GENERALIZING MACHINE-GENERATED TEXT DETECTION TO UNSEEN GENERATORS VIA HARMFUL SUBSPACE REMOVAL

Anonymous authors
Paper under double-blind review

ABSTRACT

Machine-generated text (MGT) detection is essential for identifying the provenance of online content, yet existing detectors often generalize poorly to unseen generators. We find that this failure is closely tied to generator-specific shortcuts: detectors trained for human–AI classification encode strong generator identity information, and their cross-generator transfer patterns are highly structured rather than uniformly transferable. A natural solution is to identify and remove generator-cue tokens, but our analysis shows that token-level deletion is ineffective because cue-bearing tokens also contain useful human–AI discrimination evidence. Motivated by this observation, we propose Harmful **C**ue s**U**bspace **R**emoval with R**E**play Pool (CURE), a representation-level purification framework for robust MGT detection under unseen-generator shifts. CURE first uses a frozen generator-ID teacher to estimate token-level generator cues, then maintains a three-stream replay pool to construct a low-rank harmful cue subspace by contrasting harmful-cue and clean-reference representations. Instead of dropping tokens, CURE selectively removes generator-specific directions inside token hidden states with confidence-guided strength, while training the detector on both original and purified views. Experiments on 12 unseen generators show that CURE achieves an F1 score of **89.9**, outperforming 8 representative detectors by **8.5** points. Further analyses demonstrate that subspace-level removal is substantially more effective than token-level deletion, suppresses generator-specific information while preserving human–AI discrimination signals. We will release our code and datasets upon acceptance.

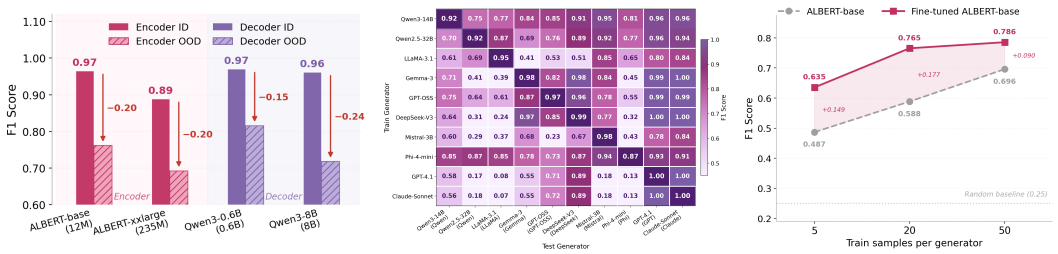
1 INTRODUCTION

The rapid advancement of large language models (LLMs), such as ChatGPT Achiam et al. (2023), LLaMA Dubey et al. (2024), and DeepSeek Guo et al. (2025), has made it increasingly easy to produce fluent and highly human-like text at scale. While these models have benefited a wide range of real-world applications, their misuse for LLM-written peer reviews, phishing emails, fake news, and other deceptive content has raised growing concerns. In response, recent studies have developed machine-generated text (MGT) detection methods Liu et al. (2022; 2024); Bao et al. (2024); Hans et al. (2024); Li et al. (2025; 2026) to determine the provenance of a given passage and help readers distinguish human-written text from machine-generated content.

Despite the growing number of proposed detectors, their generalization to unseen generators remains a persistent and challenging problem. An increasing number of benchmarks Guo et al. (2023); Dugan et al. (2024); Li et al. (2024); Wang et al. (2024b) have shown that existing detectors, regardless of their detection paradigm, suffer substantial performance degradation when evaluated on generators that are absent from the training set. As shown in Figure 1(a), the detection F1 score drops by approximately 15–24 percentage points on unseen generators, irrespective of the detector scale or architecture. This considerably undermines the reliability of MGT detection, especially in the face of the rapidly expanding landscape of newly emerging LLMs.

To further understand why detectors fail to generalize to unseen generators, we examine whether their decisions are driven by transferable evidence of machine generation or by generator-dependent shortcuts. If detectors had learned generator-invariant traces, a model trained on one generator

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Generalization gap across detectors (b) Pairwise transfer performance (c) Linear probing results

Figure 1: Empirical motivation of our method. (a) We compare representative MGT detectors on seen generators and unseen generators, showing that all detectors suffer a clear ID–OOD performance drop regardless of model scale or architecture. (b) We train detectors on one generator and evaluate them on other generators to measure pairwise transferability, revealing that generator shift is structured and that transfer failures concentrate across certain generator groups. (c) We freeze the detector representations and train lightweight linear probes to predict generator identity, showing that generator-specific information is strongly encoded in the learned representation space.

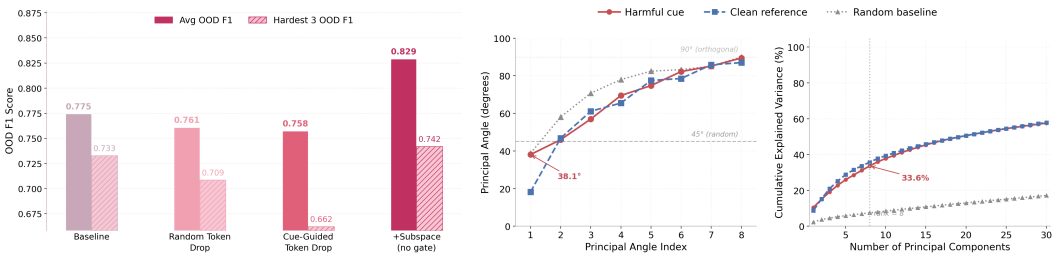
would be expected to transfer reasonably well to other generators. However, the pairwise transfer results in Figure 1(b) show a markedly different pattern: transfer performance is highly uneven and forms clear generator-dependent structures, where detectors generalize well only to a limited set of related generators but fail severely on others. This suggests that the learned decision boundary is not primarily supported by universal evidence of machine generation, but is instead strongly coupled with generator-specific regularities.

We further examine whether such regularities are encoded in the learned representations. As shown in Figure 1(c), lightweight linear probes can recover generator identity from frozen detector features with surprisingly few training examples. With only five probing samples per generator, the generator-ID F1 already reaches 0.635 on the fine-tuned detector, substantially exceeding both the random baseline of 0.25 and the pretrained encoder counterpart of 0.487. As the number of probing samples increases, the same trend persists, with the fine-tuned detector consistently yielding higher generator-ID separability than the pretrained encoder. This indicates that human-AI binary training not only preserves generator-related information but also makes generator identity more linearly separable in the representation space. Together, these observations suggest that the ID–OOD degradation arises from the detector's reliance on generator-specific cues. This motivates us to explicitly identify and suppress such cues, encouraging the detector to rely more on generator-invariant evidence of machine-generated text.

Having shown that detector representations encode generator identity, we next examine how such generator-specific cues can be suppressed. Following prior studies on generator attribution Li et al. (2025; 2026), we train an auxiliary generator classifier and use token-level gradients to identify tokens that contribute most to generator prediction. A natural strategy is then to remove these cue-bearing tokens during training.

However, Figure 2(a) shows that token dropping is ineffective: both random token dropping and cue-guided token dropping fail to improve OOD generalization, and may even hurt performance. This suggests that the identified tokens are not purely harmful. Instead, they may encode both generator-specific shortcuts and useful human–AI discrimination evidence, so deleting the entire token inevitably removes task-relevant information as well.

This observation leads us to examine the internal structure of these token representations rather than treating each token as an indivisible unit. Specifically, we collect two sets of token hidden states from the trained detector: harmful-cue tokens, identified as the top-15% tokens by gradient-based saliency from a generator classifier, and clean-reference tokens, drawn from the remaining low-saliency positions. As shown in Figure 2(b), the first principal angle between the two PCA subspaces is 38.1°, below the random baseline of 45°, indicating a non-trivial shared direction; however, higher-order directions quickly approach orthogonality, with the eighth angle reaching 89.4°. Meanwhile, the top-8 principal components explain 33.6% and 35.7% of the variance for harmful-cue and clean-reference representations, respectively, far above the random baseline of 7.5%, showing that both sets exhibit compact low-rank structures. Consistently, Figure 2(a) shows that cue-guided subspace removal improves OOD F1 from 0.775 to 0.829, whereas random token dropping and cue-guided token



(a) Token intervention analysis

(b) Subspace geometry analysis

Figure 2: Mechanistic analysis of token-level interventions. (a) During detector training, we apply different cue-suppression strategies to the selected tokens, including random token dropping, cue-guided token dropping, and cue-guided subspace removal, and then evaluate the trained detector on unseen generators. This comparison examines whether generator-specific cues can be handled by directly deleting token-level information, or require structured removal in the representation space. (b) We collect harmful-cue token representations and clean-reference token representations, apply PCA to each representation pool, and compare the resulting low-dimensional subspaces using principal angles and cumulative explained variance. This analysis characterizes the low-rank structure of cue-related directions and their geometric separation from clean reference directions.

dropping decrease it to 0.761 and 0.758. These results indicate that suppressing generator-specific directions inside token representations is substantially more effective than removing the tokens themselves.

Motivated by these findings, we propose CURE, a framework for MGT detection under unseen-generator shifts. Instead of removing cue-bearing tokens entirely, CURE identifies generator-related tokens with a pretrained generator classifier and suppresses only their generator-specific components in the hidden representation space. To obtain a stable estimate of such components, CURE maintains a replay pool of broad harmful cues, hard harmful cues, and clean-reference tokens, from which a low-rank harmful cue subspace is constructed by contrasting harmful and clean covariance structures. The detector then removes the projected cue directions from token representations with an adaptive confidence-controlled strength, while being trained on both original and purified views. This design preserves task-relevant human–AI evidence while weakening generator-dependent shortcuts, enabling stronger generalization to unseen generators.

