Manipulating Uncertainty Ranking with Poisoning Attacks for Adversarial Concept Drift Adaptation

Anonymous Author(s)

Abstract

Machine learning models have achieved remarkable success across various domains. Unfortunately, due to the continuously changing distribution of test data, deployed models quickly become outdated—a phenomenon known as concept drift. To address this, Concept Drift Adaptation methods based on Active Learning (CDA-AL) have been proposed to enhance model performance. Despite the growing interest in CDA-AL, limited research has explored its security risks. Therefore, our research focuses on the robustness of CDA-AL, with particular emphasis on its vulnerability to data poisoning attacks. Our attack goal is to cause the victim model to misclassify specific concept drift samples (i.e., attack targets) through data poisoning attack. We propose a novel poisoning attack method that manipulates the uncertainty ranking of test samples by generating poisoned samples, leading to the misallocation of the victim model's labeling budget. The inability to effectively utilize the labeling budget prevents the victim model from learning the attack target properly, leading to misclassification. Furthermore, our attack leverages naturally occurring concept drift samples to generate poisoned samples, significantly reducing the attack cost.

The attack effectiveness is evaluated across four concept drift datasets, achieving an average attack success rate of **over 90%.** In addition, we conduct a more in-depth analysis on the malware dataset, as the APIGraph dataset has the longest concept drift time span (spanning seven years) and includes 300,000 samples. On this dataset, we examined the variation in attack success rates under constrained attacker capabilities, with results demonstrating an attack success rate exceeding 80%. We further observe that the proposed attack achieves high success rates under various sample selection strategies, labeling budgets, and data collection capabilities. Our analysis of four existing countermeasuers show they fail to effectively counter our attack, highlighting the need for robust solutions. As a step forward, we propose an adaptive sample filtering method based on intra-cluster distance, which effectively mitigates our attack.

CCS Concepts

• **Security and privacy**; • **Computing methodologies** → *Artificial intelligence*;

Keywords

Concept Drift Adaptation, poisoning Attack, multi-Attacker, defenses

ACM Reference Format:
Anonymous Author(s). 2025. Manipulating Uncertainty Ranking with Poisoning Attacks for Adversarial Concept Drift Adaptation. In *Proceedings of ACM Conference on Computer and Communications Security (ACM CCS)* *(ACM CCS).* ACM, New York, NY, USA, 18 pages. https://doi.org/XXXXXXX.XXXXXXX

ACM CCS, Taipei, Taiwan
2025. ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Machine learning models are widely applied across various practical domains [9, 14, 39, 89]. However, as the gap between the test and training dataset distributions widens over time, the model's performance on the test dataset declines [33, 45, 52, 57]. This phenomenon is known as concept drift [6, 7, 41, 59], which poses severe threats in critical domains, such as industrial risk analysis and malware detection [31, 49, 68, 72, 95]. To mitigate these risks, addressing the challenge of concept drift has become a hot topic of current research [24, 26, 37, 84–86]. One widely used approach is Concept Drift Adaptation based on Active Learning (CDA-AL) [6, 12, 19, 83]. The core idea of CDA-AL is to select the most informative unlabeled samples from the test dataset based on sample selection strategy. A strong baseline is to select samples for which the classifier is most uncertain [12], treating them as informative samples. These high-uncertainty samples, referred to as concept drift samples, are then manually labeled and incorporated into the training dataset for model retraining. The retraining process enables the model to learn from uncertain concept drift samples. Therefore, the retrained model will achieve higher prediction accuracy on the test datasets. Among the steps described above, manual analysis of high-uncertainty samples incurs the highest cost, which is quantified as the labeling budget [21, 62, 88].

Unfortunately, prior research by Korycki et al. compromised the performance of concept drift adaptation methods on test datasets by injecting poisoned samples into the victim model's training dataset [30]. However, their approach relies on the strong assumption that attackers can arbitrarily inject poisoned samples into the victim model's training dataset, which overestimates the adversary's capabilities [75]. In practice, in sensitive scenarios, attackers usually only can influence the test dataset and cannot directly control the training dataset. Particularly in CDA-AL methods [12, 41, 54, 96], only samples with high uncertainty can be included in the training dataset after being analyzed manually. Therefore, the risks of poisoning attacks in machine learning paradigms like CDA-AL, where poisoned samples cannot be freely added to the training dataset, have not been fully understood.

In this paper, we propose a novel Poisoning Attack against Concept Drift Adaptation based on active learning (PACDA). Our core idea is to use poisoned samples to manipulate the uncertainty ranking of samples in the unlabeled test dataset. Once the uncertainty ranking is altered, the victim model's limited labeling budget is misused. Since the model's performance change under the CDA-AL method is determined by the sample information included in the labeling budget, the misuse of the labeling budget results in a decline in concept drift adaptation performance. This type of resource-exhaustion attack is highly dangerous because previous research [97] has shown that even if domain experts detect the poisoning attack later, the labeling budget and time have already been spent, further draining the already scarce resources. Moreover, existing concept drift adaptation attacks reduce the overall performance of the victim model and lack attack stealth [30]. In contrast, our PACDA attack selectively consumes only a portion of the labeling budget specifically for the attack target. This approach maintains the overall performance stability of the victim model, thereby enhancing the stealth of the attack.

The key challenge of the PACDA attack lies in generating high-uncertainty poisoned samples that can be selected by the victim model, thereby exhausting its labeling budget. First, we compute the labeling budget consumption required for the attack target. Next, we propose using constrained search to select high-uncertainty concept drift samples from the test dataset as poisoning seed samples. Based on the poisoning seed samples, we introduce a poisoned sample generation strategy that utilizes problem space perturbation and uncertainty attribution to exhaust the victim model's labeling budget. As a result, the victim model fails to learn from samples related to the attack target, leading to sustained misclassification of the attack target.

We conduct a comprehensive evaluation and find that existing active learning-based concept drift adaptation methods [12] are vulnerable to adversarial concept drift caused by poisoning attacks. The attack effectiveness is evaluated by using seven experimental baselines, built by combining seven models across four datasets: APIGraph [90], BODMAS [82], MNIST [78], and SPAM [27]. Our proposed attack method reduces the model's adaptation performance by consuming its labeling budget, leading to misclassification of attack targets. For example, during a 72-month test involving over 1,000 attack targets, we achieved an average attack success rate of 88%. To better understand adversarial concept drift, we further analyze key factors affecting attack success, including sample selection strategies in active learning, the victim model's labeling capacity, and the attacker's data collection ability. The attack success rate exceeds 80% across all configurations. We also evaluate four defense methods, including Activation Clustering [10] and Data-Free Pruning [34], and other widely adopted techniques for defending against poisoning attacks. Experimental results show that these methods struggle to effectively distinguish between poisoned and clean samples. These findings highlight the urgent need for new defense mechanisms against PACDA attacks. To address this limitation, we propose a poisoned sample filtering method based on intra-cluster distance to mitigate the effects of our PACDA attack, resulting in a reduction of the attack success rate by nearly 40%.

Our contributions are summarized as follows:

• Our proposed PACDA attack is the first to successfully launch poisoning attacks against concept drift adaptation methods based on active learning. By manipulating the uncertainty ranking of test samples, it demonstrates how valuable labeling budgets can be misallocated. This ultimately leads to the sustained misclassification of the attack target by the victim model throughout the concept drift adaptation process.

• We design a method for generating high-uncertainty poisoned samples. We find that naturally occurring concept drift samples can assist attackers in generating poisoned examples. By combining problem space perturbations and uncertainty-based feature attribution, our method ensures that a sufficient number of poisoned samples are used to consume the label budget of the victim model.

• We evaluate the effectiveness of the PACDA attack against existing concept drift adaptation methods across diverse datasets, model architectures, and sample selection strategies. Additionally, we analyze the impact of various factors on the attack effectiveness. Experimental results demonstrate that our attack achieves a high attack success rate under all evaluated settings.

• We identify critical limitations in existing defence mechanisms against PACDA attack. To address these gaps, we propose a poisoned sample filtering method based on cluster-adaptive distance, which can reduce the attack success rate.

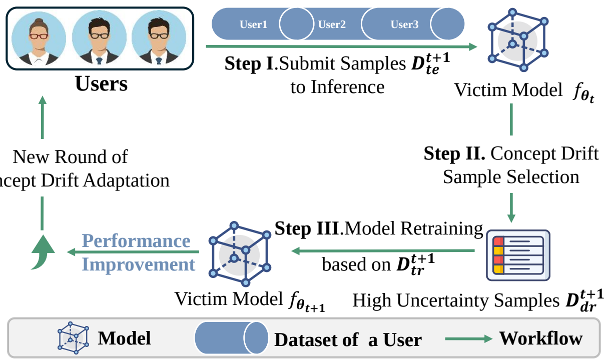
User1
User2
User3
Users
Step I.Submit Samples D_{te}^{t+1} to Inference
Victim Model f_{θ_t}
New Round of Concept Drift Adaptation
Step II. Concept Drift Sample Selection
Performance Improvement
Step III.Model Retraining based on D_{tr}^{t+1}
Victim Model $f_{\theta_{t+1}}$
High Uncertainty Samples D_{dr}^{t+1}
Model
Dataset of a User
Workflow

Figure 1: Concept drift adaptation based active learning

2 Preliminaries

In this section, we introduce the concept drift adaptation process based on active learning [12],the classification of existing data poisoning attacks [75] and the Shapley additive explanations used for uncertainty feature attribution.

2.1 Concept Drift Adaptation

Concept drift adaptation based on active learning is a paradigm that optimizes the model by identifying samples with high uncertainty within the test dataset [12, 19]. The detailed process is illustrated in Figure 1. Throughout this paper, we refer to the concept drift adaptation model as the victim model for consistency. Victim model f_{θ_t} is trained on a training dataset D_{tr}^t at time t, where θ_t represents the model parameters. Unlabeled test dataset D_{te}^{t+1} represents all the test data collected by the victim model between time t and $t+1$. The concept drift adaptation process typically involves three steps:

• Step-I: Model f_{θ_t} performs inference on the input $\mathbf{x}_i \in D_{te}^{t+1}$ and obtain the classification confidence vector $\mathbf{c}_i = f_{\theta_t}(\mathbf{x}_i)$. The different dimensions of $\mathbf{c}_i$ represent the model's confidence that the input $\mathbf{x}_i$ belongs to a specific sample label. The label with the highest confidence is selected as the predicted label $\overline{y}_i$ on the input $\mathbf{x}_i$. It is worth noting that $\overline{y}_i$ is a predicted label, not necessarily the ground truth label y_i of the sample $\mathbf{x}_i$. Each sample in the test dataset D_{te}^{t+1} undergoes the aforementioned operations.

• Step-II: Based on the classification confidence vector $\mathbf{c}_i$ obtained in Step-I, the uncertainty score $\mathbf{u}_i$ of the sample $\mathbf{x}_i$ can be quantified as $\mathbf{u}_i = uncer(f_{\theta_t}(\mathbf{x}_i))$. $\mathbf{u}_i$ represents the model's confidence in the predicted label $\overline{y}_i$, so the higher the uncertainty score $\mathbf{u}_i$ of the sample $\mathbf{x}_i$, the higher the degree of uncertainty [19]. In the following descriptions, we define the uncertainty quantification function uniformly as $uncer()$. There are various methods for calculating uncertainty [12]. For details, see Section 5.3.

• Step-III: High-uncertainty samples are selected from the test dataset D_{te}^{t+1} as concept drift samples D_{dr}^{t+1}. The size of D_{dr}^{t+1} is determined by the manual labeling capacity, referred to as the label budget β. Then, the concept drift sample dataset D_{dr}^{t+1} is manually labeled to get the ground truth label and added to the original training dataset D_{tr}^{t} to obtain a new training dataset $D_{tr}^{t+1} = D_{tr}^{t} \cup D_{dr}^{t+1}$. Subsequently, the victim model f_{θ_t} is retrained on the updated training dataset D_{tr}^{t+1}, resulting in a new victim model $f_{\theta_{t+1}}$, as shown in Equation 1.

$$f_{\theta_{t+1}} = \arg\min_{\theta_{t+1}} \sum_{\mathbf{x}_i \in D_{tr}^{t+1}} \mathcal{L}\left(f_{\theta_t}(\mathbf{x}_i), y_i\right) \tag{1}$$

Note that in this paper, we refer to the samples selected by the sample selection strategy as the concept drift sample set, meaning that this set may contain concept drift samples but does not imply that all samples in the set exhibit concept drift. Because the uncertain examples may not be due to concept drift, even if the distribution is stable over time there may still be samples close to the decision boundary. Since this issue falls within the field of concept drift detection and is beyond the scope of concept drift adaptation, we do not distinguish it in the subsequent discussion.

2.2 Data Poisoning Attacks

In data poisoning attacks, the attacker constructs a poisoned dataset, D_p, to degrade the performance of the victim model. The most common approach is to inject the poisoned dataset into the victim model's training dataset, as shown in Equation 2. We adopt the classification method proposed by Tian et al. [67] and Wang et al. [75], which divides data poisoning attacks into targeted and untargeted attacks.

$$f_{\theta_{t+1}} = \arg\min_{\theta_{t+1}} \left(\sum_{\mathbf{x}_i \in D_{tr}} \mathcal{L}\left(f_{\theta_t}(\mathbf{x}_i), y_i\right) + \sum_{\mathbf{x}_p \in D_p} \mathcal{L}\left(f_{\theta_t}(\mathbf{x}_p), y_p\right) \right) \tag{2}$$

The goal of untargeted poisoning attacks is to hinder the convergence of the victim model and eventually lead to denial-of-service [8, 51, 74, 87, 92]. For example, adversaries can flip labels of some samples in the training dataset to perturb the knowledge contained in the benign sample-label pairs [17, 50]. The challenge with untargeted attacks is that they aim to degrade the performance across all data [28]. Therefore, the poisoned samples must counteract the effect of the unpoisoned training dataset.

