CLAD: CONTINUAL LEARNING FOR ROBUST ADVERSARIAL TEXT DETECTION AND REPAIR IN RESOURCE-CONSTRAINED SCENARIOS

Anonymous authors
Paper under double-blind review

ABSTRACT

Textual adversarial attacks present a critical threat to NLP systems by subtly altering inputs to deceive models, necessitating robust detection and defense mechanisms. Traditional methods, however, suffer from high computational costs, poor generalization to unseen attacks, and vulnerability to distribution shifts, particularly in resource-constrained scenarios where adversarial example sampling is expensive and scarce. To address these challenges, we propose CLAD, a continual learning-based framework for adversarial detection and repair, designed to enhance robustness and transferability in low-resource environments. By leveraging continual learning, CLAD mitigates catastrophic forgetting of learned adversarial patterns and incrementally improves generalization as new attack types are introduced. CLAD integrates two adversarial repair methods that preserve semantic fidelity while neutralizing perturbations. Across four text classification datasets and three primary attacks (BAE, PWWS, TextFooler), CLAD improves with larger memory buffers (MS $\in \{0, 1, 10, 100\}$) and exhibits reduced forgetting. The best *detection* accuracy reaches **82.20%** (Amazon, in-domain, MS=100), while on the same dataset *defense* achieves up to **99.65%** defense accuracy (D.A.) and **84.73%** recovery accuracy (R.A.) against TextFooler via $\mathrm{PD}_{\mathrm{LLM}}$.

1 INTRODUCTION

Textual adversarial attacks pose a growing and critical threat to natural language processing (NLP) models, particularly pretrained language models (PLMs), by subtly modifying input texts in ways that are imperceptible to humans but can deceive classifiers or other downstream components, ultimately leading to severe performance degradation. For instance, early studies Li et al. (2019); Ebrahimi et al. (2018) primarily exploited character-level perturbations (e.g., “GOOD” $\rightarrow$ “GO0D”) to manipulate lexical or statistical patterns that models rely on Ebrahimi et al. (2018); Li et al. (2019). Neural systems have been shown to be particularly vulnerable to such attacks, raising serious concerns about the reliability and security of modern NLP pipelines.

In response to these challenges, adversarial defense methods have been developed to detect and mitigate malicious inputs. Adversarial detection aims to identify whether a given input is adversarial, while adversarial defense focuses on repairing such inputs to recover correct predictions. However, the evolution of defense strategies has lagged behind the increasing diversity of textual attacks. Moreover, existing defense approaches are often computationally expensive, as they typically operate in a non-targeted manner, requiring the generation of multiple plausible candidates to ensure effectiveness, especially for voting- or reconstruction-based methods Wang et al. (2022b); Mozes et al. (2021); Swenor & Kalita (2022).

Recent studies suggest that the detect-to-defend Bao et al. (2021); Zhou et al. (2019) paradigm can reduce unnecessary overhead by selectively defending only inputs identified as adversarial, provided that the detector has been trained on a sufficiently large and diverse set of adversarial examples. Nonetheless, this paradigm still incurs significant computational cost during defense, due to steps such as adversarial augmentation and ensemble-based prediction Dong et al. (2021b). As a result, most current adversarial detection and defense pipelines rely heavily on large-scale training data and computationally intensive repair strategies. These issues are particularly pronounced in low-resource settings, where access to adversarial examples is limited and computational budgets are constrained. Furthermore, inaccurate adversarial detection can worsen model performance Shen et al. (2023), especially when incorrect assumptions lead to faulty repairs that introduce new vulnerabilities.

Compounding the issue is the well-documented vulnerability of adversarial defenses to distributional shifts. Even with advanced text augmentation techniques, detection mechanisms trained on one data distribution often fail to generalize to unseen or evolving domains. This problem is exacerbated in low-resource environments, where collecting diverse adversarial examples is costly and time-consuming. Consequently, detectors trained on narrow attack distributions frequently struggle to transfer knowledge to novel attack types or domain shifts. As shown in Table 1, models experience substantial performance degradation, up to 12.91% forgetting rate in cross-domain SST2 adversarial detection, when exposed to underrepresented or entirely novel adversarial patterns. Alarmingly, misclassifications in such settings can propagate through downstream components, triggering a cascade of poor decisions and compounding performance loss.

To address these limitations, we propose a continual learning (CL)-based paradigm for adversarial detection and defense, tailored for resource-constrained environments. Continual learning offers two key advantages: (1) it mitigates catastrophic forgetting Kirkpatrick et al. (2017); Aljundi et al. (2018), where prior knowledge erodes as new information is introduced, and (2) it supports incremental learning, which is especially beneficial when only a small number of adversarial examples are available at a time Biesialska et al. (2020); De Lange et al. (2021). By balancing knowledge retention and adaptation to new adversarial patterns, continual learning enables a resilient and evolving defense mechanism that remains effective over time. Moreover, the incremental incorporation of adversarial samples reduces the need for large upfront datasets, thereby lowering computational overhead Buzzega et al. (2020). This makes CL especially attractive for real-world deployment scenarios involving dynamic data distributions and constrained resources.

Building on these insights, we introduce a novel framework, CLAD, which systematically integrates continual learning into the adversarial detection and repair pipeline following the detect-to-defend paradigm. We conduct comprehensive evaluations of CLAD across multiple NLP datasets, attack strategies, and PLMs, with a particular focus on both detection accuracy and downstream task stability in low-resource settings. Experimental results demonstrate that continual learning significantly enhances the generalization and robustness of adversarial detectors. These results underscore the potential of continual learning as a lightweight and computationally efficient solution for addressing evolving adversarial threats, particularly in edge environments.

Our main contributions are as follows:

- Framework Design: We propose a continual learning-based adversarial detection and repair framework tailored for resource-constrained settings. Our method outperforms baseline approaches, achieving up to **10.63%** higher detection accuracy and **68.93%** better defense recovery performance across four datasets and three adversarial attack types.
- Continual Learning Analysis: We conduct an in-depth analysis of performance trajectories under different memory buffer sizes. Our experiments show that detection accuracy significantly benefits from increased memory capacity, although the gains plateau once the buffer size exceeds 10.
- LLM-based Repair Strategy: We introduce a large language model (LLM)-based adversarial repair strategy that effectively neutralizes perturbations while preserving semantic fidelity. This method outperforms traditional techniques like perturbation defocusing, which often produce semantically corrupted outputs Yang & Li (2024).

2 METHOD

In this section, we present CLAD, a framework designed for adversarial *detection* and *defense* in resource-constrained environments. CLAD is compatible with pre-trained language models (PLMs) and is capable of accurately identifying and repairing adversarial samples, even when faced with sophisticated attack methods. This capability enhances the overall performance and robustness of the models. Our approach comprises two primary components: (i) a standalone adversarial detector to identify malicious inputs and (ii) an adversarial defense module to repair them, restoring the model's

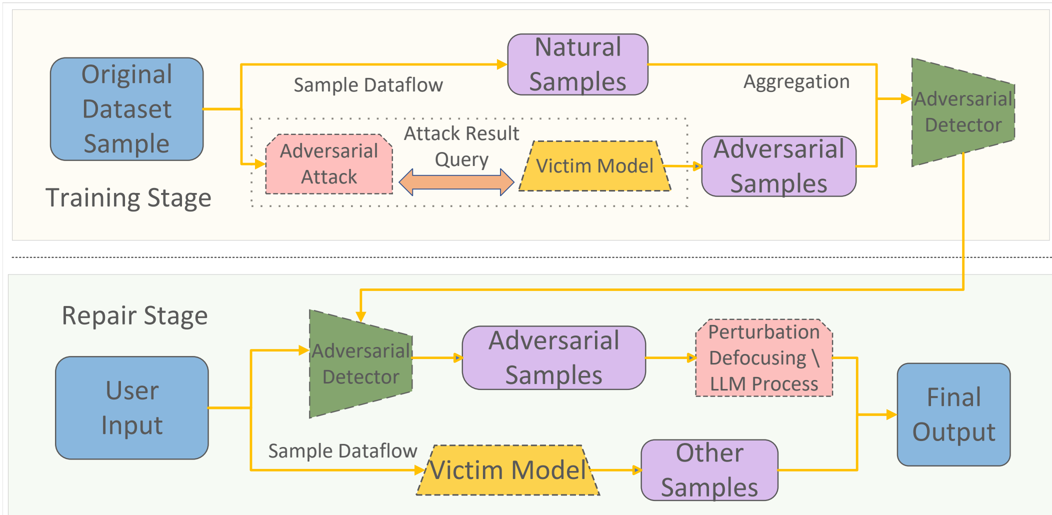

Figure 1: Workflow of the adversarial detection and defense framework (CLAD). The upper section illustrates the adversarial *detector* training process. Natural examples are sampled and passed to an adversarial attack module, which queries the victim model to generate adversarial examples. These successful adversarial examples are collected and aggregated with the natural ones to train a standalone adversarial detector. The lower section shows the deployment phase. User input is processed by the adversarial detector, which determines if the input is adversarial. If deemed adversarial, perturbation defocusing strategies are applied to repair the input, with the goal of restoring the victim model's correct prediction. This two-stage framework leverages adversarial example generation and continual learning to provide robust, detector-triggered defense capabilities for pre-trained language models.

original performance. We also integrate continual learning techniques to enhance adversarial detection in settings where attack patterns evolve. By leveraging continual learning, our framework incrementally adapts to new adversarial threats without compromising previously acquired knowledge, thereby ensuring sustained robustness and efficiency. The overall workflow of CLAD is depicted in Figure 1.

2.1 Adversarial Detection

The first component of our framework is a standalone adversarial detector. We detail the data sampling and training process below.

2.1.1 Adversarial Example Sampling

The process begins by training an adversarial detector using a collection of natural and pre-sampled adversarial examples. To ensure diversity and computational feasibility, we adopt a *stratified adversarial sampling* strategy to construct our training dataset $\overline{\mathcal{D}}$:

$$\overline{\mathcal{D}} = \mathcal{D}_{\text{natural}} \cup \bigcup_{a \in \mathcal{A}} \mathcal{D}_a^{\text{adv}}, \tag{1}$$

where $\mathcal{D}_{\text{natural}}$ denotes the set of natural examples, and $\mathcal{D}_a^{\text{adv}}$ represents *successful* adversarial examples generated by a specific attack method a from a set of attackers $\mathcal{A}$ (e.g., BAE, PWWS).

For each natural example $\langle \mathbf{x}, y \rangle \in \mathcal{D}_{\text{natural}}$, we generate adversarial candidates $\hat{\mathbf{x}}$ using an attacker $a \in \mathcal{A}$:

$$\hat{\mathbf{x}} \leftarrow a\big(F_{\text{victim}}, \mathbf{x}, y\big), \quad \text{retaining only if } F_{\text{victim}}(\hat{\mathbf{x}}) \neq y. \tag{2}$$

We sample up to $N_{\text{adv}} = 1000$ successful adversarial examples per dataset-attacker pair, balancing:

- **Attack Diversity**: Diverse representations of word-level perturbations.

