ContiGuard: A Framework for Continual Toxicity Detection Against Evolving Evasive Perturbations

Hankun Kang
School of Computer Science
Wuhan University
Wuhan, Hubei, China
kanghankun@whu.edu.cn

Xin Miao
School of Computer Science
Wuhan University
Wuhan, Hubei, China
miaoxin@whu.edu.cn

Jianhao Chen
School of Computer Science
Wuhan University
Wuhan, Hubei, China
Zhongguancun Academy
Beijing, China
chenjianhao@whu.edu.cn

Jintao Wen
School of Computer Science
Wuhan University
Wuhan, Hubei, China
23jtwen@whu.edu.cn

Mayi Xu
School of Computer Science
Wuhan University
Wuhan, Hubei, China
xumayi@whu.edu.cn

Weiyu Zhang*
Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Shandong Computer Science Center (National Supercomputer Center in Jinan)
Qilu University of Technology (Shandong Academy of Sciences)
Jinan, Shandong, China
Shandong Provincial Key Laboratory of Computing Power Internet and Service Computing
Shandong Fundamental Research Center for Computer Science
Jinan, Shandong, China
zwy@qlu.edu.cn

Wenpeng Lu*
Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Shandong Computer Science Center (National Supercomputer Center in Jinan)
Qilu University of Technology (Shandong Academy of Sciences)
Jinan, Shandong, China
Shandong Provincial Key Laboratory of Computing Power Internet and Service Computing
Shandong Fundamental Research Center for Computer Science
Jinan, Shandong, China
wenpeng.lu@qlu.edu.cn

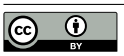
Tieyun Qian*
School of Computer Science
Wuhan University
Wuhan, Hubei, China
Zhongguancun Academy
Beijing, China
qty@whu.edu.cn

Abstract

Toxicity detection mitigates the dissemination of toxic content (e.g., hateful comments, posts, and messages within online social actions) to safeguard a healthy online social environment. However, malicious users persistently develop evasive perturbations to disguise toxic content and evade detectors. Traditional detectors or methods are static over time and are inadequate in addressing these evolving evasion tactics. Thus, continual learning emerges as a logical approach to dynamically update detection ability against evolving perturbations. Nevertheless, disparities across perturbations hinder the detector's continual learning on perturbed text. More importantly, perturbation-induced noises distort semantics to degrade comprehension and also impair critical feature learning to render detection sensitive to perturbations. These amplify the challenge of continual learning against evolving perturbations.

In this work, we present ContiGuard, the first framework tailored for continual learning of the detector on time-evolving perturbed text (termed continual toxicity detection) to enable the detector to continually update capability and maintain sustained resilience against evolving perturbations. Specifically, to boost the comprehension, we present an LLM powered semantic enriching strategy, where we dynamically incorporate possible meaning and toxicity-related clues excavated by LLM into the perturbed text to improve the comprehension. To mitigate non-critical features and amplify critical ones, we propose a discriminability driven feature learning strategy, where we strengthen discriminative features while suppressing the less-discriminative ones to shape a robust classification boundary for detection. Additionally, we introduce a historical

*Corresponding authors.



WWW '26, Dubai, United Arab Emirates

ACM ISBN 979-8-4007-2307-0/2026/04
https://doi.org/10.1145/3774904.3792545

capability replay strategy to preserve previously learned features via feature alignment to alleviate capability forgetting. To the best of our knowledge, this work is the first study on continual toxicity detection against time-evolving evasive perturbed text. Extensive experiments prove the superior performance of ContiGuard over both existing detectors and continual methods. Code and dataset are available at https://github.com/khk-abc/ContiGuard.

***Warning**: This paper contains discussions of harmful content that may be disturbing to some readers.*

CCS Concepts

• **Information systems → Social networks**; • **Security and privacy → Social aspects of security and privacy**; • **Computing methodologies → Natural language processing**.

Keywords

Online Content Moderation, Toxic Content Detection, Evolving Evasive Perturbations, Continual Toxicity Detection

ACM Reference Format:

Hankun Kang, Xin Miao, Jianhao Chen, Jintao Wen, Mayi Xu, Weiyu Zhang, Wenpeng Lu, and Tieyun Qian. 2026. ContiGuard: A Framework for Continual Toxicity Detection Against Evolving Evasive Perturbations. In *Proceedings of the ACM Web Conference 2026 (WWW '26), April 13–17, 2026, Dubai, United Arab Emirates.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3774904.3792545

1 Introduction

Toxicity detection aims to identify toxic text like hateful content [11], social bias [48], and stereotypes [16], to safeguard healthy social interactions, which is critical for society safety [7]. However, detection efforts face significant challenges due to the continuous evolution of evasive tactics: malicious users regularly create text perturbations to bypass detectors, such as character repetition (*iiiddioot*), homoglyph substitution (*id10t*), and other potential tactics.

Existing detection methods struggle to adapt to the continuous evolution of evasive perturbations. Most of them are tailored for ordinary text (e.g., idiot), limiting their ability to identify perturbed toxic text [4, 24]. A few methods incorporate specific types of perturbed text into training [4, 24], excelling at detecting known perturbations such as character repetition. But they struggle to deal with continually appearing perturbations since they do not update detection ability against appearing perturbations over time, like homoglyph substitution. Consequently, the challenge of continually detecting emerging perturbed toxic text remains an open but overlooked problem, posing risks to society's safety.

Considering continual learning can update the detector's capability over time, we model different types of perturbed text as the ones distributed in distinct perturbation domains, and we harness domain continual learning to update the detection ability of the detector, enabling the detector to evolve along with constantly appearing perturbations and making it suitable for the evolving evasive perturbed text.

While continual learning is a logical approach to address evolving perturbed text in toxicity detection, it remains hindered by several critical challenges. Specifically, the heterogeneity of perturbations (e.g., *iiiddioot* and *id10t*) leads to the forgetting of the detector's historical detection capability over time. Furthermore, perturbations disrupt the original textual structure, making the detector struggle to comprehend text, especially the hidden toxicity within text. In addition, the detector learns noisy or irrelevant features derived from perturbations, such as manipulation variations of perturbation (e.g., *iiiddioot* vs. *idiiiiottt*, both perturbed from *idiot*). These features are noisy and redundant, less critical for toxicity classification, and even obscure the learning of critical toxicity-related features, rendering the detector dependent on these features and sensitive to diverse perturbed text. In summary, the perturbations corrupt the original semantics, causing the detector to struggle to understand text. They also introduce less-critical features, which occupy the detector's representational capacity and deviate the detector's focus from critical features, degrading continual detection against diverse perturbed text.

In this work, we propose ContiGuard, the first framework to continually identify evolving perturbed toxic text. Firstly, to improve the text comprehension hindered by corrupted semantics, we propose an *LLM powered semantic enriching strategy*, where we analyze that LLM can be utilized to reason auxiliary information from its parametric knowledge to enrich the insights into perturbed text, enhancing the text comprehension for detection. During the incorporation of the information into perturbed text, it also acts as a momentum factor to mitigate the risk of trapping into local optima. Specifically, we dynamically incorporate possible meaning and toxicity-related clues excavated by LLM into perturbed text to boost the comprehension.

To mitigate the interference of non-critical features and focus more on critical ones to make the detector robust to diverse perturbed text, we propose a *discriminability driven feature learning strategy* since the discriminability is inherently critical to shape a reliable boundary for the robust classification, where we measure each feature's contribution using attribution analysis [50] and differentiate their discriminative power. We then strengthen the discriminative features via global rotation and suppress less discriminative ones by unlearning, forcing the detector to learn core features for the robust detection.

To alleviate the forgetting of detection capability due to the lost historical features caused by the gaps among perturbations, we further adopt a *historical capability replay strategy*, where we align features of memory samples between the old and current detectors to preserve historical features, maintaining the detector's prior detection capability.

Our contributions are summarized as follows.

- We first highlight a crucial yet overlooked continual toxicity detection problem, i.e., enabling the detector to continually update its detection capability along with evolving, evasive perturbed text prevalent in real-world scenarios, which is essential for safeguarding social safety.
- We propose ContiGuard, the first framework for continual toxicity detection, performing LLM powered semantic enriching, discriminability driven feature learning, and historical capability replay to address the key challenges.
- Extensive experiments show the superior performance of ContiGuard against evolving perturbed text compared to existing detectors, static methods, and continual approaches.

