Understanding Regression in Continual Learning for Malware Detection

Daniele Ghiani[1,*], Daniele Angioni[1], Angelo Sotgiu[1], Maura Pintor[1] and Battista Biggio[1]

[1]University of Cagliari, Italy
{name.surname}@unica.it

Abstract

The evolving nature of malware poses significant challenges for machine learning-based detectors, demanding frequent updates to handle new threats. As keeping all historical data is impractical due to storage constraints, Continual Learning (CL) algorithms come to help by incrementally updating the detectors without retraining over all previously collected data. Unfortunately, updating the model might cause inconsistencies: the new model can have false positives for goodware that was previously correctly classified, and malware that was detected by the previous model can become undetected by the new one. This issue, referred to as *security regression*, is often overlooked in concurrent work but can undermine user trust despite overall detection performance improvements. In this work, we address this issue by proposing a learning strategy that combines a replay-based CL method with a regression-aware penalty to preserve the correct decisions of earlier models. Specifically, we adapt the Positive Congruent Training (PCT) strategy to a CL setting, presenting the first regression-aware CL algorithm. Experiments conducted on the ELSA Android dataset demonstrate how this approach significantly reduces security regression while keeping up with the data drift, maintaining high detection performances over time.

Keywords

Android Malware, Continual Learning, Negative Flips, Regression Testing

1. Introduction

The detection of *mal*icious soft*ware* (i.e., *malware*) is a critical aspect of cybersecurity, driven by the need to protect sensitive data, maintain system integrity, and ensure the continuous operation of services. To this end, Machine learning (ML) has revolutionized malware detection by leveraging vast amounts of data to identify complex patterns and correlations that manual analysis might miss [1, 2]. Unfortunately, ML is built on the assumption that training and testing data are sampled from the same underlying distribution, which is not the case for Android applications, as they are characterized by a fast-paced evolution during deployment. In fact, malicious users continuously adapt malware to evade detection, e.g., by obfuscating lines of code usually considered malicious by most ML-based detectors. Additionally, the frequent updates in the Android OS framework, such as the introduction of new APIs or the deprecation of old ones, make legitimate applications evolve as well. This phenomenon is known as *concept drift* [3], and is the leading cause for the severe performance degradation that affects ML-based malware detectors as time passes [4].

To avoid this issue, ML-based detectors require constant retraining to keep pace with the Android malware evolution. Since millions of new Android applications appear every single day,[1] retaining all historical data for full retraining of the model reveals to be infeasible for storage limitations. On the other hand, simply retraining on the current data causes the model to forget how to classify previously learned data, a phenomenon known as *catastrophic forgetting* [5]. To this end, Continual Learning (CL) algorithms allow to incrementally learn new patterns while retaining useful past knowledge [5]. This can be achieved, for example, by adding a regularization term to preserve prior knowledge or using a replay buffer to store a small subset of past examples for rehearsal [5]. While CL may be particularly effective in updating Android malware detectors as the data distribution changes using a limited number

Joint National Conference on Cybersecurity (ITASEC & SERICS 2025), February 03-8, 2025, Bologna, IT
*Corresponding author.

[1]https://www.virustotal.com/gui/stats

of recent samples [6], the application of CL methods for malware detection remains under-explored. Nevertheless, CL algorithms do not account for a critical issue arising from the updating process itself, known as *regression*. This is a well-known term in software development, indicating a particular type of bug arising when a software update that comes with new features comes at the cost of compromising previous functionalities that should have been preserved [7]. From the machine learning perspective, the regression problem translates to samples that were correctly classified by the previous model but are now misclassified after updating it. These samples have been referred to as *negative flips* (NFs) in [8] and can consistently jeopardize the reliability of subsequent model updates. In Android malware detection, negative flips may pose a significant threat as they reopen previously patched vulnerabilities, increasing the risk of infecting devices.[2]

In the image classification domain, Yan et al. [8] mitigate this issue by proposing *Positive Congruent Training* (PCT), which minimizes the classification loss together with a knowledge-distillation term that up-weights samples that were correctly classified by the previous model. However, this kind of approach has only been applied in updating scenarios that are different from the ultimate goal of CL, i.e., train continually without retaining all past data, as they aim either to (i) improve performance on a fixed set of data, or (ii) learn additional data that are included in the whole training set. Thus, to the best of our knowledge, none of the previous works have considered the regression issue in a CL pipeline in which retaining all past data is not allowed. Moreover, this has never been applied in the malware detection domain.