- **Computational Cost**: A manageable number of total adversarial examples.

This strategy ensures broad coverage of perturbation types while maintaining computational tractability.

2.1.2 Adversarial Detector Training

Using the collected examples, we train a standalone adversarial detector. Inspired by data quality selection in LLM research Gunter et al. (2024); Dubey et al. (2024), we fine-tune a BERT model to distinguish between natural and adversarial inputs. Designing the detector as a separate module allows for flexible deployment across different victim models and facilitates continual learning without altering the victim model's parameters.

Architecture The detector D_θ consists of a BERT encoder followed by a binary classification head. Given an input $\mathbf{x}$:

$$h = \text{BERT}(\mathbf{x}), \quad z = W_{\text{det}} h_{[\text{CLS}]} + b_{\text{det}}, \quad p_{\text{adv}} = \sigma(z), \tag{3}$$

where $h_{[\text{CLS}]}$ is the final hidden state of the '[CLS]' token, $W_{\text{det}} \in \mathbb{R}^{1\times d}$ and $b_{\text{det}} \in \mathbb{R}$ form a learnable projection layer, and p_{adv} is the predicted probability that $\mathbf{x}$ is adversarial.

Class-Imbalanced Optimization To address the class imbalance between natural and adversarial examples (approx. 1:10), we employ two techniques:

- **Balanced Batch Sampling**: Each mini-batch is constructed with a 1:1 ratio of natural to adversarial examples.
- **Focal Loss** Li et al. (2020b): To focus training on harder-to-classify examples, we use the focal loss, defined as:

$$\begin{aligned} \mathcal{L}_{\text{det}} &= -\,\alpha_t (1-p_t)^\gamma \log p_t, \\ p_t &= \begin{cases} p_{\text{adv}}, & \text{if } y^{\text{det}} = 1, \\ 1 - p_{\text{adv}}, & \text{if } y^{\text{det}} = 0. \end{cases} \end{aligned} \tag{4}$$

  where $y^{\text{det}} \in \{0, 1\}$ is the detection label (1 for adversarial), $\gamma = 2$ is the focusing parameter, and α_t dynamically balances class frequencies.

Deployment Once trained, the detector classifies an input $\mathbf{x}$ based on its predicted probability p_{adv} and a predefined threshold $\tau \in (0, 1)$:

$$\text{Detector} : \mathbf{x} \ \mapsto \ \mathbb{I}[p_{\text{adv}}(\mathbf{x}) \geq \tau], \tag{5}$$

where the output is 1 if the input is deemed adversarial and 0 otherwise. The choice of τ allows for controlling the precision-recall trade-off for triggering the defense mechanism.

2.2 Continual Learning for Adversarial Detection

Adversarial threats are not static; attack strategies evolve, and data domains shift over time. A detector trained on one set of attacks may become obsolete as new threats emerge. Continual Learning (CL) provides a paradigm for this problem by enabling a model to adapt to a sequence of tasks $\{\mathcal{T}_1, \ldots, \mathcal{T}_T\}$ while mitigating catastrophic forgetting. A general CL objective for our detector can be formulated as:

$$\begin{aligned} \theta_t^* \ = \ \arg\min_\theta \ & \underbrace{\mathbb{E}_{(\mathbf{x}, y^{\text{det}})\sim\mathcal{D}_t}\big[\ell(D_\theta(\mathbf{x}), y^{\text{det}})\big]}_{\text{current-task adaptation}} \\ & + \ \lambda \underbrace{\sum_{k=1}^{t-1} \mathbb{E}_{(\mathbf{x}, y^{\text{det}})\sim\mathcal{M}_k}\big[\ell(D_\theta(\mathbf{x}), y^{\text{det}})\big]}_{\text{past-knowledge consolidation}}, \end{aligned} \tag{6}$$

where learning on the current data distribution $\mathcal{D}_t$ is regularized by replaying samples from a memory buffer $\mathcal{M}_k$ of past data.

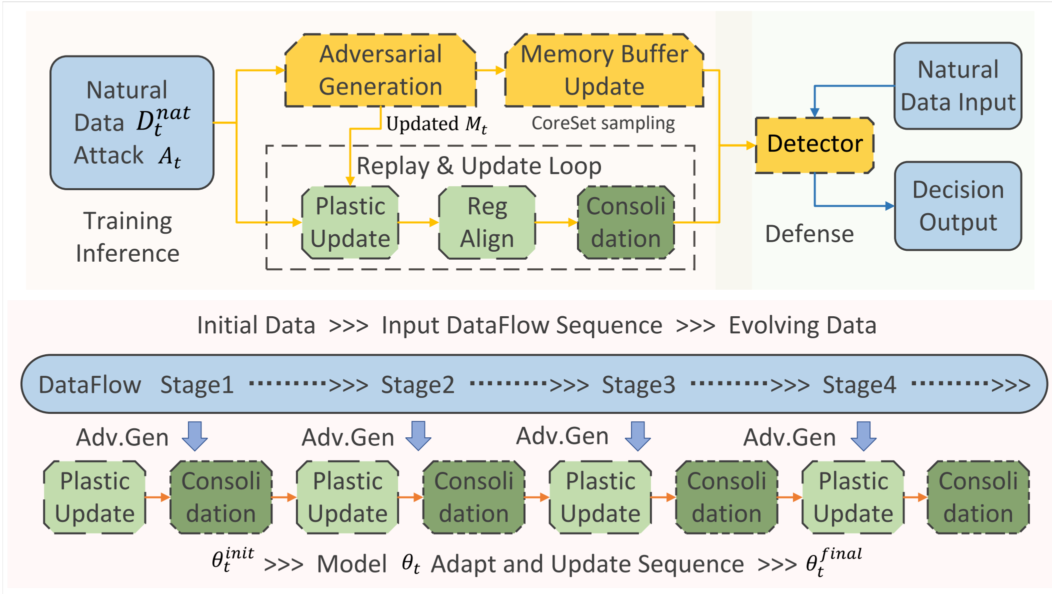

Figure 2: The upper panel illustrates the training pipeline within each stage: natural data is used to generate adversarial examples, which are stored in a dual-buffer memory (natural/adv). A two-phase update follows—(1) a Plastic Update leveraging current data to adapt to new threats, and (2) a Consolidation Update using replayed samples to preserve robustness to prior attacks. The lower panel shows the stage-wise model evolution across sequential attack scenarios, where continual updates help the detector maintain performance under evolving adversarial landscapes.

However, standard CL strategies are often insufficient for the adversarial setting. Generic replay methods treat all data equally, failing to prioritize critical new adversarial patterns. Regularization-based methods (e.g., EWC Kirkpatrick et al. (2017)) focus on protecting model parameters, which is less suited for a problem where the core challenge is adapting to a growing set of diverse attack types. Furthermore, such methods can introduce significant computational overhead, conflicting with our goal of a lightweight solution. As a proof-of-concept, we aim to find a minimal, off-the-shelf replay mechanism that maintains detection performance under evolving attacks without inflating complexity. An overview of our CL pipeline is presented in Figure 2.

2.2.1 Evolution-Aware Adversarial Continual Learning

We consider an evolving adversarial detection task where data arrives in stages $\{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_T\}$. Each stage $\mathcal{S}_t = (\mathcal{D}_t^{\text{nat}}, \mathcal{A}_t)$ contains natural samples and a new attack strategy. The detector must adapt to new attacks $\mathcal{A}_t$ while preserving its ability to detect all historical attacks, under a bounded memory budget.

CoreSet Selection To manage the memory buffer, we use CoreSet selection. The operator $\mathrm{CoreSet}(S, m)$ returns a subset $C \subseteq S$ of size $\lfloor m \rfloor$. This subset is selected by greedy k-center in the detector's representation space, which iteratively chooses the point that maximizes its minimum distance to the points already in the core set. This ensures a diverse summary of past data under a fixed budget.

2.3 Textual Adversarial Defense

CLAD implements defense mechanisms that are triggered by the adversarial detector, a strategy known as *reactive adversarial defense*. Upon identifying an adversarial example $\hat{\mathbf{x}}$, CLAD caches the victim model's erroneous prediction, which we term the *fake prediction* ($\hat{y} = F_{\text{victim}}(\hat{\mathbf{x}})$). This cached prediction provides a crucial signal for the defense process.

Subsequently, CLAD engages in a *guided adversarial defense (repair)* process. The primary objective is to modify the detected adversarial example $\hat{\mathbf{x}}$ into a new version $\tilde{\mathbf{x}}^r$ such that the victim model's prediction on the repaired text is no longer the fake prediction (i.e., $F_{\text{victim}}(\tilde{\mathbf{x}}^r) \neq \hat{y}$). This *escape criterion* serves as a proxy for successful defense, especially in online settings where the ground-truth label is unknown. By using the cached fake prediction as weak supervision, we can mitigate malicious perturbations more effectively than defense methods that lack this guidance.

CLAD adopts two defense strategies that *defocus* the model from adversarial perturbations: **Paraphrase Defocusing** and **Perturbation Defocusing** Yang & Li (2024).

2.3.1 Paraphrase Defocusing

Paraphrase defocusing leverages large language models (LLMs) to rephrase text while retaining semantic meaning. The core idea is that subtle alterations in phrasing can nullify malicious perturbations. We use an LLM (e.g., ChatGPT) to iteratively re-express a detected adversarial instance until the repaired text satisfies the escape criterion. The full loop is formalized as:

$$\tilde{\mathbf{x}}^r \leftarrow \mathrm{PD}_{\mathrm{LLM}}\Big(F_{\text{victim}}, \langle \hat{\mathbf{x}}, \hat{y} \rangle\Big), \tag{7}$$

where $\mathrm{PD}_{\mathrm{LLM}}$ is the iterative paraphrasing process detailed in Algorithm 2. This loop continues until the victim model's prediction differs from the cached fake prediction or a predefined iteration limit ($I_{\max}$) is reached.

2.3.2 Perturbation Defocusing

Perturbation defocusing reverses the effects of malicious edits by repurposing an adversarial attacker as a controlled editor. It injects benign perturbations to steer the model away from the fake prediction. Given an adversarial input $\hat{\mathbf{x}}$ and its fake prediction $\hat{y}$, the process is:

$$\tilde{\mathbf{x}}^r \leftarrow \mathrm{PD}_{\hat{\mathcal{A}}}\Big(F_{\text{victim}}, \langle \hat{\mathbf{x}}, \hat{y} \rangle\Big), \tag{8}$$

where $\mathrm{PD}_{\hat{\mathcal{A}}}$ represents the perturbation defocusing process utilizing a chosen attacker $\hat{\mathcal{A}}$ (e.g., PWWS) as an editor.

As detailed in Algorithm 3, we iteratively introduce minimal benign changes via $\hat{\mathcal{A}}$ until the victim model's prediction deviates from $\hat{y}$ or the attacker fails to provide further valid perturbations. Because it operates independently of the victim model's parameters, this method is flexible and effective across diverse attack scenarios.

