

Fight Poison with Poison: Enhancing Robustness in Few-shot Machine-Generated Text Detection with Adversarial Training

Wenjing Duan^{1,†}, Qi Zhou^{1,†}, Yuanfan Li^{1,†,*}

¹Faculty of Electronic and Information Engineering, Xi'an Jiaotong University

[†] Equal contribution, * Corresponding author

dwjduan@stu.xjtu.edu.cn

Corresponding to: liyuan7716@gmail.com

Abstract

Machine-generated text (MGT) detection is critical for regulating online information ecosystems, yet existing detectors often fail under limited fresh supervision and humanization-oriented attacks. We study this problem as *few-shot robustness recovery* under evolving generators, domains, and attack patterns, and use threat modeling to diagnose detector failures from an attacker’s viewpoint. We identify unstable local boundary estimation as a key failure mode: under scarce supervision, humanized MGT can induce prediction flips, logit drift, and clean–adversarial representation misalignment. To address this issue, we propose **RAG-GuidEd Attacker Strengthens ConTrastive Few-shot Detector (REACT)**, a threat-aware few-shot robustness framework. REACT uses a dynamic retrieval-guided attacker to generate boundary-oriented humanized rewrites from extreme positive–negative anchors, and trains the detector with a pairwise boundary contrastive objective to stabilize clean–adversarial local geometry. Alternating attacker–detector updates further form a co-evolutionary curriculum. Experiments on 4 datasets with 4 shot sizes and 3 random seeds show that REACT improves average detection F1 by **4.95** points over 8 SOTA detectors and reduces average attack success rate under 4 strong attacks by **3.66** percentage points.

1 Introduction

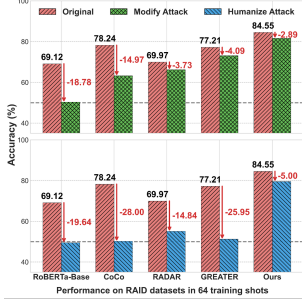
With the rapid advancement of large language models (LLMs) such as ChatGPT (OpenAI et al., 2024), LLaMA (Grattafiori et al., 2024), and DeepSeek (Guo et al., 2025), generating fluent and highly human-like text at scale has become increasingly easy. While this capability benefits many applications, it also enables misuse, including LLM-written peer-review reports, phishing emails, and fabricated news. Machine-generated text (MGT) detection has therefore become an

important tool for identifying the provenance of textual content and mitigating synthetic-text misuse (Liu et al., 2022; Hu et al., 2023; Liu et al., 2024; Bao et al., 2024; Li et al., 2025).

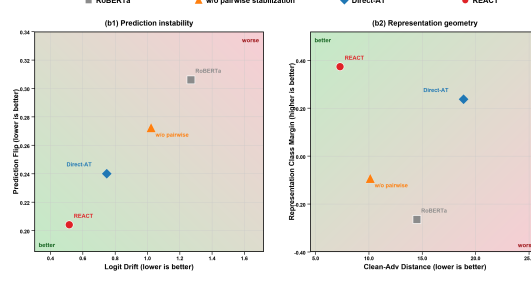
Despite recent progress, robust MGT detection remains difficult in threat-aware rapid adaptation scenarios. In practice, detectors are deployed in environments where generators, model versions, decoding policies, and application domains evolve continuously. Meanwhile, an adaptive adversary may intentionally humanize MGT to preserve its meaning while moving it toward the human-written distribution, with the goal of inducing the detector to accept it as HWT. When such a new generator or attack pattern appears, rebuilding a large labeled benchmark is often costly, since it requires collecting fresh MGT, matching human-written references, covering diverse prompts and domains, and cleaning noisy data. A more realistic requirement is therefore *few-shot robustness recovery*: the detector must quickly regain reliability from only a small number of fresh labeled examples while already facing adversarially humanized MGT.

Figure 1(a) shows that this setting exposes a severe robustness gap. Representative MGT detectors suffer substantial accuracy drops under humanization-oriented attacks in few-shot settings. Few-shot-tailored detectors improve clean detection to some extent, but remain vulnerable once small edits move MGT toward the human-written distribution. Existing adversarially trained detectors are more robust than ordinary fine-tuning, yet still degrade sharply under stronger humanization attacks (Hu et al., 2023; Koike et al., 2024; Li et al., 2025; Wang et al., 2024a). These results suggest that robust few-shot MGT detection cannot be solved by simply adding adversarial examples to training.

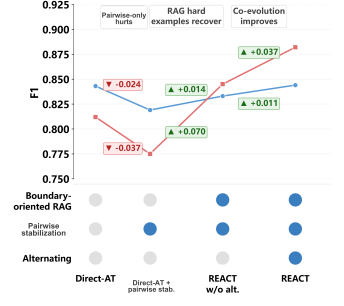
We argue that this threat-aware setting exposes a key failure mode: *unstable local boundary*



(a) Few-shot robustness collapse



(b) Boundary instability diagnosis



(c) Component interaction analysis

Figure 1: Empirical motivation and mechanism diagnosis of REACT. (a) In few-shot settings, representative MGT detectors suffer severe accuracy drops under humanization-oriented attacks, revealing a practical robustness gap. (b) Stability diagnostics show high prediction flips and logit drift, a weak representation class margin, and large clean–adversarial distance. A larger representation class margin means that an adversarial MGT is closer to the MGT region than to the HWT region; the other three metrics are better when smaller. Formal definitions are in Appendix B.4. (c) A component interaction analysis shows that direct adversarial training is insufficient and that pairwise boundary stabilization is not a plug-and-play add-on. The best performance emerges only when boundary-oriented RAG hard examples, pairwise stabilization, and alternating attacker–detector co-evolution are coupled in the full REACT framework. Details can be found in Appendix B.8.

estimation under scarce supervision. From the attacker’s perspective, successful humanization does not need to fully imitate all properties of HWT; it only needs to move MGT across the detector’s local decision boundary while preserving semantic plausibility. However, in the few-shot regime, the detector must infer a reliable boundary among HWT, clean MGT, and humanized adversarial MGT from very limited evidence. Generic adversarial training can expose the detector to perturbed samples, but the generated rewrites may be weak, off-boundary, or poorly aligned with the detector’s current vulnerabilities. Moreover, directly imposing contrastive constraints on such weak adversarial examples may over-constrain the representation and hurt optimization. Figure 1(b) diagnoses this instability with four quantities: *Prediction Flip* records whether humanization changes the detector’s predicted label; *Logit Drift* measures the induced change in detector confidence; *Representation Class Margin* measures an adversarial MGT’s relative position with respect to the HWT and MGT representation regions; and *Clean–Adversarial Distance* measures the representation shift from an original MGT to its humanized rewrite. Existing detectors show high flips and drift, weak margins, and large clean–adversarial shifts, indicating that the core issue is unstable local decision geometry rather than merely low clean accuracy.

To address this failure mode, we propose **RAG-GuidEd Attacker Strengthens ConTrastive Few-**

shot Detector (REACT), a few-shot robustness framework for MGT detection. REACT couples a detector with a retrieval-guided attacker that generates boundary-oriented humanized adversarial examples. Instead of relying on generic paraphrases, the attacker maintains a dynamic retrieval pool of training samples and previously generated adversaries, and retrieves extreme positive–negative anchors to guide humanization toward the detector’s current weak boundary regions. These on-policy hard examples provide more informative adversarial supervision for few-shot learning.

On the detector side, REACT introduces a pairwise boundary contrastive objective to stabilize the local geometry between clean MGT and its humanized counterpart. Unlike batch-level contrastive objectives that depend on many informative negatives, this pairwise signal is low-variance and naturally suited to scarce supervision. The attacker and detector are alternately updated, forming a co-evolutionary curriculum: the attacker progressively tracks the detector’s weaknesses, while the detector learns a more stable machine-generated boundary under humanization pressure. As shown in Figure 1(c), the improvement comes from this coupled interaction rather than any single plug-and-play component. Extensive experiments across 4 datasets, 3 random seeds, and 4 shot budgets demonstrate that REACT achieves an average detection F1 score of **93.72**, outperforming 8 SOTA detectors by **4.95** points. Under 4 strong attacks, REACT attains an average

attack success rate (ASR) of **10.04%**, reducing ASR by **3.66 percentage points** compared with the strongest baseline. Further analyses show that REACT reduces prediction flips and logit drift, improves representation class margins, and stabilizes clean–adversarial representation geometry. Our contributions are summarized as follows:

- **Pairwise boundary stabilization with co-evolution.** We introduce a pairwise boundary contrastive objective and an alternating attacker–detector training strategy, enabling the detector to stabilize clean–adversarial local geometry under scarce supervision and achieve stronger robustness across datasets, shot budgets, and attacks.
- **Boundary-oriented adversarial generation.** We propose a retrieval-guided attacker that selects extreme positive–negative anchors from a dynamic pool to generate on-policy humanized rewrites, producing boundary-relevant hard examples rather than generic paraphrases.
- **Comprehensive empirical validation.** Experiments across 4 datasets, 4 shot budgets, 3 random seeds, and 4 strong attacks show that REACT consistently improves both detection performance and robustness over 8 SOTA detectors.

2 Related Work

Machine-Generated Text (MGT) Detectors. Recent MGT detectors can be broadly divided into *metric-based* and *model-based* methods. Metric-based detectors (Su et al., 2023; Bao et al., 2024; Hans et al., 2024; Xu et al., 2024; Zhu et al., 2025) require little or no supervised training and rely on statistical signals of generated text, but they are often fragile to adversarial perturbations and require careful threshold calibration. Model-based detectors (Liu et al., 2022; Hu et al., 2023; Liu et al., 2024; Koike et al., 2024; Li et al., 2025) fine-tune classifiers such as BERT or RoBERTa and typically achieve stronger performance, yet they depend on sufficient labeled data. Although few-shot-tailored methods (Liu et al., 2022, 2024) reduce this data requirement, they remain vulnerable under humanization attacks; adversarially trained detectors (Hu et al., 2023; Li et al., 2025) also struggle when scarce supervision makes the decision boundary unstable. Our work therefore tar-

gets robust few-shot MGT detection, aiming to recover both accuracy and attack robustness under limited fresh supervision.

Adversarial Training. Adversarial training improves robustness by optimizing a detector on adversarial examples deliberately crafted to induce misclassification. It has been widely used for MGT detection (Hu et al., 2023; Koike et al., 2024; Li et al., 2025, 2026), but directly applying this paradigm to few-shot settings is non-trivial. Recent studies (Thorat and Caines, 2025; Umrajkar, 2025) suggest that adversarial examples can exacerbate underfitting when supervision is scarce, and existing adversarially trained MGT detectors still degrade under highly humanized attacks. Different from prior methods that mainly treat adversarial examples as additional training data, REACT targets the few-shot failure mode of unstable local boundary estimation. It uses a dynamic retrieval pool to construct boundary-oriented hard examples, applies pairwise boundary contrast to stabilize clean–adversarial local geometry, and alternately updates the attacker and detector so that adversarial examples track the detector’s current weaknesses. Thus, REACT turns adversarial training from generic hard-example augmentation into a few-shot robustness recovery mechanism.

3 Threat Model

We follow the standard threat modeling framework outlined in prior work (Biggio and Roli, 2018; Hu et al., 2023; Koike et al., 2024; Li et al., 2025) and describe the goals and capabilities of both the attacker and the detector in our adversarial-training setup.

Attacker’s Goals and Capabilities. Given an MGT passage, the attacker aims to *humanize* it to deceive the detector and outputs a rewrite. We call rewrites *from MGT inputs* as *adversarial examples*, regardless of success. During training only, we also generate auxiliary rewrites for HWTs as data augmentation and treat them as *MGT-labeled* samples for detector optimization. We do not regard these HWT-based rewrites as attacks, and they never define the attacker’s objective. To reflect real-world deployment where the detector is accessed only via an API that returns **top-1 predicted labels**, we consider a black-box setting in which the attacker has no access to the detector’s weights, architecture, or any proprietary training data beyond the provided few-shot set, and can

observe only the output labels via queries. The attacker is allowed to leverage (i) any publicly available LMs/LLMs, including both open-source models and closed-source/commercial models accessed via APIs, and (ii) previously generated rewrites together with detector-returned labels to synthesize stronger adversarial examples. The attacker makes **one** detector query per generated rewrite (including auxiliary rewrites during training).

Detector’s Goals and Capabilities. The detector aims to correctly distinguish HWTs from MGTs while remaining robust to adversarial examples. During training, the detector has access only to a few-shot labeled training set (containing both HWTs and MGTs), the adversarial examples produced from MGT inputs, and the auxiliary rewrites generated from HWT inputs for data augmentation. We further assume that the detector has no *direct access* to LLMs or large-scale external corpora beyond the provided training data. We emphasize that any LLM usage is confined to the attacker module for generating rewrites, while the deployed detector itself never queries LLMs and only optimizes on the resulting texts.

4 Methodology

In this section, we introduce our REACT framework. The architecture of REACT is shown in Figure 2. Our REACT contains an attacker and a detector. In the following subsections, we first describe the workflows of our attacker and detector, and then introduce the adversarial training procedure.

4.1 Attacker for Generating Adversarial Examples

Our attacker consists of a surrogate detector $\mathcal{D}_{\text{sur}}(\cdot)$ and a generator $\mathcal{G}(\cdot)$. To achieve the attacker’s goal in §3, we adopt a RAG-style humanization procedure to generate adversarial candidates, and update $\mathcal{D}_{\text{sur}}$ with an attacker loss after each training step to keep the retrieval scoring aligned with the target detector.

Retrieval-Augmented Generation. Given an input passage X , prior work often crafts an adversarial example $\tilde{X}^1$ via local perturbations such as

¹We use $\tilde{X}$ to denote a *rewrite* generated by the attacker for an input passage X . When X is an MGT, $\tilde{X}$ is an *adversarial example*; when X is an HWT (during training only), $\tilde{X}$ is an *auxiliary rewrite* used for data augmentation and is not considered an attack.

synonym substitution (Jin et al., 2020; Zhou et al., 2024) or mask-based replacement (Li et al., 2020), which may introduce unnatural phrasing and reduce readability. Instead, we generate $\tilde{X}$ by conditioning $\mathcal{G}$ on retrieved exemplars.

Specifically, we maintain a text-only retrieval pool (knowledge base) $\mathcal{B} = \{X_i\}_{i=1}^{|\mathcal{B}|}$, initialized with the few-shot training passages and continuously expanded with attacker-generated rewrites from previous steps. Let $P_{\text{sur}}(l | X_i)$ denote the probability assigned to label l by the surrogate, for retrieval, we score each X_i with $\mathcal{D}_{\text{sur}}$ by its MGT probability $p_i = P_{\text{sur}}(l = \text{MGT} | X_i)$, and use it as a scalar retrieval score $\mathbf{v}(X_i) = p_i$ to rank candidates in $\mathcal{B}$:

$$\begin{aligned} X^H &= \arg \min_{X_i \in \mathcal{B}} \mathbf{v}(X_i), \\ X^M &= \arg \max_{X_i \in \mathcal{B}} \mathbf{v}(X_i). \end{aligned} \quad (1)$$

Here, X^H and X^M correspond to the most human-like and most machine-like instances under $\mathcal{D}_{\text{sur}}$, serving as style anchors for rewriting. This retrieved pair provides in-context exemplars for RAG: X^H supplies human-writing cues to imitate, while X^M highlights machine-like patterns to avoid. We find that such *extreme* anchors provide the clearest style contrast under the surrogate’s scoring and we further study alternative prompting strategies in Appendix B.2. We then inject (X^H, X^M) into a prompt template $\mathcal{Z}(\cdot)$ and feed it to $\mathcal{G}$ to produce:

$$\tilde{X} = \mathcal{G}(\mathcal{Z}(X; X^H, X^M)). \quad (2)$$

For future retrieval, we insert only the rewritten text into the pool, i.e., $\mathcal{B} \leftarrow \mathcal{B} \cup \{\tilde{X}\}$, yielding a self-improving retrieval pool. We demonstrate in Appendix B.5 via human evaluation that our RAG strategy can effectively generate humanized texts, and we further provide a case study in Appendix B.6.

Attacker’s Loss. To align $\mathcal{D}_{\text{sur}}$ with the label-only feedback from $\mathcal{D}_{\text{tar}}$ in the black-box setting, for each generated rewrite $\tilde{X}$ we take $\hat{l} = \arg \max \mathcal{D}_{\text{tar}}(\tilde{X})$ as a pseudo label and minimize the negative log-likelihood under $\mathcal{D}_{\text{sur}}$:

$$\mathcal{L}_{\text{att}} = -\log P_{\text{sur}}(\hat{l} | \tilde{X}). \quad (3)$$

In each training step, we update the surrogate model using Eq. (3) while keeping the detector frozen. Detailed procedures are described in §4.3.