In this work, we address this issue by proposing *PCT-replay*, the first regression-aware CL algorithm that combats both catastrophic forgetting and regression in Android malware detectors. In particular, our PCT-replay (i) employs the objective function proposed in [8] to minimize negative flips, and (ii) learn current data jointly with a limited amount of past Android applications (a.k.a. rehearsal memory) to retain past knowledge, while adapting the model to the drifting data. Our experiments on the ELSA dataset, which contains Android applications spanning from January 2017 to December 2019, show that our PCT-replay outperforms all five state-of-the-art CL strategies while maintaining competing detection performance over time, both in terms of retaining past knowledge as well as anticipating the drift. This study lays the groundwork for consistent classification of previously learned data when applying CL to the malware detection domain. For future research, there is still room for improvement, exploring semi-supervised settings, where only a limited number of samples are labeled, reflecting more realistic conditions.

2. Continual Learning and Regression of Performance

Before delving into the details of our method, we first lay the foundations for incrementally learning new threats over time and present how regression is quantified and tackled by previous works in a non-CL update scenario.

2.1. Continual Learning Pipeline

Let us denote as $\boldsymbol{x}$ the Android application represented as a d-dimensional feature vector, for which we assign a label y that is $0/1$ if the sample belongs to the goodware/malware class.

In a CL scenario, we consider a sequential stream of data $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_K\}$ composed of K experiences. We then consider each experience $\mathcal{D}_k = \{\boldsymbol{x}_i^k, y_i^k, t_i^k\}_{i=1}^{n_k}$ comprised by n_k samples registered at time $t_i^k \in \Delta t_k = [t_{k-1}, t_k]$, such that the sequence of experiences well describes the temporal evolution of the data. This kind of scenario is typically referred to as a *domain-incremental* learning problem, which aims to continuously adapt the model to new emerging domains (in this case, the drifting Android applications).[3] We then denote as $f(\boldsymbol{x})$ the model that outputs a probability

[2] https://cloud.google.com/blog/topics/threat-intelligence/churning-out-machine-learning-models-handling-changes-in-model-predictions/

[3] We do not consider the *class-incremental* and *task-incremental* scenarios as we do not deal with an increasing number of unique classes or distinct tasks. We refer the reader to [9] for further details.

distribution over the two classes (e.g., $[0.1, 0.9]$, representing the probability assigned to the goodware and malware class, respectively), and denote the predicted label as $\hat{y} = \arg\max_{c \in \{0,1\}} f(\boldsymbol{x})$. Training the model sequentially amounts to finding, for each experience $\mathcal{D}_k$, a function f^k within a feasible domain $\mathcal{F}$ that minimizes the error on the current experience as well as on past ones. To achieve this goal, previous works proposed different types of CL strategies, which can be grouped into three categories: (i) *replay-based* methods, *regularization-based* methods, or (iii) *parameter isolation-based* methods [5]. In this work, we focus on the first two, as the latter is rarely considered for domain incremental scenarios. Replay-based methods store past samples in a limited buffer $\mathcal{D}_B$ and *replay* them when training each experience [10]. Regularization-based methods focus instead on regularizing the loss function to preserve previous knowledge by constraining model parameters important for previous experiences [11] or by using knowledge-distillation loss w.r.t. the previous model [12].

2.2. Regression in Model Updates

When updating an ML model to follow the drift, the newly-updated model introduces new mistakes on samples that were correctly predicted before the update, which have been referred to as *negative flips* (NFs) in [8]. Let us consider a single update scenario in which a model f^{old} is substituted with a model f^{new} having better performance. We can then measure the *negative flip rate* (NFR) on the dataset $\mathcal{D}$ as follows:

$$\text{NFR} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(\hat{y}_i^{new} \neq y_i \wedge \hat{y}_i^{old} = y_i), \tag{1}$$

where N is the number of samples in $\mathcal{D}$, $\mathbb{I}$ is an indicator function that outputs 1 if the input statement is true and 0 otherwise, and $\hat{y}_i^{new}$ and $\hat{y}_i^{old}$ are the predictions assigned by f^{new} and f^{old} (respectively) to the input sample $\boldsymbol{x}_i$ with true label y_i.

