Regression-aware Continual Learning for Android Malware Detection

Daniele Ghiani[1,2], Daniele Angioni[1,4], Giorgio Piras[1], Angelo Sotgiu[1], Luca Minnei[1], Srishti Gupta[1,2], Maura Pintor, *Member, IEEE*[1], Fabio Roli *Fellow, IEEE*[1,3,4], and Battista Biggio, *Fellow, IEEE*[1,4]

[1]Department of Electrical and Electronic Engineering, University of Cagliari, Italy
[2]Department of Computer, Control and Management Engineering, Sapienza University, Rome, Italy
[3]Department of Informatics, Bioengineering, Robotics, and Systems Engineering, University of Genoa, Italy
[4]CINI, Italy

***Abstract*—Malware evolves rapidly, forcing machine learning-based detectors to be continuously updated. With antivirus vendors processing hundreds of thousands of new samples daily, datasets can grow to billions of examples, making full retraining impractical. *Continual learning* (CL) has emerged as a scalable alternative, enabling incremental updates without full data access while mitigating *catastrophic forgetting*. In this work, we analyze a critical yet overlooked issue in this context: *security regression*. Unlike forgetting, which manifests as a drop in average performance on previously seen data, security regression captures harmful sample-level prediction changes, e.g., malware samples that were correctly detected before an update but evade detection afterward. This poses serious risks in security-critical applications, as the silent reintroduction of previously detected threats may undermine users' trust in the update process, leading them to perceive a regression in security even if the average model performance has actually improved. We first formalize and quantify security regression in CL-based malware detectors, revealing that up to 3-6% of malware experience it after model updates. We then address this issue by introducing a regression-aware framework to the CL setting. Specifically, we instantiate it via Positive Congruent Training (PCT), showing seamless integration with any prior CL strategy. Experiments on the ELSA, Tesseract, and AZ-Class datasets show that our method effectively halves regression across different CL scenarios while maintaining strong detection performance over time.**

***Index Terms*—Android Malware, Continual Learning, Negative Flips, Regression Testing**

I. Introduction

The detection of malicious software (a.k.a. *malware*) is a central aspect, as it helps prevent the exploitation of software vulnerabilities and protects the integrity of the entire software security chain. Malware detection is nowadays commonly entrusted to Machine Learning (ML) models, which, by leveraging large amounts of data, are able to learn complex correlations and patterns, ultimately enabling automatic detection [1], [2]. However, ML models are primarily designed to function under the assumption of stationary data distributions, i.e., data that remain consistent over time. This assumption does not hold in the "wild", where detectors can be impacted by a phenomenon known as *concept drift*, i.e., a change in the data distribution over time. In particular, existing malware is progressively modified to evade detection, while the emergence of new malware families further challenges models' predictions and security measures by introducing completely new patterns. Legitimate applications (a.k.a. *goodware*) are updated as well, but this typically has a minimal effect on the detector's performance [3], [4]. Due to this drift, ML-based detectors suffer from rapid performance degradation over time [3]. To address this degradation, the model must be regularly updated with new data. One option is to retrain it using both newly collected and previous samples. However, this quickly becomes infeasible as the dataset grows over time. For example, the AV-TEST Institute observes $\sim 450,000$ new malware samples each day, while VirusTotal processes over a million files daily.[1] An alternative approach is to retrain the detector with only newly collected data. However, this often causes the resulting detector to entirely prioritize adaptation to recent distributions, at the cost of disregarding "previously acquired knowledge". This issue is commonly known as *catastrophic forgetting* [5]. In this regard, Continual Learning (CL) addresses the forgetting problem, i.e., preserving performance on previous data [5]. While multiple CL methods exist [5], only a few works apply them to malware detection [6]–[9].

In addition to catastrophic forgetting, model updates in CL scenarios can lead to a distinct and overlooked issue known as *regression* [10]. This phenomenon refers to a degradation in the model's behavior on specific individual samples, particularly those that were correctly classified before an update but are misclassified afterward. Unlike forgetting, regression can occur even when the overall performance of the new model improves: an updated model may report better detection while introducing regression on previously well-learned distributions. This is because train and update procedures are typically designed to minimize an aggregate loss, with no explicit constraints to preserve the model's predictions on specific samples that were already correct. In security applications, this issue is particularly severe, as model updates may cause previously detected malware samples to evade detection again [11], thereby undermining user trust and creating the perception of a security drop.[2] We refer to this phenomenon as *security regression*. While recent work addresses regression on

[1]https://www.av-test.org/en/, https://www.virustotal.com/gui/
[2]https://cloud.google.com/blog/topics/threat-intelligence/churning-out-machine-learning-models-handling-changes-in-model-predictions/

computer vision and natural language domains [10], [12], they: (i) overlook security-related tasks such as malware detection; and (ii) do not consider CL settings, where the lack of past data further complicates retaining prediction consistency.

In this work, we fill this gap by analyzing the security regression phenomenon of CL strategies for malware detection. Building on our initial findings in [13],[3] we formalize this phenomenon in the context of CL for malware detection, and introduce a regression-aware training framework that can be integrated into existing CL strategies. As an instantiation of this framework, we employ *Positive Congruent Training* (PCT) [10], which jointly minimizes the classification loss and a regularization term designed to mitigate regression. We evaluate our approach on three Android malware datasets (ELSA, TESSERACT, and AZ-Class), considering five representative CL strategies and four different state-of-the-art malware detectors, each obtained as a combination of Drebin [1] or APIGraph [2] feature extractors with linear SVM or MLP models. Our analysis shows that incrementally updating ML models with CL strategies introduces security regression, i.e., up to 3–6% of previously detected malware samples are misclassified after model updates. Integrating PCT into these strategies consistently mitigates this issue, reducing regression by up to 50% across datasets and scenarios while maintaining competitive detection performance in most cases.

We summarize our contributions as follows: (i) we formalize the notion of security regression in CL for malware detection as an additional side effect besides catastrophic forgetting; (ii) we show empirically that security regression can occur even when forgetting is mitigated, affecting up to 3-6% of previously detected malware after model updates; and (iii) we introduce a regression-aware training framework instantiated via Positive Congruent Training (PCT) that can be integrated into existing CL pipelines, mitigating regression across datasets, detectors, and CL strategies.

II. Continual Learning for Malware Detection

CL algorithms update ML models with new data while addressing *catastrophic forgetting*, i.e., forgetting knowledge from past data. In this section, we discuss CL scenarios and strategies, with a specific focus on malware detection.

Learning Incrementally. CL can be formalized as learning from a sequence of K sets of data samples $\{\mathcal{D}_k\}_{k=1}^{K}$, called *experiences*, assuming no access to past data when learning new experiences. Each experience consists of a dataset $\mathcal{D}_k = \{\boldsymbol{x}_i^k, y_i^k\}_{i=1}^{n^k}$, where $\boldsymbol{x}_i^k \in \mathcal{X}^k$ is an input sample, $y_i^k \in \mathcal{Y}^k$ its ground-truth label, and n^k is the number of samples in the k-th experience. The CL literature distinguishes three main settings: (i) *domain-incremental*, (ii) *class-incremental*, and *task-incremental* learning [14]. In *domain-incremental learning* (DIL), the input distribution varies across experiences, i.e., $p(\boldsymbol{x}^k) \neq p(\boldsymbol{x}^{k+1})$, while the label space remains fixed, that is $\mathcal{Y}^k = \mathcal{Y}^{k+1}$. In *class-incremental learning* (CIL), the change affects the label space, hence $\mathcal{Y}^k \neq \mathcal{Y}^{k+1}$. Each experience $\mathcal{D}_k$ introduces new class labels, expanding the set of observed classes, $\hat{\mathcal{Y}}^k = \cup_{j=1}^{k} \mathcal{Y}^j$. While the shift originates in the label space, the posterior distribution $p(\boldsymbol{x}|y)$ for the new classes is also influenced, making it more challenging than DIL. In *task-incremental learning* (TIL), each experience requires task labels at test time, which is unrealistic for malware detection. For this reason, it is not considered in this study.

Mitigating Catastrophic Forgetting. As the model optimizes its parameters based on the current experience $\mathcal{D}_k$, it overrides knowledge from previous experiences $\mathcal{D}_j$, with $j < k$, causing forgetting. Forgetting after the k-th update is measured as:

$$F_j^k = \max_{o \in \{1,\dots,k-1\}} \mathcal{A}_j^o - \mathcal{A}_j^k, \quad \forall j < k\,, \tag{1}$$

where $\mathcal{A}$ is a performance metric (typically the accuracy). Given a model trained on the k-th training experience, the first term on the right-hand side of the equation indicates the best performance among old updates on the experience j, while the second term indicates the performance of the model trained on the k-th experience but evaluated on the experience j.

CL strategies address forgetting by learning, at each experience k, an optimal parametrization $\boldsymbol{\theta}_k$ that generalizes to both current and past data, despite limited or no access to previous training experiences. They are typically grounded into three approaches [5]: (i) *replay-based*, (ii) *regularization-based*, or (iii) *parameter isolation-based*. *Replay methods* alleviate forgetting by either storing a subset of past data or generating pseudo-samples with a generative model. These samples are then replayed during training on new experiences, either to reinforce previous knowledge through rehearsal [15] or to guide the optimization process in a way that minimizes interference with past tasks, as in Average-Gradient Episodic Memory (A-GEM [16]). *Regularization methods* reduce memory requirements and prioritize privacy by avoiding storing past data. They introduce additional loss terms constraining the update of parameters critical for past experiences. For instance, Elastic Weight Consolidation (EWC [17]) and Synaptic Intelligence (SI [18]) estimate the importance of each parameter and penalize their deviation from previous optima. *Parameter-isolation methods* prevent forgetting by dedicating different model parameters to each experience [19], [20]. Since the latter methods are mostly used for TIL scenarios, we focus only on replay and regularization methods.