In targeted poisoning attacks, the adversarial goal is to force the victim model to produce abnormal predictions on particular inputs [16, 61, 65, 79, 80]. Samples that experience a performance degradation on the victim model after such attacks are typically referred to as attack targets. For example, the attacker may force the victim model to predict all digit "3" as "5". To launch a targeted poisoning attack, an adversary needs to inject malicious knowledge into the training dataset while keeping other knowledge unaffected [67].

2.3 Shapley Additive Explanations

Research in explainable machine learning has proposed multiple systems to interpret the predictions of complex models [2–4, 47]. SHapley Additive exPlanations (SHAP) [42, 43] leverage the Shapley value concept from cooperative game theory to explain predictions by assigning a contribution value to each feature based on its impact on the final prediction. The SHAP framework has been shown to subsume several earlier model explanation techniques [60], including LIME [58] and Integrated Gradients [66]. These model explanation frameworks help determine the importance of each feature value in the prediction of the model.

$$g(\mathbf{x}) = \phi_0 + \sum_{j=1}^{M} \phi_j x_j \tag{3}$$

To achieve this, explanation frameworks construct a surrogate linear model using the input feature vectors and output predictions, leveraging its coefficients to estimate feature importance and direction, as shown in Equation 3. $\mathbf{x}$ is the sample, x_j is the j th feature for sample $\mathbf{x}$, and ϕ_j is the contribution of feature x_j to the model's prediction. The SHAP framework stands out by providing theoretical guarantees for calculating feature contributions in a model-agnostic manner. Recent studies have extended model prediction explanations to include explanations of prediction uncertainty [76]. Therefore, we consider incorporating this method into the design of poisoning attacks against concept drift adaptation methods.

3 Threat Model

Attacker Goal. Our goal is to ensure that the attack target is consistently misclassified by the victim model throughout the CDA-AL process. All attack targets are selected from the test dataset, and we do not degrade the victim model's performance on the training dataset. Therefore, our approach can be categorized as a targeted poisoning attack. Similar to previous research [61, 65], after being retrained on the poisoned samples, the victim model will misclassify the attack target. For instance, in a binary malware detection task, our PACDA attack misclassifies target malware samples as benign. Meanwhile, the overall model performance remains stable, ensuring the stealthiness of the PACDA attack. In addition, it is important to note that in our PACDA attack, the attack target has a defined source class but does not require assignment to a specific target class. This class-agnostic misclassification design enhances the stealthiness of the attack, as it decouples the poisoned samples from any fixed target class. As a result, the victim model is more likely to attribute the resulting misclassifications to natural concept drift or inherent uncertainty, rather than to an intentional poisoning attack.

Attacker Capabilities. We assume that attackers cannot access the victim model's internal parameters or influence its training process, including manual labeling [44]. Attackers only can submit samples to the victim model for querying to obtain its outputs, such as sample uncertainty score and predicted labels. They are also assumed to have access to public data and the resources to

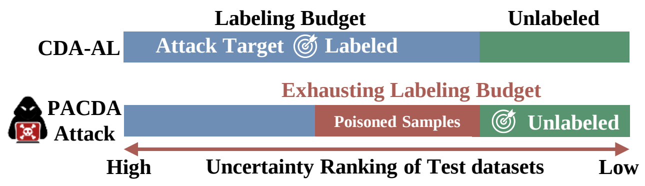

Figure 2: PACDA Attack Motivation

train surrogate models [53, 56]. In addition, consistent with existing research on active learning attacks [69], we assume that the attacker typically has access to the labeling budget settings (including the uncertainty threshold) used by the victim model. Moreover, in Section 5.3, we also conduct experimental analysis for the scenario where the attacker does not have access to the victim model's labeling budget, in order to understand how this capability affects the attack success rate.

Attack Challenge. PACDA attack faces three challenges. (1) The sample selection strategy deployed in CDA-AL restricts the arbitrary injection of poisoned samples into the training dataset. Therefore, attackers must ensure that poisoned samples are selected by the victim model. (2) The sample labeling process in active learning relies on manual analysis. Therefore, the labels of all poisoned samples in the training dataset remain correct. The attacker cannot degrade the victim model's concept drift adaptation performance by tampering with sample labels [61, 94]. (3) Existing targeted poisoning attacks are always evaluated for effectiveness on models with fixed parameters. For example, the effectiveness of backdoor attacks is evaluated on pre-trained models [11, 60, 81]. PACDA must ensure the persistence of its attack effectiveness as the victim model updates its parameters. The PACDA attack method we propose effectively addresses the first two challenges. Our approach ensures that the poisoned samples exhibit high uncertainty and the poisoned examples are correctly labeled, enhancing the natural stealth of our attack method. The final challenge is addressed through continuous attacks on the concept drift adaptation process. Additionally, we validate the persistence of the attack's effectiveness on a concept drift dataset with a seven-year timespan, as discussed in Section 5.2.1.

4 Attack Method

In this section, we first discuss the research motivation and design rationale of our PACDA attack, followed by an overview of the attack method and its details.

4.1 Motivation and Design Rationale

Motivation. As previously mentioned, our goal is to ensure that the victim model is unable to learn how to correctly classify the attack target. The key issue is how to ensure that the poisoned samples have high uncertainty. Previous research has shown that uncertainty quantification techniques can be manipulated by attackers [32]. Ledda et al. demonstrated this by employing perturbation search techniques to find the minimal perturbations that alter the uncertainty of samples [32]. But this method focuses on reducing sample uncertainty, while CDA-AL instead selects samples with high uncertainty [12]. Inspired by Yang et al. and Chen et al., who represent sample uncertainty by measuring the similarity between new samples and the training dataset [12, 83], we find that newly appearing concept drift samples always inherently exhibits high uncertainty. Because they usually have low similarity to the existing training datasets. Therefore, the core idea of the PACDA attack is to exploit the inherent high uncertainty of concept drift samples to generate poisoned samples, ensuring that these samples consume the victim model's labeling budget and prevent the attack target from being effectively learned.

Design Rationale. First, we estimate the labeling budget that the attacker needs to exhaust from the victim model based on the chosen attack target. By controlling the amount of label budget consumed, the attacker determines which knowledge the victim model is allowed to learn and which it is prevented from learning. Next, we identify the samples in the test dataset that exhibit high uncertainty. These samples are selected by the attacker as poisoning seed samples to consume the labeling budget of the victim model. To efficiently search for poisoning seed samples for different attack targets, we propose a constraint search strategy based on uncertainty ranking.

Furthermore, we find that the number of poisoned seed samples is limited and cannot fully meet the label budget consumption requirement. So we employ the poisoned samples generation method [76]. This method consists of two strategies: Strategy one is based on problem space perturbation, and Strategy two is based on Shapley additive explanations. Both strategies aim to generate poisoned samples that sufficiently consume the victim model's labeling budget, thereby causing the misclassification of the attack target. However, the two strategies are suitable for different types of attack targets. The first strategy prevents the victim model from learning specific knowledge related to the attack target by constructing poisoned samples, without introducing any incorrect knowledge about concept drift. In contrast, the second strategy not only prevents the victim model from acquiring knowledge about the attack target but also introduces misleading information about concept drift into the training process. Notably, the injected misleading information impacts the model's ability to adapt to concept drift in the test data, yet does not result in a significant overall performance drop, as reflected by the F1 score variations in Table 4. This distinguishes our approach from existing untargeted poisoning attacks.

Therefore, when the attack target is inherently difficult to learn, such as in cases exhibiting a high degree of concept drift, the problem space perturbation strategy is more suitable. Conversely, when the attack target is relatively easy to learn due to weak concept drift, the strategy based on uncertainty attribution is employed to enhance the effectiveness of the attack. Finally, we ensure that, regardless of the degree of concept drift exhibited by the attack target, the victim model fails to effectively learn the attack target due to its inability to efficiently allocate the limited label budget, ultimately leading to the misclassification of the attack target.

4.2 Overview of PACDA Attack

Based on the aforementioned design rationale, we propose the PACDA attack framework, as shown in Figure 3. The attacker first performs a value assessment of the attack target to avoid wasting limited attack resources on targets that do not hold significant

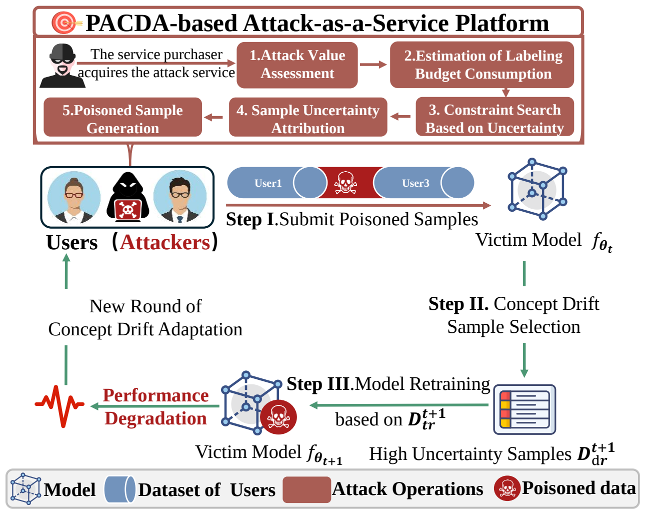
PACDA-based Attack-as-a-Service Platform
The service purchaser acquires the attack service
1.Attack Value Assessment
2.Estimation of Labeling Budget Consumption
3. Constraint Search Based on Uncertainty
4. Sample Uncertainty Attribution
5.Poisoned Sample Generation
User1
User3
Users (Attackers)
Step I.Submit Poisoned Samples
Victim Model f_{θ_t}
New Round of Concept Drift Adaptation
Step II. Concept Drift Sample Selection
Performance Degradation
Step III.Model Retraining based on D_{tr}^{t+1}
Victim Model $f_{\theta_{t+1}}$
High Uncertainty Samples D_{dr}^{t+1}
Model
Dataset of Users
Attack Operations
Poisoned data

Figure 3: PACDA Attack Process

attack value. The attacker then estimates the required labeling budget consumption and sets search constraints to identify high-uncertainty poisoned seed samples in the test dataset, guided by the attack targets. Finally, poisoned samples are generated to exhaust the victim model's labeling budget, preventing it from effectively learning the attack target. It is worth noting that PACDA provides an attack-as-a-service framework, in which the two poisoned sample generation strategies are offered at different service costs. Therefore, the choice of poisoned sample generation strategy is not determined by the attacker, but rather by the service purchaser, based on their knowledge of the attack target—such as uncertainty rankings and similarity between the attack target and existing training datasets. When the concept drift exhibited by the attack target is weak, service purchasers tend to select the poisoned sample generation strategy based on Shapley Additive Explanations, prioritizing attack effectiveness. In contrast, when the attack target displays strong concept drift, purchasers are more inclined to choose the strategy based on problem-space perturbations, which maintains attack effectiveness while further reducing the overall attack cost.

4.3 Detailed Design of PACDA

We illustrate the design of the PACDA attack method using the concept drift adaptation process from time t to $t+1$ as an example, while maintaining the same attack procedure for other time periods. During the concept drift adaptation process from time t to $t+1$, the complete test dataset collected by the victim model is denoted as $D_{te}^{t+1} \in \mathbb{R}^{N \times d}$ (with N samples in d dimensions). The attack target sample $\mathbf{x}_{tar}$ belongs to test dataset, the victim model is denoted as f_{θ_t}, and the labeling budget of the victim model is denoted as β. The victim model performs uncertainty quantification on the collected test dataset and ranks the samples accordingly. The top β samples with the highest uncertainty ranking are selected for manual analysis to obtain the concept drift dataset $D_{dr}^{t+1} \in \mathbb{R}^{\beta \times d}$ (with β samples in d dimensions).

Attack Value Assessment: The PACDA attack is essentially a resource-exhaustion attack that targets the labeling budget of the victim model. However, the attacker also faces similar risks of resource exhaustion. For example, the attacker may waste resources by targeting an attack target that already exists in the training dataset. A failed attack not only prevents the attacker from obtaining any reward but also incurs substantial attack costs. Consequently, the attacker must carefully assess whether the attack target is worth attacking. The criterion for identification is the prediction accuracy of the victim model on the attack target, as the most distinctive characteristic of samples absent from the training dataset is their tendency to be misclassified by the victim model. The attacker can leverage the victim model to make this determination. If the predicted label $\overline{y}_i$ of $\mathbf{x}_{tar}$ from the victim model f_{θ_t} differs from the ground-truth label y_i, the attack target is likely to be of high attack value.