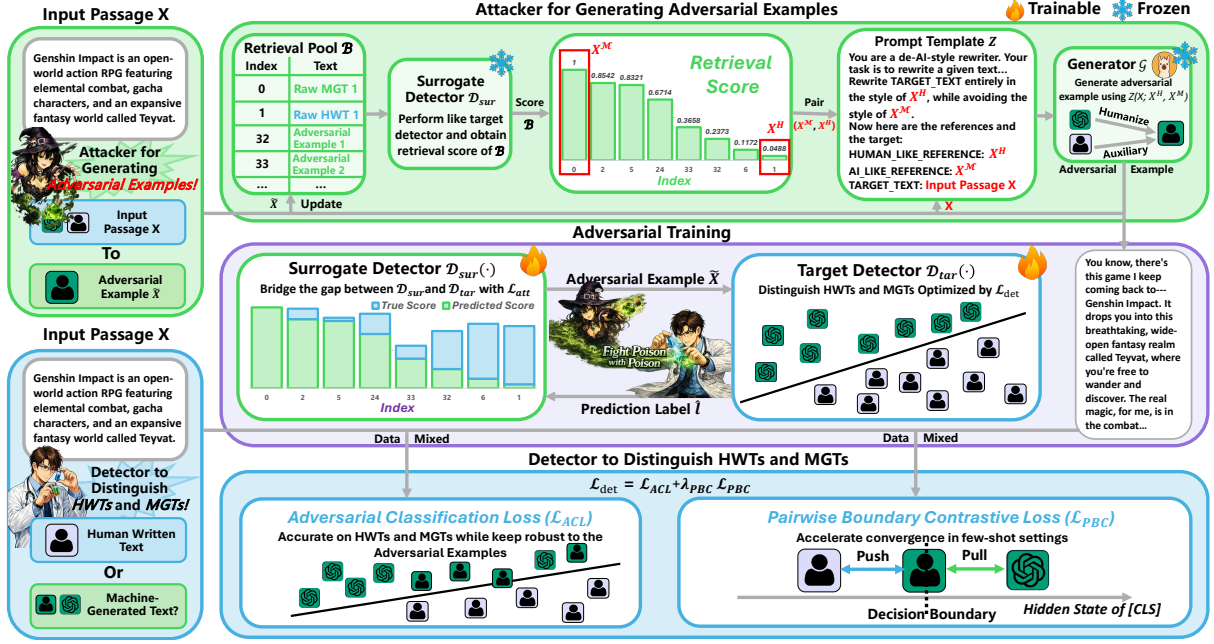


Figure 2: **Pipeline of REACT.** The attacker generates adversarial examples via RAG (§4.1), which are then fed into the detector (§4.2) and used in the adversarial training procedure to update the retrieval pool. (§4.3).

4.2 Detector for Distinguishing HWTs and MGTs

Our detector is a target classifier $\mathcal{D}_{tar}(\cdot)$ that distinguishes human-written texts (HWTs) from machine-generated texts (MGTs). Following the goal in §3, we train $\mathcal{D}_{tar}$ with two complementary objectives: an adversarial classification loss that enforces label correctness under humanization, and a pairwise boundary contrastive loss that stabilizes the local geometry between an input and its rewrite.

Adversarial Classification Loss. Given a training sample $(X, l) \in D_{train}$ and its attacker-generated humanized rewrite $\tilde{X}$, the detector is trained to correctly classify both the original input and the rewrite. Since $\tilde{X}$ is produced by the attacker as a machine-generated adversarial sample, we assign it the MGT label during detector training. Let $P_{tar}(l | X)$ denote the probability assigned by $\mathcal{D}_{tar}$ to label l . We define

$$\mathcal{L}_{ACL} = -\log P_{tar}(l | X) - \alpha \log P_{tar}(\text{MGT} | \tilde{X}), \quad (4)$$

where α controls the contribution of adversarial rewrites. This weighting is important in few-shot settings because the attacker continuously introduces additional MGT-labeled samples, which may otherwise amplify class imbalance and bias the detector toward rewrite-specific artifacts. By down-weighting the adversarial term, $\mathcal{L}_{ACL}$ improves robustness while preserving stable optimization on the limited clean data.

Pairwise Boundary Contrastive Loss. Adversarial classification alone does not explicitly constrain the representation geometry between clean inputs and humanized rewrites. To reduce clean-adversarial drift, we introduce a Pairwise Boundary Contrastive loss (PBC). Unlike batch-level contrastive objectives that require many informative negatives or memory banks (Liu et al., 2022, 2024), PBC operates on each pair $(X, \tilde{X})$, providing a low-variance signal suitable for scarce supervision.

Let $\mathbf{h}$ and $\tilde{\mathbf{h}}$ be the last-layer [CLS] representations of X and $\tilde{X}$ from $\mathcal{D}_{tar}$, and let $c = \cos(\text{norm}(\mathbf{h}), \text{norm}(\tilde{\mathbf{h}}))$ be their cosine similarity, where $\text{norm}(\mathbf{u}) = \mathbf{u} / \|\mathbf{u}\|_2$ denotes ℓ_2 normalization. The PBC loss is defined as

$$\mathcal{L}_{PBC} = \mathbb{I}[l = \tilde{l}] \cdot \max(0, (1 - \delta_{\text{same}}) - c) + \mathbb{I}[l \neq \tilde{l}] \cdot \max(0, c - \delta_{\text{diff}}), \quad (5)$$

where $\tilde{l}$ is the label assigned to $\tilde{X}$, and δ_{same} and δ_{diff} are the margins for same-label and different-label pairs. For same-label pairs, PBC keeps clean and humanized MGT close in representation space; for different-label pairs, it prevents HWT and MGT-labeled rewrites from collapsing into the same region. This directly stabilizes the local boundary that is otherwise fragile in few-shot adversarial training.

Total Loss. The final detector objective is

$$\mathcal{L}_{det} = \mathcal{L}_{ACL} + \lambda_{PBC} \mathcal{L}_{PBC}, \quad (6)$$

where λ_{PBC} balances classification robustness and boundary stabilization. At each training step, we update $\mathcal{D}_{\text{tar}}$ by minimizing Eq. (6) with the attacker frozen, as detailed in §4.3.

4.3 Adversarial Training

We propose an adversarial training framework where the attacker and the detector are updated *alternately* within each training step, enabling a co-evolution process. Unlike prior adversarial-training approaches (Hu et al., 2023; Koike et al., 2024; Li et al., 2025), our attacker adopts a dynamic strategy that generates adversarial examples on the fly. As training proceeds, the attacker leverages increasingly diverse texts in the retrieval pool $\mathcal{B}$ to produce more human-like adversarial rewrites, allowing the detector to learn richer humanization patterns. Meanwhile, the detector is optimized with the ACL and PBC objectives to balance accuracy and robustness in the few-shot settings.

Training Process. At each training step t , the attacker first generates a rewrite $\tilde{X}$ of the input passage X using the RAG procedure in Eq. (1) and Eq. (2). We then update the surrogate detector $\mathcal{D}_{\text{sur}}$ using the attacker loss in Eq. (3), while keeping $\mathcal{D}_{\text{tar}}$ frozen. Next, we update the target detector $\mathcal{D}_{\text{tar}}$ by minimizing the detector objective in Eq. (6), with $\mathcal{D}_{\text{sur}}$ frozen. Finally, we add the newly generated rewrite to the retrieval pool $\mathcal{B}$, so that subsequent steps can retrieve stronger anchors and produce more challenging adversarial examples. We detail the adversarial training process in form of pseudocode in Algorithm 1, and we further analyze the training dynamics during adversarial training in Appendix B.7, demonstrating the effectiveness of our co-evolution mechanism.

5 Experiment Results

We conduct experiments to evaluate the accuracy and robustness of REACT in few-shot settings.

5.1 Detection Performance in Few-shot Settings

Experimental setup. We conduct a comprehensive evaluation of REACT in few-shot settings over four datasets (DetectRL (Wu et al., 2024), OUTFOX (Koike et al., 2024), RAID (Dugan et al., 2024), and SemEval (Wang et al., 2024b)) and four shot sizes, focusing on detection accuracy. Our baselines include: (i) Metric-based detectors, including Fast DetectGPT (Bao et al., 2024), Binoc-

ulars (Hans et al., 2024), and LRR (Su et al., 2023); and (ii) Model-based detectors, including RoBERTa-Base (Liu et al., 2019), CoCo (Liu et al., 2022), PECOLA (Liu et al., 2024), RADAR (Hu et al., 2023), and GREATER (Li et al., 2025). For all model-based detectors, we use the same RoBERTa-Base backbone to ensure a fair comparison. For our method, we mainly adopt meta-llama/Llama-3.2-3B-Instruct (Grattafiori et al., 2024) as the generator $\mathcal{G}$. The decoding configuration is: maximum input length 4096, maximum new tokens 512, temperature 0.7, and top- p 0.9. We adopt ALBERT-Base-v2 (Lan et al., 2019) as the surrogate detector $\mathcal{D}_{\text{sur}}$, and the prompt template $\mathcal{Z}$ can be found in Figure 4. To improve the robustness of our conclusions, we train each model with training sets sampled using three random seeds, and evaluate all models on the same test set. Detailed descriptions of the baseline models and implementation details are provided in the Appendix A.4. For every model-based detector, method-specific hyperparameters are coarsely selected using only its few-shot training partition and then frozen before held-out evaluation; no attacked test example or test metric is used. For REACT, the defaults are selected once on the RAID 64-shot training partition and its training-time adversarial rewrites, then transferred unchanged to all datasets, shot budgets, seeds, and held-out attacks. Appendix B.10 gives the training-only sensitivity analysis. We further investigate the impact of different surrogate detector $\mathcal{D}_{\text{sur}}$ in Appendix B.9 and different generator $\mathcal{G}$ in Appendix B.13.

Experiment results. Table 1 reports the few-shot detection results. We summarize three findings. **1) Strong overall few-shot detection performance.** On average, REACT achieves **93.60** accuracy and **93.72** F1, outperforming the best metric-based detector Binoculars by **+7.66** accuracy and **+8.70** F1, and surpassing the strongest model-based baseline by **+5.19** accuracy and **+4.95** F1. This shows that REACT is effective not only against attacks, but also as a strong few-shot detector under limited supervision. **2) More stable optimization under scarce supervision.** REACT exhibits lower variance across random seeds, suggesting that boundary-oriented adversarial training improves few-shot stability. For example, on DetectRL at 32-shot, RADAR fluctuates heavily with $87.76_{\pm 12.49}$ accuracy, while REACT remains stable at $99.46_{\pm 0.58}$. A similar pattern ap-

