Information Processing and Management 63 (2026) 104470

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/ipm

Developing continuous toxicity detection against increasing types of perturbed toxic text

Hankun Kang [a], Jianhao Chen [a,b], Yongqi Li [a], Xin Miao [a], Mayi Xu [a], Shen Zhou [a], Ming Zhong [a], Yuanyuan Zhu [a], Tieyun Qian [a,b,*]

[a] School of Computer Science, Wuhan University, Wuhan, 430072, Hubei, China
[b] Zhongguancun Academy, Beijing, China

ARTICLE INFO

Keywords:
Toxicity detection
Domain incremental learning
Social media

ABSTRACT

Warning: The article contains offensive material due to the nature of work.

Toxicity detection is crucial for maintaining the peace of society. Current detectors perform well on normal toxic contents or those disturbed by specific types of perturbations. However, malicious users tend to constantly create new perturbations for fooling the detectors. For example, some users may circumvent the detectors by employing large language models (LLMs) to generate convincing prefixes like 'I am a scientist...' before toxic text. Existing detectors are vulnerable to these evolving perturbations because either detectors or their employed datasets overlook the changing nature of perturbations.

To fill this gap, we develop continuous toxicity detection against increasing types of perturbed toxic text for the first time. Specifically, we construct a new challenging perturbation-wise dataset (38K) disturbed by 9 types of perturbations in an adversarial attack way. With the dataset, we systematically validate the vulnerability of current methods for the emerging perturbations via zero-shot and fine-tuned cross-perturbation functional detection. Furthermore, we present a domain incremental learning method to enhance the detector's ability to constantly emerging types of perturbed toxic text. Our code and dataset will be publicly available, by which we wish to inspire more research for the security-relevant communities.

1. Introduction

Online communication plays an essential role in our life, but nowadays it is seriously disturbed by toxic contents. Toxic content means the text is rude or hateful, e.g., cyberbully (Kowalski, 2018; Slonje, Smith, & Frisén, 2013), hate speech (Del Vigna12, Cimino23, Dell'Orletta, Petrocchi, & Tesconi, 2017; Fortuna & Nunes, 2018; Wu et al., 2024), and social bias (Gongane, Munot, & Anuse, 2022; Liang, Wu, Morency, & Salakhutdinov, 2021), which will lead to disharmony and conflicts in society (Dixon, Li, Sorensen, Thain, & Vasserman, 2018; Feldman, Friedler, Moeller, Scheidegger, & Venkatasubramanian, 2015). For protecting the healthy communication environment, toxicity detection methods are proposed to limit the spread of toxic content via pre-detection or post-detection.

In reality, malicious users tend to continually create new types of perturbed toxic text to evade detectors, which makes toxicity detection more challenging. As Fig. 1 A. Problem (a) shows, *iddddiiiot* (by random repeat), 1di0t (by homoglyph), and more new types of perturbations might be continually created during the usage of detectors. Unfortunately, most of existing detectors only

* Corresponding author at: School of Computer Science, Wuhan University, Wuhan, 430072, Hubei, China.
E-mail address: qty@whu.edu.cn (T. Qian).

https://doi.org/10.1016/j.ipm.2025.104470
Received 5 March 2025; Received in revised form 2 September 2025; Accepted 1 November 2025
Available online 7 November 2025

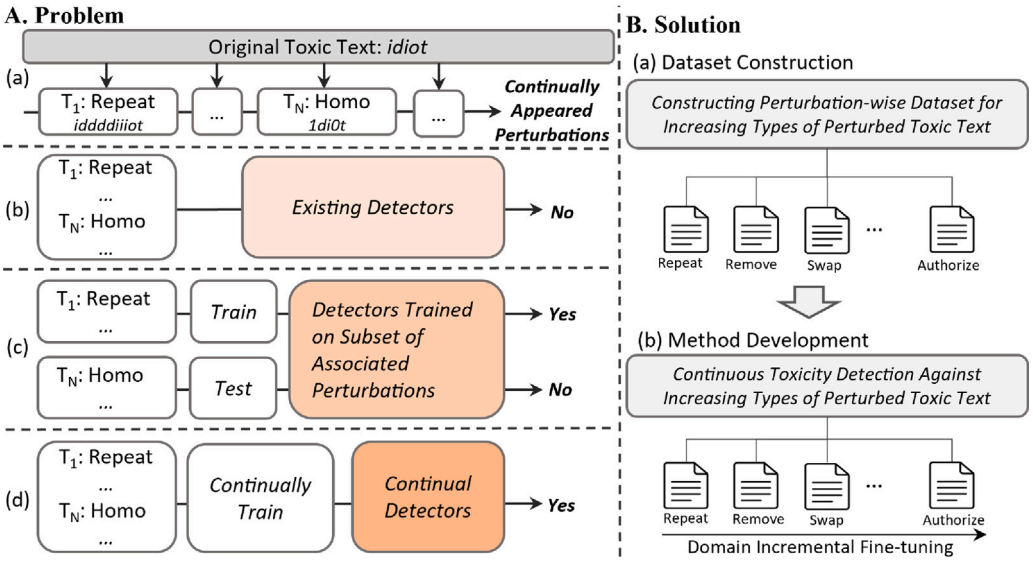

Fig. 1. Comparison of different toxicity detection methods on dynamically changing perturbations, where *Repeat* and *Homo* denote the toxic text is perturbed by repeat and homoglyph perturbation, respectively.

focus on unperturbed text, e.g., *idiot*, such that they struggle to identify the toxicity in perturbed text, as Fig. 1 A. Problem (b) shows.

Though a few methods have paid attention to the perturbed text (Bespalov, Bhabesh, Xiang, Zhou, & Qi, 2023; Emmery, Kádár, Chrupała, & Daelemans, 2022), they only fine-tune detectors on augmented data with fixed types of perturbations. Such detectors can work well on toxic contents generated by the associated perturbation like repeat, but may fail on those by other emerging types of perturbation like homoglyph, as Fig. 1 A. Problem (c) shows.

Hence, it is yet an open problem to detect continually emerged types of perturbed toxic text and there is no research focusing on the evolving nature of perturbations. In this work, we will develop a continuous toxicity detection benchmark to evaluate and improve the capability of detectors to recognize changing types of perturbed toxic text. Despite the necessity and appealing application prospects of our proposed problem, we are facing three severe challenges: (1) lack of a perturbation-wise dataset containing various types of perturbations, (2) lack of a comprehensive investigation of existing methods on this new problem, (3) lack of available solutions for this problem. We will overcome all these challenges step by step.

Firstly, we construct a challenging perturbation-wise toxicity detection dataset named *DynEscape*, i.e., Dynamic Escape, as Fig. 1 B. Solution (a) shows. Specifically, we employ an adversarial attack way to diversely disturb the text according to real-world scenarios, through which we construct a dataset covering 9 types of perturbations. With this dataset, it becomes possible to evaluate or improve the ability of detectors to adapt to continually changing perturbations.

Secondly, we perform not only zero-shot functional testing on existing normal detectors but also cross-perturbation functional testing on detectors fine-tuned with specific types of perturbations, from which we can observe the performance degradation of detectors quantitatively and conclude that current detectors all struggle to identify emerging types of perturbed toxic text.

Thirdly, we view the toxic text with different types of perturbations as the text distributed in different domains. Consequently, the toxicity detection towards adapting to changing types of perturbations can be modeled as the problem of detection with incremental perturbation domain throughout detectors' usage time. By exploiting the domain incremental learning as Fig. 1 B. Solution (b) shows, the detectors are expected to continuously recognize toxic text with either previous or emerging perturbations as Fig. 1 A. Problem (d) shows. Specifically, we present an effective continual toxicity detection approach named *DynDetect*, i.e., Dynamic Detection, which maximizes feature similarity between different domains to preserve previous and new knowledge in different domains.

Our contributions are summarized as follows.

- We highlight a practical but overlooked problem of continuous toxicity detection against constantly increasing types of perturbed toxic text.
- We contribute a benchmark towards the problem, which is consisted of a perturbation-wise toxicity detection dataset and a solution from the perspective of continual learning.
- We evaluate popular detectors and reveal their vulnerability to changing perturbations. We also show the effectiveness of our proposed continual learning baseline detector.

In addition, we organize the sections as follows: Section 2 gives an overview of the toxicity detection task and continual learning methods. Section 3 describes the overall solution for toxicity detection against evolving perturbations, including problem definition

(Section 3.1), dataset construction (Section 3.2), and different detection methods comparison (Section 3.3). Section 4 shows the experimental results for demonstrating the withdrawal of existing detection methods and the effectiveness of our proposed method when they encounter evolving perturbed text. Section 5 briefly discusses the deployment of our proposed detector, concludes our work and looks ahead to future work.

2. Related work

2.1. Toxicity detection

Various methods have been proposed for textual toxicity detection, which can be primarily categorized into two types: performance- and robustness-oriented.

The performance-oriented methods are targeted at improving the performance in terms of metrics like accuracy and F1 (Bosco et al., 2023; Kebriaei et al., 2024; Markov et al., 2023; Pan, García-Díaz, Vivancos-Vicente, & Valencia-García, 2024; Zhang, He, Ji, & Lu, 2024; Zhang, Wu, et al., 2024). Many datasets across different languages have been constructed for this task (Delbari, Moosavi, & Pilehvar, 2024; Garg, Masud, Suresh, & Chakraborty, 2023; Lashkarashvili & Tsintsadze, 2022). Most of these datasets are curated from online platforms like Twitter (now called X), Reddit, and Zhihu (Haber et al., 2023; Lu et al., 2023; Sap et al., 2020) with lots of manual efforts for annotation, and a few of them are generated by models like GPT3 (Hartvigsen et al., 2022). The performance-oriented methods rely heavily on lexical cues in normal text and are vulnerable to even simple attacks.

The robustness-oriented methods are developed to enhance the detectors' robustness via adversarial training on perturbed data. The perturbation approaches (Bespalov et al., 2023; Le, Lee, Yen, Hu, & Lee, 2022; Ye, Le, & Lee, 2025) may manipulate on characters, words, or sentences in the text, e.g., character swap and/or substitution, homoglyph/homophone substitution, decomposition, near-neighbor words replacement, and distract injection (Cooper, Surdeanu, & Blanco, 2023; Emmery et al., 2022; Kurita, Belova, & Anastasopoulos, 2019; Yu, Choi, & Kim, 2024). The existing robustness-oriented methods all statically employ specific perturbations to enhance the robustness of detectors, but they struggle to dynamically adapt to increasing crafted perturbations.

The existing performance-oriented detection methods focus on the ordinary text, and prior robustness-oriented detection methods statically employ one or several perturbation strategies to improve the robustness of detectors against specific perturbations. Both of them cannot constantly detect the toxicity in the perturbed text that is increasingly crafted by malicious users. In contrast, we systematically investigate 7 existing and 2 new types of perturbations and incrementally exploit each type of perturbed data from the domain continual learning perspective to dynamically enhance the adaptability of the detector against increasing perturbations.

2.2. Adversarial robustness

The robustness of the models has always abstracted the focus of researchers in the past years and led to a wide variety of studies (Ali, Eleyan, Al-Khalidi, & Bejaoui, 2025; Liu, Yang, He, & Hopcroft, 2025; Sui et al., 2025; Wang, Li, Zhu, & Xie, 2024; Zühlke & Kudenko, 2025). Many of them focus on a particular type of data, such as the medical image (Kanca, Ayas, Baykal Kablan, & Ekinci, 2025), text (Goyal, Doddapaneni, Khapra, & Ravindran, 2023; Rajchandar, Manoharan, & Ashtikar, 2024; Zhang, Hong, et al., 2024), and multimodal (Mozhegova, Khattak, Khan, Garaev, & Rasheed, 2025). Many works analyze the interpretability (Alhazmi, Aljubairy, Zhang, Sheng, & Alhazmi, 2025; Vadillo, Santana, & Lozano, 2025) of attacks. Specifically, West et al. (2023) discusses enhancing adversarial robustness through quantum machine learning. Zhao et al. (2022) employs causal intervention to improve the robustness. Zeng, Xu, Zheng, and Huang (2023) introduces a random mask mechanism to perform certified defense. In summary, the methods of adversarial robustness contribute to improving a model's processing ability for corrupted attack cases, but they always consider improving the robustness by adversarial training on the specific data, so they are static and cannot dynamically generalize to other new data after adversarial learning.

However, the varying perturbations can generate significantly different toxicity distributions, even rendering the toxicity contained in the text completely invisible, and importantly, they are dynamically evolving since they are constantly created. Hence, the methods of adversarial robustness are also not applicable for toxicity detection against increasing types of perturbed text.