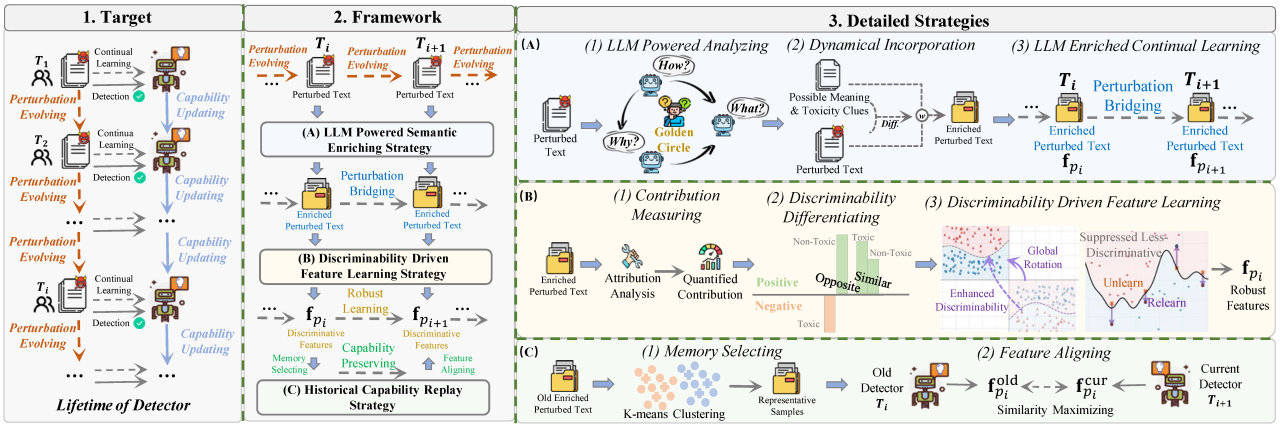


Figure 1: Illustration of our ContiGuard framework.

2 Methodology

2.1 Problem Formulation

Perturbed text is modeled as the text distributed in perturbation domains. Correspondingly, continual toxicity detection against time-evolving perturbed text is formulated as domain-incremental toxicity classification with incremental perturbation domains. Specifically, perturbation domain p_i appears at the i-th moment within the lifespan T of a lifelong detector F. Given a perturbed toxicity dataset D_{p_i} comprising N_{p_i} samples $(x_{p_i,j}, y_{p_i,j})$ where $j = \{1, 2, \ldots, N_{p_i}\}$, $x_{p_i,j}$ denotes the text perturbed by p_i, and $y_{p_i,j} \in \mathcal{Y}$ with $\mathcal{Y} = \{0, 1\}$ is the toxicity label. The task optimizes the parameters θ_F by minimizing the loss to maximize the conditional probabilities of labels, enabling the detector to continually recognize toxicity in evolving perturbed text over time as follows:

$$\theta_F^* := \arg\max_{\theta_F} \prod_{i}^{T} \prod_{j}^{N_{p_i}} p_{\theta_F}(y_{p_i,j}|x_{p_i,j}), \tag{1}$$

where $p_{\theta_F}(y_{p_i,j}|x_{p_i,j})$ denotes the conditional probability of the true label computed by the detector.

2.2 ContiGuard Framework

As Fig. 1 shows, ContiGuard mainly includes three strategies:

(1) *LLM powered semantic enriching strategy.* Evasive perturbations intentionally obfuscate text, rendering toxicity implicit and impeding text comprehension. To address this, we analyze that LLM can provide auxiliary information to enrich insights into perturbed text for toxicity detection and can smooth gradients during optimization. Correspondingly, we leverage LLM to capture potential original meanings and toxicity-related clues and we integrate them into perturbed text to enhance comprehension.

(2) *discriminability driven feature learning strategy.* Perturbations induce noises to make the features of perturbed text redundant and noisy, causing non-critical features to occupy the detector's representation capacity and rendering detection sensitive to various perturbations. Considering the discriminability is the critical property to shape a reliable classification boundary for robust toxicity detection, we measure the discriminative power of features for toxicity detection, and we strengthen the discriminative features by global flipping and suppress less discriminative ones, guiding the detector to prioritize learning critical features.

(3) *Historical capability replay strategy.* Various types of perturbations are significantly distinct so the historical detection capability against previous types of perturbed text may be lost. Hence, we retain the detector's historical capability by aligning the features of memory samples encoded with the old and current detectors.

2.2.1 *LLM Powered Semantic Enriching Strategy.*

Given the rich knowledge and reasoning ability of LLM, this strategy use LLM to mine useful auxiliary information to improve the understanding of perturbed text, such as possible meaning and toxicity-related clues.

(1) *Preliminaries* We first analyze how the auxiliary information from LLM enriches the perturbed text. Let X_{p_i}, X_{a_i}, and Y_{p_i} denote the perturbed text, LLM generated auxiliary information, and predicted label, respectively. The conditional probability of Y_{p_i} given X_{p_i} is decomposed as:

$$p(Y_{p_i}|X_{p_i}) = \int p(Y_{p_i}|X_{a_i} = x_{a_i}, X_{p_i}) p(X_{a_i} = x_{a_i}|X_{p_i}) \mathrm{d}x_{a_i}, \tag{2}$$

where x_{a_i} is the specific values of X_{a_i}. Hence, the objective of continual toxicity detection, maximizing $\prod_i^T p_{\theta_F}(Y_{p_i}|X_{p_i})$, becomes:

$$\max_{\theta_F} \prod_{i}^{T} \int p_{\theta_F}(Y_{p_i}|x_{a_i}, X_{p_i}) p_{\theta_{LLM}}(x_{a_i}|X_{p_i}) \mathrm{d}x_{a_i} \tag{3}$$

Based on $p_{\theta_F}(Y_{p_i}|x_{a_i}, X_{p_i})$, the auxiliary information enriches the insights into perturbed text and bridges the learning on different perturbed text over time via the shared insights (e.g., the same restored terms and similar clues within various perturbations). In addition, the LLM parameters θ_{LLM} remain fixed during optimization to render $p_{\theta_{LLM}}(x_{a_i}|X_{p_i})$ an optimization-independent but input-dependent factor, which encapsulates the probability tendency of the toxicity-related knowledge in the LLM to adjust the learning by weighting the detector's output probability.

Furthermore, perturbations introduce noises, resulting in a jagged optimization process that is highly prone to converging to local optima. In contrast, the auxiliary information structured via ordinary text is less noisy and more flat, and the gradient derived from such

information can act as a momentum term to mitigate the risk of the optimization process becoming trapped in local optima. Specifically, we incorporate the features of auxiliary information X_{a_i} into the features of perturbed text X_{p_i} by linearly weighting as:

$$f([X_{p_i}, X_{a_i}]) = \alpha f(X_{p_i}) + (1-\alpha) f(X_{a_i}), \tag{4}$$

where α denotes the weight and $f : X \mapsto \mathbb{R}^d$ is the function of extracting features from input, i.e., the encoder of detector. Then the gradients of incorporated features $f(\cdot) := f([X_{p_i}, X_{a_i}])$ have:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \theta_F} &= \frac{\partial \mathcal{L}}{\partial f(\cdot)} \frac{\partial f(\cdot)}{\partial f(X_{p_i})} \frac{\partial f(X_{p_i})}{\partial \theta_F} + \frac{\partial \mathcal{L}}{\partial f(\cdot)} \frac{\partial f(\cdot)}{\partial f(X_{a_i})} \frac{\partial f(X_{a_i})}{\partial \theta_F} \\ &= \alpha \frac{\partial \mathcal{L}}{\partial f(\cdot)} \frac{\partial f(X_{p_i})}{\partial \theta_F} + (1-\alpha) \frac{\partial \mathcal{L}}{\partial f(\cdot)} \frac{\partial f(X_{a_i})}{\partial \theta_F},\end{aligned} \tag{5}$$

where $\frac{\partial f(X_{p_i})}{\partial \theta_F}$ and $\frac{\partial f(X_{a_i})}{\partial \theta_F}$ represent the gradients derived from the perturbed text X_{p_i} and auxiliary information X_{a_i}, respectively. When the former introduced noises lead to the risk of trapping in local optima, the latter serves as momentum to alleviate this risk.

Next, we obtain auxiliary information and weights as follows.

(2) *Implementation* The evasive tactics make the toxicity to be implicitly hinted so that the LLM needs to conduct in-depth analysis rather than just scratching the surface. Inspired by the Golden Circle principle [49], a structured thinking logic which understands things from phenomena to essence through multi-questioning, we introduce a *How-Why-What* interrogation mechanism to require LLM to conduct considerable analysis as follows.

How Stage: LLM explores *how* to mine possible meaning and toxicity-related clues from perturbed text.

Why Stage: According to the *how* stage, LLM must give the reason *why* it performs this procedure.

What Stage: LLM explores *what* the possible meaning and toxicity-related clues is according to *how* and *why*.