Estimation of Labeling Budget Consumption: Once the attack target is confirmed to have attack value, the PACDA attack first needs to estimate how much labeling budget the attacker needs to exhaust from the victim model f_{θ_t} based on the attack target $\mathbf{x}_{tar}$. The attacker makes decisions based on the uncertainty ranking of the attack target. If the uncertainty ranking r_{tar} of the attack target is lower than the labeling budget boundary ($r_{tar} < r_\beta$), the attack target will be selected as a concept drift sample. Then the attack target will be learned during the retraining process of the victim model. The gap between the uncertainty ranking r_{tar} of attack target and the labeling budget boundary r_β represents the primary objective of the attacker's labeling budget consumption, which is $(r_\beta - r_{tar})$. It also reflects the minimum requirement for the number of poisoned samples. When $r_{tar} > r_\beta$, it indicates that the attack target $\mathbf{x}_{tar}$ will not be selected by the victim model f_{θ_t} at the current time t. It is important to note that even if the attack target is not selected, the attacker will still generate some poisoned samples. Our goal is to prevent a mismatch between the attacker's and the victim's collected test data, which may cause the estimated uncertainty ranking of attack targets to be higher than their actual ranking. Therefore, to ensure reliability, the consumption of the labeling budget is set to a fixed value λ, determined by the attacker's computational resources. The amount of labeling budget consumption is as follows.

$$C_{t+1} = \begin{cases} r_\beta - r_{tar}, & r_{tar} < r_\beta \\ \lambda, & r_{tar} > r_\beta. \end{cases} \tag{4}$$

Constraint Search based on Uncertainty Ranking: The attacker then searches the test dataset D_{te}^{t+1} for potential poisoned samples that can exhaust the victim model's labeling budget C_{t+1}. These naturally available samples suitable for poisoning are referred to as poisoning seed samples. The poisoning seed dataset $D_{seed}^{t+1} \in \mathbb{R}^{K \times d}$ (with K samples in d dimensions) is composed of samples that satisfy the uncertainty criterion defined in Equation 5. We denote each row of the test dataset as a sample vector $\mathbf{x}_i$, where $\mathbf{x}_i = D_{\text{te},i*}^{t+1}, \forall i \in \{0, \ldots, N-1\}$. The uncertainty of these poisoned seed samples must exceed the uncertainty of attack target, as shown below.

$$uncer(f_{\theta_t}(\mathbf{x}_i)) > uncer(f_{\theta_t}(\mathbf{x}_{tar})) \tag{5}$$

Furthermore, we must ensure that the poisoned samples serve solely to exhaust the victim model's labeling budget without enhancing its concept drift adaptation capability. For instance, incorporating novel malware samples into the poisoning seed dataset may improve the victim model's ability to detect malicious behavior, thereby undermining the attacker's objective of causing misclassification of the attack target. Therefore, the attacker needs to remove malware samples from the attack seed dataset to avoid a reduction in the attack success rate on the attack target. In addition, similarity-based filtering can be employed to prevent the inclusion of samples that are similar to the attack target in the poisoned samples.

$$(y_j \neq y_{tar}) \wedge [sim(\mathbf{x_j}, \mathbf{x_{tar}}) < \tau] \tag{6}$$

We denote each row of the attack seed dataset $\boldsymbol{D}_{seed}^{t+1}$ as a sample vector $\mathbf{x_j}$, where $\mathbf{x_j} = \boldsymbol{D}_{\text{seed}}^{t+1}[j,:],\forall j \in \{0,\ldots,K-1\}$. The specific constraint applied to each sample vector $\mathbf{x_j}$ is show in Equation 6. Samples that do not meet the specified criteria will be removed from the poisoning seed dataset $\boldsymbol{D}_{seed}^{t+1}$. At this point, the size of the poisoning seed dataset is reduced from K to M.

$$\boldsymbol{D}_{seed}^{t+1} = UncerSort(\boldsymbol{D}_{seed}^{t+1}) \tag{7}$$

Finally, the attack seed dataset are sorted by uncertainty in ascending order, with the sample at index 0 having the highest uncertainty, as shown in Equation 7.

Sample Uncertainty Attribution: Next, the attacker constructs an uncertainty interpreter based on Shapley Additive Explanations. The goal is to prepare for the subsequent poisoned sample generation strategy based on Shapley Additive Explanations. We employ standardized permutation-based SHAP to interpret the uncertainty of samples. This method approximates the Shapley values by iterating through permutations of the inputs. This is a model agnostic explainer that guarantees local accuracy (additivity) by iterating completely through an entire permutation of the features in both forward and reverse directions (antithetic sampling). Since the objective of our attack is to prevent the attack target from being effectively learned, the poisoned samples should be designed to suppress the victim model's adaptation to concept drift. Therefore, we select samples outside the labeling budget as targets for feature space perturbation, as their lower importance compared to in-budget samples helps avoid enhancing the victim model's ability to adapt to concept drift. The attacker performs uncertainty feature attribution on the test samples outside the labeling budget, denoted as $\boldsymbol{D}_{shap}^{t+1}$ ($\boldsymbol{D}_{shap}^{t+1} = \boldsymbol{D}_{te}^{t+1} \setminus \boldsymbol{D}_{dr}^{t+1}$). Additionally, the samples in $\boldsymbol{D}_{shap}^{t+1}$ are sorted in descending order based on their uncertainty.

$$\boldsymbol{V_{shap}} = PermutationSHAP(\theta_t^*, \boldsymbol{D}_{tr}^t, UncerSort(\boldsymbol{D}_{shap}^{t+1})) \tag{8}$$

$\boldsymbol{V_{shap}}$ represents the matrix of uncertainty feature attributions for dataset $\boldsymbol{D}_{shap}^{t+1}$. $\boldsymbol{v_i}$ denotes the feature attribution vector of a specific sample, while $v_{i,j}$ indicates the impact of a particular feature on the uncertainty of the given sample.

Poisoned Sample Generation: Poisoned seed sample dataset $\boldsymbol{D}_{seed}^{t+1}$ enters the victim model's training dataset due to their high uncertainty. However, the seed samples obtained through the constrained search strategy may not fully cover the victim model's label budget consumption C_{t+1}. Consequently, attack targets $\mathbf{x_{tar}}$ still be selected and labeled manually, causing the PACDA attack to fail. To ensure the effectiveness of the PACDA attack, the attacker needs to generate a batch of poisoned samples, ensuring that the number of poisoned samples is greater than or equal to the victim model's label budget consumption C_{t+1}.

We propose two poisoned sample generation strategies: problem space perturbation and feature space perturbation. Problem space perturbation is characterized by its ability to quickly and cost-effectively adjust the uncertainty ranking of samples within the label budget. For example, assume that sample $\mathbf{x_A}$, a row vector representing an individual sample, belongs to the poisoning seed dataset $\boldsymbol{D}_{seed}^{t+1}$. Let the attack target be denoted by $\mathbf{x_{tar}}$. Their uncertainty rankings are represented by r_A and r_{tar}, respectively, and their uncertainty scores are u_A and u_{tar} ,with $u_A > u_{tar}$. By applying problem space perturbation to sample $\mathbf{x_A}$, a poisoned sample with the similar uncertainty u_A can be generated. This increases the uncertainty ranking of the attack target r_{tar}, pushing it outside the label budget (i.e., $r_{tar} > r_\beta$).

However, the uncertainty of poisoned samples generated through problem space perturbation is limited by the uncertainty of the seed samples from $\boldsymbol{D}_{seed}^{t+1}$. As a result, it is difficult to affect samples with higher uncertainty than the poisoned seed samples. If the portion of the labeling budget with higher uncertainty than the poisoning seed samples contains samples that hinder the misclassification of the attack target (e.g., malware using similar vulnerabilities as the target), the attack success rate may decrease. To address this limitation, we propose a feature space perturbation method based on the uncertainty attribution matrix $\boldsymbol{V}_{shap}$. This approach overcomes the upper bound constraint on poisoned sample uncertainty imposed by the poisoned seed samples.

Since feature space perturbation can overcome the limitations of problem space perturbation, it serves as an enhancement to the problem space perturbation strategy. So the attacker can operate in two different attack modes based on the cost the attack service purchaser is willing to pay. The attacker can choose to use the problem space perturbation strategy alone. This approach is low-cost and highly stealthy, but the attack success rate on some samples cannot be guaranteed. If the attacker has sufficient computational resources and aims for a higher attack success rate, they can generate poisoned samples by applying feature space perturbations. Next, we provide a detailed introduction to the two poisoned sample generation strategies.

(1) Problem Space Perturbation: The problem space perturbation strategy refers to modifying the samples in the poisoning attack seed dataset $\boldsymbol{D}_{seed}^{t+1}$ to generate new poisoned samples, without altering the features of the samples. We denote each row of the attack seed dataset $\boldsymbol{D}_{seed}^{t+1}$ as a sample vector $\mathbf{x_k}$, where $\mathbf{x_k} = \boldsymbol{D}_{\text{seed}}^{t+1}[k,:],\forall k \in \{0,\ldots,M-1\}$. Due to the need for attack stealthiness, the samples generated by problem space perturbation must minimize the impact on other concept drift samples. Therefore, we select the ϵ least uncertain samples from the poisoned seed sample set for problem space perturbation. The smaller ϵ is, the lesser the impact on existing concept drift samples. In the subsequent attack effectiveness evaluation in this paper, we set ϵ to 5, which accounts for 0.025 of the label budget. A perturbation in the problem space is applied to these samples $\mathbf{x_k}$, $k \in \{0,\ldots,\epsilon-1\}$, resulting in a new dataset denoted as $\boldsymbol{D}_\alpha^{t+1}$. α denotes the problem-space perturbation

Table 1: Problem-Space Perturbation Operations

Dataset	Perturbation Operations (α)
APIGraph	Rename method names to meaningless identifiers
	Modify image and other resource files
	Transform complex control structures into simpler ones
BODMAS	Modify the system API function names
	Move variables to the header structure fields
	Dynamically adjust the size of the DOS STUB space
MNIST	Add Gaussian noise to the image pixels
	Perform geometric transformations, such as rotation
	Apply sharpening to enhance the edges of the image
SPAM	Remove non-essential words
	Split long sentences into shorter ones
	Insert random symbols or additional spaces

operations. This strategy ensures that the newly generated poisoned samples still exhibit high uncertainty, thereby maintaining their ability to exhaust the victim model's labeling budget.

The reason is that machine learning detection models, during the feature extraction process, often ensure that non-critical perturbations α in the data do not affect the features in order to construct robust representations. Therefore, when the attacker adjusts the non-essential information of a sample, the sample is altered, but its features remain unchanged. Since existing uncertainty quantification methods are based on sample features, altering the problem space does not reduce the sample's uncertainty, ensuring that the labeling budget can be effectively exhausted. For example, in software applications, when extracting key information such as permissions and API calls as features, elements like icon usage, coding style, and redundant code have no direct relationship with these critical features. Examples of problem space perturbations in different domains can be found in Table 1. All perturbation operations preserve the original labels of the poisoned seed samples. The perturbation operation in the problem space is infinite, allowing for the generation of an ample number of poisoned samples. Thus, by leveraging high-uncertainty seed samples $\boldsymbol{D}_{seed}^{t+1}$ and the problem space perturbation strategy, the attacker can generate poisoned samples $\boldsymbol{D}_{\alpha}^{t+1}$ to meet the victim model's label budget consumption requirements. This attack strategy is fast in generating poisoned samples and has low cost, making it the preferred choice for most attack targets. As long as the poisoned sample seed set is not empty, problem space perturbation can be applied.

(2) Feature Space Perturbation: The uncertainty attribution module has obtained the uncertainty attribution matrix $\boldsymbol{V}_{shap}$. We denote each row of the uncertainty attribution matrix as a vector $\mathbf{v}_i$, where $\mathbf{v}_i = \boldsymbol{V}_{\text{shap}}[i,:], \forall i \in \{0,\dots,N-1\}$. Based on the uncertainty feature attribution vector $\mathbf{v}_i$ for each sample, the attacker identifies the set of feature indices $\boldsymbol{I}_{shap}$ that have the greatest impact on increasing uncertainty. The computation of the feature index vector $\boldsymbol{I}_{shap}$ is shown in Equation 9.

$$\boldsymbol{I}_{shap} = \{argsort(\mathbf{v}_i)[0:d-1]\} \tag{9}$$

d denotes the dimension of the feature vector for each sample. Subsequently, feature space perturbation is performed on each sample based on the high-uncertainty attribution feature indices $\boldsymbol{I}_{shap}$, where the top 2% of features—ranked by their contribution to uncertainty—are selectively modified. Since the uncertainty-based SHAP attribution provides a linear approximation of the model's predictive uncertainty, modifying the key attributed features effectively increases the overall uncertainty of the sample. Taking the APIGraph dataset as an example, its features are represented as 0-1 vectors. The corresponding features are modified by flipping their values: changing 0 to 1, or 1 to 0. This corresponds to adjusting the API call patterns in the actual application. Since the purpose of the poisoned samples is to consume the labeling budget, the impact of the API calls on the program's functionality does not need to be considered. For datasets in other domains, we replace the values of the important features with the corresponding values from samples that have high uncertainty. The perturbed samples generated in this manner are considered as a new set of poisoned attack samples $\boldsymbol{D}_{shap}^{t+1}$. If the generated poisoned samples are smaller than the label budget consumption requirements C_{t+1}, the attacker can further amplify the set by applying problem space perturbation to the poisoned samples $\boldsymbol{D}_{shap}^{t+1}$.

Table 2: Concept Drift Datasets for Attack Evaluation

Dataset	Type(Num)	Duration	Real/Synthetic
APIGraph [90]	Android(320315)	7 years	Real
BODMAS [82]	Windows(149217)	2 years	Real
MNIST [78]	Image(70000)	5 cycles	Synthetic
SPAM [27]	Text(9324)	8 cycles	Synthetic

Moreover, the number of poisoned samples only needs to be sufficient to consume the victim model's labeling budget. As a result, they constitute only a small fraction of the training dataset. For example, in the real-world APIGraph [90] dataset, poisoned samples generated in a single attack account for only 1% of the training dataset. Beyond the stealthiness from the low poisoning ratio, all poisoned samples have correct labels without label flipping.

5 Empirical Evaluation

5.1 Experimental Setup

Concept Drift Datasets: The PACDA evaluation is conducted on four datasets: two synthetic datasets (MNIST [78] and Spam-Email [27]) and two real-world datasets (APIGraph [90] and BODMAS [82]). Detailed dataset information is presented in Table 2. Android malware datasets, which span a 7-year period, naturally exhibit concept drift. In contrast, non-timestamped datasets such as MNIST are typically partitioned into sub-datasets to artificially simulate concept drift. The method for synthesizing concept drift is similar to those used in existing concept drift studies [18]. By clustering the dataset based on sample similarity and selecting a subset of similar samples for initial training, we simulate a realistic scenario where less similar data is gradually introduced in the subsequent test set. Synthetic dataset creation details are in Appendix B.1.

