks, active learning allocates a limited annotation budget to these risks together with uncertain samples, and the resulting labeled samples are subsequently used to construct targeted metric-learning constraints for repairing the corresponding high-risk representation regions. In summary, our main contributions are as follows:

- We introduce a geometry-aware drift-risk ranking mechanism that decomposes each incoming sample's displacement relative to an estimated local benign tangent space. By ranking benign-predicted samples with large residual deviations and malicious-predicted samples with small residual deviations, the proposed mechanism prioritizes potential false-negative and false-positive risk candidates for human verification.
- We develop a hybrid active sample selection strategy that combines geometry-ranked risk candidates with uncertainty selection. This strategy allocates the limited annotation budget to both drift-risk and informative samples, thereby improving risk coverage and adaptation effectiveness.
- We propose a boundary-aware retraining mechanism that explicitly prioritizes difficult labeled samples through dynamically reconstructed error memory and semantic ambiguity memory. The resulting triplet constraints explicitly repair false-negative-prone regions and improve class separation in locally mixed benign–malicious regions during successive adaptation cycles.
- We evaluate GRAD against 17 baselines on two large-scale temporal malware datasets spanning a combined ten-year period. The experimental results demonstrate improved malware drift adaptation performance, reduced false-negative and false-positive errors, and stable long-term adaptation without observable catastrophic forgetting.

2. Related Work

This section reviews the related literature from two key perspectives: malware concept drift and drift detection (Section 2.1), and concept drift adaptation for malware detection (Section 2.2).

2.1. Malware Concept Drift and Drift Detection

Previous studies have emphasized the necessity of evaluating malware detection systems on unseen data, as these systems are prone to concept drift due to evolving vulnerabilities [29,30], often resulting in performance degradation. Chen et al. [31] argued that drift detection should focus on identifying drifted samples, as frequent feature updates provide minimal benefits for drift adaptation. Jordaney et al. [32] proposed a framework for early drift detection via statistical comparison of training and drifted samples. Yang et al. [25] integrated contrastive learning with distance-based uncertainty for effective drift detection,

yet overlooked malware intra-family similarities. Barbero et al. [26] proposed a conformal prediction-based method for drift detection.

2.2. Concept Drift Adaptation for Malware Detection

Early online learning methods for malware drift adaptation incurred substantial costs because they relied heavily on manual labeling [33,34]. To reduce this annotation burden, subsequent studies adopted classifier-driven pseudo-labeling strategies [35,36]. However, erroneous pseudo-labels inevitably introduce label noise, which may accumulate during adaptation and further degrade model performance, as demonstrated in previous studies [37]. This critical flaw is also shared by two representative methods: Md Tanvirul Alam et al. [38] and Zhen Liu et al. [39], mainly because they suffer from self-poisoning caused by incorrect pseudo-labels and persistent label noise. Eunjin Roh et al. [27] enhance shift resilience via feature masking and test-time fine-tuning but lack explicit drift detection, causing redundant updates. Chen et al. [17] reduce the annotation burden through contrastive active learning, but their retraining mechanism does not explicitly prioritize recurring false-negative errors or locally ambiguous samples during successive adaptation. DREAM [40] relies on manual behavior taxonomies. Reinforcement learning-based frameworks [41] lack representation-space constraints, leading to unstable learning. Pan et al. [42] use an offline drift detection setting unsuitable for streaming data and ignore labeling cost constraints.

2.3. Summary

As summarized in Table 1, existing malware concept drift methods typically address only a subset of drift detection, drift adaptation, drift-risk sample handling, and representation-space optimization. In particular, few prior approaches jointly consider drift-risk identification and structured representation optimization during adaptation. GRAD integrates all four capabilities within a unified framework, enabling coordinated drift detection, adaptation, risk-aware sample handling, and representation refinement.

Table 1. Comparison of representative studies on malware concept drift adaptation.

Category	Method	DD	DA	DRS	RSO
Drift-oriented	TRANSCEND [32]	✓	✗	✗	✗
	CADE [25]	✓	✗	✗	✗
	TRANS [26]	✓	✗	✗	✗
	OWAD [43]	✓	✗	✓	✗
	MORPH [38]	✓	✓	✗	✗
	LDCDroid [39]	✓	✓	✗	✗
	MADCAT [27]	✓	✓	✗	✗
	HCL [17]	✓	✓	✗	✓
	DREAM [40]	✓	✓	✗	✓
	DRMD [41]	✓	✓	✓	✗
	DTRACE [42]	✓	✓	✗	✗
Ours	GRAD	✓	✓	✓	✓

Notes. DD: drift detection; DA: drift adaptation; DRS: explicit handling of drift-risk samples; RSO: contrastive or structured representation-space optimization. A check mark indicates that the corresponding capability is supported, whereas a cross indicates that it is not.

3. Background

This section first defines the key notations used throughout the paper and then introduces the standard active learning pipeline for concept drift adaptation.

3.1. Notation and Description

For clarity, we summarize the main notations used throughout this paper in Abbreviations.

3.2. Active Learning Pipeline

Active learning is a learning paradigm that aims to improve model performance under limited labeling budgets by selectively querying the most informative unlabeled samples for manual annotation. Instead of labeling all incoming data, active learning prioritizes samples according to a predefined acquisition criterion, such as predictive uncertainty, thereby reducing annotation cost while retaining informative supervision. Concept drift refers to a shift in the data distribution over time that can degrade model accuracy. To counter this, CDA-AL adapts models to concept drift by actively selecting high-uncertainty samples from the testing data and retraining with them [10,17,44]. The entire concept drift process is composed of multiple concept drift cycles. Each cycle n processes a new data batch $\mathcal{D}_{te}^{n}$ (unlabeled testing data collected since cycle $n-1$) through four steps:

Step I (**Model Inference**): The model owner collects raw data $\mathcal{D}_{te}^{n}$ and then extracts the feature representation of each sample. For each sample, the detector $f_{\theta_{n-1}}$ performs inference on the input $\mathbf{x}_i$ and outputs the predictive probability vector $\boldsymbol{\pi}_i = f_{\theta_{n-1}}(\mathbf{x}_i)$. Each dimension of $\boldsymbol{\pi}_i$ corresponds to the model's predicted posterior probability that $\mathbf{x}_i$ belongs to the corresponding class. The label with the highest confidence is selected as the predicted label $\hat{y}_i$ for the input $\mathbf{x}_i$.

Step II (Uncertainty Quantification): The uncertainty score u_i of $\mathbf{x}_i$ is measured. For example, CDA-AL adopts the standard max-probability uncertainty metric [17], defined as the complement of the maximum softmax output: $u_i = 1 - \max(\boldsymbol{\pi}_i)$.

Step III (Sample Selection): Samples are ranked according to the adopted selection criterion, and the top β samples are selected for manual annotation. Their indices form the queried set $\mathcal{I}_{\text{query}}^{n}$, where β denotes the fixed annotation budget per adaptation cycle.

Step IV (Manual Annotation and Model Retraining): Following the standard active learning setting, the queried samples are assumed to be manually annotated by human analysts before the subsequent model update. The annotation does not need to be completed instantaneously, but should be available within the current adaptation cycle. The samples indexed by $\mathcal{I}_{\text{query}}^{n}$ are manually annotated to obtain the labeled queried set $\mathcal{D}_{\text{query}}^{n}$. The newly labeled samples are then added to the cumulative training set:

$$\mathcal{D}_{tr}^{n} = \mathcal{D}_{tr}^{n-1} \cup \mathcal{D}_{\text{query}}^{n}.$$

The model is subsequently retrained on the updated labeled training pool $\mathcal{D}_{tr}^{n}$, initialized from the previous cycle's parameters θ_{n-1}. The model update is defined as

$$\theta_n = \arg\min_{\theta} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_{tr}^{n}} \mathcal{L}(f_\theta(\mathbf{x}_i), y_i), \qquad \text{init. } \theta = \theta_{n-1}. \tag{1}$$

4. Method

Consistent with the standard CDA-AL paradigm [17], our framework performs concept drift adaptation via a closed-loop updating cycle within each time window. We take the n-th drift cycle as an illustrative example, where the same pipeline generalizes to all subsequent cycles. Our overall workflow (Figure 1) consists of three core modules: geometry-aware drift-risk ranking (Section 4.1), hybrid active sample selection (Section 4.2), and boundary-aware retraining (Section 4.3). GRAD is not designed as a simple sequential combination of active learning, local geometry, and metric learning. Its novelty lies in coupling these components through a common risk-oriented adaptation objective. Specifically, the geometry-aware module identifies security-relevant prediction risks by measuring local

inconsistency with historically observed benign structure; the hybrid selection module allocates the limited annotation budget jointly to geometry-flagged risk samples and uncertain samples; and the boundary-aware retraining module uses the resulting labeled risk samples to construct targeted metric-learning constraints for correcting false-negative-prone and locally ambiguous regions. In this way, the output of each stage directly guides the next, forming a closed-loop process from risk identification to selective annotation and risk-oriented representation correction.

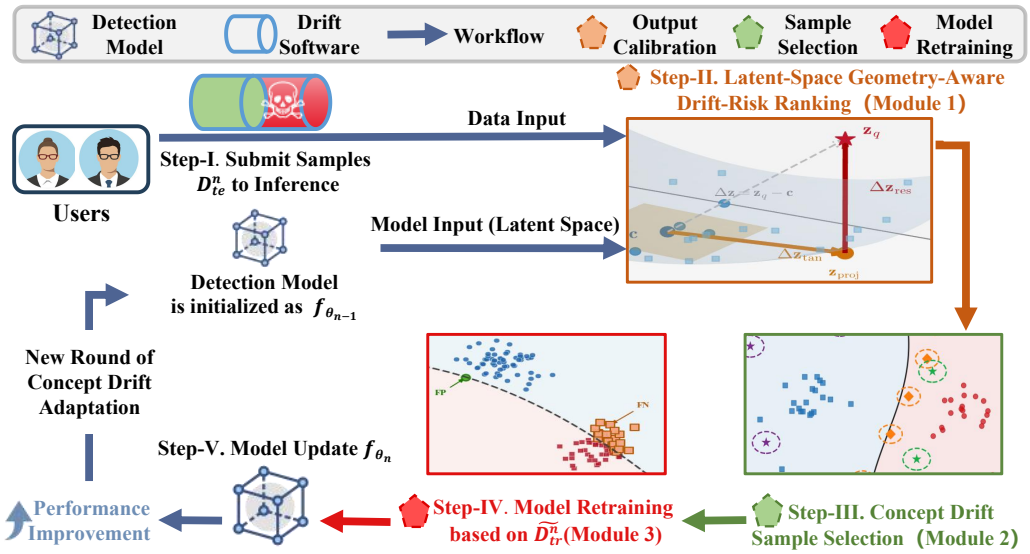


Figure 1. The workflow of our method (GRAD).

4.1. Latent-Space Geometry-Aware Drift-Risk Ranking

At the n-th learning cycle, the incoming unlabeled batch $\mathcal{D}_{te}^{n}$ may contain samples affected by evolving malware families, changes in benign applications, or other forms of temporal distribution shift. To prioritize the samples that are most valuable for model adaptation, we introduce a geometry-aware drift-risk ranking mechanism.

The mechanism evaluates each query representation relative to the local geometry of previously observed benign samples and applies geometric criteria to rank potential false-negative and false-positive risk candidates for human annotation. We use historical benign samples as the geometric reference because they represent the local structure that the detector has previously learned as benign, allowing newly arriving samples to be assessed according to whether their representations remain consistent with this known benign structure. For a query sample $x_q \in \mathcal{D}_{te}^{n}$, its embedding $\mathbf{z}_q = g_{\theta_{n-1}}(x_q)$ is computed using the trained encoder from cycle $n-1$. Our design is based on an empirically motivated local-geometry assumption in the learned representation space. As malware families and benign applications evolve over time, their representations may shift relative to historically observed benign samples, potentially altering local neighborhood structure or producing deviations from the previously learned benign geometry. We do not assume that all forms of temporal concept drift necessarily exhibit such geometric deviations. Instead, we treat local inconsistency with the historical benign distribution as a potentially informative signal for prioritizing samples that may require human verification. We use these studies only as supporting motivation for the informativeness of local geometric characteristics under distributional changes, rather than as evidence that adversarial perturbations and naturally occurring temporal malware drift follow the same mechanism. We therefore characterize each query by decomposing its displacement relative to an estimated local benign geometry into a tangent component and an orthogonal residual component. The resulting geometric scores are used as relative ranking signals. Accordingly, the geometric deviation is used

only as a relative risk-ranking signal for annotation prioritization, rather than as a definitive indicator of concept drift. Let $\mathcal{D}_{\text{ben}}^{n-1} = \{(\mathbf{x}_i, y_i) \in \mathcal{D}_{tr}^{n-1} \mid y_i = 0\}$ denote the benign subset of the labeled training set from cycle $n-1$. For each query embedding $\mathbf{z}_q$, we retrieve its k nearest benign neighbors from $\mathcal{D}_{\text{ben}}^{n-1}$ and fit a local linear approximation of the benign manifold. Specifically, we estimate the local center as the mean of the retrieved benign neighbors: $\mathbf{c}_q = \frac{1}{k}\sum_{i=1}^{k} \mathbf{z}_i$, where $\{\mathbf{z}_i\}_{i=1}^{k}$ are the k nearest retrieved benign neighbors of $\mathbf{z}_q$. We then form the centered neighborhood matrix:

$$\mathbf{X}_q = \left[\mathbf{z}_1 - \mathbf{c}_q, \ldots, \mathbf{z}_k - \mathbf{c}_q\right]^\top \in \mathbb{R}^{k \times d}. \tag{2}$$

We apply PCA to $\mathbf{X}_q$ and use its top-r orthonormal principal directions as the columns of $\mathbf{B}_q$, such that $\mathbf{B}_q^\top \mathbf{B}_q = \mathbf{I}_r$. Using this local model, we measure how $\mathbf{z}_q$ aligns with or deviates from benign local structure through its displacement $\Delta\mathbf{z}_q = \mathbf{z}_q - \mathbf{c}_q$. We decompose $\Delta\mathbf{z}_q$ into tangent and residual parts,

$$\Delta\mathbf{z}_q^{\tan} = \mathbf{B}_q \mathbf{B}_q^\top \Delta\mathbf{z}_q, \qquad \Delta\mathbf{z}_q^{\text{res}} = \Delta\mathbf{z}_q - \Delta\mathbf{z}_q^{\tan}. \tag{3}$$

By construction, $\Delta\mathbf{z}_q^{\tan}$ captures variation aligned with the estimated local benign tangent space, whereas $\Delta\mathbf{z}_q^{\text{res}}$ measures deviation orthogonal to this local tangent space. We summarize the two components using $d_{\tan}(q) = \|\Delta\mathbf{z}_q^{\tan}\|_2$ and $d_{\text{res}}(q) = \|\Delta\mathbf{z}_q^{\text{res}}\|_2$. The tangent score characterizes displacement along the dominant directions of local benign variation, whereas the residual score quantifies deviation orthogonal to the estimated local benign tangent space. We use $d_{\text{res}}(q)$ as the primary ranking measure: large values indicate weak consistency with the local benign geometry, while small values indicate stronger consistency with the estimated local benign tangent space. We construct two complementary drift-risk ranking channels and combine their selected candidates into the geometry-ranked risk set $\mathcal{I}_{\text{cal}}^n = \mathcal{I}_{\text{FP}}^n \cup \mathcal{I}_{\text{FN}}^n$.