To reduce NFR, Yan et al. [8] proposed *Positive Congruent Training* (PCT), which includes a regularization term $\mathcal{L}_{PC}$ to the classification loss $\mathcal{L}$ (e.g., the cross-entropy loss):

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{N} \mathcal{L}(y_i, \boldsymbol{x}_i) + \lambda \mathcal{L}_{PC}(f(\boldsymbol{x}_i), f^{old}(\boldsymbol{x}_i)), \tag{2}$$

where f can be initialized as f^{new} before training if available beforehand, and $\mathcal{L}_{PC}$ penalizes changes in the decision function in the regions containing samples that f^{old} already classified correctly. The $\mathcal{L}_{PC}$ term is referred to as *focal distillation*, and can be formally defined as:

$$\mathcal{L}_{PC} = \sum_{i=1}^{N} \left[\alpha + \beta \cdot \mathbb{I}\left(\hat{y}_i^{old} = y_i\right)\right] \mathcal{L}_D\left(f(\boldsymbol{x}_i), f^{old}(\boldsymbol{x}_i)\right), \tag{3}$$

where α is the base weight applied to all samples in the training set, β is the additional weight applied to the samples that are correctly predicted by the old model, and $\mathcal{L}_D$ is a generic knowledge-distillation loss. As suggested in [8], we can set $\mathcal{L}_D$ as the Euclidean distance between the probabilities of f and f^{old} as detailed in Equation 4:[4]

$$\mathcal{L}_D(f(\boldsymbol{x}), f^{old}(\boldsymbol{x}) = \frac{1}{2} \left\| f(\boldsymbol{x}) - f^{old}(\boldsymbol{x}) \right\|_2^2. \tag{4}$$

3. Mitigating Regression in Continual Learning

In this section, we (i) adapt the NFR metric to be used in a CL setting and (ii) present our PCT-replay to mitigate forgetting while learning incrementally to follow the drift.

[4]This is denoted in [8] as *logit matching* as it promotes the output logits to be as similar as possible in terms of Euclidean distance.

Measuring NFR in CL. CL strategies incorporate new information while striving to preserve existing knowledge, offering a promising solution to the problem of catastrophic forgetting. However, forgetting is typically measured as an average metric (e.g., the average classification accuracy on past experiences) and does not sufficiently account for sample-wise performance such as NFR. To this end, we adapt Equation 1 and define the NFR^k as follows:

$$\text{NFR}^k = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}(\hat{y}_i^k \neq y_i \wedge \hat{y}_i^{k-1} = y_i), \tag{5}$$

which measures the percentage of NFs on the testing experience $\mathcal{D}_{(k-1)}$ when comparing the predictions of model f^{k-1} to the newly-updated model f^k trained on the latest experience $\mathcal{D}_k$.

PCT-replay. To reduce NFR^k when learning the k-th experience $\mathcal{D}_k$, we employ the objective in Equation 3 and include a rehearsal memory $\mathcal{D}_M$ containing M samples collected from previous experiences, as done in [10]. We can then formally define the PCT-replay objective as follows:

$$\min_{f\in\mathcal{F}} \sum_{i=1}^{N+M} \mathcal{L}(y_i, \boldsymbol{x}_i) + \lambda\mathcal{L}_{PC}(f(\boldsymbol{x}_i), f^{old}(\boldsymbol{x}_i)), \tag{6}$$

where $(\boldsymbol{x}_i, y_i)$ are now sampled from the $\mathcal{D}_k \cup \mathcal{D}_M$. After training on the k-th experience, we replace some of the samples in $\mathcal{D}_M$ with the ones contained in $\mathcal{D}_k$, which allows us to repeat the process when learning on the subsequent experience $\mathcal{D}_{k+1}$ This enables minimizing NFR on each experience while retaining previous knowledge thanks to the rehearsal memory.

4. Experiments

We now quantify the ability of our PCT-replay to reduce the regression of performance while learning continually with minimal amount of forgetting.

4.1. Experimental Setup

Dataset. We conducted our experiments on the ELSA dataset, which contains Android applications collected from the AndroZoo repository.[5] Each application is labeled as malicious if detected by at least 10 detectors from VirusTotal,[6] and benign if the number of detections is zero. It is instead discarded if the number of detections is between 1 and 9 to avoid including applications for which the true label is uncertain. We consider 75000 applications (67500 goodware and 7500 malware), spanning from January 2017 to December 2019. According to the evaluation guidelines presented in [4], we consider a 3-months time window containing 6,250 for each experience. We then dedicate 5000 samples for training on each experience and leave the rest to test the performance (as detailed below). We use the first training split to extract binary features according to Drebin [1], where each one represents the presence or absence of a specific characteristic extracted from the `apk` file. We exclude all features having variance lower than 10^{-3}, resulting in a total of 4,681 features.