Dataset	Shot	Metric	Metric-based Detectors			Model-based Detectors					
			Binoculars	Fast DetectGPT	LRR	RoBERTa-Base	CoCo [†]	PECOLA [†]	RADAR*	GREATER*	REACT (Ours)*
DetectRL	32	Acc ↑	88.38 $\pm$ 0.06	86.90 $\pm$ 0.22	76.47 $\pm$ 0.68	86.08 $\pm$ 0.96	92.20 $\pm$ 3.42	50.24 $\pm$ 0.42	87.76 $\pm$ 12.49	90.17 $\pm$ 1.64	99.46$\pm$0.58
		F1 ↑	87.62 $\pm$ 0.61	86.10 $\pm$ 0.59	73.04 $\pm$ 2.60	83.82 $\pm$ 1.30	91.48 $\pm$ 4.08	0.96 $\pm$ 1.67	89.00 $\pm$ 9.98	89.11 $\pm$ 2.02	99.46$\pm$0.58
	64	Acc ↑	88.50 $\pm$ 0.49	86.65 $\pm$ 0.52	76.93 $\pm$ 0.58	99.43 $\pm$ 0.61	89.83 $\pm$ 1.38	93.78 $\pm$ 5.48	99.06 $\pm$ 0.71	99.12 $\pm$ 0.27	99.82$\pm$0.27
		F1 ↑	88.01 $\pm$ 0.57	86.02 $\pm$ 0.05	74.37 $\pm$ 1.68	99.43 $\pm$ 0.62	88.66 $\pm$ 1.71	93.16 $\pm$ 6.50	99.06 $\pm$ 0.69	99.11 $\pm$ 0.28	99.82$\pm$0.27
	128	Acc ↑	88.60 $\pm$ 0.52	86.97 $\pm$ 0.06	77.10 $\pm$ 0.39	95.54 $\pm$ 6.69	99.61 $\pm$ 0.15	99.93$\pm$0.11	98.70 $\pm$ 1.65	99.48 $\pm$ 0.27	<u>99.74$\pm$0.07</u>
		F1 ↑	88.14 $\pm$ 0.17	86.17 $\pm$ 0.30	74.69 $\pm$ 1.30	94.98 $\pm$ 7.66	99.61 $\pm$ 0.15	99.93$\pm$0.11	98.66 $\pm$ 1.71	99.48 $\pm$ 0.27	<u>99.74$\pm$0.08</u>
	256	Acc ↑	88.87 $\pm$ 0.18	87.03 $\pm$ 0.13	77.40 $\pm$ 0.13	98.08 $\pm$ 2.78	99.87 $\pm$ 0.07	99.97$\pm$0.03	98.94 $\pm$ 1.31	99.95 $\pm$ 0.05	99.79 $\pm$ 0.16
		F1 ↑	88.38 $\pm$ 0.22	86.23 $\pm$ 0.44	75.02 $\pm$ 0.37	97.99 $\pm$ 2.94	99.87 $\pm$ 0.07	99.97$\pm$0.03	98.92 $\pm$ 1.35	99.95 $\pm$ 0.05	99.79 $\pm$ 0.16
	Avg.	Acc ↑	88.59 $\pm$ 0.34	86.84 $\pm$ 0.23	76.97 $\pm$ 0.44	94.78 $\pm$ 2.76	95.38 $\pm$ 1.26	85.98 $\pm$ 1.51	96.14 $\pm$ 4.04	97.15 $\pm$ 0.56	99.70$\pm$0.27
		F1 ↑	88.04 $\pm$ 0.39	86.13 $\pm$ 0.32	74.28 $\pm$ 1.46	94.05 $\pm$ 3.13	94.91 $\pm$ 1.52	73.50 $\pm$ 2.00	96.41 $\pm$ 3.43	96.91 $\pm$ 0.68	99.70$\pm$0.27
OUTFOX	32	Acc ↑	86.62 $\pm$ 0.51	80.45 $\pm$ 0.28	78.52 $\pm$ 0.29	<u>88.15$\pm$2.78</u>	53.17 $\pm$ 2.69	56.07 $\pm$ 10.47	77.34 $\pm$ 16.01	68.85 $\pm$ 9.14	95.64$\pm$1.56
		F1 ↑	86.60 $\pm$ 0.74	78.43 $\pm$ 0.28	77.14 $\pm$ 1.12	88.48 $\pm$ 2.00	68.12 $\pm$ 1.24	23.16 $\pm$ 39.94	82.51 $\pm$ 11.60	76.20 $\pm$ 5.09	95.77$\pm$1.42
	64	Acc ↑	86.67 $\pm$ 0.45	80.38 $\pm$ 0.19	78.42 $\pm$ 0.33	90.88 $\pm$ 1.74	93.31 $\pm$ 1.23	67.69 $\pm$ 15.73	93.62 $\pm$ 3.18	88.10 $\pm$ 4.06	97.12$\pm$0.05
		F1 ↑	86.07 $\pm$ 0.79	78.19 $\pm$ 0.10	77.16 $\pm$ 1.46	90.54 $\pm$ 2.29	93.40 $\pm$ 0.90	67.88 $\pm$ 8.74	93.33 $\pm$ 3.81	88.04 $\pm$ 3.66	97.16$\pm$0.02
	128	Acc ↑	86.70 $\pm$ 0.61	80.20 $\pm$ 0.05	78.53 $\pm$ 0.25	87.65 $\pm$ 9.64	95.80 $\pm$ 1.94	92.76 $\pm$ 5.57	95.30 $\pm$ 2.26	95.57 $\pm$ 2.65	96.73$\pm$0.48
		F1 ↑	86.06 $\pm$ 0.85	78.18 $\pm$ 0.09	77.02 $\pm$ 0.85	88.01 $\pm$ 8.84	<u>95.64$\pm$2.12</u>	93.29 $\pm$ 4.62	95.39 $\pm$ 2.10	95.38 $\pm$ 2.93	96.75$\pm$0.56
	256	Acc ↑	86.82 $\pm$ 0.13	80.27 $\pm$ 0.03	78.35 $\pm$ 0.30	97.56 $\pm$ 0.82	97.93 $\pm$ 0.37	98.14$\pm$0.55	97.14 $\pm$ 0.92	98.01 $\pm$ 0.73	<u>98.03$\pm$0.52</u>
		F1 ↑	86.21 $\pm$ 0.15	78.22 $\pm$ 0.10	76.69 $\pm$ 1.03	97.55 $\pm$ 0.81	97.93 $\pm$ 0.38	98.15$\pm$0.55	97.10 $\pm$ 0.98	98.02 $\pm$ 0.73	<u>98.04$\pm$0.51</u>
	Avg.	Acc ↑	86.70 $\pm$ 0.42	80.33 $\pm$ 0.14	78.45 $\pm$ 0.29	<u>91.05$\pm$3.74</u>	85.05 $\pm$ 1.55	78.62 $\pm$ 8.08	90.85 $\pm$ 5.61	87.68 $\pm$ 4.15	96.88$\pm$0.65
		F1 ↑	86.08 $\pm$ 0.63	78.26 $\pm$ 0.14	77.00 $\pm$ 1.11	91.14 $\pm$ 3.49	88.77 $\pm$ 1.16	70.62 $\pm$ 13.46	<u>92.08$\pm$4.60</u>	89.41 $\pm$ 3.14	96.94$\pm$0.62
RAID	32	Acc ↑	<u>84.07$\pm$0.16</u>	86.40$\pm$0.48	74.19 $\pm$ 0.61	72.09 $\pm$ 4.59	53.40 $\pm$ 2.92	50.73 $\pm$ 1.06	56.85 $\pm$ 10.44	53.91 $\pm$ 5.76	79.30 $\pm$ 0.90
		F1 ↑	<u>82.31$\pm$0.20</u>	86.25$\pm$0.32	70.36 $\pm$ 1.91	65.97 $\pm$ 13.25	66.99 $\pm$ 1.25	10.31 $\pm$ 16.39	65.78 $\pm$ 2.02	65.94 $\pm$ 1.39	80.18 $\pm$ 0.86
	64	Acc ↑	83.37 $\pm$ 1.37	86.42$\pm$0.38	74.46 $\pm$ 0.42	69.12 $\pm$ 6.59	78.24 $\pm$ 4.63	50.00 $\pm$ 0.00	69.97 $\pm$ 17.30	77.21 $\pm$ 12.42	<u>84.55$\pm$3.78</u>
		F1 ↑	81.80 $\pm$ 0.75	86.23$\pm$0.25	71.27 $\pm$ 1.47	61.36 $\pm$ 17.70	78.22 $\pm$ 5.30	44.44 $\pm$ 38.49	74.05 $\pm$ 6.48	80.12 $\pm$ 6.46	<u>83.63$\pm$4.11</u>
	128	Acc ↑	83.56 $\pm$ 0.91	86.45 $\pm$ 0.35	74.80 $\pm$ 0.47	<u>86.56$\pm$1.86</u>	80.16 $\pm$ 2.40	54.35 $\pm$ 7.44	59.99 $\pm$ 17.31	83.72 $\pm$ 1.80	87.42$\pm$0.86
		F1 ↑	82.07 $\pm$ 0.52	86.28 $\pm$ 0.22	72.34 $\pm$ 1.29	85.33 $\pm$ 3.05	78.05 $\pm$ 5.53	15.66 $\pm$ 26.78	71.63 $\pm$ 8.60	81.78 $\pm$ 3.86	87.86$\pm$0.45
	256	Acc ↑	83.53 $\pm$ 0.49	86.38 $\pm$ 0.12	74.71 $\pm$ 0.34	90.59 $\pm$ 2.03	90.58 $\pm$ 1.14	76.48 $\pm$ 11.54	77.15 $\pm$ 23.51	90.41 $\pm$ 3.19	92.43$\pm$1.52
		F1 ↑	82.08 $\pm$ 0.30	86.24 $\pm$ 0.09	72.04 $\pm$ 1.32	90.65 $\pm$ 1.58	90.79 $\pm$ 0.89	70.21 $\pm$ 24.17	82.60 $\pm$ 13.81	90.07 $\pm$ 3.66	92.39$\pm$1.41
	Avg.	Acc ↑	83.63 $\pm$ 0.73	86.41$\pm$0.33	74.54 $\pm$ 0.46	79.59 $\pm$ 3.52	75.59 $\pm$ 2.77	57.89 $\pm$ 5.01	65.99 $\pm$ 17.14	76.31 $\pm$ 5.79	<u>85.93$\pm$1.76</u>
		F1 ↑	82.06 $\pm$ 0.44	86.25$\pm$0.22	71.50 $\pm$ 1.50	75.83 $\pm$ 8.40	78.51 $\pm$ 3.24	35.16 $\pm$ 26.43	73.51 $\pm$ 7.73	79.48 $\pm$ 3.84	<u>86.02$\pm$1.71</u>
SemEval	32	Acc ↑	84.50 $\pm$ 1.06	84.02 $\pm$ 1.10	79.65 $\pm$ 0.23	82.76 $\pm$ 1.91	61.85 $\pm$ 8.74	50.63 $\pm$ 1.10	85.42 $\pm$ 6.37	73.75 $\pm$ 17.10	89.01$\pm$3.53
		F1 ↑	83.73 $\pm$ 0.55	83.17 $\pm$ 0.64	79.32 $\pm$ 0.56	80.00 $\pm$ 3.19	72.57 $\pm$ 4.46	3.07 $\pm$ 5.32	87.32 $\pm$ 4.71	78.99 $\pm$ 10.13	89.57$\pm$3.23
	64	Acc ↑	84.92 $\pm$ 0.74	84.60 $\pm$ 1.00	79.57 $\pm$ 0.26	85.77 $\pm$ 3.69	87.27 $\pm$ 2.25	72.27 $\pm$ 19.29	89.40 $\pm$ 3.37	88.82 $\pm$ 2.35	90.97$\pm$2.59
		F1 ↑	83.93 $\pm$ 0.41	83.58 $\pm$ 0.65	78.95 $\pm$ 1.46	84.74 $\pm$ 4.80	87.06 $\pm$ 1.57	78.27 $\pm$ 10.22	88.80 $\pm$ 4.36	89.33 $\pm$ 1.68	<u>91.40$\pm$2.14</u>
	128	Acc ↑	84.85 $\pm$ 0.51	84.58 $\pm$ 0.82	79.70 $\pm$ 0.23	90.90 $\pm$ 2.79	88.90 $\pm$ 3.39	86.39 $\pm$ 4.03	88.09 $\pm$ 6.84	92.98$\pm$2.91	<u>92.38$\pm$2.70</u>
		F1 ↑	83.93 $\pm$ 0.18	83.61 $\pm$ 0.62	79.64 $\pm$ 0.58	90.84 $\pm$ 3.26	89.16 $\pm$ 3.92	87.84 $\pm$ 2.75	89.27 $\pm$ 5.46	93.06$\pm$2.88	<u>92.74$\pm$2.29</u>
	256	Acc ↑	85.17 $\pm$ 0.42	84.98 $\pm$ 0.31	79.60 $\pm$ 0.13	93.52 $\pm$ 1.99	<u>95.28$\pm$0.82</u>	91.86 $\pm$ 2.26	90.90 $\pm$ 2.46	95.62$\pm$1.91	95.17 $\pm$ 0.68
		F1 ↑	84.04 $\pm$ 0.09	83.87 $\pm$ 0.17	79.08 $\pm$ 0.24	93.76 $\pm$ 1.82	<u>95.38$\pm$0.78</u>	92.36 $\pm$ 1.80	90.79 $\pm$ 2.47	95.67$\pm$1.90	95.20 $\pm$ 0.78
	Avg.	Acc ↑	84.86 $\pm$ 0.68	84.55 $\pm$ 0.81	79.63 $\pm$ 0.21	88.24 $\pm$ 2.59	83.38 $\pm$ 3.80	75.29 $\pm$ 6.67	88.45 $\pm$ 4.76	87.79 $\pm$ 6.07	91.88$\pm$2.37
		F1 ↑	83.91 $\pm$ 0.31	83.56 $\pm$ 0.52	79.25 $\pm$ 0.71	87.33 $\pm$ 3.27	86.04 $\pm$ 2.68	65.39 $\pm$ 5.02	89.04 $\pm$ 4.25	<u>89.26$\pm$4.15</u>	92.23$\pm$2.11
Overall	Avg.	Acc ↑	85.94 $\pm$ 0.54	84.54 $\pm$ 0.38	77.40 $\pm$ 0.35	<u>88.41$\pm$3.15</u>	84.84 $\pm$ 2.35	74.46 $\pm$ 5.32	85.35 $\pm$ 7.88	87.23 $\pm$ 4.14	93.60$\pm$1.26
		F1 ↑	85.02 $\pm$ 0.44	83.55 $\pm$ 0.31	75.51 $\pm$ 1.20	87.09 $\pm$ 4.57	87.06 $\pm$ 2.15	61.17 $\pm$ 11.75	87.76 $\pm$ 5.01	<u>88.77$\pm$2.94</u>	93.72$\pm$1.18

Table 1: **Detection performance across 4 datasets and shot sizes.** We report the mean and standard deviation of accuracy (Acc) and F1 score (F1) over 3 random seeds for each experiment (mean $\pm$ std). [†] denotes few-shot-tailored detectors, and * indicates adversarially trained detectors. The best result in each row is **bolded**, and the second-best is underlined. We report TPR@FPR=1% in Appendix B.3.

pears on OUTFOX at 32-shot, where RADAR obtains 77.34 $\pm$ 16.01 accuracy but REACT reaches 95.64 $\pm$ 1.56. **3) Smooth recovery with increasing supervision.** REACT is already competitive at 32-shot and improves consistently as more labeled examples become available. On the challenging RAID benchmark, its accuracy increases from 79.30 $\pm$ 0.90 at 32-shot to 92.43 $\pm$ 1.52 at 256-shot. At 64-shot, REACT improves over GREATER by **+7.34** points and over CoCo by **+6.31** points; at 128-shot, it still surpasses GREATER by **+3.70** points. These results support our view of few-shot MGT detection as rapid robustness recovery: RE-

ACT can quickly regain reliable decision boundaries from limited fresh supervision. We next report OOD transfer in the main text; cross-lingual and runtime analyses remain in Appendix B.11 and Appendix B.12.

5.2 Out-of-Distribution Generalization

We train on OUTFOX and evaluate on the disjoint HC3 test set, using the same label mapping and preprocessing as the in-domain experiments. This setting probes cross-domain transfer without exposing any HC3 instance during training.

Table 2 shows that REACT remains stable across

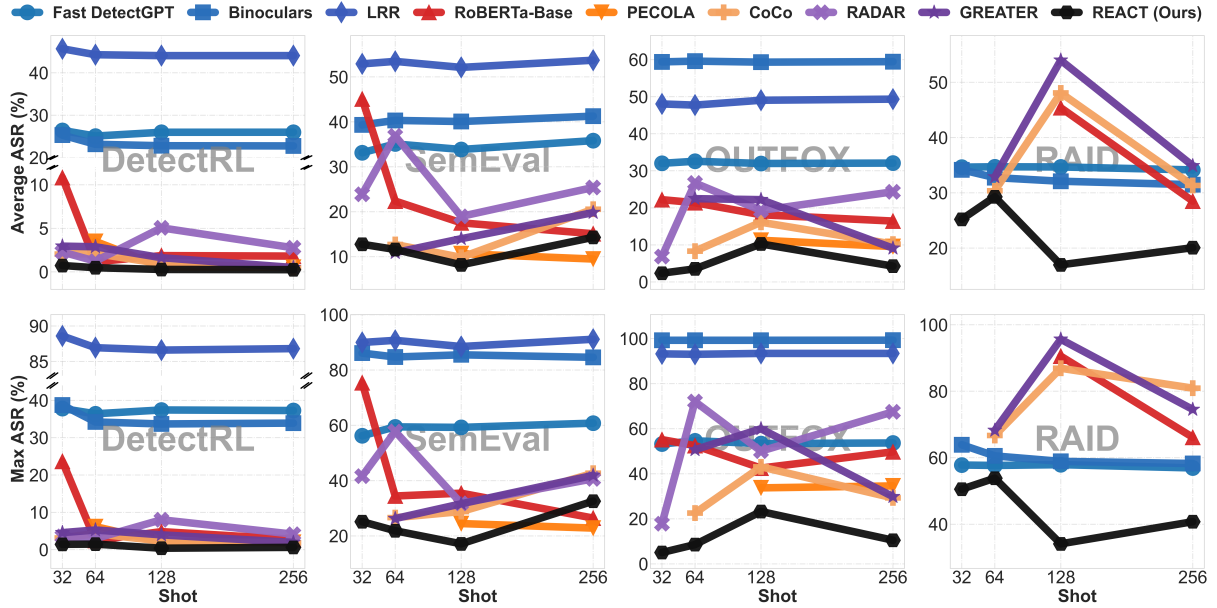


Figure 3: **Average (top) and maximum ASR (bottom) (%) across 4 datasets and 4 shot sizes under 4 attacks.** A lower ASR (%) indicates better model performance. We only include methods whose clean-test accuracy is above 75 in Table 1, since ASR is not informative when a detector is close to random guessing.

Train	Shot	Metric	RoBERTa	CoCo [†]	PECOLA [†]	RADAR*	GREATER*	REACT*
OUTFOX	32	Acc $\uparrow$	50.90 $\pm$ 3.55	50.07 $\pm$ 0.06	49.58 $\pm$ 0.73	35.40 $\pm$ 0.00	49.32 $\pm$ 8.88	87.03$\pm$3.27
		F1 $\uparrow$	11.74 $\pm$ 16.36	66.70 $\pm$ 0.03	1.34 $\pm$ 2.32	4.34 $\pm$ 0.00	45.54 $\pm$ 39.45	88.05$\pm$2.48
	64	Acc $\uparrow$	49.79 $\pm$ 1.42	44.58 $\pm$ 13.41	51.01 $\pm$ 1.59	35.40 $\pm$ 0.00	45.59 $\pm$ 5.92	86.82$\pm$2.21
		F1 $\uparrow$	8.92 $\pm$ 8.98	43.99 $\pm$ 22.72	22.97 $\pm$ 38.95	4.34 $\pm$ 0.00	17.27 $\pm$ 28.59	87.79$\pm$2.08
	128	Acc $\uparrow$	60.40 $\pm$ 15.12	54.41 $\pm$ 7.82	57.81 $\pm$ 7.97	35.40 $\pm$ 0.00	55.35 $\pm$ 9.28	85.45$\pm$5.13
		F1 $\uparrow$	40.25 $\pm$ 40.83	25.16 $\pm$ 22.56	49.84 $\pm$ 26.10	4.34 $\pm$ 0.00	17.12 $\pm$ 28.81	86.84$\pm$4.00
	256	Acc $\uparrow$	48.40 $\pm$ 2.16	60.97 $\pm$ 5.32	50.46 $\pm$ 5.04	35.40 $\pm$ 0.00	56.25 $\pm$ 1.82	87.09$\pm$1.96
		F1 $\uparrow$	12.58 $\pm$ 11.00	41.00 $\pm$ 16.65	51.96 $\pm$ 5.54	4.34 $\pm$ 0.00	34.63 $\pm$ 9.62	87.53$\pm$0.87
	Avg.	Acc $\uparrow$	52.37 $\pm$ 5.56	52.51 $\pm$ 6.65	52.21 $\pm$ 3.83	35.40 $\pm$ 0.00	51.63 $\pm$ 6.47	86.60$\pm$3.14
		F1 $\uparrow$	18.37 $\pm$ 19.29	44.21 $\pm$ 15.49	31.53 $\pm$ 18.23	4.34 $\pm$ 0.00	28.64 $\pm$ 26.62	87.55$\pm$2.36

Table 2: **OOD detection performance when training on OUTFOX and testing on HC3.** Values are mean $\pm$ std over three seeds. [†] denotes few-shot-tailored detectors and * adversarially trained detectors.

all four shot budgets, averaging **86.60** Acc and **87.55** F1. In contrast, all baselines average at most 52.51 Acc; several also show high F1 variance or degenerate predictions. The transfer gain suggests that humanization-oriented training and local boundary stabilization reduce reliance on dataset-specific artifacts.

5.3 Robustness against Attacks

Experimental setup. We evaluate the robustness of detectors against four strong attacks under few-shot settings on the test sets of the above four datasets. The evaluated attacks include Modify Attack (Wang et al., 2024a), Paraphrasing (Krishna et al., 2023), Back-translation (Sennrich et al., 2015), and Humanize Attack (Wang et al., 2024a).

Detailed descriptions of these attack methods are provided in the Appendix A.5.

Experiment results. We report robustness against four strong attacks in Figure 3 and summarize three findings. **1) Strong average-case robustness.** REACT achieves the lowest average ASR in 14 out of 16 dataset-shot settings. Across all datasets and shot budgets, it obtains an average ASR of **10.04%**, reducing ASR by **3.66** percentage points compared with the strongest baseline in each setting. The gains are especially clear in the most data-scarce regime, with an average reduction of **6.47** percentage points at 32-shot. **2) Strong worst-case robustness.** REACT also achieves the lowest maximum ASR in 15 out of 16 settings, showing that its robustness is not lim-

Shot	RoBERTa	w/o $\mathcal{L}_{PBC}$	w/o RAG	REACT
32	82.27 / 26.04 [†]	87.64 / 26.24	85.68 / 24.87	90.85 / 10.27
64	86.29 / 14.98 [†]	89.49 / 27.89	91.02 / 26.16	93.12 / 11.22
128	90.16 / 20.74	93.41 / 17.54	93.02 / 22.74	94.07 / 8.93
256	94.94 / 15.47	95.45 / 11.68	95.97 / 20.35	96.36 / 9.73
Avg.	88.41 / 21.36	91.50 / 20.84	91.42 / 23.53	93.60 / 10.04

Table 3: **Compact ablation study.** Acc / Avg ASR (%) averaged over four datasets. [†]: RAID ASR for RoBERTa at 32/64-shot is omitted because clean accuracy is below 75%. Dataset-wise results are in Appendix B.1.

ited to easier attacks. On average, it reduces maximum ASR by **8.20** percentage points, with large improvements on challenging cases such as OUT-FOX at 256-shot and RAID at 128-shot. **3) Consistent gains across supervision budgets.** The ASR reductions remain positive across all shot sizes, indicating that the benefit is not a low-shot artifact. For average ASR, the reductions are **6.47**, **1.46**, **4.52**, and **2.18** percentage points at 32, 64, 128, and 256 shots, respectively; for maximum ASR, the corresponding reductions are **9.56**, **5.80**, **10.84**, and **6.60** percentage points. Overall, these results suggest that the boundary-oriented adversarial curriculum in REACT improves both average-case and worst-case robustness under humanization attacks.