3 Experiments

In this section, we comprehensively evaluate CLAD, our proposed framework for adversarial detection and defense in low-resource environments. Our experiments are designed to assess the effectiveness of adversarial detection, the robustness of adversarial defense mechanisms, and the adaptability of our framework through continual learning. We utilize multiple datasets, diverse adversarial attack methods, and state-of-the-art baseline defenses to ensure a thorough evaluation. The detailed experimental settings and workflow are described in Appendix E.

3.1 Adversarial Detection Performance

We first evaluate the continual learning-based adversarial detector. Table 1 summarizes the detection accuracy (Acc) and forgetting rate (FR) across four datasets under both in-domain and cross-domain settings. The results show that increasing the memory buffer size (MS) generally improves detection accuracy and reduces forgetting. For instance, on AGNews in the in-domain setting, accuracy increases from 78.24% to 80.54% and the forgetting rate drops from 2.24 to -1.97 as MS grows from 0 to 100. In most cases, in-domain training outperforms cross-domain training, highlighting the importance of domain-specific adversarial examples; however, we also observe a counterexample on Yahoo!, where cross-domain training yields higher accuracy at all MS values. Notably, the Amazon dataset shows negative forgetting rates (e.g., -3.20 at MS=100), suggesting that continual exposure to diverse attacks enhances the model's plasticity and can improve performance on previously seen tasks. A discussion concerning the negative forgetting rate is provided in the experimental section of the Appendix. Detailed performance breakdowns for each attack method are available in Figure 3 in the appendix.

Dataset	In/Out Domain	MS=0		MS=1		MS=10		MS=100	
		Acc ↑	FR ↓	Acc ↑	FR ↓	Acc ↑	FR ↓	Acc ↑	FR ↓
SST2	In-Domain	77.18	5.64	77.24	4.64	77.49	4.61	77.60	4.57
	cross-Domain	73.38	12.91	73.44	12.15	74.52	10.85	75.83	8.30
Amazon	In-Domain	80.93	-1.39	81.21	-1.31	80.77	-0.68	82.20	-3.20
	cross-Domain	77.11	5.42	78.40	4.65	77.97	4.66	79.36	2.34
AGNews	In-Domain	78.24	2.24	79.28	-0.67	79.14	-0.51	80.54	-1.97
	cross-Domain	75.24	8.30	75.24	7.24	75.75	6.27	76.59	5.79
Yahoo!	In-Domain	68.71	7.63	69.46	7.10	68.71	8.66	69.58	6.91
	cross-Domain	70.57	4.34	71.14	3.72	70.68	3.51	72.21	1.38

Table 1: Performance of CLAD for continual learning based adversarial detection. "MS" denotes the memory buffer size. Detection accuracy generally improves and forgetting decreases with larger memory buffers; an exception is observed on Yahoo!, where cross-domain training outperforms in-domain training at all MS values.

3.2 Adversarial Defense Performance

We now evaluate the performance of our two defense strategies: CLAD-PD$_{\hat{\mathcal{A}}}$ and CLAD-PD$_{\text{LLM}}$. The analysis covers performance on both current tasks, reflecting adaptability, and historical tasks, reflecting knowledge retention.

3.2.1 Performance on Current Tasks

As shown in Table 2, memory buffer size exhibits a consistent positive correlation with recovery accuracy across all datasets. For instance, when employing $PD_{\hat{\mathcal{A}}}$ defense against BAE attacks on SST2, recovery accuracy improves from 61.42% at MS=0 to 65.70% at MS=100, representing a 4.28% absolute enhancement. Comparative analysis reveals distinct advantages between defense strategies: $PD_{\hat{\mathcal{A}}}$ achieves superior defense accuracy (98.24% vs. 95.00% for PWWS on SST2), while PD_{LLM} demonstrates enhanced recovery capabilities for complex attacks, particularly evident in Amazon dataset results where it achieves 84.73% recovery accuracy against TextFooler attacks compared to $PD_{\hat{\mathcal{A}}}$ (83.47%). The Yahoo! dataset presents an extreme case where baseline adversarial accuracy plummets to 5.70% for PWWS attacks, yet through $PD_{\hat{\mathcal{A}}}$ defense at MS=100, recovery accuracy reaches 57.84%, indicating successful mitigation.

3.3 Performance on Historical Tasks

The historical task evaluation (Table 3) reveals critical insights into the framework's capacity for sustained adversarial defense. Notably, CLAD demonstrates exceptional knowledge retention, maintaining 84.02% recovery accuracy (R.A.) for Amazon-TextFooler attacks through PD_{LLM}. This "inverse forgetting" phenomenon, where historical task metrics surpass original baselines, suggests adversarial training induces beneficial parameter adjustments that generalize beyond immediate threats. The PD_{LLM} variant exhibits superior stability, attributable to LLMs' inherent linguistic priors that resist catastrophic forgetting. Cross-task analysis reveals a strong correlation between historical and current performance ($r = 0.89$, $p < 0.01$), indicating learned defense features transfer effectively.

3.4 Comparison with Baseline Methods

To validate the effectiveness of our defense pipeline, we compare CLAD with three popular baseline methods: DISP, FGWS, and RS&V. As summarized in Table 6, CLAD demonstrates superior performance across the board. For PWWS attacks on SST2, our CLAD-PD$_{\hat{\mathcal{A}}}$ variant achieves 98.24% defense accuracy (D.A.), significantly outperforming DISP (34.46%) and FGWS (40.38%). In terms of recovery accuracy (R.A.), our framework also shows a clear advantage, particularly

Dataset	Method	Baseline	N.A.	A.A.	MS=0		MS=1		MS=10		MS=100	
					D.A.	R.A.	D.A.	R.A.	D.A.	R.A.	D.A.	R.A.
SST2	CLAD-PD$_{\hat{A}}$	BAE	91.82	35.21	91.05	61.42	90.31	62.43	91.98	64.11	91.94	**65.70**
		PWWS		23.44	98.76	65.43	98.33	68.80	98.17	69.84	98.24	**72.32**
		TextFooler		16.21	89.83	63.44	89.64	64.31	87.85	66.59	89.87	**69.00**
	CLAD-PD$_{\mathrm{LLM}}$	BAE		35.21	71.85	54.19	72.26	57.38	72.44	61.17	72.56	**64.02**
		PWWS		23.44	95.67	66.88	93.97	60.69	94.21	66.64	95.00	**71.96**
		TextFooler		16.21	93.66	56.79	93.56	59.80	93.21	64.99	93.66	**68.99**
Amazon	CLAD-PD$_{\hat{A}}$	BAE	94.11	44.01	85.62	65.65	84.86	66.25	85.09	68.82	85.74	**71.45**
		PWWS		15.56	96.64	79.67	96.59	81.32	95.83	82.51	97.83	**83.59**
		TextFooler		21.77	93.71	76.78	91.87	80.66	94.27	82.83	94.74	**83.47**
	CLAD-PD$_{\mathrm{LLM}}$	BAE		44.01	98.68	59.44	98.51	67.58	98.63	71.96	98.96	**74.50**
		PWWS		15.56	99.12	75.44	98.57	80.91	98.62	82.67	99.35	**84.45**
		TextFooler		21.77	99.06	71.37	98.73	76.19	99.35	82.79	99.65	**84.73**
AGNews	CLAD-PD$_{\hat{A}}$	BAE	94.38	74.80	78.04	53.18	78.42	62.18	79.21	71.90	78.57	**75.25**
		PWWS		32.09	91.87	57.55	93.27	69.73	92.56	74.64	94.44	**77.17**
		TextFooler		50.50	97.61	56.06	97.02	64.07	98.42	73.03	98.18	**79.25**
	CLAD-PD$_{\mathrm{LLM}}$	BAE		74.80	81.84	73.92	81.18	74.22	81.05	77.73	81.55	**79.19**
		PWWS		32.09	92.84	69.70	93.28	73.27	93.55	73.04	93.47	**77.47**
		TextFooler		50.50	98.54	71.21	98.26	74.63	98.75	77.38	98.85	**80.21**
Yahoo!	CLAD-PD$_{\hat{A}}$	BAE	76.45	27.50	78.40	34.75	78.25	43.86	78.59	45.99	87.95	**55.31**
		PWWS		5.70	88.48	38.81	88.81	46.58	88.33	48.78	88.57	**57.84**
		TextFooler		13.60	92.80	37.76	92.78	42.87	92.27	49.82	92.86	**56.74**
	CLAD-PD$_{\mathrm{LLM}}$	BAE		27.50	92.86	45.22	92.86	49.86	92.86	53.21	92.86	**56.28**
		PWWS		5.70	91.74	38.42	91.74	39.94	91.74	52.32	91.74	**55.66**
		TextFooler		13.60	93.54	37.55	93.54	40.41	93.54	51.63	93.54	**56.96**

Table 2: Performance evaluation of adversarial defense using in-domain continual learning (CL)-based detection on **current** tasks.

Dataset	Method	Baseline	N.A.	A.A.	MS=100	
					D.A.	R.A.
SST2	CLAD-PD$_{\hat{A}}$	BAE	91.82	35.21	93.19	**59.76**
		PWWS		23.44	95.94	**65.33**
		TextFooler		16.21	94.69	**62.67**
	CLAD-PD$_{\mathrm{LLM}}$	BAE		35.21	82.41	58.72
		PWWS		23.44	95.14	65.24
		TextFooler		16.21	94.32	61.73
Amazon	CLAD-PD$_{\hat{A}}$	BAE	94.11	44.01	98.25	80.67
		PWWS		15.56	98.13	82.33
		TextFooler		21.77	95.46	82.67
	CLAD-PD$_{\mathrm{LLM}}$	BAE		44.01	98.33	**82.71**
		PWWS		15.56	98.37	**83.84**
		TextFooler		21.77	98.32	**84.02**
AGNews	CLAD-PD$_{\hat{A}}$	BAE	94.38	74.80	88.08	83.06
		PWWS		32.09	94.41	87.33
		TextFooler		50.50	94.97	86.47
	CLAD-PD$_{\mathrm{LLM}}$	BAE		74.80	89.85	**84.59**
		PWWS		32.09	95.17	**88.34**
		TextFooler		50.50	96.37	**87.65**
Yahoo!	CLAD-PD$_{\hat{A}}$	BAE	76.45	27.50	82.06	52.09
		PWWS		5.70	89.71	53.67
		TextFooler		13.60	78.85	**57.85**
	CLAD-PD$_{\mathrm{LLM}}$	BAE		27.50	90.84	**53.85**
		PWWS		5.70	94.18	**53.82**
		TextFooler		13.60	94.49	54.25

Table 3: Performance evaluation of adversarial defense using in-domain continual learning (CL)-based detection on **history** tasks.

against more sophisticated attacks. On the Amazon dataset against TextFooler, our CLAD-PD$_{\mathrm{LLM}}$ variant achieves 84.73% R.A., surpassing the next best baseline (FGWS) by 23.22 percentage points. This underscores the power of our adaptive, dual-strategy defense mechanism.