Subsequently, we incorporate the auxiliary information into the perturbed text to enrich insights for detection. Since there are various differences (e.g., information gain) between the auxiliary information and the perturbed text in different samples, the information contributes differently to the toxicity judgment and we need to employ the auxiliary information in a dynamical way. Specifically, we introduce a difference-based dynamical cooperation, where we compute the point-wise difference between the features of both auxiliary information $\mathbf{x}_{a_i,j} \in \mathbb{R}^d$ and perturbed text $\mathbf{x}_{p_i,j} \in \mathbb{R}^d$, and we then compute the dynamical weights as follows.

$$\mathbf{w}_{p_i,j} = \sigma(\mathbf{W}_g^T(\mathrm{CONV}((\mathbf{x}_{p_i,j} - \mathbf{x}_{a_i,j})^2)) + \mathbf{b}_g), \tag{6}$$

where σ represents the sigmoid function, and CONV denotes the convolutional operation to locally smooth the differences since the point-wise differences are sensitive due to outlier values introduced by perturbations. Then dynamical weights $\mathbf{w}_{p_i,j}$ are computed by global linear projection ($\mathbf{W}_g$ and $\mathbf{b}_g$), and we further incorporate $\mathbf{x}_{a_i,j}$ and $\mathbf{x}_{p_i,j}$ to get the cooperated features $\mathbf{f}_{p_i,j} \in \mathbb{R}^d$ as follows.

$$\mathbf{f}_{p_i,j} = \frac{\mathbf{x}_{p_i,j} + \mathbf{w}_{p_i,j} \cdot \mathbf{x}_{a_i,j}}{(1 + \mathbf{w}_{p_i,j})} \tag{7}$$

2.2.2 Discriminability Driven Feature Learning Strategy. Noises in perturbations introduce trivial or irrelevant features, which renders the detector sensitive to various perturbed text. Hence, explicitly guiding the detector to learn critical features is essential for enhancing its robustness. Given that toxicity detection is inherently a classification task, the key to learning the critical feature lies in the core property of classification, i.e., discriminability. This is because the discriminability enables the detector to exhibit distinct probabilistic tendencies across classes (i.e., shape the robust inter-class classification boundary), making it the foundational support for breaking free from the interference of trivial features and enhancing the learning for critical ones. Accordingly, we propose a discriminability driven feature learning strategy to guide critical feature learning, which primarily comprises three stages:

(1) *Feature Contribution Measuring* To differentiate the discriminative power of features, we firstly quantify the probabilistic contribution caused by each feature on the class by attribution analysis based on integrated gradient as follows:

$$A_y(\mathbf{f}_{p_i,j}) = (\mathbf{f}_{p_i,j} - \mathbf{f}') \int_0^1 \frac{\partial F_y(\mathbf{f}' + \beta(\mathbf{f}_{p_i,j} - \mathbf{f}'))}{\partial \mathbf{f}_{p_i,j}} \mathrm{d}\beta, \tag{8}$$

where $\mathbf{f}' \in \mathbb{R}^d$ denotes the reference features when there is no information input (set to zero vector by default). $F_y : \mathbb{R}^d \mapsto \mathbb{R}$ represents the projection from the features to the output probability of class y, and β indicates the scaling coefficient. $A_y(\mathbf{f}_{p_i,j}) \in \mathbb{R}^d$ denotes the quantified contribution of feature $\mathbf{f}_{p_i,j}$ for the output probability of class y. When $A_y^k(\mathbf{f}_{p_i,j}) > 0$ and $A_y^k(\mathbf{f}_{p_i,j}) < 0$, it indicates that the k-th element of feature $\mathbf{f}_{p_i,j}^k$ has a positive and negative impact, i.e., increasing and decreasing probabilistic tendencies on the label y, respectively. For example, increasing the feature $\mathbf{f}_{p_i,j}^k$ will make the detector output a larger probability on the class y when $A_y^k(\mathbf{f}_{p_i,j}) > 0$ and vice versa.

(2) *Discriminative Feature Differentiating* When one feature contributes positively or negatively to both the *non-toxic* ($y = 0$) and *toxic* ($y = 1$) classes, the feature is less discriminative since it cannot make the detector exhibit significant probabilistic differences between classes, contributing less in shaping the classification boundary. Hence, we differentiate the discriminability of one feature according to whether its attribution differs on classes as follows:

$$\mathbf{m}_{\mathbf{f}_{p_i,j}}^{less} = \mathbf{I}\{A_{y=0}(\mathbf{f}_{p_i,j}) A_{y=1}(\mathbf{f}_{p_i,j}) > 0\}, \tag{9}$$

where $\mathbf{I}$ is the indicator function. $\mathbf{m}_{\mathbf{f}_{p_i,j}}^{less} \in \mathbb{R}^d$ represents the mask for the less-discriminative elements within feature $\mathbf{f}_{p_i,j}$. The parts of $\mathbf{m}_{\mathbf{f}_{p_i,j}}^{less,k} = 1$ contribute similarly on *toxic* and *non-toxic* classes.

(3) *Discriminability Driven Feature Learning* After differentiating discriminative power, we guide feature learning with discriminability as the core driver. First, we enhance the discriminative features to boost robustness. Specifically, when features rotate globally, the classification boundary should rotate accordingly, decoupling the discriminability from trivial attributes like absolute direction and position of the features and preserving it. Drawing on this insight, we rotate features and reclassify to strengthen discriminability. Given rotating high-dimensional features incurs high computational cost ($O(d^2)$), we adopt a flipping (a special case with $O(d)$ complexity) to rotate features, aiming to enhance the discriminability of features and improve robustness, shown as follows:

$$\mathcal{L}_{more} = \mathrm{CE}(\mathbf{W}_c^T(-\mathbf{f}_{p_i,j}) + \mathbf{b}_c, y_{p_i,j}), \tag{10}$$

where CE and $y_{p_{i,j}}$ are cross-entropy loss for classification and true label, respectively. $\mathbf{W}_c, \mathbf{b}_c$ denote the classifier head in the detector.

Furthermore, we unlearn less discriminative elements within features and relearn them to compose more robust features, shaping a reliable classification boundary. Considering neutral features should yield near-random predictions without any probabilistic tendency toward any class, we thus exploit a uniform distribution as supervision to unlearn the less discriminative elements, resetting them to be neutral. Subsequently, these elements are iteratively relearned multiple times to make them more robust.

$$\mathcal{L}_{\text{less}} = \text{CE}(\mathbf{W}_c^T(\mathbf{f}_{p_{i,j}} \circ \mathbf{m}^{\text{less}}_{\mathbf{f}_{p_{i,j}}}) + \mathbf{b}_c, \mathbf{u}), \tag{11}$$

where $\mathbf{u} \in \mathbb{R}^{|\mathcal{Y}|}$ is the uniform distribution over the label space.

2.2.3 Historical Capability Replay Strategy. To mitigate the forgetting of historical detection capability, we select top-k representative samples as memory samples to preserve the capability. Specifically, we mix memory samples into the training data, and we align the features of memory samples encoded by the old and current detectors since features encapsulate the detector's capability as follows:

$$\mathcal{L}_{align} = -\frac{1}{T}\sum_{i}^{T}(\frac{1}{N_{p_i}}\sum_{j}^{N_{p_i}}(\cos(\mathbf{f}^{old}_{p_{i,j}}, \mathbf{f}^{cur}_{p_{i,j}}) + Z_i)), \tag{12}$$

where $Z_i = -\log\sum_{k}^{N_{p_i}}\exp(\cos(\mathbf{f}^{old}_{p_{i,k}}, \mathbf{f}^{cur}_{p_{i,k}}))$ is the normalization factor. $\mathbf{f}^{old}_{p_{i,k}}$ and $\mathbf{f}^{cur}_{p_{i,k}}$ are the features of the j-th sample encoded by old and current detectors, respectively. cos is the cosine similarity. Finally, the total loss is:

$$\mathcal{L} = \mathcal{L}_{cls} + \lambda(\mathcal{L}_{more} + \mathcal{L}_{less}) + \gamma\mathcal{L}_{align}, \tag{13}$$

where $\mathcal{L}_{cls} = \text{CE}(\mathbf{W}_c^T\mathbf{f}_{p_{i,j}} + \mathbf{b}_c, y_{p_{i,j}})$ is the basic classification loss. λ and γ is the scale factors.

3 Experiments

3.1 Experimental Setup

Dataset Based on Jigsaw [27], a widely used large-scale English dataset for toxicity detection, we construct a perturbed dataset by perturbing the Jigsaw with 9 types of perturbations (named DynEscape). In DynEscape, each type of perturbed text is separately type-wise managed, and every type owns 2584/1294/430 samples in train/test/valid splits. The perturbations include: ***Insert*** (i#dio!t), ***Repeat*** (iiiddiott), ***Swap*** (idito), ***Remove*** (idot), ***Homo*** (id10t), ***Mask*** (id**t), ***Abbr*** (abbreviation/slang replacement, bite me → BTM), ***Auth*** (add authoritative text around the ordinary text to make it more convincing), and ***Distract*** (add distracting words around the ordinary text to make it more confusing). With DynEscape, we can evaluate the vulnerability of the existing detectors and static fine-tuning detection methods on evolving perturbed text, and we also can explore continual toxicity detection of ContiGuard against evolving perturbed text. Furthermore, to examine the generalization and practical application of ContiGuard, we conduct a cross-dataset evaluation on NoisyHate [58] dataset, which is collected from mixed multi-types of human-written perturbed text in the wild.

Evaluation Protocol All our fine-tuned detectors are composed of an encoder of BERT-base and a classification head of a two-layer forward network. We conduct three types of experiments and use accuracy as the metric for classification.

(1) We evaluate the exiting detectors on perturbed DynEscape to explore their vulnerability, including commercial APIs (PerspectiveAPI [32] and OpenAI Moderation [40]) and fine-tuned detectors (ToxicBERT/ToxicRoBERTa [20], Original/Multilingual/Unbiased variants of Detoxify [19], Paradetox [36], and DeTox [6]).

(2) To analyze performance degradation of the statically fine-tuned detector, we fine-tune the detector with one specific perturbation type and test the detector on the remaining types to simulate the scenario where the detector continually encounters the types of perturbed text created over time.

(3) We examine our continual learning (CL) strategies by comparing ContiGuard with adapted common CL methods on DynEscape, including LFL [22], EWC [26], LWF [34], GEM [37], EPI [55], CEAR [64], DUCT [65] and SOYO [54]. *Joint* trains the detector with all data, and *Stream* sequentially optimizes the detector by pipeline data, and they act as the reference bound.