Framing CL for Malware Detection. We frame the DIL and CIL scenarios in the context of malware classification. In DIL, the input distribution evolves over time, while the label space remains fixed. This is the case for the changes in the benign/malicious software landscape, while the classification task still remains to discern malware from goodware. Given that applications are typically associated with a timestamp t describing the first appearance date, we can describe concept drift, similarly to [3], by defining the k-th experience $\mathcal{D}_k = \{\boldsymbol{x}_i^k, y_i^k, t_i^k\}_{i=1}^{n^k}$ over a time interval $\Delta t^k = [t^k, t^{k+1})$, where t_i^k denotes the timestamp of each sample. In CIL, the label space expands across experiences, with each experience introducing a distinct set of classes. In the malware domain, this corresponds to the fine-grained task of distinguishing *mal-*

[3]While [13] consists of an initial exploration of the regression phenomena in CL for malware detection, in this work we provide a consolidated and significantly extended treatment, supported by a broader experimental evaluation and deeper analysis.

ware families. In this case, the model must learn new families while retaining performance on previous ones. Formally, the label set grows over time such as $\mathcal{Y}^k \subset \mathcal{Y}^{k+1}$, and the label distribution shifts accordingly: $P(\mathcal{Y}^k) \neq P(\mathcal{Y}^{k+1})$.

III. Security Regression in Continual Learning

Although CL mitigates forgetting, detector updates can lead to fine-grained performance degradation in previously correctly classified samples, known as *regression* [10], which we frame in the malware detection context as *security regression*.

A. Security Regression

Security regression is particularly problematic in malware detection, as reopening previously patched vulnerabilities can have serious consequences. In fact, while forgetting accounts for an overall degradation in detector performance, operating in such scenarios imposes special attention on previously detected malware samples that, after updates, bypass detection systems. This is particularly relevant in the Android domain, where a relatively small number of families dominate the scene for several years, continuously generating new polymorphic variants [21], [22]. Within this temporal horizon, old malware samples remain representative of real-world threats, as they can be reused or slightly modified rather than replaced. Antivirus systems often combine signature-based detection with ML: while signatures guarantee stable predictions for previously observed malware, they fail when faced with slight modifications [23], [24], making ML-based detectors essential for generalizing beyond exact signatures. As these models are periodically updated, changes in predictions across model versions may cause security regression. Thus, quantifying and addressing such inconsistencies becomes crucial.

Formal Definition. To formally characterize security regression, we rely on the general notion of *negative flips* (NFs) provided in [10], which captures prediction inconsistencies between model updates. Let f_{old} and f_{new} denote, respectively, a model before and after an update, and let $(\boldsymbol{x}, y)$ be a sample with its ground truth label. A NF occurs if $f_{old}(x) = y$ and $f_{new}(x) \neq y$, i.e., a shift from correct to incorrect classification of the same sample after the update. Being $\mathbb{1}$ the indicator function, the fraction of NFs on a dataset $\mathcal{D}$ of N samples is referred to as *negative flip rate* (NFR):

$$\text{NFR} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(f_{new}(x_i) \neq y_i \wedge f_{old}(x_i) = y_i). \quad (2)$$

B. Security Regression in CL

In a CL scenario, the model is incrementally updated over a sequence of experiences $\{\mathcal{D}_k\}_{k=1}^{K}$. Each update yields a new model parameterization, resulting in a sequence of models $\{f_k\}_{k=1}^{K}$, where f_k denotes the model after training on experience $\mathcal{D}_k$. To quantify *security regression* across updates, we can adapt Eq. (2) to the CL case as follows:

$$\text{NFR}_j^k = \frac{1}{n^j} \sum_{i=1}^{n^j} \mathbb{1}(f_k(x_i) \neq y_i \wedge f_{k-1}(x_i) = y_i) \quad (3)$$

where NFR_j^k denotes the NFR for the k-th update $f_{k-1} \rightarrow f_k$ computed on the j-th experience composed by n^j samples. This allows comparing the predictions of the newly updated model f_k with its previous version f_{k-1} on a selected test experience $\mathcal{D}_j$, and measuring the fraction of samples that were correctly classified by f_{k-1} but misclassified by f_k. We evaluate NFR_j^k after learning the k-th experience according to two modes: (i) the *backward*, and (ii) the *forward mode*.

Backward Mode. NFR in backward mode measures how the newly updated model f_k performs on earlier experiences compared to its predecessor f_{k-1}, and is computed as:

$$\text{NFR}_B^k = \frac{1}{k'} \sum_{j=1}^{k'} \text{NFR}_j^k \ , \quad (4)$$

where $k' = k$ for DIL, and $k' = k - 1$ for CIL, to ensure evaluation is restricted to classes known by both f_k and f_{k-1}. This aligns with standard protocols for measuring forgetting, tracking degradation on previously seen data after each update. We use this setting to measure *security regression* on past data.

Forward Mode. NFR in forward mode compares the updated model f_k to f_{k-1}, on data from the current and future experiences, highlighting early signs of degraded generalization to unseen future data. The forward NFR is computed as:

$$\text{NFR}_F^k = \frac{1}{(K - k + 1)} \sum_{j=k}^{K} \text{NFR}_j^k \ . \quad (5)$$

Forward evaluation is specific to DIL, where classes are fixed, and input distributions drift. We employ it to measure models' adaptability to future data distributions drifting over time.

C. Forgetting vs. Regression

We now analyze its relation and difference with the forgetting phenomenon and highlight why both should be considered when assessing update quality. Forgetting refers to a decline in model performance on previous experiences after the update. Unlike forgetting, regression captures fine-grained degradations in model behavior, thus capturing instances where the model regresses in specific predictions (i.e., focusing on negative flips), even when the overall performance improves. Therefore, regression can be defined as a sample-wise measure, rather than an average, global one, as it provides a fine-grained assessment of prediction consistency over time. A model might not exhibit forgetting and even improve the overall performance while still suffering from regression.

To formally analyze the relation between forgetting and regression, we first define the Positive Flip Rate (PFR), which follows a similar rationale to the NFR definition in Eq. (3):

$$\text{PFR}_j^k = \frac{1}{n^j} \sum_{i=1}^{n^j} \mathbb{1}(f_k(x_i) = y_i \wedge f_{k-1}(x_i) \neq y_i) \, , \quad (6)$$

where, conversely to NFR, PFR quantifies the percentage of samples previously misclassified by f_{old} and that the new model f_{new} corrected. Let us now focus on the model update $f_{k-1} \rightarrow f_k$, evaluated on the experience $j = k - 1$ (in the following, we omit the subscripts to improve readability).

This article has been accepted for publication in IEEE Transactions on Information Forensics and Security. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2026.3714132

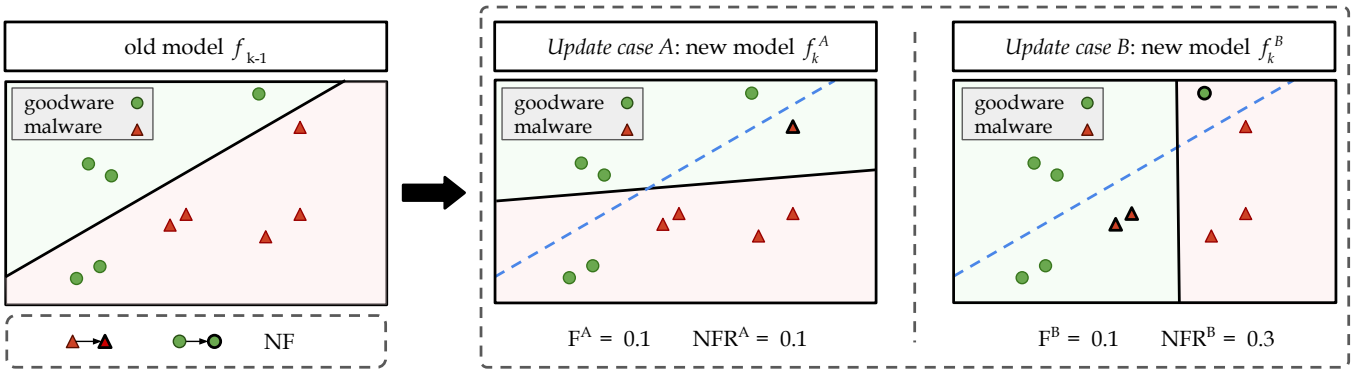


Fig. 1. Forgetting vs. Regression. Starting from the old model f_{k-1} (*left*), we show two updates, f_k^A and f_k^B (*right*), with identical forgetting ($\mathcal{F} = \text{NFR} - \text{PFR}$) but different regression: case A has 1/10 negative flips, case B has 3/10.