5.4 Ablation Study

To examine the contribution of each component, we conduct a compact ablation study averaged over four datasets. We compare the full REACT with a standard RoBERTa baseline and two variants: removing the pairwise boundary contrastive loss (**w/o $\mathcal{L}_{PBC}$**) and disabling retrieval guidance in the attacker (**w/o RAG**). Full dataset-wise ablation results are provided in Appendix B.1.

Table 3 shows that both modules are necessary for the accuracy–robustness trade-off. RoBERTa obtains weaker average performance, especially in low-shot regimes, confirming that ordinary supervised fine-tuning is insufficient for robust few-shot recovery. Removing RAG increases Avg ASR from **10.04** to **23.53**, showing that direct prompting produces less boundary-oriented adversarial rewrites. Removing $\mathcal{L}_{PBC}$ also degrades robustness, increasing Avg ASR to **20.84** and reducing accuracy from **93.60** to **91.50**. Across all shot budgets, the full REACT achieves the best averaged Acc / ASR, confirming the contribution of retrieval-guided hard examples.

Method	F1@8/16/32	Flip↓	Drift↓	Margin↑
RoBERTa	.171/.490/.729	.306	1.270	-.265
w/o pairwise stab.	.703/.703/.721	.272	1.023	-.095
Direct-AT	.660/.756/.737	.240	.749	.237
REACT	.735/.763/.762	.204	.516	.373

Table 4: **Few-shot convergence and boundary stability on RAID.** Margin is the representation class margin; larger values place adversarial MGT closer to MGT than HWT.

Pairwise stabilization is also necessary for robust few-shot training.

5.5 Boundary Stability Analysis

We directly test whether the gains reflect a more stable local decision geometry rather than only higher final accuracy. Prediction Flip records attack-induced label changes, Logit Drift measures confidence changes, and Representation Class Margin compares an adversarial MGT’s distance to the HWT and MGT representation regions. Larger margins and smaller flips/drift are better; Appendix B.4 provides the formal definitions and representation-distance analysis.

Compared with Direct-AT, REACT reduces Flip from .240 to .204 and Drift from .749 to .516, while increasing the representation class margin from .237 to .373. It also reaches the strongest F1 at 8, 16, and 32 shots, supporting the proposed boundary-stabilization mechanism.

6 Conclusion

We study robust few-shot MGT detection as rapid robustness recovery under evolving generators, domains, and humanization-oriented attacks. We identify unstable local boundary estimation as a key failure mode, where scarce supervision causes clean–adversarial drift and vulnerability to humanized rewrites. To address this, we propose **RAG-GuidEd Attacker Strengthens ConTrastive Few-shot Detector (REACT)**, which couples boundary-oriented adversarial generation with pairwise boundary stabilization. REACT uses retrieval-guided hard examples and a pairwise boundary contrastive objective to stabilize clean–adversarial local geometry. Experiments on 4 datasets with 4 shot sizes and 3 random seeds show that REACT improves both accuracy and robustness over 8 SOTA detectors.

Limitations

Despite the strong performance of REACT, it still has several limitations. **First**, while REACT performs well on English benchmarks, extending it to multilingual settings remains challenging. **Second**, although adversarial training yields clear improvements, it also increases training time and incurs additional computational and resource overhead during training. **Third**, the detector in our work operates at the document level and therefore cannot localize machine-generated content within a human–AI co-authored text. Developing fine-grained attribution methods, such as sentence-level or token-level detection, remains an important direction for future work.

Ethics Statement

This work is defensive: it studies humanization-style attacks in order to harden MGT detectors. Because human-like rewriting is dual-use, we disclose the scientific components needed to reproduce the defense but do not provide a turnkey evasion pipeline, deployment-ready attack tooling, or systematically optimized prompt ensembles. We encourage use of MGT detectors as one signal in a broader safety process with human review and due process.

All experiments use public benchmarks; we collect no private user data. The human study in Appendix B.5 involved three consenting adult annotators who viewed public benchmark passages and model-generated rewrites, could stop at any time, and were compensated at twice the local minimum hourly wage. Adversarial training adds compute overhead, which we limit through fixed budgets and report in Appendix B.12. Additional disclosure, data-handling, and release safeguards are documented in Appendix D.

Acknowledgments

We thank the anonymous reviewers for their constructive comments and suggestions, which helped improve the clarity and presentation of this work. We are also grateful to the Area Chair (AC) and Senior Area Chair (SAC) for their careful assessment, thoughtful guidance, and coordination throughout the review process.

References

- Guangsheng Bao, Yanbin Zhao, Zhiyang Teng, Linyi Yang, and Yue Zhang. 2024. Fast-detectgpt: Efficient zero-shot detection of machine-generated text via conditional probability curvature. In *ICLR*.
- Battista Biggio and Fabio Roli. 2018. Wild patterns: Ten years after the rise of adversarial machine learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 2154–2156.
- DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, and 245 others. 2025. *Deepseek-v3.2: Pushing the frontier of open large language models*. Preprint, arXiv:2512.02556.
- Liam Dugan, Alyssa Hwang, Filip Trhlík, Andrew Zhu, Josh Magnus Ludan, Hainiu Xu, Daphne Ippolito, and Chris Callison-Burch. 2024. 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)*, pages 12463–12492.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. *The llama 3 herd of models*. Preprint, arXiv:2407.21783.
- Biyang Guo, Xin Zhang, Ziyuan Wang, Minqi Jiang, Jinran Nie, Yuxuan Ding, Jianwei Yue, and Yupeng Wu. 2023. How close is chatgpt to human experts? comparison corpus, evaluation, and detection. *arXiv preprint arXiv:2301.07597*.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. *Deepseek-r1 incentivizes reasoning in llms through reinforcement learning*. *Nature*, 645(8081):633–638.
- Abhimanyu Hans, Avi Schwarzschild, Valeriia Cherepanova, Hamid Kazemi, Aniruddha Saha, Micah Goldblum, Jonas Geiping, and Tom Goldstein. 2024. Spotting llms with binoculars: zero-shot detection of machine-generated text. In *Proceedings of the 41st International Conference on Machine Learning*, pages 17519–17537.
- Xiaomeng Hu, Pin-Yu Chen, and Tsung-Yi Ho. 2023. Radar: Robust ai-text detection via adversarial learning. *Advances in neural information processing systems*, 36:15077–15095.

- Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025.
- Ryuto Koike, Masahiro Kaneko, and Naoaki Okazaki. 2024. Outfox: Llm-generated essay detection through in-context learning with adversarially generated examples. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 21258–21266.
- Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. 2023. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. *Advances in Neural Information Processing Systems*, 36:27469–27500.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.
- Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. Bert-attack: Adversarial attack against bert using bert. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6193–6202.
- Yuanfan Li, Zhaohan Zhang, Chengzhengxu Li, Chao Shen, and Xiaoming Liu. 2025. Iron sharpens iron: Defending against attacks in machine-generated text detection with adversarial training. *arXiv preprint arXiv:2502.12734*.
- Yuanfan Li, Qi Zhou, and Zexuan Xie. 2026. [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*.
- Shengchao Liu, Xiaoming Liu, Yichen Wang, Zehua Cheng, Chengzhengxu Li, Zhaohan Zhang, Yu Lan, and Chao Shen. 2024. 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)*, pages 1874–1889.
- Xiaoming Liu, Zhaohan Zhang, Yichen Wang, Hang Pu, Yu Lan, and Chao Shen. 2022. Coco: Coherence-enhanced machine-generated text detection under data limitation with contrastive learning. *arXiv preprint arXiv:2212.10341*.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.
- OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. [Gpt-4 technical report](#). *Preprint*, arXiv:2303.08774.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. [Exploring the limits of transfer learning with a unified text-to-text transformer](#). *Journal of Machine Learning Research*, 21(140):1–67.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*.
- Jinyan Su, Terry Zhuo, Di Wang, and Preslav Nakov. 2023. Detectllm: Leveraging log rank information for zero-shot detection of machine-generated text. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 12395–12412.
- Shantanu Thorat and Andrew Caines. 2025. Dactyl: Diverse adversarial corpus of texts yielded from large language models. *arXiv preprint arXiv:2508.00619*.
- Jörg Tiedemann and Santhosh Thottingal. 2020. OPUS-MT — Building open translation services for the World. In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation (EAMT)*, Lisbon, Portugal.
- Ved Umrajkar. 2025. Dac-lora: Dynamic adversarial curriculum for efficient and robust few-shot adaptation. *arXiv preprint arXiv:2509.20792*.
- Yichen Wang, Shangbin Feng, Abe Bohan Hou, Xiao Pu, Chao Shen, Xiaoming Liu, Yulia Tsvetkov, and Tianxing He. 2024a. Stumbling blocks: Stress testing the robustness of machine-generated text detectors under attacks. *arXiv preprint arXiv:2402.11638*.
- Yuxia Wang, Jonibek Mansurov, Petar Ivanov, Jinyan Su, Artem Shelmanov, Akim Tsvigun, Osama Mohammed Afzal, Tarek Mahmoud, Giovanni Puccetti, and Thomas Arnold. 2024b. [SemEval-2024 task 8: Multidomain, multimodel and multilingual machine-generated text detection](#). In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, pages 2057–2079, Mexico City, Mexico. Association for Computational Linguistics.
- Junchao Wu, Runzhe Zhan, Derek Wong, Shu Yang, Xinyi Yang, Yulin Yuan, and Lidia Chao. 2024. Detectrl: Benchmarking llm-generated text detection in real-world scenarios. *Advances in Neural Information Processing Systems*, 37:100369–100401.

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

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. *Qwen3 technical report*. *Preprint*, arXiv:2505.09388.

Ying Zhou, Ben He, and Le Sun. 2024. Humanizing machine-generated content: Evading ai-text detection through adversarial attack. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 8427–8437.

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

A Experimental Setting

A.1 Dataset

We conduct experiments on four benchmark datasets: DetectRL (Wu et al., 2024), OUTFOX (Koike et al., 2024), RAID (Dugan et al., 2024), and SemEval (Wang et al., 2024b). For each dataset, we construct few-shot training sets by sampling $\{32, 64, 128, 256\}$ labeled instances using three random seeds $\{66, 2025, 9999\}$. For evaluation, we sample 2,000 *disjoint* instances (1,000 HWT + 1,000 MGT) as a balanced test set from the corresponding dataset split, ensuring no overlap with the few-shot training set. We additionally use HC3 (Guo et al., 2023) for OOD evaluation.

A.2 Implementation

REACT is deployed on a server equipped with two RTX 4090 GPUs running Ubuntu 22.04. Training time of our REACT with 32 shots is about 5 minutes. Our adversarial training framework mainly adopts meta-llama/Llama-3.2-3B-Instruct (Grattafiori et al., 2024) as the generator $\mathcal{G}$ and ALBERT-Base-v2 (12M) (Lan et al., 2019) as the surrogate detector $\mathcal{D}_{\text{sur}}$. We use RoBERTa-Base (125M) (Liu et al., 2019) as the backbone of the target detector $\mathcal{D}_{\text{tar}}$. For metric-based detectors, we train a logistic-regression calibrator on

the few-shot training set to map raw scores to calibrated probabilities (of the MGT class), and report Acc/F1 using a fixed decision threshold of 0.5. For model-based detectors, we employ the same RoBERTa-Base backbone to ensure a fair comparison, train each method for 6 epochs under an identical protocol, and report results using the final checkpoint with a fixed decision threshold of 0.5. All model-based detectors follow the same information-access constraint: method-specific hyperparameters may be coarsely tuned only on the corresponding few-shot training partition, after which they are frozen before held-out evaluation. We do not use attacked test examples, test labels, or test-time ASR for selection. Because a separate validation split would reduce and alter the prescribed few-shot budget, no additional validation subset is created. We employ the same meta-llama/Llama3.2-3B-Instruct for auxiliary generator of RADAR to ensure a fair comparison. Under this 6-epoch protocol, REACT uses a two-stage schedule: we first pre-train on clean samples using only the cross-entropy loss for 3 epochs, and then perform adversarial training for another 3 epochs. The selected hyperparameters of REACT are summarized in Table 5.

Hyperparameter	Value
Pre-training epochs $epoch_{\text{pre}}$	3
Maximum adversarial epochs $epoch_{\text{max}}$	3
Total epochs	6
Target detector learning rate η_{tar}	2×10^{-5}
Surrogate detector learning rate η_{sur}	2×10^{-5}
AdamW weight decay β	0.03
Batch size M	2
Max sequence length L (for $\mathcal{D}_{\text{tar}}, \mathcal{D}_{\text{sur}}$)	512
Adversarial CE scaling coefficient α	0.5
PBC loss weight λ_{pbc}	1.2
PBC margin for same-class δ_{same}	0.1
PBC margin for diff-class δ_{diff}	0.3
Generator max new tokens T_{new}	512
Generator input max length L_{gen}	4096
Sampling temperature τ	0.7
Nucleus sampling p	0.9

Table 5: Hyperparameters for our REACT.

We use the following prompt template $\mathcal{Z}(\cdot)$ in Figure 4. We do not use any jailbreak prompt. The prompt template shown in Figure 4 is the actual template used in our implementation rather than a simplified illustration. We manually inspected the generated adversarial samples and did not observe refusal-style outputs in our runs. For reproducibil-

ity, we also define a filtering rule: if the generation is empty, refuses the instruction, or fails to preserve the source meaning, the sample is discarded and regenerated using the same template.

A.3 Evaluation Metrics

To evaluate detection performance, we report *accuracy* (Acc) and *F1-score* (F1). To quantify robustness against adversarial attacks, we use the *attack success rate* (ASR), where a lower ASR indicates stronger robustness. Under our threat model, the attacker perturbs only machine-generated texts (MGTs). Accordingly, ASR is defined as the fraction of originally correctly classified MGTs that become misclassified after the attack:

$$\text{ASR} = \frac{\#\{\text{MGTs misclassified after attack}\}}{\#\{\text{MGTs correctly classified before attack}\}}. \quad (7)$$

A.4 Experimental Comparison Methods

To better assess the effectiveness of REACT, we compare it with representative state-of-the-art (SOTA) MGT detectors, including three widely used *metric-based* detectors and five *model-based* detectors. Metric-based detectors typically output *raw scores* rather than calibrated probabilities. Following the standard practice in their original papers, we perform score calibration on the corresponding training set by fitting a logistic regression model that maps each detector’s raw score to an output probability for binary classification. Therefore, “few-shot” for training-free detectors should be interpreted as calibration under limited labeled validation data, not supervised detector training. For each metric-based detector, we follow the backbone choices recommended by its original paper (and official implementation) for the underlying language model(s). In particular, for methods that require an LM backbone, we use the paper-recommended scoring/sampling model for Fast DetectGPT, the recommended performer/observer pair for Binoculars, and the recommended scoring model for LRR.

- **Fast DetectGPT** (Bao et al., 2024): A metric-based zero-shot detector derived from DetectGPT, designed to reduce the computational overhead of perturbation-based scoring. Instead of repeatedly perturbing the input, it measures *conditional probability curvature*, which captures discrepancies in token-level likelihood geometry between human-written and machine-generated text. This single-pass

scoring strategy substantially accelerates inference while maintaining competitive detection quality.

- **Binoculars** (Hans et al., 2024): A metric-based zero-shot detector that detects machine-generated text by contrasting the output distributions of two closely related pretrained language models. It computes a lightweight divergence-style score that reflects subtle inconsistencies between human and machine writing, requiring neither fine-tuning nor labeled data.
- **LRR** (Su et al., 2023): A metric-based zero-shot detector that uses log-rank statistics under a reference language model. It identifies atypical token-rank patterns that commonly arise in LLM generations, offering a fast alternative to perturbation-heavy methods while retaining strong empirical performance.
- **RoBERTa-Base** (Liu et al., 2019): A standard supervised detector that fine-tunes an RoBERTa-Base classifier on the labeled training set and directly outputs class probabilities for binary MGT detection.
- **CoCo** (Liu et al., 2022): A few-shot-tailored detector that introduces contrastive training driven by *local coherence* signals. It first scores each input using local coherence computed by a set of in-domain language models, and then trains the detector with a contrastive objective to better separate human-written texts and machine-generated texts in representation space. Since CoCo is primarily designed for sample-efficient discrimination rather than adversarial robustness, it can be less robust under strong attacks in our robustness evaluation.
- **PECOLA** (Liu et al., 2024): A contrastive-learning-based detector that improves fine-tuned MGT detectors via *selective perturbation* and a contrastive objective to learn more discriminative representations. We treat PECOLA as a few-shot-tailored baseline due to its sample-efficient training design, but it is not explicitly adversarially trained, which can lead to weaker robustness under our attack suite.

Prompt template $\mathcal{Z}(X; X^H, X^M)$

Human-style reference: X^H

Machine-style reference: X^M

Target passage: X

Instruction:

You are a de-AI-style rewriter. Your task is to rewrite a given text so that it avoids being detected as AI-generated.

Your objectives:

- Preserve the original meaning, facts, and intent of the text.
- Make the rewritten text look as if it were written by a human.
- Strictly write in the SAME LANGUAGE as the TARGET_TEXT.
- Your maximum generation length is {max_tokens} tokens.