Our Work. In this paper, we propose Harmful **C**ue s**U**bspace **R**emoval with R**E**play Pool (CURE), a framework for robust machine-generated text detection under unseen-generator shifts. CURE is built upon a key observation: generator-specific cues are not cleanly isolated in discrete tokens, but are entangled with useful human–AI discrimination evidence inside token representations. Therefore, instead of deleting cue-bearing tokens, CURE identifies generator-related tokens and suppresses only their generator-specific components in the hidden representation space. To this end, CURE first estimates token-level cue saliency with a pretrained generator classifier. A sample-level confidence gate further down-weights cue estimates when the teacher's prediction confidence is low, preventing unreliable saliency from corrupting subspace estimation on unseen generators. CURE then maintains a three-stream replay pool containing broad harmful-cue representations, hard harmful-cue representations, and clean-reference representations. By contrasting the covariance structure of harmful-cue pools against clean-reference directions, CURE constructs a low-rank harmful-cue subspace and removes the projected cue components from token hidden states with confidence-aware strength. The detector is trained on both original and purified views, forcing it to preserve task-relevant human–AI evidence while reducing its dependence on generator-specific shortcuts. Experiments on 12 unseen generators show that CURE achieves an F1 score of 89.9, outperforming 8 representative detectors by 8.5 points, and further analysis confirms that subspace-level removal is substantially more effective. Our contributions are as follows:

- **Representation-level Cue Purification Framework.** We propose CURE, a token-aware representation purification framework for robust MGT detection under unseen-generator shifts. Instead of directly dropping cue-bearing tokens, CURE identifies generator-related tokens and removes only their generator-specific components in the hidden representation space, thereby suppressing shortcut cues while preserving human–AI discriminative evidence.

- **Harmful Cue Subspace Removal with Replay Pool.** We introduce a replay-based harmful cue subspace estimation strategy. Specifically, CURE uses a pretrained generator classifier to estimate token-level cue saliency, applies a confidence gate to reduce unreliable cue estimates, and maintains a three-stream replay pool containing broad harmful cues, hard harmful cues, and clean-reference representations. By contrasting harmful and clean covariance structures, CURE constructs a low-rank cue subspace and removes the projected generator-specific directions from token representations.
- **Strong OOD Generalization.** Extensive experiments on unseen generators show that CURE achieves an F1 score of 89.9, outperforming 8 representative detectors by 8.5 points. Further analysis demonstrates that subspace-level removal is substantially more effective than token-level deletion, validating that generator-specific cues should be suppressed within token representations rather than removed as entire tokens.

2 RELATED WORK

Machine-Generated Text Detection. Machine-generated text (MGT) detection aims to distinguish human-written text from content produced by language models. Existing methods can be broadly divided into metric-based and model-based detectors. Metric-based detectors rely on statistical signals from language models, such as likelihood, token-rank, curvature, or distributional discrepancy, and usually require little or no supervised training Su et al. (2023); Bao et al. (2024); Hans et al. (2024); Xu et al. (2024); Zhu et al. (2025). Model-based detectors instead fine-tune pretrained classifiers to learn discriminative human–machine representations, often achieving strong in-domain performance with labeled data Liu et al. (2022); Guo et al. (2024); Li et al. (2025); Hu et al. (2023); Li et al. (2026). However, an increasing number of benchmarks Guo et al. (2023); Dugan et al. (2024); Li et al. (2024); Wang et al. (2024a;b) show that both paradigms still face a major generalization challenge: detectors trained or calibrated on specific generators can suffer substantial degradation when evaluated on unseen LLMs, new domains, or different datasets. This suggests that many detectors may capture generator-specific artifacts rather than transferable evidence of machine generation. Our work addresses this limitation by explicitly reducing reliance on generator-specific cues and encouraging more generator-invariant representations for robust MGT detection.

3 METHODOLOGY

In this section, we introduce our CURE framework for improving the unseen-generator generalization of machine-generated text detection. The pipeline of training stage of CURE is shown in Figure 3. Our key idea is to suppress generator-specific cues at the representation level rather than deleting entire cue-bearing tokens. In the following subsections, we first describe the training-stage cue purification process, including generator-cue estimation, replay-based harmful cue subspace estimation, and confidence-guided subspace removal. We then introduce the training objective and finally describe the detection procedure at inference time.

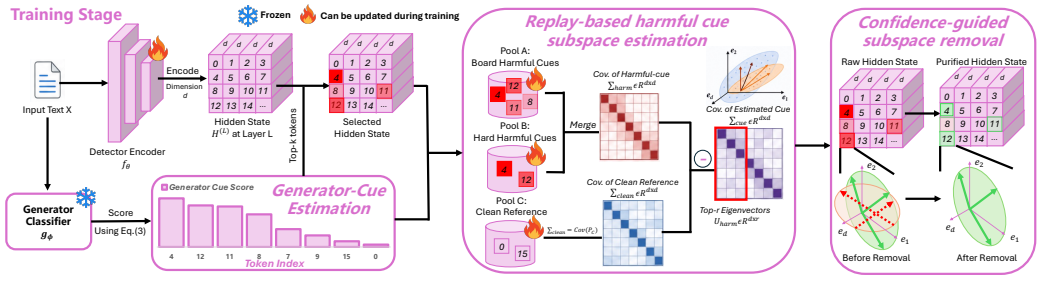


Figure 3: **Training stage pipeline of CURE.** During training, CURE first uses a frozen generator-ID teacher to estimate generator cues, then estimates the harmful cue subspace with a replay pool, and applies confidence-guided subspace removal to obtain purified hidden states. The detector is trained on both the original and purified views, so that it learns to preserve human–AI discrimination ability while reducing its reliance on generator-specific cues.

3.1 TRAINING-STAGE CUE PURIFICATION

Given an input text $x = \{w_1, \ldots, w_n\}$ and its binary label $y \in \{0, 1\}$, where $y = 1$ denotes machine-generated text and $y = 0$ denotes human-written text, the detector f_θ encodes the input into contextual token representations. We denote the token hidden states at the selected removal layer L as

$$\mathbf{H}^{(L)} = [\mathbf{h}_1^{(L)}, \mathbf{h}_2^{(L)}, \ldots, \mathbf{h}_n^{(L)}], \qquad \mathbf{h}_t^{(L)} \in \mathbb{R}^d. \tag{1}$$

The goal of CURE is to remove generator-specific components from these token representations while preserving the information required for human–AI classification.

Generator-cue estimation. Before suppressing generator-specific information, we first need to identify where such information is likely to appear. A naive solution is to treat all tokens equally, but this would unnecessarily perturb many task-relevant tokens. Instead, we estimate token-level generator cues by asking which tokens are most responsible for predicting the source generator. To this end, we use a pretrained generator classifier g_ϕ as a frozen teacher. Given an input x, the teacher predicts a generator distribution $p_\phi(g \mid x)$, and we denote its top-1 generator prediction as

$$\hat{g} = \arg\max_g p_\phi(g \mid x). \tag{2}$$

For each token embedding $\mathbf{e}_t$, we compute a gradient-based cue saliency score:

$$s_t = \left\| \nabla_{\mathbf{e}_t} \log p_\phi(\hat{g} \mid x) \odot \mathbf{e}_t \right\|_2, \tag{3}$$

where $\odot$ denotes element-wise multiplication. A larger s_t indicates that the token contributes more strongly to the teacher's generator prediction, and is therefore more likely to contain generator-specific information.

However, the teacher can be unreliable for ambiguous samples or unseen-generator texts. If such unreliable predictions are used directly, the resulting cue scores may guide the model to remove useful detection information. We therefore compute a sample-level confidence signal using the margin between the top-1 and top-2 generator probabilities. Let g_1 and g_2 be the top-1 and top-2 predicted generator classes. The confidence margin is

$$m(x) = p_\phi(g_1 \mid x) - p_\phi(g_2 \mid x). \tag{4}$$

The confidence weight is then computed as

$$\omega(x) = \max\left(\omega_{\min}, \min\left(1, \frac{m(x)}{\tau}\right)\right), \tag{5}$$

where τ is the confidence threshold and $\omega_{\min}$ prevents the cue signal from being completely suppressed. The gated cue score is defined as

$$\tilde{s}_t = \omega(x) s_t. \tag{6}$$

The gated cue scores are then used to select a sparse set of cue-bearing tokens. Specifically, we construct a binary cue-token mask:

$$q_t = \mathbb{I}\left[\tilde{s}_t \in \text{Top-}k\%\right], \tag{7}$$

where $q_t = 1$ means that token t is selected as a cue-bearing token and is eligible for subspace removal. Tokens with $q_t = 0$ are not directly modified during the removal step. Therefore, CURE uses cue scores in two stages: first, to select a sparse set of top-$k\%$ cue-bearing tokens for intervention, and second, to modulate the removal strength within this selected set.

However, token saliency alone is not sufficient. As shown by our token intervention analysis in Figure 2(a), directly dropping cue-bearing tokens does not improve OOD generalization, indicating that these tokens also contain useful human–AI discrimination evidence. Therefore, we use cue estimation only to localize suspicious token representations, and perform removal inside their representation subspace rather than deleting the tokens themselves.