Assuming that f^{k-1} is the best-performing model among the previous ones (hence removing the $\max$ operator), the forgetting in Eq. (1) takes the form of a simple difference in accuracy between the two considered models. We can thus express the accuracy of model k through the regression notation, i.e., as the balance between NFR and PFR, yielding:

$$\begin{aligned} F^k &= \mathcal{A}^{k-1} - \mathcal{A}^k = \mathcal{A}^{k-1} - (\mathcal{A}^{k-1} - \text{NFR}^k + \text{PFR}^k) \\ &= \text{NFR}^k - \text{PFR}^k \end{aligned} \tag{7}$$

Hence, forgetting reflects the balance of gains and losses, while regression focuses solely on degradations and can occur even when forgetting is zero. Clearly, this is particularly relevant in security-critical tasks, where negative shifts can have disproportionate consequences. We provide an overview of such a difference in Fig. 1, illustrating a toy example with two update scenarios that share the same overall forgetting but differ significantly in terms of regression.

IV. Mitigating Security Regression

To mitigate security regression, we leverage Positive Congruent Training (PCT) [10]. We first describe PCT and then show how it can be integrated into CL strategies for Android malware detection.

Positive Congruent Training (PCT). PCT aims to find a model f with reduced regression from a previous parameterization f_{old}. This is achieved by penalizing incorrect prediction changes of single samples through a distillation-like regularization term, $\mathcal{L}_{PC}$, which is added to the training loss $\mathcal{L}$ (e.g., the cross-entropy loss):

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{N} \mathcal{L}(y_i, \boldsymbol{x}_i) + \lambda \mathcal{L}_{PC}(f(\boldsymbol{x}_i), f_{old}(\boldsymbol{x}_i)), \tag{8}$$

where $\mathcal{L}_{PC}$ penalizes changes in the decision function regions containing samples that f_{old} already classified correctly, while λ is a hyperparameter controlling the regularization strength. The term $\mathcal{L}_{PC}$ is the *focal distillation*, defined as:

$$\mathcal{L}_{PC} = \sum_{i=1}^{N} \left[\alpha + \beta \cdot \mathbb{1}\left(f_{old}(x_i) = y_i\right)\right] \mathcal{L}_D\left(f(\boldsymbol{x}_i), f_{old}(\boldsymbol{x}_i)\right), \tag{9}$$

where $\mathbb{1}$ is the indicator function, α is a base factor applied to all samples in the training set, while β is a factor weighting samples correctly predicted by the old model, and $\mathcal{L}_D$ is a generic knowledge distillation loss. In practice, we employ the loss denoted as *focal distillation with logit matching* (FD-LM) in [10], which can be formulated as the squared Euclidean distance between the outputs of f and f_{old} as follows:

$$\mathcal{L}_D(f(\boldsymbol{x}), f_{old}(\boldsymbol{x})) = \frac{1}{2} \left\| f(\boldsymbol{x}) - f_{old}(\boldsymbol{x}) \right\|_2^2 . \tag{10}$$

Regression-aware CL through PCT. Mitigating regression in CL strategies amounts to adding the PCT regularization term $\mathcal{L}_{PC}$ described in Eq. (9), scaled through a hyperparameter λ, to the training loss. This extension is agnostic of the CL strategy and promotes prediction consistency on previously correct samples without altering the CL formulation, making it a versatile plug-in against regression. For replay methods, $\lambda\mathcal{L}_{PC}$ can directly be added to the training objective. In regularization methods, the CL strategy already consists of a regularization term. In this case, $\lambda\mathcal{L}_{PC}$ is added to the training objective and jointly optimized. We note that the proposed framework can be seamlessly instantiated with alternative non-regression penalties, such as the ones proposed in [12], [25].

V. Experiments

Our experiments aim to quantify the *security regression* affecting CL algorithms in malware detection, and to demonstrate how this issue can be addressed via PCT regularization.

A. Experimental Setup

DIL Scenario. In malware detection, a DIL scenario models a realistic setting where the feature distributions evolve over time (e.g., due to changes in software frameworks, or malware evolution), while the prediction targets, i.e., malware/goodware, remain fixed. We conduct our DIL experiments on two Android datasets: ELSA, and TESSERACT [3]. [4] Both datasets contain applications collected from the AndroZoo repository [26], labeled as malicious if detected by at least p VirusTotal scanners (ELSA: $p = 10$, TESSERACT: $p = 4$), and as benign if undetected ($p = 0$). [5] Samples

[4] https://ramd-competition.github.io/

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

with detections in the $(1, p-1)$ range are discarded. ELSA includes 75,000 apps (67,500 goodware, 7,500 malware), spanning 2017–2019. TESSERACT spans 2014–2018 and contains 237,042 apps (232,843 goodware, 26,387 malware). Following [3], we use a 3-month window per experience, splitting each into 80% training and 20% testing, and repeat experiments with 5 random seeds. ELSA contains 6,250 apps per experience (5,000 for training); TESSERACT varies in size across experiences, reflecting real-world data imbalance and making it more challenging. Binary Drebin features [1] are extracted from the first training split, encoding the presence or absence of specific characteristics from each `APK`. Additionally, for ELSA we extract APIGraph [2] features, a graph-based API representation that extends Drebin by modeling relations among API calls. For both feature sets, we remove features with variance below 10^{-3} as in [6], resulting in 4,714 and 3,997 Drebin features for ELSA and TESSERACT, respectively, and 4,564 APIGraph features for ELSA.

CIL Scenario. For CIL, we rely on the AZ-Class dataset [9], built with the same criteria as TESSERACT (i.e., APKs collected from AndroZoo, with $p = 4$). Unlike the DIL setting, the AZ-Class dataset requires classification among 100 malware families, which is inherently more difficult than the binary case. The 100 families are split to obtain 10 experiences of 10 classes each. For each class, 90% of the samples are used for training, and the remaining 10% for testing. This process is repeated 5 times with randomized class orders. We extract Drebin features [1] from the apps and remove those with variance below 10^{-3}, obtaining 2,439 features.

Model Architecture and Training. We use a multi-layer perceptron (MLP) as our main classifier, with a hidden layer of size 512 and an output layer of 2 units for DIL and 100 units for CIL. In the CIL setting, output units corresponding to classes not yet encountered during training are kept inactive throughout the incremental learning process. We minimize the cross-entropy loss using the SGD optimizer with a learning rate of 10^{-3} and momentum of 0.9. We also consider a linear support vector machine (SVM) implemented in PyTorch to support incremental training. The model minimizes the squared hinge loss using SGD with learning rate 10^{-2}, momentum 0.9, and weight decay 10^{-4}. We train both models for 30 epochs with a batch size of 32.

Evaluation Metrics. For DIL, after training on each experience, we measure the detection performance by considering three metrics: (i) *precision*, (ii) *recall* (a.k.a. detection rate), and (iii) F_1 *score*. To assess security regression, we compute the *NFR* (Eq. 3), comparing the predictions of the current model and its immediate predecessor, separately for goodware and malware classes. In the CIL scenario, we measure (i) accuracy, (ii) forgetting, and (iii) NFR. Following [16], after training on the k-th experience, we evaluate the metrics on a selected set of test experiences and aggregate the results according to the backward and forward modes, described in Sect. III-B. We obtain a curve for each metric, where each value (one for each test experience) describes the performance after learning the k-th experience, reporting its mean and standard deviation across five independent runs.

CL Methods. As our main baselines for comparison, we first consider two strategies: (i) the naïve strategy, i.e., finetuning on new experiences without using any knowledge retention mechanism, and (ii) the cumulative strategy, i.e., finetuning on both new experiences and the whole data from previous ones, to assess the lower and upper bounds of the performance, respectively. We then consider five state-of-the-art CL methods: Replay with a buffer size of 200 for DIL experiments, and of 1000 for CIL [15]; Averaged Gradient Episodic Memory (A-GEM) [16] with buffer size of 200 for DIL and 1000 for CIL. The number of examples used to compute the reference gradients is set to 32. The number of stored samples per experience is adjusted based on the number of experiences in the dataset. Elastic Weight Consolidation (EWC) [17] with λ=0.001; Synaptic Intelligence (SI) [18] with λ=0.001 and ϵ=0.1; and Learning without Forgetting (LwF) [27] with α=1 and the temperature=1. Following [10], we use PCT with $\alpha = 1$, as effective during our grid search, and β=0.5 showing a reasonable trade-off between NFR reduction and F1-score. All CL methods are taken from the Avalanche library [28].

Evaluating NFRs on Deeper Architectures. To complement the experiments on MLP and SVM classifiers, we conduct an additional evaluation in the DIL setting using the CNN-based malware detector proposed in [29] and implemented in [30]. The detector operates on Android opcode sequences extracted from APK files; we refer the reader to [29] for further details on the architecture and preprocessing pipeline. For this experiment, we apply the preprocessing pipeline to the ELSA dataset and truncate opcode sequences to a maximum length of 600,000 elements, padding shorter sequences with zeros. Models are trained for 10 epochs on each experience. Due to the computational cost of the experiments, we evaluate only the naïve, cumulative, and PCT strategies. Results are averaged over three runs with different random seeds.

