

On the Learnability of Antidistillation Reasoning Traces

Hongyu Zhu¹, Sichu Liang², Wenwen Wang³, Bofan Chen¹, Hanyi Wang¹, Shi-Lin Wang^{1,†}

¹Shanghai Jiao Tong University ²Southeast University ³Carnegie Mellon University
{hongyu_z, wsl}@sjtu.edu.cn

[†]Corresponding author Code: github.com/GradOpt/SIFT

Abstract

Unauthorized distillation turns reasoning traces into a conduit for transferring proprietary models’ reasoning behavior to smaller students. Antidistillation Sampling (ADS) perturbs decoding to preserve task utility for legitimate users while obstructing student imitation. Our token-level Metropolis–Hastings diagnosis reveals that current ADS-style defenses leave the trace’s *intrinsic learnability* largely intact: most tokens still follow the original teacher, while a few teacher-uncertain positions induce extreme loss and dominate optimization. ADS *hides* learnable reasoning behind a sparse high-loss tail instead of eliminating it.

We therefore reformulate reasoning distillation as offline reinforcement learning from imperfect demonstrations, with credit assignment that reinforces high-quality reasoning behavior while suppressing disruptive segments. Across two teachers, six students, and six datasets, the resulting learner extracts reasoning from antidistillation traces, outperforming maximum-likelihood training and eight supervised fine-tuning baselines by up to 21%. We then turn the same principle into an adaptive defense that directly targets high-value reasoning behaviors, exploiting the perceptual gap between humans and models to preserve readability while disrupting model-visible Unicode sequences. This strategy sharply degrades both classical distillation and the adaptive learner, and scales to protect GPT-5.5 traces. These results suggest that anti-distillation should optimize for low *learnability*, not high token-level surprise.

1 Introduction

Large reasoning models do not merely return answers; they emit *reasoning traces* (Zhang et al., 2025a). For benign users, these traces make cognitive behaviors such as planning, reflection, self-correction, and verification inspectable. For an abusive distiller, the same text becomes a curriculum: once a proprietary model exposes its reasoning, the

trace can be collected and used to train a smaller student (Zhang et al., 2026a).

Proprietary model providers therefore prohibit using outputs for model training and have begun announcing anti-distillation measures (Google Threat Intelligence Group, 2026; Anthropic, 2026). In practice, protection still relies on detecting or filtering suspicious API access, even though distributed harvesting can be disguised as ordinary benign usage and the provider cannot observe how returned text is later reused. A complementary defense should act on the trace itself, preserving readability and task utility while reducing *learnability*.

Antidistillation Sampling (ADS) (Savani et al., 2025) and follow-up methods (Li et al., 2025) pursue this goal by perturbing decoding so that returned traces still solve the task while increasing a proxy student’s loss. This follows the natural intuition that the harder a trace is to predict, the harder it should be to imitate. Yet, the hidden mechanism remains unclear: does ADS discover a new reasoning manifold that stays readable to humans while becoming inaccessible to students, or does it mainly interfere with the standard maximum-likelihood estimation (MLE) objective? This distinction is central to whether anti-distillation has truly reduced the learnability of the trace itself.

As Figure 2 shows, a token-level Metropolis–Hastings (MH) diagnostic reveals **sparse disruption**. Most ADS tokens remain close to the unperturbed teacher distribution, while large shifts concentrate on a few teacher-uncertain positions. For these rare tokens, the student’s predicted probability $\pi_\theta(x_t | s_t)$ is close to zero, making the MLE loss $\log \frac{1}{\pi_\theta(x_t | s_t)}$ explode and dominate the batch gradient. Thus current anti-distillation can *hide useful reasoning behind a small high-loss tail* without erasing the learnable reasoning patterns.

This suggests that MLE fails on ADS traces by pursuing a high-recall, *mode-covering* target: the student must assign probability mass to every token