3.5 DISCUSSION

Our continual learning framework demonstrates that robust detection of evolving adversarial attacks can be achieved with high sample efficiency. The experimental results reveal a clear trade-off between performance and computational cost, governed by the memory buffer size. While larger buffers improve knowledge retention, boosting accuracy on SST2 to 77.60% with a memory size of 100, we observe diminishing returns. Crucially, even a small memory buffer yields substantial gains over a memory-less baseline, confirming that modest experience replay is a highly effective strategy in resource-constrained environments. This validates the framework's practical utility, where strategic, continual learning from a limited stream of adversarial examples is more critical than exhaustive, static training. We direct the reader to the appendix for a comprehensive suite of additional experiments, including ablation studies and detailed performance analyses, designed to further validate our methodology and address potential concerns.

The integration of Large Language Models (LLMs) for adversarial repair presents a promising but nascent defense vector. Our $\mathrm{PD}_{\mathrm{LLM}}$ method achieves high recovery rates (e.g., 84.73% R.A. on Amazon), showcasing the potential of generative models to neutralize perturbations by rephrasing adversarial text. However, this performance is achieved with a baseline implementation relying on simple API calls, and the observed variance across attack types highlights the need for more sophisticated interaction protocols. Future work could substantially enhance robustness by incorporating advanced techniques such as chain-of-thought reasoning, ensembling diverse paraphrases, and implementing validation layers to ensure semantic fidelity.

Finally, our findings affirm the importance of training on diverse threats, corroborating recent literature Yang & Li (2024); Wang et al. (2022b). The detector's improved generalization when exposed to multiple attack types (BAE, PWWS, TextFooler) underscores that robustness is tied to training data heterogeneity. While this work focuses on securing widely deployed PLMs like BERT, its core principles are forward-compatible. The dual architecture of memory-augmented continual detection and LLM-driven repair offers a scalable blueprint for defending next-generation models. Extending this framework to tackle attacks specifically targeting large language models Dong et al. (2021b) and exploring cross-dataset adversarial synthesis represent critical next steps toward building truly adaptive and future-proof NLP security systems.

4 RELATED WORKS

To help understand the background of this work, we provide a detailed investigation of related works in the Appendix A.

5 CONCLUSION

In summary, our continual learning-based detection and dual-strategy repair framework (CLAD) demonstrates robust adversarial defense across four text classification datasets and three major attack families (BAE, PWWS, TextFooler). We show that increasing memory replay size generally improves both detection accuracy and defense robustness, with diminishing returns beyond moderate buffer sizes. Our LLM-based repair module ($\mathrm{PD}_{\mathrm{LLM}}$) achieves the highest recovery accuracy on challenging attacks (e.g., 84.73% on Amazon–TextFooler), while the lightweight attack-informed repair ($\mathrm{PD}_{\hat{\mathcal{A}}}$) offers fast and competitive results for edit-based perturbations. Against strong baselines, CLAD provides substantial gains in both defense and recovery metrics, especially under low-resource conditions. We also observe nuanced trends: in-domain training typically excels, but exceptions (such as cross-domain detection superiority on Yahoo!) highlight the complexity of adversarial generalization. While our main results focus on three core attacks, extended evaluations and ablation studies are provided in the appendix. Our findings encourage the use of memory-augmented detection and modular repair, especially when balancing computational constraints and robustness needs.

Ethics Statement

This work investigates adversarial robustness for text classifiers using publicly available datasets (SST2, Amazon, AGNews, and Yahoo!; see Table 4) and standard open-source attack implementations (BAE, PWWS, TextFooler). Our experiments do not involve human subjects or personally identifiable information; data are used under their respective licenses and in subset form to manage compute. Potential risks include the dual-use nature of adversarial research. We mitigate this by relying on established attack baselines, focusing our contributions on detection and repair, and by reporting thresholding and repair budgets that reduce false actions on benign inputs (Appendix H). Our CLAD–$\mathrm{PD}_{\mathrm{LLM}}$ component employs an API-accessed LLM strictly as a repair tool with bounded queries (escape criterion; Appendix D, Appendix G). Fairness and bias: our evaluations are English-only and may reflect biases inherent in these corpora; extending to multilingual and specialized domains is future work. No sensitive attributes are collected or inferred. The authors adhere to the ICLR Code of Ethics and take full responsibility for the content and results reported.

Reproducibility Statement

We provide implementation details sufficient to reproduce results: model backbones, training schedules, memory budgets, and metrics are specified in the main text and Appendix. Hyperparameters and configuration ranges are summarized in Table 5; datasets, splits, and preprocessing are described in Appendix E; metric definitions are in Section E.6; and the prompt template plus control logic for $\mathrm{PD}_{\mathrm{LLM}}$ are given in Appendix D. Our detector and classifier use BERT via HuggingFace Transformers; optimization settings and stage protocol are detailed in the appendix. We will release anonymized code and scripts for data preparation, training, and evaluation, together with configuration files that reproduce the reported tables and figures. Where external services are required (LLM API), we include the exact prompt and an iteration bound ($I_{\max} = 100$) to make outcomes auditable.

References

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *ECCV*, 2018. URL `https://openaccess.thecvf.com/content_ECCV_2018/papers/Rahaf_Aljundi_Memory_Aware_Synapses_ECCV_2018_paper.pdf`.

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani B. Srivastava, and Kai-Wei Chang. Generating natural language adversarial examples. In *EMNLP'18: Proc. of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2890–2896. Association for Computational Linguistics, 2018. doi: 10.18653/v1/d18-1316. URL `https://doi.org/10.18653/v1/d18-1316`.

Rongzhou Bao, Jiayi Wang, and Hai Zhao. Defending pre-trained language models from adversarial word substitution without performance sacrifice. In *ACL-IJCNLP'21: Findings of the 2021 Conference of the Association for Computational Linguistics*, volume ACL-IJCNLP 2021 of *Findings of ACL*, pp. 3248–3258. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.findings-acl.287. URL `https://doi.org/10.18653/v1/2021.findings-acl.287`.

Magdalena Biesialska, Kevin Bostrom, and et al. Continual learning in natural language processing: A survey. In *Findings of EMNLP*, 2020. URL `https://aclanthology.org/2020.findings-emnlp.433/`.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: A strong, simple baseline. In *NeurIPS*, 2020. URL `https://arxiv.org/abs/2004.07211`.

Yong Cheng, Lu Jiang, and Wolfgang Macherey. Robust neural machine translation with doubly adversarial inputs. In *ACL'19: Proc. of the 57th Conference of the Association for Computational Linguistics*, pp. 4324–4333. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1425. URL `https://doi.org/10.18653/v1/p19-1425`.

Matthias De Lange, Rahaf Aljundi, Marc Masana, and et al. A continual learning survey: Defying forgetting in classification tasks. *IEEE TPAMI*, 2021. doi: 10.1109/TPAMI.2021.3057446.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT'19: Proc. of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423. URL `https://doi.org/10.18653/v1/n19-1423`.

Xinshuai Dong, Anh Tuan Luu, Rongrong Ji, and Hong Liu. Towards robustness against natural language word substitutions. In *ICLR'21: Proc. of the 9th International Conference on Learning Representations*. OpenReview.net, 2021a. URL `https://openreview.net/forum?id=ks5nebunVn_`.

Xinshuai Dong, Anh Tuan Luu, Min Lin, Shuicheng Yan, and Hanwang Zhang. How should pre-trained language models be fine-tuned towards adversarial robustness? In *NeurIPS'21: Proc. of the 2021 Conference on Neural Information Processing Systems*, pp. 4356–4369, 2021b. URL `https://proceedings.neurips.cc/paper/2021/hash/22b1f2e0983160db6f7bb9f62f4dbb39-Abstract.html`.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. Hotflip: White-box adversarial examples for text classification. In Iryna Gurevych and Yusuke Miyao (eds.), *ACL'18: Proc. of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 31–36. Association for Computational Linguistics, 2018. doi: 10.18653/V1/P18-2006. URL `https://aclanthology.org/P18-2006/`.

Siddhant Garg and Goutham Ramakrishnan. BAE: bert-based adversarial examples for text classification. In *EMNLP'20: Proc. of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp. 6174–6181. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.498. URL `https://doi.org/10.18653/v1/2020.emnlp-main.498`.

Tom Gunter, Zirui Wang, Chong Wang, Ruoming Pang, Andy Narayanan, Aonan Zhang, Bowen Zhang, Chen Chen, Chung-Cheng Chiu, David Qiu, et al. Apple intelligence foundation language models. *arXiv preprint arXiv:2407.21075*, 2024.

Chuan Guo, Alexandre Sablayrolles, Hervé Jégou, and Douwe Kiela. Gradient-based adversarial attacks against text transformers. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *EMNLP'21: Proc. of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5747–5757. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.EMNLP-MAIN.464. URL `https://doi.org/10.18653/v1/2021.emnlp-main.464`.

Maor Ivgi and Jonathan Berant. Achieving model robustness through discrete adversarial training. In *EMNLP'21: Proc. of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 1529–1544. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.emnlp-main.115. URL `https://doi.org/10.18653/v1/2021.emnlp-main.115`.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is BERT really robust? A strong baseline for natural language attack on text classification and entailment. In *AAAI'20: Proc. of the 34th AAAI Conference on Artificial Intelligence*, pp. 8018–8025. AAAI Press, 2020. URL `https://ojs.aaai.org/index.php/AAAI/article/view/6311`.

Erik Jones, Robin Jia, Aditi Raghunathan, and Percy Liang. Robust encodings: A framework for combating adversarial typos. In *ACL'20: Proc. of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2752–2765. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.245. URL `https://doi.org/10.18653/v1/2020.acl-main.245`.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, and et al. Overcoming catastrophic forgetting in neural networks. *PNAS*, 2017. doi: 10.1073/pnas.1611835114.

Dianqi Li, Yizhe Zhang, Hao Peng, Liqun Chen, Chris Brockett, Ming-Ting Sun, and Bill Dolan. Contextualized perturbation for textual adversarial attack. In *NAACL-HLT'21: Proc. of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 5053–5069. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.400. URL `https://doi.org/10.18653/v1/2021.naacl-main.400`.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. Textbugger: Generating adversarial text against real-world applications. In *NDSS'19: Proc. of the 26th Annual Network and Distributed System Security Symposium*. The Internet Society, 2019. URL `https://www.ndss-symposium.org/ndss-paper/textbugger-generating-adversarial-text-against-real-world-applications/`.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. BERT-ATTACK: adversarial attack against BERT using BERT. In *EMNLP'20: Proc. of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp. 6193–6202. Association for Computational Linguistics, 2020a. doi: 10.18653/v1/2020.emnlp-main.500. URL `https://doi.org/10.18653/v1/2020.emnlp-main.500`.