Model Architecture and Training. We used a multi-layer perceptron (MLP) as our main classifier, for which we set a hidden layer of size 512 and 2 outputs. We minimize the cross-entropy loss using the SGD optimizer with learning rate of 10^{-3} and momentum set to 0.9. We train the models for one epoch, and we set the batch size to 1 for all strategies except for the cumulative one, for which we set it to 12 as it aims to minimize the loss uniformly for all training experiences considered.

Evaluation Metrics. We evaluate the classification performances by considering three main metrics: (i) the *precision*, (ii) the *recall* (a.k.a. detection rate), and (iii) the F_1 *score*, which summarize the first two by taking their harmonic mean. Following common evaluation protocols in CL, we average each metric

[5] https://androzoo.uni.lu/

[6] https://www.virustotal.com/gui/home/upload

after training on the k-th experience according to two modalities: (i) *backward* mode, which quantifies the ability to retain past knowledge by taking the average on all past and current testing experiences (i.e., $\leq k$), and (ii) *forward* mode, which quantify the ability to generalize to future data by taking the average on all future testing experiences (i.e., $> k$). We then evaluate the regression when training model f_k on the k-th experience (considering samples from each class, separately) by measuring the NFR w.r.t. its previous model f_{k-1} on the $(k-1)$-th testing experience, e.g., considering the update from M_3 to M_4 we evaluate NFR on the testing experience $\mathcal{D}_3$.

CL Methods. As our main baselines to compare our method, we first consider two strategies: (i) the naïve strategy, i.e., simple retraining on new experiences without using any knowledge retention mechanism, and (ii) the cumulative strategy, i.e., retraining on new experiences by also accumulating data from previous ones, to assess the lower and upper bound of the performance, respectively. We consider five state-of-the-art CL methods: Replay with a buffer size set to 100 [10]; Averaged Gradient Episodic Memory (A-GEM) [13] with buffer size set to 300 and number of examples set to 11; Elastic Weight Consolidation (EWC) [11] with λ=0.001; Synaptic Intelligence (SI) [14] with λ=0.01 and ϵ=0.1; and Learning without Forgetting (LwF) [12] with α=0.8 and the temperature=1. We then use our PCT-replay with α=1, β=0.3, and replay buffer size set to 200. For the sake of clarity, we only compare the classification performances of our method (and the two baselines) with the two CL methods that achieve the highest F_1 score in the backward mode, on average. We only make this selection for classification performances, instead, for NFR we consider all the tested strategies.

4.2. Experimental Results

Backward Mode Evaluation. In Figure 1, we show the precision, recall and F_1 score, considering the *backward* mode. All strategies generally maintain a high level of precision across the 12 experiences, with values mostly between 70% and 90%. LwF and Replay show strong performance, highlighting their effectiveness in maintaining high precision. Between the CL strategies, PCT is the one that exhibits the best results. In terms of recall, LwF and Replay demonstrate strong and consistent performance, highlighting their effectiveness in retaining knowledge about malware and improving detection accuracy. Instead, PCT-replay performs poorly compared with the other strategies. In terms of F_1, LwF, Replay and the proposed PCT-replay provide stability and good overall performance, considering that they are trained with much less data and that computational and time costs are significantly reduced. The cumulative baseline generally performs better, indicating that retaining the entire training dataset provides higher performance according to all the considered metrics. In contrast, the naïve baseline shows the most variability, which undermines the reliability of the predictions, emphasizing the importance of employing CL techniques to maintain stable performance.

Forward Mode Evaluation. In Figure 2, we show the precision, recall and F_1 score, considering the *forward* mode. In this evaluation setting, precision shows an upward trend. Until experience 4, there is a 10% performance range, with PCT-replay outperforming all other strategies. After experience 4, there is rapid growth, and best-performing strategies, baselines, and PCT-replay are close together. From experience 4 to 10, there is a downward trend followed by growth until experience 11, with all the strategies paired together. In terms of recall, there is still a growing trend as training experiences increase. The cumulative baseline starts with a dramatic recall value close to 20%, but it rapidly grows to match the performances of the CL strategies and the naïve baseline. PCT-replay underperforms compared to the other strategies. The F_1-score shows a stable and growing trend. The cumulative baseline performs poorly in the first experience, but by experience 2, it reaches the performance level of the other strategies. PCT-replay, on the other hand, is very close to the other strategies but performs slightly worse.

