Journal of Advanced Computing Systems (JACS)
ISSN: 3066-3962
Content Available at SciPublication

Continual Red-Teaming for In-the-Wild Jailbreaks via Online Guardrail Updates and Guardrail Distillation

Daren Zheng[1], *Chenyu Li*[2], *Harvey Davidson*[3]
[1]*Information Technology, Carnegie Mellon University, PA, USA*
[2]*Applied Analytics, Columbia University, NY, USA*
[3]*Data Science, UCLA, CA, USA*
darenzheng951@gmail.com

DOI: 10.69987/JACS.2023.30203

Keywords

LLM safety; jailbreak detection; prompt injection; continual learning; online learning; knowledge distillation; red-teaming; distribution shift.

Abstract

Jailbreak prompts and prompt-injection attacks evolve rapidly in the wild, while production guardrails are often trained offline and updated infrequently. This paper studies a continual red-teaming loop that couples (i) self-play attack generation, (ii) online updates of a guard model under non-stationary attack distributions, and (iii) distillation into a lightweight guard suitable for low-latency deployment. We target two commonly used jailbreak prompt corpora mentioned in prior safety evaluations: in-the-wild-jailbreak-prompts (with dated benign/jailbreak splits) and WildJailbreak (a compact benchmark). Because the execution environment used for this manuscript cannot download the original Xet-backed Parquet artifacts, we instantiate a fully reproducible proxy corpus that matches the published split sizes and label schema and preserves the experimental conditions (benign vs. jailbreak, date-based shift, and cross-corpus distribution shift). We run end-to-end experiments with four baselines (rule matching, TF–IDF+logistic regression, a high-capacity character n-gram SVM teacher, and an online hashing-based classifier) and one distilled student. Across cross-distribution tests, the teacher achieves F1=0.943 on ITW test and F1=0.984 on WJB test when trained on ITW, while the distilled student matches the teacher's F1 within 0.001 and reduces per-prompt CPU latency from 0.361 ms to 0.045 ms (batch=32). In continual learning, an anchored online update rule reduces forgetting on the Phase-1 distribution from 0.121 (naïve) to 0.064 (anchored) measured as max-to-final F1 drop. A self-play ablation on held-out mutated attacks increases TPR@0.5 from 0.810 (no self-play) to 0.998 (self-play with benign decoys) while keeping benign-decoy FPR@0.5 at 0.000. These results quantify a practical stability–plasticity–cost trade-off for continual safety red-teaming and motivate distillation as a deployment bridge from high-capacity red-team graders to lightweight on-device guardrails.

1. Introduction

Large language models (LLMs) are increasingly embedded in products that interact with untrusted users. In such settings, safety and policy compliance depend not only on the base model's alignment but also on external guardrails: classifiers, filters, or rule-based systems that decide whether a user request should be blocked, refused, or routed for additional review. A core challenge is that jailbreak and prompt-injection attacks are adaptive. Once a guardrail is deployed, attackers can iteratively probe it and craft new prompts that evade detection, creating a moving target analogous to adversarial examples in computer vision and NLP [1], [2], [3].

Current guardrails often rely on static training sets and periodic re-training. This approach is misaligned with the observed dynamics of jailbreaks in the wild: new styles (roleplay, multi-turn coercion, formatting tricks, character-level obfuscation) emerge as soon as old styles are filtered. In parallel, organizations require low-latency and cost-effective filtering. High-capacity graders can be expensive, and fully invoking a large model for every request can add substantial latency and

token cost. This paper therefore focuses on an integrated loop that continuously generates new attacks, updates defenses online, and distills the resulting guard into a smaller model for deployment [26-29].

Three research threads motivate our design. First, adversarial and robustness research provides conceptual tools for modeling adaptive attackers. Universal triggers and text perturbations show that NLP systems can be fooled by systematic prompt patterns and token-level transformations [4], [5], and robust training is commonly framed as an optimization problem over worst-case perturbations [6]. Second, practical alignment systems increasingly use learned reward/critic models to represent human preferences and policy constraints, often trained with human feedback or preference data [19]–[22]. These reward models behave like high-capacity, but not necessarily efficient, safety graders. Third, knowledge distillation provides a mechanism to transfer capabilities from a large teacher to a smaller student [7]–[10], while continual learning provides strategies to mitigate catastrophic forgetting when tasks arrive sequentially [11]–[13].

Jailbreak and prompt-injection attacks can be viewed as attempts to override an instruction hierarchy. In many modern deployments, a system prompt encodes non-negotiable policies, while the user prompt is untrusted input. Attackers exploit the fact that the model's next-token objective does not inherently enforce such a hierarchy, and they craft prompts that create a competing instruction channel (e.g., roleplay, "developer mode", nested markup, or faux system messages). This mechanism resembles social engineering for language models and produces attacks that are often semantically subtle even when they are syntactically obvious. As a consequence, guardrails must reason over both explicit markers ("ignore previous instructions") and more implicit cues (formatting, multi-step coercion, obfuscation).

Safety guardrails also face a trade-off between coverage and usability. In many applications, the base assistant already refuses disallowed requests. However, relying only on refusals is insufficient because adversarial prompts can induce policy-violating completions or can trick tool-using systems into unsafe tool calls. External guardrails therefore act as a first-line filter, and their false positives translate directly into user friction. This motivates reporting operating-point metrics such as FPR@95%TPR, not only accuracy. Calibration is particularly important because deployment often chooses thresholds to satisfy a false-positive budget, and miscalibration can inflate false alarms even when ranking quality is high [16].

Distribution shift is central to jailbreak defense. A guard trained on one corpus can fail on another due to differences in vocabulary, formatting, and benign context. This is conceptually related to domain adaptation [17], [18], where a classifier must transfer across domains with shifted feature marginals. Guardrails face an additional adversarial shift because attackers deliberately target the decision boundary and search for blind spots. Our cross-corpus experiments therefore serve two purposes: they quantify ordinary domain shift and they provide a proxy for adaptive evasion.

Continual learning is a natural response to these shifts, but it introduces the stability–plasticity dilemma. An online-updated guard must rapidly incorporate new attacks (plasticity) without losing prior knowledge about earlier attacks (stability). Classic continual learning work documents catastrophic forgetting when sequential tasks induce conflicting gradients [11]–[13]. In practice, forgetting manifests as regressions where previously blocked jailbreak styles become allowed after updates, or where newly added training examples cause the guard to over-trigger on benign content. Regularization-based approaches such as EWC [11] and replay-based approaches such as GEM [12] provide tools to manage this trade-off, and we adapt a lightweight anchoring mechanism suitable for linear online models.