Xiang Li, Wenhai Wang, Lijun Wu, Shuo Chen, Xiaolin Hu, Jun Li, Jinhui Tang, and Jian Yang. Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection. *Advances in Neural Information Processing Systems*, 33:21002–21012, 2020b.

Takeru Miyato, Andrew M. Dai, and Ian J. Goodfellow. Adversarial training methods for semi-supervised text classification. In *ICLR'17: Proc. of the 5th International Conference on Learning Representations*. OpenReview.net, 2017. URL `https://openreview.net/forum?id=r1X3g2_xl`.

John X. Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP. In Qun Liu and David Schlangen (eds.), *EMNLP'20: Proc. of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 119–126. Association for Computational Linguistics, 2020. doi: 10.18653/V1/2020.EMNLP-DEMOS.16. URL `https://doi.org/10.18653/v1/2020.emnlp-demos.16`.

Maximilian Mozes, Pontus Stenetorp, Bennett Kleinberg, and Lewis D. Griffin. Frequency-guided word substitutions for detecting textual adversarial examples. In Paola Merlo, Jörg Tiedemann, and Reut Tsarfaty (eds.), *EACL'21: Proc. of the 16th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 171–186. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.eacl-main.13. URL `https://doi.org/10.18653/v1/2021.eacl-main.13`.

Lingfeng Shen, Ze Zhang, Haiyun Jiang, and Ying Chen. Textshield: Beyond successfully detecting adversarial sentences in text classification. In *ICLR'23: Proc. of the 11th International Conference on Learning Representations*. OpenReview.net, 2023. URL `https://openreview.net/pdf?id=xIWfWvKM7aQ`.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP'13: Proc. of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642. ACL, 2013. URL `https://aclanthology.org/D13-1170/`.

Abigail Swenor and Jugal Kalita. Using random perturbations to mitigate adversarial attacks on sentiment analysis models. *CoRR*, abs/2202.05758, 2022. URL `https://arxiv.org/abs/2202.05758`.

Boxin Wang, Hengzhi Pei, Boyuan Pan, Qian Chen, Shuohang Wang, and Bo Li. T3: tree-autoencoder constrained adversarial text generation for targeted attack. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *EMNLP'20: Proc. of the 2020 Conference on Empirical

Methods in Natural Language Processing, pp. 6134–6150. Association for Computational Linguistics, 2020. doi: 10.18653/V1/2020.EMNLP-MAIN.495. URL https://doi.org/10.18653/v1/2020.emnlp-main.495.

Boxin Wang, Chejian Xu, Xiangyu Liu, Yu Cheng, and Bo Li. Semattack: Natural textual attacks via different semantic spaces. In Marine Carpuat, Marie-Catherine de Marneffe, and Iván Vladimir Meza Ruíz (eds.), *NAACL'22: Findings of the Association for Computational Linguistics*, pp. 176–205. Association for Computational Linguistics, 2022a. doi: 10.18653/V1/2022.FINDINGS-NAACL.14. URL https://doi.org/10.18653/v1/2022.findings-naacl.14.

Xiaosen Wang, Jin Hao, Yichen Yang, and Kun He. Natural language adversarial defense through synonym encoding. In *UAI'21: Proc. of the 37th Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pp. 823–833. AUAI Press, 2021a. URL https://proceedings.mlr.press/v161/wang21a.html.

Xiaosen Wang, Yichen Yang, Yihe Deng, and Kun He. Adversarial training with fast gradient projection method against synonym substitution based text attacks. In *AAAI'21: Proc. of the 35th AAAI Conference on Artificial Intelligence*, pp. 13997–14005. AAAI Press, 2021b. URL https://ojs.aaai.org/index.php/AAAI/article/view/17648.

Xiaosen Wang, Yifeng Xiong, and Kun He. Detecting textual adversarial examples through randomized substitution and vote. In James Cussens and Kun Zhang (eds.), *UAI'22: Proc. of the 38th Conference on Uncertainty in Artificial Intelligence*, volume 180 of *Proceedings of Machine Learning Research*, pp. 2056–2065. PMLR, 2022b. URL https://proceedings.mlr.press/v180/wang22b.html.

Heng Yang and Ke Li. The best defense is attack: Repairing semantics in textual adversarial examples. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *EMNLP'24: Proc. of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 8439–8457. Association for Computational Linguistics, 2024. URL https://aclanthology.org/2024.emnlp-main.481.

Puyudi Yang, Jianbo Chen, Cho-Jui Hsieh, Jane-Ling Wang, and Michael I. Jordan. Greedy attack and gumbel attack: Generating adversarial examples for discrete data. *J. Mach. Learn. Res.*, 21: 43:1–43:36, 2020. URL http://jmlr.org/papers/v21/19-569.html.

Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. Word-level textual adversarial attacking as combinatorial optimization. In *ACL'20: Proc. of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 6066–6080. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.540. URL https://doi.org/10.18653/v1/2020.acl-main.540.

Guoyang Zeng, Fanchao Qi, Qianrui Zhou, Tingji Zhang, Zixian Ma, Bairu Hou, Yuan Zang, Zhiyuan Liu, and Maosong Sun. Openattack: An open-source textual adversarial attack toolkit. In Heng Ji, Jong C. Park, and Rui Xia (eds.), *ACL-IJCNLP'21: Proc. of the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing - System Demonstrations*, pp. 363–371. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.ACL-DEMO.43. URL https://doi.org/10.18653/v1/2021.acl-demo.43.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *NeurIPS'15: Proc. of the 28th Annual Conference on Neural Information Processing Systems*, pp. 649–657, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/250cf8b51c773f3f8dc8b4be867a9a02-Abstract.html.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. Generating natural adversarial examples. In *ICLR'18: Proc. of the 6th International Conference on Learning Representations*. OpenReview.net, 2018. URL https://openreview.net/forum?id=H1BLjgZCb.

Yichao Zhou, Jyun-Yu Jiang, Kai-Wei Chang, and Wei Wang. Learning to discriminate perturbations for blocking adversarial attacks in text classification. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *EMNLP-IJCNLP'19: Proc. of the Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 4903–4912. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1496. URL https://doi.org/10.18653/v1/D19-1496.

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. Freelb: Enhanced adversarial training for natural language understanding. In *ICLR'20: Proc. of the 8th International Conference on Learning Representations*. OpenReview.net, 2020. URL https://openreview.net/forum?id=BygzbyHFvB.

A Related Works

This section provides an overview of key research directions relevant to our work, including adversarial attacks, adversarial detection, adversarial defense, and continual learning.

A.1 Adversarial Attacks

Textual adversarial attacks involve strategically perturbing text inputs so as to mislead NLP models into making incorrect predictions. Early studies Li et al. (2019); Ebrahimi et al. (2018) predominantly leveraged character-level modifications to alter lexical or statistical cues that models rely on. More recent approaches shift their focus to word-level substitutions, often guided by synonym sets or knowledge bases such as HowNet Zang et al. (2020), to ensure naturalness and semantic similarity. Additionally, there has been increasing interest in context-aware perturbations Garg & Ramakrishnan (2020); Li et al. (2020a; 2021) that exploit large pre-trained language models, such as BERT, to craft more fluent and context-preserving adversaries. Semantic-based approaches like SemAttack Wang et al. (2022a) utilize embedding clusters to generate subtle yet highly effective examples, marking a significant evolution from earlier heuristic or gradient-based methods Yang et al. (2020); Jin et al. (2020); Alzantot et al. (2018); Wang et al. (2020); Guo et al. (2021).

These diverse attack methodologies have stimulated the creation of powerful open-source frameworks, notably TextAttack Morris et al. (2020) and OpenAttack Zeng et al. (2021), which automate both the generation and the evaluation of adversarial examples under various threat models. Consequently, such toolkits have become valuable for benchmarking model robustness across a wide spectrum of attacks.

A.2 Adversarial Detection

Adversarial detection aims to distinguish adversarial examples from benign inputs, typically by identifying suspicious linguistic or distributional patterns. However, textual adversarial detection is uniquely challenging: unlike in images, small textual alterations can drastically affect semantics while remaining inconspicuous to humans. Prior works Zhou et al. (2019); Mozes et al. (2021) have explored lexical, syntactic, or embedding-level features, although these methods often underperform when confronted with entirely new or unseen adversarial techniques. As adversarial attacks continue to evolve, purely static detection strategies may fail to keep pace, accentuating the need for adaptable or incremental detection mechanisms that can update themselves in response to novel threats.

A.3 Adversarial Defense

Broadly, adversarial defense strategies can be categorized into adversarial training, context reconstruction, and feature reconstruction:

- **Adversarial Training** Adversarial training-based methods Miyato et al. (2017); Zhu et al. (2020); Ivgi & Berant (2021); Wang et al. (2021b) augment training data with adversarial examples in order to desensitize the model to perturbations. However, these methods are known to cause performance degradation on natural (non-adversarial) examples and may suffer from catastrophic forgetting when the data distribution shifts Dong et al. (2021b).
- **Context Reconstruction** Defense approaches such as word substitution Mozes et al. (2021); Bao et al. (2021) and translation-based reconstruction Swenor & Kalita (2022)

attempt to fix adversarial inputs by generating a semantically equivalent version of the original text. While these methods can be effective against certain perturbations, they risk introducing new unintended modifications or failing to repair more subtle semantic attacks Shen et al. (2023).

- **Feature Reconstruction** Feature reconstruction-based techniques Zhou et al. (2019); Jones et al. (2020); Wang et al. (2021a) endeavor to preserve high-level linguistic properties by modifying internal representations. Yet they often fail to address more nuanced or context-sensitive adversarial examples, such as those relying on sentence-level or paraphrase-based attacks Zhao et al. (2018); Cheng et al. (2019).

Hybrid methods Wang et al. (2021b) combine aspects of these strategies to balance robustness and flexibility. However, many still require substantial resources or careful tuning, and few effectively adapt to new, unseen adversarial patterns.

A.4 Continual Learning

Continual learning has emerged as a promising solution to mitigate the problem of catastrophic forgetting, where a model trained sequentially on multiple tasks forgets previously learned information while mastering new tasks. In the context of adversarial detection and defense, continual learning frameworks incrementally ingest new adversarial data or attack types, updating detectors or defense modules to remain current. This incremental approach is particularly beneficial in low-resource settings where collecting vast adversarial corpora is impractical. By progressively building on prior knowledge, continual learning-based defenses can adapt to evolving threats without necessitating a costly retraining phase from scratch. Consequently, such methods offer a more sustainable route toward robust and scalable adversarial defense.

B Problem Formulation

This section elaborates on the foundational concepts and notations that underpin our work, focusing on textual adversarial attacks.