Evaluating Regression. We show in Figure 3 the NFR for both the goodware (left) and the malware class (right). None of the strategies show significant NFR for the goodware class, with mean values ranging from 0.7% to 0.15%. The naïve baseline offers the worst results, while all the other CL strategies demonstrate better performance. Notably, our proposed PCT-replay strategy outperforms all other CL

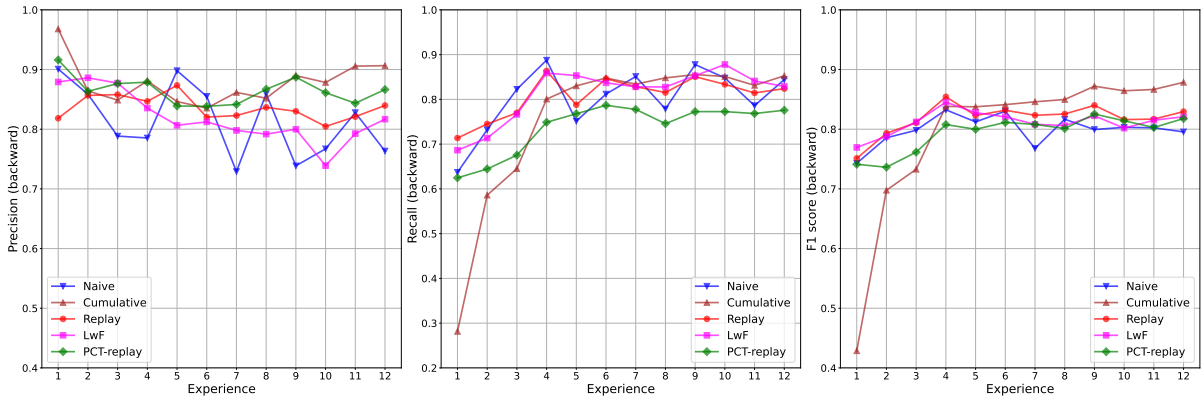

Figure 1: Average Precision, Recall and F_1 over current and past experiences. The numbers on the x-axis represent the sequential experiences, labeled from 1 to 12. Each number corresponds to the model generated after training on the training set of the specified experience $\mathbf{n}$. Models are then evaluated on the test sets of all previous experiences and the current one (from 1 to $\mathbf{n}$). The y-axis shows the average of the metric under consideration for the different models.

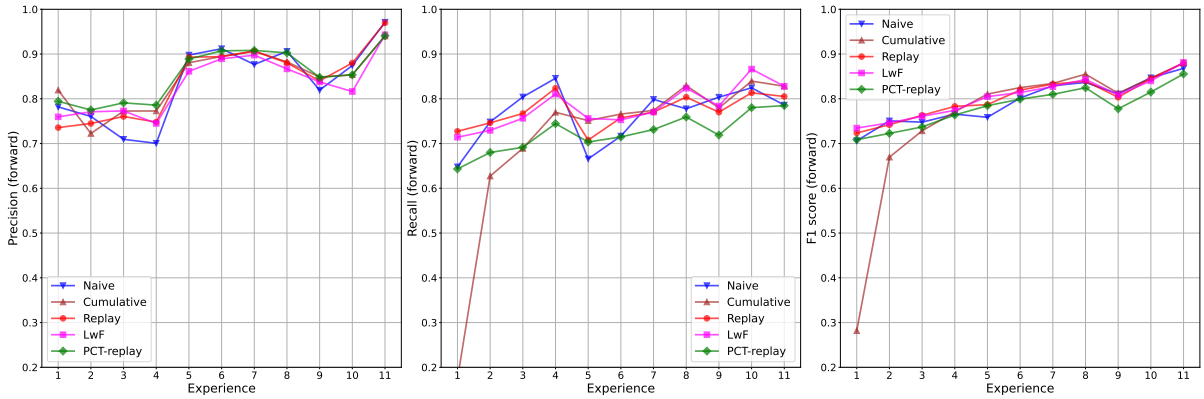

Figure 2: Average F1, Recall and Precision over future experiences. The numbers on the x-axis represent the sequential experiences, labeled from 1 to 11. In this mode, it does not make sense to train on the 12^{th} experience due to the lack of future experiences for testing. Each number corresponds to the model generated after training on the training set of the specified experience $\mathbf{n}$. Models are then evaluated on the test sets of all future experiences. The y-axis shows the average of the metric under consideration for the different models.

strategies, including the cumulative baseline, with a mean NFR of 0.15%. This result provides initial evidence of the effectiveness of PCT-replay strategy in mitigating the phenomenon of regression. For the malware class, instead, the impact of CL strategies is more notable. The naïve baseline reaches a maximum of 15.8% NFR. EWC, Replay, LwF, AGEM, and SI demonstrate that CL strategies are effective in mitigating regression caused by negative flips, with LwF having a mean NFR of 3.26%. Our PCT-replay strategy outperforms all other approaches, including the cumulative baseline, with a mean NFR of 1.18%. This further reflects the ability of the PCT-replay to preserve the knowledge of the previous model when it was correct.