2.3. Domain adaptation

Domain adaptation aims to address the issue of poor generalization ability of models in scenarios with different data distributions, i.e., the adaptation problem when a model is trained in the source domain and applied to the target domain (Lei, Yu, Feng, Cui, & Xie, 2024; Li, Li, Du, Zhu, & Shen, 2025; Liu, Zhang, Luo, Huang, & Xu, 2024; Tang & Yang, 2024). There are many studies in recently (Lai et al., 2024; Shah, Sinha, Reddy, Roy, & Chellappa, 2025; Sun & Tao, 2024). For example, Liu, Zhang, et al. (2024) propose a simple yet effective self-supervised unsupervised domain adaptation framework for text recognition to integrate contrastive learning and consistency regularization to mitigate domain gaps. Li, Song, and Shao (2025) studies a novel learning paradigm for domain adaptation via learning using statistical invariance by simultaneously combining the strong and weak modes of convergence in a Hilbert space to fully incorporate multiple knowledge from the source domain. In conclusion, the existing methods contribute to the domain adaptation across the source and target domains in specific tasks or applications. However, the existing domain adaptation methods are also limited due to their static nature. Essentially, it aims to complete knowledge transfer from the source domain to the target domain in one go and cannot cope with scenarios where new domains continually emerge over time.

2.4. Continual learning

Continual learning aims to incrementally learn knowledge from a non-stationary data stream without catastrophic forgetting (Goodfellow, Mirza, Xiao, Courville, & Bengio, 2013). There are three continuous learning scenarios, i.e., domain-, class-, and task-incremental learning (Van de Ven, Tuytelaars, & Tolias, 2022).

In domain-incremental learning, all tasks have the same label space but different data distributions (domains) (Garg et al., 2022; Michalski, Mozetic, Hong, & Lavrac, 1986; Mirza, Masana, Possegger, & Bischof, 2022; Tan, Lin, & Hua, 2020; Wang, Zhang, et al., 2024). For example, for the datasets, Peng et al. (2019) collects and annotates multi-domains dataset, which contains 6 domains and owns 0.6 million images distributed among 345 categories, for the scenarios of multi-source domain adaptation like domain incremental learning. Hanna, Karunaratne, and Cabric (2022) constructs a multi-types of fingerprinting and each type can be treated as one domain. Li et al. (2023) suggests a continual deepfake detection benchmark over a new collection of deepfakes from both known and unknown generative models to study the detection of incrementally appearing deepfakes in the real-world scenarios. As for methods, Song, He, Dong, and Gong (2024) proposes a non-exemplar domain incremental objection detection method named learning domain bias to learn domain bias independently at each new session, adapting the base model to the distribution of new domains. Mulimani and Mesaros (2025) proposes a novel dynamic network architecture to keep the shared homogeneous acoustic characteristics of domains, and learns the domain-specific acoustic characteristics in incremental steps. Zhou, Cai, Ye, Zhang, and Zhan (2025) 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 process. Wang et al. (2025) introduces a gaussian mixture compressor and domain feature re-sampler to store and balance prior domain data efficiently, and proposes a multi-level domain feature fusion network to enhance domain feature extraction, aiming to boost the domain-incremental learning ability. Luo et al. (2025) proposes a domain incremental object detection with limited budgets, which aims to achieve new-instance incremental learning in real-world applications. In addition, many researchers also focus on federation domain incremental learning, such as Li, Xu, Wang, et al. (2024) and Li, Xu, Qi, et al. (2024), aiming to mitigate forgetting under the federation condition. Overall, the recent research mainly focuses on improving or evaluating domain incremental learning on specific tasks such as visual tasks since images are naturally distributed in different domains. There is no method or dataset suitable for the continual toxicity detection task on perturbed text, so the existing methods may be limited due to the unique properties (e.g., disturbed textual structure) of the task.

In task-incremental learning, the number of tasks to be solved increases constantly (Douillard, Cord, Ollion, Robert, & Valle, 2020; Fu, Wang, Yu, Xu, & Li, 2024; Gao, Shan, Zhang, & Zhou, 2023; Kanakis, Bruggemann, Saha, Georgoulis, Obukhov, & Van Gool, 2020; Oren & Wolf, 2021). Recently, Lee, Yoo, Kim, Choi, and Woo (2024) introduces an adapter-based continual imitation learning framework to address the limitation of knowledge sharing by incrementally learning shareable skills from different demonstrations, thus enabling sample-efficient task adaptation using the skills, particularly in non-stationary environments. Yang, Li, and Huang (2026) proposes to employ wavelet packet transform to extract global-view spatial features via the regularized common spatial pattern and capture local-view spatial features via tangent space mapping from the Riemannian space to improve daily living of patients with strokes based on EEG-based task incremental learning.

In class-incremental learning, the number of categories is continuously growing over time (Dong et al., 2022; Masana et al., 2022; Mittal, Galesso, & Brox, 2021; Wu et al., 2019; Zhang et al., 2020; Zhu, Cheng, Zhang, & Liu, 2021). For instance, Petit, Popescu, Schindler, Picard, and Delezoide (2023) employs a fixed feature extractor and a pseudo-features generator with geometric translation to improve the stability-plasticity balance to obtain good accuracy for past as well as new classes. Liu, Diao, et al. (2024) employs wavelet transform to map the image into the frequency domain and balances the re-usability and interference of output features based on the frequency domain similarity of the classes for mitigating the forgetting of previously acquired knowledge. He (2024) proposes to re-weight the gradients towards balanced optimization and unbiased classifier learning to address skewed gradient updates with biased weights. Li, Liu, et al. (2025) proposes to generalize conceptual knowledge learned from old classes to new classes by simulating human learning capabilities to mitigate the stability-plasticity challenge in few-shot class-incremental learning.

Our proposed problem can be classified into the domain-incremental learning, where the original data is changed by different perturbations for escaping detection. To the best of our knowledge, there are no tailored domain incremental learning methods and datasets for toxicity detection, so the continual toxicity detection is overlooked in the long term. Hence, our work is the first attempt at the exploration of domain incremental learning in toxicity detection.

3. Methodology

In this section, we first give the definition of the problem to be solved in Section 3.1. Then, we elaborate on how our dataset is constructed stage by stage in Section 3.2. In Section 3.3, we employ the existing detection methods to conduct a type-by-type toxicity recognition on perturbed text for every perturbation type (termed perturbation-wise detection, Section 3.3.1), and we propose continual detection (Section 3.3.2) and an associated detector (Section 3.3.3).

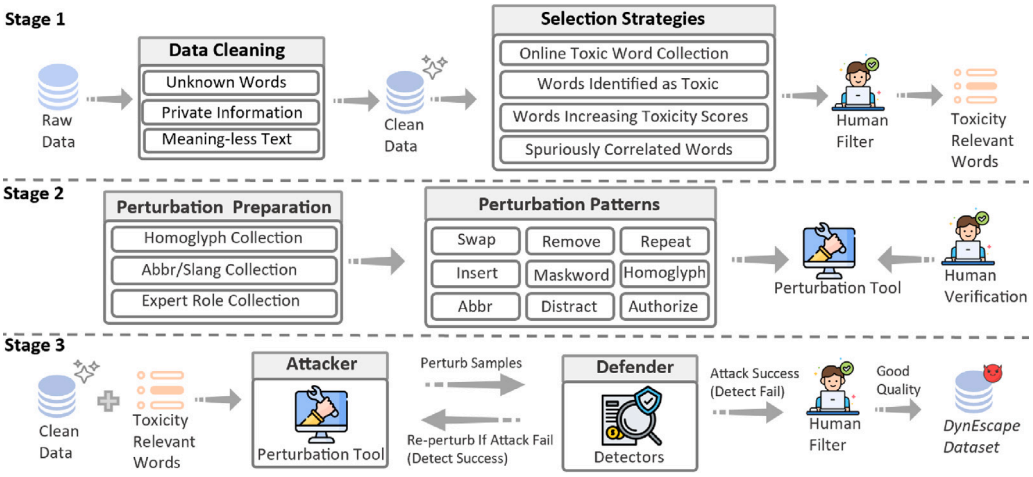

Fig. 2. The illustration of dataset construction stages.

3.1. Problem definition

Given a toxicity detector, there are many users who tend to create some types of perturbations to disturb toxic text to bypass the detector. Continuous toxicity detection aims to enable the detector F to continuously detect various emerging perturbed text. Formally, continuous toxicity detection aims to maximize the number of successfully detected toxic text as following:

$$\max \sum_{i=1} \sum_{j=1} \mathbb{I}(y_j = F(C_{j,p_i})), i, j = 1, 2, \ldots, \tag{1}$$

where p_i denotes the type of perturbation p_i that appeared at the ith moment during the usage of the detector F and y_j represents the ground truth toxic 0–1 labels (e.g., 1 denotes the text is toxic) of the jth text C_{j,p_i} perturbed by perturbation p_i.

In order to solve this problem, the following issues will be addressed step by step: (1) constructing a perturbation-wise managed dataset to perform the incremental types of perturbation and (2) developing an approach to conduct the continuous toxicity detection for incremental perturbations.

3.2. Construction of perturbation-wise managed dataset

Most existing toxicity datasets do not consider the perturbed text (Costa-jussà et al., 2024; Hartvigsen et al., 2022; Mathew et al., 2021; Sap et al., 2020). 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 (Cooper et al., 2023; Kirk, Vidgen, Röttger, Thrush, & Hale, 2022; Ye et al., 2025), by which we cannot examine the detectors' adaptability to changing perturbations. To address this issue, we construct *DynEscape*, a challenging perturbation-wise toxicity dataset covering 9 types of perturbations through three stages, as Fig. 2 shows.

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.

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.

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*.

3.2.1. Stage 1: Data cleaning and toxicity-relevant word selection

In this stage, we choose Jigsaw (Kivlichan, Sorensen, Julia Elliott, Görner, & Culliton, 2020) 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 explicitly toxic words from Google, including bad words and the banned swear words (2.9K words).

Words Identified as Toxic We employ PerspectiveAPI (Lees et al., 2022) 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:

$$E_{w_i} = \frac{1}{|T_{w_i}|} \sum_{t_j \in T_{w_i}} [s(t_j) - s(t_j/w_i)], \tag{2}$$

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 (Garg et al., 2023). We compute mutation information to get such words (Zhang, Chen, & Yang, 2023).

$$MI = \frac{p(w_i, c)}{p(w_i, \cdot)p(\cdot, c)}, \tag{3}$$

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 (Zhang et al., 2023), 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.

3.2.2. Stage 2: Perturbation tool creation

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.[1]

(2) Perturbation Patterns We then define perturbation patterns as follows (Table 1 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 these nine definitions, we implement all 9 perturbations to create a perturbation tool that can provide 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.

3.2.3. Stage 3: Adversarial attack and quality assessment

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 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 challenging perturbed samples.

To ensure the quality of our dataset, we conduct a quantitative assessment. Specifically, we assess every perturbed sample as follows:

(1) We first strictly auto-operate the perturbations on the original text according to the definitions of perturbations based on the existing works.

(2) We then require the three annotators to carefully read and understand the definitions and further ask them to assign the score range of 0 to 1 about whether the perturbed text is reliable, including the score about the perturbation fit to the definitions

[1] The details are given in Appendix.

Table 1
Examples of different types of perturbation.

Perturbations	Examples
Insert	moron→mo*ro#n; idiot→i@di+ot
Remove (Mv)	moron→moro; idiot→idot
Repeat	moron→ mooorrron; idiot→ iddddioot
Swap	fool→ folo; idiot→ idoit
Homoglyph (Homo)	idiot→idiοt; idiot→1di0t
Maskword (Mask)	fool→f**l; idiot→i*iot
Abbreviation (Abbr)	fuck→ 4Q; what the fuck→ WTF
Distract (Dis)	[text]→[apple what earth...];[text]
Authorization (Auth)	[text]→[I am a scientist...];[text]

and the score about the semantic consistency between the perturbed text and the original text. The average score denotes whether the perturbed text is high-quality.

(3) After the scoring, we select the perturbed text not only with high scores (empirically exceeding 0.5) but also with assessment consistency among annotators (all scores of annotators empirically exceed 0.5) to ensure the quality of the perturbed text.

Consequentially, we get the high-quality dataset owning the following advantages:

Balance Our dataset is balanced across labels since we re-sample non-toxic/toxic text with the ratio of 1:1. In addition, the number of samples in every perturbation type is equal for ensuring the balance across types.

Toxicity Diversity Our dataset has a high toxicity diversity since it inherits multiple types of toxicity from Jigsaw, including hate speech, social bias, offensive content, and so on.