You will be given two reference texts:

1. HUMAN_LIKE_REFERENCE: a text (or several examples) that the detector considers the most human-like.
2. AI_LIKE_REFERENCE: a text (or several examples) that the detector considers the most AI-like.

Your style constraints:

- Imitate the style, tone, and rhythm of HUMAN_LIKE_REFERENCE.
- Explicitly avoid any stylistic patterns that resemble AI_LIKE_REFERENCE.
- Do NOT mention detectors, AI, models, prompts, or the rewriting process.
- Do NOT add explanations, comments, or meta-text. Output ONLY the rewritten text.

Steps you MUST follow internally (do NOT output the steps):

1. Carefully read HUMAN_LIKE_REFERENCE and understand its style and tone.
2. Briefly compare it in your mind with AI_LIKE_REFERENCE and identify stylistic differences.
3. Rewrite TARGET_TEXT entirely in the style of HUMAN_LIKE_REFERENCE, while avoiding the style of AI_LIKE_REFERENCE.

Now here are the references and the target:

HUMAN_LIKE_REFERENCE:

X^H

AI_LIKE_REFERENCE:

X^M

TARGET_TEXT:

X

Rewritten TARGET_TEXT (remember: output ONLY the rewritten text):

Figure 4: Prompt template used to construct $\mathcal{Z}(X; X^H, X^M)$ for RAG-based rewriting.

- **RADAR** (Hu et al., 2023): An adversarially trained detector that improves robustness against paraphrasing-style attacks by training the detector jointly with a paraphraser. The paraphraser generates low-perplexity paraphrases to evade detection, while the detector learns to correctly classify both original texts and adversarial paraphrases. We employ the same meta-llama/Llama3.2-3B-Instruct for auxiliary generator of RADAR to ensure a fair comparison.
- **GREATER** (Li et al., 2025): An adversarially trained detector within the GREATER framework, where an adversary identifies and perturbs critical tokens and searches for effective adversarial examples. GREATER updates the adversary and the detector synchronously so that the detector learns to defend against evolving attacks and generalize to other perturbations.

A.5 Detailed Information of Attack Method

Here we provide detailed descriptions of the 4 strong attack methods used in the main paper.

- **Modify Attack** (Wang et al., 2024a): A mask-filling style humanization attack implemented with **T5-large** (Raffel et al., 2020). Given an input MGT, we mask a subset of tokens and let T5-large fill in the blanks, producing a rewritten text that largely preserves the original semantics while injecting more human-like lexical and syntactic variations. This attack targets detectors that rely on surface cues or token-level likelihood patterns by introducing semantics-preserving perturbations.
- **Paraphrasing** (Krishna et al., 2023): A sentence-level paraphrase attack implemented with **DIPPER** (Krishna et al., 2023). For each MGT input, DIPPER rewrites the passage to preserve meaning while substantially altering phrasing and local syntax, yielding human-like paraphrases. This attack

aims to evade detection via semantic camouflage by replacing characteristic generation artifacts with more natural expressions.

- **Back-translation** (Sennrich et al., 2015): A sentence-level paraphrase attack that round-trip translates each MGT through a pivot language to induce lexical and syntactic diversity while preserving meaning. We use public **Helsinki-NLP OPUS-MT** models (Tiedemann and Thottingal, 2020) and choose **German** as the pivot language. Concretely, we translate the input text into German and then back to the original language, and replace the original text with the back-translated output.
- **Humanize Attack** (Wang et al., 2024a): An in-context learning based humanization attack that uses **three** (MGT, HWT) pairs as in-context exemplars to guide rewriting. Given an MGT input, the attacker prompts the generator with these paired exemplars to synthesize a highly human-like rewrite while keeping the underlying meaning. We use **DeepSeek V3.2** (DeepSeek-AI et al., 2025) as the generator for this attack.

B Additional Result

B.1 Full Ablation Study

Experiment Setup. To verify the role of each component in REACT, we conduct an ablation study that strictly follows the same experimental setting as Table 1 so that the conclusions are directly comparable. Under this controlled setup, we compare four variants in Table 6: a standard supervised **RoBERTa-Base** detector, a variant that removes the pairwise boundary-constraint loss $\mathcal{L}_{PBC}$ from Eq. (6) (**w/o $\mathcal{L}_{PBC}$**), a variant that disables RAG and instead uses direct prompting to generate adversarial rewrites (**w/o RAG**), and the full **REACT**. **We do not include the ablation that removes $\mathcal{L}_{ACL}$, because $\mathcal{L}_{ACL}$ is the classification loss; without it, the detector cannot be trained to perform the classification task.**

Experiment results. We report the ablation results in Table 6 and summarize three findings. Following Table 1, we report ASR only for detectors whose clean-test accuracy exceeds 75%, since ASR is not informative when a detector is close to random guessing (hence “–” for RoBERTa-Base on RAID at 32/64 shots).

Dataset	Shot	Metric	RoBERTa-Base	w/o $\mathcal{L}_{PBC}$	w/o RAG	REACT (Ours)
DetectRL	32	Acc $\uparrow$	86.08 $\pm$ 0.96	<u>96.89$\pm$4.88</u>	96.29 $\pm$ 2.09	99.46$\pm$0.58
		Avg ASR $\downarrow$	10.84	<u>1.36</u>	2.04	0.74
	64	Acc $\uparrow$	99.43 $\pm$ 0.61	<u>99.77$\pm$0.06</u>	99.72 $\pm$ 0.10	99.82$\pm$0.27
		Avg ASR $\downarrow$	1.16	<u>0.60</u>	0.81	0.48
	128	Acc $\uparrow$	95.54 $\pm$ 6.69	<u>99.72$\pm$0.07</u>	99.71 $\pm$ 0.05	99.74$\pm$0.07
		Avg ASR $\downarrow$	1.87	<u>0.46</u>	0.44	0.28
	256	Acc $\uparrow$	98.08 $\pm$ 2.78	<u>99.79$\pm$0.16</u>	<u>99.77$\pm$0.06</u>	99.79$\pm$0.16
		Avg ASR $\downarrow$	1.82	<u>0.43</u>	<u>0.38</u>	0.26
	Avg.	Acc $\uparrow$	94.78 $\pm$ 2.76	<u>99.04$\pm$1.29</u>	98.87 $\pm$ 0.58	99.70$\pm$0.27
		Avg ASR $\downarrow$	3.92	<u>0.71</u>	0.92	0.44
OUTFOX	32	Acc $\uparrow$	88.15 $\pm$ 2.78	91.72 $\pm$ 1.76	<u>92.20$\pm$3.17</u>	95.64$\pm$1.56
		Avg ASR $\downarrow$	22.20	<u>13.79</u>	15.66	2.36
	64	Acc $\uparrow$	90.84 $\pm$ 1.74	93.12 $\pm$ 6.09	<u>94.19$\pm$3.64</u>	97.12$\pm$0.05
		Avg ASR $\downarrow$	21.39	13.09	14.59	3.53
	128	Acc $\uparrow$	87.65 $\pm$ 9.64	95.41 $\pm$ 0.35	<u>96.18$\pm$2.65</u>	96.73$\pm$0.48
		Avg ASR $\downarrow$	18.14	<u>14.82</u>	15.52	10.26
	256	Acc $\uparrow$	<u>97.56$\pm$0.82</u>	97.03 $\pm$ 0.44	97.40 $\pm$ 0.34	98.03$\pm$0.52
		Avg ASR $\downarrow$	16.43	4.92	<u>4.87</u>	4.27
	Avg.	Acc $\uparrow$	91.05 $\pm$ 3.74	94.32 $\pm$ 2.16	<u>94.99$\pm$2.45</u>	96.88$\pm$0.65
		Avg ASR $\downarrow$	19.54	<u>11.66</u>	12.66	5.11
RAID	32	Acc $\uparrow$	72.09 $\pm$ 4.59	<u>76.35$\pm$7.45</u>	75.08 $\pm$ 0.96	79.30$\pm$0.90
		Avg ASR $\downarrow$	–	72.72	<u>65.14</u>	25.21
	64	Acc $\uparrow$	69.12 $\pm$ 6.59	77.43 $\pm$ 1.95	<u>80.78$\pm$1.27</u>	84.55$\pm$3.78
		Avg ASR $\downarrow$	–	83.90	<u>74.26</u>	29.28
	128	Acc $\uparrow$	86.56 $\pm$ 1.86	86.87 $\pm$ 3.30	<u>86.91$\pm$3.19</u>	87.42$\pm$0.86
		Avg ASR $\downarrow$	45.41	42.45	61.78	16.97
	256	Acc $\uparrow$	90.59 $\pm$ 2.03	91.01 $\pm$ 3.83	<u>92.29$\pm$0.98</u>	92.43$\pm$1.52
		Avg ASR $\downarrow$	28.51	26.49	61.65	20.07
	Avg.	Acc $\uparrow$	79.59 $\pm$ 3.77	82.92 $\pm$ 4.13	<u>83.77$\pm$1.60</u>	85.93$\pm$1.77
		Avg ASR $\downarrow$	<u>36.96</u>	56.39	65.71	22.89
SemEval	32	Acc $\uparrow$	82.76 $\pm$ 1.91	<u>85.60$\pm$5.29</u>	79.13 $\pm$ 7.85	89.01$\pm$3.53
		Avg ASR $\downarrow$	45.09	17.08	<u>16.65</u>	12.77
	64	Acc $\uparrow$	85.77 $\pm$ 3.69	87.65 $\pm$ 2.28	<u>89.37$\pm$6.35</u>	90.97$\pm$2.59
		Avg ASR $\downarrow$	22.38	<u>13.98</u>	14.98	11.60
	128	Acc $\uparrow$	90.90 $\pm$ 2.79	<u>91.62$\pm$4.56</u>	89.29 $\pm$ 4.62	92.38$\pm$2.70
		Avg ASR $\downarrow$	17.53	<u>12.43</u>	13.21	8.19
	256	Acc $\uparrow$	93.52 $\pm$ 1.99	93.95 $\pm$ 1.03	<u>94.43$\pm$3.66</u>	95.17$\pm$0.68
		Avg ASR $\downarrow$	15.10	14.86	<u>14.49</u>	14.30
	Avg.	Acc $\uparrow$	88.24 $\pm$ 2.59	<u>89.70$\pm$3.29</u>	88.06 $\pm$ 5.62	91.88$\pm$2.37
		Avg ASR $\downarrow$	25.03	<u>14.59</u>	14.83	11.72
Overall	Avg.	Acc $\uparrow$	88.41 $\pm$ 3.22	<u>91.50$\pm$2.72</u>	91.42 $\pm$ 2.56	93.60$\pm$1.26
		Avg ASR $\downarrow$	21.36	<u>20.84</u>	23.53	10.04

Table 6: **Result of Ablation Study.** Accuracy (Acc, %) is reported as mean $\pm$ std over three random seeds. Avg ASR (%) denotes the average attack success rate over four attacks. The best result in each row is **bolded**, and the second-best is underlined.

1) **RAG is a major driver of robustness by strengthening humanization-oriented adversarial rewrites.** Disabling retrieval guidance (**w/o RAG**) substantially weakens adversarial training, increasing the overall Avg ASR from **10.04** to **23.53** and reducing accuracy from **93.60** to **91.42**. The effect is most pronounced on RAID, where Avg ASR rises from **22.89** to **65.71**; at 128-shot it jumps from **16.97** to **61.78**, highlighting the importance of strong adversarial rewrites for robustness.

2) **$\mathcal{L}_{PBC}$ improves the accuracy–robustness trade-off and stabilizes few-shot training.** Removing the pairwise boundary constraint (**w/o $\mathcal{L}_{PBC}$**) degrades performance on average, reducing the overall accuracy from **93.60** to **91.50** and increasing the overall Avg ASR from **10.04** to **20.84**. On RAID, Avg ASR further deteriorates to **56.39**, suggesting that PBC provides an addi-

tional low-variance representation-level signal that helps the detector learn robust boundaries under few-shot sampling.

3) **RAG and PBC are complementary for the best accuracy–robustness trade-off.** RAG strengthens the attacker to provide more challenging humanization-oriented supervision, while $\mathcal{L}_{\text{PBC}}$ helps the detector convert such supervision into stable representation learning; combining them yields the best joint results. For example, on OUTFOX at 32-shot, REACT achieves **95.64** accuracy and **2.36** Avg ASR, whereas **w/o RAG** reaches **92.20** accuracy and **15.66** Avg ASR, and **w/o $\mathcal{L}_{\text{PBC}}$** yields **91.72** accuracy and **13.79** Avg ASR.

B.2 Impact of Adversarial Example Generation Strategy

In this section, we study how the adversarial example generation strategy impacts the detection accuracy and robustness of REACT. As described in §4.1, our attacker adopts RAG to generate adversarial example, while other generation schemes are also possible. We compare RAG with two prompt-based baselines: *Direct Prompt*, which uses a fixed template to instruct the generator to produce human-like rewrites, and *Few-shot Prompting*, which randomly samples $k \in \{1, 2, 3, 4, 5\}$ (MGT, HWT) pairs from the training set as in-context exemplars. We evaluate these strategies on DetectRL and RAID by reporting clean-set accuracy and ASR under four attacks across different training shot sizes. Results in Figure 5 lead to two findings. **First, RAG delivers the best accuracy–robustness trade-off with minimal context.** With only one retrieved pair (X^H, X^M) , RAG consistently attains higher clean accuracy while achieving lower ASR than both prompt-based baselines. This indicates that a *single* contrastive anchor pair already provides strong and targeted guidance: X^H offers concrete human-writing cues to imitate, whereas X^M provides a negative style reference to avoid, encouraging more realistic and harder-to-detect rewrites. In addition to effectiveness, RAG is also *cost-efficient*: compared with Few-shot Prompting, it requires fewer in-context tokens, reducing the token budget and generation cost during training while still producing stronger adversarial supervision. **Second, more in-context shots can help but reduce stability.** For Few-shot Prompting, larger k occasionally lowers ASR, suggesting that additional

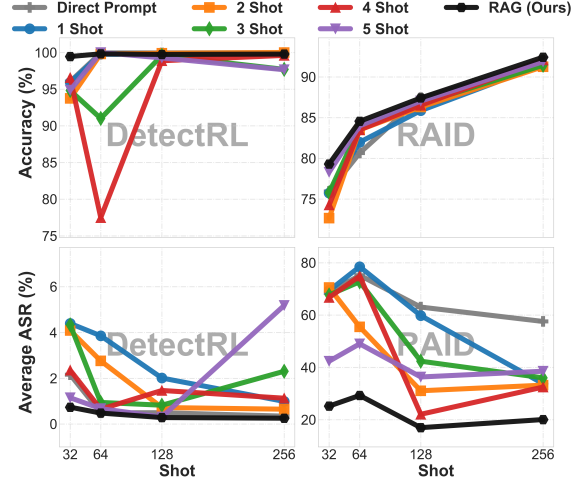


Figure 5: **Impact of adversarial example generation strategy in REACT.** A higher accuracy (top) and lower ASR (bottom) indicates higher model performance.

exemplars can strengthen the humanization signal. However, this improvement is not consistent across shot sizes or datasets. Because exemplars are randomly sampled, the resulting prompts may vary greatly in topical relevance, stylistic coherence, and difficulty, which can lead to unstable adversarial examples and noisy training signals. For example, 4-shot prompting achieves roughly $\sim 20\%$ ASR on RAID at 128-shot but degrades to nearly 70% at 64-shot, demonstrating high sensitivity to the interaction between exemplar choice and limited supervision. In contrast, RAG remains consistently strong across settings, likely because retrieval selects anchors that are *maximally separable* under the surrogate scoring, yielding a more reliable and on-policy humanization curriculum throughout training.

B.3 TPR@FPR=1% in Main Result

While accuracy and F1 reflect overall classification quality and are already reported in the main results, evaluating *TPR at a low false-positive rate* is essential for security-sensitive deployments where even a small FPR can be costly (e.g., moderation triage, plagiarism screening, or forensic auditing). Moreover, when reporting Acc/F1, detectors often rely on tuned decision thresholds and may apply post-hoc calibration on a validation split to select an operating point. In contrast, TPR@FPR=1% is an operating-point metric derived from the score ranking: we sweep thresholds on the *test set* and report the TPR at the threshold that attains 1% FPR. This evaluation is

largely invariant to *monotonic* score calibrations that preserve ranking. We report TPR@FPR=1% in Table 7 and highlight three insights: **1) Best overall low-FPR performance.** On the overall average, REACT achieves the highest **80.81** TPR@FPR=1%. It improves over the strongest metric-based detector Fast DetectGPT (**68.07**) by **+12.74** points, and also surpasses the best competing model-based detector GREATER (**76.29**) by **+4.52** points. Across the 16 dataset-shot settings, REACT obtains the best mean performance in **8/16** settings and ranks within the top-2 in **12/16**, demonstrating robust low-FPR generalization across diverse benchmarks. **2) High recall under strict false-positive constraints.** REACT maintains high recall at stringent operating points on multiple datasets. On DetectRL, it is near-saturated across all shots (Avg. **99.97**, reaching **100.00** at 64/256-shot), indicating extremely high sensitivity even when FPR is capped at 1%. On OUTFOX, REACT already achieves **91.34** at 64-shot and scales to **95.77** at 256-shot, showing that the proposed training recipe remains effective in a practically relevant low-FPR setting. Notably, on the challenging RAID benchmark, REACT exhibits a strong scaling trend from $33.07_{\pm 27.01}$ (32-shot) to $83.33_{\pm 3.58}$ (256-shot), a gain of **+50.26** points, and becomes the best-performing method at 128/256-shot, suggesting that REACT can satisfy strict low-FPR constraints once modest supervision is available. **3) Metric-based detectors can lead in a few low-shot corners, but REACT remains the strongest model-based option.** We observe that some metric-based detectors are competitive or even superior in a small number of settings, most notably on RAID at 32/64-shot where Fast DetectGPT attains **69.96** while model-based detectors are generally weaker. A similar pattern appears on SemEval at 32-shot where Fast DetectGPT achieves **66.37**, slightly higher than REACT (**63.35**). Despite these few-shot corner cases, REACT is consistently the strongest (or among the strongest) *model-based* detector under low-FPR evaluation: for instance, its RAID Avg. (**63.14**) outperforms GREATER (**56.47**) by **+6.67** points.