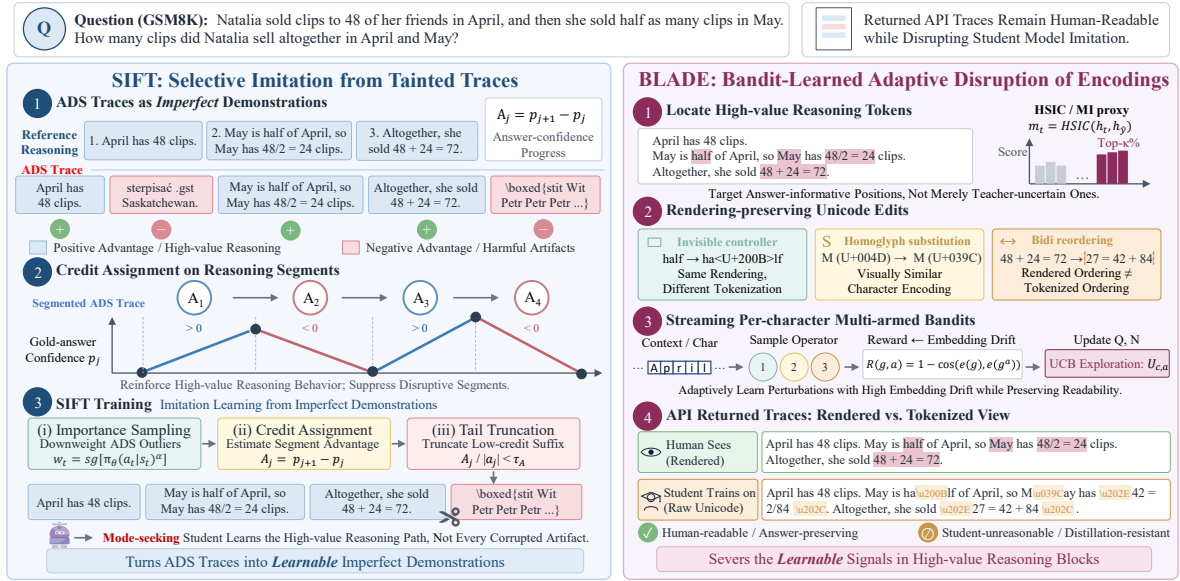


Figure 1: **Learnable reasoning signals drive both attack and defense.** SIFT (left, adversary) treats ADS traces as imperfect demonstrations and imitates surviving useful reasoning; BLADE (right, defender) targets the same signals with rendering-preserving Unicode edits that keep traces readable while breaking model-visible strings.

in the trace, including corrupted artifacts. At inference, the desired behavior is high-precision and *mode-seeking*; the student should generate high-quality reasoning traces rather than reproduce every behavior observed during training. However, simply discarding low-probability tokens is too crude, as Figure 5 shows this region is heterogeneous, mixing destructive artifacts with valuable reasoning steps beyond the student’s current competence.

Inspired by reinforcement learning (RL), which learns from trajectories containing *both successful and failed actions* (Sutton et al., 1998), we introduce **SIFT** (Selective Imitation from Tainted Traces), shown in Figure 1 (left). SIFT reformulates reasoning distillation as an offline-RL problem of *imitation learning from imperfect demonstrations*: credit assignment reinforces high-value reasoning behavior while suppressing disruptive segments, so the student seeks successful reasoning modes instead of copying the entire tainted trace. Across two teacher models, six students, and six datasets, SIFT consistently extracts useful reasoning from traces produced by current anti-distillation methods, outperforming vanilla MLE and eight SOTA supervised fine-tuning (SFT) baselines by up to 21%. Existing defenses can make MLE fail, yet they do not reliably remove the recoverable reasoning signal.

The same diagnosis can be turned into a defense by treating *learnable reasoning signals* as

the target to suppress. As shown in Figure 1 (right), **BLADE** (Bandit-Learned Adaptive Disruption of Encodings) uses SIFT’s credit-assignment signal to locate reasoning patterns to disrupt, then exploits the *perceptual gap between humans and models*: humans read rendered glyphs, while students train on the underlying Unicode string. Invisible Unicode controls, homoglyphs, and bidirectional reordering preserve the human-visible reasoning and final answer while fracturing the model-visible sequence (Boucher et al., 2022). A streaming per-character bandit chooses high-drift edits without fixed mappings. BLADE substantially degrades both MLE and SIFT, with little impact on accuracy, readability, or overhead, and scales to protect GPT-5.5 (OpenAI, 2026) traces.

Contributions. (i) *Mechanism.* We show ADS-style defenses mainly create a sparse high-loss tail that dominates MLE, while leaving much of the reasoning signal intact. (ii) *Attack.* We introduce SIFT, an offline-RL distillation that performs mode-seeking imitation from tainted traces and exposes an adaptive attack surface. (iii) *Defense.* We propose BLADE, which repurposes this diagnosis to disrupt model-visible Unicode encodings of learnable reasoning signals while preserving readability. (iv) *Evidence.* Across various settings, SIFT recovers reasoning from anti-distillation traces, while BLADE suppresses MLE and SIFT distillation and scales to frontier proprietary-model traces.