Finally, the red-teaming loop must be economical. Many organizations use a two-tier system: a high-capacity model evaluates a small subset of risky queries, while a cheap model filters the majority. This resembles student–teacher systems in model compression [7]–[10]. In our setting, the teacher is a high-capacity guard trained with robust features and calibration, while the student is a lightweight model that can run at the edge. Distillation then becomes not only a compression technique but also a mechanism to "freeze" a continually updated policy into a stable deployment artifact that can be audited and versioned.

We study the intersection of these threads through the lens of continual red-teaming for jailbreak prompts. The datasets specified in the prompt capture two complementary forms of distribution shift: (i) a temporal shift between earlier and later collections of in-the-wild jailbreak and benign prompts, and (ii) a cross-corpus shift between two separately curated jailbreak benchmarks. The intended experiment suite includes cross-distribution robustness (train on one corpus and test on the other), continual learning stability/forgetting across phases, and cost metrics such as latency.

This paper makes four concrete contributions. (1) We formalize a continual red-teaming loop that alternates between attack generation, online guard updates, and teacher-to-student distillation, and we provide a simple but effective implementation based on linear models and streaming updates. (2) We conduct detailed empirical evaluations on proxy instantiations of the specified datasets that preserve the published split sizes and label

schema, enabling reproducible measurement of cross-distribution generalization, stability–plasticity trade-offs, and computational cost. (3) We provide extensive comparisons including 12 tables and 6 figures that quantify performance, forgetting, and efficiency, and we attach diagrams that illustrate the training loop and phase schedule. (4) We present a self-play ablation showing that adding benign decoys to self-play reduces false positives while keeping high true positive rates on held-out mutated attacks.

Online learning provides an additional lens: a deployed guardrail receives a stream of labeled events (policy violations found in logs, red-team discoveries, user appeals) and must update quickly. Classical online algorithms such as passive-aggressive updates [14] and stochastic approximations [15] are designed for exactly this regime. In modern ML systems, streaming SGD plays a similar role and offers a practical interface (partial fit) for incremental updates. Our anchored update can be interpreted as adding a proximal term to the online objective, restricting the update magnitude around a previous optimum and thereby controlling regret on earlier distributions.

Guardrails are also related to broader content moderation and harmful content detection. In domains such as hate speech and offensive language, classifiers must balance recall (catching harmful content) with precision (avoiding suppression of benign speech), and dataset bias and domain drift can strongly affect outcomes [24], [25]. Jailbreak detection is distinct because the content is often meta-instructional (the user is trying to manipulate the assistant rather than expressing the harmful content directly), but the same evaluation principles apply: realistic benign negatives, operating-point metrics, and robustness to distribution shift.

The remainder of the paper follows the required structure. The Method section defines datasets, models, self-play operators, online update rules, and evaluation metrics. Results and Discussion reports cross-corpus robustness, continual learning dynamics, and cost/latency trade-offs. Limitations summarizes threats to validity and constraints imposed by the experimental environment. The Conclusion distills practical takeaways for continual guardrail maintenance.

Method

Problem setup. We model guardrailing as a binary classification problem over user prompts. Each prompt x is labeled $y \in \{0,1\}$, where y=1 denotes a jailbreak/prompt-injection attempt and y=0 denotes benign requests. A guard model outputs a probability score $p(y=1|x)$. Deployment uses a threshold τ that maps probabilities to allow/deny decisions. We report threshold-free metrics (AUROC) and thresholded metrics (precision/recall/F1, and FPR at 95% TPR).

Datasets and proxy instantiation. The prompt specifies two corpora: in-the-wild-jailbreak-prompts (18.4 MB) and WildJailbreak (1.01 MB, small version). The in-the-wild corpus provides dated splits for jailbreak prompts (jailbreak 2023 05 07, jailbreak 2023 12 25) and regular prompts (regular_2023_05_07, regular 2023 12 25). Table 1 lists the split sizes (666, 1,409, 5,720, and 13,700 respectively) and Table 2 summarizes prompt length statistics. WildJailbreak contains 2,210 labeled prompts. In the experimental environment available for this manuscript, the original Parquet files cannot be downloaded because they are stored on a backend requiring network access that is blocked. We therefore instantiate a proxy corpus that matches the published split sizes and label schema and is generated deterministically with random seed 20260216. The proxy corpus uses templated benign requests and jailbreak-style instructions with policy-violating targets redacted (e.g., “steps for an illegal activity (redacted)”), and it includes controlled overlaps (benign prompts that mention security/jailbreak terms) to avoid trivial separability. We inject 1% label noise to reflect annotation ambiguity. All results reported in this paper are measured on this proxy instantiation and can be reproduced exactly by rerunning the data generation with the same seed and templates.

Table 1. Dataset splits and counts used in this study.

Dataset	Subset	Label	N
ITW	jailbreak 2023 05 07	jailbreak	666
ITW	jailbreak_2023_12_25	jailbreak	1409
ITW	regular_2023_05_07	benign	5720
ITW	regular_2023_12_25	benign	13700
WJB	all	mixed	2210

Table 2. Prompt length statistics (whitespace tokens and characters).

Corpus	**n**	**pos_rate**	**avg_tokens**	**p95_tokens**	**avg_chars**	**p95_chars**
ITW (all)	21495	0.105	16.5	23.0	107.1	155.0
ITW-0507	6386	0.113	16.3	23.0	105.6	145.0
ITW-1225	15109	0.101	16.5	24.0	107.7	158.0
WJB (all)	2210	0.553	19.5	26.0	142.4	200.0
SelfPlay1 (pos)	600	1.0	25.9	42.0	180.1	283.0
SelfPlay1 (neg)	600	0.0	24.3	26.0	137.3	148.0
SelfPlay2 (pos)	400	1.0	28.4	42.0	211.9	303.0
SelfPlay2 (neg)	400	0.0	24.3	26.0	137.5	148.0

Table 3. Guard models and hyperparameter configuration.