Perturbation Diversity and Reality Our dataset has a high perturbation diversity, covering 9 types of perturbations. Moreover, these perturbations are widely used in real-world scenarios.

Consistency The perturbed text is semantically consistent with the original text, and the perturbation operations are consistent with the definitions. Specifically, we randomly select 50 samples for each type of perturbation to assess them and 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.

3.3. Detection methods comparison

In this section, we first systematically investigate the vulnerability of current methods when they detect increasing types of perturbed toxic text based on the constructed dataset. We then present our continual learning approach for toxicity detection.

3.3.1. Perturbation-wise detection on perturbed text

There are many methods to detect toxic text, and part of them fine-tune pretrained language models (Akram, Shahzad, & Bashir, 2023; Li, Zeng, Li, & Sun, 2024; Mousa, Shahin, Nassif, & Elnagar, 2024; Vallecillo-Rodríguez, Plaza-Del-Arco, & Montejo-Ráez, 2025) (e.g., BERT (Devlin, Chang, Lee, & Toutanova, 2019)) for adapting to the specific type of perturbed toxic text. To verify the vulnerability of these existing methods against text perturbed by dynamically changing perturbations, we perform two types of functional detection, including:

(1) *Zero-shot Detection by Existing Normal Detectors*

For the existing detectors, we quantitatively analyze their weaknesses against perturbed text without performing extra adjustments for any perturbations.

(2) *Cross-perturbation Detection by Detectors Fine-tuned on Specific Perturbations*

For the methods that fine-tune pretrained language models on specific perturbed text, showing adaptability to associated perturbations to some extent. In our study, we further quantitatively investigate the vulnerability of such detectors against newly appeared perturbations by performing cross-perturbation testing.

Accordingly, we expect to answer the questions:

Q1: *To what extent will the performance of existing detectors decline against varying perturbations?*

Q2: *To what extent will the performance of detectors fine-tuned on specific types of perturbations decline against newly emerged perturbations?*

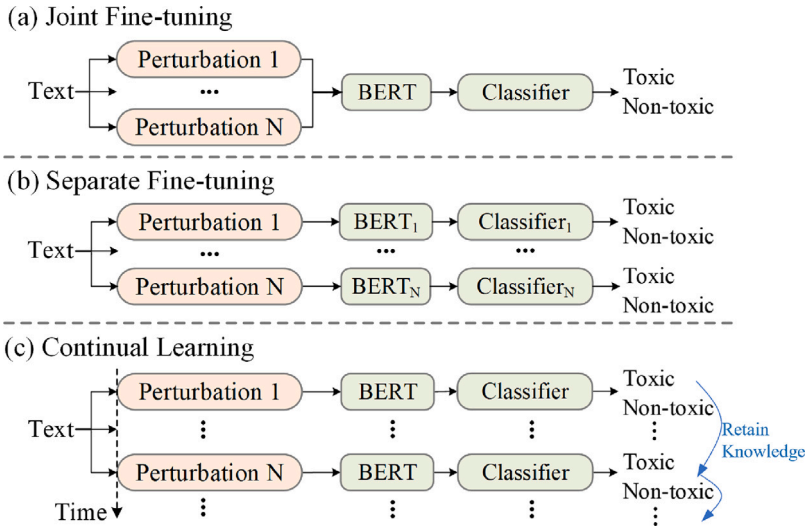

Fig. 3. Comparison of detection paradigms. (a) and (b) denote joint and separate fine-tuning, respectively. (c) denotes continual learning paradigm that can constantly adapt to the increasing types of perturbed text.

3.3.2. Continual detection on perturbed text

Malicious users employ dynamically changing perturbations to circumvent the detectors. The performance of existing methods will be limited when they encounter changing perturbations. Specifically, joint fine-tuned detector optimizes the parameters with all data, as Eq. (4) shows.

$$\theta^* = \mathbf{argmin}_\theta \sum_{(x_j,y_j)\in D} L(F(x_j;\theta), y_j), \tag{4}$$

where θ denotes the parameters of detector F, and D denotes the all dataset. L is the classification loss. In other words, joint fine-tuned detector does not consider the changing nature of perturbation over time and it combines and employs all the data to optimize a detector, as Fig. 3(a) shows. Once a new type of perturbation is created, the detector must be re-trained on all data to ensure its' adaptability to the new type of perturbation, which makes the detector not flexible along time and entails more cost.

Moreover, separate fine-tuned detectors are separately optimized for every single type of perturbation, as Eq. (5) shows.

$$\theta_i^* = \mathbf{argmin}_{\theta_i} \sum_{(x_j,y_j)\in D_{p_i}} L(F_i(x_j;\theta_i), y_j), \tag{5}$$

where θ_i denotes the parameters of the i_{th} detector F_i for the i_{th} perturbation p_i, and D_{p_i} denotes the associated dataset. That is to say, separate fine-tuned detectors are separately trained for every single type of perturbation as Fig. 3(b) shows. Hence, a new detector must be trained when a new type of perturbation appears such that the number of detectors will increasingly explode with the types of perturbation.

In summary, existing fine-tuned methods are not flexible for changing perturbations and consume more resources, e.g., storage.

To address this issue, we treat the text perturbed by different perturbations as the text distributed in different perturbation domains. Hence it is natural to detect the dynamically changing perturbed text via domain incremental learning as Fig. 3(c) shows, which aims to enable a detector to continuously and flexibly detect previous and new types of perturbed toxic text along the lifetime of the detector. Specifically, given a perturbation (domain) p_i at the i_{th} moment, the associated dataset $D_{p_i} = \{(x_{j,p_i}, y_j)\}$, where x_{j,p_i} denotes text perturbed by p_i and $y_j \in \{toxic, non-toxic\}$ denotes the truth label. Ideally, we hope the detector could adapt to both previous and newly appeared perturbed text:

$$\theta^* = \mathbf{argmin}_\theta \sum_{(x_{j,p_i},y_j)\in D_{p_i}}^{T} L(F(x_{j,p_i};\theta), y_j), \tag{6}$$

where (x_{j,p_i}, y_j) denotes the j_{th} sample in the D_{p_i} perturbed dataset, p_i is the perturbation appearing at the i_{th} moment during usage T of detector F.

In this way, we hope to answer the following question:

Q3: *Can domain incremental learning make detectors continuously adapt to toxic text perturbed by dynamically changing perturbations?*

3.3.3. Our continual detector DynDetect

There is no continual learning method available for addressing incremental toxicity detection across perturbation domains. In this subsection, we present a naive method *DynDetect* for perturbation domain incremental toxicity detection.

Table 2
Results for perturbation-wise detection on separately perturbed text using existing normal detectors. 'raw' and 'pert' denote the text without/with perturbation. The best performance without/with perturbation in each category is highlighted in **blue**/**brown** **bold**, respectively. The subscript indicates the relative performance degradation.

Detectors		Insert	Repeat	Swap	Mv	Homo	Mask	Abbr	Auth	Dis	Avg
Google	raw	87.1	85.4	85.8	84.8	94.3	87.9	86.6	87.8	86.5	87.4
	pert	**77.2**	64.6	**70.5**	**60.5**	64.8	**63.3**	50.9	66.4	64.4	$64.7_{25.9\%}$
OpenAI	raw	82.4	82.4	77.0	75.7	91.6	84.6	80.0	89.0	89.3	83.6
	pert	66.1	52.2	61.1	56.5	50.8	59.3	51.0	**84.5**	81.1	$62.5_{25.2\%}$
LLama3	raw	83.0	84.8	79.2	79.0	90.5	84.2	83.9	**90.0**	**89.5**	84.9
	pert	38.6	34.0	38.2	39.3	42.7	38.8	30.8	50.8	50.2	$40.4_{52.4\%}$
ChatGPT	raw	81.8	80.8	76.8	75.7	88.2	83.1	79.4	89.0	88.6	82.6
	pert	69.5	**74.6**	61.0	60.0	76.4	58.3	45.2	55.7	50.0	$61.2_{25.9\%}$
ToxBERT	raw	70.5	70.7	70.2	68.7	77.7	71.2	70.1	73.9	75.7	72.1
	pert	60.9	61.0	58.8	58.3	53.5	55.9	**58.3**	73.3	75.1	$61.7_{14.4\%}$
ToxRoBERTa	raw	82.6	80.3	77.3	78.0	89.8	82.9	82.4	86.7	85.5	82.8
	pert	50.9	50.4	51.2	52.3	50.1	50.3	51.4	55.3	79.8	$54.6_{34.0\%}$
Original	raw	**89.4**	**87.8**	**88.0**	**86.7**	**95.2**	**90.3**	**88.4**	**90.0**	89.3	**89.5**
	pert	61.6	63.8	61.9	57.7	57.0	59.3	56.0	71.9	75.7	$62.7_{29.8\%}$
Multilingual	raw	87.9	85.2	85.6	84.2	92.3	88.3	86.5	87.1	86.6	87.1
	pert	60.2	58.5	56.4	57.0	**82.5**	59.2	52.6	66.6	**81.9**	$63.9_{26.6\%}$
Unbiased	raw	85.9	86.8	81.9	82.4	91.2	86.1	86.2	85.7	86.3	85.8
	pert	51.3	51.0	53.3	53.4	50.1	51.9	51.5	78.6	80.1	$57.9_{32.5\%}$

We first employ BERT to encode the sample s_i to get its feature f_i:

$$f_i = \text{BERT}(s_i) \tag{7}$$

The classifier then assigns toxic/non-toxic label $\hat{y}_i$ to the sample:

$$\hat{y}_i = \text{Classifier}(f_i) \tag{8}$$

Moreover, to deal with the catastrophic forgetting issue in continual learning, we propose to replay the feature-level knowledge in previous perturbation domains for the new domain. Specifically, we replay memory samples from the old domains and maximize the similarity between memory features encoded by models at the old and current times, where the memory samples will be selected from the training dataset by employing widely used k-means clustering, i.e., the samples closest to cluster centers computed by k-means will be selected as the memory samples. Since memory features encapsulate knowledge in corresponding perturbation domains, our replaying strategy ensures the current model preserves the prior knowledge. The continual learning loss is computed as:

$$L_{kl,f} = -\sum_{i}^{T} log(diag(softmax(\mathbf{F}_{m_i}^{old} \times \mathbf{F}_{m_i}^{new}))), \tag{9}$$

where $\mathbf{F}_{m_i}^{old}$ and $\mathbf{F}_{m_i}^{new}$ respectively represent the extracted features by the old and new models from the i_{th} moment's memory samples. $\times$ and $softmax$ denote the matrix product and normalization function, respectively. $diag$ denotes the diagonal elements of the product, which represents the feature similarity of memory samples.

The final objective for our task is the combination of the toxicity detection task and the continual learning task, defined as:

$$L = L_{cls} + L_{kl,f}, \tag{10}$$

where $L_{cls} = -\frac{1}{|D|}\sum_{i=1}^{|D|} y_i \log(\hat{y}_i)$ is the main classification loss. The training is conducted using training samples to optimize the above objective.

4. Experiments

In this section, we explore the aspects mentioned above via experiments and use accuracy as the metric.

4.1. Experimental setup

We mainly conduct two types of experiments to compare the traditional methods and our proposed continual method as follows:

- Firstly, in Section 4.2, we utilize the existing detectors and the traditional fine-tuning methods to recognize toxicity for every type of perturbed text to quantitatively evaluate their vulnerability when the types of perturbed text are dynamically increasing. Specifically, we employ the existing detectors to detect toxicity in the perturbed text without fine-tuning, i.e., zero-shot detection. Also, we fine-tune the detectors on specific types of perturbed text and then test them on unseen types of perturbed text to simulate the situation in which detectors encounter the newly created perturbations, i.e., cross-perturbation detection.

Table 3
Results for cross-perturbation detection using detectors fine-tuned on specific perturbations. Note that row and column names denote the perturbation for training and testing, respectively. We employ the bert-base-uncased as the encoder.