(1) False-Negative Risk Channel. For samples predicted as benign, i.e., $\hat{y}_q = 0$, a large residual score indicates substantial deviation orthogonal to the estimated local benign tangent space. This prediction-geometry inconsistency may arise when a previously unseen or drifted malicious sample is incorrectly classified as benign, although it may also be caused by an evolving benign application. Because false negatives incur a high security cost, we use $d_{\text{res}}(q)$ as a geometric ranking signal and prioritize benign-predicted samples with large residual scores for human verification. Specifically, we rank the benign-predicted samples in descending order of $d_{\text{res}}(q)$ and select up to β_{FN} samples as potential false-negative risk candidates:

$$\mathcal{I}_{\text{FN}}^n = \text{Top}_{\min\{\beta_{\text{FN}}, |\{q:\hat{y}_q=0\}|\}}\Big(\{q : \hat{y}_q = 0\}, d_{\text{res}}(q), \text{descending}\Big).$$

(2) False-Positive Risk Channel. For samples predicted as malicious, i.e., $\hat{y}_q = 1$, a small residual score indicates limited deviation orthogonal to the estimated local benign tangent space. This geometric consistency conflicts with the model's malicious prediction and may indicate an evolving benign application that has been incorrectly classified as malicious. However, because a small residual score does not by itself establish the true class of a sample, we use it only as a relative ranking signal for potential false-positive risk. Specifically, we rank malicious-predicted samples by $d_{\text{res}}(q)$ in ascending order and select up to β_{FP} samples as potential false-positive risk candidates:

$$\mathcal{I}_{\text{FP}}^n = \text{Top}_{\min\{\beta_{\text{FP}}, |\{q:\hat{y}_q=1\}|\}}\Big(\{q : \hat{y}_q = 1\}, d_{\text{res}}(q), \text{ascending}\Big).$$

We directly evaluate whether these geometry-based ranking signals concentrate actual false-negative and false-positive errors in Appendix D.1.

4.2. Hybrid Sample Selection via Geometry-Aware Drift-Risk Ranking and Uncertainty

The geometry-aware drift-risk ranking module prioritizes samples whose model predictions are inconsistent with their local benign geometric characteristics. For benign-predicted samples, the module prioritizes large deviations from the estimated local benign tangent space. For malicious-predicted samples, it prioritizes small residual deviations, which indicate limited orthogonal deviation from the estimated local benign tangent space. However, these prediction–geometry inconsistencies may provide limited coverage of moderately ambiguous samples near the current decision boundary. We therefore complement the geometric ranking with an uncertainty-driven querying strategy applied to the unlabeled pool $\mathcal{D}_{te}^{n}$. Given the model parameters θ_{n-1} trained in the $(n-1)$-th learning cycle, we first assign a pseudo-label to each unlabeled candidate sample $x_q \in \mathcal{D}_{te}^{n}$: $\hat{y}_q = \arg\max_{y\in\{0,1\}} p_{\theta_{n-1}}(y \mid x_q)$. Standard uncertainty metrics (e.g., prediction entropy) only rely on model output probabilities and ignore local data structure. To address this limitation, we construct a local contrastive batch centered at each query sample x_q: $\mathcal{B}_q = \{x_q\} \cup \mathcal{N}_{2N_{\text{nbr}}-1}(x_q; \mathcal{D}_{tr}^{n-1})$, where $\mathcal{N}_{2N_{\text{nbr}}-1}(\cdot)$ denotes the set of $2N_{\text{nbr}}-1$ nearest neighbors retrieved exclusively from the labeled training pool $\mathcal{D}_{tr}^{n-1}$, ensuring that all retrieved neighbors have verified ground-truth labels. Let $d_{qj} = \|g_{\theta_{n-1}}(x_q) - g_{\theta_{n-1}}(x_j)\|_2$ denote the L2 embedding distance between x_q and its neighbor $x_j \in \mathcal{B}_q$, and let m be the margin hyperparameter in the embedding space. Following the hierarchical contrastive pseudo-loss formulation in [17], we partition the retrieved neighbors in $\mathcal{B}_q$ into two disjoint sets according to whether their verified ground-truth labels agree with the query pseudo-label $\hat{y}_q$:

$$\hat{\mathcal{P}}(q, \hat{y}_q) = \{j \in \mathcal{B}_q \setminus \{q\} : y_j = \hat{y}_q\}, \qquad \hat{\mathcal{N}}(q, \hat{y}_q) = \{j \in \mathcal{B}_q \setminus \{q\} : y_j \neq \hat{y}_q\}, \tag{4}$$

where $\hat{\mathcal{P}}$ denotes the pseudo-positive neighbors whose ground-truth labels agree with $\hat{y}_q$, and $\hat{\mathcal{N}}$ denotes the pseudo-negative neighbors whose ground-truth labels disagree with $\hat{y}_q$. We then define the local contrastive pseudo-loss term:

$$\hat{L}_{hc}(q) = \frac{1}{|\hat{\mathcal{P}}(q, \hat{y}_q)|} \sum_{j \in \hat{\mathcal{P}}(q, \hat{y}_q)} \max(0, d_{qj} - m) + \frac{1}{|\hat{\mathcal{N}}(q, \hat{y}_q)|} \sum_{j \in \hat{\mathcal{N}}(q, \hat{y}_q)} \max(0, 2m - d_{qj}). \tag{5}$$

This loss penalizes samples that are far from neighbors sharing their pseudo-label and close to neighbors with the opposite label, directly quantifying the local geometric inconsistency of the sample. If $|\hat{\mathcal{P}}(q, \hat{y}_q)| = 0$, the first term is omitted; if $|\hat{\mathcal{N}}(q, \hat{y}_q)| = 0$, the second term is omitted. Building on this geometric uncertainty measure, we further define the standard pseudo cross-entropy term to capture model prediction uncertainty:

$$\hat{L}_{ce}(q) = -\hat{y}_q \log p_{\theta_{n-1}}(1 \mid x_q) - (1 - \hat{y}_q) \log\left(1 - p_{\theta_{n-1}}(1 \mid x_q)\right). \tag{6}$$

We combine these two complementary uncertainty signals into the final pseudo-loss uncertainty score: $s_{\text{ps}}(q) = \hat{L}_{hc}(q) + \lambda_{\text{ce}} \hat{L}_{ce}(q)$. Following the setting adopted in prior work [17], we set $\lambda_{\text{ce}} = 0.5$ in all experiments and keep it fixed across all learning cycles. A larger $s_{\text{ps}}(q)$ indicates higher geometric inconsistency, higher predictive uncertainty, or a combination of both, and hence a higher priority for human annotation. We sort all samples in $\mathcal{D}_{te}^{n}$ in descending order of $s_{\text{ps}}(q)$ to obtain the ranked uncertainty list: $(\mathcal{I}_{\text{ps}}^{n}, \mathcal{S}_{\text{ps}}^{n}) = \text{Rank}_{\text{ps}}(\mathcal{D}_{te}^{n}; \theta_{n-1})$, where $\mathcal{I}_{\text{ps}}^{n}$ denotes the ranked index set of samples, and $\mathcal{S}_{\text{ps}}^{n}$ denotes their corresponding uncertainty scores. Let β denote the total annotation budget per learning cycle. We assume that each incoming unlabeled batch contains at least β samples, i.e., $|\mathcal{D}_{te}^{n}| \geq \beta$, ensuring that the remaining unlabeled pool is sufficient to fill the annotation budget after geometry-aware selection. Recall from Section 4.1 that β_{FN}

and β_{FP} are the pre-allocated budgets for false-negative and false-positive risk candidates selected by the geometry-aware ranking module, satisfying $\beta_{\mathrm{FN}} + \beta_{\mathrm{FP}} < \beta$. Let $\mathcal{I}^n_{\mathrm{cal}}$ denote the index set of candidates selected by the geometry-aware drift-risk ranking module. If either candidate pool contains fewer samples than its pre-allocated budget, all available candidates in that pool are selected. The unused portion of the geometry-ranking budget is automatically reassigned to the pseudo-loss ranking stage through the remaining-budget term $\beta - |\mathcal{I}^n_{\mathrm{cal}}|$.

We include all samples in $\mathcal{I}^n_{\mathrm{cal}}$ deterministically and allocate the remaining budget to the unselected samples with the highest pseudo-loss uncertainty scores:

$$\mathcal{I}^n_{\mathrm{query}} = \mathcal{I}^n_{\mathrm{cal}} \cup \mathrm{TopK}_{q \in \mathcal{I}^n_{\mathrm{ps}} \setminus \mathcal{I}^n_{\mathrm{cal}}} \left(s_{\mathrm{ps}}(q),\, \beta - |\mathcal{I}^n_{\mathrm{cal}}| \right). \tag{7}$$

The resulting query policy combines complementary geometric and uncertainty signals. The geometry-aware drift-risk ranking module allocates part of the annotation budget to geometry-flagged high-risk candidates, while pseudo-loss ranking uses the remaining budget to select locally inconsistent and model-uncertain samples.

4.3. Drift Malware Adaptation with Boundary-Aware Contrastive Loss

After manually annotating the queried indices $\mathcal{I}^n_{\mathrm{query}}$, we obtain the labeled queried set $\mathcal{D}^n_{\mathrm{query}}$. We then update the labeled training pool as follows: $\mathcal{D}^n_{tr} = \mathcal{D}^{n-1}_{tr} \cup \mathcal{D}^n_{\mathrm{query}}$, where $\mathcal{D}^n_{\mathrm{query}}$ contains both geometry-ranked risk candidates and uncertainty-selected informative samples. To adapt the model to emerging drift patterns while preserving previously learned knowledge, we warm-start training from the previous cycle's parameters θ_{n-1} and explicitly prioritize learning from difficult and recurring risk samples. Our retraining process follows a three-stage pipeline that connects risk detection with loss optimization:

- **Risk Scoring.** We compute the error indicator e_i and the neighborhood conflict rate c_i to identify false-negative hard samples and ambiguous-hard samples, respectively.
- **Memory Construction.** Rather than directly optimizing the identified hard samples, we organize them into two complementary triplet memories, namely $\mathcal{M}_{\mathrm{error}}$ and $\mathcal{M}_{\mathrm{sem}}$, which encode geometric constraints for local robustness and improve class separation in ambiguous boundary regions.
- **Joint Optimization.** The total loss combines the classification loss $\mathcal{L}_{\mathrm{cls}}$ on the current labeled data with two memory-based geometric regularization terms, $\mathcal{L}_{\mathrm{error}}$ and $\mathcal{L}_{\mathrm{sem}}$, which operate on the constructed memories.

In essence, the risk scores determine which samples should be prioritized and how the memories should be constructed, whereas the joint objective determines how the model parameters are updated under these structured constraints. The details of each stage are presented below.

Sample Risk Score Evaluation. To identify hard samples for memory construction, we evaluate two complementary difficulty signals using the pre-retraining model $g_{\theta_{n-1}}$. The error indicator is used to identify samples that are currently misclassified as false negatives from training data $\mathcal{D}^n_{tr} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{|\mathcal{D}^n_{tr}|}$, whereas the neighborhood conflict rate is evaluated for the newly labeled samples in $\mathcal{D}^n_{\mathrm{query}}$ to identify ambiguous samples in the current learning cycle. Let $\hat{y}_i$ denote the model's prediction for sample $\mathbf{x}_i$ using the pre-retraining parameters θ_{n-1}. We define an error indicator $e_i = \mathbf{1}[\hat{y}_i \neq y_i]$, which flags all misclassified samples. Among these errors, only false negatives satisfying $y_i = 1$ and $\hat{y}_i = 0$ are retained for constructing the Error Memory, because missed malicious samples represent the most security-critical errors in Android malware detection. We further define a neighborhood conflict rate: $c_i = \frac{1}{k} \sum_{j \in \mathcal{N}_k(i; \mathcal{D}^n_{tr} \setminus \{\mathbf{x}_i\})} \mathbf{1}[y_j \neq y_i]$, where $\mathcal{N}_k(i; \mathcal{D}^n_{tr} \setminus \{\mathbf{x}_i\})$ denotes the k nearest labeled neighbors of $\mathbf{x}_i$, excluding $\mathbf{x}_i$ itself, under the fixed encoder $g_{\theta_{n-1}}$. A large c_i

indicates that the sample lies in a locally mixed region with a complex, ambiguous decision boundary. These scores directly guide sample categorization. Specifically, based on e_i and c_i, we classify samples into two complementary groups that will drive memory construction: **(1) False-negative hard samples**: Misclassified malicious samples satisfying $y_i = 1$ and $\hat{y}_i = 0$, i.e., malicious samples that the current model incorrectly predicts as benign. **(2) Ambiguous-hard samples**: At each learning cycle, we compute the neighborhood conflict scores for the correctly classified samples ($e_i = 0$) in the newly labeled set $\mathcal{D}^n_{\text{query}}$, using the cumulative labeled training set $\mathcal{D}^n_{tr}$ as the neighborhood reference pool. The top-B samples with the largest conflict scores are selected as the ambiguous-hard samples of the current cycle. These two groups capture security-critical false-negative failures and structurally challenging boundary cases that are likely to cause future errors under drift.

Memory Construction from Risk-Scored Samples. We leverage the risk scores e_i and c_i computed to construct two auxiliary triplet memories that impose targeted geometric regularization. Specifically, we construct the two memories from different data sources. The first memory, denoted as the error memory $\mathcal{M}_{\text{error}}$, is built exclusively from false-negative hard samples. It stores triplets $(a_i^{\text{err}}, p_i^{\text{err}}, n_i^{\text{err}})$ where a_i^{err} is a labeled malicious anchor that is currently misclassified as benign by $g_{\theta_{n-1}}$, p_i^{err} is a nearby malicious positive sample, and n_i^{err} is a nearby benign negative sample. This memory specifically targets local regions that are vulnerable to false negatives, which are the most costly errors in Android malware detection. The second memory, denoted as the Semantic Ambiguity Memory $\mathcal{M}_{\text{sem}}$, is constructed from the ambiguous-hard samples identified in the current learning cycle. These semantic anchors are not accumulated across cycles. Instead, they are reselected at each cycle by recomputing the neighborhood conflict scores using the current pre-retraining model. It stores triplets $(a_i^{\text{sem}}, p_i^{\text{sem}}, n_i^{\text{sem}})$, where anchors are selected from locally mixed regions containing both benign and malicious representations. The two memories are complementary: the error memory corrects local false-negative misclassifications, while the semantic ambiguity memory improves class separation in high-conflict boundary regions.

For each anchor in either memory, we select its nearest same-class sample from $\mathcal{D}^n_{tr} \setminus \{a_i\}$ as the positive and its nearest opposite-class sample as the hard negative. All distances used for triplet construction are computed using the fixed pre-retraining encoder $g_{\theta_{n-1}}$. The anchor sets and their resulting triplets remain fixed during the optimization of cycle n and are discarded after retraining. At cycle $n+1$, the updated model g_{θ_n} is used to reevaluate the entire cumulative labeled training set $\mathcal{D}^{n+1}_{tr}$. Samples that are still misclassified as false negatives are reselected as Error Memory anchors by re-evaluating the cumulative labeled training set, from which new Error triplets are constructed. In contrast, Semantic Memory anchors are selected only from the newly labeled set $\mathcal{D}^{n+1}_{\text{query}}$ by recomputing their neighborhood conflict scores using g_{θ_n}. Therefore, neither the Error nor the Semantic anchors are directly carried over from the previous cycle; historical information contributes through the repeated reevaluation of samples retained in the cumulative labeled training set.

Joint Optimization over Classifier and Memory Constraints. Starting from the initialization $\theta \leftarrow \theta_{n-1}$, we optimize the following joint training objective:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{cls}} + \mu\mathcal{L}_{\text{error}} + \lambda\mathcal{L}_{\text{sem}}, \tag{8}$$

where $\mathcal{L}_{\text{cls}}$ is the standard binary cross-entropy classification loss, and $\mathcal{L}_{\text{error}}$ and $\mathcal{L}_{\text{sem}}$ are triplet losses derived from the two auxiliary memories. The hyperparameters μ

and λ control the strength of the geometric regularization terms. For each triplet $(a_i^{\text{err}}, p_i^{\text{err}}, n_i^{\text{err}}) \in \mathcal{M}_{\text{error}}$, we define the per-triplet loss:

$$\ell_{\text{error}}^{(i)} = \left[m_{\text{error}} + \|g_\theta(a_i^{\text{err}}) - g_\theta(p_i^{\text{err}})\|_2^2 - \|g_\theta(a_i^{\text{err}}) - g_\theta(n_i^{\text{err}})\|_2^2 \right]_+, \tag{9}$$

where $[\cdot]_+ = \max(0, \cdot)$ denotes the hinge function, and m_{error} is the margin hyperparameter for the Error Memory. We aggregate over all triplets to obtain the total Error Memory loss:

$$\mathcal{L}_{\text{error}} = \frac{1}{|\mathcal{M}_{\text{error}}|} \sum_{i \in \mathcal{M}_{\text{error}}} \ell_{\text{error}}^{(i)}. \tag{10}$$

This loss pulls false-negative-prone anchors toward nearby malicious positives and pushes them away from benign hard negatives, directly repairing regions with poor malicious recall.