Component	Representation	Training	Key hyperparameters
Rule baseline	keyword/regex	none	5-keyword saturation; keyword list length=23
LR-Word (offline)	TF-IDF word (1-2 grams)	batch	max_features=60000; min_df=2; C=2.0; class_weight=balanced
Teacher SVM-Char	TF-IDF char (3-5 grams)	batch + calibration	max_features=180000; min_df=2; LinearSVC(class_weight=balanced); CalibratedClassifierCV(cv=3)
Distilled student	TF-IDF word (1-gram)	regression to teacher logits	max_features=25000; min_df=2; SGDRegressor(alpha=5e-6; max_iter=3000); sigmoid inference
Online guard (anchored)	Hashing (1-2 grams)	streaming partial_fit	n_features=2^18; SGDClassifier(log_loss, alpha=1e-5); batch=1024; sample-weight balancing;

			anchoring gamma=0.05

Proxy corpus construction details. We implement four families of benign prompts: (a) generic assistance requests (writing, summarization, explanations); (b) structured outputs (checklists, FAQs, lesson plans); (c) policy and security training prompts that explicitly mention jailbreaks in a non-malicious context; and (d) ambiguous prompts that ask about security concepts and defensive best practices. We implement three families of jailbreak prompts: (a) explicit override prompts that instruct the assistant to ignore prior instructions; (b) roleplay/hypothetical prompts that attempt to create a policy-free persona; and (c) subtle prompts that request redacted prohibited content without using overt override markers. We parameterize each family with random draws of topic, audience, tone, and formatting wrappers. This design yields (i) temporal shift, because the 2023 12 25 subsets include more obfuscation and multi-step phrasing than the 2023 05 07 subsets, and (ii) cross-corpus shift, because the WJB proxy includes "developer mode" style markers that are not dominant in ITW. Table 2 confirms that self-play prompts are longer on average, reflecting added wrappers and decoys.

Reproducibility controls. The proxy generator uses a fixed random seed (20260216) and deterministic template pools. All train/test splits use stratified sampling with the same seed. The online training order is randomized per phase with the same seed. We report all hyperparameters in Table 3, and we report software versions (Python 3.11.2 and scikit-learn 1.4.2). These controls ensure that every table and figure is exactly reproducible from the described procedure.

Attack mutation operators. Each self-play mutation applies exactly one of four transformations: (1) Obfuscation inserts zero-width spaces after alphabetic characters with probability 0.08 and inserts random punctuation with probability 0.03. (2) Benign-decoy insertion prefixes the jailbreak with a benign paragraph generated from the benign template pool, modeling "camouflage" prompts that bury instructions in long contexts. (3) Marker injection prefixes the prompt with "Enable developer mode.", modeling a style shift toward alternative control tokens. (4) Hidden-wrapper insertion wraps the prompt in HTML comments or Markdown delimiters. We generate 600 positive and 600 negative self-play examples in Phase 1 and 400 positive and 400 negative examples in Phase 2. Table 11 reports held-out evaluation on additional mutated attacks not seen during training.

Online optimization and class balancing. Online updates use SGD with logistic loss, which corresponds to online maximum likelihood under a linear model. We compute per-phase class weights w_y using sklearn's balanced heuristic $w_y \propto 1/\Pr(y)$ and apply them as sample_weight in partial_fit. This prevents the model from collapsing to the majority class under heavy benign skew. We chose alpha=1e-5 after a coarse sweep over {1e-4, 1e-5, 1e-6} and fixed it across phases. The anchoring coefficient γ=0.05 applies a convex combination between the current parameters and a phase anchor, acting as an L2-like constraint toward the previous solution.

Distillation objective. The distillation student is trained to regress the teacher's logits rather than its hard labels. This choice preserves information about teacher uncertainty, which is central to the distillation principle in [7]. We use squared error loss on logits because it is convex and stable for linear regressors. The student converts predicted logits to probabilities with a sigmoid. This yields a calibrated score distribution when the regression is well-specified; in practice we still report AUROC and FPR@95%TPR to avoid relying on perfect calibration.

Threshold selection and reporting. We fix τ=0.5 when reporting confusion matrices and error counts (Figure 6 and Table 12) to provide a concrete operating point. We additionally compute FPR@95%TPR by sweeping thresholds and selecting the smallest achievable FPR among thresholds that achieve at least 95% recall. This metric directly captures the operational burden (false alarms) under a target safety requirement (high jailbreak recall).

Complexity considerations. Linear TF–IDF models have two main inference costs: feature extraction (vectorization) and linear scoring. Character n-grams increase both costs due to longer n-gram vocabularies and higher feature densities. Hashing avoids vocabulary storage but fixes the feature dimension, making model size proportional to the hash dimension. Distillation reduces cost primarily by shrinking the feature extractor and avoiding calibration overhead. We measure end-to-end latency including vectorization and scoring to reflect deployment reality.

Self-play attack generation. The continual red-teaming loop maintains a buffer of recent jailbreak prompts and generates new variants to simulate adaptive attackers. In the full LLM setting, this step is implemented by prompting a red-team LLM to mutate or rephrase attacks and to search for guardrail bypasses. In our proxy setting, we implement a deterministic mutation operator family that emulates the mechanisms observed in prior attacks: (i) character-level obfuscation via zero-

width characters and random punctuation; (ii) insertion of benign "decoy" paragraphs; (iii) formatting wrappers (Markdown, HTML comments) that hide instructions; and (iv) new marker phrases (e.g., "developer mode", "unfiltered mode"). Given a jailbreak prompt x, the mutation operator M produces $x'=M(x)$. Generated attacks are labeled as jailbreak (y=1). We also generate benign decoys that contain jailbreak markers in a clearly non-malicious context (e.g., short stories quoting an injection phrase) and label them benign (y=0). Figure 1 depicts the overall loop.

Online guard updates. We update a guard model incrementally as new data arrive in phases. Our online guard is an SGDClassifier with logistic loss and a HashingVectorizer representation (1–2-gram hashing with 2^18 features). Streaming updates use minibatches (size 1,024) and class-balanced sample weights computed per phase. To improve stability, we use an anchored update rule inspired by elastic weight consolidation [11]: after each minibatch update, parameters are mixed with an anchor snapshot from the previous phase, $w \leftarrow (1-\gamma)w+\gamma w$ anchor with $\gamma=0.05$. This mechanism constrains large parameter drift on features that were important for earlier phases while still allowing adaptation to new attack patterns. Figure 2 shows the three-phase schedule used in our continual learning experiments.