In addition, concept drift datasets require the test data to be partitioned according to the order of occurrence. Following prior work [12], we adopt a similar setup. For datasets with timestamps, the APIGraph [90] dataset serves as an example: data from 2012 is used for training, while data from 2013 to 2018 is used for testing, as illustrated in Table 3. In the subsequent concept drift experiments, the test data of the APIGraph dataset is released progressively on

Table 3: Android Malware Concept Drift Dataset (APIGraph)

Year	Malware	Benign	Malware Family
Train-2012	3,061	27,472	104
Test-2013	4,854	43,714	172
Test-2014	5,809	52,676	175
Test-2015	5,508	51,944	193
Test-2016	5,324	50,712	199
Test-2017	2,465	24,847	147
Test-2018	3,783	38,146	128
Total	**30,804**	**289,511**	**1,118**

a monthly basis. For datasets without timestamps, the test set is divided into multiple approximately equal-sized segments, which are presented to the model sequentially. The detailed training and testing splits for the other three datasets are provided in Appendix B.2.

Victim Models: The configuration of the victim model consists of two main parts: the classifier architecture and the sample selection strategy. For the classifier setup on the Android malware dataset APIGraph [90], we use both traditional models (e.g., SVM) and deep learning models (e.g., ResNet) as classifier architectures. The settings of sample selection strategies follow existing research on concept drift adaptation [6, 12, 19, 83]. For the other datasets (MNIST, SPAM, and BODMAS), the victim model is a multilayer perceptron consisting of five hidden layers, one output layer, and LeakyReLU activation functions. To prevent overfitting, the model includes batch normalization and dropout layers. For the sample selection strategy, we use the classic confidence-based method [19]. Detailed parameter settings for all victim models can be found in Appendix C.1. All settings are designed to ensure that the victim model achieves good concept drift adaptation performance in the absence of attacks.

Attack Target: The attacker aims to continuously misclassify newly emerging concept drift samples during the testing phase while maintaining the overall performance of the victim model. So attack targets are selected from test samples emerging during the concept drift adaptation process. For example, in the malware dataset APIGraph [90], new malware samples from the test dataset are chosen as attack targets. To ensure that attack targets represent concept drift samples (e.g., malware from a new family), we select those that are misclassified upon their first appearance. Detailed attack target information is provided in Appendix C.2. Additionally, in the real world, multiple attack targets may exist at the same concept drift time point t, such as multiple new malware samples emerging within the same month as attack targets. Therefore, we classify the evaluation of attack effectiveness into two types: single-target attack and multi-target attack, as shown in Section 5.2. This setup aims to better understand the impact of different attack target settings on attack success rates in real-world scenarios.

Metrics: The effectiveness of PACDA is evaluated using the following metrics: 1) F1 Score: The harmonic mean of precision and recall, providing a balanced measure of the model's overall performance on the concept drift test dataset. 2) Attack Effectiveness: Effectiveness is assessed via the attack success rate (ASR), where success is defined as the victim model misclassifying the attack target (e.g., classifying benign as malware in malware detection). Additionally, since attack targets may already be misclassified during the early stages of concept drift, we adopt a more stringent criterion for measuring attack success. Only when a PACDA attack prolongs the duration of the attack target's misclassification will the attack be regarded as successful, as defined in Equation 11.

Table 4: PACDA Attacks with Single Attack Target

Dataset	Target (Top 10)	Metric		
		ASR	F1	ACC(%)
Android	mecor	**100%**	$0.85_{-0.07}$	$97.49_{-1.07}$
	mobidash	**100%**	$0.90_{-0.02}$	$98.28_{-0.28}$
	svpeng	**100%**	$0.90_{-0.02}$	$98.26_{-0.30}$
	smforw	**100%**	$0.81_{-0.11}$	$96.8_{-1.76}$
	fobus	**60%**	$0.88_{-0.04}$	$97.84_{-0.72}$
	adflex	**100%**	$0.90_{-0.02}$	$98.26_{-0.30}$
	vnapstore	**100%**	$0.85_{-0.07}$	$97.41_{-1.15}$
	clicks	**100%**	$0.90_{-0.02}$	$98.40_{-0.16}$
	mogap	**100%**	$0.90_{-0.02}$	$98.24_{-0.32}$
	congur	**100%**	$0.91_{-0.01}$	$98.40_{-0.16}$

The proportion of poisoned samples in the monthly test dataset averages less than 6%, demonstrating a high level of attack stealthiness.

$$Judge(y^*_{tar}, \overline{y}_{tar}, t) = \begin{cases} 1, y^*_{tar} = \overline{y}_{tar} \; at \; \text{t}, \\ 0, y^*_{tar} \neq \overline{y}_{tar} \; at \; \text{t}. \end{cases} \tag{10}$$

$$ASR(y^*_{tar}, \overline{y}_{tar}, t, N) = \frac{\sum_{t=1}^{N} Judge(y^*_{tar}, \overline{y}_{tar}, t)}{N} \tag{11}$$

N denotes the PACDA attack testing cycle length for the attack target $\mathbf{x}_{tar}$. Two settings were adopted in the subsequent experiments to standardize the effectiveness evaluation: increasing the misclassification duration by one cycle or 100%. Function $Judge()$ compares the target label y^*_{tar} of $\mathbf{x}_{tar}$ with the victim model's predicted class $\overline{y}_{tar}$. It indicates a successful attack during the testing cycle if they are equal. It is important to note that the target label y^*_{tar} of the attack target $\mathbf{x}_{tar}$ vary depending on the attacker's objectives. For example, in a malware detection scenario, the attacker's target category is "benign," while in an industrial risk analysis scenario, the attacker's target category is "secure." Since all experiments in this study require running multiple testing cycles, the testing phase incurs significant time overhead.[1] Therefore, in subsequent experimental evaluations, the testing cycle extension is set to 1 for all cases except for dedicated attack persistence tests, which utilize a full 100% testing cycle extension.

5.2 Attack Effectiveness

The PACDA attack's effectiveness is evaluated on four datasets.[2] Due to its long time span and being a real-world dataset, the APIGraph dataset [90] contains the most attack targets. So we use it for both single-target attack and multi-target attack. The manually

[1]For example, under the TRANS (Section 5.3) strategy, experiment takes approximately 4 hours per sample to complete (device specifications are detailed in Appendix C), with an average of around 200 test samples per experiment.

[2]All experimental performance metrics were obtained by averaging the results from multiple tests (five attacks were performed for each target and averaged). In addition, a smaller standard deviation indicates that the data has less variability or fluctuation.

synthesized concept drift dataset (MINIST [78] and SPAM [27]) and the BODMAS malware concept drift dataset [82] have shorter time spans, so they are used for the multi-target attack test.

5.2.1 *Single-Target Attack.*

We assess the effectiveness of PACDA attack with simple attack target on the APIGraph [90] dataset. The attack target list is detailed in Appendix C.2. To efficiently demonstrate the effectiveness of the PACDA attack and show how feature space perturbation enhances problem space perturbation, we primarily use a low-cost problem-space perturbation strategy for generating poisoned samples. Additionally, in the APIGraph [90] dataset, which has the longest available concept drift span, we apply feature space perturbation to enhance poisoned sample generation for instances where the problem space perturbation strategy fails. The failure of the problem space perturbation indicates that these attack targets exhibits weak concept drift properties, which are correlated with concept drift samples either outside the scope of the existing training dataset or beyond the victim model's labeling budget consumption range.

The PACDA attack demonstrates high effectiveness, achieving an average ASR of 88%. Table 4 presents the top 10 malware families of attack targets with the largest number of samples, along with their attack success rates and the performance metrics of the victim model. The impact on non-targeted samples is also analyzed. To maintain stealthiness, the attack minimizes its effect on overall detection performance. F1 fluctuations remain within 0.2, and ACC stays above 95%. The model's average TNR under PACDA attacks reaches 99%, ensuring minimal false alarms. It can be observed that the poisoned sample generation strategy based on problem space perturbation effectively prevents the victim model from learning the attack target, without affecting the overall performance of the victim model during the concept drift adaptation process.

Since the victim model is continuously updated during the concept drift adaptation process, it is necessary to test whether the effectiveness of our attack persists over time. To ensure fairness, the testing period was standardized. The testing cycle for each attack target is extended to 200% of the original duration, based on the initial misclassification results, as shown in Appendix C.3. For instance, if the attack target was first detected as malicious after 4 months in June 2014 underwent an 8-month attack test, the attack was deemed successful only if it evaded detection for the whole 8 months. To illustrate attack persistence trends, we selected malware samples per month from January 2013 to December 2018 as attack targets. Over 1,000 targets were tested during the 6-year concept drift adaptation process, achieving an average ASR of 86.7%, as shown in Figure 4. Thus, it is evident that our PACDA attack ensures the attack target remains consistently misclassified throughout the long-term concept drift adaptation process of the victim model.

Additionally, we further tested the impact on the PACDA attack when attackers with further limited capabilities as Weak (W) Attackers. First, the attacker is unable to query the victim model for sample uncertainty scores. In this scenario, the attacker can only obtain the predicted labels from the victim model. Other information, such as sample uncertainty, must be obtained through the surrogate model.

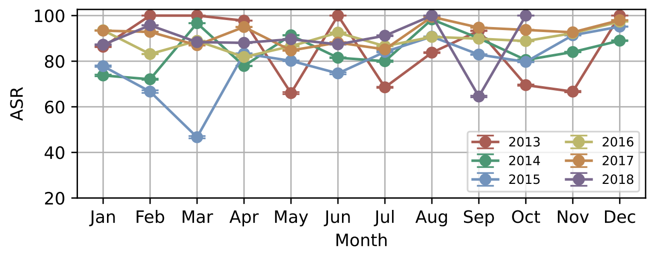

Figure 4: Attack Persistence over 6 years

So we propose a surrogate model construction method based on knowledge distillation, where the surrogate model's parameters $\boldsymbol{\theta}_t^*$ approximate those of the victim model $\boldsymbol{\theta}_t$.

$$\overline{Y} = Q(\boldsymbol{\theta}_t, \boldsymbol{D}_{te}^t) \tag{12}$$

The attacker constructs a surrogate model $f_{\theta_t^*}$ using the input-output query function Q, where the pseudo label set $\overline{Y}$ by querying the victim model f_{θ_t} with input data $\boldsymbol{D}_{te}^t$. The purpose of the surrogate model is to identify the optimal parameters θ_t^* such that the prediction error between the surrogate model and the victim model is minimized for all inputs $x_i \in \boldsymbol{D}_{te}^t$, as shown in Equation 13.

$$\boldsymbol{\theta}_t^* = \arg\min_{\theta} \sum_{\mathbf{x}_i \in \boldsymbol{D}_{te}^t} \mathcal{L}\left(f_{\theta_t}(\mathbf{x}_i), \overline{y}_i\right), \overline{y}_i \in \overline{Y} \tag{13}$$

It is important to emphasize that the attacker uses pseudo-labels $\overline{Y}$ generated by the victim model f_{θ_t} as training labels for the surrogate model, rather than relying on ground truth labels. There are two main reasons for this approach. First, the methods of concept drift adaptation is primarily applied in sensitive domains such as malware detection and industrial security risk analysis, where acquiring ground truth labels is prohibitively expensive. Therefore, by using pseudo labels $\overline{Y}$, attackers can significantly reduce the attack cost. Second, the role of the surrogate model $f_{\theta_t^*}$ is to approximate the detection capabilities of the victim model f_{θ_t}, but the victim model does not provide correct labels for the entire test dataset $\boldsymbol{D}_{te}^t$. As a result, the surrogate model, trained with pseudo-labels, performs more similarly to the victim model. When constructing the surrogate model $f_{\theta_t^*}$, it is important to properly set the query range for the test dataset $\boldsymbol{D}_{te}^t$. During the concept drift adaptation process from time t to $(t+1)$, the attacker queries the test dataset $\boldsymbol{D}_{te}^t$ from time $(t-1)$ to t. This is because the victim model has already completed learning from the concept drift data between time $(t-1)$ and t, and this model is used to quantify the uncertainty of samples from time t to $(t+1)$. Therefore, to ensure that the surrogate model $f_{\theta_t^*}$ approximates the victim model f_{θ_t} between time t and $(t+1)$, the attacker queries the test dataset $\boldsymbol{D}_{te}^t$ from time $(t-1)$ to t and uses the query results to train the surrogate model.

Second, since computational cost is the primary concern in machine learning scenarios, the attacker's model was weakened to reduce computational overhead.[3] Following state-of-the-art Android malware concept drift adaptation methods [12], which employ an encoder (ENC) and classifier (CLA), we tested four weakened settings based on the victim model, as shown in Figure 5. Previous

[3]By reducing the number of neural network layers in the encoder or classifier.