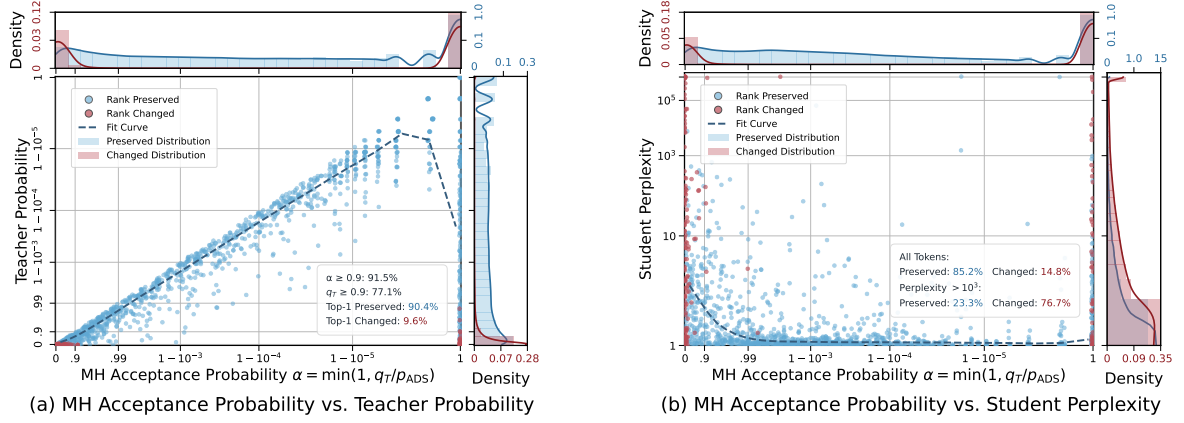


Figure 2: **MH diagnostics on GSM8K ADS traces.** (a) **ADS-Medium**; (b) **ADS-Heavy**. Teacher/Student: DeepSeek-R1-Distill-Qwen-7B/Qwen-2.5-1.5B. MH acceptance increases with Teacher Probability and decreases with Student Perplexity: most tokens stay teacher-aligned, while rare token changes create a high-loss tail.

2 Preliminary Study

2.1 Problem Formulation

Let π_T be a proprietary reasoning model served by a provider $\mathcal{M}$ through a black-box API. Given a query q , the model autoregressively samples a trace $x = (x_1, \dots, x_T)$, where the decoding state is $s_t = (q, x_{<t})$ and $x_t \sim \pi_T(\cdot | s_t)$. For verifiable tasks, an answer extractor ζ maps a trace to $\hat{y} = \zeta(x)$, and correctness is measured by the indicator function $r(x, y) = \mathbb{I}\{\zeta(x) = y\}$ (Guo et al., 2025; Zhang et al., 2025a).

Threat model. An adversary $\mathcal{A}$ collects training questions $\mathcal{Q}_{\text{tr}} = \{(q_i, y_i)\}_{i=1}^N$, queries the API, and obtains a trace set $\mathcal{D} = \{(q_i, x_i)\}_{i=1}^N$. The adversary can choose the queries and the distillation algorithm $\phi \in \Phi$, but only observes the returned text traces: no parameters, gradients, logits, or token probabilities are available. The standard attack trains a student π_θ by Maximum Likelihood Estimation (MLE) (Ouyang et al., 2022; Zhang et al., 2026a; Savani et al., 2025),

$$\mathcal{L}_{\text{MLE}}(\theta; \mathcal{D}) = -\mathbb{E}_{(q,x) \sim \mathcal{D}} \sum_{t=1}^{T_x} \log \pi_\theta(x_t | q, x_{<t}). \quad (1)$$

Defender objective. The provider applies an antidistillation mechanism $\psi \in \Psi$ during decoding or after generation, yielding $\tilde{x} \sim (\psi \circ \pi_T)(\cdot | q)$ and $\mathcal{D}_\psi = \{(q_i, \tilde{x}_i)\}_{i=1}^N$. Let $\text{Acc}_{\mathcal{Q}}(\mu) = |\mathcal{Q}|^{-1} \sum_{(q,y) \in \mathcal{Q}} \mathbb{E}_{x \sim \mu(\cdot|q)} r(x, y)$, and abbreviate $\text{Acc}_{\text{te}} = \text{Acc}_{\mathcal{Q}_{\text{te}}}$ for a held-out set $\mathcal{Q}_{\text{te}} \cap \mathcal{Q}_{\text{tr}} = \emptyset$.