3.2 Main Results

Firstly, the results of existing detectors are shown in Tab. 1. We find that the performance of existing detectors degrades significantly under all perturbations, with relative drops ranging from a maximum of 35.8% to a minimum of 14.4% compared to the ordinary text, denoting their susceptibility to perturbed text. This is because the perturbations obfuscate text to hide the toxicity, which makes the perturbed text significantly different from the ordinary ones, but the existing detectors are trained on ordinary text and ignore the adaptability to the perturbed text.

Secondly, the results of statically fine-tuned detectors are shown in Fig. 2. We find that the statically fine-tuned detectors perform well on the types of perturbed text used for fine-tuning (numbers on the diagonal), but perform poorly on other types of perturbed text (numbers outside the diagonal), where the performance drops in almost all types, and they even perform randomly on many types. We believe that the disparities among perturbations seriously impede the generalization across types so that the statically fine-tuned detectors on fixed specific types of perturbations are not suitable when the perturbations are continually created.

Finally, the continual detection results under different data orders are shown in Table 2. All CL methods continually update the detection capability of the detector to capture continual adaptability to perturbations that occur over time. However, existing CL methods are limited because they never consider the unique properties of the continuous toxicity detection task and even perform worse than *Stream*, while ContiGuard performs the best and outperforms *Joint*, despite *Joint* utilizing all data at every moment. We believe this is due to the hindrance of understanding the perturbed text due to semantic corruption and the difficulty in learning key features that are surrounded by the perturbation-introduced noisy features. Existing CL methods are never designed for these unique issues, resulting in limited performance. In contrast, ContiGuard enriches the perturbed text and performs robust feature learning to mitigate semantic corruption and noisy feature interference, achieving excellent and robust continuous detection.

Furthermore, Fig. 3 shows the performances in every type of perturbed text at the T_9 moment. We find that ContiGuard performs the best in most perturbation types and is more balanced than

Table 1: The results of ContiGuard and existing detectors on perturbed text. The subscripts indicate the relative performance degradation compared with ordinary text. The best scores are highlighted in bold.

Existing Detectors	Insert	Repeat	Swap	Remove	Homo	Mask	Abbr	Auth	Distract	Avg
PespectiveAPI	77.20	64.61	70.48	60.51	64.84	63.29	50.93	66.38	64.37	$64.73_{25.9\%\downarrow}$
OpenAI Moderation	66.15	52.24	61.13	56.49	50.78	59.35	51.00	84.47	81.14	$62.53_{25.2\%\downarrow}$
ToxBERT	60.90	60.97	58.81	58.34	53.55	55.87	58.35	73.26	75.12	$61.69_{14.4\%\downarrow}$
ToxRoBERTa	50.93	50.39	51.24	52.32	50.15	50.31	51.39	55.26	79.83	$54.65_{34.0\%\downarrow}$
$\text{Detoxify}_{\text{Original}}$	61.59	63.76	61.90	57.73	56.96	59.27	56.03	71.87	75.66	$62.75_{29.8\%\downarrow}$
$\text{Detoxify}_{\text{Multilingual}}$	60.20	58.50	56.41	56.96	82.53	59.20	52.63	66.62	81.92	$63.88_{26.6\%\downarrow}$
$\text{Detoxify}_{\text{Unbiased}}$	51.31	51.00	53.32	53.40	50.15	51.93	51.55	78.59	80.14	$57.93_{32.5\%\downarrow}$
Paradetox	51.31	51.47	51.85	51.62	49.92	52.63	51.16	69.55	72.57	$55.79_{35.8\%\downarrow}$
DeTox	69.55	52.47	58.73	59.58	52.63	62.44	50.23	53.71	66.92	$58.48_{28.3\%\downarrow}$
ContiGuard	85.57	87.23	85.18	83.97	90.40	82.94	80.58	92.19	90.15	$\mathbf{86.47}_{3.0\%\downarrow}$

Table 2: The results of ContiGuard and adapted CL methods. T_i denote averaged performance of all orders at i-th moment. $O^j_{T_9}$ denote performance of j-th order at T_9 moment. * indicates significant difference at p<0.001.

Continual	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	$O^1_{T_9}$	$O^2_{T_9}$	$O^3_{T_9}$	$O^4_{T_9}$
Joint	84.14	86.63	85.43	85.07	84.56	85.05	85.03	85.31	85.88	85.63	85.51	86.31	86.06
Stream	84.23	84.64	76.52	77.36	78.17	79.55	79.73	74.28	78.26	79.49	74.08	78.95	80.49
LFL	83.46	82.88	77.15	76.59	75.30	78.25	78.44	75.01	78.15	79.48	74.19	79.07	79.85
EWC	84.66	84.88	76.41	78.48	77.89	80.89	78.64	76.01	79.32	79.81	75.43	80.25	81.80
LWF	83.46	85.08	79.39	79.86	76.31	76.92	75.56	72.58	72.30	72.29	74.94	65.40	76.56
EPI	84.12	81.46	81.60	79.13	79.90	78.23	76.98	78.26	79.20	79.03	79.50	79.19	79.07
GEM	83.94	84.83	78.39	79.96	80.82	80.76	80.56	78.47	80.99	80.51	80.23	81.19	82.03
CEAR	84.35	84.39	72.56	78.60	80.45	80.97	81.22	81.30	80.60	82.71	80.04	77.01	82.65
DUCT	78.75	75.51	67.60	67.97	66.86	69.47	70.82	67.29	62.98	66.87	56.81	68.01	60.23
SOYO	80.47	81.10	80.12	79.67	79.52	79.46	79.27	79.66	80.07	80.18	79.99	80.12	79.98
ContiGuard	87.17	89.27	86.20	86.42	85.41	86.29	85.41	84.92	**86.47***	**85.67**	**86.28**	**86.34**	**87.58**

Figure 2: Results of statically fine-tuned detectors. Rows: types used for fine-tuning. Columns: types used for testing.

existing CL methods across types. For example, despite EPI being close to ContiGuard in *Auth* and *Distract*, it works very poorly in *Insert* and *Swap*, indicating a serious imbalance. By contrast, ContiGuard is close to or exceeds *Joint* on all perturbations.

3.3 Analysis

Ablation Analysis To investigate the effectiveness of our proposed strategies, we conduct the ablated variants: (1) ablated LLM power: without LLM powered auxiliary information (w/o aux) and without dynamical cooperation (w/o coop); (2) ablated discriminability driven: without discriminability driven feature learning (w/o disc), without enhancement of more discriminative features (w/o more), and without suppression of less discriminative features (w/o less); (3) ablated historical replay: without memories (w/o mem) and without feature alignment (w/o align).

As Tab. 3 shows, the performance of the variants drops to different degrees at most moments, and all drop significantly at the T_9 moment. These results prove that the proposed strategies are effective for continual toxicity detection against evolving perturbed text. Especially, the LLM powered enriching is helpful since the provided extra information enriches the insights of toxicity judgment.

Analysis of the Memory Sample Number To study the performance tendency under different numbers of memory samples, we experiment with varying numbers to observe the performance

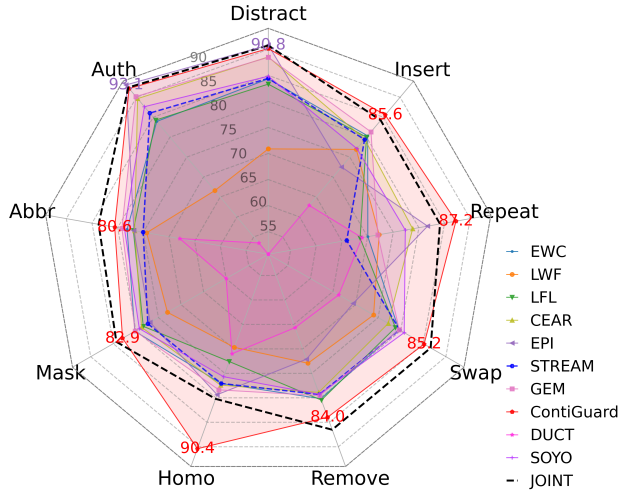


Figure 3: Results on different types of perturbed text at T_9. The numbers denote the best accuracy for each type.

Table 3: Ablation results at different moments.

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9
ContiGuard	87.17	89.27	86.20	86.42	85.41	86.29	85.41	84.92	**86.47**
w/o aux	83.11	84.63	80.52	81.03	80.03	81.85	82.12	80.06	81.97
w/o coop	87.42	88.88	85.62	85.71	85.90	87.16	85.63	86.22	85.96
w/o disc	86.20	87.71	84.08	85.63	86.18	85.30	84.49	84.74	85.71
w/o more	86.36	88.84	85.61	86.07	86.22	85.78	84.75	84.59	85.89
w/o less	86.78	88.86	86.49	86.11	86.17	85.59	84.44	84.06	86.25
w/o mem	87.17	88.86	83.22	85.99	84.99	84.61	83.79	83.84	84.67
w/o align	87.17	88.81	85.57	85.73	86.42	86.33	84.39	83.08	84.87

changes as Fig. 4 shows. We find that under varying numbers, the performance change trend of toxicity detection is basically similar over time. In addition, as the number increases, the accuracies overall increase, and the performance degradation becomes relatively gentle. For example, when the number is 0, the performance decline trend is significant, while when it is 8, the decline trend becomes slower. We believe this is because memory samples can replay the detection capability to mitigate the forgetting.