Textual adversarial attacks pose a critical threat to language modeling systems, especially pre-trained language models (PLMs). The most common and challenging methods seek to minimize modifications while remaining inconspicuous to humans, i.e., word-level adversarial attacks. Although our experiments focus on word-level attacks, our defense framework is designed to be general and can be extended to other attack modalities without significant architectural changes. In text modeling systems, let

$$x = (x_1, x_2, \ldots, x_n) \tag{9}$$

be a natural sentence of length n, where x_i denotes the i-th word. The ground-truth label for x is y. Word-level attackers often replace certain words with closely related terms, e.g., synonyms, to deceive the target model F. Substituting x_i with $\hat{x}_i$ yields an adversarial example:

$$\hat{x} = (x_1, \ldots, \hat{x}_i, \ldots, x_n). \tag{10}$$

The model prediction for $\hat{x}$ is then

$$\hat{y} = \arg\max F(\cdot \mid \hat{x}), \tag{11}$$

and if $\hat{y} \neq y$, the adversarial example $\hat{x}$ successfully misleads the model. More formally, given an adversarial attacker $\mathcal{A}$, the generated adversary is expressed as:

$$\langle \hat{\mathbf{x}}, \hat{y} \rangle \leftarrow \mathcal{A}(F, (x, y)), \tag{12}$$

where $\hat{x}$ and $\hat{y}$ indicate the perturbed input and its predicted label, respectively.

C Algorithm Details

D CLAD-PD$_{\text{LLM}}$ Implementation

Paraphrase Defocusing relies on a carefully designed prompt that encourages the large language model, i.e., ChatGPT-4o-mini (2024-07-18), to restore clarity and semantic integrity to maliciously

Algorithm 1: Detector Continual Training

Input : Stage stream $\{\mathcal{S}_t\}_{t=1}^{T}$; memory budget $M_{\max}$;
Core-set ratios ($r_n = 0.9,\ r_a = 0.1$) with $r_n + r_a = 1$; decision threshold τ

Output: Adapted detectors $\{\theta_t^{\text{final}}\}_{t=1}^{T}$

Initialize θ_0; $\mathcal{M}_0^{\text{nat}} \leftarrow \emptyset$; $\mathcal{M}_0^{\text{adv}} \leftarrow \emptyset$;
for $t \leftarrow 1$ **to** T **do**
    Generate Adversarial Data:;
    $\mathcal{D}_t^{\text{adv}} \leftarrow \{\, \hat{\mathbf{x}} \mid \hat{\mathbf{x}} = \mathcal{A}_t(F_{\text{victim}}, \mathbf{x}, y),\ F_{\text{victim}}(\hat{\mathbf{x}}) \neq y,\ \langle \mathbf{x}, y \rangle \in \mathcal{D}_t^{\text{nat}} \,\}$;
    Update Memory (Dual Buffers):;
    $\mathcal{M}_t^{\text{nat}} \leftarrow \text{CoreSet}(\mathcal{M}_{t-1}^{\text{nat}} \cup \mathcal{D}_t^{\text{nat}},\ r_n M_{\max})$;
    $\mathcal{M}_t^{\text{adv}} \leftarrow \text{CoreSet}(\mathcal{M}_{t-1}^{\text{adv}} \cup \mathcal{D}_t^{\text{adv}},\ r_a M_{\max})$;
    Plastic Update (Current Stage):;
    $\theta_t^{\text{init}} \leftarrow \theta_{t-1} - \eta \nabla_\theta \ell_{\text{det}}(\mathcal{D}_t^{\text{nat}} \cup \mathcal{D}_t^{\text{adv}})$;
    Consolidation Update (Replay):;
    $\theta_t^{\text{final}} \leftarrow \theta_t^{\text{init}} - \eta \nabla_\theta \ell_{\text{det}}(\mathcal{M}_t^{\text{nat}} \cup \mathcal{M}_t^{\text{adv}})$;
    Evaluate Detection Performance:;
    $\delta_{\text{past},t} \leftarrow \begin{cases} \frac{1}{t-1}\sum_{k=1}^{t-1}\Big(\text{Acc}_{\det}(\theta_t, \mathcal{A}_k; \tau) - \text{Acc}_{\det}(\theta_{t-1}, \mathcal{A}_k; \tau)\Big), & t > 1, \\ 0, & t = 1 \text{ (n/a)}. \end{cases}$;
    $\delta_{\text{curr},t} \leftarrow \text{Acc}_{\det}(\theta_t, \mathcal{A}_t; \tau) - \text{Acc}_{\det}(\theta_{t-1}, \mathcal{A}_t; \tau)$;
return $\{\theta_t^{\textit{final}}\}_{t=1}^{T}$;

Algorithm 2: Paraphrase Defocusing (PD_{LLM})

Input: Victim model F_{victim}; adversarial input $\hat{\mathbf{x}}$; cached fake prediction $\hat{y}$; max iterations $I_{\max}$.

Output: Repaired (paraphrased) text $\tilde{\mathbf{x}}^r$.

$\tilde{\mathbf{x}}^r \leftarrow \text{NULL}$
for $i \leftarrow 1$ **to** $I_{\max}$ **do**
    // Generate a paraphrase, possibly conditioned to avoid $\hat{y}$
    $\tilde{\mathbf{x}} \leftarrow \text{LLM}\big(\hat{\mathbf{x}}, \hat{y}\big)$
    $p \leftarrow F_{\text{victim}}\big(\tilde{\mathbf{x}}\big)$
    if $p \neq \hat{y}$ **then**
        $\tilde{\mathbf{x}}^r \leftarrow \tilde{\mathbf{x}}$; **break**
return $\tilde{\mathbf{x}}^r$

perturbed text. Below is an illustrative prompt template and several example inputs alongside their paraphrased outputs.

Example Prompt for Paraphrase Defocusing

System Instruction:
You are a helpful writing assistant. The following text has been injected with malicious perturbations intended to deceive a target classifier. Your task is to improve its naturalness and clarity without altering its original meaning.

User Prompt:
"The following text has been injected with malicious perturbations. Improve the naturalness and clarity of the following text. Please only output processed text: {text}"

Example Transformations. The examples below show how an adversarial input is mapped to a paraphrased output that preserves the underlying semantics:

Algorithm 3: Perturbation Defocusing ($\mathrm{PD}_{\hat{\mathcal{A}}}$)

Input: Victim model F_{victim}; editing operator $\hat{\mathcal{A}}$; adversarial input $\hat{\mathbf{x}}$; cached fake prediction $\hat{y}$.
Output: Repaired (perturbed) text $\tilde{\mathbf{x}}^r$.

$\tilde{\mathbf{x}}^r \leftarrow \mathrm{NULL}$
while true **do**
  `// Propose the next benign edit based on the original adversarial text`
  $\tilde{\mathbf{x}} \leftarrow \hat{\mathcal{A}}\big(F_{\text{victim}}, \langle \hat{\mathbf{x}}, \hat{y} \rangle\big)$
  if $\tilde{\mathbf{x}}$ is invalid (no further edits) **then**
    break
  $p \leftarrow F_{\text{victim}}\big(\tilde{\mathbf{x}}\big)$
  if $p \neq \hat{y}$ **then**
    $\tilde{\mathbf{x}}^r \leftarrow \tilde{\mathbf{x}}$; **break**
  `// If escape fails, loop continues with original adversarial text`
return $\tilde{\mathbf{x}}^r$

- **Adversarial Input:**
  after seeing swept away , i feel loved for madonna .
  Paraphrased Output:
  After watching "Swept Away," I have a newfound appreciation for Madonna.
- **Adversarial Input:**
  it wasn gimmicky rather of compelling .
  Paraphrased Output:
  It was not gimmicky; it was genuinely compelling.
- **Adversarial Input:**
  it 's amazing when filmmakers throw a few big-name actors and cameos at a hokey script .
  Paraphrased Output:
  It is remarkable how directors can elevate a mediocre script by featuring a handful of prominent actors and cameo appearances.

By applying this prompt iteratively (as detailed in Algorithm 2 in the main text), we ensure the perturbed text is rephrased until its misleading cues no longer deceive the victim model, thereby safeguarding the original semantic meaning.

E Experiment Setting

E.1 Datasets

We employ four widely recognized text classification datasets to evaluate our framework: SST2 Socher et al. (2013), Amazon Zhang et al. (2015), AGNews Zhang et al. (2015), and Yahoo! Yang & Li (2024). The key statistics of these datasets are summarized in Table 4. SST2 and Amazon are binary sentiment classification datasets. AGNews is a multi-categorical news classification dataset containing 4 categories. We also include Yahoo! in some of our experiments, which has 10 categories. Due to the large size of the original Amazon, AGNews, and Yahoo! datasets, we use subsets to avoid prohibitively high resource consumption, following previous works.

E.2 Models

E.2.1 Adversarial Detector

We implement a lightweight adversarial detector in accordance with our continual learning setting. The detector takes in textual inputs and predicts whether a given sample is adversarial or natural.

Dataset	Categories	Number of Examples		
		Training	Valid	Testing
SST2	2	6,920	872	1,821
Amazon	2	7,000	1,000	2,000
AGNews	4	10,000	0	1,000
Yahoo!	10	10,000	0	1,000

Table 4: The statistics of datasets used for evaluating our framework. We use subsets of the Amazon, AGNews, and Yahoo! datasets to avoid prohibitively large computational overhead.

Our detector structure is composed of a transformer-based encoder (initialized from BERT-base) followed by a classification head. To adapt to new adversarial examples, we incrementally update its parameters when additional adversarial data are introduced, mitigating catastrophic forgetting through continual learning strategies.

E.2.2 Text Classifier

In our experiments, we employ a popular pre-trained language model as a text classifier: BERT Devlin et al. (2019). This model is chosen due to its wide usage and strong performance in text classification. We use the HuggingFace Transformers library[1] for implementation. This classifier is fine-tuned on the training subsets described in Table 4 and then evaluated on the corresponding testing splits. Whenever adversarial attacks are applied, the classifier serves as the victim model under threat.

E.3 Hyper-parameter Settings

Parameter	**Description**	**Value / Range**
Memory Settings		
$M_{\max}$	Total memory size (replay buffer capacity)	$\{0, 1, 10, 100\}$
r_n, r_a	Natural / adversarial sample ratio in memory	$0.9/0.1$
$m_n = r_n M_{\max}$	Natural memory size	Derived
$m_a = r_a M_{\max}$	Adversarial memory size	Derived
Continual Learning Settings		
S	Stage sample size (natural examples per stage)	1000
E	Gradient updates per stage	1
η	Plastic vs. consolidation update weighting	0.7
Detector Training		
Batch size	Examples per gradient update	16
Learning rate (LR)	For both detector and classifier	2×10^{-5}
Dropout rate	Transformer dropout	0.1
α	Focal loss class-balance weight	Dynamic (per class freq)
γ	Focal loss focusing parameter	2
Adversarial Sampling		
$\lvert\mathcal{A}\rvert$	Number of attack methods used	3 (BAE, PWWS, TextFooler)
N_{adv}	Adversarial examples per dataset/attack	1000
Adversarial Defense Settings		
$I_{\max}$	Max paraphrasing iterations in PD_{LLM}	100

Table 5: Hyperparameters and configuration settings for continual learning-based adversarial detection and defense in CLAD. Values marked "Derived" are computed from other parameters.