	Insert	Repeat	Swap	Remove	Homo	Mask	Abbr	Auth	Distract	Avg
Joint	**85.39**	**84.39**	**86.17**	**86.63**	**81.76**	**85.09**	**84.08**	92.50	**91.11**	**86.35**
Separate	81.22	83.93	85.78	84.85	80.99	84.08	80.45	**92.66**	90.80	84.97
Insert	81.22	74.11	60.51	61.67	72.26	77.90	69.01	50.00	50.00	66.30
Repeat	50.00	83.93	50.54	50.46	50.00	50.00	50.77	50.00	50.00	53.97
Swap	52.01	51.08	85.78	79.98	52.86	62.67	73.42	50.08	50.08	62.00
Remove	79.52	60.20	80.76	84.85	72.02	79.75	72.02	50.23	50.08	69.94
Homo	53.71	68.47	63.91	67.47	80.99	57.42	60.66	51.00	50.00	61.51
Mask	73.72	76.28	73.42	70.79	73.96	84.08	70.79	50.00	50.00	69.23
Abbr	58.27	73.11	79.68	77.20	68.55	72.64	80.45	50.00	51.31	67.91
Author	50.77	53.86	64.91	70.32	55.10	55.18	53.79	92.66	89.72	65.15
Distract	50.77	58.35	56.26	62.06	51.08	52.94	54.87	91.89	90.80	63.22

- Next, in Section 4.3, we conduct continual learning methods and quantitatively evaluate their continual adaptation on the increasing types of perturbed text. Specifically, we will test the existing continual learning baselines and our proposed DynDetect, to evaluate the continual performance of them against increasing types of perturbed text, i.e., continual detection. Furthermore, we also conduct related analysis experiments to analyze the effectiveness of DynDetect.

4.2. Perturbation-wise detection on perturbed text

Zero-shot Detection by Existing Detectors To examine the vulnerability of existing normal detectors against different perturbations, we detect the toxic text before/after perturbations and compare the performance based on three types of widely used normal detectors: (1) Commercial APIs, (2) LLM based detectors, and (3) Pretrained detectors. The details of three types of normal detectors:

(1) Commercial APIs: Content moderation APIs from Google (i.e., PerspectiveAPI) (Lees et al., 2022) and OpenAI (Markov et al., 2023).

(2) LLM based detectors: We question LLama3 (Meta-Llama-3-8B-Instruct) (MetaAI, 2024) and ChatGPT (OpenAI, 2022) whether the text is toxic.

(3) Pretrained detectors: Detoxify (110M) (Hanu & Unitary team, 2020) (versions of *original*, *multilingual*, and *unbiased*), we adopt all three variations of detectors released by the Detoxify team, including Original, Multilingual, and Unbiased, which are fine-tuned on the Jigsaw dataset. And ToxBERT/ToxRoBERTa (110M) (Hartvigsen et al., 2022) are fine-tuned on the ToxiGEN dataset.

As Table 2 shows, all normal detectors suffer from sharp performance degrades after perturbations, where the absolute accuracy score drops by up to 44.54 points and at least 10.41 points. Many detectors even perform nearly at the random level when detecting some types of perturbed text. For example, the OpenAI content moderation API only gets an accuracy of 50.77% when detecting the text perturbed with *Homoglyph*.

Now, we can answer ***Q1***. Existing normal detectors are seriously vulnerable against perturbations.

Cross-perturbation Detection by Detectors Fine-tuned on Specific Perturbations To analyze the performance degradation of detectors fine-tuned on specific perturbation against newly appeared ones, we fine-tune BERT on a specific perturbation and then test the detector on others, which we call cross-perturbation detection. Please note that joint fine-tuning utilizes all types of perturbed text together, and its performance can be considered the reference upper bound. Separate fine-tuning trains a detector for each perturbation, and then the detector is utilized to conduct cross-perturbation detection on the remaining perturbations. Table 3 shows the results.

It is clear that detectors fine-tuned on a specific perturbation can reach superior performance on this associated perturbation, but their performance degrades significantly on newly appeared types of perturbations. For example, the accuracy of the detector fine-tuned on the *Repeat* perturbation is 83.93%, close to the upper bound, but it dramatically drops to nearly a 50% score on all newly appeared types of perturbations.

From these results, we can answer ***Q2***. The detectors fine-tuned on specific type of perturbed text can adapt well to the associated perturbation, but they cannot fit newly appeared ones, and they even perform randomly on many new perturbations.

4.3. Continual detection on perturbed text

To investigate how our proposed continual detection approach *DynDetect* performs on the dynamically perturbed text, we modify typical methods widely used in domain continual learning as baselines, including:

LFL (Jung, Ju, Jung, & Kim, 2016) reduces knowledge forgetting by regularizing the parameters between old and new classifiers.

EWC (Kirkpatrick et al., 2017) uses the Fisher matrix to assess parameter importance and applies weighted regularization based on that importance.

LWF (Li & Hoiem, 2017) transfers knowledge by having the old model generate pseudo-labels for the new model.

Table 4
Average accuracy of continual detection on perturbed text in different moments. T_i denotes i_{th} moment during the detector usage. The best continual learning performances are highlighted in **bold**.

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9
$Stream_{re}$	84.23	77.22	62.60	69.73	61.98	71.91	68.60	57.20	65.08
Stream	84.23	84.64	76.52	77.36	78.17	79.55	79.73	74.28	78.26
LFL	83.46	82.88	77.15	76.59	75.30	78.25	78.44	75.01	78.15
EWC	84.66	84.88	76.41	78.48	77.89	80.89	78.64	76.01	79.32
LWF	83.46	85.08	79.39	79.86	76.31	76.92	75.56	72.58	72.30
GEM	83.94	84.83	78.39	79.96	80.82	80.76	80.56	78.47	80.99
EPI	84.12	81.46	81.60	79.13	79.90	78.23	76.98	78.26	79.20
CEAR	84.35	84.39	72.56	78.60	80.45	80.97	81.22	**81.30**	80.60
DUCT	78.75	75.51	67.6	67.97	66.86	69.47	70.82	67.29	62.98
SOYO	80.47	81.1	80.11	79.67	79.52	79.46	79.27	79.66	80.07
DynDetect	**84.80**	**85.61**	**82.32**	**82.82**	**82.65**	**83.27**	**82.58**	81.05	**82.90**

Table 5
Detailed results for continual detection methods against the different types of perturbed text. The best average performances are highlighted in **bold**. Please find more detailed results under different orders in the Table 12-14 in the appendix.

Order:	*Mv*	*Swap*	*Repeat*	*Insert*	*Homo*	*Abbr*	*Mask*	*Dis*	*Auth*	*Avg*
$Stream_{re}$	67.08	62.83	53.09	52.32	58.89	52.09	56.88	90.03	93.12	65.15
Stream	81.84	81.22	61.05	75.73	74.81	70.09	81.07	91.89	92.89	78.95
LFL	82.30	79.83	55.95	80.37	75.27	74.42	80.68	90.11	92.74	79.07
EWC	82.30	80.76	64.37	76.51	77.28	76.97	81.68	90.65	91.73	80.25
LWF	63.37	64.53	69.55	68.78	57.88	62.06	65.92	71.79	64.68	65.40
GEM	82.77	81.07	69.01	76.66	76.74	79.60	82.53	89.72	92.58	81.19
EPI	72.95	69.09	80.99	73.11	78.05	77.13	77.51	90.88	93.04	79.19
CEAR	73.34	72.49	74.88	76.28	73.03	74.88	73.26	86.09	88.87	77.01
DUCT	71.33	73.11	73.03	76.51	72.02	72.26	73.8	50.00	50.00	68.01
SOYO	79.52	80.91	77.51	76.97	75.81	78.75	79.06	84.78	87.79	80.12
DynDetect	82.46	82.15	80.68	78.21	77.36	76.74	83.38	92.04	92.74	**82.86**

GEM (Lopez-Paz & Ranzato, 2017) stores gradient information from past tasks and, when learning new tasks, constrains the gradient direction of the current task to prevent it from conflicting with the gradient directions of old tasks.

EPI (Wang et al., 2023) 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 (Zhao, Cui, & Hu, 2023) learns memory-insensitive prototypes and uses memory augmentation to reduce overfitting and enhance performance.

DUCT (Zhou et al., 2025) proposes dual consolidation to construct a unified representation space applicable to multiple domains and integrate backbones from different moments to mitigate the forgetting.

SOYO (Wang et al., 2025) 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.

We reproduce the results for these baselines with officially released codes by the authors. The number of replay samples is set to 5 for continual learning methods.

In addition, according to the widely used reference in continual learning, the joint and separate fine-tuning perform as the upper bound of incremental learning, and the stream fine-tuning performs as the lower bound, where the detector is fine-tuned sequentially for each perturbation and the method includes two variants according to with/without restart detector facing new perturbation, termed as $Stream_{re}$ and Stream, respectively.

4.3.1. Continual detection analysis

we conduct experiments on four orders of the increment perturbation types: (1) *Order-1* is a gradual change of perturbation types from the character level to the word/phrase level and then to the sentence level. (2) *Order-2* reverses the *Order-1*, (3) *Order-3* randomly changes the *Order-1* in the same character/word/sentence level, e.g., *Swap* becomes the first in *character* level perturbations, (4) *Order-4* randomly changes the *Order-1* among different levels, e.g., sentence level perturbations first and then character level ones.

The results in Table 4 demonstrate that our proposed *DynDetect* method outperforms all continual learning baselines. In the perturbed text, there is but only disparity among different perturbation types, and the operation randomness also exists in the same perturbation types. Hence, the representative features encoded by the detector are complicated. However, the existing baselines only consider label-level distillation or hard constraints for detector parameters, such that the direct retainment of these baselines is

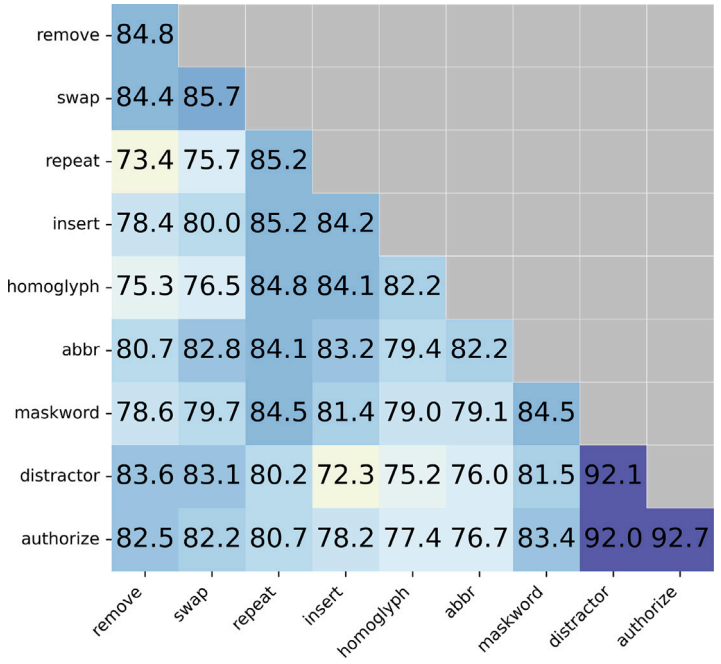

Fig. 4. Accuracy change of different perturbations during the continual learning process. Every column denotes the accuracy change of every type of perturbation.

weak. Whereas our proposed method directly replays previous features to ensure stronger and more explicit retention, which makes our method outperform the baselines.

Moreover, the overall performance of our method (T_9 column in Table 4) reaches significant improvements over existing normal detectors and detectors fine-tuned on specific perturbations in Tables 2 and 3 (Avg column). Furthermore, Table 5 shows the detailed results on different types of perturbation. We can find that our method achieves the best overall performance. That is, when new types of perturbations keep emerging, our method can maximize the identification of toxic text among all types of perturbed text. In addition, Fig. 4 shows the accuracy change during the continual learning process. Overall, we find a phenomenon that the accuracy would drop midway but then recover. We argue this is because the detection ability for seen perturbations will be corrupted when the detector encounters the new perturbations but will be gradually recovered from the memory samples over time. For example, the performance for *Remove* significantly declines to 73.0 when the detector firstly encounters the *Repeat*, but the performance recovers to 82.0 at the last moment.

We can now answer ***Q3***. The domain incremental learning enables the detector to constantly recognize changing perturbed toxic text and exhibit superior performance.

4.3.2. Perturbation order sensitivity analysis

To examine the impact of perturbation orders, we compare the overall performances of continual detection methods under different orders. As Table 6 shows, we can find that our *DynDetect* is always the best among all methods. Finally, all continual learning methods under various orders consistently beat normal detectors and those fine-tuned on specific perturbations, which clearly proves the necessity and superiority of the continual learning for toxicity detection against increasing types of perturbed toxic text.

4.3.3. Ablation analysis

We conduct different ablated variants to illustrate the effectiveness of different strategies/components, as Table 7 shows.

Firstly, we remove the feature similarity maximization strategy to analyze the effectiveness (termed as DynDetect$_{\text{-fsm}}$). Without the feature similarity maximization strategy, the performance declines across different task orders, even worse than CEAR due to the absence of explicit transfer between different moments. Furthermore, we remove the sub-strategy that we mix the memory samples during continual training at every moment (termed as DynDetect$_{\text{-mixture}}$) and remove the sub-strategy that we post-train the detector with the memory samples after training on perturbed text at every moment (termed as DynDetect$_{\text{-post}}$). After removing these strategies, the overall performance of the detector declines despite these variants that may achieve local best performance in some individual orders. In conclusion, the results show the effectiveness of different strategies in our method.