Retention Rate of Discriminative Features To examine the effectiveness of discriminability driven feature learning and historical capability replay, we analyze the retention rate changing of discriminative features after removing the two strategies. As Fig. 5 shows, the retention rates of discriminative features are all above 80% over the moments in ContiGuard. However, when we ablate the two strategies, the retention rates decline to different degrees. Especially, the retention rate for perturbed text appearing at the T_1 moment significantly drops at the T_9 moment (see the first column), demonstrating that ContiGuard enables the detector to retain most of the learned discriminative features critical for toxicity detection, which mitigates the issue of capability forgetting.

Analysis of Contribution Measurement To explore different attribution methods, we compare integrated gradient (IG) and

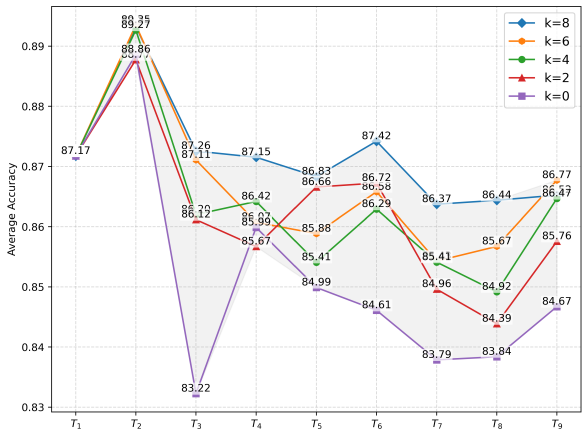


Figure 4: Results of different memory sample numbers.

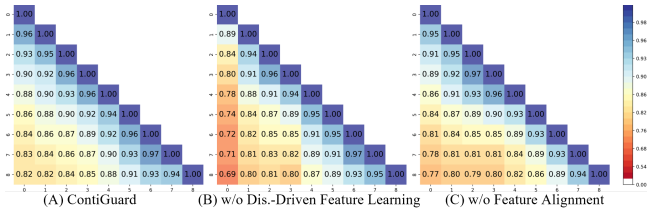


Figure 5: Retention rate of critical features, which denotes the proportion of historical critical features in current critical features. Each column shows the changes over time.

widely used SHAP [39], which resample output changes to estimate feature contribution. As Tab. 4 shows, IG outperforms SHAP and we attribute it to the distinct responses of the two methods to subtle feature contribution changes induced by different perturbation manipulations: IG leverages gradients to capture such subtle changes sensitively, whereas SHAP resamples on the output for estimation, and this may attenuate the contribution of these subtle changes.

Table 4: Results of different contribution measuring methods.

Methods	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9
IG	87.17	89.27	86.20	86.42	85.41	86.29	85.41	84.92	86.47
SHAP	86.73	89.33	84.86	85.04	86.30	85.96	84.15	85.44	85.94

Application Evaluation To evaluate ContiGuard's practical application, we test CL methods on NoisyHate, as Fig. 6 shows. NHbest is the original best detector (67.2) [58] on NoisyHate. We observe that CL methods mostly outperform NHbest, and ContiGuard exhibits the best performance and excellent cross-dataset adaptability. Furthermore, considering users' demand for costs, efficiency, and performance, we analyze from two perspectives: (1). Training with sampled datasets: we conduct training using sampled training data, with sampling proportions ranging from 20% to 100%. This analysis aims to quantify how the detector performance changes as training costs are lowered by reducing data usage. (2). Testing without LLM: we test the detector under a setting where

LLM is excluded (referred to as ContiGuard$_{Eco}$). This setup is designed to explore how performance shifts when detection speed is enhanced by removing LLM during real application.

As Fig. 7 shows, a reduction in training data volume leads to a significant decrease in the training cost required per type of perturbation. For instance, when only using 20% of training data, the training time per type of perturbation is reduced to approximately ***0.15 hours*** (9 minutes), and ContiGuard still performs well on both the DynEscape and NoisyHate datasets. Furthermore, ContiGuard$_{Eco}$ delivers excellent performance on NoisyHate, with a detection speed about ***220 items/s***. This confirms ContiGuard$_{Eco}$ can balance detection performance and efficiency. Notably, we observe a trend: when training data is reduced to around 20%–60% of the original data volume, both ContiGuard and ContiGuard$_{Eco}$ show better performance on NoisyHate. We attribute this to differences in the perturbation characteristics of the datasets: DynEscape considers the situation of extreme perturbations, whereas NoisyHate is collected in the wild and owns milder perturbations, so less modeling on DynEscape makes it easier to generalize to NoisyHate. In addition, the consideration of extreme perturbations also enables ContiGuard trained on DynEscape to have a greater generalization margin when applied to real-world data like NoisyHate. For example, ContiGuard's performances on NoisyHate are all higher than those on DynEscape across proportions. In conclusion, users can tailor ContiGuard to their needs by adjusting two aspects: the volume of training data and whether to exploit LLM during testing, which allows a trade-off among cost, efficiency, and performance.

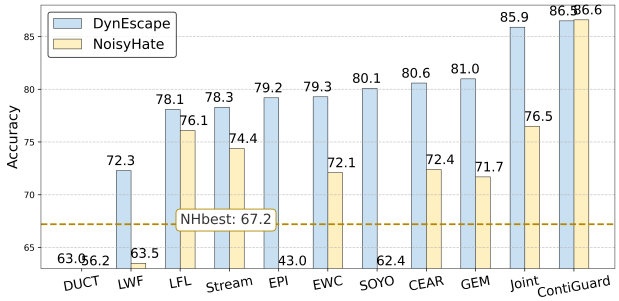


Figure 6: Results on DynEscape and NoisyHate.

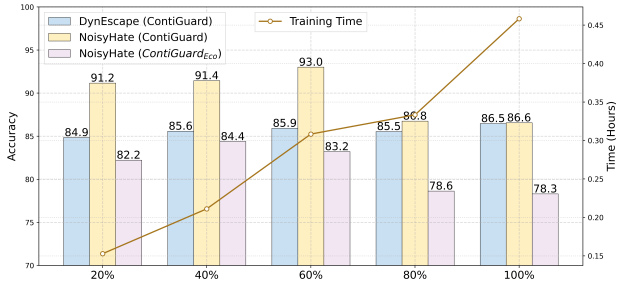


Figure 7: Results of different training data sampling rates.

Case Study To examine the impact of LLM, we compare ContiGuard and ContiGuard$_{Eco}$ (excluding LLM) via cases in Fig. 8.

Case (1) firstly appears at the historical moment and is obfuscated by homoglyph substitution. ContiGuard$_{Eco}$ and ContiGuard all identify the toxicity. However, when the last moment comes, after being trained on the text perturbed by character insertion, ContiGuard$_{Eco}$ fails to recognize the toxicity in case (1) since the disparity between perturbations breaks the historically learned detection ability. In contrast, ContiGuard enriches the perturbed text with the insights of aggressive attitude and the strong profanity 'fuck' captured with LLM and learns from the insights to mitigate the ability breaking so that it still correctly detects the toxic case.

Case (2) firstly appears at the last moment and is perturbed by character insertion. ContiGuard$_{Eco}$ cannot associate and unify toxicity patterns in this case with the historically learned ones (e.g., perturbed 'fuck') since the difference across perturbations obfuscates the detection. Hence, it fails to recognize the toxicity. ContiGuard can capture the insights of the aggressive attitude and the toxic term 'fuck', which helps ContiGuard to perceive the similar toxic patterns from the insights in cases (1) and (2), enabling ContiGuard to successfully identify the toxicity in case (2).

Overall, ContiGuard uses LLM's analysis to enrich perturbed text to improve comprehension and to bridge different perturbations through shared insights derived from the analysis, which mitigates the forgetting of toxicity detection ability.

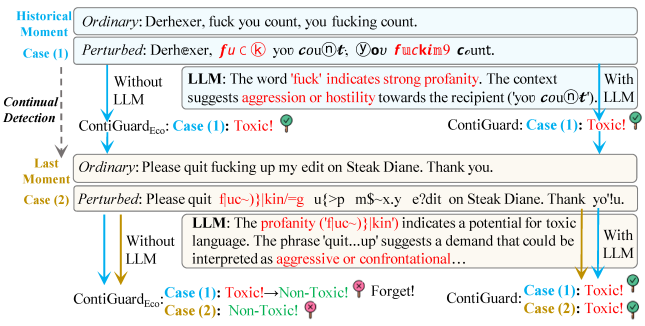


Figure 8: Case study for examining the impact of the LLM.

4 Conclusion

In this work, we first highlight the crucial yet overlooked challenge of continual toxicity detection against evolving evasive perturbed text drafted by malicious users, and we propose a novel ContiGuard framework exploiting strategies of LLM powered semantic enriching, discriminability driven feature learning, and historical capability replay to enable the detector to perform robust continual toxicity detection against evolving evasive perturbed text. Extensive experimental results show the superior performance of ContiGuard over the existing detectors and continual methods.