Replay-based harmful cue subspace estimation. After locating cue-bearing tokens, the next question is what part of their representation should be removed. Since generator-specific cues may be distributed across hidden dimensions and vary across mini-batches, estimating the harmful directions

from a single batch can be noisy. We therefore maintain a replay pool to accumulate stable cue-related representation statistics over training. Specifically, CURE maintains three streams of token representations, $\mathcal{P}_A$, $\mathcal{P}_B$ and $\mathcal{P}_C$, where $\mathcal{P}_A$ stores broad harmful-cue representations, selected from the top cue-scored tokens in each sample; $\mathcal{P}_B$ stores hard harmful-cue representations, selected from tokens whose cue scores exceed a moving statistical threshold; and $\mathcal{P}_C$ stores clean-reference representations, collected from low-saliency tokens in human-written samples. Formally, for each mini-batch, we update

$$\mathcal{P}_A \leftarrow \mathcal{P}_A \cup \{\mathbf{h}_t^{(L)} \mid q_t = 1\}, \tag{8}$$

$$\mathcal{P}_B \leftarrow \mathcal{P}_B \cup \{\mathbf{h}_t^{(L)} \mid \tilde{s}_t > \mu_s + k\sigma_s\}, \tag{9}$$

where μ_s and σ_s are running statistics of gated cue scores. For the clean-reference pool, we collect low-saliency token representations from human-written texts:

$$\mathcal{P}_C \leftarrow \mathcal{P}_C \cup \{\mathbf{h}_t^{(L)} \mid y = 0,\ \tilde{s}_t \in \text{Bottom-}k\%\}. \tag{10}$$

At the end of each epoch, we estimate the harmful cue subspace from the replay pools. We compute the covariance matrices of harmful-cue and clean-reference representations:

$$\mathbf{\Sigma}_{\text{harm}} = \text{Cov}(\mathcal{P}_A \cup \mathcal{P}_B), \qquad \mathbf{\Sigma}_{\text{clean}} = \text{Cov}(\mathcal{P}_C). \tag{11}$$

To avoid confusing clean semantic directions with harmful generator cues, we construct a contrastive covariance matrix:

$$\mathbf{\Sigma}_{\text{cue}} = \mathbf{\Sigma}_{\text{harm}} - \eta\mathbf{\Sigma}_{\text{clean}}, \tag{12}$$

where η controls the suppression of clean-reference directions. We then take the top-r eigenvectors associated with the largest positive eigenvalues:

$$\mathbf{U}_{\text{harm}} = \text{Top-}r\text{-EigVecs}\left(\mathbf{\Sigma}_{\text{cue}}\right), \qquad \mathbf{U}_{\text{harm}} \in \mathbb{R}^{d\times r}, \tag{13}$$

where d is the hidden dimension of the detector encoder at layer L. This subspace captures directions that are prominent in harmful-cue representations but suppressed in clean-reference representations. We further validate the geometric meaningfulness and directional selectivity of this subspace in Section B.3.

Confidence-guided subspace removal. Even with a learned harmful cue subspace, applying the same removal strength to all selected tokens can be risky. Some selected tokens may contain stronger generator-specific information than others, and the reliability of the teacher also differs across samples. Therefore, CURE removes the harmful subspace projection only from the selected cue-bearing tokens, while further modulating the removal strength by confidence and normalized cue saliency.

Given $\mathbf{U}_{\text{harm}}$, we remove the projection of each selected token representation onto the harmful cue subspace:

$$\tilde{\mathbf{h}}_t^{(L)} = \mathbf{h}_t^{(L)} - q_t\rho_t\mathbf{U}_{\text{harm}}\mathbf{U}_{\text{harm}}^{\top}\mathbf{h}_t^{(L)}. \tag{14}$$

Because of the mask q_t, tokens outside the top-$k\%$ cue-bearing set remain unchanged. The token-specific removal coefficient is defined as

$$\rho_t = \rho_{\text{eff}}(x) \cdot \text{Norm}(\tilde{s}_t), \tag{15}$$

where $\text{Norm}(\cdot)$ normalizes cue scores within each sample. The effective removal strength is clipped into a stable range:

$$\rho_{\text{eff}}(x) = \text{clip}\left(\omega(x) \cdot \frac{\mu_{\tilde{s}}}{\sigma_{\tilde{s}} + \epsilon}, \rho_{\min}, \rho_{\max}\right), \tag{16}$$

where $\mu_{\tilde{s}}$ and $\sigma_{\tilde{s}}$ are the mean and standard deviation of gated cue scores. In this way, subspace removal is applied only to the selected top-$k\%$ cue-bearing tokens, while tokens outside this set are preserved. Among the selected tokens, reliable high-cue tokens receive stronger purification, whereas low-confidence samples receive weaker removal. The effectiveness of the confidence gate is empirically analyzed in Section 4.3, where we show that it prevents harmful over-removal when teacher cue estimates are unreliable.

Training objective. After obtaining the purified hidden states, the detector is trained to make correct predictions on both the original and purified views. This design prevents the detector from relying solely on the purified representation, especially in the early training stage when the cue subspace

is still being estimated. The original view preserves the standard supervised learning signal, while the purified view encourages the detector to remain accurate after generator-specific directions are removed.

Let

$$p_\theta^{\mathrm{raw}}(y \mid x) = f_\theta(x), \qquad p_\theta^{\mathrm{pur}}(y \mid x) = f_\theta(\tilde{x}), \tag{17}$$

where $\tilde{x}$ denotes the forward pass whose layer-L hidden states are purified by Eq. (14). For a training batch $\mathcal{B}$, the total training objective is

$$\mathcal{L} = \frac{1}{|\mathcal{B}|} \sum_{(x_i, y_i) \in \mathcal{B}} \left[-\log p_\theta^{\mathrm{raw}}(y_i \mid x_i) - \log p_\theta^{\mathrm{pur}}(y_i \mid x_i) \right]. \tag{18}$$

This objective keeps the framework simple and avoids introducing additional contrastive or adversarial losses. The ablation study in Section 4.2 shows that the full training design achieves the strongest OOD performance compared with token dropping and subspace removal without the confidence-gated mechanism.

3.2 Detection Stage

During inference, CURE uses the harmful cue subspace estimated during training to purify token representations. Given an input text x, the frozen generator teacher first computes token-level cue scores, sample-level confidence, and the top-$k\%$ cue-token mask. The detector then performs a forward pass with the same layer-L purification hook used during training, producing the purified prediction $p_\theta^{\mathrm{pur}}(y \mid x)$.

We use both the original and purified views at inference time. This is because subspace removal can suppress generator-specific shortcuts, but the original view may still preserve complementary task-relevant evidence. Therefore, we average the two prediction distributions:

$$p_\theta^{\mathrm{final}}(y \mid x) = \frac{1}{2} \left[p_\theta^{\mathrm{raw}}(y \mid x) + p_\theta^{\mathrm{pur}}(y \mid x) \right]. \tag{19}$$

The final detection result is

$$\hat{y} = \arg\max_y p_\theta^{\mathrm{final}}(y \mid x). \tag{20}$$

This dual-view prediction reduces the risk of over-relying on a single representation view and yields more stable detection under unseen-generator shifts. We present the training and detection procedure of CURE in Algorithm 1.

4 Experiment Results

We evaluate CURE on unseen generators, including ImBD and M-RangeDetector comparisons.

4.1 Detection Performance on Unseen Generators

Experimental setup. We evaluate CURE under a strict unseen-generator setting. We construct the training set from abstracts in the arXiv dataset, where the machine-generated texts are produced by four seen generators: Qwen/Qwen3-4B Team (2025), Qwen/Qwen3-235B-A22B Team (2025), google/gemma-2b-it Team et al. (2023), and openai/gpt-oss-20b OpenAI (2025). None of these generators appears in the OOD test set. Before detector training, we first train a generator classifier g_ϕ on the generated training samples to predict their source generator among the four seen generators. The trained g_ϕ is then frozen and used as a teacher for CURE. For evaluation, we build an OOD test set with 12 unseen generators, including meta-llama/Llama-3.1-8B-Instruct, Qwen/Qwen3-14B Team (2025), Qwen/Qwen2.5-32B-Instruct Yang et al. (2024), mistralai/Mistral-Small-3.1-24B-Instruct-2503, microsoft/Phi-4-mini-instruct Abouelenin et al. (2025), google/gemma-4-26B-A4B-it Team et al. (2023), DeepSeek V3.2, GPT-4.1, Claude 4.6 Sonnet, Moonshot-v1-8k, openai/gpt-oss-120b OpenAI (2025), and Gemini 3 Flash Team et al. (2023). Detailed information about the dataset can be found in Appendix A.1. The per-generator comparison in Table 1 includes eight established detectors: Binoculars Hans et al. (2024), Fast-DetectGPT Bao et al. (2024), LastDE Xu et al. (2024), ALBERT-base, CoCo Liu et al. (2022), TASTE Li et al. (2026), GREATER Li et al. (2025), and DeTeCtive Guo et al. (2024). For this comparison, model-based methods use ALBERT-base and metric-based methods use Qwen/Qwen3-8B. We additionally compare ImBD Chen et al. (2025) and

Table 1: Performance of different detectors on unseen generators. We report accuracy (Acc) and F1 score (F1) on 12 unseen generators. OOD Avg denotes the average over all unseen generators, and Hardest-3 denotes the average over the three most challenging generators, i.e., Phi-4, Gemma, and LLaMA. The best results are highlighted in **bold**, and the second-best results are underlined.