Ideally, the defense solves the bilevel game

$$\begin{aligned} \min_{\psi \in \Psi} \max_{\phi \in \Phi} \quad & \text{Acc}_{\text{te}}(\pi_{\phi(\mathcal{D}_\psi)}) \\ \text{s.t.} \quad & \text{Acc}_{\text{te}}(\pi_T) - \text{Acc}_{\text{te}}(\psi \circ \pi_T) \leq \epsilon. \end{aligned} \quad (2)$$

Thus, ψ makes returned traces hard to distill while preserving the protected model’s task utility. It should also retain trace readability so benign users and safety monitors can inspect the reasoning.

2.2 Antidistillation Sampling

Directly optimizing the bilevel objective is infeasible, since it would require searching over attacks and retraining students. Antidistillation Sampling (ADS) (Savani et al., 2025) instead follows a contrastive-decoding idea (Li et al., 2023). At each state, it tilts the teacher distribution toward tokens predicted to increase a proxy student’s loss.

For a proxy student π_P , let ℓ be its MLE loss computed on a held-out clean trace set $\mathcal{H}$, and set $g = \nabla_{\theta_P} \ell$. Using nearby proxy policies $\pi_{\pm} = \pi_{\theta_P \pm \varepsilon_{\text{fd}} g}$, ADS approximates the directional derivative of the token negative log-likelihood (NLL) by

$$\begin{aligned} \Delta_t(v) &= \frac{-\log \pi_+(v|s_t) + \log \pi_-(v|s_t)}{2\varepsilon_{\text{fd}}}, \\ \pi_{\text{ADS}}(v | s_t) &= \pi_T(v | s_t) e^{\lambda_{\text{ADS}} \Delta_t(v)} / Z_t. \end{aligned} \quad (3)$$

where Z_t normalizes over the vocabulary and λ_{ADS} controls the strength of the contrastive tilt. Subsequent methods such as DOGe (Li et al., 2025) amortize this objective into output embeddings, ultimately following the same principle of perturbing decoding along directions that raise student loss.

2.3 What Makes ADS Traces Hard to Learn?

What hidden mechanism makes ADS traces hard to learn? We diagnose this with a token-level Metropolis–Hastings acceptance ratio, following the correction logic used in speculative decoding (Leviathan et al., 2023). Let $q_T(\cdot | s_t) = \pi_T(\cdot | s_t)$ be the clean teacher distribution and $p_{\text{ADS}}(\cdot | s_t) = \pi_{\text{ADS}}(\cdot | s_t)$ be the ADS proposal. For an ADS token x_t , define

$$\alpha_t = \min \left\{ 1, \frac{q_T(x_t | s_t)}{p_{\text{ADS}}(x_t | s_t)} \right\}. \quad (4)$$

Viewed as speculative decoding with p_{ADS} as the draft and q_T as the target, accepting draft tokens by Eq. (4) *preserves the target distribution exactly*. Thus, α_t is a calibrated token-level measure of how closely ADS matches the teacher.

Figure 2 reveals a *sparse-disruption* mechanism.

(1) Teacher alignment: ADS remains globally close to the teacher: roughly 90% of tokens have high acceptance and preserve the teacher’s top-1 token. **(2) Uncertainty targeting:** In Figure 2(a), teacher probability increases with MH acceptance, and rank changes concentrate in low-confidence regions. ADS mainly perturbs tokens the teacher already finds replaceable. **(3) Student loss amplification:** MLE token loss $-\log \pi_\theta(x_t | s_t) = \log(1/\pi_\theta(x_t | s_t))$ is unbounded as student confidence goes to zero. In Figure 2(b), rank-changed tokens are only 14.8% of all tokens but 76.7% of tokens with perplexity above 10^3 . *This small high-loss tail can dominate the batch gradient.*

Overall, ADS does not completely erase useful reasoning; rather, a few student-confusing outliers hide it from standard MLE.

3 Learning from ADS Traces

3.1 Mode-Covering vs. Seeking Imitation

In ADS traces, most tokens remain close to the teacher, yet a small high-loss tail can dominate standard MLE updates. The key mismatch is *high-recall, mode-covering* imitation versus *high-precision, mode-seeking* reasoning: MLE forces the student to put probability mass on all ADS tokens, including confusing artifacts. At inference, however, every student output should be a high-quality reasoning trace; the model need not reproduce every observed behavior. We therefore propose **SIFT** (Selective Imitation from Tainted Traces), casting ADS-trace distillation as *imitation learning from imperfect demonstrations*, with

offline-RL credit assignment reinforcing useful reasoning actions while suppressing harmful ones.