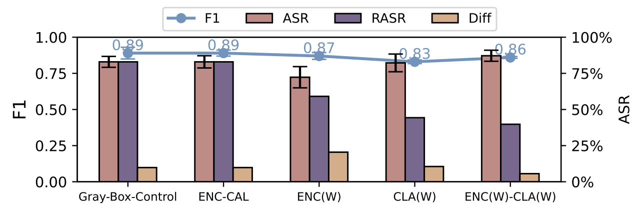

Figure 5: Attack Effectiveness under Weakened Capabilities

studies have shown that when the attacker's capabilities are limited, the effectiveness of poisoning attacks decreases [75]. However, PACDA attack mitigates this issue, achieving an average attack success rate of 81.17% with limited attacker capabilities, as shown in Figure 5. We also found that weakening the encoder reduces the attack success rate by nearly 10%, which is a greater impact than weakening the classifier. This indicates that the encoder part of the surrogate model plays a crucial role in approximating the capabilities of the victim model. Additionally, we found that weakening both the encoder and the classifier in the attacker's surrogate model does not result in the lowest attack success rate. Under this configuration, the attack success rate is even higher than that of the setup where only the classifier is weakened, at 86% compared to 83%. This phenomenon contradicts the expectation that attackers would need to invest more computational resources to achieve a higher attack success rate. By analyzing the set of attack targets under different configurations, we found that when the surrogate model is weakened, there is a significant bias in its selection of attack targets.

$$RASR = NSAS_{weak}/NSAS_{control} \tag{14}$$

In the synchronized weakening setting, attack targets are only 55% of those in control, which underscores the need to assess attack effectiveness under weakened attacker capabilities using ASR and the number of attack targets. To address this issue, we define the Number of Successful Attack Samples (NSAS) and the Relative Attack Success Rate (RASR), which represents the ratio of successfully attacked targets in the weakened attacker's setup to those in the control group, as shown in Equation 14. As model capability weakens, RASR declines, with the weakest setup showing a 43.18% drop.

5.2.2 ***Multi-Target Attack.*** In the multi-target attack setting, the attacker performs PACDA attacks on multiple attack targets simultaneously. To evaluate the effectiveness of the PACDA attack in a multi-target setting, we selected 8 months with different attack targets for testing, covering a 6-year evaluation period, as illustrated in Figure 8. For each multi-target setting, we conducted a 20-month attack test. Detailed information on the settings can be found in Appendix C.2. The multi-target attack achieved an average success rate of 97.5% across different testing times, demonstrating its effectiveness in executing PACDA attack on multiple targets. During the 80-month attack test, only the attack success rate in May 2015 was not 100%, although it still maintained an effective success rate of 80%. For malware in non-alliances, 50% of malware samples were correctly detected by the CDA-AL system within 20 testing cycles, with 75% of them being identified as malicious in the second cycle.

In certain special cases, the attacker may need to target all attack targets simultaneously. This represents a special scenario within multi-target attacks. Therefore, we also conducted PACDA attack effectiveness tests on all attack targets at different concept drift time points. **The average attack success rate is 90%.** However, the F1 score drops by 26.24% on average. False Negative Rate (FNR), a critical metric in sensitive domains, increases by an average of 26.82%, reaching 78.48% in the worst month, resulting in numerous missed malware instances. As a result, it is evident that when the attack target set is too large, the stealthiness of the PACDA attack decreases.

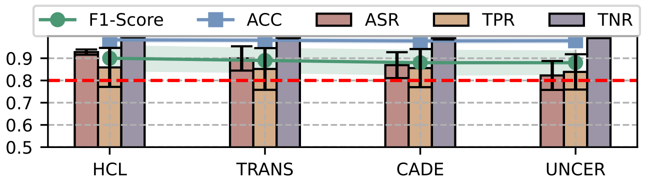

Figure 6: Different Concept Drift Adaptation Strategies

5.3 Attack Influencing Factors

We have demonstrated the effectiveness of PACDA. To further investigate how real-world conditions affect attack performance, we analyze the factors influencing PACDA attacks using a real-world Android malware dataset (APIGraph [90]) spanning seven years.

(1) Impact of Different CDA Strategies: To evaluate the impact of different concept drift adaptation strategies on the effectiveness of the PACDA attack, we evaluated PACDA's effectiveness on four mainstream strategies. Our strategy selection follows existing research [12] in the field of malware concept drift.

- **UNCER [19]** is the most classic concept drift adaptation strategy. A common uncertainty measure for a neuralnetwork is to use one minus the max softmax output of thenetwork.
- **CADE [83]** trains an encoder with labeled data to learn compressed input representations. It uses a distance function to identify concept drift samples that deviate from the training dataset.
- **TRANS [6]** applies conformal prediction [71] to concept drift adaptation. It calculates a test sample's non-conformity score, credibility (proportion of calibration samples with higher scores), and confidence (1 minus the opposite label's credibility). Low scores indicate potential drift.
- **HCL [12]** proposes the latest concept drift adaptation strategy of Android malware, combining an encoder and classifier. Its training loss integrates hierarchical contrast and classification losses, improving drift adaptation by refining encoder rules and leveraging malware family similarities.

The evaluation covers traditional uncertainty-based methods and advanced concept drift strategies like contrastive learning. As shown in Figure 6, PACDA attacks achieve over 80% ASR across all four strategies while maintaining F1 scores above 0.88. PACDA attacks attain a 92.77% ASR against the latest adaptation method (HCL).

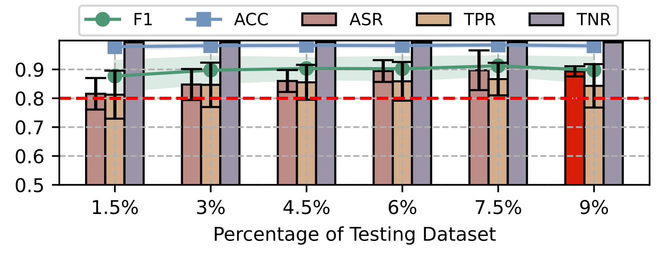

Figure 7: PACDA Attacks Under Different Label Budget

(2) Impact of Label Budget: The label budget represents the cost of manual labeling and is one of the most valuable resources in concept drift adaptation. We evaluated the attack effectiveness under 6 different label budget settings. Each labeling budget setting represents its proportion of the test dataset, following the settings of existing studies [12]. The first 5 label budget settings assume that the attacker is aware of the victim model's label budget, while the last setting assumes that the attacker does not have access to the victim model's label budget. When the attacker is unaware of the label budget settings, poisoned samples are generated based on the maximum computational capacity of attacker. In this experiment, we set the attacker's capacity to be four times the potential label budget computation capacity. As shown in Figure 7, PACDA remains effective across budgets,with an average attack success rate of 86.30%. Even when the attacker is unable to access the victim model's label budget but uses the maximum available resources for the attack, a high attack success rate is maintained, reaching 89.33%. The high attack success rate, even without access to detailed label budget information, is due to the significantly lower cost of poisoning attacks compared to the victim's concept drift adaptation cost, which is primarily driven by labeling expenses. Thus, the attacker can generate as many poisoned samples as possible to consume the victim model's labeling budget. This strategy is inspired by DDoS [48] attacks, where the attacker maximizes attack traffic without knowing the victim's total resources to achieve the attack objective.

(3) Impact of Data Collection Capability: In multi-target attacks, there is a special case where the attack target consists of all the test data. In this case, the attacker's poisoning seed samples need to be the highest uncertainty samples from the test dataset, making data collection capability crucial to its effectiveness. So we conducted attack experiments with five settings on four datasets. PACDA attacks rely on identifying high-uncertainty attack seeds to generate poisoned samples. The uncertainty of these seeds is influenced by the attacker's data collection capabilities, affecting the overall attack effectiveness. Tests were conducted across four datasets under different data collection capacities to analyze their effects on attack performance, as illustrated in Figure 9. No clear correlation was observed between reduced data collection capability and diminished attack effectiveness. Across all four datasets, reduced data collection sometimes improved attack performance. For instance, in the MNIST dataset, attacks with 90% data collection capability outperformed those with 100% capability in over 80% of the attack test cycles. Similarly, in the APIGraph [90] dataset, 50% capability occasionally surpassed 100% in specific periods. The above analysis shows that attack effectiveness relies more on the presence of high-uncertainty samples than on data collection capability alone. Even when there are differences in data quantity and distribution between the attacker and the victim model, attackers can still achieve significant success by acquiring such samples.

(4) Impact of Attack Value Assessment: Analyzing the attack value of attack targets provides crucial support for the PACDA attack. To demonstrate the significance of this step, we conduct ablation experiments where attackers skip the attack value assessment phase and proceed directly to attack seed sample selection and poisoned sample generation. This approach implies that attackers target all new malware, leading to a significant increase in attack cost. Our ablation study involving two different concept drift strategies (CADE and HCL) and a label budget of 200 samples reveals that removing the attack value assessment module leads to an average decrease of 15.62% in attack success rates. This highlights the critical role of the attack value assessment module in PACDA.

Table 5: Attack value assessment necessity analysis

Stratege	Ablation-ASR (%)	ASR (%)
HCL (Model Based)	81.73	**92.77**$_{+11.04}$
CADE (Data Based)	72.43	**86.90**$_{+14.47}$

6 Potential Defenses

CDA-AL methods lack effective defences against poisoning attacks. While Lin et al. [36] proposed defences based on static unlabeled and clean datasets, these assumptions fail in CDA-AL due to evolving unlabeled data and outdated clean datasets. Therefore, we tested the PACDA attack against four existing defense methods [10, 34, 38, 40] for poisoning attacks.

- **Activation Clustering (AC)** [10] is a data inspection method that assumes poisoned data forms a distinct cluster within the target class, either small or distant from the class centre.
- **Data-Free Pruning (DFP)** [34] sanitizes poisoned models by pruning neurons dormant with respect to clean data, assuming poisoned and clean data activate different neurons.
- **Fine-Tuning (FT)** [40] can mitigate targeted poisoning attacks, but it requires extensive labeled training data to avoid overfitting.
- **Fine-Pruning (FP)** [38] mitigates poisoned models by pruning neurons dormant on clean validation data, measuring average neuron activation, and removing the least active ones.

The defence assumptions of FT and FP rely on the victim model having access to a large amount of labeled clean data for fine-tuning or model pruning. However, this assumption is impractical in concept drift adaptation scenarios, where the labeled budget is limited each month, and the training data distribution is continuously evolving. The effectiveness of defences based on feature separability (AC) and model parameter adjustments (DFP) is insufficient, as illustrated in Table 6. The attack success rate is reduced by less than 20% on average. Given these limitations, we explore alternative defense strategies in countering PACDA attacks.

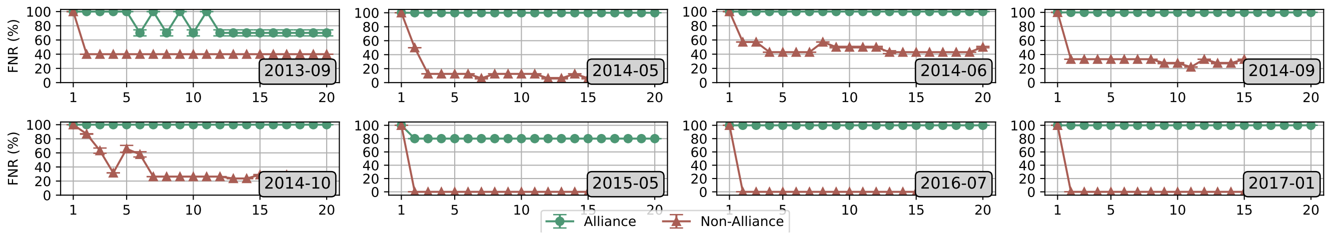

Figure 8: Attack Effectiveness with Multi-Target Attack (6 year Attack Testing)

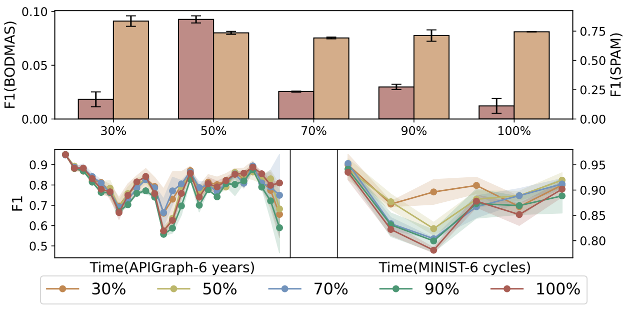

Figure 9: Impact of Data Collection Ability

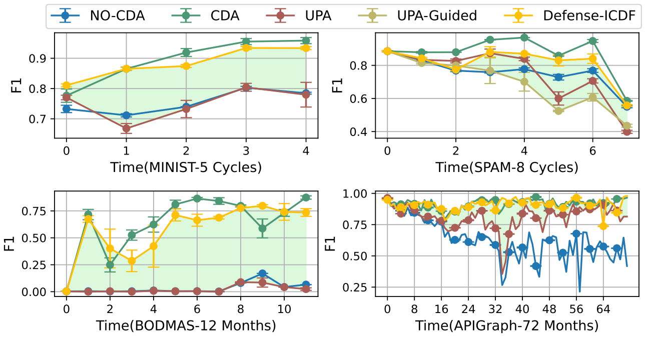

Figure 10: ICDF Defense (Multi-Target PACDA)

Table 6: Existing Defenses Against PACDA Attacks

Method	F1-score	Avarage ASR Decrease (268 Months)
AC [10]	0.90	16.5%
DFP [34]	0.87	21.8%
FT [40]	-	without extensive labeled training dataset
FP [38]	-	without clean labeled validation dataset
ICDF(Our)	0.90	**39%**

Intra-Cluster Distance Filtering (ICDF): We found that in the PACDA attack, the attacker relies on attack seed samples to generate poisoned samples, which reduces costs but also leads to similarity among the poisoned samples. This is because different poisoned samples may originate from the same attack seed sample. We measure this similarity using Euclidean distance in the feature space, which serves as the basis for our defense method (ICDF). Unlike AC-based defenses, ICDF filters poisoned samples by exploiting the similarities between the poisoned samples themselves. So, we propose an adaptive sample filtering method based on intra-cluster distance. Using unsupervised clustering algorithms (e.g., K-means), concept drift samples are partitioned into cluster set C ($C = \{c_1, c_2, .., c_k\}$). Each cluster c_k calculates an intra-cluster distance threshold τ (Equation 15) as the mean Euclidean distance $d_{Euc}(x_i, x_j)$ between its samples.

$$\tau = \frac{1}{|c_i|(|c_i|-1)} \sum_{i \neq j} d_{Euc}(x_i, x_j) \quad i, j \in c_k \tag{15}$$

$|c_i|$ is the number of samples in the cluster c_i. Samples with distances smaller than τ to others in the same cluster are removed. Figure 10 demonstrates that the ICDF defence effectively counters multi-target attack across all four datasets. Green shading highlights performance improvements after applying ICDF, with notable gains in MNIST and BODMAS, where model performance during some periods surpassed the optimal performance of concept drift adaptation. The superior performance over the best original concept drift adaptation is due to ICDF's clustering approach, which removes poisoned samples and reduces data redundancy in clean samples, improving training efficiency. Although there is a lack of research in the concept drift field, studies in computer vision have shown that data redundancy reduces model training efficiency [29]. Since improving concept drift adaptation is beyond the scope of this study, we will address it in future research. BODMAS exhibited the highest improvement, with an average F1 increase of 0.52 (400% improvement over the PACDA attack) and a maximum increase of 0.78. Under single-target attack mode, ICDF achieved the best defence effect, reducing the PACDA attack success rate by 39% across the Top 10 target sets (Figure 11). While AC and DFP showed some effectiveness, they were 23.5% and 18.8% less effective than ICDF. Additional analysis of ICDF under varying parameter settings is provided in Appendix E.