Acknowledgments

This work was supported by the grant from the National Natural Science Foundation of China (NSFC) project (Grant No. 62576256), the grant from Zhongguancun Academy (Grant No. 20240302), and the grant from the Key Laboratory of Computing Power Network and Information Security, Ministry of Education (Grant No. 2024ZD027).

References

[1] Dmitriy Bespalov, Sourav Bhabesh, Yi Xiang, Liutong Zhou, and Yanjun Qi. 2023. Towards Building a Robust Toxicity Predictor. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*. 581–598.

[2] Cristina Bosco, Viviana Patti, Simona Frenda, Alessandra Teresa Cignarella, Marinella Paciello, and Francesca D'Errico. 2023. Detecting racial stereotypes: An Italian social media corpus where psychology meets NLP. *Information Processing & Management* 60, 1 (2023), 103118.

[3] Portia Cooper, Mihai Surdeanu, and Eduardo Blanco. 2023. Hiding in plain sight: Tweets with hate speech masked by homoglyphs. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. 2922–2929.

[4] Marta Costa-jussà, Mariano Meglioli, Pierre Andrews, David Dale, Prangthip Hansanti, Elahe Kalbassi, Alexandre Mourachko, Christophe Ropers, and Carleigh Wood. 2024. MuTox: Universal MUltilingual Audio-based TOXicity Dataset and Zero-shot Detector. In *Findings of the Association for Computational Linguistics ACL 2024*. 5725–5734.

[5] Zahra Delbari, Nafise Sadat Moosavi, and Mohammad Taher Pilehvar. 2024. Spanning the spectrum of hatred detection: a persian multi-label hate speech dataset with annotator rationales. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 17889–17897.

[6] Daryna Dementieva, Daniil Moskovskiy, Nikolay Babakov, Abinew Ali Ayele, Naquee Rizwan, Frolian Schneider, Xintog Wang, Seid Muhie Yimam, Dmitry Ustalov, Elisei Stakovskii, et al. 2024. Overview of the multilingual text detoxification task at pan 2024. *Working Notes of CLEF* (2024).

[7] Lucas Dixon, John Li, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2018. Measuring and mitigating unintended bias in text classification. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*. 67–73.

[8] Jiahua Dong, Lixu Wang, Zhen Fang, Gan Sun, Shichao Xu, Xiao Wang, and Qi Zhu. 2022. Federated class-incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10164–10173.

[9] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. 2020. PODNet: Pooled Outputs Distillation for Small-Tasks Incremental Learning. In *European Conference on Computer Vision*. 86–102.

[10] Chris Emmery, Ákos Kádár, Grzegorz Chrupała, and Walter Daelemans. 2022. Cyberbullying Classifiers are Sensitive to Model-Agnostic Perturbations. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*. 2976–2988.

[11] Paula Fortuna and Sérgio Nunes. 2018. A survey on automatic detection of hate speech in text. *ACM Computing Surveys (CSUR)* 51, 4 (2018), 1–30.

[12] Zhiling Fu, Zhe Wang, Chengwei Yu, Xinlei Xu, and Dongdong Li. 2024. Double Confidence Calibration Focused Distillation for Task-Incremental Learning. *IEEE Transactions on Neural Networks and Learning Systems* (2024), 1–14.

[13] Qiang Gao, Xiaojun Shan, Yuchen Zhang, and Fan Zhou. 2023. Enhancing knowledge transfer for task incremental learning with data-free subnetwork. *Advances in Neural Information Processing Systems* 36 (2023), 68471–68484.

[14] Prachi Garg, Rohit Saluja, Vineeth N Balasubramanian, Chetan Arora, Anbumani Subramanian, and CV Jawahar. 2022. Multi-domain incremental learning for semantic segmentation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 761–771.

[15] Tanmay Garg, Sarah Masud, Tharun Suresh, and Tanmoy Chakraborty. 2023. Handling bias in toxic speech detection: A survey. *Comput. Surveys* 55, 13s (2023), 1–32.

[16] Vaishali U Gongane, Mousami V Munot, and Alwin D Anuse. 2022. Detection and moderation of detrimental content on social media platforms: current status and future directions. *Social Network Analysis and Mining* 12, 1 (2022), 129.

[17] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2013. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211* (2013).

[18] Janosch Haber, Bertie Vidgen, Matthew Chapman, Vibhor Agarwal, Roy Ka-Wei Lee, Yong Keong Yap, and Paul Röttger. 2023. Improving the detection of multilingual online attacks with rich social media data from singapore. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 12705–12721.

[19] Laura Hanu and Unitary team. 2020. Detoxify. Github. https://github.com/unitaryai/detoxify.

[20] Thomas Hartvigsen, Saadia Gabriel, Hamid Palangi, Maarten Sap, Dipankar Ray, and Ece Kamar. 2022. ToxiGen: A Large-Scale Machine-Generated Dataset for Adversarial and Implicit Hate Speech Detection. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 3309–3326.

[21] Jiangpeng He. 2024. Gradient reweighting: Towards imbalanced class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 16668–16677.

[22] Heechul Jung, Jeongwoo Ju, Minju Jung, and Junmo Kim. 2016. Less-forgetting learning in deep neural networks. *arXiv preprint arXiv:1607.00122* (2016).

[23] Menelaos Kanakis, David Bruggemann, Suman Saha, Stamatios Georgoulis, Anton Obukhov, and Luc Van Gool. 2020. Reparameterizing Convolutions for Incremental Multi-Task Learning Without Task Interference. In *European Conference on Computer Vision*. 689–707.

[24] Emad Kebriaei, Ali Homayouni, Roghayeh Faraji, Armita Razavi, Azadeh Shakery, Heshaam Faili, and Yadollah Yaghoobzadeh. 2024. Persian offensive language detection. *Machine Learning* 113, 7 (2024), 4359–4379.

[25] Hannah Kirk, Bertie Vidgen, Paul Röttger, Tristan Thrush, and Scott Hale. 2022. Hatemoji: A Test Suite and Adversarially-Generated Dataset for Benchmarking and Detecting Emoji-Based Hate. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 1352–1368.

[26] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences* 114, 13 (2017), 3521–3526.

[27] Ian Kivlichan, Jeffrey Sorensen, Lucy Vasserman Julia Elliott, Martin Görner, and Phil Culliton. 2020. Jigsaw Multilingual Toxic Comment Classification.

[28] Keita Kurita, Anna Belova, and Antonios Anastasopoulos. 2019. Towards robust toxic content classification. *arXiv preprint arXiv:1912.06872* (2019).

[29] Nineli Lashkarashvili and Magda Tsintsadze. 2022. Toxicity detection in online Georgian discussions. *International Journal of Information Management Data Insights* 2, 1 (2022), 100062.

[30] Thai Le, Jooyoung Lee, Kevin Yen, Yifan Hu, and Dongwon Lee. 2022. Perturbations in the Wild: Leveraging Human-Written Text Perturbations for Realistic Adversarial Attack and Defense. In *Findings of the Association for Computational Linguistics: ACL 2022*. 2953–2965.

[31] Daehee Lee, Minjong Yoo, Woo Kyung Kim, Wonje Choi, and Honguk Woo. 2024. Incremental learning of retrievable skills for efficient continual task adaptation. *Advances in Neural Information Processing Systems* 37 (2024), 17286–17312.

[32] Alyssa Lees, Vinh Q Tran, Yi Tay, Jeffrey Sorensen, Jai Gupta, Donald Metzler, and Lucy Vasserman. 2022. A new generation of perspective api: Efficient multilingual character-level transformers. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*. 3197–3207.

[33] Shuo Li, Fang Liu, Licheng Jiao, Lingling Li, Puhua Chen, Xu Liu, and Wenping Ma. 2025. Prompt-Based Concept Learning for Few-Shot Class-Incremental Learning. *IEEE Transactions on Circuits and Systems for Video Technology* 35, 5 (2025), 4991–5005. doi:10.1109/TCSVT.2025.3525545

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

[35] Ruiqi Liu, Boyu Diao, Libo Huang, Zijia An, Zhulin An, and Yongjun Xu. 2024. Continual learning in the frequency domain. *Advances in Neural Information Processing Systems* 37 (2024), 85389–85411.

[36] Varvara Logacheva, Daryna Dementieva, Sergey Ustyantsev, Daniil Moskovskiy, David Dale, Irina Krotova, Nikita Semenov, and Alexander Panchenko. 2022. Paradetox: Detoxification with parallel data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 6804–6818.

[37] David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. *Advances in neural information processing systems* 30 (2017).

[38] Junyu Lu, Bo Xu, Xiaokun Zhang, Changrong Min, Liang Yang, and Hongfei Lin. 2023. Facilitating Fine-grained Detection of Chinese Toxic Language: Hierarchical Taxonomy, Resources, and Benchmarks. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 16235–16250.

[39] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems* 30 (2017).

[40] Todor Markov, Chong Zhang, Sandhini Agarwal, Florentine Eloundou Nekoul, Theodore Lee, Steven Adler, Angela Jiang, and Lilian Weng. 2023. A holistic approach to undesired content detection in the real world. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 15009–15018.

[41] Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost Van De Weijer. 2022. Class-incremental learning: survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 5 (2022), 5513–5533.