Generator	Metric	Metric-based Detectors			Model-based Detectors					
		Binoculars	FastDetectGPT	LastDE	ALBERT-base	CoCo	TASTE	GREATER	DeTeCtive	CURE
LLaMA3-8B	*Acc* ↑	76.4	73.0	69.4	73.5	67.5	73.0	79.4	80.2	**89.0**
	F1 ↑	75.8	72.5	69.3	71.9	62.2	69.5	80.1	79.5	**89.0**
Qwen3-14B	*Acc* ↑	84.7	81.2	78.2	77.3	73.0	75.6	82.9	85.5	**90.7**
	F1 ↑	84.2	80.6	78.1	76.9	70.4	73.3	83.4	85.0	**91.0**
Qwen2.5-32B	*Acc* ↑	83.1	79.8	76.7	74.1	71.5	72.0	77.5	83.9	**88.4**
	F1 ↑	82.5	79.1	76.6	72.6	68.0	67.9	77.8	83.4	**88.6**
Mistral	*Acc* ↑	78.9	75.7	72.8	84.8	76.6	82.1	85.7	82.6	**92.0**
	F1 ↑	78.3	75.0	72.6	85.6	75.5	81.6	86.5	82.0	**92.4**
Phi-4	*Acc* ↑	73.2	70.5	67.0	71.9	66.6	69.5	72.4	77.5	**82.6**
	F1 ↑	72.6	69.8	66.9	70.3	61.7	64.7	71.6	76.8	**81.9**
Gemma4-26B-A4B	*Acc* ↑	77.8	74.1	70.8	72.3	68.4	72.3	75.9	81.0	**88.6**
	F1 ↑	77.0	73.2	70.6	70.2	63.8	68.4	75.1	80.3	**88.9**
DeepSeek V3.2	*Acc* ↑	80.6	77.4	74.0	80.2	76.5	79.7	78.9	83.1	**90.4**
	F1 ↑	80.0	76.8	73.9	79.9	74.3	78.1	78.7	82.5	**90.7**
GPT-4.1	*Acc* ↑	75.5	72.9	69.5	76.7	71.2	75.6	80.6	80.8	**91.8**
	F1 ↑	74.8	72.1	69.4	76.7	68.3	73.8	81.7	80.1	**92.2**
Claude 4.6 Sonnet	*Acc* ↑	76.2	73.8	70.3	79.9	78.5	79.0	81.9	82.0	**92.7**
	F1 ↑	75.5	73.0	70.2	82.8	81.1	80.7	84.3	81.4	**93.1**
Moonshot v1	*Acc* ↑	74.7	72.0	68.4	75.5	74.4	75.0	76.9	79.6	**86.4**
	F1 ↑	74.0	71.3	68.4	74.8	72.1	72.9	76.5	78.9	**86.4**
GPT-OSS 120B	*Acc* ↑	84.1	80.7	77.6	89.9	91.1	92.5	88.2	86.0	**93.2**
	F1 ↑	83.6	80.1	77.6	90.8	91.8	93.0	89.4	85.5	**93.6**
Gemini 3 Flash	*Acc* ↑	76.5	74.0	70.2	75.9	69.8	75.0	80.1	81.8	**90.7**
	F1 ↑	75.9	73.3	70.1	77.0	68.8	74.7	81.7	81.2	**91.1**
OOD Avg	*Acc* ↑	78.5	75.4	72.1	77.7	73.8	76.8	80.0	82.0	**89.7**
	F1 ↑	77.9	74.7	72.0	77.5	71.5	74.9	80.6	81.4	**89.9**
Hardest-3	*Acc* ↑	75.8	72.5	69.1	72.6	67.5	71.6	75.9	79.6	**86.7**
	F1 ↑	75.1	71.8	68.9	70.8	62.6	67.5	75.6	78.9	**86.6**

M-RangeDetector Jiao et al. (2025) on the same 12 unseen generators over five seeds; results are given below and in Appendix B.1.

Experiment results. Table 1 reports the results on 12 unseen generators. We highlight three observations. **1) Best overall OOD performance.** CURE achieves the highest OOD average, with **89.7** accuracy and **89.9** F1. It improves over the best metric-based detector, Binoculars, by **+11.2** accuracy and **+12.0** F1, and over the strongest model-based baseline, DeTeCtive, by **+7.7** accuracy and **+8.5** F1. CURE also ranks first on all 12 unseen generators, showing consistent gains across OOD sources. In the five-seed comparison on the same 12-generator split, CURE scores 88.9 ± 0.8 F1 versus 82.5 ± 3.4 for ImBD and 79.3 ± 1.5 for M-RangeDetector (Appendix B.1). **2) Larger gains on challenging generators.** On Hardest-3, including Phi-4, Gemma, and LLaMA, CURE reaches **86.7** accuracy and **86.6** F1, outperforming the best metric-based detector by **+10.9** accuracy and **+11.5** F1, and the strongest model-based baseline by **+7.1** accuracy and **+7.7** F1. This supports our claim that OOD failures largely stem from generator-specific shortcuts rather than insufficient detector capacity alone. **3) Consistent generalization across generator families.** CURE improves over the second-best detector by **+11.0/+10.5** Acc/F1 on GPT-4.1 and **+10.7/+8.8** Acc/F1 on Claude 4.6 Sonnet. It also yields large F1 gains on open-source models, including LLaMA3-8B (**+8.9**) and Gemma4-26B-A4B (**+8.6**). These results indicate that suppressing harmful generator-cue directions helps the detector focus on transferable MGT evidence. Additional unseen-domain and unseen-dataset results are provided in Appendix B.4, and we also report the inference latency in Appendix B.6.

4.2 ABLATION STUDY

Experimental setup. We conduct ablation studies to examine the contribution of each component in CURE. Starting from the ALBERT-base detector as the baseline, we compare four variants: Random Drop, which randomly removes token representations during training; Cue Drop, which removes tokens selected by generator-cue saliency; +Subspace, which applies cue-guided subspace removal without the full confidence-gated mechanism; and the complete CURE, which further incorporates confidence-gated cue estimation. All variants are evaluated on the same 12 unseen generators using F1 score.

Experiment results. Table 2 reports the ablation results. We summarize three findings. **1) Token-level deletion is insufficient.** Both token-dropping variants degrade overall OOD performance. Compared with the baseline average F1 of **0.775**, Random Drop decreases F1 to **0.761**, and Cue Drop further drops to **0.758**. This supports our motivation that generator-specific cues are not localized in entire tokens: cue-bearing tokens also contain useful human–AI discrimination evidence, so directly deleting them can remove task-relevant information. **2) Subspace removal is the key contributor.** Suppressing cues in the representation subspace improves the average OOD F1 from **0.775** to **0.829**, yielding a gain of **+0.054**. Unlike token deletion, +Subspace improves over the baseline on 10 out of 12 unseen generators, with clear gains on LLaMA3-8B (**0.719**→**0.768**), Gemma4-26B-A4B (**0.702**→**0.817**), GPT-4.1 (**0.767**→**0.882**), and Gemini 3 Flash (**0.770**→**0.862**). This shows that removing harmful generator-cue directions within token representations is more effective than discarding tokens. **3) Confidence gating further stabilizes cue suppression.** The full CURE achieves the best average F1 of **0.899**, improving over +Subspace by **+0.070** and over the baseline by **+0.124**. It obtains the best performance on all 12 unseen generators, showing that confidence gating improves both average performance and consistency. The gains are especially large on challenging cases such as LLaMA3-8B (**+0.171**), Qwen2.5-32B (**+0.160**), Phi-4 (**+0.116**), and Gemma4-26B-A4B (**+0.187**). These results indicate that confidence gating helps avoid unreliable cue estimates and over-removal, enabling CURE to suppress harmful cue directions while preserving task-relevant MGT evidence.

Table 2: Ablation study of CURE. We report F1 scores on 12 unseen generators. Random Drop randomly removes tokens during training; Cue Drop removes tokens selected by generator-cue saliency; +Subspace applies cue-guided subspace removal without the confidence-gated replay mechanism; CURE denotes the full model. The best results are highlighted in **bold**.

Generator	Baseline	Random Drop	Cue Drop	+Subspace	+Conf Gate (CURE)
LLaMA3-8B	0.719	0.733	0.658	0.768	**0.890**
Qwen3-14B	0.769	0.774	0.766	0.823	**0.910**
Qwen2.5-32B	0.726	0.704	0.633	0.703	**0.886**
Mistral	0.856	0.814	0.843	0.896	**0.924**
Phi-4	0.703	0.649	0.588	0.699	**0.819**
Gemma4-26B-A4B	0.702	0.710	0.627	0.817	**0.889**
DeepSeek V3.2	0.799	0.758	0.757	0.847	**0.907**
GPT-4.1	0.767	0.802	0.838	0.882	**0.922**
Claude 4.6 Sonnet	0.828	0.839	0.909	0.915	**0.931**
Moonshot v1	0.748	0.732	0.788	0.809	**0.864**
GPT-OSS 120B	0.908	0.850	0.930	0.929	**0.936**
Gemini 3 Flash	0.770	0.767	0.756	0.862	**0.911**
OOD Avg	0.775	0.761	0.758	0.829	**0.899**

4.3 IMPACT OF CONFIDENCE GATE

To analyze when subspace removal helps or hurts, we conduct a bucketed analysis on 8 unseen generators. For each generator, we sample 150 AI texts and the same number of human texts, estimate cue strength using the signal-to-noise ratio of teacher cue scores, and divide AI samples into five equal-sized buckets. We then compare AI recall before and after subspace removal. Table 3 shows that subspace removal is reliability-sensitive. In the Very Low bucket, where teacher confidence is the lowest (0.464), AI recall drops from 0.971 to 0.846 after purification, decreasing by 12.5 percentage points. This bucket also has the largest average gap of 42.51, suggesting that unreliable teacher cues can induce erroneous subspace estimation and remove task-relevant MGT evidence. When teacher cues are more reliable, subspace removal consistently improves AI recall. From the Low to Very High buckets, purified recall increases by 2.9–4.6 percentage points. These results support our confidence-gated design: down-weighting low-confidence cue estimates prevents over-removal in ambiguous cases while preserving the benefit of harmful cue suppression when generator cues are trustworthy.

Table 3: Bucketed analysis of the confidence gate. AI samples are grouped by cue strength. We report AI recall before and after subspace removal.

Bucket	Cue Strength	Conf.	Gap	Raw	Pur. / Δ
Very Low	[0.17, 0.63]	0.464	42.51	0.971	0.846 / -12.5
Low	[0.63, 0.72]	0.684	37.16	0.967	1.000 / $+3.3$
Medium	[0.72, 0.79]	0.758	36.00	0.967	1.000 / $+3.3$
High	[0.79, 0.87]	0.824	35.15	0.950	0.996 / $+4.6$
Very High	[0.87, 1.26]	0.946	35.46	0.967	0.996 / $+2.9$

5 CONCLUSION

We introduce CURE to suppress generator-specific shortcuts by estimating a low-rank harmful-cue subspace and removing it with confidence-guided strength. Across 12 unseen generators, it improves OOD detection over the compared detectors. Ablations show that subspace removal outperforms token deletion, while confidence gating limits over-removal when teacher cues are unreliable.

REFERENCES

Abdelrahman Abouelenin, Atabak Ashfaq, Adam Atkinson, Hany Awadalla, Nguyen Bach, Jianmin Bao, Alon Benhaim, Martin Cai, Vishrav Chaudhary, Congcong Chen, et al. Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-loras. *arXiv preprint arXiv:2503.01743*, 2025.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Guangsheng Bao, Yanbin Zhao, Zhiyang Teng, Linyi Yang, and Yue Zhang. Fast-detectgpt: Efficient zero-shot detection of machine-generated text via conditional probability curvature. In *ICLR*, 2024.