Similarly, for each triplet $(a_i^{\text{sem}}, p_i^{\text{sem}}, n_i^{\text{sem}}) \in \mathcal{M}_{\text{sem}}$, we define

$$\ell_{\text{sem}}^{(i)} = \left[m_{\text{sem}} + \|g_\theta(a_i^{\text{sem}}) - g_\theta(p_i^{\text{sem}})\|_2^2 - \|g_\theta(a_i^{\text{sem}}) - g_\theta(n_i^{\text{sem}})\|_2^2 \right]_+, \tag{11}$$

where m_{sem} is the margin hyperparameter for the Semantic Ambiguity Memory. The total Semantic Ambiguity Memory loss is: $\mathcal{L}_{\text{sem}} = \frac{1}{|\mathcal{M}_{\text{sem}}|} \sum_{i \in \mathcal{M}_{\text{sem}}} \ell_{\text{sem}}^{(i)}$. This loss applies the same metric learning principle to semantically mixed regions, improving class separation where benign and malicious representations overlap most heavily. If no valid Error Memory triplet is available in cycle n, we set $\mathcal{L}_{\text{error}} = 0$. Similarly, if the Semantic Ambiguity Memory is empty, we set $\mathcal{L}_{\text{sem}} = 0$. Thus, each auxiliary term is activated only when its corresponding memory contains at least one valid triplet.

Taken together, the two auxiliary losses enhance robustness to recurring false negatives and improve class discriminability in ambiguous boundary regions. The final model parameters for cycle n, denoted θ_n, are obtained by optimizing $\mathcal{L}_{\text{total}}$. This warm-started training strategy enables adaptation to new drift patterns while preserving knowledge accumulated from all previous cycles.

5. Evaluation

We evaluate our drift-adaptive Android malware detection framework to answer three key questions:

- **(RQ1)** How does it compare to existing methods in malware concept drift adaptation (Section 5.2)?
- **(RQ2)** What is the contribution of each component (Section 5.3)?
- **(RQ3)** Does GRAD exhibit catastrophic forgetting during long-term malware concept drift adaptation (Section 5.4)?

We first describe experimental settings, then present comparative results, ablation studies, and forgetting analysis. Details of the model settings (Appendix C.1) and hyperparameter settings (Appendix C.2) are provided in the Appendix C.

5.1. Experimental Settings

5.1.1. Hardware and Software Environment

We conduct all experiments on Ubuntu 22.04 with 256 GB RAM, an Intel Xeon Gold 6330N @ 2.20 GHz, and a single NVIDIA GeForce RTX 4090 (24 GB VRAM). The implementation is based on Python 3.12.3 and PyTorch 2.5.1 with CUDA 12.4 and cuDNN 9.1. We use NumPy 2.1.3, pandas 3.0.2, and scikit-learn 1.8.0 for data processing and evaluation.

5.1.2. Datasets

We evaluate our method on two large-scale, real-world malware drift datasets: APIGraph [21] and AndroZoo [22]. Together they span up to 10 years, covering more than 100 malware families and hundreds of thousands of samples. To our knowledge, these datasets form one of the most comprehensive resources for concept drift analysis in Android malware detection [17], providing a solid basis for comparison with state-of-the-art adaptive methods.

APIGraph [21] is a representative temporal dataset for Android malware detection. It covers app samples spanning seven years, from 2012 to 2018, including 30,804 malware samples and 330,617 benign applications. The benign-to-malicious ratio is approximately 10:1, which reflects the real-world Android ecosystem where benign applications dominate. The malware samples are collected from VirusTotal, VirusShare, and the AMD dataset, while benign applications are sourced from AndroZoo. All samples are chronologically ordered according to their first appearance timestamps recorded in VirusTotal, ensuring temporal consistency for drift evaluation. APIGraph also addresses common spatial and temporal biases in malware datasets by maintaining a relatively balanced sample distribution across the seven-year period. For feature construction, APIGraph follows the DREBIN feature representation, which includes eight categories of static Android features: hardware component requirements, requested permissions, application components, filtered intents, restricted API calls, actually used permissions, suspicious API calls, and network addresses. These features characterize the behavioral and functional properties of Android applications and provide discriminative information for Android malware detection. In our experiments, we use the data from 2012 as the baseline training set, and the chronological data from January 2013 to December 2018 as the concept drift test set.

AndroZoo [22] is used as an extended Android malware dataset to evaluate model generalization under concept drift. It covers app samples from 2019 to 2021, including 10,199 malware samples and 89,853 benign applications. The benign-to-malicious ratio is also around 9:1, consistent with the real-world Android ecosystem. All samples are collected from the AndroZoo platform. Malware samples are selected based on VirusTotal scanning results, where samples with more than 15 antivirus detections are treated as malicious. Benign applications are selected from samples with zero VirusTotal detections. Malware family labels are generated using AVClass2, and samples that cannot be assigned to a specific family are labeled as "unknown." To preserve temporal authenticity, all samples are ordered according to their actual appearance time. Similar to APIGraph, AndroZoo is constructed to support time-aware evaluation and mitigate temporal bias in Android malware detection experiments. In terms of feature representation, AndroZoo follows the same feature pattern as APIGraph. For our experiments, we use the data from 2019 as the baseline training set, and the chronological data from January 2020 to December 2021 as the concept drift test set.

In our experiments, we directly use the released and processed feature representations of both APIGraph and AndroZoo and do not introduce additional APK-level preprocessing or feature extraction. APIGraph follows the original DREBIN-style static feature representation with API semantic clustering [21], while AndroZoo likewise uses its released processed feature representation without feature regeneration from raw APK files [22].

All binary and malware-family labels used in our experiments are directly adopted from the released datasets, and we do not regenerate or relabel the samples. For APIGraph, the original binary labels are determined based on VirusTotal detections, while the malware family labels are normalized using Euphony. For AndroZoo, we directly use the released binary and family labels provided with the dataset. AVClass2 is part of the

original AndroZoo dataset construction process and is neither reconfigured nor rerun in our experiments.

5.1.3. Metric

Throughout the evaluation, we use the binary F1 score with malware treated as the positive class. To provide a comprehensive assessment, we report FPR, FNR, ACC, F1, and F1-AUT, covering point-wise classification performance, temporal stability, and long-lived Android malware detection failures. Note that all these metrics are evaluated on the newly arriving, still unlabeled samples of each adaptation cycle before any annotation query is issued in that cycle. The samples selected for annotation are labeled and used for retraining only after this evaluation, and therefore cannot influence the model used to evaluate their own adaptation cycle. They may, however, be incorporated into the training set and affect the model evaluated in subsequent cycles. Thus, our evaluation follows a test-then-train, or prequential, protocol: each batch of newly arriving samples is first used to assess the current adapted model, after which selected samples may be annotated and used for model adaptation. Accordingly, every reported FPR, FNR, ACC, F1, and F1-AUT value reflects the model's performance on samples that were unseen by the model at the time they were evaluated, thereby measuring its generalization to newly emerging malware rather than performance on samples already used for retraining.

Malware Forgetting Metric. Let $r \in \{1, \ldots, R\}$ denote the current concept drift cycle, and let θ_r denote the model parameters after completing adaptation in cycle r. We use i to index a preceding cycle before r. For each cycle i, let $\mathcal{U}^i$ denote the complete incoming batch and let $\mathcal{L}^i \subset \mathcal{U}^i$ denote the subset selected for annotation and subsequent model updating. We define the fixed hold-out set of cycle i as

$$\mathcal{T}^i = \mathcal{U}^i \setminus \mathcal{L}^i.$$

The ground-truth labels of $\mathcal{T}^i$ are used exclusively for offline evaluation and are never accessed during sample selection or model retraining. At cycle r, we define the data used for forgetting evaluation as the collection of all preceding cycles, namely $\{\mathcal{T}^1, \ldots, \mathcal{T}^{r-1}\}$. For each preceding concept drift cycle i, we use $\mathrm{F1}(\theta_i, \mathcal{T}^i)$ as the reference performance obtained when cycle i was learned, and compare it against the performance of the current model θ_r on the same cycle. For each adaptation cycle $r \geq 2$, we compute

$$\mathrm{BWT}_{\mathrm{F1}}(r) = \frac{1}{r-1} \sum_{i=1}^{r-1} \Big(\mathrm{F1}(\theta_r, \mathcal{T}^i) - \mathrm{F1}(\theta_i, \mathcal{T}^i) \Big).$$

Negative values indicate average degradation on older cycles after learning new ones, reflecting forgetting. Positive values indicate beneficial backward transfer to previously learned concepts.

Malware Drift Evaluation Metrics. For a sample $\mathbf{x}_i$, let T^i_{born} be the evaluation-cycle index of its first appearance and T^i_{det} be the earliest time that it is correctly detected. The per-sample malware lifespan (measured in the evaluation time unit, e.g., rounds or months) is then $\mathrm{MSL}_i = T^i_{\mathrm{det}} - T^i_{\mathrm{born}}$. For a malware sample that is not correctly detected before the end of the evaluation horizon, we set $T^i_{\mathrm{det}} = R + 1$, where R is the final evaluation cycle. Such a sample is therefore treated as surviving beyond the observed evaluation horizon.

We further define F1-AUT as the normalized area under the F1 curve over the entire drift horizon. Let $F1_t$ denote the binary F1 score at evaluation time point t, where malware is treated as the positive class, and let T denote the total number of evaluation points.

Because all evaluation points are equally spaced at monthly intervals, the normalized trapezoidal area is computed as

$$\text{F1-AUT} = \frac{1}{T-1} \sum_{t=1}^{T-1} \frac{F1_t + F1_{t+1}}{2}. \tag{12}$$

Based on these definitions, we introduce LLMS (Long-Lived Malware Samples). LLMS is defined as the number of malware samples whose lifespan exceeds a predefined long-lived threshold. We set this threshold to one sixth of the total number of evaluation cycles:

$$T^{\text{thr}} = \left\lceil \frac{R}{6} \right\rceil,$$

where R denotes the total number of evaluation cycles. Accordingly, LLMS is computed as

$$\text{LLMS} = \sum_{i=1}^{N_{\text{eval}}} \mathbf{1}_{[\text{MSL}_i > T^{\text{thr}}]}, \tag{13}$$

where N_{eval} denotes the number of malware samples considered in the LLMS evaluation.

5.1.4. Baselines for Concept Drift Adaptation Strategies

As baselines, we adopt 17 baselines (UNCER [23], CORESET [24], CADE [25], TRANS [26], HCL [17], MORPH [38], LDCDroid [39], MADCAT [27], MALFSCIL [28], and SIMPLE [45], among others). We organize the comparison baselines into two explicit groups: classical concept drift adaptation baselines and few-shot adaptation baselines. This design allows us to assess not only whether a sampling strategy is effective under standard retraining, but also whether it remains effective when adaptation must be performed under scarce supervision. Specifically, we apply the same five core sampling strategies to both settings, yielding paired baseline groups: UNCERTAINTY/F-UNCER, CORESET/F-CORESET, TRANS/F-TRANS, CADE/F-CADE, and HCL/F-HCL. In the classical concept drift adaptation setting, these methods proactively identify informative samples from newly arriving data and add them to the training set for targeted retraining. In the few-shot setting, the same sampling mechanisms are embedded into a few-shot adaptation pipeline, so that the comparison is controlled at the strategy level rather than confounded by different query principles. For each monthly adaptation task, all compared methods are assigned the same annotation budget β. The corresponding sampling strategy selects β samples from the incoming unlabeled batch for manual annotation. These newly labeled samples are then added to the cumulative training set for model adaptation. For the few-shot variants, the same annotation protocol is used. After the newly labeled samples are incorporated into the cumulative training set, few-shot retraining is performed episodically by sampling support and query subsets from the labeled training pool. Therefore, the support and episodic query sets used by the prototype-based classifier do not consume additional annotation budget and are distinct from the samples queried for manual labeling during active adaptation.

To ensure a fair comparison across the heterogeneous baselines, all methods are evaluated using the same temporal data splits and input feature representations. For methods compatible with the common classifier backbone, we use the same backbone architecture, while method-specific architectures or representation mechanisms that are intrinsic to a baseline are preserved according to the original design. All active adaptation methods are subject to the same per-cycle annotation budget and query samples from the same newly arriving unlabeled batch. Moreover, all methods are provided with the same level of temporal information: they start from the same initial training period and can use only

labels of samples that have been queried in the current or previous adaptation cycles. Labels of unqueried samples and information from future evaluation periods are never available during training or adaptation.

It is worth noting that UNCER, CADE, and TRANS were originally introduced primarily as drift-detection or uncertainty-estimation methods rather than complete drift-adaptation frameworks. Specifically, UNCER identifies potentially drifted samples based on model confidence, CADE measures the Euclidean distance between sample embeddings and known-class prototypes, and TRANS detects distributional inconsistency based on conformal prediction. In our experiments, we preserve these original detection or scoring mechanisms without modification. To enable a fair comparison under a common adaptation protocol, the samples identified by these methods are manually labeled under the same annotation budget and then incorporated into the cumulative training set for model retraining. Therefore, the retraining-based adaptation step is an experimental extension rather than a component of the original methods. This treatment follows the evaluation protocol adopted by HCL, which similarly integrates detection-oriented strategies into a retraining-based adaptation pipeline for comparative evaluation. Next, we first outline the five core sampling strategies, followed by several additional representative adaptive baselines used in our experiments:

- **Uncertainty [23]:** This strategy adopts the complement of the model's prediction confidence, $1 - \max_y P(y \mid x)$, as the uncertainty metric and prioritizes the model's most uncertain samples for manual labeling. Its core logic is intuitive and its implementation is straightforward, making it a foundational basis for various subsequent complex strategies.
- **CORESET [24]:** This method formalizes batch sample selection as a core-set problem and aims to select a fixed number of k samples in the feature embedding space to minimize the maximum distance from any unlabeled sample to its nearest selected sample: $\min_{S \subset U, |S| \leq k} \max_{\mathbf{x}_i \in U} \min_{x_j \in S} d(\mathbf{x}_i, x_j)$, where U denotes the unlabeled set and $d(\cdot, \cdot)$ is the distance metric. By iteratively selecting the point farthest from the currently selected set, this method ensures the chosen batch maximally covers the overall data distribution, thereby enhancing sample diversity and representativeness.
- **TRANS [26]:** This method identifies concept drift by checking whether a model's prediction for a sample is consistent with historical experience. Its core mechanism is to maintain a calibration set and compute a credibility score for each incoming sample: $p_{value} = \frac{|\{j \in I_{cal} : a_j \geq a_i\}|}{|I_{cal}|+1}$, where a_i is the sample's non-conformity score and I_{cal} denotes the calibration set. A lower p_{value} means the current prediction pattern is less consistent with historical observations, indicating a higher probability of concept drift.
- **CADE [25]:** This method uses supervised contrastive learning to create a semantic embedding space, which brings malware samples from the same family closer together. Its uncertainty metric is defined as the minimum normalized Euclidean distance between the test sample's embedding z and the prototype embeddings $Proto_c$ of all known families: $OOD - Score = \min_{c \in \mathcal{C}} \|z - Proto_c\|_2$. A larger distance means the sample is more likely to be out-of-distribution. During the active learning loop, samples with high OOD scores are prioritized for labeling, aiming to capture emerging malware threats.
- **HCL [17]:** This method improves model robustness by building a hierarchical embedding space. Specifically, the space separates malicious and benign samples at the macro level, and within malicious samples, it clusters those from the same family closely together while maintaining weak similarity across different families. To detect concept-drift samples, HCL uses pseudo-loss as its uncertainty metric. A higher pseudo-loss means greater uncertainty, so the sample is prioritized for labeling. Over-

all, HCL provides a systematic approach to tackle the coexisting challenges of class imbalance and concept drift in Android malware detection.