7 Related Work

Poisoning Attacks Against Active Learning. Zhao et al. [93] investigated the security of sampling strategies in active learning. However, their approach assumes that attackers can remove samples from the test dataset, preventing the victim model from collecting them. In contrast, in real-world scenarios, attackers typically do not have control over the data collection process of the victim model. Miller et al. [46] have discussed the adversarial threats to active learning and pointed out the risk of adversaries manipulating

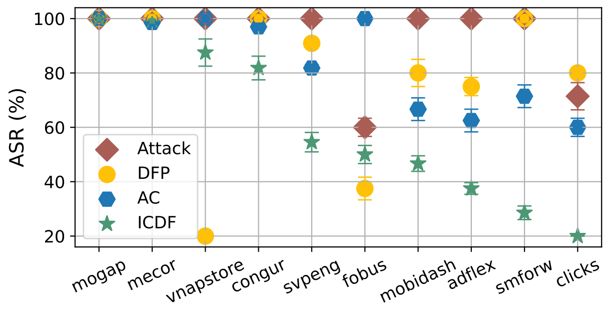

Figure 11: ICDF Defense (Single-Target PACDA)

the selection process that identifies samples for labeling. However, this type of attack is indiscriminate, aiming to cause misclassification of all test inputs, lacking sufficient stealth and not aligning with our attack goal. Vicarte et al. [69] proposed a backdoor attack against active learning, leading to misclassification of specific attack targets. Nevertheless, their method requires adding triggers to the attack targets, which incurs significant overhead during concept drift. This is because the victim model is continuously updated throughout the concept drift process, and the attacker must constantly generate new triggers. In contrast, our PACDA attack requires no modifications to the attack targets.

Adversarial Concept Drift. Korycki et al. [30] pointed out that existing drift detectors are unable to distinguish between real and adversarial concept drift. Their proposed adversarial concept drift scenario assumes a strong adversary capable of directly manipulating the victim model's training dataset. This differs from our setting, where only selected samples are labeled and used for training. Apruzzese et al. [5] analyzed the impact of adversarial perturbations occurring simultaneously with real concept drift in the context of intrusion detection. However, their study primarily focuses on the intrusion detection scenario, with limited analysis and validation in other domains. Moreover, the study lacks an explanatory analysis of why some adversarial perturbations negatively impact the attacker.

8 Conclusion and Future Work

In this paper, we propose a novel poisoning attack against concept drift adaptation based on active learning. We introduce a method that manipulates the uncertainty ranking of test samples by crafting poisoned examples, leading to the misallocation of the victim model's labeling budget and preventing it from effectively learning the attack target. Furthermore, we design a novel poisoned sample generation method that leverages naturally occurring concept drift samples as the foundation, significantly reducing the construction cost of poisoned samples. Extensive evaluations of attack effectiveness demonstrate that existing concept drift adaptation methods based on active learning are vulnerable to our poisoning attacks, particularly in sensitive domains such as malware detection. In addition, we analyzed existing defenses against poisoning attacks, revealing their limitations in addressing the attacks we propose, and introduced a new poisoned sample filtering method targeting the concept drift adaptation process. Another important future direction is to extend our research to the security of continual learning [20, 22, 35]. As both concept drift adaptation and continual learning are machine learning paradigms designed to address the challenges posed by ever-changing real-world data environments [15, 73, 77], they share similar motivations. However, continual learning typically assumes the emergence of new tasks over time, whereas concept drift assumes a fixed task with shifting data distributions. This key difference motivates our future work, which aims to explore the impact of data poisoning attacks within continual learning frameworks.

References

[1] 2018. virbox. https://shell.virbox.com/. Accessed: 2025-01-06.
[2] Imran Ahmed, Gwanggil Jeon, and Francesco Piccialli. 2022. From artificial intelligence to explainable artificial intelligence in industry 4.0: a survey on what, how, and where. *IEEE Transactions on Industrial Informatics* 18, 8 (2022), 5031–5042.
[3] Sajid Ali, Tamer Abuhmed, Shaker El-Sappagh, Khan Muhammad, Jose M Alonso-Moral, Roberto Confalonieri, Riccardo Guidotti, Javier Del Ser, Natalia Díaz-Rodríguez, and Francisco Herrera. 2023. Explainable Artificial Intelligence (XAI): What we know and what is left to attain Trustworthy Artificial Intelligence. *Information fusion* 99 (2023), 101805.
[4] Plamen P Angelov, Eduardo A Soares, Richard Jiang, Nicholas I Arnold, and Peter M Atkinson. 2021. Explainable artificial intelligence: an analytical review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 11, 5 (2021), e1424.
[5] Giovanni Apruzzese, Aurore Fass, and Fabio Pierazzi. 2024. When adversarial perturbations meet concept drift: an exploratory analysis on ml-nids. In *Proceedings of the 2024 Workshop on Artificial Intelligence and Security*. 149–160.
[6] Federico Barbero, Feargus Pendlebury, Fabio Pierazzi, and Lorenzo Cavallaro. 2022. Transcending TRANSCEND: Revisiting Malware Classification in the Presence of Concept Drift. In *2022 IEEE Symposium on Security and Privacy (SP)*. 805–823. doi:10.1109/SP46214.2022.9833659
[7] Firas Bayram, Bestoun S Ahmed, and Andreas Kassler. 2022. From concept drift to model degradation: An overview on performance-aware drift detectors. *Knowledge-Based Systems* 245 (2022), 108632.
[8] Battista Biggio, Blaine Nelson, and Pavel Laskov. 2011. Support Vector Machines Under Adversarial Label Noise. In *Proceedings of the Asian Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 20)*, Chun-Nan Hsu and Wee Sun Lee (Eds.). PMLR, South Garden Hotels and Resorts, Taoyuan, Taiwain, 97–112. https://proceedings.mlr.press/v20/biggio11.html
[9] Ying-Ying Chang, Wei-Yao Wang, and Wen-Chih Peng. 2024. SeGA: Preference-Aware Self-Contrastive Learning with Prompts for Anomalous User Detection on Twitter. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 30–37.
[10] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. 2018. Detecting backdoor attacks on deep neural networks by activation clustering. *arXiv preprint arXiv:1811.03728* (2018).
[11] Jian Chen, Yuan Gao, Gaoyang Liu, Ahmed M Abdelmoniem, and Chen Wang. 2024. Manipulating Pre-Trained Encoder for Targeted Poisoning Attacks in Contrastive Learning. *IEEE Transactions on Information Forensics and Security* (2024).
[12] Yizheng Chen, Zhoujie Ding, and David Wagner. 2023. Continuous Learning for Android Malware Detection. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, Anaheim, CA, 1127–1144. https://www.usenix.org/conference/usenixsecurity23/presentation/chen-yizheng
[13] Tianshuo Cong, Xinlei He, Yun Shen, and Yang Zhang. 2023. Test-Time Poisoning Attacks Against Test-Time Adaptation Models. arXiv:2308.08505 [cs.CR] https://arxiv.org/abs/2308.08505
[14] Chaoqun Cui and Caiyan Jia. 2024. Propagation Tree Is Not Deep: Adaptive Graph Contrastive Learning Approach for Rumor Detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 73–81.
[15] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. 2021. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence* 44, 7 (2021), 3366–3385.
[16] Yi Ding, Zi Wang, Zhen Qin, Erqiang Zhou, Guobin Zhu, Zhiguang Qin, and Kim-Kwang Raymond Choo. 2023. Backdoor Attack on Deep Learning-Based Medical Image Encryption and Decryption Network. *IEEE Transactions on Information Forensics and Security* (2023).
[17] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. 2020. Local Model Poisoning Attacks to Byzantine-Robust Federated Learning. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, 1605–1622. https://www.usenix.org/conference/usenixsecurity20/presentation/fang