[42] Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. 2021. Hatexplain: A benchmark dataset for explainable hate speech detection. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 14867–14875.

[43] Ryszard S Michalski, Igor Mozetic, Jiarong Hong, and Nada Lavrac. 1986. The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. In *Proceedings of the Fifth AAAI National Conference on Artificial Intelligence*. 1041–1045.

[44] M Jehanzeb Mirza, Marc Masana, Horst Possegger, and Horst Bischof. 2022. An efficient domain-incremental learning approach to drive in all weather conditions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 3001–3011.

[45] Sudhanshu Mittal, Silvio Galesso, and Thomas Brox. 2021. Essentials for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer*

Vision and Pattern Recognition. 3513–3522.
[46] Guy Oren and Lior Wolf. 2021. In defense of the learning without forgetting for task incremental learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2209–2218.
[47] Ronghao Pan, José Antonio García-Díaz, Pedro José Vivancos-Vicente, and Rafael Valencia-García. 2024. UMUTeam at SemEval-2024 Task 8: Combining Transformers and Syntax Features for Machine-Generated Text Detection. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*. 697–702.
[48] Maarten Sap, Saadia Gabriel, Lianhui Qin, Dan Jurafsky, Noah A Smith, and Yejin Choi. 2020. Social Bias Frames: Reasoning about Social and Power Implications of Language. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 5477–5490.
[49] Simon Sinek. 2009. *Start with why: How great leaders inspire everyone to take action*. Penguin.
[50] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International conference on machine learning*. 3319–3328.
[51] Daniel Stanley Tan, Yong-Xiang Lin, and Kai-Lung Hua. 2020. Incremental learning of multi-domain image-to-image translations. *IEEE Transactions on Circuits and Systems for Video Technology* 31, 4 (2020), 1526–1539.
[52] Gido M Van de Ven, Tinne Tuytelaars, and Andreas S Tolias. 2022. Three types of incremental learning. *Nature Machine Intelligence* 4, 12 (2022), 1185–1197.
[53] Keyao Wang, Guosheng Zhang, Haixiao Yue, Ajian Liu, Gang Zhang, Haocheng Feng, Junyu Han, Errui Ding, and Jingdong Wang. 2024. Multi-domain incremental learning for face presentation attack detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 5499–5507.
[54] Qiang Wang, Xiang Song, Yuhang He, Jizhou Han, Chenhao Ding, Xinyuan Gao, and Yihong Gong. 2025. Boosting Domain Incremental Learning: Selecting the Optimal Parameters is All You Need. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 4839–4849.
[55] Zhicheng Wang, Yufang Liu, Tao Ji, Xiaoling Wang, Yuanbin Wu, Congcong Jiang, Ye Chao, Zhencong Han, Ling Wang, Xu Shao, et al. 2023. Rehearsal-free continual language learning via efficient parameter isolation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 10933–10946.
[56] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. 2019. Large scale incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 374–382.
[57] Yufei Yang, Mingai Li, and Fubiao Huang. 2025. EEG-DTIL: An EEG-based dynamic task-incremental learning method for decoding ADL-oriented motor imagery pairs. *Expert Systems with Applications* (2025), 128927.
[58] Yiran Ye, Thai Le, and Dongwon Lee. 2025. NoisyHate: Mining Online Human-Written Perturbations for Realistic Robustness Benchmarking of Content Moderation Models. In *Proceedings of the International AAAI Conference on Web and Social Media*, Vol. 19. 2603–2612.
[59] Seunguk Yu, Juhwan Choi, and Youngbin Kim. 2024. Don't be a Fool: Pooling Strategies in Offensive Language Detection from User-Intended Adversarial Attacks. In *Findings of the Association for Computational Linguistics: NAACL 2024*. 3456–3467.
[60] Jiang Zhang, Qiong Wu, Yiming Xu, Cheng Cao, Zheng Du, and Konstantinos Psounis. 2024. Efficient toxic content detection by bootstrapping and distilling large language models. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 38. 21779–21787.
[61] Junting Zhang, Jie Zhang, Shalini Ghosh, Dawei Li, Serafettin Tasci, Larry Heck, Heming Zhang, and C-C Jay Kuo. 2020. Class-incremental learning via deep model consolidation. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*. 1131–1140.
[62] Min Zhang, Jianfeng He, Taoran Ji, and Chang-Tien Lu. 2024. Don't Go To Extremes: Revealing the Excessive Sensitivity and Calibration Limitations of LLMs in Implicit Hate Speech Detection. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 12073–12086.
[63] Zhehao Zhang, Jiaao Chen, and Diyi Yang. 2023. Mitigating biases in hate speech detection from A causal perspective. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. 6610–6625.
[64] Wenzheng Zhao, Yuanning Cui, and Wei Hu. 2023. Improving Continual Relation Extraction by Distinguishing Analogous Semantics. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1162–1175.
[65] Da-Wei Zhou, Zi-Wen Cai, Han-Jia Ye, Lijun Zhang, and De-Chuan Zhan. 2025. Dual consolidation for pre-trained model-based domain-incremental learning. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 20547–20557.
[66] Fei Zhu, Zhen Cheng, Xu-yao Zhang, and Cheng-lin Liu. 2021. Class-incremental learning via dual augmentation. *Advances in neural information processing systems* 34 (2021), 14306–14318.

A Related Work

A.1 Toxicity Detection

Various methods are proposed for toxicity detection, which can be primarily categorized into two types: ordinary text-focused and perturbations-augmented.

Ordinary text-focused methods are targeted at improving the performance on text without perturbations [2, 24, 40, 47, 60, 62]. Many datasets across different languages have been constructed for this task [5, 15, 29]. Most of these datasets are curated from social platforms like X, Reddit, and Zhihu [18, 38, 48] with lots of manual efforts for annotation, and a few of them are generated by LLMs like GPT3 [20]. The ordinary text-focused methods rely on lexical clues in normal text and are vulnerable to even simple attacks.

Perturbation-augmented methods enhance the detectors' robustness by incorporating perturbed text. They [1, 30, 58] may manipulate characters, words, or sentences in the text, e.g., character swap and/or substitution, homoglyph/homophone substitution, decomposition, near-neighbor word replacement, and distract injection [3, 10, 28, 59]. The existing perturbation-augmented methods statically employ fixed specific types of perturbations to enhance the detectors' robustness, but they struggle to dynamically deal with increasing crafted perturbations over time.

The existing methods do not update the detector's capability so they cannot deal with the perturbed text that appears increasingly. In contrast, we collect 9 types of perturbations and conduct continual learning for the detector when these perturbations emerge incrementally over time, enabling the detector to continually update its capability against increasing perturbations.

A.2 Continual Learning

Continual learning aims to continually learn from an incrementally appearing data stream [17]. There are three incremental learning scenarios, i.e., domain-, class-, and task-incremental learning [52].

In domain-incremental learning, all tasks have the same label space but different data domains [14, 43, 44, 51, 53], e.g., [65] proposes dual consolidation to create a representation space suitable for multiple domains incrementally and to merge the backbone of different stages, accommodating all seen domains throughout the learning. [54] introduces a gaussian mixture compressor and domain feature re-sampler to store and balance prior domain data, and proposes a multi-level domain feature fusion network to extract domain feature, boosting the domain-incremental learning ability.

In task-incremental learning, the tasks are increasingly appearing and being solved [9, 12, 13, 23, 46]. Recently, [31] introduces an adapter-based continual imitation learning framework to address the limitation of knowledge sharing by incrementally learning shareable skills from demonstrations, enabling sample-efficient task adaptation using the skills. [57] employs wavelet packet transform to extract global-view spatial features via the regularized common spatial pattern and captures local-view spatial features via tangent space mapping from the Riemannian space to improve daily living of patients with strokes by task incremental learning.

In class-incremental learning, the number of categories is continually growing [8, 41, 45, 56, 61, 66]. For instance, [35] employs wavelet transform to map the image into the frequency domain and balances the reusability and interference of output features based on

the frequency domain similarity of the classes to mitigate the forgetting of previously acquired knowledge. [21] proposes to re-weight the gradients towards balanced optimization and unbiased classifier learning to address skewed gradient updates with biased weights. [33] proposes to generalize conceptual knowledge learned from old classes to new classes by simulating human learning capability.

The continual toxicity detection problem falls under domain-incremental learning, where ordinary text is perturbed over time to be distributed in varying perturbation domains for evading detectors. To the best of our knowledge, there are no methods or datasets focusing on the continual toxicity detection problem, and our work is the first to explore the problem.

B Dataset Construction

Most existing toxicity datasets do not consider the perturbed text [4, 20, 42, 48]. Though a few datasets contain perturbed samples, there are only limited types of perturbed text produced by monotonous perturbation operations. Moreover, these datasets mix all types of perturbations together [3, 25, 58], by which we cannot examine the detectors' adaptability to changing perturbations. To address this issue, we construct *DynEscape*, a perturbation-wise toxicity dataset covering 9 types of perturbations through three stages.

Stage 1. ***Data Cleaning and Toxicity-relevant Word Selection***. This stage aims to remove noises in raw data and find toxicity-relevant words so that we can perturb them to confuse detectors.