In addition to the above sampling-based baselines, we further include several representative adaptive frameworks that address malware concept drift from different perspectives:

- **MORPH [38]:** This method performs concept drift adaptation through neural self-training with pseudo-labels. Its core idea is to use the model's own predictions on newly arriving samples as temporary supervision, and then incrementally update the detector to track the evolving data distribution.
- **LDCDroid [39]:** This method combines concept drift detection with pseudo-label-driven adaptation. It first identifies distributional changes through representation learning and clustering-based analysis, and then uses high-confidence pseudo-labeled samples to update the model when drift is detected.
- **MADCAT [27]:** This method improves robustness against distribution shift by learning shift-resilient representations through feature masking during training, and further adapts the detector at test time via self-supervised fine-tuning on unlabeled incoming samples. The key idea is to increase the model's invariance to feature perturbation so that it remains effective under evolving malware distributions.
- **MALFSCIL [28]:** This method introduces a malware-oriented few-shot class-incremental learning paradigm, aiming to maintain adaptation capability when new malware patterns emerge under scarce supervision. Its core idea is to continually absorb newly observed classes or concepts while preserving previously learned knowledge, so that the model can adapt to evolving malware distributions without full retraining from scratch.
- **SIMPLE [45]:** This method is proposed for few-shot malware family classification and unknown-family recognition using supervised multi-prototype modeling. In our experiments, we use SIMPLE as a prototype-based fast-adaptation baseline and adapt its original prototype construction and inference procedure to the temporally ordered concept-drift setting. It updates the detector with newly available samples through a relatively simple retraining mechanism, providing a strong reference point for measuring whether more sophisticated concept drift adaptation strategies yield clear practical gains.
- **DRMD [41]:** This method formulates malware drift adaptation as a deep reinforcement learning problem. It unifies malware classification, rejection, and active learning into a one-step decision process, allowing an agent to decide how to respond to newly arriving samples under concept drift.
- **DTRACE [42]:** This method performs concept drift handling through drift detection and explanatory analysis. Its core idea is to first identify drifted samples and then interpret the detected drift to guide subsequent model updating.

5.2. Overall Performance of Malware Concept Drift Adaptation

As shown in Table 2, GRAD achieves the highest overall F1 performance on both datasets while maintaining a favorable balance between false-positive and false-negative errors. On APIGraph, it ranks first in all five metrics (FPR/FNR/ACC/F1/F1-AUT), achieving a balanced improvement by simultaneously reducing false positives (0.33%, ↓10.81% vs. UNCER) and false negatives (7.12%, ↓39.15% vs. UNCER). On AndroZoo, GRAD achieves the lowest FPR (0.11%), the highest ACC (97.94%), and the highest F1 score (0.89).

Table 2. Overall metrics averaged over APIGraph and AndroZoo time points.

(a) APIGraph

Method	FPR ↓	FNR ↓	ACC ↑	F1 ↑	F1-AUT ↑
UNCER [23]	**0.37$_{\pm 0.02}$%**	**11.70$_{\pm 0.21}$%**	**98.55$_{\pm 0.03}$%**	**0.92$_{\pm 0.002}$**	**0.92$_{\pm 0.002}$**
CORESET [24]	0.74$_{\pm 0.05}$%	15.79$_{\pm 0.31}$%	97.83$_{\pm 0.06}$%	0.88$_{\pm 0.004}$	0.88$_{\pm 0.004}$
CADE [25]	0.45$_{\pm 0.03}$%	13.36$_{\pm 0.24}$%	98.32$_{\pm 0.04}$%	0.91$_{\pm 0.003}$	0.91$_{\pm 0.003}$
MORPH [38]	0.87$_{\pm 0.08}$%	9.09$_{\pm 0.46}$%	98.36$_{\pm 0.07}$%	0.91$_{\pm 0.005}$	0.91$_{\pm 0.005}$
LDCDroid [39]	0.64$_{\pm 0.09}$%	24.88$_{\pm 0.73}$%	97.03$_{\pm 0.10}$%	0.82$_{\pm 0.007}$	0.82$_{\pm 0.007}$
MADCAT [27]	0.85$_{\pm 0.11}$%	32.00$_{\pm 0.94}$%	96.00$_{\pm 0.13}$%	0.77$_{\pm 0.010}$	0.77$_{\pm 0.011}$
MALFSCIL [28]	7.39$_{\pm 0.17}$%	36.73$_{\pm 0.88}$%	89.81$_{\pm 0.19}$%	0.54$_{\pm 0.009}$	0.54$_{\pm 0.009}$
SIMPLE [45]	6.91$_{\pm 0.15}$%	9.34$_{\pm 0.57}$%	92.85$_{\pm 0.16}$%	0.71$_{\pm 0.008}$	0.71$_{\pm 0.009}$
TRANS [26]	0.42$_{\pm 0.04}$%	12.78$_{\pm 0.25}$%	98.40$_{\pm 0.05}$%	0.91$_{\pm 0.004}$	0.91$_{\pm 0.004}$
HCL [17]	0.49$_{\pm 0.06}$%	11.98$_{\pm 0.22}$%	98.41$_{\pm 0.04}$%	0.91$_{\pm 0.003}$	0.91$_{\pm 0.003}$
DRMD [41]	1.66$_{\pm 0.18}$%	62.50$_{\pm 1.56}$%	92.61$_{\pm 0.22}$%	0.49$_{\pm 0.016}$	0.49$_{\pm 0.016}$
DTRACE [42]	2.25$_{\pm 0.12}$%	43.29$_{\pm 1.02}$%	93.89$_{\pm 0.14}$%	0.63$_{\pm 0.012}$	0.63$_{\pm 0.012}$
F-CADE [25,46]	4.38$_{\pm 0.14}$%	15.13$_{\pm 0.49}$%	94.60$_{\pm 0.13}$%	0.75$_{\pm 0.010}$	0.75$_{\pm 0.011}$
F-CORESET [24,46]	3.20$_{\pm 0.13}$%	15.93$_{\pm 0.44}$%	95.59$_{\pm 0.12}$%	0.78$_{\pm 0.009}$	0.78$_{\pm 0.009}$
F-HCL [17,46]	4.06$_{\pm 0.17}$%	16.31$_{\pm 0.51}$%	94.79$_{\pm 0.16}$%	0.75$_{\pm 0.009}$	0.75$_{\pm 0.009}$
F-TRANS [26,46]	1.51$_{\pm 0.07}$%	11.07$_{\pm 0.34}$%	97.58$_{\pm 0.08}$%	0.88$_{\pm 0.006}$	0.88$_{\pm 0.006}$
F-UNCER [23,46]	3.97$_{\pm 0.16}$%	16.07$_{\pm 0.52}$%	94.88$_{\pm 0.15}$%	0.76$_{\pm 0.008}$	0.76$_{\pm 0.008}$
Ours (GRAD)	**0.33$_{\pm 0.02}$%** (↓10.81%)	**7.12$_{\pm 0.18}$%** (↓39.15%)	**99.02$_{\pm 0.03}$%** (↑0.47%)	**0.95$_{\pm 0.001}$** (↑3.26%)	**0.95$_{\pm 0.001}$** (↑3.26%)

(b) AndroZoo

Method	FPR ↓	FNR ↓	ACC ↑	F1 ↑	F1-AUT ↑
UNCER [23]	**0.40$_{\pm 0.02}$%**	**23.86$_{\pm 0.33}$%**	**97.17$_{\pm 0.04}$%**	**0.83$_{\pm 0.003}$**	**0.82$_{\pm 0.003}$**
CORESET [24]	1.46$_{\pm 0.06}$%	20.07$_{\pm 0.37}$%	96.62$_{\pm 0.07}$%	0.82$_{\pm 0.005}$	0.82$_{\pm 0.005}$
CADE [25]	0.85$_{\pm 0.05}$%	29.43$_{\pm 0.49}$%	96.18$_{\pm 0.08}$%	0.77$_{\pm 0.006}$	0.77$_{\pm 0.006}$
MORPH [38]	0.89$_{\pm 0.10}$%	32.34$_{\pm 1.03}$%	95.88$_{\pm 0.14}$%	0.77$_{\pm 0.013}$	0.76$_{\pm 0.014}$
LDCDroid [39]	0.83$_{\pm 0.12}$%	31.58$_{\pm 1.08}$%	96.03$_{\pm 0.16}$%	0.78$_{\pm 0.014}$	0.77$_{\pm 0.015}$
MADCAT [27]	1.48$_{\pm 0.13}$%	41.26$_{\pm 1.24}$%	94.39$_{\pm 0.18}$%	0.64$_{\pm 0.016}$	0.64$_{\pm 0.016}$
MALFSCIL [28]	15.32$_{\pm 0.28}$%	60.44$_{\pm 1.75}$%	79.93$_{\pm 0.31}$%	0.28$_{\pm 0.019}$	0.27$_{\pm 0.020}$
SIMPLE [45]	11.77$_{\pm 0.25}$%	16.40$_{\pm 0.84}$%	87.75$_{\pm 0.29}$%	0.61$_{\pm 0.012}$	0.61$_{\pm 0.012}$
TRANS [26]	0.33$_{\pm 0.03}$%	29.16$_{\pm 0.41}$%	96.66$_{\pm 0.05}$%	0.79$_{\pm 0.004}$	0.79$_{\pm 0.004}$
HCL [17]	0.49$_{\pm 0.04}$%	30.95$_{\pm 0.45}$%	96.33$_{\pm 0.06}$%	0.78$_{\pm 0.004}$	0.77$_{\pm 0.005}$
DRMD [41]	0.86$_{\pm 0.16}$%	50.28$_{\pm 1.63}$%	93.96$_{\pm 0.22}$%	0.59$_{\pm 0.018}$	0.59$_{\pm 0.018}$
DTRACE [42]	1.30$_{\pm 0.18}$%	54.87$_{\pm 1.74}$%	93.08$_{\pm 0.25}$%	0.54$_{\pm 0.019}$	0.53$_{\pm 0.020}$
F-CADE [25,46]	11.81$_{\pm 0.22}$%	19.90$_{\pm 0.93}$%	87.36$_{\pm 0.27}$%	0.60$_{\pm 0.013}$	0.59$_{\pm 0.014}$
F-CORESET [24,46]	10.08$_{\pm 0.21}$%	17.99$_{\pm 0.88}$%	89.11$_{\pm 0.24}$%	0.65$_{\pm 0.012}$	0.64$_{\pm 0.013}$
F-HCL [17,46]	10.88$_{\pm 0.23}$%	17.42$_{\pm 0.91}$%	88.45$_{\pm 0.26}$%	0.65$_{\pm 0.013}$	0.64$_{\pm 0.013}$
F-TRANS [26,46]	10.07$_{\pm 0.19}$%	29.47$_{\pm 1.02}$%	87.90$_{\pm 0.23}$%	0.59$_{\pm 0.014}$	0.59$_{\pm 0.014}$
F-UNCER [23,46]	10.54$_{\pm 0.20}$%	23.66$_{\pm 0.97}$%	88.07$_{\pm 0.24}$%	0.60$_{\pm 0.013}$	0.59$_{\pm 0.014}$
Ours (GRAD)	**0.11$_{\pm 0.01}$%** (↓72.50%)	**18.89$_{\pm 0.27}$%** (↓20.83%)	**97.94$_{\pm 0.03}$%** (↑0.79%)	**0.89$_{\pm 0.002}$** (↑7.23%)	**0.89$_{\pm 0.002}$** (↑8.54%)

Top-1 and **Top-2** denote the best and second-best methods ranked by overall F1 performance. For Ours, the percentage in parentheses denotes the relative improvement over the overall Top-2 method, i.e., **UNCER**. For APIGraph and AndroZoo, FPR, FNR, ACC, and F1 are computed independently at each monthly evaluation point and then averaged over 72 and 24 points, respectively. F1-AUT is computed from the complete temporal F1 trajectory. ↓ indicates that a lower value is better, whereas ↑ indicates that a higher value is better. Bold values indicate the best and second-best overall-performing methods, as additionally distinguished by the corresponding background shading.

In addition to averaged statistical metrics, we evaluate the long-term stability of GRAD together with the top five baseline methods under concept drift to assess their detection capability. As shown in Figure 2, our method exhibits greater robustness and smaller performance fluctuations on both datasets. On AndroZoo, although all methods experience performance degradation at several evaluation points, GRAD generally exhibits smaller F1 fluctuations and maintains relatively high performance over most of the evaluation horizon. Quantitatively, it achieves the best F1 performance at 17 of the 24 AndroZoo test

points (70.83%). This stability is more evident on APIGraph, where our method maintains consistently high F1 scores across most evaluation points.

We further evaluate long-term Android malware detection using the LLMS metric defined in Section 5.1.3. As shown in Figure 3, lower LLMS values indicate that fewer malware samples remain undetected for an extended period. On APIGraph, GRAD achieves the lowest LLMS value of 644, compared with 736 for UNCER and 803 for HCL, corresponding to reductions of 12.50% and 19.80%, respectively. These results indicate that GRAD reduces the number of long-lived undetected malware samples on APIGraph. On AndroZoo, GRAD does not achieve the lowest LLMS value but remains competitive, suggesting that its long-term detection performance remains stable under a shorter temporal evaluation horizon.

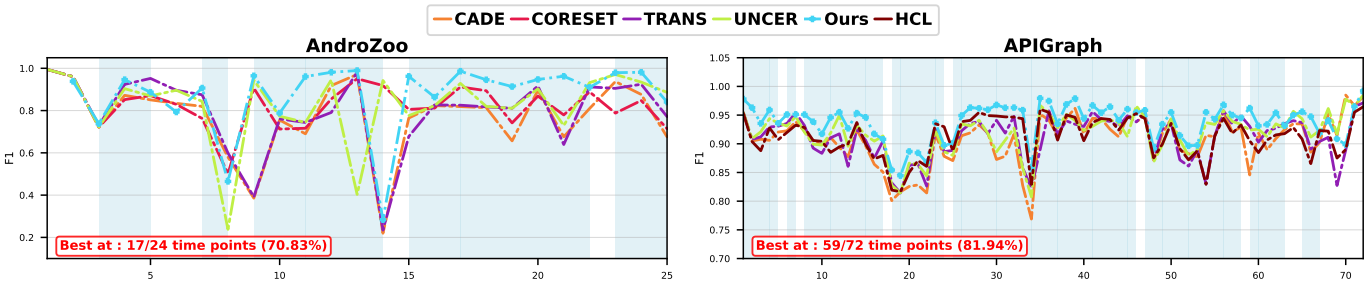


Figure 2. Stability comparison of GRAD and the top five baseline methods in terms of overall performance.

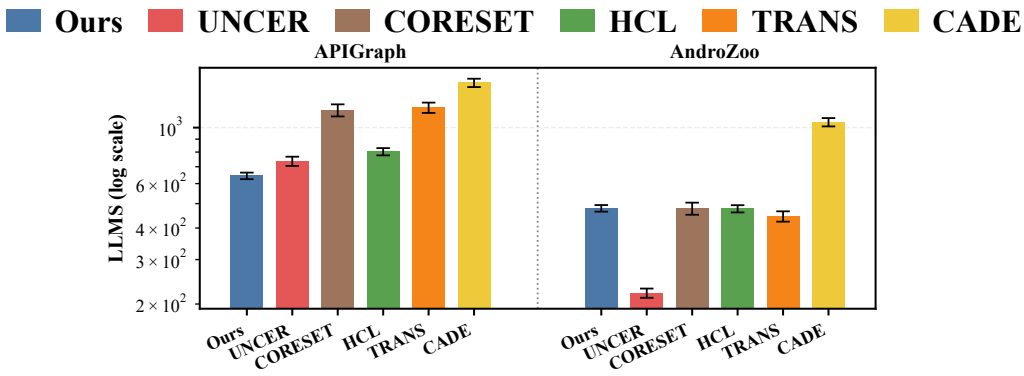


Figure 3. Long-lived malware sample counts of GRAD and the top five baselines. The bars represent the mean values, and the error bars indicate the corresponding standard deviations.