Jiaqi Chen, Xiaoye Zhu, Tianyang Liu, Ying Chen, Chen Xinhui, Yiwen Yuan, Chak Tou Leong, Zuchao Li, Long Tang, Lei Zhang, et al. Imitate before detect: Aligning machine stylistic preference for machine-revised text detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 23559–23567, 2025. doi: 10.1609/aaai.v39i22.34525.

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

Liam Dugan, Alyssa Hwang, Filip Trhlík, Andrew Zhu, Josh Magnus Ludan, Hainiu Xu, Daphne Ippolito, and Chris Callison-Burch. Raid: A shared benchmark for robust evaluation of machine-generated text detectors. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12463–12492, 2024.

Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 889–898, 2018.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. ELI5: long form question answering. In Anna Korhonen, David R. Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 3558–3567. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1346. URL https://doi.org/10.18653/v1/p19-1346.

Biyang Guo, Xin Zhang, Ziyuan Wang, Minqi Jiang, Jinran Nie, Yuxuan Ding, Jianwei Yue, and Yupeng Wu. How close is chatgpt to human experts? comparison corpus, evaluation, and detection. *arXiv preprint arXiv:2301.07597*, 2023.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Xun Guo, Yongxin He, Shan Zhang, Ting Zhang, Wanquan Feng, Haibin Huang, and Chongyang Ma. Detective: Detecting ai-generated text via multi-level contrastive learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Abhimanyu Hans, Avi Schwarzschild, Valeriia Cherepanova, Hamid Kazemi, Aniruddha Saha, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Spotting llms with binoculars: zero-shot detection of machine-generated text. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 17519–17537, 2024.

Xiaomeng Hu, Pin-Yu Chen, and Tsung-Yi Ho. Radar: Robust ai-text detection via adversarial learning. *Advances in neural information processing systems*, 36:15077–15095, 2023.

Kaijie Jiao, Quan Wang, Licheng Zhang, Zikang Guo, and Zhendong Mao. M-RangeDetector: Enhancing generalization in machine-generated text detection through multi-range attention masks. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 8971–8983, 2025. doi: 10.18653/v1/2025.findings-acl.469.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

Yafu Li, Qintong Li, Leyang Cui, Wei Bi, Zhilin Wang, Longyue Wang, Linyi Yang, Shuming Shi, and Yue Zhang. Mage: Machine-generated text detection in the wild, 2024.

Yuanfan Li, Zhaohan Zhang, Chengzhengxu Li, Chao Shen, and Xiaoming Liu. Iron sharpens iron: Defending against attacks in machine-generated text detection with adversarial training. *arXiv preprint arXiv:2502.12734*, 2025.

Yuanfan Li, Qi Zhou, and Zexuan Xie. Learning from dictionary: Enhancing robustness of machine-generated text detection in zero-shot language via adversarial training. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=bTcFHJo1Zk.

Shengchao Liu, Xiaoming Liu, Yichen Wang, Zehua Cheng, Chengzhengxu Li, Zhaohan Zhang, Yu Lan, and Chao Shen. Does detectgpt fully utilize perturbation? bridging selective perturbation to fine-tuned contrastive learning detector would be better. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1874–1889, 2024.

Xiaoming Liu, Zhaohan Zhang, Yichen Wang, Hang Pu, Yu Lan, and Chao Shen. Coco: Coherence-enhanced machine-generated text detection under data limitation with contrastive learning. *arXiv preprint arXiv:2212.10341*, 2022.

OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025. URL https://arxiv.org/abs/2508.10925.

Jinyan Su, Terry Zhuo, Di Wang, and Preslav Nakov. Detectllm: Leveraging log rank information for zero-shot detection of machine-generated text. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 12395–12412, 2023.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Yuxia Wang, Jonibek Mansurov, Petar Ivanov, Jinyan Su, Artem Shelmanov, Akim Tsvigun, Osama Mohanned Afzal, Tarek Mahmoud, Giovanni Puccetti, Thomas Arnold, et al. M4gt-bench: Evaluation benchmark for black-box machine-generated text detection. *arXiv preprint arXiv:2402.11175*, 2024a.

Yuxia Wang, Jonibek Mansurov, Petar Ivanov, jinyan su, Artem Shelmanov, Akim Tsvigun, Osama Mohammed Afzal, Tarek Mahmoud, Giovanni Puccetti, Thomas Arnold, Chenxi Whitehouse, Alham Fikri Aji, Nizar Habash, Iryna Gurevych, and Preslav Nakov. Semeval-2024 task 8: Multidomain, multimodel and multilingual machine-generated text detection. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, pp. 2041–2063, Mexico City, Mexico, June 2024b. Association for Computational Linguistics. URL https://aclanthology.org/2024.semeval2024-1.275.

Yihuai Xu, Yongwei Wang, Yifei Bi, Huangsen Cao, Zhouhan Lin, Yu Zhao, and Fei Wu. Training-free llm-generated text detection by mining token probability sequences. *arXiv preprint arXiv:2410.06072*, 2024.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang

Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *NIPS*, 2015.

Xiaowei Zhu, Yubing Ren, Fang Fang, Qingfeng Tan, Shi Wang, and Yanan Cao. Dna-detectllm: Unveiling ai-generated text via a dna-inspired mutation-repair paradigm. *arXiv preprint arXiv:2509.15550*, 2025.

A Experimental Setting

A.1 Dataset

We construct both the training and evaluation data from the arXiv abstract corpus. To avoid temporal leakage from recent LLM-generated or LLM-assisted writing, we only use human-written abstracts published before 2022 as the human source corpus. After basic preprocessing, including removing duplicated abstracts and filtering extremely short or malformed samples, we sample human abstracts to build balanced human–machine detection datasets.

For training, we sample 1,000 human abstracts from the pre-2022 human corpus. Each seen generator is then prompted to generate 1,000 machine-written abstracts, resulting in a training set that contains 1,000 human samples and 1,000 generated samples per LLM. The seen generators used for training are Qwen/Qwen3-4B, Qwen/Qwen3-235B-A22B, google/gemma-2b-it, and openai/gpt-oss-20b. For evaluation, we construct a strict unseen-generator test set. Each evaluation LLM generates 1,000 machine-written abstracts, while another disjoint set of human abstracts is sampled from the pre-2022 arXiv corpus. For each unseen generator, we pair its 1,000 generated samples with 1,000 human samples, forming a balanced 1:1 human–machine test set. The human abstracts used in evaluation are not included in the training set. This protocol ensures that the evaluation measures generalization to unseen generators rather than memorization of human samples or overlap with the training distribution.

The unseen generators used for evaluation include meta-llama/Llama-3.1-8B-Instruct, Qwen/Qwen3-14B, Qwen/Qwen2.5-32B-Instruct, mistralai/Mistral-Small-3.1-24B-Instruct-2503, microsoft/Phi-4-mini-instruct, google/gemma-4-26B-A4B-it, DeepSeek V3.2, GPT-4.1, Claude 4.6 Sonnet, Moonshot-v1-8k, openai/gpt-oss-120b, and Gemini 3 Flash.

A.2 Implementation

We implement CURE with ALBERT-base-v2 as the backbone. Unless otherwise specified, all experiments are conducted on a single NVIDIA A100-80GB GPU. On our training set, CURE takes approximately 15 minutes to train. The detailed hyperparameters are summarized in Table 4.

For metric-based detectors, which produce continuous detection scores rather than directly optimized classification logits, we train a logistic-regression calibrator on the training set to map raw detector scores into calibrated probabilities of the machine-generated class. We then apply a fixed decision threshold of 0.5 to compute accuracy and F1. For model-based detectors, we use ALBERT-base-v2 as the shared backbone whenever applicable, and train all methods for 6 epochs under the same optimization protocol. Their final predictions are also obtained using a fixed threshold of 0.5. This unified protocol ensures that the comparison mainly reflects the detection method itself rather than differences in backbone capacity, calibration strategy, or decision threshold tuning.

Table 4: Hyperparameters of CURE.

Category	Hyperparameter	Value
Training	Backbone	ALBERT-base-v2
	Max sequence length	512
	Batch size	16
	Epochs	6
	Learning rate	2×10^{-5}
Cue Score	Top-k ratio for cue token selection	0.15
	Confidence gate threshold	0.05
Subspace Removal	Removal layer index	4
	Subspace rank r	8
	Maximum removal strength $\rho_{\max}$	0.9
	Minimum removal strength $\rho_{\min}$	0.2
Replay Pool	Pool A size, broad harmful cues	2000
	Pool B size, hard harmful cues	1000
	Pool C size, clean references	2000
	Pool B cue threshold $\mu + k\sigma$	$k = 1.0$
	Clean suppression weight η	0.3

A.3 Evaluation Metrics

To evaluate detection performance, we report accuracy (Acc) and F1 score (F1). For robustness evaluation, we adopt attack success rate (ASR), which measures how often an attack causes originally correct detections of machine-generated text to become incorrect. Lower ASR therefore indicates stronger robustness. Formally, let $\mathcal{M}$ denote the set of MGT samples that are correctly classified before the attack, and let $f(\cdot)$ be the detector. Given an attack transformation $\mathcal{A}(\cdot)$, ASR is defined as

$$\mathrm{ASR} = \frac{\sum_{x \in \mathcal{M}} \mathbb{I}\left[f(\mathcal{A}(x)) \neq y_x\right]}{|\mathcal{M}|}, \quad \mathcal{M} = \{x \mid y_x = 1,\ f(x) = y_x\}, \tag{21}$$

where $y_x = 1$ denotes the machine-generated class and $\mathbb{I}[\cdot]$ is the indicator function.