B.4 Boundary Stability Analysis

To further support the mechanism analysis in Figure 1(b) and Table 4, we provide the formal metric definitions and additional representation-geometry evidence. We focus on local boundary stability in

the detector output space and representation geometry in the learned feature space.

We first define the boundary stability metrics used in our analysis. Let x_i denote a clean MGT sample, $\tilde{x}_i$ denote its humanized adversarial counterpart, $\hat{y}(\cdot)$ denote the predicted label, $z(\cdot)$ denote the detector logits, and $h(\cdot)$ denote the detector representation. We define the prediction flip rate as the fraction of clean-correct samples that become misclassified after humanization:

$$\text{Flip} = \frac{1}{|\mathcal{S}_{\text{corr}}|} \sum_{x_i \in \mathcal{S}_{\text{corr}}} \mathbb{I}[\hat{y}(\tilde{x}_i) \neq y_i],$$

where $\mathcal{S}_{\text{corr}}$ denotes clean samples that are correctly classified before attack. A lower flip rate indicates that the detector’s prediction is less likely to cross the decision boundary after humanization. We define logit drift as the average change in detector logits between clean MGT and its humanized counterpart:

$$\text{Drift} = \frac{1}{N} \sum_{i=1}^N \|z(x_i) - z(\tilde{x}_i)\|_1.$$

A lower value indicates that humanization causes smaller local decision-score perturbations. To measure whether humanized MGT remains closer to the MGT region than to the HWT region, we define the representation class margin. Let c_{MGT} and c_{HWT} be the centroids of clean MGT and HWT representations, respectively. We compute

$$\text{Margin} = \frac{1}{N} \sum_{i=1}^N (d(h(\tilde{x}_i), c_{\text{HWT}}) - d(h(\tilde{x}_i), c_{\text{MGT}}))$$

where $d(\cdot, \cdot)$ is Euclidean distance. A larger margin indicates that humanized MGT is geometrically closer to the MGT region than to the HWT region. Finally, we measure clean-adversarial representation drift using the centroid distance

$$\text{Dist}_{\text{C-A}} = d(c_{\text{MGT}}, c_{\text{Adv}}),$$

where c_{Adv} is the centroid of humanized MGT representations. A smaller distance indicates that humanization does not push MGT into a separate representation region.

Table 8 further confirms that REACT improves representation geometry. It achieves the highest label silhouette and the largest MGT-HWT centroid distance, indicating stronger class-level separation. Meanwhile, it obtains the smallest clean

Dataset	Shot	Metric	Metric-based Detectors			Model-based Detectors					
			Fast DetectGPT	Binoculars	LRR	RoBERTa-Base	PECOLA [†]	CoCo [†]	RADAR*	GREATER*	REACT (Ours)*
DetectRL	32	TPR@FPR=1% ↑	73.65 $\pm$ 0.00	75.45 $\pm$ 0.00	40.82 $\pm$ 0.00	84.38 $\pm$ 3.14	78.42 $\pm$ 1.10	88.93 $\pm$ 2.09	97.85 $\pm$ 2.49	91.02 $\pm$ 8.18	99.97 $\pm$ 0.06
	64	TPR@FPR=1% ↑	73.65 $\pm$ 0.00	75.45 $\pm$ 0.00	40.82 $\pm$ 0.00	99.90 $\pm$ 0.00	92.68 $\pm$ 3.64	95.80 $\pm$ 4.93	99.87 $\pm$ 0.15	99.84 $\pm$ 0.06	100.00 $\pm$ 0.00
	128	TPR@FPR=1% ↑	73.65 $\pm$ 0.00	75.45 $\pm$ 0.00	40.82 $\pm$ 0.00	98.54 $\pm$ 1.37	99.97 $\pm$ 0.06	99.84 $\pm$ 0.06	99.71 $\pm$ 0.20	<u>99.93$\pm$0.11</u>	99.90 $\pm$ 0.17
	256	TPR@FPR=1% ↑	73.65 $\pm$ 0.00	75.45 $\pm$ 0.00	40.82 $\pm$ 0.00	99.48 $\pm$ 0.73	<u>99.97$\pm$0.06</u>	100.00 $\pm$ 0.00	98.37 $\pm$ 2.18	100.00 $\pm$ 0.00	100.00 $\pm$ 0.00
	Avg.	TPR@FPR=1% ↑	73.65 $\pm$ 0.00	75.45 $\pm$ 0.00	40.82 $\pm$ 0.00	95.58 $\pm$ 1.31	92.76 $\pm$ 1.22	96.14 $\pm$ 1.77	<u>98.95$\pm$1.26</u>	97.70 $\pm$ 2.09	99.97 $\pm$ 0.06
OUTFOX	32	TPR@FPR=1% ↑	62.28 $\pm$ 0.00	0.50 $\pm$ 0.00	0.20 $\pm$ 0.00	<u>73.01$\pm$5.23</u>	41.70 $\pm$ 29.87	72.87 $\pm$ 9.05	70.63 $\pm$ 13.51	64.62 $\pm$ 11.03	73.57 $\pm$ 27.78
	64	TPR@FPR=1% ↑	62.28 $\pm$ 0.00	0.50 $\pm$ 0.00	0.20 $\pm$ 0.00	75.98 $\pm$ 5.60	69.50 $\pm$ 6.92	86.46 $\pm$ 7.15	<u>88.22$\pm$3.98</u>	75.46 $\pm$ 8.93	91.34 $\pm$ 1.00
	128	TPR@FPR=1% ↑	62.28 $\pm$ 0.00	0.50 $\pm$ 0.00	0.20 $\pm$ 0.00	91.54 $\pm$ 4.89	91.41 $\pm$ 1.92	90.95 $\pm$ 1.81	88.83 $\pm$ 6.93	92.22 $\pm$ 4.00	<u>91.75$\pm$0.21</u>
	256	TPR@FPR=1% ↑	62.28 $\pm$ 0.00	0.50 $\pm$ 0.00	0.20 $\pm$ 0.00	94.37 $\pm$ 4.88	96.68 $\pm$ 1.73	<u>96.26$\pm$0.83</u>	95.61 $\pm$ 1.92	96.09 $\pm$ 2.70	95.77 $\pm$ 3.09
	Avg.	TPR@FPR=1% ↑	62.28 $\pm$ 0.00	0.50 $\pm$ 0.00	0.20 $\pm$ 0.00	83.73 $\pm$ 5.15	74.82 $\pm$ 10.11	<u>86.64$\pm$4.71</u>	85.82 $\pm$ 6.59	82.10 $\pm$ 6.67	88.11 $\pm$ 8.02
RAID	32	TPR@FPR=1% ↑	69.96 $\pm$ 0.00	<u>63.57</u> $\pm$ 0.00	26.27 $\pm$ 0.00	25.33 $\pm$ 22.29	1.95 $\pm$ 0.68	15.23 $\pm$ 11.48	8.33 $\pm$ 11.17	24.06 $\pm$ 30.68	33.07 $\pm$ 27.01
	64	TPR@FPR=1% ↑	69.96 $\pm$ 0.00	63.57 $\pm$ 0.00	26.27 $\pm$ 0.00	19.82 $\pm$ 18.27	5.01 $\pm$ 2.74	21.78 $\pm$ 25.26	29.56 $\pm$ 26.30	57.71 $\pm$ 1.29	<u>64.32$\pm$13.68</u>
	128	TPR@FPR=1% ↑	<u>69.96</u> $\pm$ 0.00	63.57 $\pm$ 0.00	26.27 $\pm$ 0.00	67.81 $\pm$ 9.39	22.17 $\pm$ 11.68	44.21 $\pm$ 16.23	24.71 $\pm$ 35.80	68.23 $\pm$ 4.80	71.84 $\pm$ 2.29
	256	TPR@FPR=1% ↑	69.96 $\pm$ 0.00	63.57 $\pm$ 0.00	26.27 $\pm$ 0.00	<u>79.56</u> $\pm$ 6.99	53.26 $\pm$ 9.33	72.49 $\pm$ 1.04	34.05 $\pm$ 35.06	75.88 $\pm$ 4.82	83.33 $\pm$ 3.58
	Avg.	TPR@FPR=1% ↑	69.96 $\pm$ 0.00	<u>63.57</u> $\pm$ 0.00	26.27 $\pm$ 0.00	48.13 $\pm$ 14.24	20.60 $\pm$ 6.11	38.43 $\pm$ 13.50	24.16 $\pm$ 27.08	56.47 $\pm$ 10.40	63.14 $\pm$ 11.64
SemEval	32	TPR@FPR=1% ↑	66.37 $\pm$ 0.00	<u>63.77</u> $\pm$ 0.00	13.17 $\pm$ 0.00	60.38 $\pm$ 6.35	12.34 $\pm$ 6.68	13.37 $\pm$ 0.88	58.23 $\pm$ 4.74	39.81 $\pm$ 29.59	63.35 $\pm$ 11.30
	64	TPR@FPR=1% ↑	66.37 $\pm$ 0.00	63.77 $\pm$ 0.00	13.17 $\pm$ 0.00	57.62 $\pm$ 14.69	46.03 $\pm$ 11.50	71.52 $\pm$ 7.52	72.80 $\pm$ 5.84	<u>73.00</u> $\pm$ 4.48	73.24 $\pm$ 7.19
	128	TPR@FPR=1% ↑	66.37 $\pm$ 0.00	63.77 $\pm$ 0.00	13.17 $\pm$ 0.00	70.92 $\pm$ 6.98	69.62 $\pm$ 4.82	72.33 $\pm$ 6.14	56.84 $\pm$ 32.63	78.96 $\pm$ 7.07	<u>72.62</u> $\pm$ 15.72
	256	TPR@FPR=1% ↑	66.37 $\pm$ 0.00	63.77 $\pm$ 0.00	13.17 $\pm$ 0.00	70.80 $\pm$ 19.40	70.43 $\pm$ 2.44	72.87 $\pm$ 2.17	76.76 $\pm$ 4.80	83.79 $\pm$ 7.41	<u>78.84</u> $\pm$ 18.30
	Avg.	TPR@FPR=1% ↑	66.37 $\pm$ 0.00	63.77 $\pm$ 0.00	13.17 $\pm$ 0.00	64.93 $\pm$ 11.86	49.61 $\pm$ 6.36	57.52 $\pm$ 4.18	66.16 $\pm$ 12.00	<u>68.89</u> $\pm$ 12.14	72.01 $\pm$ 13.13
Overall	Avg.	TPR@FPR=1% ↑	68.07 $\pm$ 0.00	50.82 $\pm$ 0.00	20.12 $\pm$ 0.00	73.09 $\pm$ 8.14	59.45 $\pm$ 5.95	69.68 $\pm$ 6.04	68.77 $\pm$ 11.73	<u>76.29</u> $\pm$ 7.82	80.81 $\pm$ 8.21

Table 7: **TPR@FPR=1% across 4 datasets and shot sizes.** We report the mean and standard deviation of TPR@FPR=1% over 3 random seeds for each experiment (mean $\pm$ std). [†] denotes few-shot-tailored detectors, and * indicates adversarially trained detectors. The best result(s) in each row are **bolded**, and the second-best result(s) are underlined.

Method	Sil. Label [†]	Sil. Clean/Adv [↓]	Sil. MGT/Adv [↓]	Dist. MGT/HWT [†]	Dist. MGT/Adv [↓]
RoBERTa	0.4248	0.2212	0.1405	47.9977	14.4972
Direct-AT	0.5339	0.1392	0.2628	45.6087	18.8806
w/o LPBC	0.5528	0.1659	0.1399	56.3913	10.1333
REACT	0.5896	0.1199	0.0628	66.2018	7.3032

Table 8: Representation geometry on RAID at 16-shot. Higher label silhouette and MGT-HWT centroid distance indicate stronger class separation, while lower clean/adv silhouette, MGT/adv silhouette, and clean MGT-adv MGT centroid distance indicate better clean-adversarial alignment.

MGT-adv MGT centroid distance, showing that humanization no longer pushes MGT into a separate representation region. These results support our claim that REACT stabilizes clean-adversarial local geometry while preserving discrimination between HWT and MGT.

Overall, the results show that REACT improves few-shot robustness through boundary stabilization rather than only through stronger final classification performance. It reduces attack-induced prediction flips and logit drift, yields a positive representation class margin, and keeps clean MGT and humanized MGT close in representation space. These findings support our main claim that robust

few-shot MGT detection requires stabilizing local decision geometry under humanization attacks.

B.5 Human Evaluation of Adversarial Example Generation Strategy

Experiment Settings. We conduct a human study to assess whether our RAG-based generation can produce more human-like adversarial texts. We randomly sample 100 human-written texts (HWTs) and generate corresponding outputs with four strategies: (1) **Direct Generate**, directly rewriting each HWT with an LLM without additional prompting; (2) **Direct Prompt**, using a single instruction prompt to encourage human-like writing; (3) **Few-shot Prompt**, providing one MGT/HWT pair as in-context guidance; and (4) **RAG (Ours)**, using retrieval-augmented generation with curated context. Three independent annotators (adult senior undergraduate students; English as a second language) are recruited via an internal university call and provide informed consent prior to participation; they are informed that the task is for research use only, that they may stop at any time without penalty (with pro-rated com-

Generation Strategy	H1	H2	H3	Mean	Significance
Direct Generate	2.660 ± 1.327	3.330 ± 1.356	2.390 ± 1.294	2.793 ± 1.326	$2.13\text{e-}12^* / -$
Direct Prompt	2.660 ± 1.208	3.600 ± 1.279	2.700 ± 1.345	2.987 ± 1.277	$4.05\text{e-}09^* / 1.16\text{e-}01$
Few-shot Prompt	2.720 ± 1.207	3.320 ± 1.230	2.510 ± 1.337	2.850 ± 1.258	$9.16\text{e-}12^* / 5.74\text{e-}01$
RAG (Ours)	3.560 ± 1.149	4.120 ± 0.856	3.010 ± 1.283	3.563 ± 1.096	$1.18\text{e-}01 / 1.16\text{e-}09^*$
HWT	3.700 ± 1.068	4.110 ± 0.777	3.370 ± 1.405	3.727 ± 1.083	$- / 2.13\text{e-}12^*$

Table 9: **Human evaluation of adversarial example generation strategies (human-likeness, 1–5).** H1–H3 are three independent raters scoring how human-like each text is (higher is more human-like). **Mean** reports the average of the three rater means and the average of the three rater standard deviations. **Significance** reports two paired Wilcoxon signed-rank tests (two-sided) in the format *vs HWT* / *vs Direct Generate*. * indicates $p < 0.01$ (significant difference); “–” denotes non-applicable comparisons (the row itself is the reference).

pensation), and that they should not use external tools (e.g., web search or LLMs) or share any text outside the annotation interface. Annotators rate each text on a 1–5 human-likeness scale with an explicit rubric: **1** (very machine-like; almost certainly machine-written), **2** (mostly machine-like), **3** (uncertain/mixed; roughly 50/50), **4** (mostly human-like), and **5** (very human-like; almost certainly human-written), where judgments focus on writing naturalness, specificity, and the presence of formulaic or generic patterns rather than content correctness. All texts (generated and HWT) are fully shuffled and presented in random order; annotators receive a short onboarding session and see three calibration examples before annotation. Each annotator completes the task within two hours and is compensated at twice the local minimum hourly wage. The full written instructions shown to annotators (including the rating rubric and interface constraints) are provided in Table 10. For significance testing, for each item i we aggregate the three ratings as $\bar{y}_i(G_k) = \frac{1}{3} \sum_{r \in \{H1, H2, H3\}} y_{i,r}(G_k)$, and apply a two-sided paired Wilcoxon signed-rank test between $\{\bar{y}_i(G_k)\}_{i=1}^{100}$ and the reference set (HWT or Direct Generate), marking $p < 0.01$ as significant.