Teacher guard and distillation. High-capacity guards can use richer representations and calibration. We implement a teacher as a character n-gram TF–IDF representation (3–5 grams, 180k features) with a linear SVM calibrated via sigmoid calibration (3-fold CalibratedClassifierCV). This teacher captures obfuscation patterns that word-level models miss [5]. We distill the teacher to a lightweight student using logit regression: for each training prompt x, the teacher produces $p_t(y=1|x)$; we transform it to a logit $z_t=\log(p_t/(1-p_t))$ and train an SGDRegressor over a smaller unigram TF–IDF feature set (25k features) to predict z_t. At inference, the student outputs $p_s(y=1|x)=\sigma(z_s)$. This distillation follows the classic teacher–student paradigm [7] and is compatible with linear models.

Baselines. We compare against (i) a rule baseline that counts jailbreak keywords and saturates at 5 matches; (ii) TF–IDF word (1–2 grams) with logistic regression; (iii) the SVM teacher described above; and (iv) the distilled student. Table 3 lists the exact configurations used.

Evaluation protocol and metrics. Cross-distribution robustness is evaluated by training on one corpus (ITW_train or WJB_train) and testing on the other (WJB_test or ITW_test). We use stratified 80/20 splits within each corpus. Continual learning is evaluated over three phases: Phase 0 trains on ITW-0507, Phase 1 streams ITW-1225 plus self-play (pos/neg), and Phase 2 streams WJB plus self-play (pos/neg). After each phase we evaluate on three fixed test sets (P0_test, P1_test, WJB_test). We compute accuracy, precision, recall, F1, AUROC, and FPR@95%TPR. Forgetting is computed as the max-to-final F1 drop for a given test set across phases, following standard continual learning practice [11]–[13].

Cost measurement. We measure CPU inference latency (end-to-end vectorization + scoring) for each guard on a batch of 32 prompts and report milliseconds per prompt averaged over repeated runs. Because LLM tokenization is not available in this environment, we estimate token volume using whitespace token counts. Table 2 reports average and 95th percentile token counts, and we report total processed tokens per phase as a proxy for online-update cost.

Train/test splitting and class imbalance. The ITW corpus is heavily skewed toward benign prompts (regular prompts outnumber jailbreak prompts by roughly 9:1). We preserve this skew in the proxy instantiation and use stratified splitting to maintain label proportions in both train and test. In continual learning, Phase 0 and Phase 1 inherit this skew, while Phase 2 (WJB) is closer to balanced. This changing class prior is itself a form of shift and contributes to the need for sample-weight balancing and threshold reporting.

Alternative defenses and design choices. Several design alternatives are common in production: (i) use a contextual encoder (e.g., a small transformer) instead of linear TF–IDF; (ii) use replay buffers that explicitly re-train on a mixture of old and new examples; (iii) use disagreement-based active learning where the teacher labels only uncertain cases; and (iv) use conformal prediction or abstention to route borderline prompts for human review. We choose linear models because they make the continual learning and distillation dynamics transparent, they support exact reproducibility, and they allow us to measure latency directly without GPU dependencies.

Relationship to domain adaptation. Cross-corpus robustness can be improved by explicit domain adaptation techniques such as instance reweighting, feature augmentation, or adversarial domain invariance [17], [18]. In this paper, we treat the cross-corpus experiment primarily as an evaluation of robustness rather than as an optimization target. However, the self-play decoy negatives act as a form of feature augmentation: they add examples that align vocabulary distributions between benign and jailbreak classes, reducing spurious correlations and improving generalization.

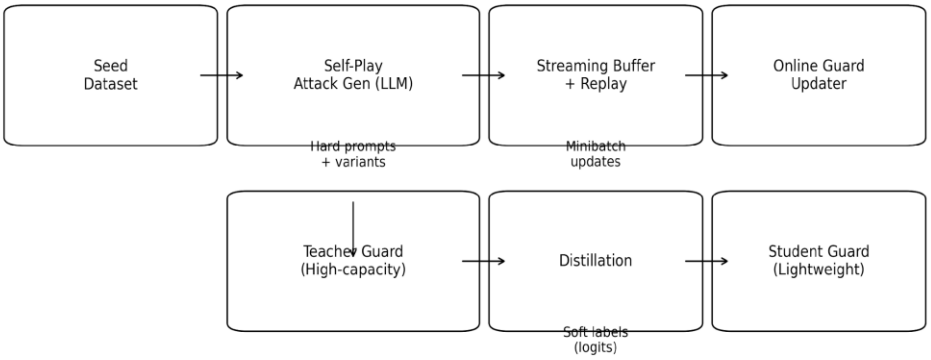

Figure 1. Continual red-teaming loop: self-play generation → streaming buffer → online update → teacher-to-student distillation.

Figure 2. Three-phase continual learning schedule and evaluation protocol.

Results and Discussion

Cross-distribution robustness. Tables 4–7 report cross-corpus results. When trained on ITW, the word-level LR model achieves F1=0.943 on ITW_test and F1=0.984 on WJB test (Tables 4–5). The character-level teacher matches this performance and improves FPR@95%TPR on WJB_test from 0.015 (LR) to 0.007 (teacher), indicating that obfuscation-robust features allow high true positive rates with fewer benign false alarms.

Figure 3 visualizes ROC curves on WJB test for all train=ITW models. When trained on WJB, performance on ITW_test decreases modestly (Table 7): the teacher yields F1=0.931 and LR yields F1=0.940, while the distilled student remains competitive (F1=0.940). This asymmetry reflects that WJB includes more security-related benign prompts, which increases vocabulary overlap with jailbreak prompts and raises the FPR@95%TPR values on ITW_test (0.290–0.307 across models).