For LLMS, the temporal threshold is set to one sixth of the corresponding evaluation horizon. Since APIGraph and AndroZoo span 72 and 24 evaluation months, respectively, this corresponds to 12 months for APIGraph and 4 months for AndroZoo. We use a relative rather than fixed absolute threshold because the two datasets have substantially different evaluation horizons, and the one-sixth criterion therefore represents the same normalized temporal proportion in both settings. This choice does not imply that 12 months on APIGraph and 4 months on AndroZoo have identical absolute temporal meanings. A smaller threshold makes the LLMS criterion easier to satisfy, whereas a larger threshold makes it more stringent. We therefore additionally examine the sensitivity of LLMS to different threshold ratios in Appendix D.3.

As shown in Figure 4, our method delivers performance gains without introducing unacceptable time overhead. On both datasets, our method falls within the top 50% of methods by runtime efficiency.

These results indicate that, under a longer evaluation horizon, our method avoids the heavier training overhead incurred by competing approaches. Regarding scalability, the cost of nearest-neighbor search increases with the size of the accumulated training

pool, since each query retrieves its local neighbors from the labeled reference set. In contrast, local PCA is performed only on a fixed-size neighborhood in a fixed-dimensional representation space, and thus its computational cost does not directly increase with the total number of training samples. The cumulative training pool may also increase model retraining cost across successive drift cycles. Nevertheless, as shown in Figure 4, GRAD maintains a favorable balance between detection performance and runtime efficiency over the evaluated temporal horizon, indicating that the resulting computational overhead remains manageable in our experimental setting.

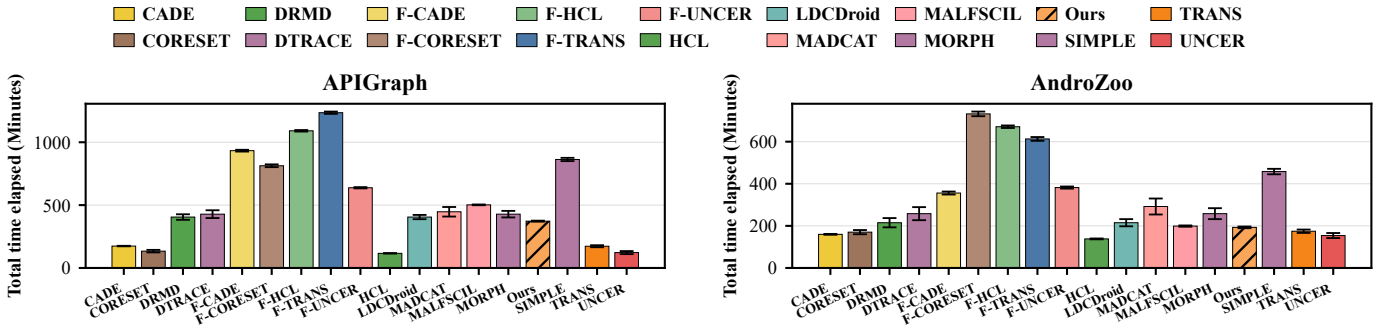


Figure 4. Runtime Ccomparison of concept drift adaptation methods. The bars represent the mean values, and the error bars indicate the corresponding standard deviations.

5.3. Ablation Study

To examine the contribution of each component in the proposed framework, we conduct ablation studies on its three key modules: geometry-aware drift-risk ranking, hybrid sample selection based on geometry-aware drift-risk ranking and uncertainty, and boundary-aware retraining with dynamically constructed error and semantic ambiguity memories. By progressively removing or replacing each component, we evaluate how they affect adaptation performance under concept drift (as shown in Table 3). Specifically, we consider the following variants. **Full** denotes the complete framework. **w/o GeOAware** removes geometry-aware drift-risk selection and constructs the queried set solely using pseudo-loss-based uncertainty. **w/o Uncer** removes uncertainty-based selection and constructs the queried set solely using geometry-ranked drift-risk candidates. **w/o BoundaryLoss** removes the memory-guided boundary constraints introduced by Error Memory and Semantic Ambiguity Memory, while retaining the base training objective. Experimental results show that removing or replacing individual components generally degrades the overall adaptation performance across the two datasets.

Table 3. Ablation study on APIGraph and AndroZoo (averaged over time points).

Dataset	Method	FPR ↓	FNR ↓	ACC ↑	PREC ↑	F1 ↑
APIGraph	w/o GeOAware	$0.51_{\pm 0.03}$%	$11.31_{\pm 0.28}$%	$98.46_{\pm 0.04}$%	$94.83_{\pm 0.24}$%	$0.92_{\pm 0.003}$
APIGraph	Full	**$0.33_{\pm 0.02}$% (↓35.29%)**	**$7.12_{\pm 0.18}$% (↓37.05%)**	**$99.02_{\pm 0.03}$% (↑0.57%)**	**$96.75_{\pm 0.17}$% (↑2.02%)**	**$0.95_{\pm 0.001}$ (↑3.26%)**
AndroZoo	w/o GeOAware	$0.53_{\pm 0.04}$%	$29.44_{\pm 0.53}$%	$96.46_{\pm 0.07}$%	$93.05_{\pm 0.35}$%	$0.78_{\pm 0.005}$
AndroZoo	Full	**$0.11_{\pm 0.01}$% (↓79.25%)**	**$18.89_{\pm 0.27}$% (↓35.84%)**	**$97.94_{\pm 0.03}$% (↑1.53%)**	**$98.83_{\pm 0.12}$% (↑6.21%)**	**$0.89_{\pm 0.002}$ (↑14.10%)**
APIGraph	w/o BoundaryLoss	$0.43_{\pm 0.03}$%	$8.25_{\pm 0.23}$%	$98.83_{\pm 0.03}$%	$96.07_{\pm 0.18}$%	$0.94_{\pm 0.002}$
APIGraph	Full	**$0.33_{\pm 0.02}$% (↓23.26%)**	**$7.12_{\pm 0.18}$% (↓13.70%)**	**$99.02_{\pm 0.03}$% (↑0.19%)**	**$96.75_{\pm 0.17}$% (↑0.71%)**	**$0.95_{\pm 0.001}$ (↑1.06%)**
AndroZoo	w/o BoundaryLoss	$0.18_{\pm 0.02}$%	$19.15_{\pm 0.31}$%	$97.85_{\pm 0.04}$%	$97.55_{\pm 0.14}$%	$0.88_{\pm 0.003}$
AndroZoo	Full	**$0.11_{\pm 0.01}$% (↓38.89%)**	**$18.89_{\pm 0.27}$% (↓1.36%)**	**$97.94_{\pm 0.03}$% (↑0.09%)**	**$98.83_{\pm 0.12}$% (↑1.31%)**	**$0.89_{\pm 0.002}$ (↑1.14%)**
APIGraph	w/o Uncer	$0.48_{\pm 0.03}$%	$10.51_{\pm 0.27}$%	$98.56_{\pm 0.04}$%	$95.26_{\pm 0.22}$%	$0.92_{\pm 0.003}$
APIGraph	Full	**$0.33_{\pm 0.02}$% (↓31.25%)**	**$7.12_{\pm 0.18}$% (↓32.25%)**	**$99.02_{\pm 0.03}$% (↑0.47%)**	**$96.75_{\pm 0.17}$% (↑1.56%)**	**$0.95_{\pm 0.001}$ (↑3.26%)**
AndroZoo	w/o Uncer	$0.18_{\pm 0.02}$%	$19.55_{\pm 0.34}$%	$97.80_{\pm 0.05}$%	$98.15_{\pm 0.13}$%	$0.87_{\pm 0.003}$
AndroZoo	Full	**$0.11_{\pm 0.01}$% (↓38.89%)**	**$18.89_{\pm 0.27}$% (↓3.38%)**	**$97.94_{\pm 0.03}$% (↑0.14%)**	**$98.83_{\pm 0.12}$% (↑0.69%)**	**$0.89_{\pm 0.002}$ (↑2.30%)**

↓ indicates that a lower value is better, whereas ↑ indicates that a higher value is better. The shaded and bold entries indicate the results of the full GRAD model; percentages in parentheses denote relative improvements over the corresponding ablated variant. Red shading and boldface indicate the best result for each metric.

5.4. Model Forgetting Evaluation Under Concept Drift Adaptation

To measure past-task retention under continual concept drift adaptation [47,48], we evaluate Backward Transfer (BWT) per cycle. BWT compares the current model's performance on each preceding cycle with the reference performance obtained immediately after that cycle was learned (defined in Section 5.1.3), where negative values indicate average forgetting and positive values indicate beneficial backward transfer. Figure 5 shows that F1-BWT values of all methods stay positive on both datasets. The consistently non-negative F1-BWT values indicate that, on average, subsequent adaptation does not systematically degrade performance on previously observed drift cycles. These results provide no evidence of systematic catastrophic forgetting under the adopted BWT criterion.

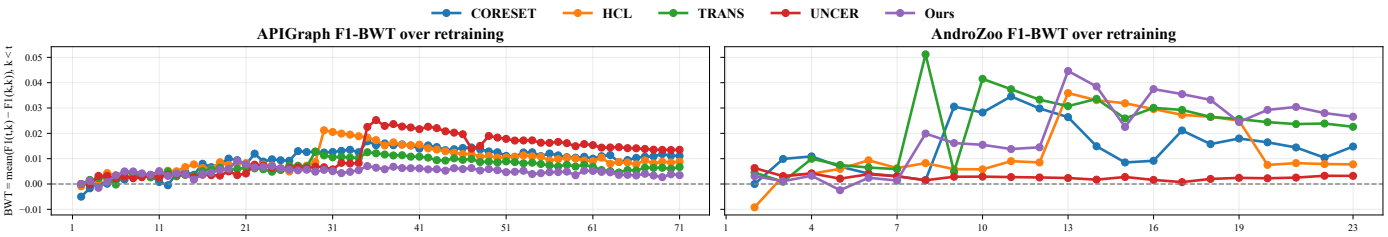


Figure 5. Forgetting evaluation of concept drift adaptation methods.

5.5. Concept Drift Adaptation Performance Under Different Hyperparameter Settings

As shown in Figure 6, the DIST parameter, which controls the coverage range of the local reference neighborhood, has a limited impact on the overall performance of GRAD. Specifically, DIST is implemented through the number of nearest neighbors k, with DIST8, DIST12, and DIST16 corresponding to $k = 8$, $k = 12$, and $k = 16$, respectively. A larger DIST therefore represents broader local neighborhood coverage rather than a larger metric-distance threshold.

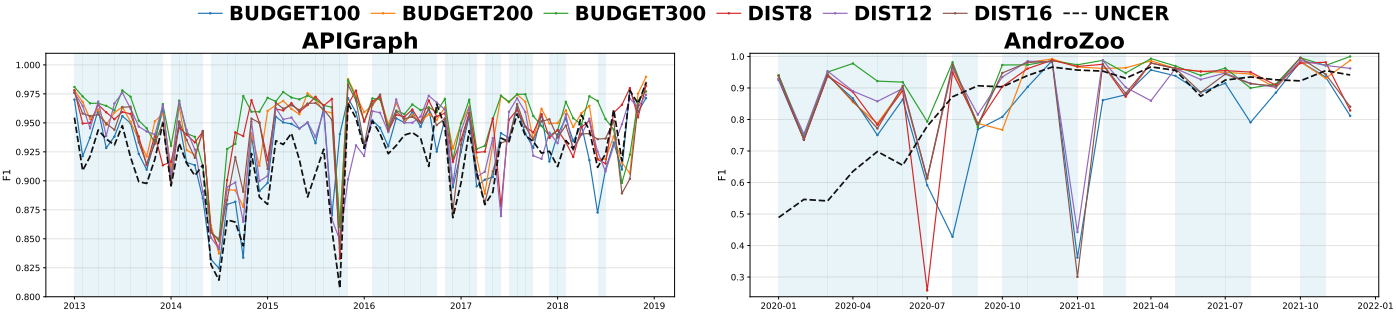


Figure 6. Concept drift adaptation performance under different hyperparameter settings.

On APIGraph, using UNCER as the reference baseline, the average F1 scores under DIST settings DIST8, DIST12, and DIST16 are 0.9441, 0.9358, and 0.9388, respectively, all exceeding UNCER's 0.9194. The corresponding numbers of monthly test points at which our method is no worse than UNCER are 61/72, 57/72, and 64/72. Qualitatively, the three DIST curves remain above the reference baseline for most test points. Moreover, the shaded regions show that all DIST settings simultaneously outperform the baseline in 52/72 months on APIGraph and 15/24 months on AndroZoo. These results indicate that varying DIST only introduces minor fluctuations and does not alter the overall performance advantage of GRAD. In contrast, BUDGET shows a clearer and more consistent effect. On both datasets, the average F1 increases monotonically as BUDGET grows from BUDGET100 to BUDGET200 and BUDGET300. Specifically, on APIGraph, the mean F1 improves from 0.9296 to 0.9449 and 0.9533, corresponding to gains of 1.53 and 0.84 percentage points. On AndroZoo, the improvement is more pronounced, with the mean F1 increasing from

0.8237 to 0.9064 and 0.9372, yielding gains of 8.27 and 3.08 percentage points. This suggests that a larger adaptation budget enables more effective model updates. Overall, the hyperparameter analysis shows that our method is relatively insensitive to DIST, while benefiting consistently from a larger BUDGET. Additional sensitivity analyses of the tangent-space dimension, auxiliary loss weights, and the allocation between FN and FP risk candidates are provided in Appendix D.2.

6. Conclusions and Future Work

This paper addresses the challenge of concept drift in Android malware detection, which can substantially degrade the effectiveness of supervised detectors over time. We identify key limitations of existing CDA-AL methods in prioritizing security-critical drift-risk samples, constructing informative retraining sets under limited annotation budgets, and sufficiently emphasizing difficult samples during model adaptation. Accordingly, we propose GRAD, a geometry-aware and risk-guided active drift adaptation framework that integrates drift-risk ranking, hybrid active sample selection, and boundary-aware retraining. Specifically, GRAD exploits local benign representation geometry to prioritize potential false-negative and false-positive risk candidates, combines these candidates with uncertainty-driven samples to construct an informative queried set, and dynamically emphasizes false-negative and semantically ambiguous samples during retraining. Experimental results demonstrate the effectiveness of GRAD in improving Android malware detection performance under long-term concept drift while maintaining stable adaptation across successive drift cycles. For future work, we plan to further investigate annotation latency and imperfect human labeling in practical deployment. The current study assumes that queried samples can be annotated by human experts within each adaptation cycle rather than instantaneously, and that the resulting labels are reliable. In practice, delayed or erroneous annotations may influence subsequent adaptation performance. Therefore, extending GRAD to explicitly account for delayed feedback and label noise will be an important direction for future research. Future work will further evaluate GRAD on other malware platforms, dynamic and multimodal feature representations, and broader cybersecurity scenarios to assess its generalizability beyond the current Android malware setting.

Author Contributions: Conceptualization, K.W.; methodology, K.W.; software, K.W. and K.C.; validation, K.W., K.C. and Y.L.; resources, J.L. and J.W.; writing and editing, K.W., J.S., and Y.Y.; visualization, K.W. and K.C.; project administration, J.L.; funding acquisition, J.L. and J.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported in part by the National Key R&D Program of China under Grant 2023YFB2703702.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All datasets used in this study are publicly available and can be accessed via the references provided in the paper.

Acknowledgments: The authors are grateful to the providers of the open datasets used in this study. The open sharing of these resources greatly facilitated our research. The authors also sincerely thank all collaborators for their valuable discussions, constructive feedback, and continued support throughout this work.