B. Experimental Results

Measuring Security Regression in CL. To evaluate the extent of security regression, we test the considered CL strategies in *backward* mode, i.e., measuring performance on past and current data. This evaluation will assess how much of the previously acquired knowledge is retained. In Table I, we report the precision, recall, F_1, and NFR values for the malware (mw) and goodware (gw) classes under the backward evaluation setting. Here, NFR_{mw} reflects the rate at which previously detected threats are no longer recognized after an update, while NFR_{gw} indicates the rate at which previously accepted benign samples start being misclassified as malicious. While all CL strategies maintain high performance on previously seen data, on average, they exhibit high NFR, indicating a considerable amount of security regression. For example, on the ELSA dataset, SI and A-GEM reach NFR_{mw} of 3.40% and 3.46%, exceeding the naïve baseline (3.17%). On TESSERACT, NFR values are slightly lower but remain high, with EWC and SI reaching nearly 3%. In both datasets, NFR_{gw} is significantly lower than for malware. Overall, CL strategies achieve strong precision, recall, and F_1 on past and current data, thus mitigating forgetting. Yet, security regression remains a consistent issue across methods and datasets. Additional

backward temporal plots are provided in the supplementary material (Fig. S1 and Fig. S2).

Reducing Security Regression in CL. We apply the regression-aware PCT penalty to mitigate security regression. The results in Table I show that adding PCT to CL strategies reduces regression. In particular, SI and A-GEM, whose NFR_{mw} values in the ELSA dataset were 3.40% and 3.46%, respectively, show a reduction of about 50% after integrating PCT, reaching 1.19% and 1.76%, respectively. The further addition of a replay buffer further reduces the NFR. This is evident in the TESSERACT dataset, where SI+Replay+PCT achieves an NFR of 1.43%, compared to 1.75% with its exemplar-free version. This NFR reduction affects the recall but improves precision, resulting in an F_1 score that is close to that of CL strategies alone for the ELSA dataset, and only slightly lower for the TESSERACT dataset. In practice, this means the model becomes more consistent in maintaining past correct predictions (especially avoiding new false positives) while slightly reducing detection. Overall, integrating PCT into CL strategies results in a substantial reduction in NFR, particularly when combined with a replay buffer.

Influence of α and β. We analyze the trade-off between classification performance and security regression by investigating the influence of PCT's α and β parameters. We recall that these are the hyperparameters of PCT, where the scalar α acts as a weighting factor applied to all training samples, ensuring a minimum level of distillation, and the scalar β adds extra weight to the distillation loss for samples that were correctly predicted by the old model. In Fig. 2, we report the F_1 score (left), NFR_{gw} (middle), and NFR_{mw} (right) for different configurations of α and β in *backward* mode. For both goodware and malware classes, NFR decreases as α and β increase. The impact of β on NFR becomes less significant at higher α compared to lower ones. Furthermore, the leftmost plot shows that the F_1 score also declines as α increases, while the effect of β remains minimal, except for small α.

Influence of Buffer Size. We now analyze the influence of the buffer size in reducing regression on Replay, keeping α and β constant. We show in Fig. 3 classification and regression metrics (backward mode) for increasing memory buffer sizes. Larger buffer sizes significantly reduce regression by nearly 50% without significantly affecting classification performance.

Reducing Security Regression in SOTA Detectors. We measure and mitigate security regression in state-of-the-art detectors via additional ELSA experiments using Drebin [1] and APIGraph [2] feature sets with Linear SVM and MLP models. In Table II we report F_1 and NFR_{mw} (backward mode). All strategies across models and feature sets achieve high F_1 scores, with the MLP performing slightly better than the Linear SVM. However, CL strategies alone exhibit non-negligible NFR_{mw}, especially for the Linear SVM, where SI on the APIGraph reaches 6.44%. In comparison, the MLP consistently shows lower regression on both feature sets. When combining CL strategies with PCT (values in parentheses), predictive performance improves, leading to higher F_1 scores across models and feature sets, while the NFR_{mw} is substantially reduced. For instance, with Linear SVM, NFR_{mw} for SI on APIGraph drops from 6.44% to 1.87%, highlighting PCT's impact in reducing regression. Overall, these results show the effectiveness of PCT in improving prediction stability while enhancing predictive performance of state-of-the-art detectors.

Resilience to Concept Drift. Robustness to concept drift reduces the need for model updates. We evaluate CL methods in *forward* mode, using future data to measure adaptability to shifting distributions. In Table III, we show the precision, recall, F_1 score, NFR_{mw}, and NFR_{gw} measured in this setting. While performance drops compared to *backward* mode, CL strategies still adapt to unseen data. However, significant regression remains, as seen in Table III. EWC and SI on ELSA reach NFR_{mw} of 3.12% and 3.06%, respectively, more than double that of the Cumulative baseline (1.46%). PCT drastically reduces NFR: EWC+PCT and SI+PCT lower these values to 0.98% and 0.90%. Adding a replay buffer further improves results, with SI+Replay+PCT achieving 0.57% malware NFR on TESSERACT, down from 0.79% with SI+PCT. Even in this case, PCT reduces recall but improves precision, especially in TESSERACT, where F_1 drops more noticeably. Detailed forward temporal evolution plots are reported in the supplementary material (Fig. S3 and Fig. S4).

Reducing Regression when Learning Malware Families Incrementally (CIL Scenario). Beyond detection, family classification is required to better understand and mitigate threats. In the CIL scenario, at each experience, 10 new families are introduced as output classes; therefore, only *backward* evaluation is applicable. In this setting, a replay buffer is essential to maintain stability across experiences. Experiments show that this scenario also suffers from security regression. Again, the addition of PCT mitigates the presence of NFs. Table IV shows the results in terms of average accuracy, forgetting, and NFR, reporting the worst case throughout the incremental learning process. All CL strategies perform worse than the cumulative baseline, with A-GEM being the worst (51.43% accuracy, 62.28% forgetting, 18.94% NFR). Adding PCT improves all strategies across metrics. For instance, the accuracy of SI increases from 72.46% to 78.12%, while average forgetting and NFR decrease from 29.99% to 22.16% and from 9.76% to 7.42%, respectively. EWC and LwF show similar trends. These results show that integrating PCT into CL strategies consistently mitigates performance and regression.

Evaluating NFRs on a CNN-based Malware Detector [29]. Table V reports the results in terms of precision, recall, F_1, NFR_{mw}, and NFR_{gw}. The naïve baseline reaches 7.08% NFR_{mw}, while PCT reduces it to 3.35%, approaching the cumulative baseline (2.91%). This reduction comes at the cost of lower recall and F_1 score. Overall, the results confirm that security regression also affects deeper, non-linear model architectures and that it can be mitigated through PCT.

VI. Related Work

Continual Learning for Malware Detection. Using CL in malware detection has been first discussed in [6], which provides an extensive analysis of CL methods for malware classification. By considering different CL scenarios (DIL, CIL, and TIL), they find replay-based methods useful for malware detection, while regularization-based methods are

TABLE I

RESULTS ON **ELSA** AND **TESSERACT** (**BACKWARD** MODE). WE SHOW TWO BASELINES AND THE CL STRATEGIES, ALONE OR COMBINED WITH PCT, REPLAY, OR BOTH. FOR EACH, WE MEASURE PRECISION, RECALL, F_1, NFR ON GOODWARE (GW), AND MALWARE (MW).