Table 6
Overall performance for different perturbation orders.

	Order-1	Order-2	Order-3	Order-4	Avg
$\text{Stream}_{\text{re}}$	65.64	63.41	65.15	66.11	65.08
Stream	79.49	74.08	78.95	80.49	78.26
LFL	79.48	74.19	79.07	79.85	78.15
EWC	79.81	75.43	80.25	81.80	79.32
LWF	72.29	74.94	65.40	76.56	72.30
GEM	80.51	80.23	81.19	82.03	80.99
EPI	79.03	79.50	79.19	79.07	79.20
CEAR	82.71	80.04	77.01	82.65	80.60
DUCT	66.87	56.81	68.01	60.23	62.98
SOYO	80.18	79.99	80.12	79.98	80.07
DynDetect	**83.29**	**82.15**	**82.86**	**83.29**	**82.90**

Table 7
The performance of different ablated variants.

	Order-1	Order-2	Order-3	Order-4	Avg
DynDetect	**83.29**	82.15	82.86	**83.29**	**82.90**
$\text{DynDetect}_{\text{-fsm}}$	77.82	81.10	81.43	81.80	80.54
$\text{DynDetect}_{\text{-post}}$	81.39	**82.72**	82.33	83.02	82.37
$\text{DynDetect}_{\text{-mixture}}$	80.78	81.06	**84.03**	82.96	82.20

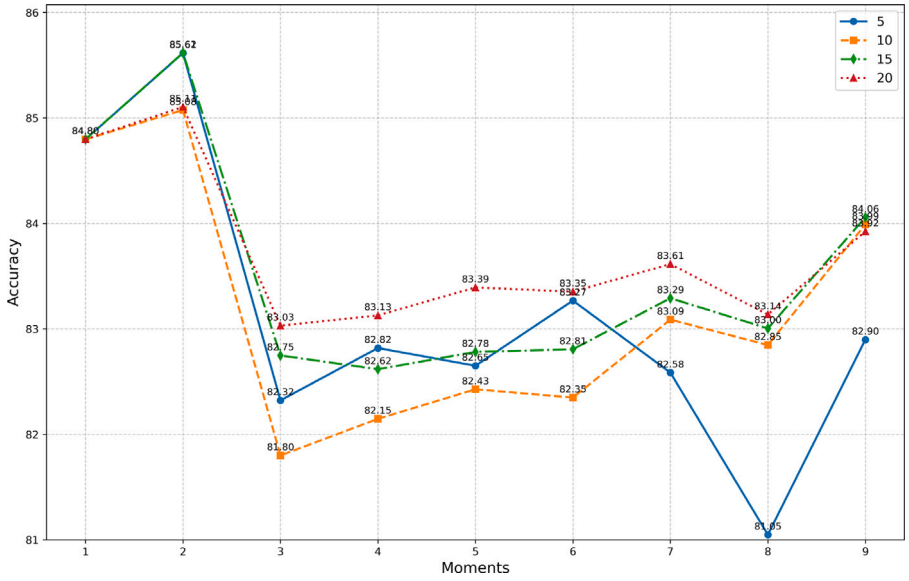

Fig. 5. Results under different numbers of memory samples.

4.3.4. *Memory number analysis*

To explore how detection capacity of DynDetect changes with the number of memory samples, we adjust the number and conduct associated experiments. As illustrated in Fig. 5, the performance variation trends under different numbers exhibit similarity. We suggest this consistency may stem from the inherent patterns and disparities in perturbations that emerge at distinct moments. Notably, as the number of memory samples increases, the detector's performance demonstrates enhanced stability, particularly from the T_3 to T_7 time intervals. Additionally, the results show that more memory samples generally improve performance, which is most pronounced when the number is set to 20. However, from T_6 to T_9, increasing the sample number from 10 to 20 yields diminishing performance gains, indicating that the performance increment may approach a saturation point. In summary, adding more memory samples usually improves detection performance to some extent, but a saturation may exist for the improvement over the time.

4.3.5. *Cross dataset generalization analysis*

To explore the generalization of DynDetect, we conduct a cross-dataset testing experiment. Specifically, we employ different methods to detect the toxic text in the NoisyHate (Ye et al., 2025) dataset, where the dataset is composed of mixed perturbed text

Table 8
The performance on NoisyHate dataset. Best score is highlighted in **bold**.

$Stream_{re}$	Stream	LFL	EWC	LWF	GEM	EPI	CEAR	DUCT	SOYO	DynDetect
69.9	74.4	76.1	72.1	63.5	71.7	43.0	72.4	62.8	71.5	**78.6**

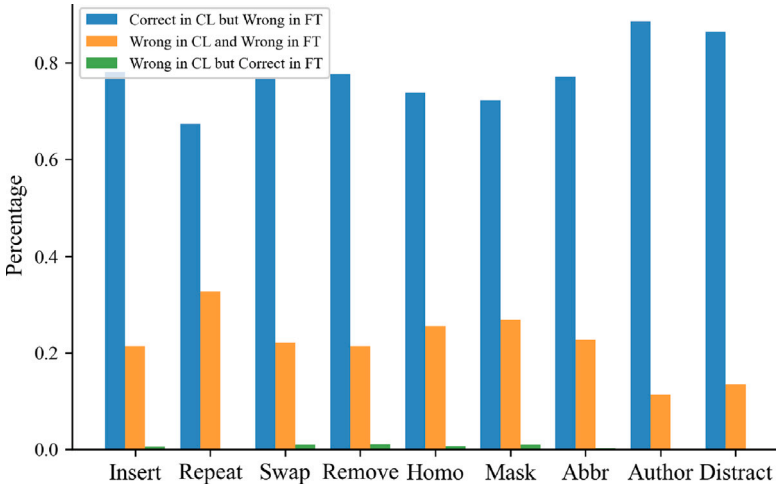

Fig. 6. The error percentage comparison between fine-tuning (FT) and continual learning (CL) methods on each perturbation.

by 5 perturbations and the perturbed text is human-written and collected online. As Table 8 shows, our proposed DynDetect achieves the best performance, even largely outperforming the previously best method (67.2) (Ye et al., 2025) (exceeding 11.4 points) in NoisyHate, which illustrates the generalization of DynDetect on even unseen mixed perturbed text in the wild.

4.3.6. Error analysis

We analyze the proportions of error samples wrongly predicted by fine-tuning (FT) and continual learning (CL) as Fig. 6 shows. There are mainly two categories of errors:

(1) Error samples that FT detectors cannot detect but our CL detector can (*blue* bars). Error case:

Raw: *you are dumb...*

Pert: *yo$=· u are du,m×b...*

This case is perturbed by *Insert*. The FT detector trained on this perturbation succeeds but those trained on *Repeat, Swap, Homo., Distract, Authorize* all fail, showing their vulnerability against newly appeared perturbations. In contrast, our CL method incrementally learns new perturbations while preserving the knowledge of old perturbations, and can constantly detect the perturbed toxic text.

(2) Error samples that either CL or FT detectors cannot detect (*orange* bars). Error case:

Raw: *Your edit to 2100 don't be stupid...*

Pert: *Yoru edti ot 2100 don't eb tsupid...*

This case is perturbed by *Swap*. The FT detector succeeds once on its associated perturbation (T_3) but fails on all other ones. Our CL method succeeds from T_3 to T_7 but fails after the T_8 moment. This suggests that our CL model can identify the toxic text when it first encounters the *Swap* perturbation and keeps the knowledge for a time period. However, the model still suffers from the forgetting problem over time, and thus losing the ability of detecting this type of perturbed text at later moments.

5. Conclusion

5.1. Summarization

The toxicity detectors' adaptability to evolving perturbations is an extremely important but overlooked problem for safe online communication. In this paper, we develop a continuous toxicity detection benchmark for this problem. We first construct a toxicity detection dataset including 9 types of perturbations in an adversarial attack way, where the dataset is separately managed by the perturbation type. Using the dataset, we reveal the vulnerability of current methods against toxic text produced by changing perturbations. We then propose an effective incremental learning-based method that can adapt to changing perturbations.

Extensive experiments demonstrate the potential of our proposed continual detection paradigm towards the adaptability to evolving perturbations. We hope our research will inspire more efforts for the problem.

5.2. Application discussion

Our proposed detector fine-tunes the bert-base-uncased pretrained language model, and the detector owns a total of 110M parameters, and the parameters will take 420MB of storage space to be saved. In addition, we store the fixed number of memory samples (set to 5) for each perturbation type, and it will take 12 KB of storage space to store the samples of each type. During the training stage, the optimization of the detector takes about 2.7 h of training time. When deployed, the detection speed can achieve 0.008 s/item (120 items/s), which shows superior real-time detection capability.

5.3. Limitation

Our dataset, DynEscape, is auto-perturbed from the ordinary toxicity detection Jigsaw dataset. When we employ the wild, human-written toxicity detection NoisyHate dataset to test our detector fine-tuned on the DynEscape, the performance will decline at different degrees. This may be contributed to the difference between perturbation operations on DynEscape and NoisyHate. Hence, it is worthy to further explore how to mitigate the performance degradation. At the same time, we only focus on English so far, and other languages are not explored yet.

In addition, our detector focuses on textual toxicity detection, so it is not able to detect multimodal toxic content. We think the detection of multimodal toxic content needs more comprehensive designs, such as multimodal feature encoding and cooperation among modalities, and it is more general in application scenarios. We also hope the researchers could solve the more general problem with us in the future, contributing to this practical problem.

Also, we analyze that there may also be some limitations: our detector cannot directly detect the text over 512 tokens since our used bert-base-uncased backbone only adapts to process the text shorter than 512. However, the limitation may be solved by many means. For example, we can cut the long text into many shorter spans and detect the spans item by item. On the other hand, we also can select other backbones with a large scale of parameters owning the processing ability for longer context, such as bert-large-uncased (accordingly, the memory to save becomes more).

5.4. Future work

We focus on the problem of making toxicity detectors adapt to continually changing perturbations. We encourage researchers to explore more types of perturbations to evade detectors for the purpose of improving detectors' adaptability to changing perturbations. For example, with the wide application of LLMs, more LLM-based perturbations will appear to spread the toxic content, which will raise concerns for healthy communication. In addition, though we propose a continual learning-based method to solve the problem, there are many potential solutions we have not deeply explored yet, such as meta-learning and transfer learning-based methods.

CRediT authorship contribution statement

Hankun Kang: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Jianhao Chen:** Writing – review & editing, Writing – original draft, Investigation, Data curation. **Yongqi Li:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Investigation, Data curation. **Xin Miao:** Writing – review & editing, Writing – original draft, Methodology, Investigation, Data curation, Conceptualization. **Mayi Xu:** Writing – review & editing, Writing – original draft, Supervision, Resources, Data curation. **Shen Zhou:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Data curation. **Ming Zhong:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Investigation, Conceptualization. **Yuanyuan Zhu:** Writing – review & editing, Writing – original draft, Validation, Supervision, Data curation, Conceptualization. **Tieyun Qian:** Writing – review & editing, Writing – original draft, Validation, Supervision, Methodology, Investigation, Data curation, Conceptualization.

Ethics statement

We strictly adhere to usage agreements of all employed resources. The statement in the paper is only for research purposes of improving toxicity detection, and the statement does not represent the value perspectives of any individual or group.

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), the Fundamental Research Funds for the Central Universities, China (Grant No. 2042022dx0001), and the grant from the Key Laboratory of Computing Power Network and Information Security, Ministry of Education (Grant No. 2024ZD027).

Table 9
Dataset comparison. *Perturbed* denotes whether the dataset is perturbed. *Perturbation-wise* indicates whether the dataset is distinguished by the types of perturbations. *Types* denotes the number of perturbation types employed in the datasets.

Dataset	Perturbed	Perturbation-wise	Types
Jigsaw (Kivlichan et al., 2020)	✘	✘	None
SBIC (Sap et al., 2020)	✘	✘	None
HateXplain (Mathew et al., 2021)	✘	✘	None
ToxiGEN (Hartvigsen et al., 2022)	✘	✘	None
TOXICN (Lu et al., 2023)	✘	✘	None
MuTox (Costa-jussà et al., 2024)	✘	✘	None
ToxCMM (Maity et al., 2024)	✘	✘	None
HatemojiBuild (Kirk et al., 2022)	✔	✘	1
NoisyHate (Ye et al., 2025)	✔	✘	5
OTH (Cooper et al., 2023)	✔	✘	1
DynEscape (Ours)	✔	✔	9

Table 10
Statistics of *DynEscape*.