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

Abbreviations

The following abbreviations are used in this manuscript:

Notation	Description
n	Index of the current concept-drift adaptation cycle.
$\mathcal{D}_{tr}^{n}$, $\mathcal{D}_{te}^{n}$	Labeled training pool after cycle n and incoming unlabeled test batch at cycle n.
$\mathcal{D}_{\text{query}}^{n}$	Labeled queried set obtained by manually annotating $\mathcal{I}_{\text{query}}^{n}$.
x_q, y_q, $\hat{y}_q$	Query sample, its ground-truth label, and its predicted pseudo-label.
f_{θ_n}, g_{θ_n}	Detector and encoder parameterized by θ_n after cycle n.
$\mathbf{z}_q$	Embedding representation of query sample x_q, i.e., $\mathbf{z}_q = g_{\theta_{n-1}}(x_q)$.
$\mathcal{D}_{\text{ben}}^{n-1}$	Benign subset of the labeled training pool from the previous cycle.
k, r	Number of nearest neighbors and local tangent-space dimension used for benign manifold estimation.
$\mathbf{c}_q$, $\mathbf{B}_q$	Local benign center and PCA-based tangent basis around query sample x_q.
$\Delta\mathbf{z}_q^{\text{tan}}$, $\Delta\mathbf{z}_q^{\text{res}}$	Tangent and residual components of the displacement from $\mathbf{z}_q$ to the local benign manifold.
$d_{\text{tan}}(q)$, $d_{\text{res}}(q)$	Tangent and residual distances characterizing local benign geometry in the latent space.
$\mathcal{I}_{\text{FN}}^{n}$, $\mathcal{I}_{\text{FP}}^{n}$	Potential false-negative and false-positive risk candidate sets selected by geometry-aware drift-risk ranking.
$\mathcal{I}_{\text{cal}}^{n}$	Geometry-ranked drift-risk set, defined as $\mathcal{I}_{\text{FN}}^{n} \cup \mathcal{I}_{\text{FP}}^{n}$.
β, β_{FN}, β_{FP}	Total annotation budget and its pre-allocated portions for false-negative and false-positive risk samples.
$s_{\text{ps}}(q)$	Pseudo-loss uncertainty score combining local contrastive inconsistency and pseudo cross-entropy.
$\mathcal{I}_{\text{query}}^{n}$	Final queried sample index set selected for manual annotation at cycle n.
$\mathcal{M}_{\text{error}}$, $\mathcal{M}_{\text{sem}}$	Error memory and semantic ambiguity memory used for boundary-aware retraining.
$\mathcal{L}_{\text{total}}$	Overall retraining objective, combining classification loss, error-memory loss, and semantic-memory loss.

Appendix A. Limitations and Discussion

Appendix A.1. Limitations

Our current evaluation focuses on Android malware and primarily static feature representations. Although APIGraph enriches API-based features with semantic relations, broader validation on other malware platforms, dynamic behavioral features, and multimodal representations remains to be investigated. In addition, our experiments are conducted on two temporal Android malware datasets, which provide complementary long-term and recent evaluation settings but do not cover all possible malware evolution scenarios. Our active-learning protocol assumes that queried samples can be reliably labeled before the subsequent model update, following the common setting adopted in existing concept-drift adaptation studies. Practical factors such as annotation delay or imperfect labels are beyond the scope of the current study and represent meaningful directions for future investigation. Therefore, the conclusions of this work are primarily grounded in the evaluated Android malware detection settings.

Appendix A.2. Discussion

Differences between the two datasets. The experimental results show that GRAD consistently improves adaptation performance on both APIGraph and AndroZoo, while the overall difficulty differs between the two datasets. APIGraph provides a substantially longer temporal evaluation horizon, whereas AndroZoo exhibits higher false-negative rates for most compared methods, indicating a more challenging detection setting. The consistent

advantage of GRAD across these different temporal conditions suggests that its effectiveness is not restricted to a single dataset. At the same time, the remaining performance differences between APIGraph and AndroZoo indicate that adaptation difficulty is affected by the underlying temporal distribution and feature characteristics.

Contribution of individual modules. The three components of GRAD play complementary roles in the adaptation process. The geometry-aware drift-risk ranking module identifies samples that are locally inconsistent with the historical benign representation structure and prioritizes potential false-negative and false-positive risks. The hybrid selection module combines this risk-oriented criterion with uncertainty-based querying, allowing the limited annotation budget to cover both security-relevant risk samples and generally informative samples. Finally, the boundary-aware retraining module uses the resulting labeled difficult samples to construct targeted metric-learning constraints for correcting false-negative-prone and locally ambiguous representation regions. Therefore, the three modules respectively contribute to risk identification, selective annotation, and representation-space correction.

Applicability and assumption limitations. GRAD relies on the assumption that historical benign samples provide a meaningful local geometric reference in the learned representation space. This assumption may become less reliable under abrupt and large-scale benign distribution shifts, where historical benign neighborhoods no longer represent newly arriving benign applications. The geometry-aware signal may also weaken when the learned representation does not preserve meaningful local neighborhood structure or when only a small number of relevant benign neighbors are available in sparse regions. In these situations, local tangent estimation may become unstable and geometric deviation may provide less reliable ranking information. Importantly, GRAD does not treat the geometric score as a definitive indicator of concept drift; it is used only as one ranking signal and is complemented by uncertainty-based sample selection.

Appendix B. Implementation Detail

Appendix B.1. Dataset Settings

The detailed temporal splits of the two datasets are reported in Tables A1 and A2. For both datasets, samples are assigned to monthly subsets according to their first-appearance timestamps. The initial training sets consist of all samples from 2019 for AndroZoo and from 2012 for APIGraph, while the remaining samples are organized chronologically into monthly drift-evaluation batches. Each table reports the numbers of malicious samples, benign samples, total samples, and malware families for every monthly subset, providing the exact data partition used in our experiments.

Table A1. Monthly statistics of AndroZoo [22].

Month	Malicious	Benign	Total	Malware Families
2019 (Train)	4542	40,947	45,489	122
2020-01	333	2985	3318	26
2020-02	406	3697	4103	31
2020-03	366	3301	3667	28
2020-04	337	2966	3303	26
2020-05	389	3408	3797	31
2020-06	197	1752	1949	34
2020-07	492	4334	4826	22
2020-08	491	4230	4721	17
2020-09	288	2441	2729	27
2020-10	170	1481	1651	18

Table A1. *Cont.*

Month	Malicious	Benign	Total	Malware Families
2020-11	134	1139	1273	14
2020-12	380	3187	3567	13
2021-01	445	3706	4151	20
2021-02	41	362	403	12
2021-03	128	1087	1215	16
2021-04	73	619	692	12
2021-05	68	576	644	19
2021-06	44	415	459	11
2021-07	262	2241	2503	17
2021-08	173	1395	1568	13
2021-09	228	1867	2095	12
2021-10	120	953	1073	5
2021-11	54	423	477	12
2021-12	40	341	381	10
Total	**10,201**	**89,853**	**100,054**	**255**

The Total row is shown in bold to distinguish the aggregate statistics from monthly entries.

Table A2. Monthly statistics of APIGraph [21].

Month	Malicious	Benign	Total	Malware Families
2012 (Train)	3061	27,472	30,533	105
2013-01	500	4491	4991	37
2013-02	320	2865	3185	38
2013-03	337	3028	3365	44
2013-04	389	3497	3886	46
2013-05	482	4383	4865	49
2013-06	338	3046	3384	50
2013-07	253	2267	2520	36
2013-08	242	2194	2436	46
2013-09	500	4484	4984	57
2013-10	499	4490	4989	70
2013-11	495	4477	4972	49
2013-12	499	4492	4991	49
2014-01	493	4490	4983	63
2014-02	495	4486	4981	46
2014-03	494	4489	4983	57
2014-04	440	3990	4430	66
2014-05	497	4488	4985	64
2014-06	498	4491	4989	64
2014-07	499	4495	4994	54
2014-08	496	4493	4989	64
2014-09	422	3828	4250	68
2014-10	489	4480	4969	55
2014-11	493	4482	4975	62
2014-12	493	4464	4957	57
2015-01	478	4453	4931	60
2015-02	482	4472	4954	61
2015-03	471	4468	4939	64
2015-04	467	4479	4946	61
2015-05	444	4490	4934	67
2015-06	457	4489	4946	68

Table A2. *Cont.*

Month	Malicious	Benign	Total	Malware Families
2015-07	477	4347	4824	79
2015-08	306	2805	3111	66
2015-09	482	4477	4959	88
2015-10	475	4488	4963	78
2015-11	488	4487	4975	47
2015-12	481	4489	4970	70
2016-01	436	4495	4931	75
2016-02	434	4483	4917	62
2016-03	453	4497	4950	66
2016-04	432	4494	4926	80
2016-05	477	4491	4968	76
2016-06	492	4482	4974	57
2016-07	481	4489	4970	73
2016-08	497	4484	4981	64
2016-09	493	4472	4965	63
2016-10	492	4485	4977	55
2016-11	330	2997	3327	60
2016-12	307	2843	3150	69
2017-01	320	3090	3410	71
2017-02	368	3402	3770	63
2017-03	332	3264	3596	68
2017-04	155	1750	1905	41
2017-05	99	1166	1265	35
2017-06	113	1286	1399	40
2017-07	187	1684	1871	51
2017-08	296	2693	2989	40
2017-09	157	1607	1764	30
2017-10	221	2693	2914	27
2017-11	92	895	987	29
2017-12	125	1317	1442	36
2018-01	153	1683	1836	26
2018-02	145	1646	1791	35
2018-03	274	4491	4765	36
2018-04	281	2685	2966	31
2018-05	242	2236	2478	31
2018-06	478	4492	4970	65
2018-07	488	4495	4983	46
2018-08	291	2698	2989	19
2018-09	458	4495	4953	32
2018-10	493	4496	4989	27
2018-11	287	2697	2984	28
2018-12	193	2032	2225	19
Total	**30,804**	**289,511**	**320,315**	**1119**

The Total row is shown in bold to distinguish the aggregate statistics from monthly entries.

In our experiments, we directly use the released and processed feature representations associated with the two datasets. Therefore, no additional APK-level preprocessing or feature extraction is introduced in our experimental pipeline. The feature-generation procedures follow the original dataset construction protocols described in the corresponding studies [21,22].

No feature vocabulary is reconstructed, no additional feature selection is performed, and no preprocessing step uses information from future evaluation periods. The temporal

protocol is strictly chronological: samples from a future month are introduced only when that month is reached and are never used to process or train models for earlier time points.

Appendix B.2. Baseline Settings

SIMPLE [45] was originally proposed for few-shot malware family classification and unknown-family recognition rather than concept-drift adaptation. Its core mechanism is supervised multi-prototype modeling, which represents each malware family using multiple prototypes to capture multimodal intra-family distributions. During its adapting phase, SIMPLE follows a supervised "try-catch" strategy, in which samples that are inconsistent with existing prototypes can lead to the construction of additional prototypes. During inference, query embeddings are classified according to their similarity to the resulting prototype set. In our experiments, we preserve SIMPLE's original multi-prototype construction and prototype-based inference mechanisms. The backbone network is kept fixed throughout the temporal adaptation process and is neither retrained nor fine-tuned at each drift cycle. The adaptation introduced in our evaluation is only at the protocol level: instead of using static few-shot episodes, we sequentially apply SIMPLE's adapting and inference phases to the temporally ordered malware stream. At each adaptation cycle, the samples queried and labeled under the common annotation budget are provided to SIMPLE to update its prototype set, after which the updated prototypes are used for subsequent inference. Therefore, adaptation is achieved through prototype construction and update in the embedding space without modifying the backbone parameters. Accordingly, SIMPLE is evaluated as a prototype-based fast-adaptation baseline rather than as an originally proposed concept-drift adaptation method.

Appendix C

Appendix C.1. Model Settings

Our model configuration consists of two main components: a feature extraction encoder and a sample classifier. To ensure a controlled comparison, GRAD and the classical concept drift adaptation baselines use the same encoder and neural binary classifier architecture. The few-shot variants use the same encoder architecture but replace the neural classifier with a prototype-based classifier. The detailed configurations are described below.

a. Classic adaptive scheme for concept drift. For the classical concept drift adaptation methods, including UNCER, CORESET, CADE, TRANS, and HCL, the encoder input dimension is set to the feature dimension of each dataset, namely 1159 for APIGraph and 16,978 for AndroZoo. The encoder consists of three fully connected hidden layers with dimensions 512, 384, and 256, respectively, each followed by a ReLU activation function. The output of the final hidden layer is further projected into a 128-dimensional embedding space. The classifier takes the 128-dimensional embedding as input and contains two fully connected hidden layers, each containing 100 units and followed by a ReLU activation function. A final fully connected layer produces two output logits corresponding to the benign and malicious classes. The output logits are converted into class probabilities using the Softmax function during prediction. GRAD and the classical concept drift adaptation baselines share this encoder and classifier architecture to ensure that performance differences primarily reflect their adaptation and sample-selection strategies rather than different network capacities.

b. Adaptive scheme for concept drift based on few-shot learning. For the few-shot variants, including F-UNCER, F-CORESET, F-CADE, F-TRANS, and F-HCL, we use the same feature extraction encoder as in the classical adaptation setting. Specifically, the input features are processed by three fully connected layers with dimensions 512, 384, and 256, respectively, each followed by a ReLU activation function, and are subsequently mapped

into the same 128-dimensional embedding space. Unlike the classical adaptation methods, the few-shot variants use a prototype-based classifier instead of the fully connected neural classifier. Few-shot optimization is performed episodically using labeled samples from the current training pool. During initial training, episodic support and query samples are drawn from the initial labeled training set. At adaptation cycle n, after the samples selected from the incoming unlabeled batch $\mathcal{D}_{te}^{n}$ have been manually annotated and incorporated into the cumulative labeled training set,

$$\mathcal{D}_{tr}^{n} = \mathcal{D}_{tr}^{n-1} \cup \mathcal{D}_{\text{query}}^{n},$$

the support and episodic query sets used for few-shot retraining are sampled from $\mathcal{D}_{tr}^{n}$. Therefore, these episodic support and query samples are already labeled training samples and do not require additional manual annotation. For each episode, class-balanced sampling is used to construct the support set. Given N_{shot} support samples per class, the support set contains labeled benign and malicious samples from the current training pool. For each class $c \in \{0, 1\}$, we compute its prototype as the mean of the 128-dimensional embeddings of its support samples:

$$\mathbf{p}_c = \frac{1}{|\mathcal{S}_c|} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{S}_c} g_\theta(\mathbf{x}_i),$$

where $\mathcal{S}_c$ denotes the support subset belonging to class c. For an episodic query sample x_q, we compute its Euclidean distance to the benign and malicious prototypes in the embedding space.

The negative distances are used as class similarity scores:

$$s_c(x_q) = -\left\| g_\theta(x_q) - \mathbf{p}_c \right\|_2.$$

These similarity scores are converted into class probabilities through the Softmax function:

$$p(y = c \mid x_q) = \frac{\exp(s_c(x_q))}{\sum_{c' \in \{0,1\}} \exp(s_{c'}(x_q))}.$$

The predicted class is the class associated with the highest resulting probability. It is important to distinguish the episodic query samples used in the few-shot optimization procedure from the samples queried for manual annotation during active concept drift adaptation. The latter are selected from the incoming unlabeled batch according to the corresponding active sampling strategy and consume the fixed annotation budget β. After manual annotation, they are incorporated into the cumulative labeled training pool. By contrast, the episodic support and query samples are subsequently sampled from this labeled training pool for prototype-based optimization and therefore do not consume any additional annotation budget.