A.4 Experimental Comparison Methods

To comprehensively evaluate the effectiveness of CURE, we compare it with representative MGT detectors from two major categories: *metric-based detectors* and *model-based detectors*. Metric-based detectors directly compute detection scores from language-model statistics and usually do not output calibrated probabilities. Therefore, following common practice, we fit a logistic-regression calibrator on the training set to map each raw score into the probability of the machine-generated class. For a fair comparison, all metric-based detectors use the same Qwen/Qwen3-8B scoring backbone whenever an external language model is required. Model-based detectors, in contrast, are trained as supervised classifiers on the labeled training set. Unless otherwise specified, all model-based detectors use ALBERT-base-v2 as the backbone and are trained under the same optimization protocol.

- **Binoculars** Hans et al. (2024): A metric-based zero-shot detector that identifies machine-generated text by comparing the output distributions of two related pretrained language models. It computes a lightweight contrastive score between the performer and observer models, capturing distributional discrepancies that tend to emerge in machine-generated text. Since Binoculars does not require supervised fine-tuning, it serves as a strong metric-based baseline.
- **Fast-DetectGPT** Bao et al. (2024): A metric-based detector derived from DetectGPT, designed to reduce the computational cost of perturbation-based detection. Instead of repeatedly perturbing the input text, Fast-DetectGPT estimates curvature-like likelihood patterns from a single forward computation, making it substantially more efficient while retaining competitive detection performance.
- **LastDE** Xu et al. (2024): A recent metric-based detector that exploits language-model scoring statistics to distinguish human-written and machine-generated text. Compared with earlier likelihood- or rank-based detectors, LastDE is designed to provide a stronger and more stable scoring signal under diverse generation settings. We include it as a representative recent metric-based baseline.
- **ALBERT-base** Lan et al. (2019): A standard supervised MGT detector obtained by fine-tuning an ALBERT-base-v2 classifier on the labeled training set. It directly outputs binary classification probabilities and serves as the backbone-level supervised baseline for model-based methods.
- **CoCo** Liu et al. (2022): A contrastive learning based detector that improves MGT detection by modeling coherence-related differences between human-written and machine-generated text. It encourages the detector to learn more discriminative representations through contrastive objectives, making it a representative model-based baseline for supervised MGT detection.
- **TASTE** Li et al. (2026): A model-based multilingual MGT detector that improves cross-lingual and cross-domain robustness through adversarial training. Instead of relying only on standard supervised fine-tuning, it introduces adversarial examples during training to expose fragile language- or generator-specific decision patterns, encouraging the detector to learn more transferable representations across different languages and generation sources. We include TASTE as a strong adversarially trained baseline and implement it with the same ALBERT-base-v2 backbone for controlled comparison.
- **GREATER** Li et al. (2025): An adversarially trained detector under the GREATER framework. It trains a detector together with an adversary that searches for vulnerable token-level perturbations, encouraging the detector to become robust against adversarially modified machine-generated text. Since GREATER focuses on adversarial robustness, it provides a strong robustness-oriented model-based baseline.
- **DeTeCtive** Guo et al. (2024): A recent model-based detector designed to improve the reliability of MGT detection under distribution shifts. It strengthens supervised detection by introducing additional training or representation-learning mechanisms beyond vanilla fine-tuning. We include DeTeCtive as another competitive model-based baseline for evaluating unseen-generator generalization.

- **ImBD** Chen et al. (2025): A style-preference scoring method designed to identify machine-revised text. We report its aggregate five-seed performance on the same unseen-generator split in Table 5; unlike the ALBERT-based classifiers above, it retains its method-specific scoring architecture.
- **M-RangeDetector** Jiao et al. (2025): A detector that uses global, band, dilated, and random attention masks to capture writing-strategy features. Its aggregate five-seed result is reported in Table 5; we retain its defining multi-range architecture rather than forcing an ALBERT-only implementation.

A.5 Detailed Information of Attack Methods

We provide detailed descriptions of the four text perturbation attacks used in our robustness evaluation. These attacks cover character-level deletion, visual character substitution, LLM-based humanization rewriting, and formatting-based perturbation.

- **Character Deletion.** This attack randomly removes a small portion of characters from the input machine-generated text while keeping the overall content largely readable. It introduces local spelling and surface-form corruption without intentionally changing the semantic meaning of the passage. This attack mainly targets detectors that rely on fragile lexical patterns, token-level likelihood cues, or exact surface forms.
- **Homoglyph Substitution.** This attack replaces selected characters with visually similar Unicode homoglyphs, such as substituting Latin characters with visually confusable counterparts from other scripts. The perturbed text appears nearly unchanged to human readers, but its tokenization and character-level statistics can be significantly altered. Homoglyph substitution therefore tests whether a detector is robust to visual-level perturbations that preserve semantics while disrupting low-level textual features.
- **Humanization Rewriting.** We implement a few-shot contrastive humanization attack with an instruction-tuned LLM. The goal is to rewrite machine-generated text into a more human-like style while preserving its original meaning and factual content. Specifically, the attacker is prompted with both human-written examples to imitate and machine-generated examples to avoid, so that the model can contrast the two writing styles during rewriting. The system prompt is:

  You are an expert human writer and editor. Your task is to rewrite the given MACHINE-GENERATED text so that it is indistinguishable from HUMAN-WRITTEN text.
  Strict rules: 1) Preserve the original meaning and factual content. 2) Keep the SAME LANGUAGE as the input. 3) Do NOT add new facts, citations, disclaimers, or meta commentary. 4) Output ONLY the rewritten text.
  Below are style references. Imitate the HUMAN writing style and avoid the MACHINE style.

  ```
  == Human-written examples (imitate these) ==
  [H1] {human_example_1}
  [H2] {human_example_2}

  == Machine-generated examples (avoid these patterns) ==
  [M1] {machine_example_1}
  [M2] {machine_example_2}

  == Target text to rewrite (machine-generated) ==
  {target_text}

  Rewrite the target text to sound human-written. Output only the
  rewritten text.
  ```

  The user prompt contains two human-written examples and two machine-generated examples sampled from the attack training set, followed by the target machine-generated text. The attacker is asked to imitate the human writing style and avoid machine-like patterns. We control the total token budget to 4096 tokens, allocate approximately half of the budget to the target text, and use the remaining budget for few-shot style references. If the prompt exceeds the budget, we progressively reduce the number of few-shot pairs. In our experiments, we

use **microsoft/Phi-4-mini-instruct** as the rewriting model and perform local inference with **vLLM** using a batch size of 32.

- **Format Perturbation.** This attack modifies the formatting of the input text without changing its semantic content. Typical operations include inserting or removing whitespace, changing line breaks, altering paragraph structure, and introducing zero-width or formatting-related characters. Such perturbations are designed to preserve readability while disrupting tokenization, positional patterns, and formatting-sensitive detector features.

B Additional Result

B.1 ImBD and M-RangeDetector Comparisons

ImBD Chen et al. (2025) is a style-preference detector originally developed for machine-revised text, while M-RangeDetector Jiao et al. (2025) models writing strategies with multiple attention ranges. We evaluate these two detector baselines on the same 12 unseen-generator split used in the main paper.

Table 5: Five-seed OOD-average F1 on the original 12 unseen generators.

Detector	F1 (mean $\pm$ std.)
M-RangeDetector	79.3 ± 1.5
DeTeCtive	81.1 ± 2.3
ImBD	82.5 ± 3.4
CURE	$\mathbf{88.9 \pm 0.8}$

On this five-seed evaluation, CURE exceeds ImBD by 6.4 F1 points and M-RangeDetector by 9.6 points.

B.2 Representation Probe Analysis

In the Section 1, we have shown that detector representations contain strong generator-specific information, which can be recovered by lightweight linear probes. We further examine whether CURE actually suppresses such information after harmful-cue subspace removal, while still preserving the discriminative evidence required for human–AI detection. This analysis directly evaluates the central goal of CURE: removing generator-specific cues without destroying task-relevant representations.

Specifically, we use the trained CURE detector and extract three CLS representations for the same input:

$$\mathbf{z}_{\text{raw}} = \text{CLS}(\mathbf{x}), \qquad \mathbf{z}_{\text{pur}} = \text{CLS}(\tilde{\mathbf{x}}), \qquad \mathbf{z}_{\text{res}} = \mathbf{z}_{\text{raw}} - \mathbf{z}_{\text{pur}}. \tag{22}$$

Here, $\mathbf{z}_{\text{raw}}$ is the original representation, $\mathbf{z}_{\text{pur}}$ is the representation after subspace removal, and $\mathbf{z}_{\text{res}}$ is the removed residual component. We construct a probing set with 12 generators and human-written samples, randomly split it into 70% training and 30% testing, and train linear Logistic Regression probes on each representation view. We evaluate three tasks: Generator-ID classification, Family classification, and AI-vs-Human classification.

As shown in Table 6, the purified representation contains substantially less generator-specific information. The Generator-ID probing accuracy drops from 0.553 to 0.454, and the Family probing accuracy drops from 0.685 to 0.586, both decreasing by 0.099 after purification. This indicates that subspace removal effectively weakens the generator identity signal encoded in the detector representation.

Table 6: Linear probing results on different representation views. We report classification accuracy. $\mathbf{z}_{\text{raw}}$ denotes the original CLS representation, $\mathbf{z}_{\text{pur}}$ denotes the purified representation after subspace removal, and $\mathbf{z}_{\text{res}}$ denotes the removed residual component.

Probe Task	**Random**	$\mathbf{z}_{\text{raw}}$	$\mathbf{z}_{\text{pur}}$	$\mathbf{z}_{\text{res}}$	Δ
Generator-ID	0.083	0.553	0.454	0.483	-0.099
Family	0.125	0.685	0.586	0.600	-0.099
AI vs. Human	0.500	0.947	0.933	0.942	-0.014

Table 7: Selective intervention sensitivity on 12 unseen generators. Δ_{harm}, Δ_{rand}, and Δ_{clean} denote the change in AI probability after removing the learned harmful subspace, a random subspace, and a clean-reference subspace, respectively. Negative values indicate suppressed AI evidence. Harm/Rand measures the selectivity of the learned harmful subspace relative to random directions.