[18] Bhargav Ganguly and Vaneet Aggarwal. 2023. Online federated learning via non-stationary detection and adaptation amidst concept drift. *IEEE/ACM Transactions on Networking* 32, 1 (2023), 643–653.
[19] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, et al. 2023. A survey of uncertainty in deep neural networks. *Artificial Intelligence Review* 56, Suppl 1 (2023), 1513–1589.
[20] Zhen Guo, Abhinav Kumar, and Reza Tourani. 2024. Persistent Backdoor Attacks in Continual Learning. *arXiv preprint arXiv:2409.13864* (2024).
[21] Guy Hacohen and Daphna Weinshall. 2023. How to select which active learning strategy is best suited for your specific problem and budget. *Advances in Neural Information Processing Systems* 36 (2023), 13395–13407.
[22] Gyojin Han, Jaehyun Choi, Hyeong Gwon Hong, and Junmo Kim. 2023. Data poisoning attack aiming the vulnerability of continual learning. In *2023 IEEE International Conference on Image Processing (ICIP)*. IEEE, 1905–1909.
[23] Ping He, Yifan Xia, Xuhong Zhang, and Shouling Ji. 2023. Efficient query-based attack against ML-based Android malware detection under zero knowledge setting. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. 90–104.
[24] Yiling He, Junchi Lei, Zhan Qin, and Kui Ren. 2024. DREAM: Combating Concept Drift with Explanatory Detection and Adaptation in Malware Classification. arXiv:2405.04095 [cs.CR] https://arxiv.org/abs/2405.04095
[25] Trung-Hieu Hoang, Duc Minh Vo, and Minh N Do. 2024. RIP: A Simple Black-box Attack on Continual Test-time Adaptation. *arXiv preprint arXiv:2412.01154* (2024).
[26] Mahmood Karimian and Hamid Beigy. 2023. Concept drift handling: A domain adaptation perspective. *Expert Systems with Applications* 224 (2023), 119946.
[27] Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. 2010. Tracking recurring contexts using ensemble classifiers: an application to email filtering. *Knowledge and Information Systems* 22 (2010), 371–391.
[28] Jonathan Knauer, Phillip Rieger, Hossein Fereidooni, and Ahmad-Reza Sadeghi. 2024. Phantom: Untargeted Poisoning Attacks on Semi-Supervised Learning. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*. 615–629.
[29] Zhenglun Kong, Haoyu Ma, Geng Yuan, Mengshu Sun, Yanyue Xie, Peiyan Dong, Xin Meng, Xuan Shen, Hao Tang, Minghai Qin, et al. 2023. Peeling the onion: Hierarchical reduction of data redundancy for efficient vision transformer training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 8360–8368.
[30] Łukasz Korycki and Bartosz Krawczyk. 2023. Adversarial concept drift detection under poisoning attacks for robust data stream mining. *Machine Learning* 112, 10 (2023), 4013–4048.
[31] Kusum Lata, Prashant Singh, and Sandeep Saini. 2024. Exploring Model Poisoning Attack to Convolutional Neural Network Based Brain Tumor Detection Systems. In *2024 25th International Symposium on Quality Electronic Design (ISQED)*. IEEE, 1–7.
[32] Emanuele Ledda, Daniele Angioni, Giorgio Piras, Giorgio Fumera, Battista Biggio, and Fabio Roli. 2023. Adversarial Attacks Against Uncertainty Quantification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*. 4599–4608.
[33] Bo Li, Peng Qi, Bo Liu, Shuai Di, Jingen Liu, Jiquan Pei, Jinfeng Yi, and Bowen Zhou. 2023. Trustworthy AI: From principles to practices. *Comput. Surveys* 55, 9 (2023), 1–46.
[34] Changjiang Li, Ren Pang, Bochuan Cao, Zhaohan Xi, Jinghui Chen, Shouling Ji, and Ting Wang. 2024. On the Difficulty of Defending Contrastive Learning against Backdoor Attacks. In *33rd USENIX Security Symposium (USENIX Security 24)*. 2901–2918.
[35] Huayu Li and Gregory Ditzler. 2022. Targeted data poisoning attacks against continual learning neural networks. In *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.
[36] Jing Lin, Ryan Luley, and Kaiqi Xiong. 2021. Active Learning Under Malicious Mislabeling and Poisoning Attacks. In *2021 IEEE Global Communications Conference (GLOBECOM)*. 1–6. doi:10.1109/GLOBECOM46510.2021.9685101
[37] Anjin Liu, Jie Lu, and Guangquan Zhang. 2020. Diverse instance-weighting ensemble based on region drift disagreement for concept drift adaptation. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 293–307.
[38] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2018. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International symposium on research in attacks, intrusions, and defenses*. Springer, 273–294.
[39] Shengheng Liu, Xingkang Li, Zihuan Mao, Peng Liu, and Yongming Huang. 2024. Model-driven deep neural network for enhanced AoA estimation using 5G gNB. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 214–221.
[40] Yuntao Liu, Yang Xie, and Ankur Srivastava. 2017. Neural trojans. In *2017 IEEE International Conference on Computer Design (ICCD)*. IEEE, 45–48.
[41] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. 2018. Learning under concept drift: A review. *IEEE transactions on knowledge and data engineering* 31, 12 (2018), 2346–2363.
[42] Scott M Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. 2020. From local explanations to global understanding with explainable AI for trees. *Nature machine intelligence* 2, 1 (2020), 56–67.
[43] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems* 30 (2017).
[44] Kaleel Mahmood, Rigel Mahmood, Ethan Rathbun, and Marten van Dijk. 2022. Back in Black: A Comparative Evaluation of Recent State-Of-The-Art Black-Box Attacks. *IEEE Access* 10 (2022), 998–1019. doi:10.1109/ACCESS.2021.3138338
[45] Navid Malekghaini, Elham Akbari, Mohammad A Salahuddin, Noura Limam, Raouf Boutaba, Bertrand Mathieu, Stephanie Moteau, and Stephane Tuffin. 2023. Deep learning for encrypted traffic classification in the face of data drift: An empirical study. *Computer Networks* 225 (2023), 109648.
[46] Brad Miller, Alex Kantchelian, Sadia Afroz, Rekha Bachwani, Edwin Dauber, Ling Huang, Michael Carl Tschantz, Anthony D Joseph, and J Doug Tygar. 2014. Adversarial active learning. In *Proceedings of the 2014 workshop on artificial intelligent and security workshop*. 3–14.
[47] Dang Minh, H Xiang Wang, Y Fen Li, and Tan N Nguyen. 2022. Explainable artificial intelligence: a comprehensive review. *Artificial Intelligence Review* (2022), 1–66.
[48] Jelena Mirkovic and Peter Reiher. 2004. A taxonomy of DDoS attack and DDoS defense mechanisms. *ACM SIGCOMM Computer Communication Review* 34, 2 (2004), 39–53.
[49] Mehran Mozaffari-Kermani, Susmita Sur-Kolay, Anand Raghunathan, and Niraj K Jha. 2014. Systematic poisoning attacks on and defenses for machine learning in healthcare. *IEEE journal of biomedical and health informatics* 19, 6 (2014), 1893–1905.
[50] Luis Muñoz González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C. Lupu, and Fabio Roli. 2017. Towards Poisoning of Deep Learning Algorithms with Back-gradient Optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security* (Dallas, Texas, USA) *(AISec '17)*. Association for Computing Machinery, New York, NY, USA, 27–38. doi:10.1145/3128572.3140451
[51] Andrew Newell, Rahul Potharaju, Luojie Xiang, and Cristina Nita-Rotaru. 2014. On the practicality of integrity attacks on document-level sentiment analysis. In *Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop*. 83–93.
[52] Andrei Paleyes, Raoul-Gabriel Urma, and Neil D Lawrence. 2022. Challenges in deploying machine learning: a survey of case studies. *ACM computing surveys* 55, 6 (2022), 1–29.
[53] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. 2017. Practical Black-Box Attacks against Machine Learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security* (Abu Dhabi, United Arab Emirates) *(ASIA CCS '17)*. Association for Computing Machinery, New York, NY, USA, 506–519. doi:10.1145/3052973.3053009
[54] Cheong Hee Park and Youngsoon Kang. 2016. An active learning method for data streams with concept drift. In *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 746–752.
[55] Fabio Pierazzi, Feargus Pendlebury, Jacopo Cortellazzi, and Lorenzo Cavallaro. 2020. Intriguing Properties of Adversarial ML Attacks in the Problem Space. In *2020 IEEE Symposium on Security and Privacy (SP)*. 1332–1349. doi:10.1109/SP40000.2020.00073
[56] Yunxiao Qin, Yuanhao Xiong, Jinfeng Yi, and Cho-Jui Hsieh. 2023. Training Meta-Surrogate Model for Transferable Adversarial Attack. *Proceedings of the AAAI Conference on Artificial Intelligence* 37, 8 (Jun. 2023), 9516–9524. doi:10.1609/aaai.v37i8.26139
[57] Stephan Rabanser, Stephan Günnemann, and Zachary Lipton. 2019. Failing loudly: An empirical study of methods for detecting dataset shift. *Advances in Neural Information Processing Systems* 32 (2019).
[58] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 1135–1144.
[59] Denise Maria Vecino Sato, Sheila Cristiana De Freitas, Jean Paul Barddal, and Edson Emilio Scalabrin. 2021. A survey on concept drift in process mining. *ACM Computing Surveys (CSUR)* 54, 9 (2021), 1–38.
[60] Giorgio Severi, Jim Meyer, Scott Coull, and Alina Oprea. 2021. {Explanation-Guided} backdoor poisoning attacks against malware classifiers. In *30th USENIX security symposium (USENIX security 21)*. 1487–1504.
[61] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. 2018. Poison frogs! targeted clean-label poisoning attacks on neural networks. *Advances in neural information processing systems* 31 (2018).
[62] Jingyu Shao, Qing Wang, and Fangbing Liu. 2019. Learning to sample: an active learning framework. In *2019 IEEE international conference on data mining (ICDM)*. IEEE, 538–547.

[63] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning important features through propagating activation differences. In *International conference on machine learning*. PMlR, 3145–3153.

[64] Octavian Suciu, Radu Marginean, Yigitcan Kaya, Hal Daume III, and Tudor Dumitras. 2018. When does machine learning {FAIL}? generalized transferability for evasion and poisoning attacks. In *27th USENIX Security Symposium (USENIX Security 18)*. 1299–1316.

[65] Wenli Sun, Xinyang Jiang, Shuguang Dou, Dongsheng Li, Duoqian Miao, Cheng Deng, and Cairong Zhao. 2023. Invisible backdoor attack with dynamic triggers against person re-identification. *IEEE Transactions on Information Forensics and Security* (2023).

[66] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International conference on machine learning*. PMLR, 3319–3328.

[67] Zhiyi Tian, Lei Cui, Jie Liang, and Shui Yu. 2022. A Comprehensive Survey on Poisoning Attacks and Countermeasures in Machine Learning. *ACM Comput. Surv.* 55, 8, Article 166 (dec 2022), 35 pages. doi:10.1145/3551636

[68] Alanna Titterington. 2023. Google Play malware clocks up more than 600 million downloads in 2023. https://get.zimperium.com/mobile-banking-heists-2023/.

[69] Jose Rodrigo Sanchez Vicarte, Gang Wang, and Christopher W Fletcher. 2021. {Double-Cross} attacks: Subverting active learning systems. In *30th USENIX Security Symposium (USENIX Security 21)*. 1593–1610.

[70] VirusTotal. 2004. VirusTotal: Free Online Virus, Malware and URL Scanner. https://www.virustotal.com/. Accessed: 2025-01-08.

[71] Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. 2005. *Algorithmic learning in a random world*. Vol. 29. Springer.

[72] Feilong Wang, Xin Wang, and Xuegang Jeff Ban. 2024. Data poisoning attacks in intelligent transportation systems: A survey. *Transportation Research Part C: Emerging Technologies* 165 (2024), 104750.

[73] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. 2024. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024).

[74] Shuyi Wang and Guido Zuccon. 2023. An analysis of untargeted poisoning attack and defense methods for federated online learning to rank systems. In *Proceedings of the 2023 ACM SIGIR International Conference on Theory of Information Retrieval*. 215–224.

[75] Zhibo Wang, Jingjing Ma, Xue Wang, Jiahui Hu, Zhan Qin, and Kui Ren. 2022. Threats to Training: A Survey of Poisoning Attacks and Defenses on Machine Learning Systems. *ACM Comput. Surv.* 55, 7, Article 134 (dec 2022), 36 pages. doi:10.1145/3538707

[76] David Watson, Joshua O' Hara, Niek Tax, Richard Mudd, and Ido Guy. 2023. Explaining Predictive Uncertainty with Information Theoretic Shapley Values. In *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Eds.), Vol. 36. Curran Associates, Inc., 7330–7350. https://proceedings.neurips.cc/paper_files/paper/2023/file/16e4be78e61a3897665fa01504e9f452-Paper-Conference.pdf

[77] Buddhi Wickramasinghe, Gobinda Saha, and Kaushik Roy. 2023. Continual learning: A review of techniques, challenges, and future directions. *IEEE Transactions on Artificial Intelligence* 5, 6 (2023), 2526–2546.

[78] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747* (2017).

[79] Xingyu Xing, Wei Meng, Dan Doozan, Alex C Snoeren, Nick Feamster, and Wenke Lee. 2013. Take this personally: Pollution attacks on personalized services. In *22nd USENIX Security Symposium (USENIX Security 13)*. 671–686.

[80] Yuan Xun, Xiaojun Jia, Jindong Gu, Xinwei Liu, Qing Guo, and Xiaochun Cao. 2024. Minimalism is King! High-Frequency Energy-based Screening for Data-Efficient Backdoor Attacks. *IEEE Transactions on Information Forensics and Security* (2024).

[81] Limin Yang, Zhi Chen, Jacopo Cortellazzi, Feargus Pendlebury, Kevin Tu, Fabio Pierazzi, Lorenzo Cavallaro, and Gang Wang. 2023. Jigsaw puzzle: Selective backdoor attack to subvert malware classifiers. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 719–736.

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

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

[84] Li Yang, Dimitrios Michael Manias, and Abdallah Shami. 2021. Pwpae: An ensemble framework for concept drift adaptation in iot data streams. In *2021 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 01–06.

[85] En Yu, Jie Lu, Bin Zhang, and Guangquan Zhang. 2024. Online Boosting Adaptive Learning under Concept Drift for Multistream Classification. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (Eds.). AAAI Press, 16522–16530. doi:10.1609/AAAI.V38I15.29590

[86] Hang Yu, Weixu Liu, Jie Lu, Yimin Wen, Xiangfeng Luo, and Guangquan Zhang. 2023. Detecting group concept drift from multiple data streams. *Pattern Recognition* 134 (2023), 109113.

[87] Yang Yu, Qi Liu, Likang Wu, Runlong Yu, Sanshi Lei Yu, and Zaixi Zhang. 2023. Untargeted attack against federated recommendation systems via poisonous item embeddings and the defense. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 4854–4863.

[88] Shuai Yuan, Xian Sun, Hannah Kim, Shuzhi Yu, and Carlo Tomasi. 2022. Optical flow training under limited label budget via active learning. In *European conference on computer vision*. Springer, 410–427.

[89] Xi Zeng, Xiaotian Hao, Hongyao Tang, Zhentao Tang, Shaoqing Jiao, Dazhi Lu, and Jiajie Peng. 2024. Designing Biological Sequences without Prior Knowledge Using Evolutionary Reinforcement Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 383–391.

[90] Xiaohan Zhang, Yuan Zhang, Ming Zhong, Daizong Ding, Yinzhi Cao, Yukun Zhang, Mi Zhang, and Min Yang. 2020. 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*. 757–770.

[91] Kaifa Zhao, Hao Zhou, Yulin Zhu, Xian Zhan, Kai Zhou, Jianfeng Li, Le Yu, Wei Yuan, and Xiapu Luo. 2021. Structural attack against graph based android malware detection. In *Proceedings of the 2021 ACM SIGSAC conference on computer and communications security*. 3218–3235.

[92] Mengchen Zhao, Bo An, Wei Gao, and Teng Zhang. 2017. Efficient label contamination attacks against black-box learning models.. In *IJCAI*. 3945–3951.

[93] Wentao Zhao, Jun Long, Jianping Yin, Zhiping Cai, and Geming Xia. 2012. Sampling attack against active learning in adversarial environment. In *Modeling Decisions for Artificial Intelligence: 9th International Conference, MDAI 2012, Girona, Catalonia, Spain, November 21-23, 2012. Proceedings 9*. Springer, 222–233.

[94] Chen Zhu, W Ronny Huang, Hengduo Li, Gavin Taylor, Christoph Studer, and Tom Goldstein. 2019. Transferable clean-label poisoning attacks on deep neural nets. In *International conference on machine learning*. PMLR, 7614–7623.

[95] zimperium. 2023. 2023 Mobile Banking Heists Report. https://www.kaspersky.co.uk/blog/malware-in-google-play-2023/26904/.

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

[97] Indrė Žliobaitė, Marcin Budka, and Frederic Stahl. 2015. Towards cost-sensitive adaptation: When is it worth updating your predictive model? *Neurocomputing* 150 (2015), 240–249. doi:10.1016/j.neucom.2014.05.084 Bioinspired and knowledge based techniques and applications The Vitality of Pattern Recognition and Image Analysis Data Stream Classification and Big Data Analytics.

A Notation

Table 7 presents the symbols used in the description of the PACDA attack.

Table 7: List of Symbols on Concept Drift Adaptation

Notation	
Symbol	**Description**
f_{θ_t}	Victim model with parameters θ_t at time t
D_{tr}^{t}	Training dataset of victim model at time t
D_{te}^{t+1}	Test dataset between t and $t+1$
D_{dr}^{t+1}	Concept drift samples between t and $t+1$
R_{t+1}	Uncertainty ranking of the test dataset is denoted
β	Label budget of the victim model
$\mathbf{x}_{tar}$	Attack target
C_{t+1}	Labeling budget consumption for the specific attack target
D_{seed}^{t+1}	Attack seed dataset
$f_{\theta_t^*}$	Surrogate model with parameters θ_t^* at time t
V	Uncertainty feature attributions for test dataset
D_{α}^{t+1}	Poisoned samples generated by problem space perturbation
D_{shap}^{t+1}	Poisoned samples generated by feature space perturbation

B Concept Drift Dataset Settings

Concept drift datasets have significantly advanced machine learning research due to their rich temporal attribute information. However, collecting natural concept drift datasets poses challenges, as it incurs significant costs and demands accurate temporal attributes. Therefore, synthetic datasets are often constructed for experimental evaluation in existing studies, in addition to real-world datasets.