In this stage, we choose Jigsaw [27] dataset as raw data since it contains lots of toxic text (223k). However, Jigsaw cannot be directly used for our purpose since it contains many noises. More importantly, perturbing toxicity-irrelevant words will have weak impact on the detector. Hence we perform data cleaning and toxicity-relevant word selection as the preprocessing.

(1) Data Cleaning. We clean noises in the Jigsaw, i.e., unknown words (identified by *spellchecker* tool), private information (emails, user ids), and meaning-less text (repeated sentences less than 5 words). We also find that Jigsaw is seriously imbalanced between non-toxic and toxic samples, with a ratio approaching 10:1. Hence we re-sample the toxic and non-toxic samples in 1:1 ratio, and obtain 20k/20k toxic/non-toxic samples for perturbing.

(2) Toxicity-relevant Word Selection. We select words significantly influencing toxicity recognition. By perturbing these words, we wish to bypass detectors. Note that besides toxic words, many non-toxic words are also relevant to toxicity. Hence we propose four strategies to select toxicity-relevant word:

Online Toxic Words. We collect online toxic words from Google, including bad words and the banned swear words (2.9K words).

Words Identified as Toxic. We employ PerspectiveAPI [32] to detect the toxicity of each word in the dataset. We collect the word identified as toxic (1.1K words).

Words Increasing Toxicity Scores. We compute the expectation of decreased toxicity score (supplied by PerspectiveAPI) for each word to examine its contribution to the sample's toxicity. Specifically, given the word w_i and the sentences T_{w_i} it appears, we compute the expectation E_{w_i} as:

$$E_{w_i} = \frac{1}{|T_{w_i}|} \sum_{t_j \in T_{w_i}} [s(t_j) - s(t_j/w_i)], \tag{14}$$

where $s(t_j)$ and $s(t_j/w_i)$ denote the toxicity score of the text $t_j \in T_{w_i}$ before and after removing w_i, and $s(t_j) - s(t_j/w_i)$ denotes the decreased toxicity score after removing w_i. The expectation E_{w_i} represents the contribution of the word w_i to the toxicity of text. We select the words whose expectations are greater than 0 (1.5K words).

Spuriously Correlated Words. Some words often appear with toxic label so that detectors tend to identify the text including these words as toxic, i.e., the words are spuriously correlated to labels [15]. We compute mutation information to get such words [63].

$$MI = \frac{p(w_i, c)}{p(w_i, \cdot)p(\cdot, c)}, \tag{15}$$

where $p(w_i, \cdot)$ and $p(\cdot, c)$ are the marginal distribution of the word w_i and the label c, and $p(w_i, c)$ is the joint distribution of w_i and c. The larger MI, the stronger the spurious correlation between w_i and c. Based on multiple manual checks and previous empirical setting [63], we set the spurious correlation threshold as the sum of the mean and standard deviation of MI (1.1K words).

With these strategies, we totally collect 5.6K toxicity-relevant words after de-duplicating.

Stage 2. *Perturbation Tool Creation.* This stage aims to provide the definitions of perturbations and then create a perturbation tool, which will be used as the attacker to perturb samples.

In this stage, we create a perturbation tool that could perform 9 types of perturbations.

(1) Perturbation Preparation We prepare necessary material for perturbation, including 2.7K homoglyphs, 80.6K abbrs/slangs, and 3.3K roles.

(2) Perturbation Patterns We then define perturbation patterns as follows (Tab. 5 shows the examples).

Character-level Perturbation. **Insert** randomly adds special characters (from Python *string* package) into the text. **Remove**/**Repeat** randomly deletes/duplicates characters. **Swap** randomly exchanges the order of characters. **Homoglyph** randomly replaces characters with visually similar characters (using prepared 2.7K homoglyphs).

Word/Phrase-level Perturbation. **Maskword** randomly obscures characters in the selected words with special characters. **Abbreviation** employs abbreviations/slangs to replace the associated words or phrases (using prepared 80.6K abbrs/slangs).

Sentence-level Perturbation. **Distract** adds the prefix consisting of extra random non-toxic words before the text. **Authorization** firstly instructs the LLMs to generate a self-introduction by acting as authoritative experts (using prepared 3.3K roles). Then, it adds the introduction before text to confuse detectors. This is an exploratory perturbation for evading detectors since the emergence of LLMs.

Based on the definitions, we implement 9 perturbations to create a perturbation tool that provides automated perturbation patterns.

We finally conduct a human verification by manually checking 50 perturbed samples for each perturbation type (450 in total). Results show that all perturbed samples conform to the definitions, which proves our tool's reliability.

Stage 3. *Adversarial Attack and Quality Assessment.* This stage aims to attack detectors using the samples perturbed by our tool and to select the challenging perturbed samples to compose *DynEscape*.

In this stage, we utilize the perturbation tool as an attacker to disturb clean samples for evading detectors (e.g., safety-aligned LLama3). The perturbed samples bypassing detectors are selected

Table 5: Examples of different perturbations.

Perturbations	Examples
Insert	moron→mo*ro#n; idiot→idi+ot
Remove (Mv)	moron→moro; idiot→idot
Repeat	moron→ mooron; idiot→ iddiot
Swap	fool→ folo; idiot→ idoit
Homoglyph (Homo)	idiot→id10t; fool→ f001
Maskword (Mask)	fool→f**l; idiot→i*iot
Abbreviation (Abbr)	fuck→ 4Q; bite me→ BTM
Distract (Dis)	[text]→[apple earth...];[text]
Authorization (Auth)	[text]→[I am a scientist...];[text]

to compose our *DynEscape*. Note that to prevent detectors from taking shortcuts by judging whether the text has been disturbed, non-toxic samples are also perturbed in the same way as toxic ones.

We collect all successful samples to evade detectors, and they are reallocated with 4K non-overlapped samples per perturbation. We then split them into train/valid/test subsets with a ratio of 6:1:3 (2584/430/1294). Finally, our perturbation-wise dataset *DynEscape* consists of 38K ((2584+430+1294)*9) challenging perturbed samples.

To ensure the quality of our dataset, we randomly select 50 samples per perturbation and employ three master students to evaluate the semantic consistency before/after perturbations on a scale of 0 to 1. We get a score of $0.88_{\pm 0.07}$ averaged over all samples and more than two masters assign a score exceeding 0.5 in 99% of the samples, indicating a high consistency of our dataset.

During data collection, we employ online toxic words collected by Google, which are available by 'https://github.com/coffee-and-fun/google-profanity-words/tree/main', and we set perturbation rate to 20%, i.e., 20% of the words in the original text are perturbed. We utilize punctuation as special characters from the Python package '*string*', and we collect homoglyphs from the '*homoglyphs*' python tool to replace the characters. In addition, we collect the abbreviations/slangs from the online resources, including 'onlineslangdictionary.com','slang.net', 'www.acronymfinder.com' and 'acronymsandslang.com'.

C Experimental

C.1 Details of Experimental Implementation

In all experiments, we use the bert-base-uncased model (110M) as the encoder of the detector to encode the sentences, where the maximum sequence length is set to 360. The parameters of the models are updated using the AdamW optimizer with a learning rate of 2e-5. The random seed for all experiments is set to 0, and training is conducted with an early stopping strategy. All experiments are conducted on the A800 GPU. Considering the strong reasoning ability, we employ gpt-4o-mini to capture the auxiliary information.

C.2 Introduction of Existing Methods

LFL [22] reduces knowledge forgetting by regularizing the parameters between old and new classifiers.

EWC [26] uses the Fisher matrix to assess parameter importance and applies weighted regularization based on that importance.

LWF [34] transfers knowledge by having the old model generate pseudo-labels for the new model.

GEM [37] stores gradient information from past tasks and constrains the gradient direction of the current task to prevent it from conflicting with the gradient directions of old tasks.

EPI [55] trains task-specific prefix parameters for each task and identifies the task IDs of test samples to select the appropriate prefix parameters for classification.

CEAR [64] learns memory-insensitive prototypes and uses memory augmentation to reduce overfitting and enhance performance.

DUCT [65] proposes dual consolidation to construct a unified representation space applicable to multiple domains and integrate backbones from different moments to mitigate the forgetting.

SOYO [54] introduces a gaussian mixture compressor and feature re-sampler to store balance prior domain data, and proposes a multi-level feature fusion network to enhance domain feature extraction, boosting the domain-incremental learning ability.

C.3 Supplementary Experiments

The performance of directly using gpt-4o-mini We also explore how 4o-mini performs on perturbed text, as Tab. 6 shows. We find that 4o-mini performs well on ordinary text (DynEscape (ord.)), but its performance still degrades to 69.50 with a drop of 19% when dealing with perturbed text (DynEscape). In addition, compared to DynEscape, 4o-mini performs better on NoisyHate due to minor perturbations within NoisyHate, but it was still worse than ContiGuard and even ContiGuard$_{Eco}$. We believe that using 4o-mini's reasoning capability to extract auxiliary information to enhance the detector is a good alternative approach compared to directly using it for toxicity detection against perturbed text.

Table 6: Result of 4o-mini. DynEscape (ord.) represents the original ordinary sample corresponding to DynEscape.

	DynEscape (ord.)	DynEscape	NoisyHate
4o-mini	85.81	$69.50_{19\%\downarrow}$	80.06