Reasoning as a Markov Decision Process (MDP).

We formulate each reasoning trace as a finite-horizon MDP (Pang and He, 2021): at step t , the state is the query plus prefix $s_t = (q, x_{<t})$, the token action is $a_t = x_t$, and the transition is deterministic because the next state appends the chosen token to the context. For verifiable tasks, rewards are delayed: $r_T = \mathbb{I}\{\zeta(x) = y\}$, $r_t = 0$ for $t < T$, and we set $\gamma = 1$ to avoid exponential decay of delayed reward. Let the return be $G(\tau) = \sum_t r_t$ and $J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[G(\tau)]$. The policy gradient (Sutton et al., 1999) is

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_t \nabla_\theta \log \pi_\theta(a_t | s_t) A_t^\pi \right]. \quad (5)$$

The *advantage* is defined as

$$A_t^\pi = Q^\pi(s_t, a_t) - V^\pi(s_t) \quad (6)$$

It is the credit assigned to action a_t : positive actions are reinforced, while negative actions are suppressed. Unlike MLE, this selectively matches actions according to their contribution to the final outcome rather than copying every token equally.

Although likelihoods are token-level, we estimate credit on reasoning segments since a meaningful reasoning step is often expressed by a short phrase or clause. For segment $a_j = x_{b_j:e_j}$ at state s_j , teacher forcing gives

$$\log \pi_\theta(a_j | s_j) = \sum_{t=b_j}^{e_j-1} \log \pi_\theta(x_t | q, x_{<t}). \quad (7)$$

Thus tokens remain the likelihood units, while segments serve as MDP actions for credit assignment.

3.2 Importance Sampling (IS)

Equation (5) is on-policy, whereas distillation uses offline traces sampled from a demonstration policy μ (e.g., π_{ADS}). An exact importance sampling (IS) correction (Precup et al., 2000) gives an unbiased estimate using the cumulative ratio $\rho_{1:t} = \prod_{u \leq t} r_u$, where $r_u = \pi_\theta(a_u | s_u) / \mu(a_u | s_u)$:

$$g_{\text{IS}} = \mathbb{E}_{\tau \sim \mu} \left[\sum_t \rho_{1:t} \nabla_\theta \log \pi_\theta(a_t | s_t) A_t^\pi \right]. \quad (8)$$

As this product has high variance and black-box APIs rarely expose sampler probabilities after ADS

perturbations, we use a local, denominator-free IS weight for token action $a_t = x_t$:

$$w_t = \text{sg}[\pi_\theta(a_t | s_t)^{\alpha_{\text{IS}}}], \quad \alpha_{\text{IS}} \in [0, 1], \quad (9)$$

where $\text{sg}[\cdot]$ denotes stop-gradient. This mildly downweights vanishing-probability ADS outliers.

3.3 Credit Assignment (CA)

The remaining term in Eq. (5) is the advantage A_t^π . Standard estimators fit this setting poorly: Monte Carlo (MC) returns assign the same terminal score to every step, temporal-difference (TD) critics are unstable under long sparse rewards, and GRPO-style group baselines broadcast one trace-level advantage to all tokens (Kazemnejad et al., 2025). We instead exploit the delayed-reward structure of verifiable reasoning.

Proposition 1 (Credit as progress). *Consider a deterministic reasoning MDP with $r_t = 0$ for $t < T$, $r_T \in \{0, 1\}$, and $\gamma = 1$. Let $s^+ = f(s, a)$ be the next state after taking action a at state s , and define $p_\pi(s) = \Pr_\pi(r_T = 1 | s)$. Then*

$$A^\pi(s, a) = p_\pi(s^+) - p_\pi(s). \quad (10)$$

Moreover, for $Q^\pi(s, a)$, $V^\pi(s) > 0$,

$$\begin{aligned} A_{\log}^\pi(s, a) &\triangleq \log Q^\pi(s, a) - \log V^\pi(s) \\ &= i_\pi(r_T = 1; a | s), \end{aligned} \quad (11)$$

where $i_\pi(\cdot; \cdot | s)$ is pointwise conditional mutual information between the action and final success.

The proof is in Appendix A. Proposition 1 gives two useful views of credit. The difference form is bounded and stable, so SIFT uses it to ask whether a segment increases answer-readiness confidence. The log form is an information signal for ranking answer-relevant actions, which Section 5.1 reuses to localize high-value tokens.