	ELSA					TESSERACT				
Method	**Precision (%)↑**	**Recall (%)↑**	**F_1 (%)↑**	**NFR$_{mw}$ (%)↓**	**NFR$_{gw}$ (%)↓**	**Precision (%)↑**	**Recall (%)↑**	**F_1 (%)↑**	**NFR$_{mw}$ (%)↓**	**NFR$_{gw}$ (%)↓**
Cumulative	$89.17_{\pm 1.93}$	$87.08_{\pm 3.94}$	$87.81_{\pm 2.13}$	$1.16_{\pm 1.00}$	$0.23_{\pm 0.19}$	$95.96_{\pm 1.24}$	$92.28_{\pm 1.95}$	$93.91_{\pm 1.37}$	$0.54_{\pm 0.18}$	$0.07_{\pm 0.05}$
Naive	$86.12_{\pm 3.77}$	$84.00_{\pm 3.99}$	$84.52_{\pm 1.65}$	$3.17_{\pm 1.56}$	$0.48_{\pm 0.36}$	$93.71_{\pm 2.12}$	$84.89_{\pm 5.71}$	$88.78_{\pm 3.59}$	$2.92_{\pm 2.00}$	$0.27_{\pm 0.16}$
Replay	$86.86_{\pm 2.26}$	$83.19_{\pm 2.93}$	$84.62_{\pm 1.18}$	$2.76_{\pm 1.23}$	$0.36_{\pm 0.18}$	$93.53_{\pm 1.57}$	$85.67_{\pm 3.84}$	$89.22_{\pm 2.56}$	$2.28_{\pm 1.31}$	$0.25_{\pm 0.11}$
PCT	$89.99_{\pm 0.72}$	$83.68_{\pm 1.94}$	$86.48_{\pm 1.10}$	$1.05_{\pm 0.36}$	$0.13_{\pm 0.07}$	$96.58_{\pm 1.09}$	$81.30_{\pm 5.13}$	$87.99_{\pm 3.44}$	$1.75_{\pm 1.22}$	$0.08_{\pm 0.05}$
PCT+Replay	$90.30_{\pm 0.64}$	$81.86_{\pm 1.49}$	$85.59_{\pm 0.87}$	$1.00_{\pm 0.44}$	$0.10_{\pm 0.05}$	$96.01_{\pm 1.14}$	$81.40_{\pm 2.85}$	$87.89_{\pm 2.20}$	$1.33_{\pm 0.89}$	$0.09_{\pm 0.10}$
LwF	$87.84_{\pm 1.39}$	$84.49_{\pm 3.15}$	$85.82_{\pm 1.27}$	$1.48_{\pm 0.53}$	$0.23_{\pm 0.11}$	$95.37_{\pm 1.44}$	$84.58_{\pm 5.04}$	$89.38_{\pm 3.23}$	$1.87_{\pm 1.24}$	$0.12_{\pm 0.06}$
LwF+Replay	$87.35_{\pm 1.48}$	$83.51_{\pm 2.69}$	$85.06_{\pm 1.09}$	$1.29_{\pm 0.62}$	$0.19_{\pm 0.09}$	$94.32_{\pm 1.30}$	$84.87_{\pm 3.47}$	$89.08_{\pm 2.46}$	$1.51_{\pm 1.03}$	$0.15_{\pm 0.12}$
LwF+PCT	$90.40_{\pm 0.83}$	$82.06_{\pm 1.81}$	$85.73_{\pm 0.92}$	$1.03_{\pm 0.50}$	$0.10_{\pm 0.05}$	$96.67_{\pm 0.96}$	$79.86_{\pm 4.25}$	$87.19_{\pm 2.86}$	$1.55_{\pm 1.06}$	$0.06_{\pm 0.04}$
LwF+Replay+PCT	$89.87_{\pm 0.82}$	$80.33_{\pm 1.89}$	$84.50_{\pm 0.98}$	$1.01_{\pm 0.46}$	$0.08_{\pm 0.03}$	$95.74_{\pm 1.29}$	$80.63_{\pm 2.55}$	$87.25_{\pm 2.13}$	$1.09_{\pm 0.67}$	$0.08_{\pm 0.08}$
EWC	$86.18_{\pm 3.58}$	$83.85_{\pm 3.70}$	$84.49_{\pm 1.59}$	$3.38_{\pm 1.59}$	$0.47_{\pm 0.35}$	$93.80_{\pm 2.08}$	$84.85_{\pm 5.65}$	$88.80_{\pm 3.56}$	$2.93_{\pm 1.99}$	$0.27_{\pm 0.16}$
EWC+Replay	$86.35_{\pm 2.93}$	$83.64_{\pm 3.65}$	$84.53_{\pm 1.20}$	$2.56_{\pm 1.03}$	$0.39_{\pm 0.22}$	$93.45_{\pm 1.64}$	$85.41_{\pm 3.99}$	$88.92_{\pm 2.72}$	$2.41_{\pm 1.48}$	$0.27_{\pm 0.23}$
EWC+PCT	$90.03_{\pm 0.68}$	$83.57_{\pm 2.04}$	$86.44_{\pm 1.17}$	$1.21_{\pm 0.46}$	$0.13_{\pm 0.07}$	$96.40_{\pm 1.05}$	$81.50_{\pm 5.23}$	$88.03_{\pm 3.47}$	$1.78_{\pm 1.35}$	$0.08_{\pm 0.05}$
EWC+Replay+PCT	$90.00_{\pm 0.73}$	$81.32_{\pm 1.61}$	$85.18_{\pm 0.92}$	$1.11_{\pm 0.42}$	$0.10_{\pm 0.05}$	$95.79_{\pm 1.02}$	$81.38_{\pm 3.08}$	$87.77_{\pm 2.26}$	$1.32_{\pm 0.98}$	$0.07_{\pm 0.05}$
SI	$85.90_{\pm 3.42}$	$83.87_{\pm 3.73}$	$84.38_{\pm 1.66}$	$3.40_{\pm 1.60}$	$0.49_{\pm 0.36}$	$93.80_{\pm 2.06}$	$85.01_{\pm 5.57}$	$88.89_{\pm 3.51}$	$2.92_{\pm 2.03}$	$0.27_{\pm 0.16}$
SI+Replay	$86.52_{\pm 2.16}$	$82.50_{\pm 3.65}$	$84.05_{\pm 1.45}$	$2.82_{\pm 1.47}$	$0.36_{\pm 0.25}$	$93.47_{\pm 1.45}$	$85.08_{\pm 4.24}$	$88.79_{\pm 2.72}$	$2.38_{\pm 1.41}$	$0.26_{\pm 0.14}$
SI+PCT	$90.18_{\pm 0.80}$	$83.09_{\pm 2.18}$	$86.22_{\pm 1.12}$	$1.19_{\pm 0.40}$	$0.13_{\pm 0.06}$	$96.57_{\pm 1.06}$	$81.18_{\pm 5.13}$	$87.91_{\pm 3.42}$	$1.75_{\pm 1.26}$	$0.08_{\pm 0.05}$
SI+Replay+PCT	$89.95_{\pm 0.83}$	$81.51_{\pm 2.47}$	$85.22_{\pm 1.19}$	$1.16_{\pm 0.59}$	$0.10_{\pm 0.04}$	$96.26_{\pm 0.80}$	$81.31_{\pm 4.02}$	$87.92_{\pm 2.62}$	$1.43_{\pm 1.07}$	$0.05_{\pm 0.02}$
A-GEM	$86.85_{\pm 2.94}$	$83.49_{\pm 3.48}$	$84.73_{\pm 1.08}$	$3.46_{\pm 1.36}$	$0.42_{\pm 0.22}$	$93.82_{\pm 1.57}$	$85.06_{\pm 4.81}$	$88.96_{\pm 2.78}$	$2.83_{\pm 1.49}$	$0.26_{\pm 0.12}$
A-GEM+PCT	$88.55_{\pm 2.15}$	$81.96_{\pm 1.37}$	$84.74_{\pm 1.63}$	$1.76_{\pm 0.94}$	$0.24_{\pm 0.26}$	$93.81_{\pm 3.00}$	$78.04_{\pm 7.20}$	$84.51_{\pm 6.06}$	$2.51_{\pm 3.61}$	$0.18_{\pm 0.15}$

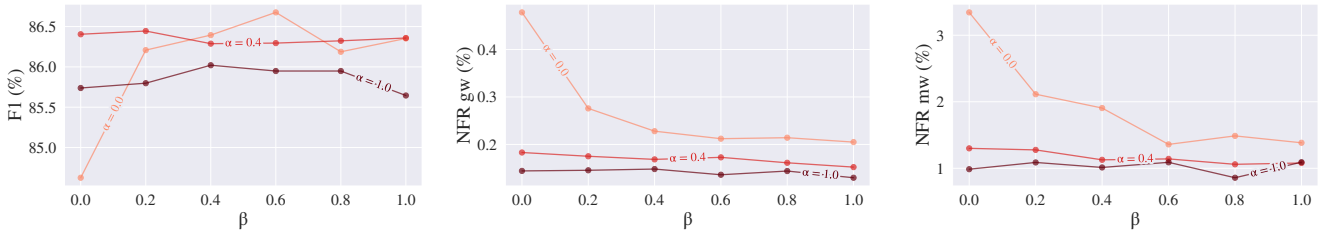


Fig. 2. Hyperparameter sensitivity of PCT without memory buffer. We show F_1 (*left*), NFR for goodware samples (*middle*), and for malware (*right*), on a grid search over α and β in $[0, 1]$. Each curve corresponds to a fixed value of α, while β varies.

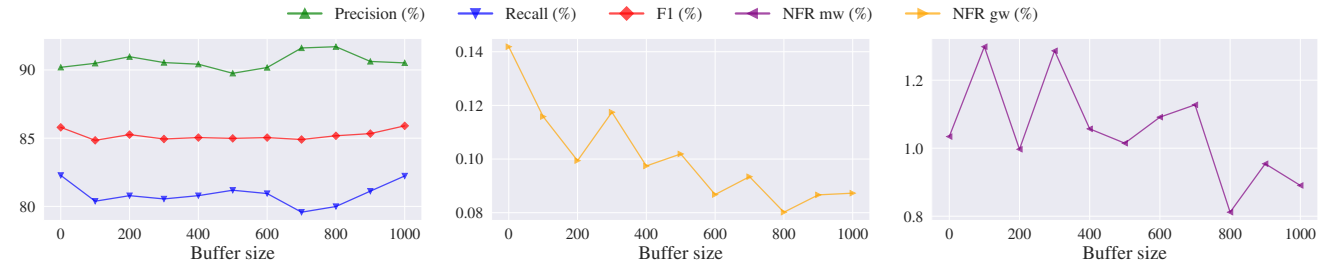


Fig. 3. Influence of buffer size, with $\alpha = 1$ and $\beta = 0.5$. Metrics are shown for different values of buffer size in $[0, 1000]$.

shown to be less effective. MADAR proposes a novel CL method that employs diversity-aware replay to mitigate catastrophic forgetting [7]. SSCL-TransMD trains a transformer-based architecture with a replay memory buffer and uses pseudo-labeling [8]. MalFSCIL [31] proposes a few-shot CIL approach using decoupled training and a variational autoencoder. Further approaches tackle malware detection as continual semi-supervised one-class learning [32], use multi-modal features [22], or propose a generative replay-based approach to create high-quality synthetic malware samples [9]. However, these studies mainly focus on catastrophic forgetting and overlook the security regression issue.