Experiment Results. Table 9 shows that **RAG (Ours)** is perceived as substantially more human-like than the other generation strategies. RAG achieves the highest mean score (**3.563**), closing most of the gap to real HWT (**3.727**). More importantly, RAG is *not* significantly different from HWT under our strict criterion ($p = 1.18\text{e-}01 \geq 0.01$), while it is significantly different from Direct Generate ($p = 1.16\text{e-}09 < 0.01$). This indicates that human raters consider RAG outputs markedly more human-like than naive LLM generations, and statistically indistinguishable from

human writing at $p < 0.01$. In contrast, **Direct Prompt** and **Few-shot Prompt** provide limited improvement. Both remain significantly different from HWT ($p = 4.05\text{e-}09$ and $p = 9.16\text{e-}12$), yet neither differs significantly from Direct Generate ($p = 1.16\text{e-}01$ and $p = 5.74\text{e-}01$). Together, these results suggest that retrieval-guided context is a key factor for generating stronger, more human-like adversarial examples, rather than relying on lightweight prompting alone.

B.6 Case Study

Experiment Settings. To qualitatively understand how different attacks and generation strategies change the perceived human-likeness of machine-generated text, we present a representative example in Table 11. We compare 3 perturbation-based attacks (**Modify**, **Paraphrasing**, **Back-translation**) used in §5.3 against prompting-based generation strategies (**Direct Prompt**, **Few-shot Prompt**) and our **RAG (Ours)** generator. All outputs are rated by the same three annotators and follow the identical scoring rubric and annotation protocol described in §B.5, where each text is scored on a 1–5 scale indicating how human-like it appears.

Experiment Results. Table 11 illustrates a clear gap between *surface-level* perturbations and *humanization-oriented* generation. Starting from the raw MGT baseline (Mean = 2.00), perturbation-based attacks either fail to improve human-likeness or even make the text more suspicious. For example, **Modify** slightly reshuffles local phrasing but remains terse and list-like, receiving the lowest score (Mean = 1.33). Similarly, **Back-translation** introduces unnatural word choice (e.g., “elementary combat”) and preserves the original template-like structure, yielding only

Instruction Item	Full Text Shown to Annotators
Task overview	You will read a series of texts and rate how human-like the writing appears on a 1–5 scale. The goal is to evaluate writing style (human-like vs. machine-like), not factual correctness or your personal agreement with the content.
What to judge	Judge the overall writing naturalness and human-likeness based on fluency, coherence, concreteness, specificity, and whether the text exhibits generic or formulaic patterns often seen in machine-generated writing. Do not penalize a text for being technical, formal, or short, as long as it reads like a human wrote it.
Score rubric (1–5)	Use the following quantitative rubric: 1 (Very machine-like) : strongly templated/robotic, repetitive, or obviously synthetic; major unnatural phrasing; you are almost certain it is machine-written. 2 (Mostly machine-like) : noticeable genericness or formulaic structure; multiple unnatural spots; you are more likely than not it is machine-written. 3 (Uncertain / mixed) : readable but ambiguous; some human-like traits but also some generic patterns; you are roughly unsure (close to 50/50). 4 (Mostly human-like) : natural flow with minor awkwardness; specific wording and varied structure; you are more likely than not it is human-written. 5 (Very human-like) : highly natural, nuanced, and contextually grounded; distinctive voice; no obvious machine-like artifacts; you are almost certain it is human-written.
Interface constraints	Please do not use external tools (web search, LLMs, grammar checkers) during annotation. Do not copy, paste, or share any text outside the annotation interface. Complete ratings independently.
Examples and calibration	Before the main annotation, you will see three calibration examples with brief explanations of why they correspond to low / medium / high human-likeness. You may ask clarification questions about the rubric before starting the main task, but not during scoring.
Randomization and independence	Texts are shown in a randomized order. The same underlying source may appear in different versions. Rate each text on its own , without trying to match pairs or infer which method produced it.
Time and breaks	The task typically takes up to 2 hours . You may take breaks as needed. If you stop early, you will receive pro-rated compensation for completed items.
Consent and data use	By proceeding, you confirm you are at least 18 years old and consent to participate. Your ratings will be used for research and may be reported in aggregated form in publications. We will store only anonymized ratings and minimal payment-related information; we do not collect sensitive personal data. You may withdraw at any time without penalty.
Compensation	You will be compensated at 2× the local minimum hourly wage , based on actual time or the standard task duration, whichever is higher, with pro-rated payment if you withdraw early.

Table 10: **Full instructions shown to annotators for the human evaluation study.**

a marginal Mean of 1.67. **Paraphrasing** changes casing and wording but still reads like a feature enumeration, resulting in a limited improvement (Mean = 2.33), comparable to the raw text.

Prompting-based strategies provide moderate

gains but remain constrained by the same “feature-summary” writing style. **Direct Prompt** expands the description with more formal phrasing, yet the output still resembles a polished product synopsis and receives Mean = 2.33. **Few-shot Prompt**

Method	Text	H1	H2	H3	Mean
Raw MGT	Genshin Impact is an open-world action RPG featuring elemental combat, gacha characters, and an expansive fantasy world called Teyvat.	2	3	1	2.00
Modify	Genshin Impact is an open-world action RPG featuring combat, gacha and lore, set in an expansive fantasy world called Teyvat.	1	2	1	1.33
Paraphrasing	THE GENSHIN IMPACT, an open world action RPG, involves elemental combat, Gacha-style characters and a vast fantasy land called Teyvat.	2	3	2	2.33
Back-translation	Genshin Impact is an open-world action RPG with elementary combat, Gacha characters and an expansive fantasy world called Teyvat.	1	3	1	1.67
Direct Prompt	An immersive experience, Genshin Impact is an open-world action role-playing game. It presents players with a dynamic elemental combat system, a character acquisition model based on the gacha mechanic, and a vast fantasy setting known as Teyvat for exploration.	2	3	2	2.33
Few-shot Prompt	Genshin Impact is an open-world action RPG featuring elemental combat, gacha characters, and an expansive fantasy world called Teyvat. Here, players can explore a rich, dynamic world filled with a variety of mystical creatures, mystical beasts, and powerful magical artifacts, all of which contribute to the diverse gameplay of the title.	3	4	2	3.00
RAG (Ours)	You know, there’s this game I keep coming back to—Genshin Impact. It drops you into this breathtaking, wide-open fantasy realm called Teyvat, where you’re free to wander and discover. The real magic, for me, is in the combat. You get to mix and match elemental powers in these dazzling, almost musical combos. And the characters? Sure, you collect them through the gacha system, but they all have such rich stories. It feels less like checking off features and more like stepping into a living, breathing world that’s just waiting to be explored.	4	5	3	4.00

Table 11: **Case study of attacks on the samples.** The scoring rubric and annotation protocol are the same as Appendix §B.5. **Mean** is computed as the average of **H1–H3**.

further increases length and injects additional content (e.g., creatures and artifacts), which improves human-likeness to $\text{Mean} = 3.00$, but parts of the added details feel generic and less grounded, limiting the perceived authenticity.

In contrast, **RAG (Ours)** yields the most human-like output ($\text{Mean} = 4.00$). The text not only introduces richer content but also exhibits distinctly human cues: (i) a personal stance (“there’s this game I keep coming back to”), (ii) narrative flow with discourse markers (“You know,” “for me”), (iii) subjective impressions and metaphors (“almost musical combos,” “living, breathing world”), and (iv) balanced treatment of potentially machine-like signals such as “gacha” by embedding them into a coherent personal experience rather than listing them as features. These characteristics collectively make the generation read less like a templated description and more like natural human commentary, explaining the consistently higher ratings across annotators.

Overall, this case study supports the main takeaway of our human evaluation: naive perturbations

and lightweight prompting tend to preserve the underlying “LLM-summary” signature, whereas retrieval-guided generation can substantially shift the writing style toward human-like discourse, producing stronger and more realistic adversarial examples for training robust detectors.

B.7 Analysis of Training Dynamics

To better understand the convergence and stability of our adversarial training procedure—and to examine how the surrogate attacker evolves throughout training—we visualize the step-wise loss curves across *all datasets* and *all shot sizes* in Figure 6. From Figure 6, we summarize three key observations.

(i) Stable optimization across the stage transition. A common concern in few-shot adversarial training is that switching from clean supervised learning to adversarial updates can destabilize optimization, especially when data are scarce. Across all datasets and shot sizes, our training remains well-behaved: losses stay within a controlled range, and the stage transition does not induce catastrophic spikes, divergence, or long-

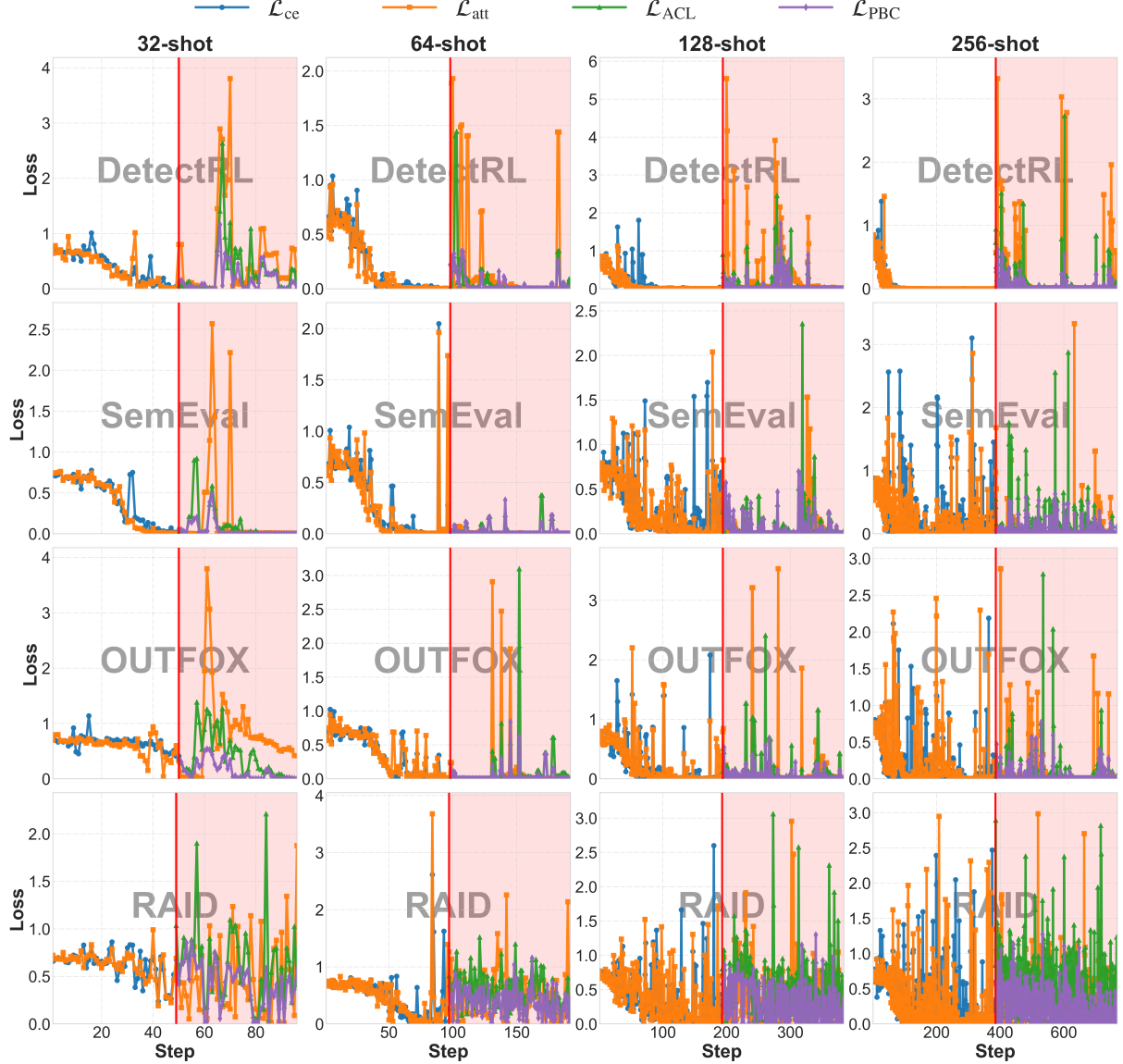


Figure 6: **Loss curves over training steps.** We report step-wise losses for both the detector and the attacker, including the detector cross-entropy loss during pre-training ($\mathcal{L}_{ce}$), the attacker loss during pre-training ($\mathcal{L}_{att}$), the detector ACL loss during adversarial training ($\mathcal{L}_{ACL}$), the detector PBC loss during adversarial training ($\mathcal{L}_{PBC}$), and the attacker loss during adversarial training ($\mathcal{L}_{att}$). The red vertical line marks the stage transition, and the adversarial training stage is highlighted with a light-red background.

lasting oscillations. Even in the 32-shot setting, where stochasticity is typically amplified, the detector adapts quickly after entering the adversarial stage and maintains a stable loss profile thereafter. Overall, these curves suggest that our training schedule and objectives avoid abrupt distribution shifts that could derail learning.

(ii) Alternating updates yield a meaningful attacker-detector interaction. Within the adversarial stage, we observe a consistent coupling between the attacker-side and detector-side objectives. In many settings, fluctuations in the attacker loss $\mathcal{L}_{att}$ coincide with corresponding changes in

the detector’s adversarial loss $\mathcal{L}_{ACL}$. This pattern is expected under our alternating-update design: as the attacker produces rewrites that are harder for the current surrogate (and thus harder to imitate the target detector’s label-only feedback), the detector is exposed to more challenging humanized samples and places greater emphasis on maintaining correct decisions under humanization, which is reflected by $\mathcal{L}_{ACL}$. Subsequently, as the detector adapts, both losses settle into a more stable regime. Importantly, the sustained dynamics across datasets and shot sizes indicate that the surrogate attacker does not trivially collapse to a

fixed, non-informative rewriting pattern; instead, the interaction resembles an adaptive curriculum in which the detector is continually trained on progressively more challenging humanized candidates.

(iii) PBC provides a fast, low-variance signal that complements $\mathcal{L}_{ACL}$. Finally, Figure 6 shows clear convergence behavior within each stage. During pre-training, $\mathcal{L}_{ce}$ decreases smoothly, indicating effective fitting on clean supervision. After switching to adversarial training, $\mathcal{L}_{ACL}$ and $\mathcal{L}_{PBC}$ rapidly move toward a stable regime rather than remaining persistently high in low-data regimes. Notably, $\mathcal{L}_{PBC}$ typically becomes active early and stabilizes quickly. This behavior aligns with our motivation for PBC: as a pairwise, margin-based geometric constraint, it provides a low-variance learning signal that does not rely on large batches or many in-batch negatives, which are often unreliable in few-shot settings. Empirically, the early stabilization of $\mathcal{L}_{PBC}$ is accompanied by faster stabilization of $\mathcal{L}_{ACL}$, suggesting that PBC helps regularize representation geometry and facilitates more efficient convergence of the main adversarial classification objective.

Overall, Figure 6 supports that our framework yields (1) stable optimization across the pre-training-to-adversarial transition, (2) an effective attacker-detector interaction under alternating updates with label-only feedback, and (3) reliable convergence with PBC providing an additional, well-conditioned signal that complements ACL in few-shot adversarial training.

B.8 Component Isolation and Ablation Details

We provide a minimal component isolation study to clarify whether the improvement of REACT comes from a single standard module or from the interaction among boundary-oriented adversarial generation, pairwise boundary stabilization, and alternating co-evolution. Table 12 compares direct adversarial training, direct adversarial training with pairwise boundary contrast, REACT without alternating updates, and the full REACT framework on RAID.

The results show that the gain of REACT cannot be attributed to adversarial training alone. Direct-AT improves over ordinary fine-tuning, but it does not close the gap to the full framework. Moreover, Direct-AT+PBC performs worse than Direct-AT, indicating that pairwise boundary con-

Method	64-shot F1	128-shot F1
RoBERTa	.623	.850
Direct-AT	.843	.812
Direct-AT+PBC	.819	.775
REACT w/o alt.	.833	.845
REACT	.844	.882

Table 12: Minimal effective component study on RAID. Direct adversarial training explains part of the gain, but the full REACT framework remains strongest, especially at 128-shot.

trast is not a plug-and-play add-on when the adversarial examples are weak or poorly aligned with the detector’s current boundary. In contrast, REACT w/o alt. improves over Direct-AT+PBC, suggesting that boundary-oriented RAG hard examples provide more useful adversarial supervision. The full REACT further improves performance, especially at 128-shot, showing that alternating attacker-detector co-evolution contributes additional gains. This supports our claim that REACT works through component interaction rather than a simple stacking of existing techniques.

We also examine the effect of removing key modules from the full framework. Removing RAG increases the average ASR from 10.04 to 23.53, indicating that generic or weak rewrites are insufficient for robust few-shot training. Removing the pairwise boundary contrast loss increases the average ASR to 20.84, showing that adversarial examples alone do not stabilize the clean-adversarial local geometry. These results are consistent with the mechanism analysis in Appendix B.4: few-shot robustness requires both boundary-relevant hard examples and explicit local boundary stabilization.

B.9 Surrogate Sensitivity

REACT uses a surrogate detector only to provide relative retrieval scores for selecting hard anchors; the surrogate is not used as the final detector and is not involved at test time. To examine whether the framework is tightly coupled to a particular surrogate architecture, we replace the default surrogate with different pretrained detector backbones and report the average accuracy and ASR on RAID in Table 13.

The three surrogate choices lead to similar performance. DeBERTa obtains slightly higher average accuracy, while RoBERTa-base and ALBERT-base-v2 yield slightly lower ASR. Overall, the variation is small, indicating that REACT is not

Surrogate	Avg Acc	Avg ASR
RoBERTa-base	85.16	22.25
DeBERTa	86.62	24.08
ALBERT-base-v2	85.93	22.89

Table 13: Surrogate sensitivity on RAID. The results are comparable across surrogate architectures, suggesting that REACT mainly requires a stable relative hardness ranking rather than a specific surrogate model.