For a segmented trace $x = a_1 \circ \dots \circ a_m$ with states $s_j = (q, a_{<j})$, let π_{θ_0} be the frozen pre-SIFT student. At each boundary, append the cue c_{ans} , written as "\n\nFinal Answer: \boxed{", and teacher-force the gold answer y . The answer confidence $\hat{p}_j$ is the geometric-mean probability of y under π_{θ_0} , and the segment credit is $\hat{A}_j = \hat{p}_{j+1} - \hat{p}_j$, where s_{j+1} appends segment a_j to s_j . We smooth positive and negative credits separately:

$$\tilde{A}_j = \begin{cases} \hat{A}_j^{\alpha_A^+}, & \hat{A}_j \geq 0, \\ -(-\hat{A}_j)^{\alpha_A^-}, & \hat{A}_j < 0, \end{cases} \quad (12)$$

with $\alpha_A^+, \alpha_A^- \in [0, 1]$. These credits are computed once before training and kept fixed.

3.4 Tail Truncation (TT)

ADS traces often deteriorate near the end: once the trajectory enters a bad belief state, the suffix may contain circular arguments, repeated restatements, or fluent but non-progressive reasoning. We therefore scan backward and discard trailing segments while $\hat{A}_j/|a_j| < \tau_A$, retaining prefix $a_{1:m^*}$ for training. Tail truncation removes misleading suffixes after belief deviation, reduces train-inference mismatch from corrupted long contexts, and shortens training sequences. Together, IS weighting, CA, and TT turn ADS traces from brittle token targets into usable imperfect demonstrations.

4 SIFT Evaluation

Setup. Following ADS (Savani et al., 2025), our main setting uses DeepSeek-R1-Distill-Qwen-7B (Guo et al., 2025) as the teacher and Qwen-2.5-1.5B¹ or LLaMA-3.2-3B (Grattafiori et al., 2024) as the student on GSM8K, MATH, and MMLU (Cobbe et al., 2021; Hendrycks et al., 2021b,a). For each dataset, we train on 5,000 teacher-query traces and evaluate on 1,000 questions. For scaling, we use GPT-OSS-20B (Agarwal et al., 2025) as the teacher and Qwen-3-4B (Yang et al., 2025), Qwen-3.5-2B (Qwen Team, 2026), Ministral-3-8B (Liu et al., 2026a), and OLMo-3-7B (Olmo et al., 2025) as students on AIME24, AIME25, and GPQA-Diamond (Rein et al., 2024). Training queries come from NuminaMath-1.5² and SuperGPQA (Du et al., 2025), deduplicated against all test sets. We report accuracy averaged over 8 runs (ACC@8) for AIME and ACC@3 otherwise.

Trace sources include clean teacher outputs and ADS (Savani et al., 2025) or DOGe (Li et al., 2025) outputs at MEDIUM and HEAVY perturbation strengths. Baselines include vanilla MLE; SOTA SFT objectives DFT (Wu et al., 2026), ASFT (Zhu et al., 2026a), and ProFit (Liu et al., 2026b); student rephrasing (Pan et al., 2025); data selection with QuRating (Wettig et al., 2024) and GRAPE (Zhang et al., 2025b); and noise-tolerant objectives SCE (Wang et al., 2019) and VCE (Wang et al., 2026). Appendix C provides further implementation details.

Learning from antidistillation traces. Table 1 shows that vanilla MLE is brittle: medium antidistillation already weakens student distillation, and

¹<https://huggingface.co/Qwen/Qwen2.5-1.5B>

²<https://huggingface.co/datasets/AI-MO/NuminaMath-1.5>

Table 1: SIFT with DeepSeek-R1-Distill-Qwen-7B teacher. **Green**: SIFT-MLE gain; **red**: below pre-MLE.