Reducing Regression. Pioneering work in the image classification domain first addressed regression and proposed Positive-Congruent Training (PCT) as a mitigation strategy [10]. This approach has been extended by distilling an ensemble rather than a single model [12], while another work adapts PCT for adversarial robustness [25]. Regression has been empirically observed, without the focus on ML, in real-world antivirus systems [11], where previously detected malware may become undetected over time. However, analyzing negative flips or integrating regression-aware objectives into the CL pipeline has not been previously explored. In contrast, our contribution extends CL for malware detection by formally introducing, quantifying, and mitigating security regressions in model updates. Moreover, our methodology remains applicable to other regression-aware formulations such as [12], [25].

Concept Drift Mitigation Several works in malware de-

TABLE II

RESULTS ON **ELSA** (**BACKWARD** MODE) USING **LINEAR SVM** AND **MLP** ACROSS **DREBIN** AND **APIGRAPH** FEATURE SETS. WE SHOW THREE BASELINES AND THE CL STRATEGIES, ALONE OR COMBINED WITH PCT (VALUES IN PARENTHESES REFER TO THE +PCT VARIANT, GREEN INDICATES AN IMPROVEMENT, AND RED A DETERIORATION). FOR EACH, WE MEASURE F_1 AND NFR ON MALWARE (MW).

	Linear SVM				**MLP**			
	Drebin		**APIGraph**		**Drebin**		**APIGraph**	
Method	F_1 (%)↑	NFR_{mw} (%)↓	F_1 (%)↑	NFR_{mw} (%)↓	F_1 (%)↑	NFR_{mw} (%)↓	F_1 (%)↑	NFR_{mw} (%)↓
Cumulative	86.48	2.38	86.45	2.06	87.81	1.16	89.48	1.11
Naive	80.73	6.13	82.20	6.36	84.52	3.17	85.13	3.95
PCT	85.63	1.71	85.07	2.00	86.48	1.05	85.67	1.59
Replay (+PCT)	82.35 (85.64)	5.24 (1.28)	82.59 (84.39)	5.38 (1.62)	84.62 (85.59)	2.76 (1.00)	84.22 (84.32)	3.38 (1.22)
LwF (+PCT)	84.69 (85.62)	2.92 (1.33)	83.83 (84.75)	3.65 (1.69)	85.82 (85.73)	1.48 (1.03)	85.48 (85.01)	2.23 (1.29)
EWC (+PCT)	80.53 (85.75)	6.16 (1.57)	81.98 (84.82)	6.29 (2.09)	84.49 (86.44)	3.38 (1.21)	85.10 (85.83)	3.92 (1.59)
SI (+PCT)	80.49 (85.70)	6.10 (1.74)	82.09 (85.19)	6.44 (1.87)	84.38 (86.22)	3.40 (1.19)	85.18 (85.75)	3.86 (1.54)
AGEM (+PCT)	81.18 (84.98)	6.02 (1.60)	82.26 (83.43)	6.01 (2.43)	84.73 (84.74)	3.46 (1.76)	84.60 (83.37)	4.02 (2.02)

TABLE III

RESULTS ON **ELSA** AND **TESSERACT** (**FORWARD** MODE). WE SHOW TWO BASELINES AND THE CL STRATEGIES, ALONE OR COMBINED WITH PCT, REPLAY, OR BOTH. FOR EACH, WE MEASURE PRECISION, RECALL, F_1, NFR ON GOODWARE (GW), AND MALWARE (MW).

	ELSA					**TESSERACT**				
Method	**Precision** (%)↑	**Recall** (%)↑	F_1 (%)↑	NFR_{mw} (%)↓	NFR_{gw} (%)↓	**Precision** (%)↑	**Recall** (%)↑	F_1 (%)↑	NFR_{mw} (%)↓	NFR_{gw} (%)↓
Cumulative	$86.05_{\pm 3.96}$	$80.54_{\pm 4.75}$	$82.71_{\pm 4.27}$	$1.46_{\pm 0.90}$	$0.23_{\pm 0.17}$	$84.99_{\pm 7.14}$	$71.11_{\pm 12.29}$	$75.75_{\pm 9.76}$	$1.04_{\pm 0.77}$	$0.47_{\pm 0.64}$
Naive	$87.42_{\pm 5.94}$	$79.28_{\pm 4.52}$	$82.66_{\pm 4.63}$	$3.08_{\pm 1.25}$	$0.34_{\pm 0.31}$	$84.14_{\pm 7.85}$	$70.83_{\pm 11.14}$	$75.10_{\pm 9.26}$	$2.09_{\pm 1.60}$	$0.63_{\pm 0.80}$
Replay	$86.77_{\pm 6.28}$	$78.75_{\pm 4.12}$	$82.06_{\pm 4.56}$	$2.59_{\pm 1.27}$	$0.31_{\pm 0.27}$	$82.55_{\pm 8.00}$	$69.55_{\pm 11.71}$	$73.73_{\pm 9.58}$	$1.82_{\pm 1.54}$	$0.63_{\pm 0.85}$
PCT	$88.70_{\pm 3.65}$	$78.47_{\pm 3.81}$	$82.81_{\pm 3.45}$	$0.84_{\pm 0.52}$	$0.09_{\pm 0.06}$	$89.63_{\pm 4.70}$	$61.91_{\pm 11.45}$	$70.74_{\pm 8.36}$	$0.76_{\pm 0.58}$	$0.30_{\pm 0.54}$
PCT+Replay	$89.27_{\pm 3.54}$	$76.93_{\pm 3.61}$	$82.17_{\pm 3.15}$	$0.83_{\pm 0.56}$	$0.07_{\pm 0.06}$	$87.66_{\pm 4.67}$	$64.02_{\pm 9.51}$	$71.53_{\pm 6.84}$	$0.65_{\pm 0.42}$	$0.22_{\pm 0.53}$
LwF	$87.18_{\pm 5.24}$	$79.24_{\pm 4.23}$	$82.57_{\pm 4.36}$	$1.36_{\pm 0.62}$	$0.17_{\pm 0.15}$	$85.42_{\pm 6.80}$	$66.49_{\pm 11.65}$	$72.72_{\pm 9.05}$	$1.01_{\pm 0.88}$	$0.41_{\pm 0.64}$
LwF+Replay	$88.06_{\pm 4.01}$	$78.82_{\pm 4.29}$	$82.70_{\pm 3.86}$	$1.08_{\pm 0.52}$	$0.15_{\pm 0.12}$	$85.77_{\pm 5.16}$	$71.61_{\pm 7.76}$	$76.20_{\pm 5.42}$	$1.08_{\pm 1.06}$	$0.34_{\pm 0.53}$
LwF+PCT	$88.54_{\pm 3.32}$	$76.95_{\pm 3.27}$	$81.86_{\pm 3.01}$	$0.79_{\pm 0.49}$	$0.08_{\pm 0.06}$	$90.33_{\pm 4.59}$	$59.58_{\pm 11.05}$	$69.41_{\pm 8.26}$	$0.63_{\pm 0.44}$	$0.24_{\pm 0.70}$
LwF+Replay+PCT	$89.00_{\pm 3.41}$	$75.14_{\pm 2.96}$	$81.01_{\pm 2.84}$	$0.76_{\pm 0.37}$	$0.06_{\pm 0.06}$	$88.63_{\pm 4.29}$	$62.59_{\pm 8.67}$	$70.77_{\pm 6.31}$	$0.57_{\pm 0.39}$	$0.18_{\pm 0.33}$
EWC	$87.50_{\pm 5.79}$	$79.22_{\pm 4.54}$	$82.68_{\pm 4.66}$	$3.12_{\pm 1.38}$	$0.33_{\pm 0.29}$	$84.04_{\pm 7.70}$	$70.67_{\pm 11.33}$	$74.90_{\pm 9.31}$	$2.11_{\pm 1.66}$	$0.63_{\pm 0.73}$
EWC+Replay	$87.83_{\pm 4.92}$	$78.78_{\pm 4.74}$	$82.55_{\pm 4.31}$	$2.29_{\pm 1.29}$	$0.28_{\pm 0.25}$	$83.70_{\pm 5.68}$	$74.33_{\pm 8.51}$	$76.71_{\pm 6.52}$	$1.82_{\pm 1.21}$	$0.44_{\pm 0.51}$
EWC+PCT	$88.67_{\pm 3.90}$	$78.58_{\pm 3.59}$	$82.86_{\pm 3.39}$	$0.98_{\pm 0.55}$	$0.10_{\pm 0.07}$	$88.88_{\pm 4.73}$	$62.14_{\pm 11.33}$	$70.75_{\pm 8.24}$	$0.81_{\pm 0.62}$	$0.29_{\pm 0.54}$
EWC+Replay+PCT	$88.02_{\pm 3.69}$	$76.82_{\pm 3.36}$	$81.51_{\pm 3.30}$	$0.91_{\pm 0.35}$	$0.09_{\pm 0.07}$	$88.68_{\pm 4.45}$	$66.94_{\pm 7.26}$	$74.24_{\pm 5.10}$	$0.62_{\pm 0.38}$	$0.26_{\pm 0.52}$
SI	$87.54_{\pm 5.79}$	$79.29_{\pm 4.53}$	$82.73_{\pm 4.56}$	$3.06_{\pm 1.36}$	$0.33_{\pm 0.27}$	$84.12_{\pm 7.75}$	$70.64_{\pm 11.40}$	$74.89_{\pm 9.33}$	$2.14_{\pm 1.64}$	$0.63_{\pm 0.81}$
SI+Replay	$88.08_{\pm 5.08}$	$78.10_{\pm 4.71}$	$82.30_{\pm 4.55}$	$2.52_{\pm 1.20}$	$0.27_{\pm 0.22}$	$84.23_{\pm 5.82}$	$73.67_{\pm 8.17}$	$76.77_{\pm 6.33}$	$1.98_{\pm 1.33}$	$0.49_{\pm 0.63}$
SI+PCT	$89.07_{\pm 3.70}$	$78.23_{\pm 3.80}$	$82.84_{\pm 3.45}$	$0.90_{\pm 0.59}$	$0.09_{\pm 0.08}$	$89.41_{\pm 4.75}$	$61.98_{\pm 11.28}$	$70.79_{\pm 8.25}$	$0.79_{\pm 0.55}$	$0.28_{\pm 0.54}$
SI+Replay+PCT	$88.63_{\pm 3.47}$	$76.56_{\pm 3.27}$	$81.64_{\pm 3.16}$	$0.90_{\pm 0.48}$	$0.09_{\pm 0.06}$	$89.11_{\pm 4.41}$	$65.35_{\pm 8.55}$	$73.57_{\pm 5.99}$	$0.58_{\pm 0.31}$	$0.25_{\pm 0.57}$
A-GEM	$86.57_{\pm 6.20}$	$79.27_{\pm 4.25}$	$82.28_{\pm 4.57}$	$3.03_{\pm 1.37}$	$0.41_{\pm 0.36}$	$83.39_{\pm 8.26}$	$71.16_{\pm 10.82}$	$75.01_{\pm 9.16}$	$2.36_{\pm 1.96}$	$0.69_{\pm 0.83}$
A-GEM+PCT	$88.38_{\pm 3.55}$	$76.31_{\pm 3.36}$	$81.38_{\pm 3.41}$	$1.33_{\pm 0.62}$	$0.12_{\pm 0.08}$	$87.96_{\pm 4.32}$	$66.04_{\pm 9.81}$	$72.72_{\pm 6.96}$	$1.01_{\pm 0.81}$	$0.28_{\pm 0.47}$