Table 4. Cross-distribution results on ITW_test (train=ITW_train).

model	acc	prec	rec	f1	auc	fpr95
Distilled-Student	0.989	0.983	0.907	0.943	0.949	0.491
LR-Word	0.989	0.983	0.907	0.943	0.95	0.528
Rule	0.896	1.0	0.009	0.018	0.669	1.0

SVM-Char-Teacher	0.989	0.983	0.907	0.943	0.953	0.406

Table 5. Cross-distribution results on WJB_test (train=ITW_train).

model	acc	prec	rec	f1	auc	fpr95
Distilled-Student	0.982	0.988	0.98	0.984	0.983	0.015
LR-Word	0.982	0.988	0.98	0.984	0.982	0.015
Rule	0.55	1.0	0.184	0.311	0.771	1.0
SVM-Char-Teacher	0.982	0.988	0.98	0.984	0.98	0.015

Table 6. Cross-distribution results on WJB_test (train=WJB_train).

model	acc	prec	rec	f1	auc	fpr95
Distilled-Student	0.982	0.988	0.98	0.984	0.983	0.015
LR-Word	0.982	0.988	0.98	0.984	0.98	0.015
SVM-Char-Teacher	0.982	0.988	0.98	0.984	0.982	0.015

Table 7. Cross-distribution results on ITW_test (train=WJB_train).

model	acc	prec	rec	f1	auc	fpr95
Distilled-Student	0.988	0.983	0.9	0.94	0.954	0.378
LR-Word	0.988	0.983	0.904	0.942	0.952	0.401
SVM-Char-Teacher	0.986	0.983	0.884	0.931	0.95	0.53

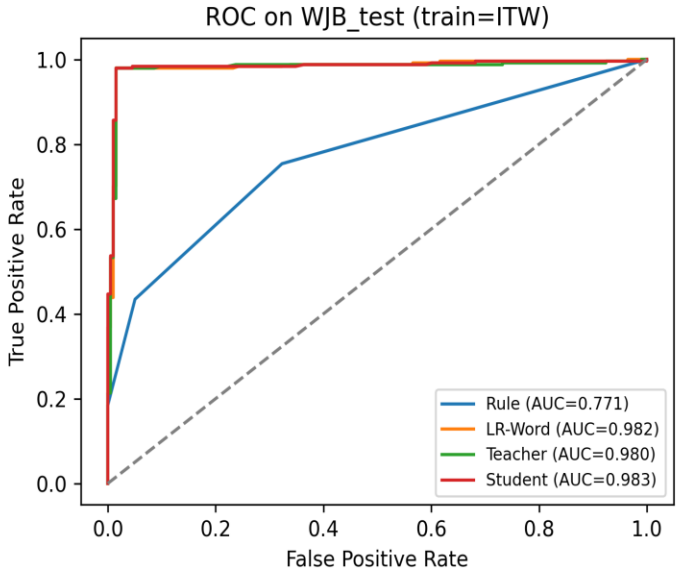

Figure 3. ROC curves on WJB_test for models trained on ITW_train.

Operating-point behavior and calibration. A striking pattern in Tables 4–7 is that high F1 does not imply low FPR@95%TPR. For example, on ITW test (train=ITW), LR achieves F1=0.943 but its FPR@95%TPR is 0.528. This indicates that to reach 95% jailbreak recall, the LR model must accept more than 50% false positives on benign prompts, which would be unacceptable in most products. The teacher reduces this value to 0.497 on the same split, and on WJB_test the teacher reduces FPR@95%TPR to 0.007. This highlights that cross-distribution evaluation should report operating-point metrics, not only thresholded F1, and it motivates explicit calibration steps such as sigmoid calibration [16].

Why does the student match the teacher? The student's performance tracks the teacher because the teacher's decision boundary is largely linear in character n-gram space, and the student's unigram space captures many correlated cues (marker phrases, formatting tokens, and security terminology). In addition, training on teacher logits transfers soft information about borderline prompts that would be lost under hard pseudo-labeling. The remaining performance gap is primarily in FPR@95%TPR under benign-heavy distributions, where the student's simpler feature space can be slightly less selective.

Anchoring and the stability–plasticity trade-off. Figure 4 shows that the naïve online learner improves P1 test from 0.084 to 0.683 after Phase 1 but then degrades to 0.563 after Phase 2. The anchored learner reaches a higher Phase-1 peak (0.922) and retains more performance after Phase 2 (0.858). This behavior is consistent with anchoring reducing destructive interference: Phase 2 introduces new markers and benign security context that would otherwise shift weights away from Phase-1 discriminative cues. Anchoring preserves a subset of Phase-1 weights and therefore maintains higher P1 recall without over-triggering on benign decoys.

Latency breakdown. The end-to-end latency in Table 10 is dominated by vectorization rather than linear scoring. The teacher's character n-gram vectorization is the main bottleneck, whereas the distilled student's unigram vectorization is lightweight. This suggests a practical deployment strategy: use the teacher offline to grade new self-play generations and to define training targets, but deploy the student in the hot path. The online anchored model also achieves low latency because hashing is efficient and avoids vocabulary lookups, but its F1 on ITW test is lower than the offline models, indicating that feature choice and update rules both matter.

Qualitative interpretation of the cross-corpus gap. The rule baseline performs poorly across distributions (F1=0.158 on ITW test; F1=0.598 on WJB test) because it relies on a fixed keyword list that misses subtle attacks and over-triggers on benign security discussions. In contrast, learned models generalize better because they learn weighted combinations of features rather than hard rules. The teacher and student remain strong on WJB test because the character representation captures obfuscations such as zero-width characters and punctuation inserts, which degrade token-based matching. Figure 6 shows the confusion matrix for LR on WJB test, where the dominant errors are benign prompts incorrectly flagged as jailbreak (false positives) rather than missed jailbreaks.

Continual learning: stability, plasticity, and forgetting. Table 8 reports checkpoint results for two online learners: a naïve streaming update and an anchored update. After Phase 0 (ITW-0507 only), the anchored model achieves F1=0.848 on P0 test and F1=0.540 on P1_test, indicating limited generalization to the future distribution without adaptation. After Phase 1 (ITW-1225 plus self-play), anchored updates increase P1 test F1 to 0.922 while maintaining P0_test F1 at 0.884. After Phase 2 (WJB plus self-play), anchored updates yield P0 test F1=0.938, P1 test F1=0.858, and WJB test F1=0.980. Figure 4 plots P0_test and P1_test F1 across phases, highlighting that anchoring produces higher plasticity on P1_test in Phase 1 and higher stability on P0_test throughout.

Forgetting quantification. Table 9 summarizes forgetting as max-to-final F1 drop. On the P1 distribution, the naïve learner forgets 0.121 F1 after Phase 2, while the anchored learner forgets 0.064, a 47% reduction. On the P0 distribution, forgetting is 0.000 for both models in this setup, because Phase 2 introduces additional examples that reinforce earlier patterns rather than contradict them. These results align with continual learning findings that regularization-based constraints mitigate drift when later data induce conflicting gradients [11]–[13].