Generator	**Baseline F1**	**CURE F1**	**Gain**	Δ_{harm}	Δ_{rand}	Δ_{clean}	**Harm/Rand**
LLaMA3-8B	0.719	0.873	+0.153	-0.072	-0.011	-0.039	6.66×
Qwen3-14B	0.769	0.897	+0.128	-0.048	-0.008	-0.028	6.09×
Qwen2.5-32B	0.726	0.826	+0.100	-0.078	-0.015	-0.046	5.09×
Mistral	0.856	0.945	+0.090	-0.016	-0.000	-0.010	36.2×
Phi-4	0.703	0.791	+0.088	-0.158	-0.023	-0.077	6.84×
Gemma4-26B-A4B	0.702	0.831	+0.129	-0.063	-0.021	-0.032	3.04×
DeepSeek V3.2	0.799	0.892	+0.094	-0.054	-0.006	-0.024	9.92×
GPT-4.1	0.767	0.918	+0.152	-0.035	-0.004	-0.020	10.0×
Gemini 3 Flash	0.770	0.919	+0.149	-0.033	-0.004	-0.014	9.31×
Claude 4.6 Sonnet	0.828	0.972	+0.143	-0.006	-0.001	-0.003	5.70×
Moonshot v1	0.748	0.846	+0.098	-0.111	-0.011	-0.045	10.2×
GPT-OSS 120B	0.908	0.979	+0.071	0.000	-0.000	0.000	–
High-gain group	–	–	–	-0.043	-0.008	–	5.38×
Low-gain group	–	–	–	-0.070	-0.009	–	7.78×

More importantly, the AI-vs-Human probing accuracy is only slightly affected, decreasing from 0.947 to 0.933. The small drop of 0.014 suggests that CURE does not indiscriminately damage the representation, but selectively suppresses generator-related information while largely preserving the binary detection signal. In addition, the residual representation $\mathbf{z}_{\text{res}}$ still achieves 0.483 Generator-ID accuracy, far above the random baseline of 0.083, showing that the removed component itself carries substantial generator-specific information. These results confirm that CURE performs selective representation purification rather than generic feature degradation.

B.3 SELECTIVE INTERVENTION SENSITIVITY

In the Section 1, we have shown that detector representations encode generator-specific information and that unseen-generator shifts exhibit clear structural patterns. We further ask whether the harmful cue subspace learned by CURE is truly directionally selective: if the learned subspace captures generator-specific cues, removing it should suppress AI evidence more strongly than removing random directions or clean-reference directions. This analysis directly examines whether CURE identifies meaningful generator-related directions rather than arbitrary representation components.

Specifically, we take 200 AI samples from each of the 12 unseen generators and evaluate a frozen detector under three different subspace interventions. The first intervention uses $\mathbf{U}_{\text{harm}}$, the harmful cue subspace estimated from the harmful replay pools. The second uses $\mathbf{U}_{\text{rand}}$, a random orthogonal subspace with the same rank, as a control. The third uses $\mathbf{U}_{\text{clean}}$, estimated from clean-reference token representations. For each intervention, we measure the change in AI probability:

$$\Delta = p_{\text{AI}}^{\text{pur}}(\mathbf{x}) - p_{\text{AI}}^{\text{raw}}(\mathbf{x}), \tag{23}$$

where a negative value indicates that the intervention suppresses the detector's AI evidence. We further report the ratio $|\Delta_{\text{harm}}|/|\Delta_{\text{rand}}|$ to quantify how much more targeted the learned harmful subspace is than random removal.

Table 7 shows that the learned harmful cue subspace is much more targeted than random directions. For nearly all generators, removing $\mathbf{U}_{\text{harm}}$ leads to a substantially larger decrease in AI probability than removing $\mathbf{U}_{\text{rand}}$. The Harm/Rand ratio is at least 3.04× and often exceeds 6×, with an average ratio of about 8× over the valid generators. This confirms that CURE does not simply remove arbitrary representation energy; instead, it identifies directions that are specifically aligned with generator-dependent evidence.

The generator-wise results also reveal that the intervention effect is structured rather than uniform. For example, Phi-4 has the largest suppression magnitude, with $\Delta_{\text{harm}} = -0.158$, much stronger than the random control of -0.023. This is consistent with its role as one of the hardest generators

Table 8: OOD generalization results across domains, generators, and datasets. We report AUROC. AGNews evaluates cross-domain transfer, WritingPrompts and ELI5 evaluate simultaneous cross-domain and cross-generator transfer, and HC3 evaluates cross-dataset generalization. The best results are highlighted in **bold**, and the second-best results are underlined.

Method	Cross-Domain	Cross-Domain + Cross-Generator		Cross-Dataset	Avg.
	AGNews	WritingPrompts	ELI5	HC3	
Binoculars	84.8	68.6	68.0	73.7	73.8
FastDetectGPT	57.6	**72.9**	65.2	70.7	66.6
LastDE	95.8	71.4	64.5	**77.3**	77.3
ALBERT-base	94.6	54.0	72.7	72.3	73.4
CoCo	95.6	63.4	73.3	67.3	74.9
TASTE	**96.2**	67.9	73.7	74.6	78.1
GREATER	90.6	64.0	63.9	57.2	68.9
DeTeCtive	72.6	55.1	60.8	65.9	63.6
CURE	**96.2**	71.7	**75.5**	75.7	**79.8**

in our main evaluation, suggesting that its representation contains strong harmful cue signals. In contrast, Mistral has a relatively small absolute suppression magnitude, $\Delta_{\text{harm}} = -0.016$, but its random intervention is almost ineffective, yielding a Harm/Rand ratio of $36.2\times$. This indicates that Mistral's generator-specific information is concentrated in a highly specific direction that can be captured by the learned harmful subspace but not by random removal.

Finally, the group-level comparison shows that larger suppression does not necessarily imply larger F1 improvement. The low-gain group has a larger average suppression magnitude than the high-gain group, with $\Delta_{\text{harm}} = -0.070$ versus -0.043, but its F1 gain is smaller. This suggests that the effectiveness of CURE is not determined by how much signal is removed, but by whether the removed directions correspond to harmful generator-specific cues. Together with the representation probing results, this analysis provides further evidence that CURE performs selective cue removal rather than indiscriminate feature degradation.

B.4 CROSS-DOMAIN AND CROSS-DATASET GENERALIZATION

Experimental setup. Beyond unseen-generator evaluation within the same data source, we further evaluate whether CURE generalizes under broader OOD shifts. We consider four evaluation sets and report AUROC as the threshold-independent metric. AGNews Zhang et al. (2015) is used to evaluate cross-domain transfer, where the domain differs from the arXiv training data. WritingPrompts Fan et al. (2018) and ELI5 Fan et al. (2019) evaluate a harder setting that combines domain shift with generator shift, since both the writing domain and the generation source differ from those in training. HC3 Guo et al. (2023) is used as an additional cross-dataset benchmark to test generalization beyond our constructed evaluation pipeline.

Experiment results. We report the results in Table 8 and summarize three findings. **1) Best average OOD generalization.** Across the four OOD settings, CURE achieves the best average AUROC of **79.8**, outperforming the strongest baseline TASTE by **+1.7** points and LastDE by **+2.5** points. This shows that the benefit of CURE is not limited to unseen generators within the same domain, but also transfers to broader domain and dataset shifts. **2) Strong performance under cross-domain and cross-generator shifts.** On AGNews, which mainly evaluates cross-domain transfer, CURE reaches **96.2** AUROC, matching the best-performing baseline. More importantly, on the more challenging CD+CG settings, CURE achieves the best AUROC on ELI5 with **75.5** and the second-best AUROC on WritingPrompts with 71.7. Compared with strong model-based baselines such as GREATER and DeTeCtive, the gains are especially clear, suggesting that suppressing generator-specific directions helps the detector retain more transferable evidence under simultaneous domain and generator shifts. **3) Consistent gains over supervised detectors.** Although several baselines perform well on one specific dataset, their performance is less stable across OOD settings. For example, FastDetectGPT achieves the best AUROC on WritingPrompts but performs poorly on AGNews, while LastDE performs strongly on AGNews and HC3 but drops on ELI5. In contrast, CURE remains competitive

across all datasets and obtains the best average result. This echoes our main motivation: detector failures under OOD shifts are closely related to non-transferable generator- or domain-specific cues, and representation-level cue suppression encourages more stable generalization.

B.5 Robustness against Attacks

Experimental setup. We evaluate the robustness of different detectors on the ID test sets under four text perturbation attacks: character deletion, homoglyph substitution, humanization rewriting, and format perturbation. We report attack success rate (ASR), where a lower value indicates stronger robustness. Detailed descriptions of these attack methods are provided in the Appendix A.5.

Experiment results. We report the robustness results in Table 9 and summarize three findings. **1) Best overall robustness.** CURE achieves the lowest average ASR of **4.92**, outperforming the strongest baseline GREATER with 6.70 average ASR by **1.78** percentage points. Compared with ALBERT-base, TASTE, and DeTeCtive, CURE reduces average ASR by **2.14**, **2.90**, and **4.58** percentage points, respectively. This shows that suppressing generator-specific cue directions does not sacrifice robustness, but instead improves resistance to text perturbations. **2) Strong gains against rewriting and formatting attacks.** The advantage of CURE is especially clear under attacks that substantially change surface forms. For humanization rewriting, CURE reduces ASR to **0.89**, improving over the second-best detector TASTE by **3.04** percentage points. For format perturbation, CURE obtains **12.14** ASR, outperforming the second-best ALBERT-base by **1.80** percentage points. These results suggest that CURE is less dependent on superficial generator-specific patterns and is therefore more robust to perturbations that alter style or formatting. **3) Competitive robustness across all attack types.** CURE achieves the best ASR on three out of four attacks, including character deletion, humanization rewriting, and format perturbation. On homoglyph substitution, CURE attains the second-best ASR of 0.29, very close to the best result of **0.00** from GREATER. Overall, CURE provides consistent robustness across both character-level and rewriting-level attacks, further supporting our claim that representation-level cue removal improves detector reliability beyond unseen-generator generalization.