Discussion. In summary, our analysis of the various CL strategies reveals critical insights into their knowledge retention effectiveness. The cumulative baseline, while demonstrating superior performance due to its retention of all previously seen data, incurs higher computational and time costs. naïve baseline offers good results as well, suggesting that, despite the realistic 9:1 goodware to malware ratio setting, the samples in the dataset do not fully capture realistic changes in data distributions over experiences. Nevertheless, it exhibits significant fluctuations, underscoring the necessity of incorporating knowledge retention mechanisms to ensure model stability and reliability. The relatively lower performance of PCT-replay in recall metric, compared to the other strategies, may be due to the replay buffer selecting samples from previous experiences randomly. Given the 9:1 goodware to malware proportion, it is

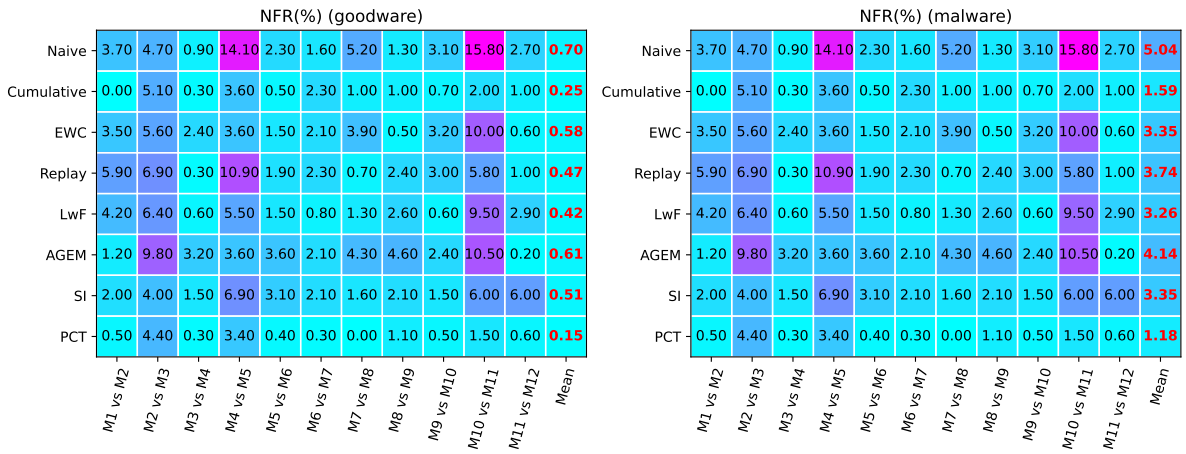

Figure 3: NFR for goodware and malware. The columns in the heatmaps represent the comparison between adjacent models. Model $\mathbf{n-1}$ vs Model $\mathbf{n}$, obtained respectively by training on experiences $\mathbf{n-1}$ and $\mathbf{n}$. NFR is computed on the test set of experience $\mathbf{n-1}$. Each row shows the strategy under consideration. The last column represents the mean of row values.

likely that the majority of stored samples are goodware, causing the strategy to primarily preserve the knowledge of the goodware class. It can be said that none of the strategies achieve high performance in detecting malware samples. This could be attributed to the unbalanced nature of the dataset, which, on the other hand, reflects a real-world scenario where goodware samples significantly outnumber malware samples. LwF, Replay and our PCT-replay show good stability and performance, offering a balanced trade-off between performances and resource efficiency. These findings underscore the importance of evaluating multiple metrics to gain a comprehensive understanding of each strategy's strengths and limitations in handling unbalanced datasets and maintaining robust performance over time.