TABLE IV

RESULTS ON THE **AZ-CLASS** DATASET IN THE **CIL SETTING**. FOR EACH STRATEGY, WE COMPUTE ACCURACY, FORGETTING, AND NFR, REPORTING THE MEAN, STANDARD DEVIATION, AND WORST VALUE.

	Accuracy (%)↑		**Forgetting** (%)↓		**NFR** (%)↓	
Method	Avg.	Worst	Avg.	Worst	Avg.	Worst
Cumulative	$90.81_{\pm 3.12}$	86.89	$2.65_{\pm 0.85}$	3.51	$1.52_{\pm 0.41}$	2.10
Replay	$77.70_{\pm 11.19}$	62.94	$22.71_{\pm 9.52}$	35.24	$7.74_{\pm 2.15}$	10.66
PCT+Replay	$77.00_{\pm 8.15}$	67.25	$22.39_{\pm 5.24}$	29.06	$7.19_{\pm 1.49}$	9.93
LwF+Replay	$77.00_{\pm 7.82}$	66.65	$17.02_{\pm 6.71}$	25.86	$5.21_{\pm 1.29}$	7.32
LwF+Replay+PCT	$78.56_{\pm 7.44}$	68.66	$16.12_{\pm 4.78}$	22.60	$4.89_{\pm 1.13}$	6.55
EWC+Replay	$73.09_{\pm 10.07}$	60.87	$29.38_{\pm 6.25}$	37.95	$9.68_{\pm 1.73}$	12.46
EWC+Replay+PCT	$77.98_{\pm 8.66}$	66.99	$22.30_{\pm 5.57}$	29.82	$7.51_{\pm 1.64}$	9.83
SI+Replay	$72.46_{\pm 10.07}$	60.45	$29.99_{\pm 6.17}$	38.31	$9.76_{\pm 1.56}$	12.24
SI+Replay+PCT	$78.12_{\pm 8.59}$	67.40	$22.16_{\pm 5.51}$	29.32	$7.42_{\pm 1.67}$	9.98
A-GEM	$51.43_{\pm 20.61}$	30.46	$62.28_{\pm 11.24}$	74.34	$18.94_{\pm 4.19}$	25.45
A-GEM+PCT	$52.04_{\pm 18.88}$	30.44	$62.94_{\pm 6.68}$	74.00	$18.87_{\pm 4.73}$	28.33

TABLE V

RESULTS ON THE **ELSA** DATASET FOR THE CNN-BASED MALWARE DETECTOR [29] IN THE **DIL SETTING**. WE SHOW TWO BASELINES AND THE PCT STRATEGY. FOR EACH, WE MEASURE PRECISION, RECALL, F_1, AND NFR ON GOODWARE (GW) AND MALWARE (MW).

Method	**Precision** (%)↑	**Recall** (%)↑	F_1 (%)↑	NFR_{mw} (%)↓	NFR_{gw} (%)↓
Cumulative	$87.88_{\pm 5.71}$	$85.21_{\pm 3.79}$	$85.43_{\pm 5.09}$	$2.91_{\pm 1.45}$	$1.05_{\pm 1.60}$
Naive	$86.42_{\pm 8.30}$	$76.33_{\pm 5.38}$	$79.24_{\pm 4.65}$	$7.08_{\pm 4.61}$	$1.84_{\pm 3.37}$
PCT	$92.45_{\pm 1.74}$	$63.90_{\pm 4.75}$	$74.34_{\pm 3.06}$	$3.35_{\pm 2.55}$	$0.20_{\pm 0.15}$

tection focus instead on anticipating concept drift to mitigate performance degradation on future data. Pendelbury et al. [3] first highlighted the impact of temporal evaluation protocols, showing how concept drift can significantly affect malware detectors over time. Other approaches rely on drift detection mechanisms to trigger updating procedures only when distributional changes are detected [33]–[35], whereas others aim to make models more robust to drift [4], [36]. In our work, we focus on preserving past decisions rather than adapting to future data distributions, with the update frequency dictated by the CL setting. Nevertheless, these directions are complementary: drift detection can be used to determine when updates should occur, while our proposed contribution can mitigate forgetting and regression.

VII. Conclusions and Future Work

In this work, we analyze the phenomenon of security regression in CL applied to malware detection, showing that although many CL strategies limit forgetting, they often fail to ensure consistent predictions after model updates. We introduce a regression-aware training framework and instantiate it using the PCT regularizer [10], demonstrating that it consistently reduces regression with minimal impact on performance across different evaluation modes and scenarios. We acknowledge that this study assumes clean labels and access to prior model outputs, conditions not always met in real-world deployments. Moreover, the AZ-Class dataset used for multi-class experiments lacks temporal structure, limiting its ability to reflect realistic malware evolution. Future work could address these gaps by leveraging active learning to reduce labeling demands or by adopting temporally-aware benchmarks for multi-class settings, better capturing malware dynamics. We also aim to extend our framework by instantiating it with more recent state-of-the-art regression-aware penalties, such as ELODI [12] and RCAT [25]. Lastly, our results highlight an important trade-off between preserving past decisions and maximizing average forward performance. Enforcing regression mitigation is particularly beneficial in security-critical scenarios where reintroducing previously mitigated threats carries a high cost. Conversely, in highly dynamic environments where malware rapidly becomes obsolete, prioritizing forward performance may be sufficient, making regression mitigation a scenario-dependent design choice. Despite these limitations, our findings remain broadly applicable and provide a foundation for more robust CL in security-sensitive domains. Minimizing regression is key to maintaining trust in continuously updated malware detectors, making them more reliable and scalable in the face of evolving threats.

Acknowledgments

This research has been partially supported by the Horizon Europe projects ELSA (GA no. 101070617), Sec4AI4Sec (GA no. 101120393), and CoEvolution (GA no. 101168560); and by SERICS (PE00000014) and FAIR (PE00000013) under the MUR NRRP funded by the EU-NGEU. This work was carried out while D. Ghiani and S. Gupta were enrolled in the Italian National Doctorate on AI run by the Sapienza Univ. of Rome in collaboration with the Univ. of Cagliari.

References

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

[2] X. Zhang, Y. Zhang, M. Zhong, D. Ding, Y. Cao, Y. Zhang, M. Zhang, and M. Yang, "Enhancing state-of-the-art classifiers with api semantics to detect evolved android malware," in *CSS*, pp. 757–770, 2020.

[3] F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder, and L. Cavallaro, "TESSERACT: Eliminating experimental bias in malware classification across space and time," in *28th USENIX Security Symp.*, pp. 729–746, 2019.

[4] D. Angioni, L. Demetrio, M. Pintor, and B. Biggio, "Robust machine learning for malware detection over time," in *ITASEC*, vol. 3260, pp. 169–180, 2022.

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

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

[7] M. S. Rahman, S. Coull, Q. Yu, and M. Wright, "Madar: Efficient continual learning for malware analysis with diversity-aware replay," 2025.

[8] L. Kou, D. Zhao, H. Han, X. Xu, S. Gong, and L. Wang, "Sscl-transmd: Semi-supervised continual learning transformer for malicious software detection," *Applied Sciences*, vol. 13, no. 22, p. 12255, 2023.