Appendix C.2. Parameter Settings

The parameter settings of our model are provided below.

a. Classic adaptive scheme for concept drift. For classical concept drift adaptation methods, model parameters vary slightly across datasets, and their specific configurations are detailed below. For the APIGraph dataset, we train the initial model with the SGD optimizer (initial learning rate: 0.003), a learning rate scheduling strategy (decaying every 10 epochs), and a 250-epoch training period. After new samples are introduced each month, we switch to the Adam optimizer (learning rate: 0.00015) with a 100-epoch training period. For the AndroZoo dataset: since its feature dimension is much larger than that of APIGraph, we adjust the initial learning rate to 0.001, while all other parameters remain consistent with APIGraph.

b. Adaptive scheme for concept drift based on few-shot learning. For comparative methods incorporating few-shot learning, we adopt a 2-way-2-shot-2-query episodic configuration. Each training episode contains two classes, with two support samples and two episodic query samples sampled from each class. Therefore, each episode contains four support samples and four episodic query samples in total. The episodic query samples are used only for few-shot optimization and are distinct from the samples queried for manual annotation during monthly active adaptation. During the initial training phase, each epoch contains 1000 randomly sampled episodes. We use the Adam optimizer with a learning rate of 0.0005 and a learning-rate scheduler that decays the learning rate every 50 epochs. The initial training stage is conducted for 200 epochs. For monthly adaptation, the model is updated using the same episodic training paradigm. Each epoch contains 250 sampled episodes, and the optimizer configuration and initial learning rate remain the same as in the initial training stage. The model is updated for 50 epochs in each monthly adaptation cycle.

c. GRAD-specific hyperparameters. We set the benign neighborhood size $k = 8$, tangent-space dimension $r = 5$, and local contrastive neighborhood parameter $N_{\text{nbr}} = 10$. The pseudo-loss margin is set to $m = 0.3$, and the total annotation budget is $\beta = 200$ per adaptation cycle. The drift-risk budgets are allocated as $\beta_{\text{FN}} = 30$ and $\beta_{\text{FP}} = 30$. For boundary-aware retraining, we set the number of ambiguous-hard samples to $B = 30$. The loss weights are set to $\mu = 0.1$ and $\lambda = 0.1$, and the triplet margins are set to $m_{\text{error}} = 0.3$ and $m_{\text{sem}} = 0.3$ for the Error Memory and Semantic Ambiguity Memory, respectively.

Appendix C.3. Standard Deviation Reporting and Computation

In this paper, entries reported in the mean $\pm$ standard deviation format (e.g., Tables 2 and 3) are computed across multiple independent runs under fixed experimental settings. All experiments were repeated five times using the fixed random seeds $\{0, 1, 2, 3, 4\}$. For each run, the same seed was used to control model initialization, mini-batch shuffling, and other stochastic operations involved in model training. The temporal data splits, nearest-neighbor retrieval, local PCA estimation, and geometry-based ranking procedures are deterministic once the data and model state are fixed. For each metric, we report the mean and sample standard deviation over the five independent runs. We report each metric as $\bar{m} \pm s$, where $\bar{m}$ is the mean over independent runs and s is the sample standard deviation, i.e.,

$$\bar{m} = \frac{1}{R} \sum_{r=1}^{R} m_r, \tag{A1}$$

and

$$s = \sqrt{\frac{1}{R-1} \sum_{r=1}^{R} (m_r - \bar{m})^2}. \tag{A2}$$

Here, R denotes the number of independent runs with different random seeds.

Appendix D. Supplementary Experiments

This section provides supplementary experimental results that further support the analyses presented in the main text. Due to space limitations, these additional evaluations are reported in the Appendix rather than the main body.

Appendix D.1. Effectiveness of Geometry-Based Risk Ranking

To directly evaluate whether the proposed geometry-aware ranking identifies the intended prediction risks, we measure the precision of the candidates selected by the false-negative and false-positive risk channels (as shown in Table A3). For the false-negative

channel, a selected candidate is counted as a hit if it is predicted as benign but its ground-truth label is malicious. For the false-positive channel, a selected candidate is counted as a hit if it is predicted as malicious but its ground-truth label is benign. We compare the proposed geometry-based ranking with uncertainty-based and random selection using the same predicted-class candidate pools and the same selection budgets. Specifically, 30 candidates are selected for each risk channel. Ground-truth labels are used only for retrospective evaluation and are not available to any selection strategy during querying.

Table A3. Precision of selected false-negative and false-positive risk candidates.

Dataset	Selection	FN Precision	FP Precision
APIGraph	Random	0.9%	3.2%
	Uncertainty	14.8%	10.5%
	Geometry	31.6%	21.3%
AndroZoo	Random	2.2%	1.1%
	Uncertainty	19.7%	3.6%
	Geometry	38.4%	8.4%

The results provide direct empirical support for using local benign geometry as a risk-ranking signal. On both datasets, geometry-based ranking concentrates actual false-negative and false-positive errors more effectively than uncertainty-based and random selection under the same candidate pools and selection budgets. This does not imply that geometric deviation is a sufficient indicator of concept drift. Rather, it shows that consistency with the historical benign geometry contains useful information for prioritizing samples whose current predictions are more likely to be incorrect.

Appendix D.2. Sensitivity of GRAD-Specific Hyperparameters

In addition to the annotation budget and neighborhood-size analyses reported in the main text, we further investigate several GRAD-specific parameters that directly affect boundary-aware retraining and risk-oriented sample selection. Unless otherwise specified, only the parameter under investigation is varied, while all remaining parameters are fixed to their default values.

a. Boundary-aware retraining parameters. We first analyze the number of ambiguous-hard samples B, the weights of the two auxiliary losses, and their triplet margins. We vary $B \in \{10, 20, 30, 40, 50\}$, $\mu = \lambda \in \{0.05, 0.1, 0.2, 0.5, 1.0\}$, and $m_{\text{error}} = m_{\text{sem}} \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$. The two auxiliary loss weights and the two triplet margins are varied jointly because they use identical default values and jointly control complementary geometric constraints in the same representation space. This controlled setting allows us to examine the overall strength of the boundary-aware regularization without introducing an unnecessarily large combinatorial search space.

As shown in Table A4, GRAD remains stable over a reasonable range of the examined hyperparameters on both datasets, while excessively small or large values lead to moderate performance degradation. For the number of ambiguous-hard samples B, the F1 score increases as B grows from 10 to 30, and then slightly decreases when more samples are included. This suggests that retaining a moderate number of ambiguous-hard samples provides sufficient boundary information without introducing too many less informative samples into the semantic ambiguity memory. For the auxiliary loss weights, $\mu = \lambda = 0.1$ achieves the strongest overall performance. Increasing the weights to 0.5 or 1.0 results in a more noticeable decrease in F1, indicating that overly strong geometric regularization may interfere with the primary classification objective. Similarly, the triplet margins show the best performance around $m_{\text{error}} = m_{\text{sem}} = 0.3$. Smaller margins provide weaker separation constraints, whereas excessively large margins impose stronger representation-separation

requirements and slightly reduce detection performance. Overall, the results indicate that the adopted settings $B = 30$, $\mu = \lambda = 0.1$, and $m_{\text{error}} = m_{\text{sem}} = 0.3$ lie in a stable high-performance region on both APIGraph and AndroZoo. The relatively small variations around the default settings further suggest that GRAD is not highly sensitive to moderate changes in these boundary-aware retraining hyperparameters.

Table A4. Sensitivity of GRAD to boundary-aware retraining hyperparameters. Results are reported as mean ± standard deviation over five independent runs. The default settings are highlighted in bold.

Parameter	Setting	APIGraph F1	AndroZoo F1
B	10	0.946 ± 0.002	0.884 ± 0.003
	20	0.948 ± 0.002	0.887 ± 0.003
	30	$\mathbf{0.950 \pm 0.001}$	$\mathbf{0.890 \pm 0.002}$
	40	0.949 ± 0.002	0.888 ± 0.003
	50	0.947 ± 0.002	0.885 ± 0.004
$\mu = \lambda$	0.05	0.947 ± 0.002	0.886 ± 0.003
	0.10	$\mathbf{0.950 \pm 0.001}$	$\mathbf{0.890 \pm 0.002}$
	0.20	0.949 ± 0.002	0.888 ± 0.003
	0.50	0.945 ± 0.003	0.883 ± 0.004
	1.00	0.941 ± 0.003	0.878 ± 0.005
$m_{\text{error}} = m_{\text{sem}}$	0.10	0.946 ± 0.002	0.884 ± 0.003
	0.20	0.948 ± 0.002	0.888 ± 0.003
	0.30	$\mathbf{0.950 \pm 0.001}$	$\mathbf{0.890 \pm 0.002}$
	0.40	0.948 ± 0.002	0.887 ± 0.003
	0.50	0.945 ± 0.003	0.883 ± 0.004

b. Allocation between FN and FP risk candidates. To examine the allocation of the geometry-aware annotation budget, we keep the total number of geometry-ranked candidates fixed at $\beta_{\text{FN}} + \beta_{\text{FP}} = 60$ and vary only its allocation between the two risk channels. Specifically, we evaluate $(\beta_{\text{FN}}, \beta_{\text{FP}}) \in \{(15, 45), (20, 40), (30, 30), (40, 20), (45, 15)\}$. Because the total geometry-ranking budget remains unchanged, differences in performance can be attributed to the relative emphasis placed on the two risk channels rather than to a change in the overall annotation cost.

As shown in Table A5, changing the relative allocation between the two geometry-aware risk channels produces the expected trade-off. Allocating more candidates to the FN channel generally reduces FNR, but simultaneously leaves fewer samples for FP-oriented risk identification and therefore leads to a moderate increase in FPR. Conversely, allocating more candidates to the FP channel improves false-positive coverage at the cost of a higher FNR. The balanced setting $\beta_{\text{FN}} = \beta_{\text{FP}} = 30$ achieves the strongest overall F1 performance on both datasets while maintaining competitive FNR and FPR. These results indicate that the adopted allocation does not simply optimize one error type at the expense of the other, but provides a favorable balance between the two complementary risk channels. Moreover, because only 60 of the total annotation budget $\beta = 200$ is assigned to geometry-ranked candidates, the remaining 140 samples are still available for uncertainty-based selection.

Taken together, these results show that GRAD is reasonably robust to the examined method-specific hyperparameters and that the adopted FN/FP allocation provides a stable trade-off between the two complementary error risks under a fixed annotation budget.

c. Tangent-space dimension. The tangent-space dimension r determines the number of principal directions retained when approximating the local benign geometry. To examine whether the geometry-aware ranking depends strongly on this parameter, we vary $r \in \{3, 4, 5, 6, 7\}$ while keeping all other experimental settings fixed at their default values. The setting $r = 5$ corresponds to the default configuration used in the main experiments.

This controlled experiment isolates the effect of the tangent-space dimension from other components of GRAD.

Table A5. Sensitivity to the allocation between false-negative and false-positive risk candidates under a fixed geometry-ranking budget of $\beta_{FN} + \beta_{FP} = 60$. Results are reported as mean ± standard deviation over five independent runs. The adopted allocation is highlighted in bold.

Dataset	(β_{FN}, β_{FP})	FNR	FPR	F1
APIGraph	$(15, 45)$	0.0785 ± 0.0031	0.0027 ± 0.0003	0.946 ± 0.002
	$(20, 40)$	0.0751 ± 0.0028	0.0029 ± 0.0003	0.948 ± 0.002
	$(\mathbf{30, 30})$	$\mathbf{0.0712 \pm 0.0024}$	$\mathbf{0.0033 \pm 0.0003}$	$\mathbf{0.950 \pm 0.001}$
	$(40, 20)$	0.0696 ± 0.0027	0.0038 ± 0.0004	0.949 ± 0.002
	$(45, 15)$	0.0689 ± 0.0030	0.0043 ± 0.0004	0.947 ± 0.002
AndroZoo	$(15, 45)$	0.2055 ± 0.0058	0.0008 ± 0.0002	0.882 ± 0.004
	$(20, 40)$	0.1974 ± 0.0051	0.0009 ± 0.0002	0.886 ± 0.003
	$(\mathbf{30, 30})$	$\mathbf{0.1889 \pm 0.0045}$	$\mathbf{0.0011 \pm 0.0002}$	$\mathbf{0.890 \pm 0.002}$
	$(40, 20)$	0.1831 ± 0.0048	0.0015 ± 0.0003	0.889 ± 0.003
	$(45, 15)$	0.1798 ± 0.0052	0.0019 ± 0.0003	0.886 ± 0.004

As shown in Table A6, the performance of GRAD remains stable over the examined tangent-space dimensions. On APIGraph, the mean F1 varies only within a narrow range from approximately 0.946 to 0.950, while on AndroZoo it remains between approximately 0.885 and 0.890. In particular, the settings $r = 4$, $r = 5$, and $r = 6$ produce very similar performance on both datasets, indicating a relatively broad stable operating region around the adopted setting. A smaller tangent dimension may retain insufficient local variation when approximating the benign manifold, causing some normal benign variation to be treated as residual deviation. In contrast, an excessively large tangent dimension may incorporate less informative directions into the local tangent space and consequently weaken the discriminative effect of the residual component. The adopted setting $r = 5$ provides a balanced local approximation, while the relatively small performance variations across neighboring settings show that GRAD is not highly sensitive to the precise choice of the tangent-space dimension.

Table A6. Sensitivity of GRAD to the tangent-space dimension r.

Tangent Dimension r	APIGraph F1	AndroZoo F1
3	0.946 ± 0.002	0.885 ± 0.003
4	0.949 ± 0.002	0.889 ± 0.002
5	$\mathbf{0.950 \pm 0.001}$	$\mathbf{0.890 \pm 0.002}$
6	0.949 ± 0.002	0.889 ± 0.002
7	0.947 ± 0.002	0.886 ± 0.003

Bold indicates the default hyperparameter setting used in the main experiments.

Appendix D.3. Sensitivity to the LLMS Temporal Threshold

To examine the influence of the temporal threshold used by LLMS, we conduct an additional sensitivity analysis using several fractions of the evaluation horizon. The default 1/6 threshold was chosen as a normalized temporal criterion rather than a fixed absolute duration, so that the definition of a long-lived sample reflects the same relative portion of the evaluation horizon on datasets with substantially different temporal spans. Specifically, we consider threshold ratios of 1/12, 1/8, 1/6, and 1/4. For APIGraph, whose evaluation horizon is 72 months, these settings correspond to 6, 9, 12, and 18 months, respectively. For AndroZoo, whose evaluation horizon is 24 months, they correspond to 2, 3, 4, and 6 months. The model predictions and all other experimental settings remain

unchanged; only the threshold used to compute LLMS is varied. This isolates the effect of the LLMS definition itself from changes in model training or adaptation.

As shown in Tables A7 and A8, the absolute LLMS values vary systematically with the temporal threshold. Shorter thresholds generally yield larger LLMS values because the required temporal persistence condition is easier to satisfy, whereas longer thresholds impose a stricter criterion and consequently reduce the reported LLMS values. Despite these changes in absolute magnitude, the overall comparative pattern among the evaluated methods remains largely stable across the examined threshold ratios on both APIGraph and AndroZoo. In particular, methods that exhibit relatively high or low LLMS under the default 1/6 setting generally remain in similar performance regions when the threshold is shortened or extended. This observation suggests that the overall conclusions drawn from LLMS are not strongly dependent on the specific 1/6 threshold, although the absolute LLMS values naturally vary with the strictness of the temporal criterion.

Table A7. Sensitivity of LLMS to different temporal threshold ratios on APIGraph. The default threshold ratio of 1/6 is highlighted in bold.

Method	1/12	1/8	1/6	1/4
Ours	764	704	**644**	515
UNCER	889	812	**736**	591
CORESET	1435	1294	**1172**	964
CADE	1786	1634	**1505**	1220
TRANS	1463	1313	**1200**	993
HCL	954	877	**803**	661

Table A8. Sensitivity of LLMS to different temporal threshold ratios on AndroZoo. The default threshold ratio of 1/6 is highlighted in bold.

Method	1/12	1/8	1/6	1/4
Ours	578	520	**479**	387
UNCER	260	241	**221**	183
CORESET	581	519	**478**	399
CADE	1285	1151	**1051**	840
TRANS	599	536	**445**	421
HCL	621	574	**477**	432

Appendix D.4. Detailed Analysis of Backward Transfer

To complement the average BWT analysis, we further examine backward transfer at a finer granularity. For each current adaptation cycle r and historical cycle $i < r$, we define the pairwise backward-transfer difference as

$$\Delta_{r,i} = \mathrm{F1}(\theta_r, \mathcal{T}^i) - \mathrm{F1}(\theta_i, \mathcal{T}^i).$$

A negative value of $\Delta_{r,i}$ indicates that the current model performs worse on historical cycle i than the model obtained when that cycle was originally learned. In addition to the average cycle-level BWT, we report the minimum cycle-level BWT and the minimum pairwise backward-transfer difference. These two statistics provide a finer-grained characterization of the largest degradation observed during temporal adaptation, even when the average BWT remains positive.