In the *forward* mode, precision shows worse results compared to precision in the *backward* mode. This suggests that while CL strategies are able to retain knowledge and preserve precision on old experiences, they struggle to achieve good precision on unseen data. In terms of recall, our PCT-replay underperforms with respect to the other strategies, aligning with the conclusions in *backward* setting. It is possible to say that, although CL strategies primarily focus on mitigating the phenomenon of catastrophic forgetting, this does not compromise the models' ability to generalize to unseen data. The evaluation results indicate that these strategies not only help retain knowledge from previous experiences but also enable the models to adapt effectively to new, unseen data. This dual capability is crucial for applications such as malware detection, where models must continuously learn from evolving data distributions while maintaining high performance on previously encountered data.

PCT-replay consistently outperforms other CL strategies in mitigating *regression* and maintaining stable performance across updates. For both goodware and malware classes, it demonstrates a superior ability to preserve the knowledge of the previous models, highlighting its effectiveness and reliability in CL scenarios. With respect to the results obtained in *backward* evaluation, it can be said that the PCT strategy loses some performances in terms of detection rate, but, on the other hand, it is very good at preventing negative flips, ensuring that the knowledge of the old model is positively transferred to the updated one.

5. Related Work

We now discuss the state-of-the-art related to our work and highlight the main differences.

Continual Learning for Malware Detection. Few works have applied CL methods in the malware

detection domain. Rahman et al. [6] investigate the application of 11 CL techniques to malware classification tasks. It explores their performance across task, class, and domain incremental learning scenarios, using realistic EMBER and Drebin datasets to evaluate their effectiveness in handling evolving malware threats. Kou et al. [15] propose SSCL-TransMD a transformer-based semi-supervised continual learning model for malware detection. It handles evolving malware by combining new and historical samples and leverages pseudo-labeling to improve the detection of emerging variants. MalFSCIL [16] introduces a method that enables malware detection systems to incrementally learn new malware classes from limited samples without forgetting previously learned classes. By leveraging few-shot learning techniques, the approach addresses the challenges of class imbalance and the dynamic nature of malware evolution, enhancing the adaptability and robustness of detection models. The work in [17] addresses malware detection from a continual semi-supervised one-class learning perspective, relying only on normal/benign data. It focuses on the application of two replay strategies on anomaly detection models. Sun et al. [18] propose a novel approach to adapt malware classifiers to temporal shifts by training on chronologically organized data. Their method leverages multimodal features of malware and claims to enable efficient updates.

None of the existing works explore the role of regression in a CL setting, and we are the first to formulate and quantify this problem in contrast to catastrophic forgetting.

Reducing Regression. In our work, we focus on the regression problem that hinders model updates. Previous works tackled this issue by either employing weight regularization [8] (as also discussed in section 2) or ensembles [19, 20, 21]. Zhao et. al [19] propose ELODI a method to reduce negative flips during model updates by using ensembles as references. It employs Logit Difference Inhibition (LDI) to align a student model with ensemble predictions while improving accuracy. Instead, [20] proposes BCWI to reduce negative flips during model updates by interpolating between old and new model weights. This method maintains backward compatibility while preserving the accuracy improvements of the updated model. Gated Fusion [21] uses a learnable gating mechanism to blend predictions from old and new models. The gate decides whether to prioritize backward compatibility (old model) or improvements (new model).

None of these works address the regression problem within a CL scenario, nor do they extend the concept to the domain of malware detection. Moreover, we do not consider methods that make use of ensemble while training, as they typically involve higher computational costs and increased complexity, making them less practical. However, our methodology can still apply to other regression-aware formulations as well.

6. Conclusions and Future Work

In this work, we propose PCT-replay, the first Regression-aware CL algorithm that combines the focal distillation in [8] with the replay-based CL method in [10]. We designed this approach to mitigate the regression problem by preserving consistency with previous predictions while enabling the model to adapt to new data. Our strategy effectively mitigates the problem of regression, reaching a mean NFR of 1.18% on the malware class, outperforming all the other strategies even the cumulative baseline. On the other hand, experiments in *backward* mode reveal that our PCT-replay is also effective in retaining past knowledge, offering comparable performances with the other CL strategies.

However, this work presents some limitations that we intend to tackle in future research. First, the considered training pipeline requires that all training data at each experience is readily labeled. Unfortunately, this is not the case in many real-world applications, especially in the malware domain, for which obtaining a single label from manual inspection requires a considerable amount of time. Second, we might have different outcomes when considering benign and malicious applications coming from different sources and with a higher level of uncertainty in their label (e.g., also including samples for which the number of detections in VirusTotal is between 1 and 9). As future research directions, we aim to extend our methodology to encompass not only new emerging CL-based malware detection strategies but also novel regression-aware learning algorithms.