Dataset	Pre-MLE		Method	Teacher	Qwen-2.5-1.5B		LLaMA-3.2-3B	
	Qwen	LLaMA			Vanilla-MLE	SIFT	Vanilla-MLE	SIFT
GSM8K	57.5	31.4	Clean	91.7	72.5 $\pm$ 1.3	74.9 $\pm$ 1.7 (+2.4)	60.3 $\pm$ 1.7	64.2 $\pm$ 2.0 (+3.9)
			ADS (Medium)	84.9	66.0 $\pm$ 2.8	70.4 $\pm$ 1.5 (+4.4)	51.3 $\pm$ 2.7	59.1 $\pm$ 2.1 (+7.9)
			ADS (Heavy)	75.6	42.7 $\pm$ 1.7	63.6 $\pm$ 2.2 (+21.0)	32.3 $\pm$ 1.6	50.4 $\pm$ 1.7 (+18.1)
			DOGe (Medium)	84.2	64.1 $\pm$ 2.1	68.9 $\pm$ 2.0 (+4.8)	50.6 $\pm$ 2.5	55.5 $\pm$ 1.2 (+4.9)
			DOGe (Heavy)	74.5	54.5 $\pm$ 1.7	62.0 $\pm$ 1.9 (+7.5)	44.4 $\pm$ 1.9	50.6 $\pm$ 0.9 (+6.2)
MATH	34.9	10.4	Clean	76.4	43.3 $\pm$ 3.3	43.9 $\pm$ 1.7 (+0.6)	16.3 $\pm$ 1.0	16.4 $\pm$ 1.8 (+0.1)
			ADS (Medium)	71.4	38.1 $\pm$ 2.4	42.7 $\pm$ 1.8 (+4.6)	12.7 $\pm$ 0.6	17.1 $\pm$ 1.3 (+4.3)
			ADS (Heavy)	65.8	32.9 $\pm$ 2.5	40.3 $\pm$ 2.5 (+7.5)	11.4 $\pm$ 1.8	13.9 $\pm$ 2.8 (+2.5)
			DOGe (Medium)	71.0	34.6 $\pm$ 1.1	38.7 $\pm$ 2.5 (+4.1)	12.8 $\pm$ 1.4	14.3 $\pm$ 1.0 (+1.5)
			DOGe (Heavy)	64.8	33.3 $\pm$ 4.0	37.9 $\pm$ 1.0 (+4.7)	12.3 $\pm$ 2.0	13.4 $\pm$ 1.5 (+1.1)
MMLU	49.4	41.9	Clean	64.4	54.3 $\pm$ 1.3	56.3 $\pm$ 1.4 (+2.0)	51.4 $\pm$ 2.5	54.3 $\pm$ 1.0 (+2.9)
			ADS (Medium)	60.9	49.6 $\pm$ 1.8	54.5 $\pm$ 1.7 (+4.9)	46.4 $\pm$ 2.4	51.0 $\pm$ 2.0 (+4.6)
			ADS (Heavy)	56.9	47.0 $\pm$ 1.5	53.8 $\pm$ 1.3 (+6.8)	42.2 $\pm$ 1.7	50.4 $\pm$ 1.0 (+8.2)
			DOGe (Medium)	60.1	49.9 $\pm$ 1.5	54.7 $\pm$ 1.1 (+4.8)	47.8 $\pm$ 1.3	50.6 $\pm$ 1.6 (+2.9)
			DOGe (Heavy)	56.2	47.5 $\pm$ 1.6	54.0 $\pm$ 1.5 (+6.5)	45.7 $\pm$ 1.8	47.8 $\pm$ 2.3 (+2.1)

heavy perturbations can erase the gain or push the student below its pre-MLE checkpoint. SIFT reverses this failure in every setting, improving over MLE by up to 21.0%, with larger gains under stronger perturbations. Table 2 shows the same pattern with GPT-OSS-20B and harder benchmarks: SIFT improves over MLE across models and perturbation strengths. Notably, *ADS-style perturbations also lower teacher accuracy sharply*, by much more than they lower the student side under SIFT, revealing a large teacher-utility cost.

selection ratio and can be hacked by repetitive patterns. Noise-tolerant losses avoid overconfidently fitting low-confidence tokens, yet still raise disruptive patterns along with helpful reasoning. SIFT is consistently stronger since it assigns *credit* to reasoning segments, rather than to tokens solely by confidence or surface quality.

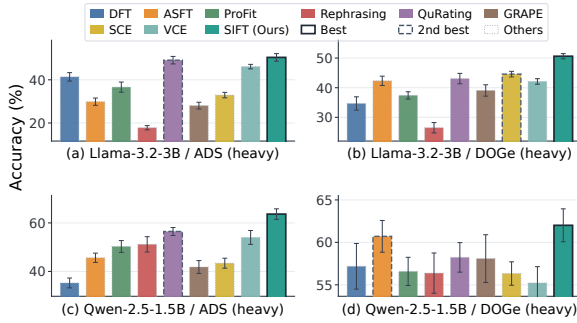


Figure 3: SIFT versus SFT baselines on GSM8K.