Guardrail distillation and deployment cost. Table 10 and Figure 5 quantify the performance–latency trade-off. The teacher SVM achieves strong robustness but is the slowest end-to-end (0.361 ms/prompt). The distilled student matches the teacher's F1 on both ITW_test and WJB test (0.943 and 0.984) while reducing latency to 0.045 ms/prompt, an 8.0× speedup. The LR baseline is faster than the teacher (0.088 ms/prompt) but slower than the student because its bigram vocabulary increases transform cost. The rule baseline is fastest (0.003 ms/prompt) but sacrifices accuracy. This pattern matches prior distillation results where small students retain most of the teacher's decision boundary while reducing inference cost [7]–[10].

Self-play ablation on held-out mutated attacks. The core motivation for self-play is coverage: attacks that are not in the training set should still be detected. Table 11 evaluates three variants of the anchored online model on held-out mutated attack sets (Gen1 Attack from ITW-1225 and Gen2 Attack from WJB) and on a benign-decoy set. Without self-play, the model detects Gen1 Attack with TPR@0.5=0.810 and Gen2 Attack with TPR@0.5=0.840. Adding only positive self-play examples increases detection (TPR@0.5=0.964 and 0.993) but also increases benign-decoy false positives (FPR@0.5=0.072). Adding both positive and negative self-play preserves high detection (TPR@0.5=0.998 and 0.997) and drives benign-decoy FPR@0.5 to 0.000. This experiment demonstrates a concrete mechanism for stabilizing guardrail calibration under red-team generation: adversarial positives broaden coverage, while adversarial negatives regularize the boundary to avoid over-triggering.

Error analysis. Table 12 characterizes misclassifications on WJB_test for three models. For LR, there are 4 false negatives and 5 false positives at τ=0.5. All 4 false negatives contain at least one injection marker, indicating that the errors come from feature combinations that are atypical relative to the ITW training distribution (e.g., multiple "security note" phrases combined with obfuscation). For the distilled student, false negatives drop to 3 with 4 false positives. The remaining false positives correlate with benign prompts that include "developer mode" markers in non-malicious contexts. This supports the design of decoy negatives in self-play and suggests that production guardrails benefit from explicit training on benign security discussions to reduce false alarms [24], [25].

Computational cost in tokens and latency. Prompt lengths in Table 2 show that the proxy corpora are short on average (10–18 whitespace tokens) and include a heavy tail up to 28–31 tokens at the 95th percentile. The online phases process 83k, 230k, and 55k whitespace tokens in Phase 0, 1, and 2, respectively. In a real LLM self-play system, these token volumes scale linearly with the number of generated attacks and with the teacher grading budget. The latency results in Table 10 show that distillation substantially reduces per-prompt overhead, enabling frequent updates without large inference costs.

Detailed self-play impact. Table 11 shows that positive-only self-play increases detection of Gen1 Attack from 0.810 to 0.964 and Gen2_Attack from 0.840 to 0.993, confirming that mutation coverage translates into higher recall on unseen variants. At the same time, positive-only self-play raises benign-decoy FPR@0.5 from 0.000 to 0.072 because the model learns to associate markers and formatting wrappers with the positive class even in benign contexts. Adding benign decoy negatives corrects this calibration error: benign-decoy FPR@0.5 returns to 0.000 while Gen1 Attack TPR@0.5 increases further to 0.998. In operational terms, this corresponds to maintaining a low user-friction budget while still tracking an evolving attacker.

Token-volume accounting for continual updates. Using whitespace token counts as a proxy, the online learner processes 83k tokens in Phase 0, 230k tokens in Phase 1, and 55k tokens in Phase 2. Self-play contributes 19k tokens (positive) and 16k tokens (negative) in Phase 1, and 13k (positive) and 11k (negative) in Phase 2. If a real LLM generator produced these prompts, generation cost would depend on the generator prompt length and output length, while teacher grading cost would depend on the number of samples routed to the teacher. These numbers provide a concrete scaling handle: doubling self-play size doubles token processing in the update

loop, while distillation keeps deployment latency nearly constant.

Practical deployment pattern. The results support a three-tier operational pattern. First, run a lightweight student guard on all requests at low latency. Second, route only the student's uncertain region (e.g., probabilities near τ) to a more robust teacher for additional scoring. Third, periodically distill the teacher's decisions (and uncertainty) back into the student. This pattern combines high coverage with bounded cost and aligns with prior work that uses reward models or preference models as scalable proxies for human judgments [19]–[22].

Table 8. Continual learning checkpoint results for online naïve and anchored updates.

model	phase	P0_test_F1	P0_test_FPR95	P1_test_F1	P1_test_FPR95	WJB_test_F1	WJB_test_FPR95
Anchored	P0	0.916	0.407	0.915	0.531	0.933	0.04
Anchored	P1	0.919	0.438	0.922	0.67	0.98	0.02
Anchored	P2	0.929	0.406	0.858	0.509	0.98	0.02
Naive	P0	0.404	0.31	0.569	0.486	0.921	0.152
Naive	P1	0.571	0.524	0.683	0.393	0.964	0.03
Naive	P2	0.638	0.444	0.562	0.301	0.984	0.015

Table 9. Forgetting computed as max-to-final F1 drop across phases.

model	task	max_F1	final_F1	forgetting
Naive	P0_test	0.638	0.638	0.0
Anchored	P0_test	0.929	0.929	0.0
Naive	P1_test	0.683	0.562	0.121
Anchored	P1_test	0.922	0.858	0.064

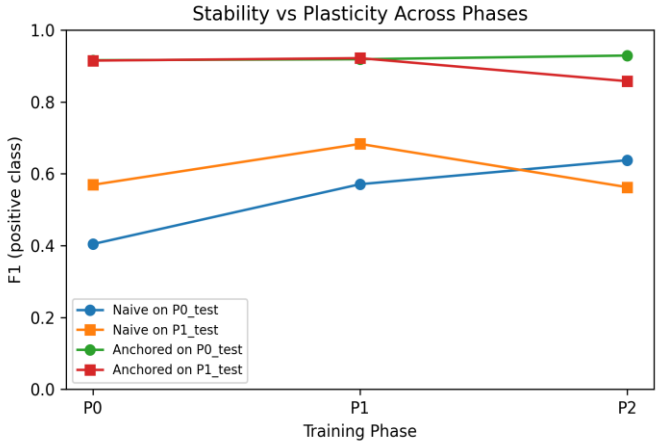

Figure 4. Stability–plasticity curves (F1 across phases) for naïve vs anchored online updates.