This work offers a promising path forward for developing trustworthy and scalable Android malware detection systems that can incrementally include new knowledge while minimizing the regression that hinders each model update and, as a consequence, the trust each user relies on such systems.

Acknowledgments

This work was partially supported by project SERICS (PE00000014) and FAIR (PE00000013, CUP: J23C24000090007) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU. This work has been conducted while Daniele Ghiani was enrolled in the Italian National Doctorate on AI run by Sapienza University of Rome in collaboration with the University of Cagliari.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, C. Siemens, Drebin: Effective and explainable detection of android malware in your pocket., in: Ndss, volume 14, 2014, pp. 23–26.

[2] E. Mariconti, L. Onwuzurike, P. Andriotis, E. D. Cristofaro, G. Ross, G. Stringhini, Mamadroid: Detecting android malware by building markov chains of behavioral models, 2017. arXiv:1612.04433.

[3] G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen, F. Petitjean, Characterizing concept drift, Data Mining and Knowledge Discovery 30 (2016) 964–994.

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

[5] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, T. Tuytelaars, A continual learning survey: Defying forgetting in classification tasks, IEEE transactions on pattern analysis and machine intelligence 44 (2021) 3366–3385.

[6] M. S. Rahman, S. Coull, M. Wright, On the limitations of continual learning for malware classification, in: Conference on Lifelong Learning Agents, PMLR, 2022, pp. 564–582.

[7] W. E. Wong, J. R. Horgan, S. London, H. Agrawal, A study of effective regression testing in practice, in: ISSRE, IEEE Computer Society, 1997, pp. 264–274.

[8] S. Yan, Y. Xiong, K. Kundu, S. Yang, S. Deng, M. Wang, W. Xia, S. Soatto, Positive-congruent training: Towards regression-free model updates, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 14299–14308.

[9] G. M. Van de Ven, T. Tuytelaars, A. S. Tolias, Three types of incremental learning, Nature Machine Intelligence 4 (2022) 1185–1197.

[10] D. Rolnick, A. Ahuja, J. Schwarz, T. P. Lillicrap, G. Wayne, Experience replay for continual learning, in: NeurIPS, 2019, pp. 348–358.

[11] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al., Overcoming catastrophic forgetting in neural networks, Proceedings of the national academy of sciences 114 (2017) 3521–3526.

[12] Z. Li, D. Hoiem, Learning without forgetting, IEEE transactions on pattern analysis and machine intelligence 40 (2017) 2935–2947.

[13] A. Chaudhry, M. Ranzato, M. Rohrbach, M. Elhoseiny, Efficient lifelong learning with a-gem, arXiv preprint arXiv:1812.00420 (2018).

[14] F. Zenke, B. Poole, S. Ganguli, Continual learning through synaptic intelligence, in: International conference on machine learning, PMLR, 2017, pp. 3987–3995.
[15] L. Kou, D. Zhao, H. Han, X. Xu, S. Gong, L. Wang, Sscl-transmd: Semi-supervised continual learning transformer for malicious software detection, Applied Sciences 13 (2023) 12255.
[16] Y. Chai, X. Chen, J. Qiu, L. Du, Y. Xiao, Q. Feng, S. Ji, Z. Tian, Malfscil: A few-shot class-incremental learning approach for malware detection, IEEE Transactions on Information Forensics and Security (2024).
[17] M. Chin, R. Corizzo, Continual semi-supervised malware detection, Machine Learning and Knowledge Extraction 6 (2024) 2829–2854.
[18] T. Sun, N. Daoudi, W. Pian, K. Kim, K. Allix, T. F. Bissyandé, J. Klein, Temporal-incremental learning for android malware detection, ACM Transactions on Software Engineering and Methodology (2024).
[19] Y. Zhao, Y. Shen, Y. Xiong, S. Yang, W. Xia, Z. Tu, B. Schiele, S. Soatto, Elodi: Ensemble logit difference inhibition for positive-congruent training, IEEE Transactions on Pattern Analysis and Machine Intelligence (2024).
[20] R. Schumann, E. Mansimov, Y.-A. Lai, N. Pappas, X. Gao, Y. Zhang, Backward compatibility during data updates by weight interpolation, arXiv preprint arXiv:2301.10546 (2023).
[21] Y.-A. Lai, E. Mansimov, Y. Xie, Y. Zhang, Improving prediction backward-compatiblility in nlp model upgrade with gated fusion, arXiv preprint arXiv:2302.02080 (2023).