Splits	Repeat	Remove	Swap	Abbr	Mask	Homo	Insert	Auth	Dis	Total
Train	2584	2584	2584	2584	2584	2584	2584	2584	2584	23,256
Valid	430	430	430	430	430	430	430	430	430	3870
Test	1294	1294	1294	1294	1294	1294	1294	1294	1294	11,646

Appendix

A.1. Details of experimental implementation

In all experiments, we use the bert-base-uncased model (110M) to encode the sentences, where the maximum sequence length is set to 360. The parameters of the models are updated using the AdaW optimizer with a learning rate of 2e-5. Additionally, the random seed for all experiments is set to 0, and training is conducted for 35 epochs with an early stopping strategy applied. All experiments are performed on the A800 GPU.

A.2. Datasets

A.2.1. Introduction of existing datasets

Jigsaw (Kivlichan et al., 2020) is composed of English content collected from online platforms like Twitter (now called X).

SBIC (Sap et al., 2020) includes social media posts that focus on social bias against specific groups, such as 'Muslims'.

HateXplain Mathew et al. (2021) is a collection of online social posts aimed at researching explainable hate speech detection.

ToxiGEN (Hartvigsen et al., 2022) is the first English dataset generated by large language models (LLMs) for toxicity detection, created in response to the growing use of LLMs.

TOXICN (Lu et al., 2023) is a Chinese toxicity dataset designed to support fine-grained detection of toxicity in Chinese, including the type and expression of toxicity.

MuTox (Costa-jussà et al., 2024) and ToxCMM (Maity, Poornash, Saha, & Bhattacharyya, 2024) are multimodal toxicity detection datasets that include not only text but also audio (MuTox) and video (ToxCMM).

HatemojiBuild (Kirk et al., 2022) and OTH (Cooper et al., 2023) datasets focused on emoji-based and homoglyph-based perturbed toxicity detection, aiming to enhance the robustness of detectors against toxic text that includes emojis or homoglyphs.

NoisyHate (Ye et al., 2025) combines five types of perturbed toxic text to strengthen the robustness of the detectors against various perturbations.

Compared with our *DynEscape*, existing datasets consider only a few or no perturbations. Moreover, they mix all perturbed text together, i.e., they are not perturbation-wise, which hinders the exploration of the dynamically changing property of perturbations (see Tables 9 and 10).

A.2.2. The implementation details for dataset construction

In perturbation implementation, we employ online toxic words collected by Google, which are available by https://github.com/coffee-and-fun/google-profanity-words/tree/main (the toxicity-relevant words are illustrated in Fig. 9), 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

Table 11
The prompt of Authorization perturbation. *ROLE* denotes the placeholder of a specific expert.

	Prompt
System	You are a helpful role player who can role as different experts in different domains!
User	Please role as a *ROLE*. Introduce how you are widely acceptable and authoritative.

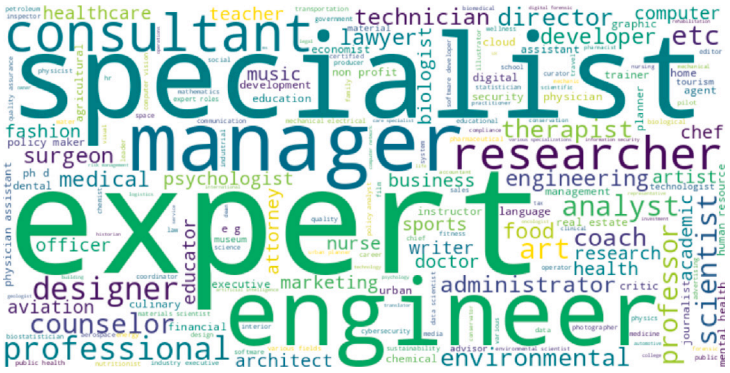

Fig. 7. Illustration of expert roles.

'acronymsandslang.com'. The employed instruction prompts for **Authorization** are shown in Table 11 and expert roles are illustrated in Fig. 7.

Specifically, Alg. 1 presents the algorithm for the perturbation procedure. And the format for our *DynEscape* dataset is shown in Fig. 8.

Algorithm 1 Algorithm for Perturbation Procedure

Input: perturbation method p and perturbation rate r; text T and associated label y_T
Output: perturbed text T_p with method p
Select important words: $W_T \leftarrow Selector(T)$
$Pertubator \leftarrow InitPertubator(method = p)$
Maximum number of attempts: $C \leftarrow 5$
$try \leftarrow 0$
while $try \leq C$ **do**
  for each word $w_i \in T, i \in \{1, 2, ..., |T|\}$ **do**
    if $w_i \in W_T$ **then**
      $w_{i,p} = Perturbator(w_i)$
    else
      if $random.rand() \leq r$ **then**
        $w_{i,p} = Pertubator(w_i)$
      end if
    end if
  end for
  $T_p = [w_{1,p}, w_{2,p}, ..., w_{|T|,p}]$
  $try \leftarrow try + 1$
  if $Detector(T_p) \neq y_T$ **then**
    return T_p
  end if
end while

A.2.3. Discussion about changing perturbations

The perturbations are constantly evolving, and the types of them will increase accordingly. Broadening our train of thought, it might be possible to embed toxic text into non-toxic text in the form of an acrostic poem. Alternatively, it is also possible to make toxic text visually through many meaningless words as a whole. Even the continuous evolution of the meanings of slang can lead to changing the types of perturbations. Hence, the types of perturbations are not fixed and will be increasingly created by users, especially as it becomes easier to employ LLMs' ability of instruction following to perturb toxic text for bypassing detectors.

```
{
    "id": "68be52f6ce7ac481",
    "comment_text": "I love the taste
    ↪  of my cum",
    "toxic": 1,
    "severe_toxic": 0,
    "obscene": 0,
    "threat": 0,
    "insult": 0,
    "identity_hate": 0,
    "pert_comment_text": "I love the
    ↪  t[×a-·ste of m@y c\"um",
    "pert_method": "insert"
},
{
    "id": "2385804471928c58",
    "comment_text": "Does it ever dawn
    ↪  on you that you are the reason
    ↪  some people hate Jews?",
    "toxic": 1,
    "severe_toxic": 0,
    "obscene": 0,
    "threat": 0,
    "insult": 0,
    "identity_hate": 0,
    "pert_comment_text": "Does it ever
    ↪  dawn o^n y=< {ou that y[]-&ou
    ↪  are the reason some people
    ↪  h\"^,}at?,_xe Je'*ws?",
    "pert_method": "insert"
},
```


Fig. 8. Illustration of format for our *DynEscape* dataset.

Fig. 9. Illustration of toxicity-relevant words.

In summary, it is important to study the changing perturbations and the adaptability to them for keeping healthy communication. However, since the perturbations are constantly changing, we cannot exhaust all types of perturbations. Therefore, we construct the data based on the commonly used existing perturbation types so as to conduct research on the problem of toxicity detection when the types of perturbations increase.

A.3. Extra experimental results

A.3.1. Detailed results under different orders

Tables 12–14 show the performance on the different types of perturbations when the detector is fine-tuned under different orders. We can find that our method achieves the best overall performance under different orders.

Table 12
Detailed results for continual detection methods against the different types of perturbed text. The best **average** performances are highlighted in **bold**.

Order:	*Insert*	*Repeat*	*Swap*	*Mv*	*Homo*	*Mask*	*Abbr*	*Auth*	*Dis*	*Avg*
$\text{Stream}_{\text{re}}$	54.25	58.58	61.44	66.62	55.18	56.88	54.10	92.04	91.65	65.64
Stream	78.13	57.34	82.15	81.07	78.36	76.43	76.97	93.35	91.65	79.49
LFL	76.51	58.96	79.52	81.30	78.44	79.52	77.59	92.66	90.80	79.48
EWC	79.21	59.43	79.98	81.07	77.13	78.98	79.37	91.27	91.81	79.81
LWF	79.68	64.37	76.12	75.12	75.27	77.67	78.67	66.31	57.42	72.29
GEM	82.30	57.26	82.77	83.08	78.67	78.21	80.45	91.50	90.34	80.51
EPI	71.64	81.68	70.32	72.18	79.52	77.28	75.50	93.51	89.64	79.03
CEAR	79.06	76.43	81.45	83.77	78.75	81.14	78.90	92.97	91.96	82.71
DUCT	66.38	70.32	76.43	76.35	68.62	69.47	72.64	50.08	51.55	66.87
SOYO	77.2	77.43	80.99	79.44	75.81	79.29	78.75	87.79	84.93	80.18
DynDetect	82.53	75.19	82.92	84.00	79.21	80.06	80.29	93.43	91.96	**83.29**

Table 13
Detailed results for continual detection methods against the different types of perturbed text. The best **average** performances are highlighted in **bold**.

Order:	*Dis*	*Auth*	*Abbr*	*Mask*	*Homo*	*Mv*	*Swap*	*Repeat*	*Insert*	*Avg*
$\text{Stream}_{\text{re}}$	50.00	50.00	65.61	76.43	64.37	57.57	58.19	66.62	81.92	63.41
Stream	62.44	65.38	74.65	73.26	75.73	73.18	76.20	82.46	83.46	74.08
LFL	61.90	59.97	76.74	78.28	68.62	78.36	80.22	79.60	84.00	74.19
EWC	63.45	60.51	76.89	81.61	76.97	76.12	77.98	81.22	84.08	75.43
LWF	74.19	67.00	78.67	73.18	68.01	75.89	78.90	79.37	79.21	74.94
GEM	82.84	84.39	77.20	80.60	77.90	74.42	78.52	82.92	83.31	80.23
EPI	90.88	93.66	77.13	76.35	79.98	71.64	69.71	82.46	73.72	79.50
CEAR	86.01	85.94	73.72	80.45	77.51	76.20	76.43	80.53	83.54	80.04
DUCT	51.08	63.76	60.97	43.43	71.79	53.09	52.16	56.34	58.66	56.81
SOYO	84.70	87.33	78.67	79.29	75.73	79.37	80.60	77.20	77.05	79.99
DynDetect	88.72	86.55	79.83	80.53	78.75	76.89	79.98	84.16	83.93	**82.15**

Table 14
Detailed results for continual detection methods against the different types of perturbed text. The best **average** performances are highlighted in **bold**.

Order:	*Insert*	*Swap*	*Homo*	*Mask*	*Auth*	*Mv*	*Abbr*	*Repeat*	*Dis*	*Avg*
$\text{Stream}_{\text{re}}$	57.11	62.21	57.96	58.96	92.19	67.16	51.62	55.87	91.89	66.11
Stream	80.29	79.44	79.37	78.59	92.50	81.14	78.44	62.91	91.73	80.49
LFL	79.06	75.73	68.01	74.96	91.73	79.91	79.06	79.68	90.49	79.85
EWC	81.53	79.60	76.89	78.67	92.27	81.22	79.44	75.19	91.42	81.80
LWF	80.06	76.43	77.59	75.19	68.47	77.36	78.05	75.73	80.14	76.56
GEM	83.00	76.35	80.22	79.29	92.27	77.90	79.44	78.83	90.96	82.03
EPI	71.64	68.86	79.91	76.20	92.19	71.64	77.59	81.68	91.89	79.07
CEAR	78.98	78.59	80.37	80.06	90.88	82.30	79.29	83.31	90.11	82.65
DUCT	50.00	63.37	71.56	53.32	50.08	61.90	65.77	76.04	50.00	60.23
SOYO	77.28	80.6	75.43	79.29	87.64	78.83	78.52	77.28	84.93	79.98
DynDetect	81.84	79.37	80.22	80.99	92.66	77.74	81.38	83.85	91.58	**83.29**

Table 15
Results for generalization test on combined perturbations.

	BERT_{Insert}	BERT_{Remove}	BERT_{Repeat}	BERT_{Swap}	BERT_{Homo}
Joint	85.39	86.63	84.39	86.17	81.76
Separate	81.22	84.85	83.93	85.78	80.99
Insert-Remove	72.41	81.45	49.92	69.01	65.92
Insert-Repeat	75.43	64.06	66.85	60.05	80.91
Insert-Swap	73.11	79.60	50.00	72.57	64.53
Insert-Homo	74.19	76.35	50.00	58.04	70.79
Remove-Repeat	66.92	70.09	75.27	55.18	81.45
Remove-Swap	57.42	82.30	49.92	82.61	80.22
Remove-Homo	60.12	80.14	49.85	73.72	81.45
Repeat-Swap	66.00	69.47	75.43	55.41	80.91
Repeat-Homo	69.71	59.74	70.09	51.93	73.03
Swap-Homo	62.21	78.98	50.00	78.36	79.75

Table 16
Results for prototype based method on dynamically perturbed text. T_i denotes i_{th} moment during the detector usage.