Table 9: Attack success rate (ASR) under different text perturbation attacks. Lower ASR indicates stronger robustness. The best results are highlighted in **bold**, and the second-best results are underlined.

Attack	ALBERT-base	GREATER	TASTE	DeTeCtive	CURE
Char. Deletion	8.18	6.74	9.12	10.50	**6.36**
Homoglyph Sub.	0.30	**0.00**	0.59	0.85	0.29
Humanization	5.83	4.82	3.93	6.54	**0.89**
Format Perturb.	13.94	15.25	17.65	20.10	**12.14**
Avg ASR	7.06	6.70	7.82	9.50	**4.92**

B.6 Efficiency Analysis

Table 10: Efficiency comparison of different detectors. We report OOD average accuracy and F1 from Table 1, together with inference latency and GPU memory usage. Higher Acc/F1 is better, while lower latency and memory usage are better.

Method	Acc ↑	F1 ↑	Latency ↓ (ms)	Memory ↓ (GiB)
Binoculars	78.5	77.9	105.72	54.90
FastDetectGPT	75.4	74.7	68.61	37.70
LastDE	72.1	72.0	42.91	16.90
ALBERT-base	77.7	77.5	5.31	**2.04**
CoCo	73.8	71.5	**4.10**	**2.04**
TASTE	76.8	74.9	8.26	**2.04**
GREATER	80.0	80.6	6.22	**2.04**
DeTeCtive	82.0	81.4	5.76	5.08
CURE	**89.7**	**89.9**	5.07	4.04

Besides detection accuracy, inference efficiency is also important for practical MGT detection. We therefore compare different detectors in terms of OOD performance, inference latency, and GPU memory usage. As shown in Table 10, metric-based detectors incur substantially higher computational

cost because they rely on large language models for scoring. For example, Binoculars requires 105.72 ms latency and 54.90 GiB memory, while FastDetectGPT requires 68.61 ms and 37.70 GiB. Although these methods avoid supervised fine-tuning, their inference overhead makes them less suitable for efficient large-scale deployment.

In contrast, model-based detectors are much more efficient because they only require encoder-based classification. Among them, CURE achieves the best OOD performance with **89.7** accuracy and **89.9** F1, while maintaining a low latency of only 5.07 ms. Compared with DeTeCtive, CURE improves OOD F1 by **+8.5** points and is also slightly faster, with 5.07 ms versus 5.76 ms latency. Although CURE introduces an additional purification view and subspace-removal operation, its memory usage remains modest at 4.04 GiB, far below metric-based detectors and lower than DeTeCtive. These results show that CURE substantially improves unseen-generator generalization without introducing prohibitive inference overhead.

C TRAINING AND DETECTION PROCEDURE OF CURE

Algorithm 1 presents the training and detection procedure of CURE.

D LIMITATIONS

This work has two main limitations. **First**, our experiments focus on English texts, and we have not systematically evaluated whether the proposed harmful cue subspace removal strategy transfers to multilingual MGT detection. Since generator-specific cues may differ across languages, scripts, and writing conventions, future work should extend the evaluation to multilingual and code-mixed scenarios. **Second**, our detector operates at the document level and predicts whether an entire passage is machine-generated or human-written. It does not provide token-level provenance or fine-grained localization of generated spans, which may be important for mixed-authorship documents where human-written and machine-generated content are interleaved. Future work could adapt the representation purification mechanism to token-level or span-level detection settings.

E ETHICS STATEMENT

This work aims to improve the reliability of machine-generated text (MGT) detection and to help users assess the provenance of online content. More reliable detectors can contribute to mitigating misuse of LLM-generated text, such as fabricated reviews, phishing emails, fake news, academic misconduct, and other forms of deceptive or large-scale synthetic content. At the same time, MGT detection is a high-risk application if used without appropriate safeguards, and we therefore discuss the potential ethical implications of our work.

False positives and high-stakes deployment. MGT detectors are inherently imperfect. False positives may incorrectly label human-written text as machine-generated, which could unfairly harm students, researchers, journalists, non-native speakers, or writers with atypical styles. Therefore, our method should not be used as the sole basis for punitive or high-stakes decisions, such as academic discipline, hiring, content removal, or legal judgment. Instead, it should be treated as a decision-support tool and combined with human review, transparent appeal mechanisms, and additional contextual evidence.

Bias and fairness. Although our goal is to improve generalization to unseen generators, detector behavior may still vary across domains, writing styles, demographic groups, languages, and levels of writing proficiency. Since our main experiments focus on English text, the results should not be assumed to transfer directly to multilingual or code-mixed settings. Future deployment should include fairness audits across linguistic backgrounds, domains, and user groups to reduce the risk of systematic over-flagging.

Privacy and attribution boundaries. Our method studies generator-specific cues for the purpose of improving human–machine classification. It is not designed to identify individual authors, infer sensitive attributes, or perform user profiling. We do not use private user data in our experiments, and the datasets are constructed from public or generated text sources. Any practical deployment should

avoid storing unnecessary user content and should follow applicable privacy and data-protection requirements.

Dual-use risks. This work includes analyses of detector weaknesses and robustness under text perturbation attacks. Such information could potentially be misused to design stronger evasion strategies against MGT detectors. We include these analyses because they are necessary for scientific evaluation and for building more reliable defenses. To reduce misuse risk, we do not intend to release tools that automate harmful evasion beyond what is necessary for reproducibility and controlled robustness evaluation. Released code and data processing scripts will focus on reproducing the detector and evaluation protocol rather than facilitating real-world abuse.

Responsible release and transparency. We will release code, dataset construction scripts, and experimental configurations to support reproducibility, subject to license and safety considerations. We will document the intended use, limitations, and known failure modes of the method. We also encourage users of MGT detectors to report uncertainty rather than binary judgments whenever possible, and to communicate clearly that detection results are probabilistic evidence rather than definitive proof of authorship.

Algorithm 1 Training and Detection Procedure of CURE

Input: training set $D_{\text{train}} = \{(x, y)\}$, frozen generator teacher g_ϕ, detector f_θ, removal layer L, maximum epochs E, cue ratio k, subspace rank r, confidence threshold τ, removal bounds $\rho_{\min}, \rho_{\max}$, clean suppression weight η
Initialize: replay pools $\mathcal{P}_A \leftarrow \emptyset$, $\mathcal{P}_B \leftarrow \emptyset$, $\mathcal{P}_C \leftarrow \emptyset$, harmful cue subspace $\mathbf{U}_{\text{harm}} \leftarrow \emptyset$
**** training-stage cue purification begins ****
for $epoch = 1$ to E **do**
  for each mini-batch $\mathcal{B} \subset D_{\text{train}}$ **do**
**** Step A: Generator-cue estimation. ****
    Run the frozen teacher g_ϕ and obtain generator probabilities $p_\phi(g \mid x)$
    Obtain the teacher's top-1 generator prediction $\hat{g}$ using Eq. (2)
    Calculate token-level generator-cue scores $\{s_t\}$ using Eq. (3)
    Calculate the teacher confidence margin $m(x)$ using Eq. (4)
    Calculate the teacher confidence weight $\omega(x)$ using Eq. (5)
    Obtain confidence-gated cue scores $\{\tilde{s}_t\}$ using Eq. (6)
    Select top-$k\%$ cue-bearing tokens and build cue-token masks $\{q_t\}$ using Eq. (7)
**** Step B: Update replay pools. ****
    Extract token hidden states $\mathbf{H}^{(L)}$ from the detector using Eq. (1)
    Update broad harmful-cue pool $\mathcal{P}_A$ using Eq. (8)
    Update hard harmful-cue pool $\mathcal{P}_B$ using Eq. (9)
    Update clean-reference pool $\mathcal{P}_C$ using Eq. (10)
**** Step C: Confidence-guided subspace removal. ****
    Compute the raw prediction $p_\theta^{\text{raw}}(y \mid x)$ using Eq. (17)
    if $\mathbf{U}_{\text{harm}} \neq \emptyset$ **then**
      Calculate token-specific removal coefficients $\{\rho_t\}$ using Eq. (15)
      Calculate the effective removal strength $\rho_{\text{eff}}(x)$ using Eq. (16)
      Purify selected layer-L token hidden states using Eq. (14)
      Continue the forward pass with purified hidden states and obtain $p_\theta^{\text{pur}}(y \mid x)$ using Eq. (17)
    else
      Set $p_\theta^{\text{pur}}(y \mid x) \leftarrow p_\theta^{\text{raw}}(y \mid x)$
    end if
**** Step D: Update detector. ****
    Calculate the dual-view training loss $\mathcal{L}$ using Eq. (18)
    Update detector parameters θ with gradient descent
  end for
**** Step E: Estimate harmful cue subspace. ****
  Estimate harmful-cue and clean-reference covariance matrices using Eq. (11)
  Construct the contrastive cue covariance matrix using Eq. (12)
  Update the harmful cue subspace $\mathbf{U}_{\text{harm}}$ using Eq. (13)
end for
**** detection stage begins ****
Input: test text x_{test}, trained detector f_θ, frozen teacher g_ϕ, harmful cue subspace $\mathbf{U}_{\text{harm}}$
Compute raw prediction $p_\theta^{\text{raw}}(y \mid x_{\text{test}})$ using Eq. (17)
Calculate token-level cue scores using Eq. (3)
Calculate the confidence margin using Eq. (4)
Calculate the confidence weight using Eq. (5)
Obtain gated cue scores using Eq. (6)
Select top-$k\%$ cue-bearing tokens and build cue-token masks using Eq. (7)
Calculate token-specific removal coefficients using Eq. (15)
Calculate the effective removal strength using Eq. (16)
Purify selected token hidden states using Eq. (14)
Compute purified prediction $p_\theta^{\text{pur}}(y \mid x_{\text{test}})$ using Eq. (17)
Average raw and purified predictions using Eq. (19)
Output: final prediction $\hat{y}$ using Eq. (20)

AI USE

Generative AI tools assisted with language polishing, manuscript formatting, and drafting the presentation of author-provided baseline results. The authors remain responsible for verifying the numerical results and technical claims and for the final content of this paper.