[9] J. Park, A. Ji, M. Park, M. S. Rahman, and S. E. Oh, "Malcl: Leveraging gan-based generative replay to combat catastrophic forgetting in malware classification," in *AAAI*, vol. 39, pp. 658–666, 2025.

[10] S. Yan, Y. Xiong, K. Kundu, S. Yang, S. Deng, M. Wang, W. Xia, and S. Soatto, "Positive-congruent training: Towards regression-free model updates," in *CVPR*, pp. 14299–14308, 2021.

[11] M. Botacin, F. Ceschin, P. De Geus, and A. Grégio, "We need to talk about antiviruses: challenges & pitfalls of av evaluations," *Comput. & Secur.*, vol. 95, p. 101859, 2020.

[12] Y. Zhao, Y. Shen, Y. Xiong, S. Yang, W. Xia, Z. Tu, B. Schiele, and S. Soatto, "Elodi: Ensemble logit difference inhibition for positive-congruent training," *IEEE TPAMI*, vol. 46, no. 12, pp. 7529–7541, 2024.

[13] D. Ghiani, D. Angioni, A. Sotgiu, M. Pintor, and B. Biggio, "Understanding regression in continual learning for malware detection," in *ITASEC*, vol. 3962, 2025.

[14] G. M. Van de Ven, T. Tuytelaars, and A. S. Tolias, "Three types of incremental learning," *Nat. Mach. Intell.*, vol. 4, no. 12, pp. 1185–1197, 2022.

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

[16] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, "Efficient lifelong learning with a-gem," *arXiv preprint arXiv:1812.00420*, 2018.

[17] 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," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 114, no. 13, pp. 3521–3526, 2017.

[18] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *ICML*, pp. 3987–3995, 2017.

[19] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, 2016.

[20] J. Xu and Z. Zhu, "Reinforced continual learning," *NeurIPS*, vol. 31, 2018.

[21] M. A. Haque, I. Hossain, M. M. Kamol, M. J. Alam, S. K. Amalapuram, S. Talukder, and M. S. Rahman, "Lamda: A longitudinal android malware benchmark for concept drift analysis," *arXiv preprint arXiv:2505.18551*, 2025.

[22] T. Sun, N. Daoudi, W. Pian, K. Kim, K. Allix, T. F. Bissyandé, and J. Klein, "Temporal-incremental learning for android malware detection," *ACM Trans. Softw. Eng. and Methodol.*, vol. 34, no. 4, pp. 1–30, 2025.

[23] P. Faruki, A. Bharmal, V. Laxmi, V. Ganmoor, M. S. Gaur, M. Conti, and M. Rajarajan, "Android security: a survey of issues, malware penetration, and defenses," *IEEE Commun. Surveys & Tuts.*, vol. 17, no. 2, pp. 998–1022, 2014.

[24] G. Canfora, A. Di Sorbo, F. Mercaldo, and C. A. Visaggio, "Obfuscation techniques against signature-based detection: a case study," in *2015 Mobile Syst. Technol. Workshop (MST)*, pp. 21–26, 2015.

[25] D. Angioni, L. Demetrio, M. Pintor, L. Oneto, D. Anguita, B. Biggio, and F. Roli, "Robustness-congruent adversarial training for secure machine learning model updates," *IEEE TPAMI*, 2025.

[26] K. Allix, T. F. Bissyandé, J. Klein, and Y. Le Traon, "Androzoo: Collecting millions of android apps for the research community," in *IEEE/ACM Work. Conf. Mining Softw. Repos. (MSR)*, pp. 468–471, 2016.

[27] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE TPAMI*, vol. 40, no. 12, pp. 2935–2947, 2017.

[28] A. Carta, L. Pellegrini, A. Cossu, H. Hemati, and V. Lomonaco, "Avalanche: A pytorch library for deep continual learning," *J. Mach. Learn. Res.*, vol. 24, pp. 363:1–363:6, 2023.

[29] N. McLaughlin, J. Martinez del Rincon, B. Kang, S. Yerima, P. Miller, S. Sezer, Y. Safaei, E. Trickel, Z. Zhao, A. Doupé, *et al.*, "Deep android malware detection," in *ACM Conf. Data Appl. Secur. Privacy*, pp. 301–308, 2017.

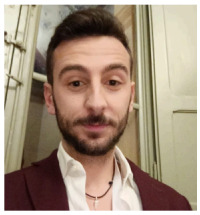
[30] J. Liu, J. Zeng, F. Pierazzi, Z. Yang, L. Cavallaro, and Z. Liang, "Unraveling the key of machine learning-based android malware detection," *ACM Trans. Softw. Eng. and Methodol.*, 2026.
[31] Y. Chai, X. Chen, J. Qiu, L. Du, Y. Xiao, Q. Feng, S. Ji, and Z. Tian, "Malfscil: A few-shot class-incremental learning approach for malware detection," *IEEE TIFS*, vol. 20, pp. 2999–3014, 2024.
[32] M. Chin and R. Corizzo, "Continual semi-supervised malware detection," *Mach. Learn. Knowl. Extr.*, vol. 6, no. 4, pp. 2829–2854, 2024.
[33] F. Barbero, F. Pendlebury, F. Pierazzi, and L. Cavallaro, "Transcending transcend: Revisiting malware classification in the presence of concept drift," *arXiv preprint arXiv:2010.03856*, 2020.
[34] K. Xu, Y. Li, R. Deng, K. Chen, and J. Xu, "Droidevolver: Self-evolving android malware detection system," in *2019 EuroS&P*, pp. 47–62, 2019.
[35] L. Yang, W. Guo, Q. Hao, A. Ciptadi, A. Ahmadzadeh, X. Xing, and G. Wang, "CADE: detecting and explaining concept drift samples for security applications," in *USENIX Security Symp.*, 2021.
[36] M. Botacin and H. Gomes, "Towards more realistic evaluations: The impact of label delays in malware detection pipelines," *Comput. & Secur.*, vol. 148, p. 104122, 2025.

Daniele Ghiani received his BSc in Computer Engineering (2021) and MSc in Computer Engineering, Cybersecurity, and AI (2024) from the University of Cagliari. He is currently a PhD student in the Italian national PhD programme in AI at Sapienza University, co-located at the University of Cagliari. His research addresses Continual Learning and regression issues in Android malware detection.

Daniele Angioni is a Postdoctoral Researcher at the University of Cagliari, Italy. He received his PhD in Artificial Intelligence in January 2025 from the Sapienza University of Rome. His research addresses machine learning security in the real world, applied to both malware and image domains. He serves as a reviewer for top-tier journals, including IEEE TIFS and Pattern Recognition.

Giorgio Piras is a Postdoctoral Researcher at the University of Cagliari. He received his PhD in Artificial Intelligence in January 2025 from the Sapienza University of Rome (with honors). His research mainly focuses on adversarial machine learning, with a particular attention to neural network pruning, explainable AI, and LLM security. He serves as a reviewer for journals and conferences, including Pattern Recognition and Neurocomputing journals.

Angelo Sotgiu is an Assistant Professor at the University of Cagliari. He received from the University of Cagliari (Italy) the PhD in Electronic and Computer Engineering in February 2023. His research mainly focuses on the security of machine learning, also considering practical applications like malware detection. He serves as a reviewer for several journals and conferences.



Luca Minnei is a Ph.D. student in Informatics, Electronics, and Computer Engineering at the University of Cagliari, Italy, he earned a B.Sc. in Computer Science in 2022 and an M.Sc. in Computer Engineering, Cybersecurity, and Artificial Intelligence in 2024, both with honors. His research focuses on malware detection and concept drift in machine learning security.

Srishti Gupta is currently a PhD student in Italian National PhD program in AI at Sapienza University, Rome co-hosted by the University of Cagliari. She received her MS from University of Arizona, US in 2021 and B.Tech from Bharati Vidyapeeth College in Delhi, India in 2017. Her research interests include Continual Learning, Out-of-Distribution Detection and security of LLM models. She serves as a reviewer to several journals and conferences.

Maura Pintor is an Assistant Professor at the University of Cagliari, Italy. She received her PhD in Electronic and Computer Engineering (with honors) in 2022 from the University of Cagliari. Her research interests include adversarial machine learning and trustworthy security evaluations of ML models, with applications in cybersecurity. She serves as an AC for NeurIPS, and as AE for Pattern Recognition.

Fabio Roli Fabio Roli is Full Professor of Computer Engineering at the Universities of Genova and Cagliari, Italy. He is Director of the sAIfer Lab. He is a recipient of the Pierre Devijver Award for his contributions to statistical pattern recognition. He has been appointed Fellow of the IEEE, Fellow of the International Association for Pattern Recognition, Fellow of the Asia-Pacific Artificial Intelligence Association.

Battista Biggio is Full Professor of Computer Engineering at the University of Cagliari, Italy. He has provided pioneering contributions to machine learning security. His work on "Poisoning Attacks against SVMs" won the 2022 ICML Test of Time Award. He chaired IAPR TC1 (2016-2020), and served as Associate Editor for IEEE TNNLS and IEEE CIM. He is Associate Editor-in-Chief for Pattern Recognition. He is IEEE Fellow, ACM Senior Member, and Member of IAPR, AAAI, and ELLIS.