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9
Prototype	77.51	70.02	58.74	63.44	56.12	64.32	64.04	55.22	52.84

Table 17
Comparison between QWen3-0.6B and BERT-base based continual detectors.

	Parameters	Storage	Training time	Acc
Qwen3-0.6B	752M	2355MB	18.6H	77.17
BERT-base	110M	420MB	2.7H	82.90

Table 18
Continual performance comparison.

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9
DynDetect_{Qwen3}	84.70	85.33	77.07	77.55	73.50	75.81	77.54	74.04	77.17
DynDetect_{BERT}	**84.80**	**85.61**	**82.32**	**82.82**	**82.65**	**83.27**	**82.58**	**81.05**	**82.90**

A.3.2. Detection results of mixed perturbations

To further investigate the adaptability of detectors across different perturbations, we evaluate detectors fine-tuned on specific perturbations using text with mixed perturbations (e.g., text perturbed by both *Insert* and *Remove* together). The results are presented in Table 15.

From Table 15, we find that the performance almost drops, even when the mixed perturbations include the perturbation on which the detector has been trained. This indicates that fine-tuning detectors on specific perturbations is not sufficient, as users can still bypass detection by altering slightly the type of perturbation. While continuous detection can continuously fine-tune the detector according to the changing perturbations.

A.3.3. Meta-learning based detection

It is possible to employ the meta-learning method to detect perturbed text, and we implement the classical prototype method to explore its performance against evolving perturbed text. Specifically, we model the appeared/appearing perturbations as source/target domains, respectively. According to the prototype method, we detect the toxic text based on the distance between the source prototypes and the text from target domains. As Table 16 shows, the prototype does not perform well against evolving perturbations. We argue that the prototype method requires specifying source and target domains, and the number of domains usually is limited. When the number of domains continuously increases, the ability of the prototype method may be limited. However, in any case, meta-learning methods such as prototype provide a research direction for adapting to evolving perturbations.

A.3.4. LLM-tuned continual detection

In the field of toxicity detection, given the requirements for low-cost and high-efficiency detection, researchers prioritize the use of lightweight detection backbone networks, such as pre-trained language models like BERT. Therefore, in this work, we select BERT-base as the backbone encoder.

Furthermore, considering that fellow researchers recognize the strong capabilities of Large Language Models (LLMs) demonstrated in discriminative Natural Language Processing (NLP) tasks, we employ Qwen3-0.6B, whose the parameters are 7 times of that of BERT-base and required storage space is approximately 6 times that of BERT-base, as the backbone encoder of the detector and conduct fine-tuning for exploring its continual toxicity detection. As shown in Tables 17 and 18, we can observe that the detector based on Qwen3-0.6B not only significantly increases the number of parameters, storage costs, and training time, but also exhibits limited performance in the continual toxicity detection task. In fact, its performance is even inferior to that of the continual detector with BERT-base. We argue that although LLMs possess more powerful capabilities on discriminative task, the forgetting of the ability also becomes more severe and intractable in continual learning scenarios, which is consistent with the findings of existing studies (Kalajdzievski, 2024; Zhai et al., 2023).

A.3.5. Glossary of terms

To make readers understand our work better, the following table shows the glossary of terms that may be confusing for readers (see Table 19).

Data availability

Data will be made available on request.

Table 19
Glossary of terms.

Term	Definition
Perturbation-wise Detection	Employ detector to recognize the toxicity in every single type of perturbed text.
Cross-perturbation Detection	Employ the detectors fine-tuned on one type of perturbed text to recognize the toxicity in another unseen types of perturbed text.
Continual Detection	Enable detectors to continually detect the toxicity in the increasing types of perturbed text.

References

Akram, M. H., Shahzad, K., & Bashir, M. (2023). ISE-hate: A benchmark corpus for inter-faith, sectarian, and ethnic hatred detection on social media in Urdu. *Information Processing & Management*, *60*(3), Article 103270.

Alhazmi, A., Aljubairy, A., Zhang, W. E., Sheng, Q. Z., & Alhazmi, E. (2025). Can interpretability of deep learning models detect textual adversarial distribution? *ACM Transactions on Intelligent Systems and Technology*, *16*(4), 1–24.

Ali, T., Eleyan, A., Al-Khalidi, M., & Bejaoui, T. (2025). GAN-guarded fine-tuning: Enhancing adversarial robustness in NLP models. In *2025 5th IEEE middle east and north Africa communications conference* (pp. 1–6). IEEE.

Bespalov, D., Bhabesh, S., Xiang, Y., Zhou, L., & Qi, Y. (2023). Towards building a robust toxicity predictor. In *Proceedings of the 61st annual meeting of the Association for Computational Linguistics (volume 5: industry track)* (pp. 581–598).

Bosco, C., Patti, V., Frenda, S., Cignarella, A. T., Paciello, M., & D'Errico, F. (2023). Detecting racial stereotypes: An Italian social media corpus where psychology meets NLP. *Information Processing & Management*, *60*(1), Article 103118.

Cooper, P., Surdeanu, M., & Blanco, E. (2023). Hiding in plain sight: Tweets with hate speech masked by homoglyphs. In *Findings of the Association for Computational Linguistics: EMNLP 2023* (pp. 2922–2929).

Costa-jussà, M., Meglioli, M., Andrews, P., Dale, D., Hansanti, P., Kalbassi, E., et al. (2024). Mutox: Universal multilingual audio-based toxicity dataset and zero-shot detector. In *Findings of the Association for Computational Linguistics ACL 2024* (pp. 5725–5734).

Del Vigna12, F., Cimino23, A., Dell'Orletta, F., Petrocchi, M., & Tesconi, M. (2017). Hate me, hate me not: Hate speech detection on facebook. In *Proceedings of the first Italian conference on cybersecurity* (pp. 86–95).

Delbari, Z., Moosavi, N. S., & Pilehvar, M. T. (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*, (16), (pp. 17889–17897).

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the Association for Computational Linguistics: human language technologies, volume 1 (long and short papers)* (pp. 4171–4186).

Dixon, L., Li, J., Sorensen, J., Thain, N., & Vasserman, L. (2018). Measuring and mitigating unintended bias in text classification. In *Proceedings of the 2018 AAAI/ACM conference on AI, ethics, and society* (pp. 67–73).

Dong, J., Wang, L., Fang, Z., Sun, G., Xu, S., Wang, X., et al. (2022). Federated class-incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10164–10173).

Douillard, A., Cord, M., Ollion, C., Robert, T., & Valle, E. (2020). PODNet: Pooled outputs distillation for small-tasks incremental learning. In *European conference on computer vision* (pp. 86–102).

Emmery, C., Kádár, Á., Chrupała, G., & Daelemans, W. (2022). Cyberbullying classifiers are sensitive to model-agnostic perturbations. In *Proceedings of the thirteenth language resources and evaluation conference* (pp. 2976–2988).

Feldman, M., Friedler, S. A., Moeller, J., Scheidegger, C., & Venkatasubramanian, S. (2015). Certifying and removing disparate impact. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 259–268).

Fortuna, P., & Nunes, S. (2018). A survey on automatic detection of hate speech in text. *ACM Computing Surveys*, *51*(4), 1–30.

Fu, Z., Wang, Z., Yu, C., Xu, X., & Li, D. (2024). Double confidence calibration focused distillation for task-incremental learning. *IEEE Transactions on Neural Networks and Learning Systems*, 1–14.

Gao, Q., Shan, X., Zhang, Y., & Zhou, F. (2023). Enhancing knowledge transfer for task incremental learning with data-free subnetwork. *Advances in Neural Information Processing Systems*, *36*, 68471–68484.

Garg, T., Masud, S., Suresh, T., & Chakraborty, T. (2023). Handling bias in toxic speech detection: A survey. *ACM Computing Surveys*, *55*(13s), 1–32.

Garg, P., Saluja, R., Balasubramanian, V. N., Arora, C., Subramanian, A., & Jawahar, C. (2022). Multi-domain incremental learning for semantic segmentation. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision* (pp. 761–771).

Gongane, V. U., Munot, M. V., & Anuse, A. D. (2022). Detection and moderation of detrimental content on social media platforms: current status and future directions. *Social Network Analysis and Mining*, *12*(1), 129.

Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A., & Bengio, Y. (2013). An empirical investigation of catastrophic forgetting in gradient-based neural networks. arXiv preprint arXiv:1312.6211.

Goyal, S., Doddapaneni, S., Khapra, M. M., & Ravindran, B. (2023). A survey of adversarial defenses and robustness in nlp. *ACM Computing Surveys*, *55*(14s), 1–39.

Haber, J., Vidgen, B., Chapman, M., Agarwal, V., Lee, R. K.-W., Yap, Y. K., et al. (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)* (pp. 12705–12721).

Hanna, S., Karunaratne, S., & Cabric, D. (2022). WiSig: A large-scale WiFi signal dataset for receiver and channel agnostic RF fingerprinting. *IEEE Access*, *10*, 22808–22818.

Hanu, L., & Unitary team (2020). Detoxify. Github. https://github.com/unitaryai/detoxify.

Hartvigsen, T., Gabriel, S., Palangi, H., Sap, M., Ray, D., & Kamar, E. (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)* (pp. 3309–3326).

He, J. (2024). Gradient reweighting: Towards imbalanced class-incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 16668–16677).

Jung, H., Ju, J., Jung, M., & Kim, J. (2016). Less-forgetting learning in deep neural networks. arXiv preprint arXiv:1607.00122.