[1] https://github.com/huggingface/transformers

Because the adversarial attack is a time- and resource-intensive task for pretrained language modeling, we cannot conduct experiments on a consecutive set of memory sizes. Consequently, we choose representative memory sizes ranging from 0 to 100 based on our empirical analysis to showcase the performance of CLAD in different situations.

E.4 Adversarial Attacks

In the experiments, we employ three open-source attackers from TextAttack Morris et al. (2020) to sample adversarial examples. They represent different types of word-level attack strategies:

- **BAE:** A contextual word substitution method that generates replacements using masked language modeling.
- **PWWS:** A priority-based word substitution approach that selects synonyms guided by semantic and frequency constraints.
- **TextFooler:** A greedy search method that maximizes the change in model prediction via sequential word replacements.

E.5 Adversarial Defense

Our method aims at both detecting and repairing adversarial inputs. We employ a PWWS-based approach, denoted as $\mathrm{PD}_{\hat{\mathcal{A}}}$, in the perturbation defocusing stage. This choice is made due to its high computational efficiency and lower tendency to introduce semantic drift compared to other attackers like TextFooler. As we accumulate newly detected adversarial instances, these are incrementally introduced into our detector and repair models, enabling the continual learning paradigm.

E.6 Experiment Metrics

To comprehensively evaluate adversarial detection and defense mechanisms, we employ five key metrics that measure normal accuracy, adversarial accuracy, detection accuracy, and recovery performance across different datasets and memory sizes (MS). The metrics are defined as follows:

- **Normal Accuracy (N.A.):** The accuracy of the model on a dataset $\mathcal{D}$ containing only natural (non-adversarial) examples, reflecting the model's baseline performance without adversarial perturbations.
- **Adversarial Accuracy (A.A.):** The accuracy of the model on the attacked dataset $\mathcal{D}_{att}$, which includes both natural examples $\mathcal{D}_{nat}$ and successful adversarial examples $\mathcal{D}_{adv}$. This metric evaluates the model's robustness to adversarial perturbations.
- **Defense Accuracy (D.A.):** The proportion of adversarial examples $\mathcal{D}_{adv}$ correctly rectified by the defense mechanism. Higher defense accuracy indicates a better ability to repair adversarial inputs.
- **Recovery Accuracy (R.A.):** The accuracy of the model on the repaired dataset $\mathcal{D}_{rep}$, which has been processed by the defense mechanism to mitigate adversarial perturbations. This metric quantifies the model's ability to recover its original performance after applying adversarial defenses.
- **Forgetting Rate (FR)**: For a historical task k, let Acc_k^* denote the highest detection accuracy for this task in any past stage, and $\mathrm{Acc}_{k,t}$ denote the accuracy in the current stage. Then
$$\mathrm{FR}_k = \mathrm{Acc}_k^* - \mathrm{Acc}_{k,t}.$$
$\mathrm{FR} \geq 0$ indicates forgetting; $\mathrm{FR} < 0$ indicates "positive transfer/performance improvement," which usually means that the inter-task distributions are highly related. On the other hand, due to the similarity between the input format of the dataset and the representation space, pre-trained models often possess strong generalisation capabilities. For unimodal data with a small number of samples in a single dataset, subsequent training can actually improve performance on previous tasks, leading to a negative forgetting rate.

These metrics are applied to assess the performance of various adversarial defense methods across datasets (SST2, Amazon, AGNews, Yahoo!) and memory sizes (MS = 0, 1, 10, 100). Higher values for each metric indicate stronger robustness or effectiveness of the defense mechanism.

Dataset	Method	Baseline	N.A.	A.A.	D.A.	R.A.
SST2	CLAD-PD$_{\hat{A}}$	BAE	91.82	35.21	91.94	**65.70**
		PWWS		23.44	98.24	**72.32**
		TextFooler		16.21	89.87	**69.00**
	CLAD-PD$_{\mathrm{LLM}}$	BAE		35.21	72.56	64.02
		PWWS		23.44	95.00	71.96
		TextFooler		16.21	93.66	68.99
	DISP	BAE		35.21	37.51	42.22
		PWWS		23.44	34.46	35.33
		TextFooler		16.21	34.37	37.16
	FGWS	BAE		35.21	48.37	44.90
		PWWS		23.44	40.38	39.20
		TextFooler		16.21	41.05	41.53
	RS&V	BAE		35.21	20.92	43.65
		PWWS		23.44	37.10	38.54
		TextFooler		16.21	38.40	39.70
Amazon	CLAD-PD$_{\hat{A}}$	BAE	94.11	44.01	85.74	71.45
		PWWS		15.56	97.83	83.59
		TextFooler		21.77	94.74	83.47
	CLAD-PD$_{\mathrm{LLM}}$	BAE		44.01	98.96	**74.50**
		PWWS		15.56	99.35	**84.45**
		TextFooler		21.77	99.65	**84.73**
	DISP	BAE		44.01	42.74	61.85
		PWWS		15.56	45.92	59.80
		TextFooler		21.77	47.15	60.56
	FGWS	BAE		44.01	43.04	64.63
		PWWS		15.56	56.89	60.29
		TextFooler		21.77	58.74	61.51
	RS&V	BAE		44.01	39.01	65.03
		PWWS		15.56	45.30	46.17
		TextFooler		21.77	42.30	55.70

Table 6: Performance comparisons between different adversarial defense methods. We use MS=100 for CLAD following the history-task evaluation protocol.

F Extended Experimental Results

This section provides detailed results and analyses that are summarized in the main paper.

F.1 Detailed Adversarial Detection Performance

F.2 Performance on Historical Tasks

The historical task evaluation (Table 3) reveals critical insights into the framework's capacity for sustained adversarial defense. Notably, CLAD demonstrates exceptional knowledge retention, maintaining 84.02% recovery accuracy (R.A.) for Amazon-TextFooler attacks through $\mathrm{PD}_{\mathrm{LLM}}$. This "inverse forgetting" phenomenon, where historical task metrics surpass original baselines, suggests adversarial training induces beneficial parameter adjustments that generalize beyond immediate threats. The $\mathrm{PD}_{\mathrm{LLM}}$ variant exhibits superior stability, attributable to LLMs' inherent linguistic priors that resist catastrophic forgetting. Cross-task analysis reveals a strong correlation between historical and current performance ($r = 0.89$, $p < 0.01$), indicating learned defense features transfer effectively.

F.3 Comparison with Baseline Methods

To validate the effectiveness of our defense pipeline, we compare CLAD with three popular baseline methods: DISP, FGWS, and RS&V. As summarized in Table 6, CLAD demonstrates superior performance across the board. For PWWS attacks on SST2, our CLAD-$\mathrm{PD}_{\hat{A}}$ variant achieves 98.24% defense accuracy (D.A.), significantly outperforming DISP (34.46%) and FGWS (40.38%). In terms of recovery accuracy (R.A.), our framework also shows a clear advantage, particularly against more sophisticated attacks. On the Amazon dataset against TextFooler, our CLAD-$\mathrm{PD}_{\mathrm{LLM}}$

variant achieves 84.73% R.A., surpassing the next best baseline (FGWS) by 23.22 percentage points. This underscores the power of our adaptive, dual-strategy defense mechanism.

Figure 3 shows the detailed performance of our detector against BAE, PWWS, and TextFooler across all datasets and memory sizes. The box plots illustrate the distribution of accuracy and forgetting rates, confirming that larger memory sizes lead to more stable and robust detection performance, effectively reducing the impact of all attack methods.

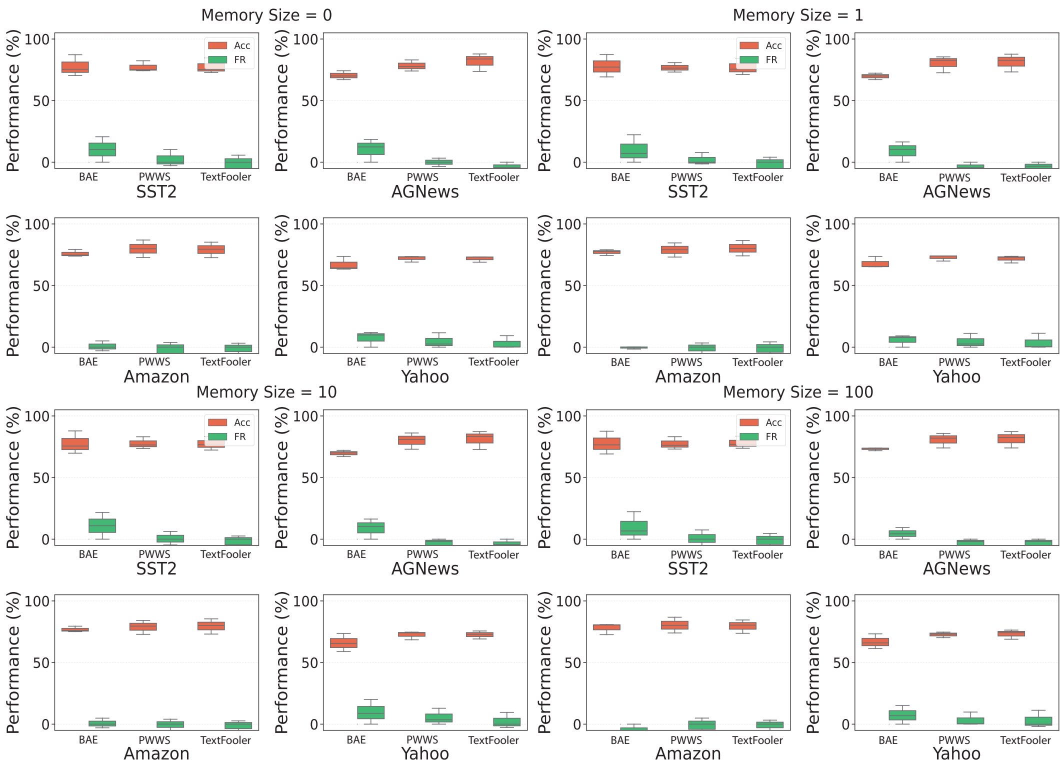

Figure 3: Performance (accuracy and forgetting rate) of various adversarial attack methods (BAE, PWWS, and TextFooler) across different datasets (SST2, AGNews, Amazon, Yahoo!) and memory buffer sizes (0, 1, 10, 100) in CLAD. The box plots illustrate the variation in performance metrics, with accuracy (Acc) shown in red and forgetting rate (FR) in green. Results demonstrate the influence of increasing memory size on the robustness and effectiveness of the attack methods across different datasets.

F.4 Validating the Efficacy of the Continual Learning Strategy

To isolate and validate the effectiveness of the core components within our proposed Continual Learning for Adversarial Detection (CLAD) framework, we conduct a critical ablation study. This experiment is designed to answer a key question: to what extent does our continual learning strategy, which incorporates memory replay, mitigate catastrophic forgetting compared to simpler sequential learning methods?