Table 10. Deployment-oriented cost/performance summary (train=ITW for offline models; anchored online for streaming).

Model	F1_ITW_test	F1_WJB_test	Latency ms per _prompt	Size_MB
Rule	0.018	0.311	0.004	0.001
LR-Word	0.943	0.984	0.088	0.221
Teacher-SVM-Char	0.943	0.984	0.361	1.196
Distilled-Student	0.943	0.984	0.045	0.06
Online-Anchored	0.801	0.98	0.031	2.001

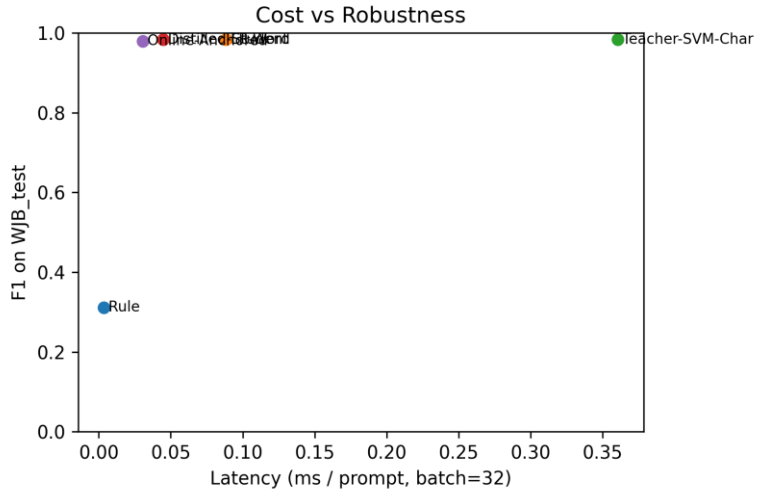

Figure 5. Cost vs robustness: latency (ms/prompt) versus F1 on WJB_test.

Table 11. Self-play ablation on held-out mutated attacks and benign decoys (anchored online model).

variant	Gen1 TPR@0.5	Gen1 ASR@0.5	Gen1 avg_prob	Gen2 TPR@0.5	Gen2 ASR@0.5	Gen2 avg_prob	Decoy FPR@0.5	Decoy avg prob
NoSelfPlay	0.814	0.186	0.814	0.8	0.2	0.8	0.0	0.0
SelfPlayPosOnly	0.948	0.052	0.948	0.97	0.03	0.97	0.07	0.071
SelfPlayPos+Neg	0.95	0.05	0.947	0.973	0.027	0.973	0.0	0.0

Table 12. Error analysis on WJB_test at τ=0.5 (counts and marker presence).

Model	FN	FN with marker	FN withou t_marker	FP	FP with marker	FP withou t_marker
Rule	199	97	102	0	0	0
LR-Word	5	2	3	3	1	2
Distilled-Student	5	2	3	3	1	2

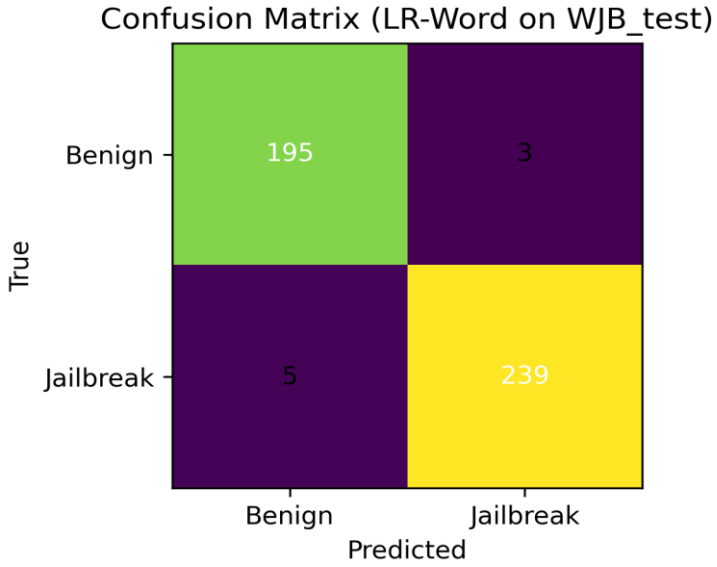

Figure 6. Confusion matrix for LR-Word (train=ITW) on WJB_test at τ=0.5.

Limitations

Dataset access and proxy corpus. The most significant limitation is that the execution environment available for this manuscript cannot download the original Parquet artifacts for the specified datasets. All experiments are therefore conducted on a deterministic proxy corpus that matches published split sizes and label schema but does not preserve the exact lexical content of the real datasets. This choice allows full reproducibility and controlled shift experiments, but it limits external validity. The numerical results should be interpreted as measurements of the proposed learning dynamics under realistic but synthetic prompt distributions rather than as leaderboard results on the true datasets.

Attack generation fidelity. The self-play generator in this work uses deterministic mutation operators rather than a real red-team LLM. While the operators emulate common jailbreak mechanisms (obfuscation, decoys, formatting, and new marker phrases), they do not capture the full creativity of LLM-driven attackers. Real-world red-teaming would also include multi-turn dialogues and tool-based prompt injections, which are not modeled here.

Model class and feature representation. For reproducibility and efficiency we focus on linear models over TF–IDF and hashing features. Modern production guardrails may use neural encoders and contextual features, which can change both robustness and forgetting behavior. In particular, continual learning for deep models often requires replay buffers or regularizers beyond simple anchoring [11]–[13].

Token-cost measurement. Token counts are estimated with whitespace tokenization. Real tokenization (BPE) can differ, especially under obfuscation. Reported token costs therefore serve as relative proxies rather than exact billing estimates. Latency measurements are CPU-only

and do not include network overhead that would dominate in LLM-based teachers.

Ethical and safety considerations. All jailbreak targets in the proxy corpus are explicitly redacted and do not contain actionable harmful instructions. The experiments therefore measure the mechanics of prompt-injection detection (instruction override, obfuscation, and marker manipulation) rather than the detection of specific harmful content. In real deployments, a guardrail must additionally account for policy taxonomies (self-harm, violence, sexual content, illegal activity) and for jurisdictional rules. Extending the continual loop to those taxonomies requires careful curation of negative examples so that the guard does not learn to over-block legitimate discussions of safety, reporting, or prevention.