Kalajdzievski, D. (2024). Scaling laws for forgetting when fine-tuning large language models. arXiv preprint arXiv:2401.05605.
Kanakis, M., Bruggemann, D., Saha, S., Georgoulis, S., Obukhov, A., & Van Gool, L. (2020). Reparameterizing convolutions for incremental multi-task learning without task interference. In *European conference on computer vision* (pp. 689–707).
Kanca, E., Ayas, S., Baykal Kablan, E., & Ekinci, M. (2025). Evaluating and enhancing the robustness of vision transformers against adversarial attacks in medical imaging. *Medical & Biological Engineering & Computing*, *63*(3), 673–690.
Kebriaei, E., Homayouni, A., Faraji, R., Razavi, A., Shakery, A., Faili, H., et al. (2024). Persian offensive language detection. *Machine Learning*, *113*(7), 4359–4379.
Kirk, H., Vidgen, B., Röttger, P., Thrush, T., & Hale, S. (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* (pp. 1352–1368).
Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, *114*(13), 3521–3526.
Kivlichan, I., Sorensen, J., Julia Elliott, L. V., Görner, M., & Culliton, P. (2020). *Jigsaw multilingual toxic comment classification*. Kaggle.
Kowalski, R. (2018). Cyberbullying. In *The routledge international handbook of human aggression* (pp. 131–142). Routledge.
Kurita, K., Belova, A., & Anastasopoulos, A. (2019). Towards robust toxic content classification. arXiv preprint arXiv:1912.06872.
Lai, Z., Bai, H., Zhang, H., Du, X., Shan, J., Yang, Y., et al. (2024). Empowering unsupervised domain adaptation with large-scale pre-trained vision-language models. In *Proceedings of the ieee/cvf winter conference on applications of computer vision* (pp. 2691–2701).
Lashkarashvili, N., & Tsintsadze, M. (2022). Toxicity detection in online Georgian discussions. *International Journal of Information Management Data Insights*, *2*(1), Article 100062.
Le, T., Lee, J., Yen, K., Hu, Y., & Lee, D. (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* (pp. 2953–2965).
Lee, D., Yoo, M., Kim, W. K., Choi, W., & Woo, H. (2024). Incremental learning of retrievable skills for efficient continual task adaptation. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, & et al. (Eds.), *Advances in neural information processing systems*: *vol. 37*, (pp. 17286–17312). Curran Associates, Inc., URL https://proceedings.neurips.cc/paper_files/paper/2024/file/1f0832859514e53a0e4f229fc9b3a4a2-Paper-Conference.pdf.
Lees, A., Tran, V. Q., Tay, Y., Sorensen, J., Gupta, J., Metzler, D., et al. (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* (pp. 3197–3207).
Lei, B., Yu, K., Feng, M., Cui, M., & Xie, X. (2024). Diffusiongan3d: Boosting text-guided 3d generation and domain adaptation by combining 3d gans and diffusion priors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10487–10497).
Li, Z., & Hoiem, D. (2017). Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*(12), 2935–2947.
Li, C., Huang, Z., Paudel, D. P., Wang, Y., Shahbazi, M., Hong, X., et al. (2023). A continual deepfake detection benchmark: Dataset, methods, and essentials. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision* (pp. 1339–1349).
Li, X., Li, J., Du, Z., Zhu, L., & Shen, H. T. (2025). Unified modality separation: A vision-language framework for unsupervised domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
Li, S., Liu, F., Jiao, L., Li, L., Chen, P., Liu, X., et al. (2025). Prompt-based concept learning for few-shot class-incremental learning. *IEEE Transactions on Circuits and Systems for Video Technology*, *35*(5), 4991–5005. http://dx.doi.org/10.1109/TCSVT.2025.3525545.
Li, C., Song, Y., & Shao, Y.-H. (2025). Domain adaptation via learning using statistical invariant. *IEEE Transactions on Knowledge and Data Engineering*.
Li, Y., Xu, W., Qi, Y., Wang, H., Li, R., & Guo, S. (2024). Sr-fdil: Synergistic replay for federated domain-incremental learning. *IEEE Transactions on Parallel and Distributed Systems*, *35*(11), 1879–1890.
Li, Y., Xu, W., Wang, H., Qi, Y., Guo, J., & Li, R. (2024). Personalized federated domain-incremental learning based on adaptive knowledge matching. In *European conference on computer vision* (pp. 127–144). Springer.
Li, T., Zeng, Z., Li, Q., & Sun, S. (2024). Integrating GIN-based multimodal feature transformation and multi-feature combination voting for irony-aware cyberbullying detection. *Information Processing & Management*, *61*(3), Article 103651.
Liang, P. P., Wu, C., Morency, L.-P., & Salakhutdinov, R. (2021). Towards understanding and mitigating social biases in language models. In *International conference on machine learning* (pp. 6565–6576). PMLR.
Liu, R., Diao, B., Huang, L., An, Z., An, Z., & Xu, Y. (2024). Continual learning in the frequency domain. *Advances in Neural Information Processing Systems*, *37*, 85389–85411.
Liu, X., Yang, Y., He, K., & Hopcroft, J. E. (2025). Parameter interpolation adversarial training for robust image classification. *IEEE Transactions on Information Forensics and Security*.
Liu, X.-Q., Zhang, P.-F., Luo, X., Huang, Z., & Xu, X.-S. (2024). Textadapter: Self-supervised domain adaptation for cross-domain text recognition. *IEEE Transactions on Multimedia*, *26*, 9854–9865.
Lopez-Paz, D., & Ranzato, M. (2017). Gradient episodic memory for continual learning. *Advances in Neural Information Processing Systems*, *30*.
Lu, J., Xu, B., Zhang, X., Min, C., Yang, L., & Lin, H. (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)* (pp. 16235–16250).
Luo, G., Sun, J., Jin, L., Zhou, Y., Xu, Q., Fu, R., et al. (2025). Domain incremental learning for object detection. *Pattern Recognition*, Article 111882.
Maity, K., Poornash, A., Saha, S., & Bhattacharyya, P. (2024). In L.-W. Ku, A. Martins, & V. Srikumar (Eds.), *ToxVidLM: A multimodal framework for toxicity detection in code-mixed videos* (pp. 11130–11142). Bangkok, Thailand: the Association for Computational Linguistics ACL.
Markov, T., Zhang, C., Agarwal, S., Nekoul, F. E., Lee, T., Adler, S., et al. (2023). A holistic approach to undesired content detection in the real world. In *Proceedings of the AAAI conference on artificial intelligence*: *vol. 37*, (12), (pp. 15009–15018).
Masana, M., Liu, X., Twardowski, B., Menta, M., Bagdanov, A. D., & Van De Weijer, J. (2022). Class-incremental learning: survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *45*(5), 5513–5533.
Mathew, B., Saha, P., Yimam, S. M., Biemann, C., Goyal, P., & Mukherjee, A. (2021). Hatexplain: A benchmark dataset for explainable hate speech detection. In *Proceedings of the AAAI conference on artificial intelligence*: *vol. 35*, (17), (pp. 14867–14875).
MetaAI (2024). Introducing meta llama 3: The most capable openly available LLM to date. https://ai.meta.com/blog/meta-llama-3/.
Michalski, R. S., Mozetic, I., Hong, J., & Lavrac, N. (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* (pp. 1041–1045).
Mirza, M. J., Masana, M., Possegger, H., & Bischof, H. (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* (pp. 3001–3011).
Mittal, S., Galesso, S., & Brox, T. (2021). Essentials for class incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 3513–3522).
Mousa, A., Shahin, I., Nassif, A. B., & Elnagar, A. (2024). Detection of Arabic offensive language in social media using machine learning models. *Intelligent Systems with Applications*, *22*, Article 200376.
Mozhegova, E., Khattak, A. M., Khan, A., Garaev, R., & Rasheed, B. (2025). Assessing the adversarial robustness of multimodal medical AI systems: insights into vulnerabilities and modality interactions. *Frontiers in Medicine*, *12*, Article 1606238.
Mulimani, M., & Mesaros, A. (2025). Domain-incremental learning for audio classification. In *ICASSP 2025-2025 IEEE international conference on acoustics, speech and signal processing* (pp. 1–5). IEEE.
OpenAI (2022). ChatGPT: Optimizing language models for dialogue. https://openai.com/blog/chatgpt/.

Oren, G., & Wolf, L. (2021). In defense of the learning without forgetting for task incremental learning. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 2209–2218).
Pan, R., García-Díaz, J. A., Vivancos-Vicente, P. J., & Valencia-García, R. (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* (pp. 697–702).
Peng, X., Bai, Q., Xia, X., Huang, Z., Saenko, K., & Wang, B. (2019). Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 1406–1415).
Petit, G., Popescu, A., Schindler, H., Picard, D., & Delezoide, B. (2023). Fetril: Feature translation for exemplar-free class-incremental learning. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision* (pp. 3911–3920).
Rajchandar, K., Manoharan, G., & Ashtikar, S. P. (2024). Robustness in natural language processing: Addressing challenges in text-based AI systems. In *2024 11th international conference on computing for sustainable global development* (pp. 1435–1439). IEEE.
Sap, M., Gabriel, S., Qin, L., Jurafsky, D., Smith, N. A., & Choi, Y. (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* (pp. 5477–5490).
Shah, K., Sinha, A., Reddy, A., Roy, A., & Chellappa, R. (2025). DIFFUSE2ADAPT: Controlled diffusion for synthetic-to-real domain adaptation. In *2025 IEEE international conference on image processing* (pp. 2235–2240). IEEE.
Slonje, R., Smith, P. K., & Frisén, A. (2013). The nature of cyberbullying, and strategies for prevention. *Computers in Human Behavior*, *29*(1), 26–32.
Song, X., He, Y., Dong, S., & Gong, Y. (2024). Non-exemplar domain incremental object detection via learning domain bias. In *Proceedings of the AAAI conference on artificial intelligence*: *vol. 38*, (13), (pp. 15056–15065).
Sui, C., Wang, A., Wang, H., Liu, H., Gong, Q., Yao, J., et al. (2025). ISDAT: An image-semantic dual adversarial training framework for robust image classification. *Pattern Recognition*, *158*, Article 110968.
Sun, S., & Tao, Y. (2024). Research on a software system size evaluation model combining BiLSTM-CRF with domain adaptation learning. In *Proceedings of the international conference on image processing, machine learning and pattern recognition* (pp. 492–497).
Tan, D. S., Lin, Y.-X., & Hua, K.-L. (2020). Incremental learning of multi-domain image-to-image translations. *IEEE Transactions on Circuits and Systems for Video Technology*, *31*(4), 1526–1539.
Tang, Z., & Yang, Y.-B. (2024). IODA: Instance-guided one-shot domain adaptation for super-resolution. *Advances in Neural Information Processing Systems*, *37*, 117291–117314.
Vadillo, J., Santana, R., & Lozano, J. A. (2025). Adversarial attacks in explainable machine learning: A survey of threats against models and humans. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *15*(1), Article e1567.
Vallecillo-Rodríguez, M. E., Plaza-Del-Arco, F. M., & Montejo-Ráez, A. (2025). Combining profile features for offensiveness detection on Spanish social media. *Expert Systems with Applications*, Article 126705.
Van de Ven, G. M., Tuytelaars, T., & Tolias, A. S. (2022). Three types of incremental learning. *Nature Machine Intelligence*, *4*(12), 1185–1197.
Wang, Z., Li, X., Zhu, H., & Xie, C. (2024). Revisiting adversarial training at scale. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 24675–24685).
Wang, Z., Liu, Y., Ji, T., Wang, X., Wu, Y., Jiang, C., 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)* (pp. 10933–10946).
Wang, Q., Song, X., He, Y., Han, J., Ding, C., Gao, X., et al. (2025). Boosting domain incremental learning: Selecting the optimal parameters is all you need. In *Proceedings of the computer vision and pattern recognition conference* (pp. 4839–4849).
Wang, K., Zhang, G., Yue, H., Liu, A., Zhang, G., Feng, H., et al. (2024). Multi-domain incremental learning for face presentation attack detection. In *Proceedings of the AAAI conference on artificial intelligence*: *vol. 38*, (6), (pp. 5499–5507).
West, M. T., Tsang, S.-L., Low, J. S., Hill, C. D., Leckie, C., Hollenberg, L. C., et al. (2023). Towards quantum enhanced adversarial robustness in machine learning. *Nature Machine Intelligence*, *5*(6), 581–589.
Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., et al. (2019). Large scale incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 374–382).
Wu, F., Gao, B., Pan, X., Li, L., Ma, Y., Liu, S., et al. (2024). Fuser: an enhanced multimodal fusion framework with congruent reinforced perceptron for hateful memes detection. *Information Processing & Management*, *61*(4), Article 103772.
Yang, Y., Li, M., & Huang, F. (2026). EEG-DTIL: An EEG-based dynamic task-incremental learning method for decoding ADL-oriented motor imagery pairs. *Expert Systems with Applications*, [ISSN: 0957-4174] *296*, Article 128927. http://dx.doi.org/10.1016/j.eswa.2025.128927, URL https://www.sciencedirect.com/science/article/pii/S0957417425025448.
Ye, Y., Le, T., & Lee, D. (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*, (pp. 2603–2612).
Yu, S., Choi, J., & Kim, Y. (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* (pp. 3456–3467).
Zeng, J., Xu, J., Zheng, X., & Huang, X. (2023). Certified robustness to text adversarial attacks by randomized [mask]. *Computational Linguistics*, *49*(2), 395–427.
Zhai, Y., Tong, S., Li, X., Cai, M., Qu, Q., Lee, Y. J., et al. (2023). Investigating the catastrophic forgetting in multimodal large language models. In *NeurIPS 2023 workshop on instruction tuning and instruction following*.
Zhang, Z., Chen, J., & Yang, D. (2023). Mitigating biases in hate speech detection from a causal perspective. In *Findings of the Association for Computational Linguistics: EMNLP 2023* (pp. 6610–6625).
Zhang, M., He, J., Ji, T., & Lu, C.-T. (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)* (pp. 12073–12086).
Zhang, X., Hong, H., Hong, Y., Huang, P., Wang, B., Ba, Z., et al. (2024). Text-crs: A generalized certified robustness framework against textual adversarial attacks. In *2024 IEEE symposium on security and privacy* (pp. 2920–2938). IEEE.
Zhang, J., Wu, Q., Xu, Y., Cao, C., Du, Z., & Psounis, K. (2024). Efficient toxic content detection by bootstrapping and distilling large language models. In *Proceedings of the AAAI conference on artificial intelligence*: *vol. 38*, (19), (pp. 21779–21787).
Zhang, J., Zhang, J., Ghosh, S., Li, D., Tasci, S., Heck, L., et al. (2020). Class-incremental learning via deep model consolidation. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision* (pp. 1131–1140).
Zhao, W., Cui, Y., & Hu, W. (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)* (pp. 1162–1175).
Zhao, H., Ma, C., Dong, X., Luu, A. T., Deng, Z.-H., & Zhang, H. (2022). Certified robustness against natural language attacks by causal intervention. In *International conference on machine learning* (pp. 26958–26970). PMLR.
Zhou, D.-W., Cai, Z.-W., Ye, H.-J., Zhang, L., & Zhan, D.-C. (2025). Dual consolidation for pre-trained model-based domain-incremental learning. In *Proceedings of the computer vision and pattern recognition conference* (pp. 20547–20557).
Zhu, F., Cheng, Z., Zhang, X.-y., & Liu, C.-l. (2021). Class-incremental learning via dual augmentation. *Advances in Neural Information Processing Systems*, *34*, 14306–14318.
Zühlke, M.-M., & Kudenko, D. (2025). Adversarial robustness of neural networks from the perspective of lipschitz calculus: A survey. *ACM Computing Surveys*, *57*(6), 1–41.