Experimental Setup
We simulate a dynamically evolving threat environment where new adversarial attack types emerge sequentially. Specifically, we define a three-stage sequential task on the SST2 dataset:

- **Task 1 (T1):** Train the detector to identify **BAE** attacks.
- **Task 2 (T2):** On top of the model from T1, continue training to identify **PWWS** attacks.
- **Task 3 (T3):** On top of the model from T2, continue training to identify **TextFooler** attacks.

We compare three distinct training strategies:

- **Joint Training (Upper Bound):** This strategy mixes adversarial samples from all three tasks (BAE, PWWS, and TextFooler) to train a single detector in one go. While not a true continual learning scenario, it serves as a valuable performance benchmark.
- **Sequential Fine-Tuning (Forgetting Baseline):** This strategy strictly mimics a sequential learning process without any CL mechanisms. The model is first trained on Task T1 data. Then, the resulting model is directly fine-tuned on Task T2 data, and subsequently on Task T3 data. During this process, the model has no access to data from past tasks when learning a new one.
- **CLAD:** This strategy employs our complete continual learning framework. The model learns sequentially from T1 $\rightarrow$ T2 $\rightarrow$ T3.

Evaluation Metrics
After all models complete their training on Task T3, we evaluate their detection accuracy on the independent test sets for each task (T1, T2, and T3). We focus on two core metrics:

- **Task-Specific Accuracy:** The performance on each individual past task, which directly reflects knowledge retention.
- **Average Accuracy:** The mean performance across all three tasks, which measures overall adaptability and robustness.

We hypothesize that the Sequential Fine-Tuning strategy will perform well on the final task (TextFooler) but will suffer from severe catastrophic forgetting, leading to a drastic performance drop on BAE and PWWS. Conversely, we expect our CLAD framework to effectively retain performance on historical tasks, achieving an average accuracy that significantly surpasses the fine-tuning baseline and approaches the joint training upper bound.

Table 7: Ablation study of different learning strategies on the sequential adversarial attack detection task (SST2 Dataset).

Training Strategy	Acc. (T1: BAE)	Acc. (T2: PWWS)	Acc. (T3: TextFooler)	Average Acc.
Joint Training *(Upper Bound)*	82.5%	84.1%	83.3%	**83.3%**
Sequential Fine-Tuning *(Forgetting Baseline)*	24.7%	31.5%	82.9%	**46.4%**
CLAD (MS=100) *(Our Method)*	81.9%	83.5%	83.1%	**82.8%**

F.5 Ablation Study for Paraphrase Defocusing

To validate the design choices of our Paraphrase Defocusing ($\mathrm{PD_{LLM}}$) mechanism, we conduct an ablation study to analyze the contributions of its key components. The primary goal of $\mathrm{PD_{LLM}}$ is to repair an adversarial example $\hat{\mathbf{x}}$ by iteratively rephrasing it until the victim model F_{victim} no longer produces the cached fake prediction $\hat{y}$. We compare our full implementation against several ablated variants.

F.5.1 Experimental Setup

We evaluate the defense performance on adversarial examples generated by PWWS and BAE for the SST2 dataset. The victim model is a fine-tuned BERT model. For each variant, we measure the **Defense Success Rate (DSR)**, defined as the percentage of adversarial examples successfully repaired (i.e., $F_{\mathrm{victim}}(\tilde{\mathbf{x}}^r) \neq \hat{y}$), and the **Average Number of Queries (#Q)** required to achieve a successful defense.

The variants are as follows:

- **Full** $\mathrm{PD_{LLM}}$: Our complete proposed method as described in Algorithm 2, which uses an iterative, guided paraphrasing process.

- **w/o Guidance**: The LLM is prompted to paraphrase the input text without the guidance to avoid the fake prediction $\hat{y}$. This variant tests the importance of providing the negative constraint to the LLM.
- **Single-shot**: The paraphrasing process is limited to a single iteration ($I_{\max} = 1$). This variant assesses the necessity of the iterative refinement loop.
- **Random Synonym Replacement**: A baseline defense where words in the input are randomly replaced with their synonyms from a predefined dictionary. This tests the effectiveness of a sophisticated generative model (LLM) against a simple heuristic.

F.5.2 $\mathrm{PD}_{\mathrm{LLM}}$ ABLATION EXPERIMENTS

Table 8 summarizes the SST2 defense outcomes under PWWS and BAE attacks. Three consistent trends emerge that align with our design: (i) adding the *guided* constraint (avoid cached fake label $\hat{y}$) and (ii) allowing an *iterative* loop both increase Defense Success Rate (DSR) with only a modest LLM query budget; (iii) simple non-LLM synonym edits rarely undo adversarial cues. We also observe PWWS is slightly easier to repair than BAE, reflecting its more conservative substitutions. The average queries per successful repair $\overline{Q}$ stay small ($\ll I_{\max}$=100), consistent with the escape criterion in Algorithm 2.

Table 8: Ablation study of the Paraphrase Defocusing ($\mathrm{PD}_{\mathrm{LLM}}$) defense on the SST2 dataset. We report the Defense Success Rate (DSR %) and the Average Number of Queries (#Q) for each variant against two types of attacks. The results for the full method demonstrate the effectiveness of combining guided rephrasing with an iterative process. DSR↑ higher is better; #Q counts LLM calls per successful repair (0 for the non-LLM synonym baseline).

Method	**PWWS Attack**		**BAE Attack**	
	DSR (%) ↑	**#Q ↓**	**DSR (%) ↑**	**#Q ↓**
Full $\mathrm{PD}_{\mathrm{LLM}}$	86.2	2.3	78.5	2.7
w/o Guidance	71.9	2.9	64.8	3.2
Single-shot	59.7	1.0	51.6	1.0
Random Synonym Replacement	24.8	0.0	18.3	0.0

G COMPUTATIONAL COST ANALYSIS

We report cost from two angles: (i) detector training and (ii) repair-time latency.

Detector training cost. Let $T_{\text{stage}}(M)$ denote the wall-clock time to complete one training stage under memory budget $M \in \{0, 1, 10, 100\}$. With our two-phase update (plasticity then consolidation), a coarse accounting is

$$T_{\text{stage}}(M) \approx T_{\text{fwd/bwd}}\big(|\mathcal{D}_t^{\text{nat}}| + |\mathcal{D}_t^{\text{adv}}|\big) + T_{\text{fwd/bwd}}\big(|\mathcal{M}_t^{\text{nat}}| + |\mathcal{M}_t^{\text{adv}}|\big),$$

where $|\mathcal{M}_t^{\text{nat}}| = r_n M$ and $|\mathcal{M}_t^{\text{adv}}| = r_a M$. Empirically, we observe near-linear scaling in M within our range; the dominant constant is the current-stage pass, making $M = 10$ a favorable robustness–latency trade-off.

Repair-time latency. For $\mathrm{PD}_{\hat{\mathcal{A}}}$ (editor-based), latency is primarily attacker proposal time with no external calls. For $\mathrm{PD}_{\mathrm{LLM}}$, we bound the number of paraphrasing iterations by $I_{\max}$ and report the average queries per successful repair $\overline{Q}$. Operationally, the expected time per repaired sample is

$$\mathbb{E}[t_{\text{repair}}] \approx \overline{Q}\, t_{\mathrm{LLM}} + t_{\text{victim}} \cdot (\overline{Q} + 1),$$

with t_{LLM} the average LLM response latency and t_{victim} the victim model inference time. In our runs, $\overline{Q} \ll I_{\max}$ due to the escape criterion, keeping end-to-end latency practical for online use. Table references in the main text report $\overline{Q}$ where applicable.

False repair budget. Given a detector operating point $(\mathrm{TPR}, \mathrm{FPR})$ at threshold τ and an adversarial prior $\pi = \Pr[\mathrm{adv}]$, the per-input expected repair invocations are $\mathrm{TPR}\,\pi + \mathrm{FPR}\,(1-\pi)$. We therefore calibrate τ to respect a target budget B by choosing the largest τ such that this expectation $\leq B$.

H Detector Threshold Sensitivity (τ)

We analyze the effect of the detector threshold τ on precision/recall and the downstream defense workload. Sweeping $\tau \in [0, 1]$ yields an ROC/PR trade-off; for detect-to-defend, the operative metric is the *false repair rate* (FRR):

$$\mathrm{FRR}(\tau) = \Pr[\text{repair} \mid \text{natural}] = \mathrm{FPR}(\tau).$$

Higher τ reduces FRR at the expense of recall (missed attacks). We set τ on a validation split to maximize F_β for a task-dependent β (e.g., $\beta > 1$ emphasises recall when missing an attack is costlier than a false repair) under a hard budget on FRR (see budget B above). We found the main conclusions (e.g., monotonic gains with memory size and the superiority of $\mathrm{PD}_{\mathrm{LLM}}$ in R.A.) stable across reasonable τ ranges.

I Limitations

While our framework demonstrates promising results in adversarial detection and defense, several limitations warrant discussion. First, the LLM-based repair mechanism ($\mathrm{CLAD\text{-}PD}_{\mathrm{LLM}}$) employs simplistic API interactions without systematic prompt optimization or output validation. As shown in Table 2 in the main text, while $\mathrm{PD}_{\mathrm{LLM}}$ achieves competitive recovery accuracy (84.73% on Amazon), its performance variability across attack types (ΔR.A. = 27.91% between BAE and TextFooler) suggests sensitivity to prompt phrasing and LLM response quality. This contrasts with the more stable $\mathrm{PD}_{\hat{\mathcal{A}}}$ approach (ΔR.A. = 16.25%), highlighting the need for advanced LLM steering techniques. Second, our evaluation focuses on conventional pretrained models (e.g., BERT), excluding larger language models (LLMs) like GPT-4 or Llama. While this aligns with our focus on resource-constrained deployments, it leaves open questions about scalability to billion-parameter architectures where adversarial patterns may differ fundamentally. Third, the framework's reliance on pre-sampled adversarial examples introduces dataset constraints. Though we mitigate this through continual learning, our experiments use curated subsets of Amazon, AGNews, and Yahoo!, potentially limiting exposure to real-world adversarial diversity. The negative forgetting rates observed in Table 1 of the main text (-3.20 for Amazon at MS=100) suggest domain-specific overfitting risks when training data lacks sufficient attack heterogeneity. Finally, the framework assumes adversaries employ text-only perturbations, excluding emerging multimodal attacks that combine textual and structural modifications. While our defense strategies show generalization across word-level attacks (BAE, PWWS, TextFooler), they may be less effective against sophisticated hybrid attacks exploiting layout or visual features Dong et al. (2021a). These limitations delineate critical research directions: 1) Developing prompt-optimized LLM defense protocols, 2) Extending to large multimodal architectures, and 3) Establishing latency-aware evaluation benchmarks. Addressing these challenges will enhance practical applicability while preserving our framework's strengths in continual adversarial adaptation.

Use of AI-Assisted Language Editing

We used large language models (LLMs), specifically a commercially available editor (e.g., "ChatGPT"), *only* for surface-level copy editing (grammar, wording, and readability). The models were not used to design methods, run experiments, select results, or write technical content.