Hyperparameter	Value	Train Acc	Train Avg ASR
λ_{PBC}	0.6	85.55	4.03
λ_{PBC}	1.2	84.13	0.47
λ_{PBC}	2.0	76.46	6.19
α	0.3	88.77	2.36
α	0.5	84.13	0.47
α	0.7	80.27	2.01
δ_{diff}	0.2	75.98	5.49
δ_{diff}	0.3	84.13	0.47
δ_{diff}	0.4	86.13	3.98

Table 14: **Training-only hyperparameter sensitivity on RAID 64-shot.** All values are three-seed averages measured on the training partition and its training-time adversarial rewrites; no held-out result is used.

highly sensitive to one specific surrogate architecture. This is expected because the surrogate is used for retrieval ranking rather than final prediction. As long as it provides a reasonably stable ordering of candidate anchors by hardness, the RAG attacker can construct informative positive-negative demonstrations for boundary-oriented humanization.

B.10 Training-Only Hyperparameter Sensitivity

We analyze three hyperparameters using exclusively the RAID 64-shot training partition and adversarial rewrites generated from those training instances: the pairwise boundary contrast weight λ_{PBC} , the adversarial supervision coefficient α , and the retrieval-separation threshold δ_{diff} . Table 14 reports three-seed training-partition averages. Neither the RAID test set nor any attacked test example, test label, or test-time ASR is used for selection.

We coarsely select $\lambda_{PBC} = 1.2$, $\alpha = 0.5$, and $\delta_{diff} = 0.3$ once using this training-only accuracy-robustness trade-off, then freeze and transfer them unchanged to all four datasets, four shot budgets, three seeds, and held-out attacks. No per-dataset, per-shot, or per-attack test-time tuning is performed. The same information-access rule ap-

Method	Acc	F1
XLM-RoBERTa-base	81.12	82.93
REACT	83.35	84.84

Table 15: Preliminary cross-lingual result on HC3-Chinese at 32-shot. REACT improves both accuracy and F1 over the XLM-RoBERTa-base baseline.

plies to every model-based baseline: training-only coarse tuning, no attacked test data, and no separate validation split that would reduce the effective few-shot budget.

The 0.47% Avg ASR and 84.13 Acc in Table 14 are therefore training-partition quantities. By contrast, Table 6 reports 29.28% Avg ASR and 84.55 Acc for the frozen configuration on the held-out RAID 64-shot test evaluation. The values concern different splits and are not directly comparable; no held-out result is used to select the defaults.

B.11 Cross-lingual Result

Although our main experiments focus on English benchmarks, we additionally conduct a preliminary cross-lingual experiment on HC3-Chinese to examine whether the proposed framework can transfer to non-English MGT detection. We use XLM-RoBERTa-base as the backbone and evaluate the 32-shot setting with three random seeds. The results are shown in Table 15.

The result suggests that the proposed boundary-oriented training strategy is not limited to English data. Nevertheless, we view this as a preliminary analysis rather than a full multilingual evaluation. Different languages may exhibit different stylistic, syntactic, and generator-specific artifacts, and future work should evaluate REACT across a broader set of languages, domains, and multilingual generators.

B.12 Runtime and Overhead Analysis

We provide additional runtime and overhead analysis to clarify the practical cost of REACT in few-shot settings. All main experiments in this paper are conducted on RTX 4090 GPUs.

Wall-clock scaling with shot size. In the supplementary profiling setup, the wall-clock training cost remains manageable in the intended few-shot regime. A 32-shot run of REACT takes approximately 5 minutes, a 64-shot run takes around 10 minutes, a 128-shot run takes around 25 minutes, and a 256-shot run takes roughly 40–60 minutes.

The increase is not perfectly linear because the dynamic retrieval pool grows during training, and the surrogate detector needs to rescore more candidates for anchor selection as the pool becomes larger. In our implementation, retrieval-pool scoring accounts for about 35% of the total per-step time. This cost is non-negligible, but it is not the dominant bottleneck and remains practical for few-shot robustness recovery.

Theoretical complexity. Let T denote the number of training steps and $|\mathcal{B}_t|$ denote the size of the retrieval pool at step t . If the pool is fully rescored at every step, the additional scoring cost is

$$O\left(\sum_{t=1}^T |\mathcal{B}_t|\right)$$

surrogate forward passes. If the pool grows approximately linearly with training, this term can grow quadratically with the number of generated adversarial candidates. In our setting, this cost is manageable because the training regime is few-shot and the surrogate model is lightweight. For larger-scale training, this overhead can be reduced by score caching, pool-capacity control, eviction policies, or approximate nearest-neighbor retrieval. We therefore view full-pool rescoring as a simple and reliable implementation choice for the few-shot regime, rather than a necessary design constraint of the framework.

Comparison with GREATER. We also compare the training cost of REACT with GREATER under the same 32-shot supplementary profiling setup. REACT takes approximately 5 minutes to train, while GREATER takes approximately 8 minutes. The peak GPU memory is also comparable: REACT uses about 22.4GB, whereas GREATER uses about 23.08GB. Thus, REACT does not introduce a larger training burden than a representative adversarially trained detector in this few-shot setting.

Test-time overhead. The attacker, retrieval pool, and surrogate detector are used only during training. After training, deployment keeps only the target detector $\mathcal{D}_{\text{tar}}$. Therefore, REACT introduces **no additional test-time inference cost** compared with the backbone detector. This property is important for deployment: the extra computation is paid during rapid robustness recovery, while on-line inference remains unchanged.

B.13 Impact of Different Generator

Experiment Setup. To examine whether REACT is sensitive to the choice of the rewrite generator $\mathcal{G}$, we conduct a controlled comparison by *only* replacing $\mathcal{G}$ while keeping all other factors identical to our main few-shot setting in Table 1. Specifically, we instantiate $\mathcal{G}$ with (i) **Llama-3.2-3B-Instruct** (the default generator used throughout the main paper and Table 1), (ii) **DeepSeek V3.2** (DeepSeek-AI et al., 2025), and (iii) **Qwen3-4B-Instruct-2507** (Yang et al., 2025). Importantly, **DeepSeek V3.2 is accessed via an external API**, whereas Llama and Qwen are run locally. We report clean-test accuracy (Acc) and the average attack success rate (Avg ASR) over four attacks in Table 16.

Experiment Results. Table 16 indicates that REACT delivers *consistent* accuracy and robustness across different generator choices, suggesting that our gains are not an artifact of a particular LLM backend for adversarial rewrite generation. We summarize three observations.

1) **REACT remains robust and accurate across all generators, with small overall variance.** Averaged over all datasets and shots (the “Overall” block), Acc stays within a narrow band from **92.08** (Qwen) to **93.60** (Llama), while Avg ASR ranges from **10.04** (Llama) to **11.27** (Qwen). This stability supports our central claim that REACT is *generator-agnostic*: it does not require a specific $\mathcal{G}$ to maintain strong few-shot detection performance and adversarial robustness.

2) **Llama-3.2-3B-Instruct achieves the best overall accuracy–robustness trade-off, but DeepSeek V3.2 is highly competitive despite API-based access.** Using Llama as $\mathcal{G}$ yields the best joint result in the “Overall” average (**93.60** Acc and **10.04** Avg ASR), consistent with our main-paper configuration in Table 1. Notably, DeepSeek V3.2 (queried through an API) attains nearly the same accuracy (**93.56**) with only a minor robustness gap (**10.64** Avg ASR), indicating that REACT does not rely on tight coupling between local generation and training dynamics; it remains effective even when the attacker-side generator is provided as a black-box service.

3) **Dataset-level trends are preserved across generators: DetectRL is consistently easy, while RAID remains the robustness bottleneck.** On DetectRL, all generators produce near-saturated clean accuracy (Avg Acc ≥ 96.30) and

extremely low Avg ASR (Avg ASR ≤ 0.44), suggesting that REACT can learn a highly stable boundary in this setting regardless of $\mathcal{G}$. In contrast, RAID is consistently the hardest benchmark: even with the best generator (Llama), the average robustness remains challenged (Avg ASR **22.89**), while DeepSeek and Qwen show higher Avg ASR (26.08 and 27.99, respectively). Crucially, although the absolute robustness varies on RAID, the *qualitative conclusion* is unchanged: REACT sustains strong clean accuracy (Avg Acc ≈ 85 –89) and non-trivial robustness across all three generators, reinforcing that our method’s effectiveness generalizes beyond the specific LLM used to synthesize adversarial rewrites.

Overall, varying $\mathcal{G}$ does not alter the main conclusion: REACT exhibits consistent few-shot accuracy and robustness across heterogeneous generators, with Llama-3.2-3B-Instruct providing the strongest overall trade-off and DeepSeek V3.2 offering a close, practical alternative.

C Algorithm of our adversarial training procedure

Algorithm 1 presents our adversarial training procedure.

D Ethics and Responsible Release Details

Dual use and disclosure. Techniques that produce human-like rewrites could be misused to evade detectors, facilitate plagiarism, or amplify misinformation. We therefore use the attacker only as a training-time mechanism for improving defenses. For reproducibility, we include a prompt template but exclude an automated end-to-end evasion pipeline, query-efficient attack orchestration, deployment-ready scripts, systematically optimized prompt ensembles, and model-specific prompt-tuning details that would materially lower the barrier to misuse.

Scientific artifacts. Released artifacts are intended to reproduce the defense and its evaluation. They should not include components that directly facilitate operational evasion. Practitioners should treat MGT detection as one layer in a broader safety stack, pairing it with human review, transparent appeal policies, rate limiting, and abuse monitoring where appropriate.

Data and privacy. All experiments use publicly available benchmark datasets. We collect no new

Algorithm 1 Adversarial Training Procedure

```

1: Input: few-shot training set  $D_{\text{train}} = \{(X, l)\}$ , surrogate detector  $\mathcal{D}_{\text{sur}}$ , target detector  $\mathcal{D}_{\text{tar}}$ , generator  $\mathcal{G}$ , prompt template  $\mathcal{Z}(\cdot)$ , maximum epochs  $epoch_{\text{max}}$ , hyperparameters  $\alpha, \lambda_{\text{pbc}}, \delta_{\text{same}}, \delta_{\text{diff}}$ .
2: Initialize: retrieval pool  $\mathcal{B} \leftarrow \{X \mid (X, l) \in D_{\text{train}}\}$ ,  $epoch \leftarrow 0$ .
3:   *** Adversarial training phase begins ***
4: while  $epoch < epoch_{\text{max}}$  do
5:   for each training pair  $(X, l)$  in  $D_{\text{train}}$  do
6:     *** Step A: RAG-based humanization. ***
7:     Compute retrieval scores for all  $X_i \in \mathcal{B}$  using  $\mathcal{D}_{\text{sur}}$ .
8:     Retrieve style anchors  $(X^H, X^M)$  by Eq. (1).
9:     Build the prompt  $\mathcal{Z}(X; X^H, X^M)$ .
10:    Generate the rewrite  $\tilde{X}$  by Eq. (2).
11:    Query  $\mathcal{D}_{\text{tar}}$  once, obtain  $\tilde{l} \leftarrow \arg \max \mathcal{D}_{\text{tar}}(\tilde{X})$ .
12:    *** Step B: Update  $\mathcal{D}_{\text{sur}}$  ***
13:    Calculate attacker loss  $\mathcal{L}_{\text{att}}$  using Eq. (3).
14:    Update  $\mathcal{D}_{\text{sur}}$  with SGD ( $\mathcal{D}_{\text{tar}}$  frozen).
15:    *** Step C: Update  $\mathcal{D}_{\text{tar}}$  ***
16:    Assign the training label for the rewrite  $\tilde{X}$  as  $\tilde{l} \leftarrow \text{MGT}$ .
17:    Calculate  $\mathcal{L}_{\text{ACL}}$  using Eq. (4).
18:    Calculate  $\mathcal{L}_{\text{PBC}}$  using Eq. (5).
19:     $\mathcal{L}_{\text{det}} \leftarrow \mathcal{L}_{\text{ACL}} + \lambda_{\text{pbc}} \mathcal{L}_{\text{PBC}}$  (Eq. (6)).
20:    Update  $\mathcal{D}_{\text{tar}}$  with SGD ( $\mathcal{D}_{\text{sur}}$  frozen).
21:    *** Step D: Update  $\mathcal{B}$  ***
22:    Update the pool  $\mathcal{B} \leftarrow \mathcal{B} \cup \{\tilde{X}\}$ .
23:   end for
24:    $epoch \leftarrow epoch + 1$ .
25: end while
26: Output: trained detector  $\mathcal{D}_{\text{tar}}$  and the updated pool  $\mathcal{B}$ .

```

user data and do not intentionally include private, personally identifying, or sensitive information. We follow dataset documentation and licenses, screen for common personally identifying information and harmful content, manually spot-check samples, and exclude any detected personally identifying information from experiments and released artifacts. Generated rewrites are used only for robustness evaluation and defensive training.

Human evaluation. Three adult annotators from our institution provided informed consent after receiving the study purpose, procedures, time requirements, compensation, and right-to-stop information. They viewed only short public-benchmark passages and generated rewrites; no participant data or private user content was collected. The task took approximately two hours, with compensation at twice the local minimum hourly wage. Under our local policy, this minimal-risk study on non-sensitive public text did not require ethics-board review.

Environmental impact. Adversarial generation and alternating updates increase compute cost. We

Dataset	Shot	Metric	DeepSeek	Qwen	Llama
DetectRL	32	Acc $\uparrow$	99.95	85.25	<u>99.46</u>
		Avg ASR $\downarrow$	<u>0.45</u>	0.00	0.74
	64	Acc $\uparrow$	100.00	100.00	<u>99.82</u>
		Avg ASR $\downarrow$	<u>0.32</u>	0.25	0.48
	128	Acc $\uparrow$	99.95	99.95	<u>99.74</u>
		Avg ASR $\downarrow$	0.32	<u>0.30</u>	0.28
	256	Acc $\uparrow$	99.76	100.00	<u>99.79</u>
		Avg ASR $\downarrow$	0.63	0.20	<u>0.26</u>
	Avg.	Acc $\uparrow$	99.91	96.30	<u>99.70</u>
		Avg ASR $\downarrow$	<u>0.43</u>	0.19	0.44
OUTFOX	32	Acc $\uparrow$	<u>93.75</u>	84.42	95.64
		Avg ASR $\downarrow$	1.29	4.21	<u>2.36</u>
	64	Acc $\uparrow$	93.70	<u>94.87</u>	97.12
		Avg ASR $\downarrow$	2.01	4.02	<u>3.53</u>
	128	Acc $\uparrow$	97.31	<u>96.78</u>	96.73
		Avg ASR $\downarrow$	<u>4.18</u>	2.95	10.26
	256	Acc $\uparrow$	97.51	<u>97.80</u>	98.03
		Avg ASR $\downarrow$	2.44	<u>4.01</u>	4.27
	Avg.	Acc $\uparrow$	<u>95.57</u>	93.47	96.88
		Avg ASR $\downarrow$	2.48	<u>3.80</u>	5.11
RAID	32	Acc $\uparrow$	78.42	81.40	<u>79.30</u>
		Avg ASR $\downarrow$	17.19	32.03	<u>25.21</u>
	64	Acc $\uparrow$	80.96	87.35	<u>84.55</u>
		Avg ASR $\downarrow$	24.61	29.40	<u>29.28</u>
	128	Acc $\uparrow$	<u>89.55</u>	91.89	87.42
		Avg ASR $\downarrow$	35.83	<u>28.93</u>	16.97
	256	Acc $\uparrow$	<u>94.04</u>	94.19	92.43
		Avg ASR $\downarrow$	26.68	<u>21.60</u>	20.07
	Avg.	Acc $\uparrow$	85.74	88.71	<u>85.93</u>
		Avg ASR $\downarrow$	<u>26.08</u>	27.99	22.89
SemEval	32	Acc $\uparrow$	93.80	86.96	<u>89.01</u>
		Avg ASR $\downarrow$	14.45	4.62	<u>12.77</u>
	64	Acc $\uparrow$	88.92	<u>89.55</u>	90.97
		Avg ASR $\downarrow$	<u>11.83</u>	22.74	11.60
	128	Acc $\uparrow$	<u>93.26</u>	93.31	92.38
		Avg ASR $\downarrow$	<u>12.78</u>	16.41	8.19
	256	Acc $\uparrow$	96.09	89.55	<u>95.17</u>
		Avg ASR $\downarrow$	15.23	8.65	<u>14.30</u>
	Avg.	Acc $\uparrow$	93.02	89.84	<u>91.88</u>
		Avg ASR $\downarrow$	13.57	<u>13.10</u>	11.72
Overall	Avg.	Acc $\uparrow$	<u>93.56</u>	92.08	93.60
		Avg ASR $\downarrow$	<u>10.64</u>	11.27	10.04

Table 16: **Performance of REACT under Different Generator $\mathcal{G}$.** The best result in each row is **bolded**, and the second-best is underlined.

mitigate this through fixed budgets, controlled comparisons, and explicit runtime reporting. More sample-efficient adversarial selection and lower-cost generation are promising directions for reducing the footprint further.