B.1 Synthetic Dataset Construction

In this experimental evaluation, we constructed synthetic concept drift datasets using a publicly available spam email dataset (SPAM-Email [27]) and the MNIST image dataset [78].

(1) MNIST: We processed the MNIST handwritten digit dataset to simulate the concept drift phenomenon. Specifically, we selected two clusters of digits, close in the reduced feature space, to represent two classes: digits 3, 5, and 8 (labeled as 0), and digits 4, 7, and 9 (labeled as 1). Each digit formed a subclass, with new subclasses introduced monthly based on feature similarity to simulate gradual concept drift. We randomly selected 30% of the data from digits 3 and 4 for the initial training set. The subsequent five cycles were used as testing periods, each selecting a portion of the unused data as the test dataset.

(2) SPAM-Email: Similar to the concept drift construction in the MNIST dataset, we conducted an in-depth analysis of the original SPAM dataset. Using sklearn's k-means algorithm with default parameters, we subdivided legitimate and spam emails into 12 clusters, each representing a distinct subclass. Based on this, we designed a concept drift partitioning scheme with 9 cycles, each containing 1,036 samples and an approximate 3:1 ratio of legitimate to spam emails. To simulate concept drift, we introduced a new subclass of legitimate and spam emails in each cycle while ensuring data from the same subclass appeared in consecutive cycles to reflect emerging trends. The first cycle served as the training set, containing one subclass for legitimate emails and one for spam. In subsequent testing cycles, new subclasses were introduced, and the proportions of existing subclasses were adjusted, ensuring at least four subclasses were present in each cycle. This approach enabled a smooth evolution of the data distribution over time while ensuring orderly family replacement.

B.2 Training and Testing Data Splits

Most training and test data partitions are static, with the model trained on the training dataset and evaluated on a fixed test dataset. However, test datasets in concept drift scenarios are dynamic and continuously evolving over time. Both the distribution and size of the test dataset change as time progresses.

(1) APIGraph: The APIGraph[90] dataset is trained on data from 2012, with concept drift testing conducted on data from 2013 to 2018, where the data for each year is incrementally released on a monthly basis. The ratio of legitimate to malicious software for each year is roughly maintained at 9:1, simulating the real-world scenario where legitimate samples significantly outnumber malicious ones.

(2) BODMAS: The BODMAS [82] dataset comprises 57,293 malware samples and 77,142 benign samples for 134,435 instances. The malware samples were randomly selected monthly from a security company's internal malware database, with data collection spanning August 29, 2019, to September 30, 2020. The data collected before October 2019 is used as the initial training dataset, while the data collected afterward serves as the concept drift testing dataset, as shown in Table 8. The testing data is evaluated on a monthly basis.

Table 8: Windows Malware Concept Drift Dataset (BODMAS)

Year	Malware	Benign	Malware Family
Train(before 10/19)	4741	17655	330
Test-10/19	4549	3925	236
Test-11/19	2494	3718	146
Test-12/19	4039	6120	147
Test-01/20	4510	5926	183
Test-02/20	4269	3703	166
Test-03/20	4990	3577	192
Test-04/20	4640	5201	162
Test-05/20	5449	6121	180
Test-06/20	4217	8182	111
Test-07/20	4995	6392	144
Test-08/20	3823	2514	125
Test-09/20	4577	4198	152
Total	**57293**	**77142**	**2274**

(3) SPAM-Email: The dataset contains spam and legitimate messages, with a moderate spam ratio of approximately 25%. It includes 9324 instances and 500 features. The initial training dataset consists of 771 legitimate emails and 265 spam emails. Subsequently, in each of the 8 concept drift testing cycles, 265 new spam emails and 771 legitimate emails are introduced.

Table 9: Synthetic Concept Drift Dataset for SPAM-Email

miner	Spam	Legitimate
Train	265	771
Test-1	265	771
Test-2	265	771
Test-3	265	771
Test-4	265	771
Test-5	265	771
Test-6	265	771
Test-7	265	771
Test-8	265	771
Total	**2387**	**6937**

(4) MNIST: The MNIST dataset contains handwritten digits, with 3,591 samples used for training and 31,860 samples distributed across 5 concept drift testing cycles.

C Victim Model Settings

We adopt commonly used or well-performing architectures from prior research [6, 12, 83] in machine learning tasks for the victim model setup. All experiments are conducted on a Windows 11 system with 96GB memory, one Intel® Core™ i7-14700K 3.4GHz CPU, and an NVIDIA GeForce RTX 4080 SUPER (16GB).

Table 10: Synthetic Concept Drift Dataset for MNIST

Year	Label-1	Label-0
Train	1752	1839
Test-1	1752	2923
Test-2	2421	2852
Test-3	2463	2867
Test-4	2734	2603
Test-5	6930	4315
Total	**18052**	**17399**

C.1 Victim Model Parameters

We present the model parameters optimized for respective classification tasks. Detailed parameter settings are shown in Table 11. Table 11 summarizes the parameter settings for experiments conducted on four datasets: APIGraph [90], MNIST [78], BODMAS [82], and SPAM-Email [27]. APIGraph: SGD optimizer with a learning rate (LR) of 0.003, batch size 1024, and hi-dist-xent loss function. The learning rate decay was set to 0.05 and applied at epochs 10, 500, and 10. The model was trained for 50 epochs. MNIST: Adam optimizer with an LR of 0.0004, batch size 64, and triplet-my loss function. No learning rate decay was applied, and the model was trained for 5 epochs. BODMAS and SPAM-Email: AdamW optimizer with an LR of 0.0004, batch size 64, and binary cross-entropy (BCE) loss function. No learning rate decay was applied. The model was trained for 50 epochs on BODMAS and 5 epochs on SPAM-Email.

Table 11: Parameters Setting of CDA-AL

Parameter	APIGraph	MNIST
Optimizer	SGD	ADAM
LR	0.003	0.0004
Batch size	1024	64
Loss	hi-dist-xent	triplet-mse
LR decay	0.05	-
Decay epochs	10,500,10	-
Learning epochs	50	5
Parameter	**BODMAS**	**SPAM-Email**
Optimizer	AdamW	AdamW
LR	0.0004	0.0004
Batch size	64	64
Loss	BCE	BCE
LR decay	-	-
Decay epochs	-	-
Learning epochs	50	5

C.2 Attack Target List

The experimental evaluation used multiple attack target configurations to demonstrate the attack's effectiveness and influencing factors. The detailed information is as follows.

(1) Single-Target Attack: We followed two principles for selecting attack targets. First, the attack target must not be included in the training set. Second, the attack target exhibited a 100% False Negative Rate (FNR) in the month of its occurrence. This indicates that the attack target is a newly emerging concept drift sample in the testing phase rather than a simple modification of existing malware in the training dataset. Based on these rules, the final attack target set includes 103 families comprising 724 samples.

(2) Multi-Target Attack: The attack targets for multi-target attack are composed of multiple single-attack targets that emerge simultaneously. The targets are detailed in Table 12. The concept drift adaptation associated with the attack target spans a period of five years.

Table 12: Attack Target for Multi-target Attack

Month	Type	Family
2013-09	Non-Target	ansca cardserv svpeng
	Target	mecor smforw vietsms
2014-05	Non-Target	gabas simplocker smssend
	Target	mecor svpeng
2014-06	Non-Target	chyapo pletor spyware tebak
	Target	mecor adflex
2014-09	Non-Target	fobus gamecheater ransomware
	Target	mecor spyware
2014-10	Non-Target	fakebank systemmonitor webapp
	Target	airpush mecor
2015-05	Non-Target	adflex kalfere styricka
	Target	mobidash vnapstore
2016-07	Non-Target	clicks
	Target	adflex blouns mspy
2017-01	Non-Target	mobidash
	Target	batmob kalfere

C.3 Attack Target Initial Misclassification Time

Since the effectiveness of the PACDA attack is defined by prolonging the misclassification duration of the original attack target, it is essential to test the misclassification duration of the attack target in the absence of any attacks. We analyzed the misclassification time of malware under different label budget settings and concept drift adaptation strategies, as shown in Figure 12. Most malware is misclassified for 1-5 months under CDA-AL, while a small portion

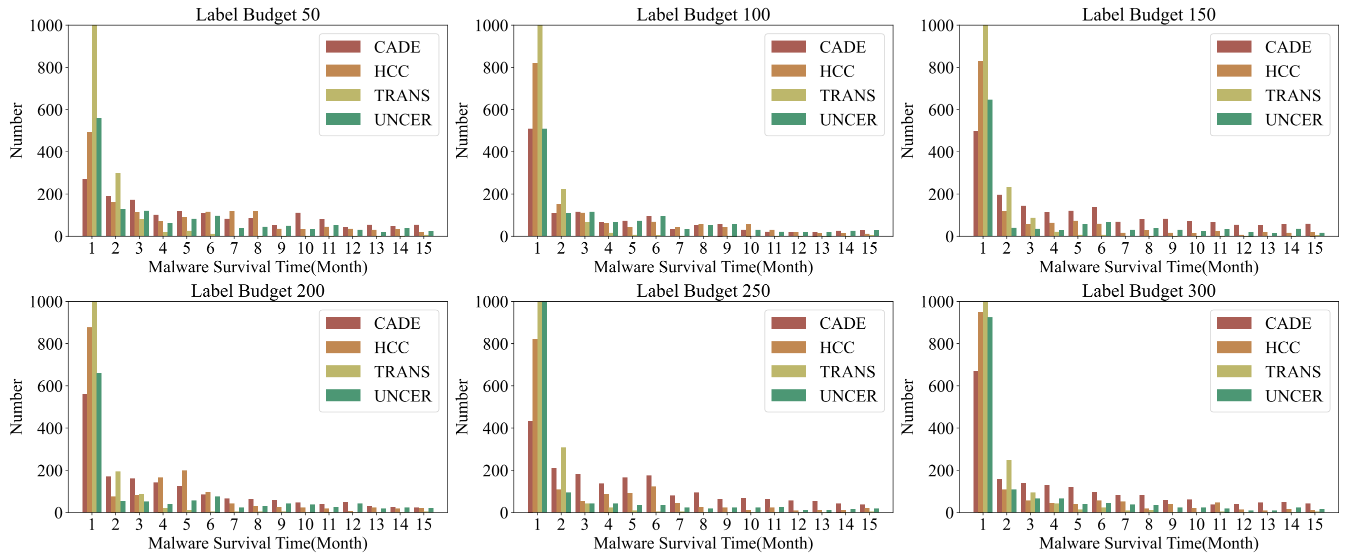

Figure 12: Original misclassification time of attack targets

survives for over 5 months. To improve clarity, the bar chart for 0-2 months survival time samples in the TRANS statistics was truncated, maintaining the relative proportions of the overall sample volumes. The PACDA attack aims to extend the misclassification duration of targeted samples. Therefore, we selected all malware samples in the testing phase with a misclassification duration of 15 months or less as attack targets.

D Existence of Attack Seeds

This section examines whether extreme conditions could impact attack effectiveness. For example, under extreme conditions, the attack target is the most uncertain sample of that month. Therefore, the attacker is unable to access the attack seed samples. However, since the attacker can adjust the release timing of the attack targets based on the uncertainty ranking of the attack targets, they may temporarily halt malware release and wait for a test cycle with attack seeds before deploying malware in the real world. The long-term value of sustained misclassification significantly outweighs the immediate benefits of premature malware release, as attack success depends more on the persistence of misclassification than the timing of release. In conclusion, even when attack seeds are absent in specific months, PACDA remains viable for attackers.

E Defence Parameter Analysis

Due to the longer time span of multi-target attack testing, we first selected the APIGraph dataset for defence parameter analysis. The total number of clusters was the only parameter requiring manual configuration during defence. We evaluated four cluster settings (6, 10, 20, and 40) under the multi-target attack. The mean F1 score across these four settings was 0.91, with a variance of 0.0025. In the single-target attack, we selected the family 'clicks' (with the best ICDF defence performance) and the family 'mogap' (with the worst ICDF defence performance) from the Top 10 attack targets. For each family, we conducted four experiments with clustering parameter settings of 6, 10, 20, and 40, and observed that the defence performance remained consistent across all settings. This indicates that the defence parameter settings have minimal impact on defence effectiveness, which can effectively reduce the deployment costs of the defence method.

F Potential Ethical Concerns

The primary purpose of our research is to evaluate the security of concept drift adaptation methods based on active learning, as related methods have received attention from researchers. Attackers are motivated to exploit the weaknesses of concept drift adaptation strategies to gain advantages, such as prolonging the survival time of new malware. Even though the intent is strict about evaluating the weaknesses of concept drift adaptation strategies, potential ethical concerns are associated with our research. For example, attackers can leverage our methods to attacks or improve malware. Therefore, following previous research precedents [23, 55, 91], we will restrict code sharing to verified academic researchers. In addition, our dataset includes public datasets and collected original APK files containing a significant amount of malware samples. Direct sharing of such data could facilitate testing by potential attackers. To mitigate this risk, we will restrict dataset sharing to verified academic researchers only.

G Open Science

In order to enhance the reproducibility and replicability of scientific findings, we will share our research artifacts. Our code (https://anonymous.4open.science/r/Denial-of-Concept-Drift-Adaptation-Code-C0EF) and CDAMAL dataset (https://anonymous.4open.science/r/CDAMAL-Concept-Drift-Dataset-3E08) are all available. In addition, considering research ethics, our dataset and the source code of the PACDA attack will only share minimized demonstration examples by default to illustrate the attack's effectiveness.