All evaluated methods achieve positive average BWT values on both datasets, indicating that subsequent adaptation generally preserves or improves performance on previously observed cycles. However, the minimum BWT results confirm that positive average backward transfer does not necessarily imply uniformly positive behavior at every

adaptation cycle. On APIGraph, GRAD achieves an average BWT of 0.0048, while its minimum cycle-level BWT is only −0.0015. On AndroZoo, GRAD maintains a positive average BWT of 0.0200, while its minimum BWT is −0.0025. Although these values are negative, their magnitudes are very small, indicating that the corresponding cycle-level degradation is minor relative to the overall positive backward-transfer tendency.

Table A9. Detailed backward-transfer analysis on APIGraph and AndroZoo, including the average cycle-level BWT, minimum cycle-level BWT, and minimum pairwise backward-transfer difference.

Dataset	Method	Avg. BWT	Min. BWT	Min. Pair Δ
APIGraph	CADE	0.0098	−0.0050	−0.0475
	HCL	0.0096	−0.0009	−0.0233
	TRANS	0.0070	−0.0002	−0.0615
	UNCER	0.0115	−0.0002	−0.0298
	GRAD	0.0048	−0.0015	−0.0208
AndroZoo	CORESET	0.0150	0.0000	−0.0455
	HCL	0.0135	−0.0092	−0.0130
	TRANS	0.0227	0.0009	−0.0126
	UNCER	0.0027	0.0008	−0.0069
	GRAD	0.0200	−0.0025	−0.0084

The minimum pairwise transfer provides a finer-grained view of the worst degradation observed on an individual historical cycle. On APIGraph, the minimum pairwise F1 difference of GRAD is −0.0208, compared with −0.0475 for CADE, −0.0233 for HCL, −0.0615 for TRANS, and −0.0298 for UNCER. Thus, although GRAD does not completely avoid negative transfer on every previous cycle, its worst observed decrease remains limited in magnitude. On AndroZoo, the minimum pairwise difference of GRAD is −0.0084, compared with −0.0455 for CORESET, −0.0130 for HCL, −0.0126 for TRANS, and −0.0069 for UNCER. Although UNCER exhibits a slightly smaller worst-case decrease on this dataset, the negative pairwise transfer of GRAD remains small in absolute magnitude.

Taken together, these results show that isolated negative backward transfer can occur even when the average BWT remains positive, but the observed degradation of GRAD is limited in magnitude. In particular, the minimum cycle-level BWT values are close to zero on both datasets, and the worst pairwise F1 decreases are also relatively small. Therefore, the localized negative transfer has only a limited effect on the overall historical performance retention of GRAD under the evaluated temporal settings. This more detailed analysis complements the average BWT and provides a more complete characterization of forgetting during temporal adaptation.

The remaining localized degradation also motivates future investigation of finer-grained historical-retention mechanisms. Future work could dynamically identify previous cycles that are more susceptible to forgetting and selectively reinforce them during subsequent adaptation. Such extensions may further reduce these small worst-case degradations while preserving the ability of the model to adapt to newly emerging malware distributions.

References

1. Gopinath, M.; Sethuraman, S.C. A comprehensive survey on deep learning based malware detection techniques. *Comput. Sci. Rev.* **2023**, *47*, 100529. [CrossRef]
2. Willems, E. Android Malware: An Underestimated Problem? 2022. Available online: https://www.gcst.ae/android-malware-an-underestimated-problem/ (accessed on 25 September 2026).
3. AppBrainStatistics. Number of Android Apps on Google Play (Nov 2024). 2024. Available online: https://www.bankmycell.com/blog/how-many-android-users-are-there (accessed on 25 September 2026).

4. Statcounter. Mobile Operating System Market Share Worldwide. 2024. Available online: https://gs.statcounter.com/os-market-share/ (accessed on 25 September 2026).
5. Gama, J.; Žliobaitė, I.; Bifet, A.; Pechenizkiy, M.; Bouchachia, A. A survey on concept drift adaptation. *ACM Comput. Surv. (CSUR)* **2014**, *46*, 44. [CrossRef] [PubMed]
6. Sato, D.M.V.; De Freitas, S.C.; Barddal, J.P.; Scalabrin, E.E. A survey on concept drift in process mining. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 189. [CrossRef]
7. Meng, Z.; Zhang, J.; Guo, J.; Wang, W.; Huang, W.; Cui, J.; Zhong, H.; Xiong, Y. Detecting Android Malware by Visualizing App Behaviors From Multiple Complementary Views. *IEEE Trans. Inf. Forensics Secur.* **2025**, *20*, 2915–2929. [CrossRef]
8. Hu, Q.; Wang, W.; Song, H.; Guo, S.; Zhang, J.; Zhang, S. ASDroid: Resisting Evolving Android Malware with API Clusters Derived from Source Code. *IEEE Trans. Inf. Forensics Secur.* **2025**, *20*, 1822–1835. [CrossRef]
9. Kim, T.; Kang, B.; Rho, M.; Sezer, S.; Im, E.G. A Multimodal Deep Learning Method for Android Malware Detection Using Various Features. *IEEE Trans. Inf. Forensics Secur.* **2019**, *14*, 773–788. [CrossRef]
10. Park, C.H.; Kang, Y. An active learning method for data streams with concept drift. In *Proceedings of the 2016 IEEE International Conference on Big Data (Big Data)*; IEEE: Piscataway, NJ, USA, 2016; pp. 746–752.
11. Malekghaini, N.; Akbari, E.; Salahuddin, M.A.; Limam, N.; Boutaba, R.; Mathieu, B.; Moteau, S.; Tuffin, S. Deep learning for encrypted traffic classification in the face of data drift: An empirical study. *Comput. Netw.* **2023**, *225*, 109648. [CrossRef]
12. Lu, J.; Liu, A.; Dong, F.; Gu, F.; Gama, J.; Zhang, G. Learning under concept drift: A review. *IEEE Trans. Knowl. Data Eng.* **2018**, *31*, 2346–2363. [CrossRef]
13. Bayram, F.; Ahmed, B.S.; Kassler, A. From concept drift to model degradation: An overview on performance-aware drift detectors. *Knowl.-Based Syst.* **2022**, *245*, 108632. [CrossRef]
14. Karimian, M.; Beigy, H. Concept drift handling: A domain adaptation perspective. *Expert Syst. Appl.* **2023**, *224*, 119946. [CrossRef]
15. Yu, H.; Liu, W.; Lu, J.; Wen, Y.; Luo, X.; Zhang, G. Detecting group concept drift from multiple data streams. *Pattern Recognit.* **2023**, *134*, 109113. [CrossRef]
16. Yu, E.; Lu, J.; Zhang, B.; Zhang, G. Online boosting adaptive learning under concept drift for multistream classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*; AAAI Press: Washington, DC, USA, 2024; Volume 38, pp. 16522–16530.
17. Chen, Y.; Ding, Z.; Wagner, D. Continuous Learning for Android Malware Detection. In Proceedings of the 32nd USENIX Security Symposium (USENIX Security 23), Anaheim, CA, USA, 9–11 August 2023; pp. 1127–1144.
18. Liu, W.; Zhang, H.; Ding, Z.; Liu, Q.; Zhu, C. A comprehensive active learning method for multiclass imbalanced data streams with concept drift. *Knowl.-Based Syst.* **2021**, *215*, 106778. [CrossRef]
19. Krawczyk, B.; Pfahringer, B.; Woźniak, M. Combining active learning with concept drift detection for data stream mining. In *Proceedings of the 2018 IEEE International Conference on Big Data*; IEEE: Piscataway, NJ, USA, 2018; pp. 2239–2244.
20. Ren, P.; Xiao, Y.; Chang, X.; Huang, P.Y.; Li, Z.; Gupta, B.B.; Chen, X.; Wang, X. A survey of deep active learning. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 180. [CrossRef]
21. Zhang, X.; Zhang, Y.; Zhong, M.; Ding, D.; Cao, Y.; Zhang, Y.; Zhang, M.; Yang, M. 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*; Association for Computing Machinery: New York, NY, USA, 2020; pp. 757–770.
22. Allix, K.; Bissyandé, T.F.; Klein, J.; Le Traon, Y. AndroZoo: Collecting Millions of Android Apps for the Research Community. In *Proceedings of the 13th International Conference on Mining Software Repositories*; IEEE: Piscataway, NJ, USA, 2016; pp. 468–471.
23. Gawlikowski, J.; Tassi, C.R.N.; Ali, M.; Lee, J.; Humt, M.; Feng, J.; Kruspe, A.; Triebel, R.; Jung, P.; Roscher, R.; et al. A survey of uncertainty in deep neural networks. *Artif. Intell. Rev.* **2023**, *56*, 1513–1589. [CrossRef]
24. Sener, O.; Savarese, S. Active learning for convolutional neural networks: A core-set approach. *arXiv* **2017**, arXiv:1708.00489.
25. Yang, L.; Guo, W.; Hao, Q.; Ciptadi, A.; Ahmadzadeh, A.; Xing, X.; Wang, G. CADE: Detecting and Explaining Concept Drift Samples for Security Applications. In *Proceedings of the 30th USENIX Security Symposium (USENIX Security 21)*; USENIX Association: Berkeley, CA, USA, 2021; pp. 2327–2344.
26. Barbero, F.; Pendlebury, F.; Pierazzi, F.; Cavallaro, L. Transcending TRANSCEND: Revisiting Malware Classification in the Presence of Concept Drift. In *Proceedings of the 2022 IEEE Symposium on Security and Privacy (SP)*; IEEE: Piscataway, NJ, USA, 2022; pp. 805–823. [CrossRef]
27. Roh, E.; Kaya, Y.; Kruegel, C.; Vigna, G.; Hong, S. MADCAT: Combating Malware Detection Under Concept Drift with Test-Time Adaptation. *arXiv* **2025**, arXiv:2505.18734. [CrossRef]
28. Chai, Y.; Chen, X.; Qiu, J.; Du, L.; Xiao, Y.; Feng, Q.; Ji, S.; Tian, Z. MalFSCIL: A Few-Shot Class-Incremental Learning Approach for Malware Detection. *IEEE Trans. Inf. Forensics Secur.* **2025**, *20*, 2999–3014. [CrossRef]
29. Arp, D.; Quiring, E.; Pendlebury, F.; Warnecke, A.; Pierazzi, F.; Wressnegger, C.; Cavallaro, L.; Rieck, K. Dos and don'ts of machine learning in computer security. In *Proceedings of the 31st USENIX Security Symposium (USENIX Security 22)*; USENIX Association: Berkeley, CA, USA, 2022; pp. 3971–3988.

30. Pendlebury, F.; Pierazzi, F.; Jordaney, R.; Kinder, J.; Cavallaro, L. TESSERACT: Eliminating experimental bias in malware classification across space and time. In *Proceedings of the 28th USENIX Security Symposium (USENIX Security 19)*; USENIX Association: Berkeley, CA, USA, 2019; pp. 729–746.
31. Chen, Z.; Zhang, Z.; Kan, Z.; Yang, L.; Cortellazzi, J.; Pendlebury, F.; Pierazzi, F.; Cavallaro, L.; Wang, G. Is It Overkill? Analyzing Feature-Space Concept Drift in Malware Detectors. In *Proceedings of the 2023 IEEE Security and Privacy Workshops (SPW)*; IEEE: Piscataway, NJ, USA, 2023; pp. 21–28. [CrossRef]
32. Jordaney, R.; Sharad, K.; Dash, S.K.; Wang, Z.; Papini, D.; Nouretdinov, I.; Cavallaro, L. Transcend: Detecting concept drift in malware classification models. In *Proceedings of the 26th USENIX Security Symposium (USENIX Security 17)*; USENIX Association: Berkeley, CA, USA, 2017; pp. 625–642.
33. Narayanan, A.; Chandramohan, M.; Chen, L.; Liu, Y. Context-Aware, Adaptive, and Scalable Android Malware Detection Through Online Learning. *IEEE Trans. Emerg. Top. Comput. Intell.* **2017**, *1*, 157–175. [CrossRef]
34. Narayanan, A.; Yang, L.; Chen, L.; Jinliang, L. Adaptive and scalable android malware detection through online learning. In *Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN)*; IEEE: Piscataway, NJ, USA, 2016; pp. 2484–2491.
35. Xu, K.; Li, Y.; Deng, R.; Chen, K.; Xu, J. Droidevolver: Self-evolving android malware detection system. In *Proceedings of the 2019 IEEE European Symposium on Security and Privacy (EuroS&P)*; IEEE: Piscataway, NJ, USA, 2019; pp. 47–62.
36. Botacin, M.; Gomes, H. Towards more realistic evaluations: The impact of label delays in malware detection pipelines. *Comput. Secur.* **2025**, *148*, 104122. [CrossRef]
37. Kan, Z.; Pendlebury, F.; Pierazzi, F.; Cavallaro, L. Investigating labelless drift adaptation for malware detection. In *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security*; Association for Computing Machinery: New York, NY, USA, 2021; pp. 123–134.
38. Alam, M.T.; Fieblinger, R.; Mahara, A.; Rastogi, N. MORPH: Towards Automated Concept Drift Adaptation for Malware Detection. *arXiv* **2024**, arXiv:2401.12790 . [CrossRef]
39. Liu, Z.; Wang, R.; Peng, B.; Qiu, L.; Gan, Q.; Wang, C.; Zhang, W. LDCDroid: Learning data drift characteristics for handling the model aging problem in Android malware detection. *Comput. Secur.* **2025**, *150*, 104294. [CrossRef]
40. He, Y.; Lei, J.; Qin, Z.; Ren, K.; Chen, C. Combating Concept Drift with Explanatory Detection and Adaptation for Android Malware Classification. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*; Association for Computing Machinery: New York, NY, USA, 2025; pp. 978–992. [CrossRef]
41. McFadden, S.; Foley, M.; D'Onghia, M.; Hicks, C.; Mavroudis, V.; Paoletti, N.; Pierazzi, F. DRMD: Deep Reinforcement Learning for Malware Detection under Concept Drift. *AAAI Conf. Artif. Intell.* **2026**, *40*, 854–862. [CrossRef]
42. Pan, Y.; Zhao, L.; Leng, T.; Luo, Z.; Cai, L.; Yu, A.; Meng, D. DriftTrace: Combating Concept Drift in Security Applications Through Detection and Explanation. *IEEE Trans. Inf. Forensics Secur.* **2026**, *21*, 1957–1972. [CrossRef]
43. Han, D.; Wang, Z.; Chen, W.; Wang, K.; Yu, R.; Wang, S.; Zhang, H.; Wang, Z.; Jin, M.; Yang, J.; et al. Anomaly Detection in the Open World: Normality Shift Detection, Explanation, and Adaptation. In *Proceedings of the 30th Annual Network and Distributed System Security Symposium (NDSS)*; Internet Society: Reston, VA, USA, 2023.
44. Žliobaitė, I.; Bifet, A.; Pfahringer, B.; Holmes, G. Active learning with drifting streaming data. *IEEE Trans. Neural Netw. Learn. Syst.* **2013**, *25*, 27–39. [CrossRef] [PubMed]
45. Wang, P.; Tang, Z.; Wang, J. A novel few-shot malware classification approach for unknown family recognition with multi-prototype modeling. *Comput. Secur.* **2021**, *106*, 102273. [CrossRef]
46. Snell, J.; Swersky, K.; Zemel, R. Prototypical Networks for Few-shot Learning. In *Proceedings of the Advances in Neural Information Processing Systems*; Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2017; Volume 30.
47. Angioni, D.; Demetrio, L.; Pintor, M.; Oneto, L.; Anguita, D.; Biggio, B.; Roli, F. Robustness-Congruent Adversarial Training for Secure Machine Learning Model Updates. *IEEE Trans. Pattern Anal. Mach. Intell.* **2025**, *47*, 7457–7469. [CrossRef] [PubMed]
48. Abusnaina, A.; Anwar, A.; Saad, M.; Alabduljabbar, A.; Jang, R.; Salem, S.; Mohaisen, D. One step forward, two steps back: ML-based malware detection under concept drift. *Computing* **2025**, *107*, 207. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.