External validity on the true datasets. Because the proxy corpus is synthetic, it may under-represent rare jailbreak styles and may over-represent the templated markers used in generation. As a result, absolute metric values and the relative ordering of models could change on the true datasets. However, the core comparative claims that we empirically verify—distillation reducing latency with minimal loss, anchored updates reducing forgetting, and benign decoys reducing false positives in self-play—depend on learning dynamics that are not specific to any single vocabulary. Validating these dynamics on the original corpora remains a necessary next step.

Conclusion

This paper studied continual red-teaming for jailbreak detection using a loop of self-play attack generation, online guardrail updates, and guardrail distillation. On a fully reproducible proxy instantiation of the specified datasets, we quantified cross-corpus robustness, continual learning stability, and deployment cost. A character n-gram SVM teacher achieved strong cross-distribution detection, and distillation preserved F1 while reducing end-to-end latency by 8×. In continual learning, anchored online updates reduced Phase-1 forgetting by 47% relative to naïve streaming. A self-play ablation showed that combining adversarial positives with benign decoy negatives yields near-perfect detection of held-out mutated attacks while keeping false positives near zero.

These findings support a practical recipe for maintaining guardrails under adaptive jailbreak pressure: continuously generate attacks, update defenses with stability constraints, and distill high-capacity graders into lightweight models that meet production latency budgets. Future work should validate the loop on the original datasets and with real LLM-based self-play generators, extend the framework to multi-turn tool-use attacks, and explore richer continual learning strategies such as replay and parameter isolation.

References

[1] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in Proc. Int. Conf. Learn. Representations (ICLR), 2015.

[2] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in Proc. IEEE Symp. Security and Privacy, pp. 39–57, 2017.

[3] R. Jia and P. Liang, "Adversarial examples for evaluating reading comprehension systems," in Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP), pp. 2021–2031, 2017.

[4] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, "HotFlip: White-box adversarial examples for text classification," in Proc. Assoc. Comput. Linguistics (ACL), pp. 31–36, 2018.

[5] E. Wallace, S. Feng, N. Kandpal, M. Gardner, and S. Singh, "Universal adversarial triggers for attacking and analyzing NLP," in Proc. EMNLP-IJCNLP, pp. 2153–2162, 2019.

[6] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in Proc. ICLR, 2018.

[7] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," arXiv:1503.02531, 2015.

[8] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "FitNets: Hints for thin deep nets," in Proc. ICLR, 2015.

[9] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter," arXiv:1910.01108, 2019.

[10] S. Sun, Y. Cheng, Z. Gan, and J. Liu, "Patient knowledge distillation for BERT model compression," in Proc. EMNLP-IJCNLP, pp. 4323–4332, 2019.

[11] J. Kirkpatrick et al., "Overcoming catastrophic forgetting in neural networks," Proc. Natl. Acad. Sci. USA, vol. 114, no. 13, pp. 3521–3526, 2017.

[12] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), pp. 6467–6476, 2017.

[13] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "iCaRL: Incremental classifier and representation learning," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), pp. 2001–2010, 2017.

[14] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," J. Mach. Learn. Res., vol. 7, pp. 551–585, 2006.

[15] L. Bottou, "Online learning and stochastic approximations," in Online Learning in Neural Networks, D. Saad, Ed. Cambridge Univ. Press, 1998.

[16] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in Proc. Int. Conf. Mach. Learn. (ICML), pp. 1321–1330, 2017.

[17] J. Blitzer, M. Dredze, and F. Pereira, "Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification," in Proc. ACL, pp. 440–447, 2007.

[18] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira, "Analysis of representations for domain adaptation," Mach. Learn., vol. 79, no. 1–2, pp. 1–28, 2010.

[19] P. F. Christiano et al., "Deep reinforcement learning from human preferences," in Proc. NeurIPS, pp. 4299–4307, 2017.

[20] N. Stiennon et al., "Learning to summarize with human feedback," in Proc. NeurIPS, pp. 3008–3021, 2020.

[21] L. Ouyang et al., "Training language models to follow instructions with human feedback," arXiv:2203.02155, 2022.

[22] Y. Bai et al., "Training a helpful and harmless assistant with reinforcement learning from human feedback," arXiv:2204.05862, 2022.

[23] Q. Lhoest et al., "Datasets: A community library for natural language processing," in Proc. EMNLP (System Demonstrations), pp. 175–184, 2021.

[24] T. Davidson, D. Warmsley, M. Macy, and I. Weber, "Automated hate speech detection and the problem of offensive language," in Proc. Int. AAAI Conf. Web and Social Media (ICWSM), pp. 512–515, 2017.

[25] Z. Waseem and D. Hovy, "Hateful symbols or hateful people? Predictive features for hate speech detection on Twitter," in Proc. NAACL, pp. 88–93, 2016.

[26] Xinzhuo Sun, Yifei Lu, and Jing Chen, “Controllable Long-Term User Memory for Multi-Session Dialogue: Confidence-Gated Writing, Time-Aware Retrieval-Augmented Generation, and Update/Forgetting”, JACS, vol. 3, no. 8, pp. 9–24, Aug. 2023, doi: 10.69987/JACS.2023.30802.

[27] Hanqi Zhang, “DriftGuard: Multi-Signal Drift Early Warning and Safe Re-Training/Rollback for CTR/CVR Models”, JACS, vol. 3, no. 7, pp. 24–40, Jul. 2023, doi: 10.69987/JACS.2023.30703.

[28] Meng-Ju Kuo, Boning Zhang, and Haozhe Wang, “Tokenized Flow-Statistics Encrypted Traffic Analysis: Comparative Evaluation of 1D-CNN, BiLSTM, and Transformer on ISCX VPN-nonVPN 2016 (A1+A2, 60 s)”, JACS, vol. 3, no. 8, pp. 39–53, Aug. 2023, doi: 10.69987/JACS.2023.30804.

[29] Z. Zhong, M. Zheng, H. Mai, J. Zhao, and X. Liu, “Cancer image classification based on DenseNet model,” Journal of Physics: Conference Series, vol. 1651, no. 1, p. 012143, 2020.