Strong SFT baselines. Figure 3 tests whether learning from antidistillation traces could be solved by a stronger supervised objective. Recent SFT objectives such as DFT, ASFT, and ProFit reduce overfitting to low-probability tokens, yet converge slowly when data are limited and noisy. Rephrasing frequently removes high-value reasoning behaviors such as reflection and backtracking. Data selection is more competitive; however, it requires a fixed

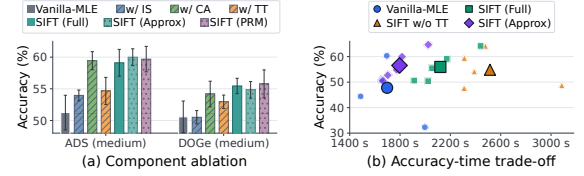


Figure 4: (a) Ablation of SIFT components; (b) Accuracy-runtime trade-off across SIFT variants.

Ablations. Figure 4(a) shows importance sampling (IS), credit assignment (CA), and tail truncation (TT) each improve over MLE, while their combination performs best, indicating outlier reweighting, advantage estimation, and corrupted-tail removal are complementary. To reduce credit-estimation cost, SIFT-Approx evaluates only 10% of segments and predicts the remaining credits from student hidden-state embeddings using a two-layer feed-forward network; its MSE rarely exceeds 0.1, and the final accuracy remains close to standard SIFT. A variant that uses a process reward model (PRM) (Cheng et al., 2025) achieves comparable performance, suggesting that SIFT can flexibly incorporate different sources of credit. Figure 4(b) further shows the accuracy-time trade-off: TT reduces cost by truncating low-quality trailing seg-

Table 2: Scaling to GPT-OSS-20B and harder benchmarks. AIME uses ACC@8; GPQA-Diamond uses ACC@3.

Model	Method	AIME24				AIME25				GPQA-Diamond			
		Pre-MLE	Clean	Medium	Heavy	Pre-MLE	Clean	Medium	Heavy	Pre-MLE	Clean	Medium	Heavy
Teacher	N/A	N/A	65.0	52.5	45.0	N/A	58.3	54.2	45.8	N/A	65.2	61.6	49.0
Qwen-3-4B	Vanilla-MLE	6.7	17.5	16.7	15.0	4.2	16.7	14.2	14.2	35.4	45.0	39.9	38.4
	SIFT		20.0	20.8	20.0		22.5	17.5	16.7		46.5	42.9	42.4
Qwen-3.5-2B	Vanilla-MLE	15.0	19.2	17.5	18.3	13.3	16.7	15.8	14.2	43.4	51.0	46.5	43.4
	SIFT		20.0	20.8	19.2		21.7	15.8	15.0		53.0	52.1	50.0
Ministral-3-8B	Vanilla-MLE	4.2	15.0	15.8	8.3	4.2	15.0	12.5	11.7	29.3	30.8	29.8	27.3
	SIFT		19.2	16.7	12.5		16.7	14.2	12.5		40.9	37.4	33.3
OLMo-3-7B	Vanilla-MLE	10.0	14.2	10.0	9.2	10.8	13.3	11.7	10.8	22.2	36.4	30.8	34.3
	SIFT		17.5	10.8	11.7		13.3	11.7	10.8		37.4	32.8	35.4

ments, while SIFT-Approx preserves most gains with only marginal overhead over MLE.



Figure 5: Top-200 Low-probability ADS tokens.

Why credit assignment matters. Since ADS creates sparse extreme-loss tokens, directly suppressing low-probability tokens appears to be a natural remedy; DFT, ASFT, and ProFit instantiate a similar principle. However, Figure 5 shows why probability alone is inadequate: low-probability tokens are heterogeneous. In addition to disruptive artifacts, they include reasoning, arithmetic, and connective patterns that the student initially underestimates but must learn for reasoning. Thus, low probability does not imply low value.

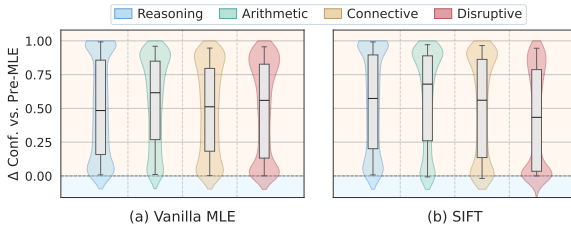


Figure 6: Mode covering vs. mode seeking learner.

Credit assignment supplies this missing distinction. Figure 6 shows that MLE raises the probabilities of nearly all observed low-probability tokens,

exhibiting mode-covering behavior. SIFT is *mode seeking* as it reinforces reasoning, arithmetic, and connective tokens more strongly while fitting disruptive tokens less aggressively. This matches our offline-RL view: the learner should reinforce segments that improve final correctness rather than imitate every token in a tainted trace.

5 BLADE: Severing Learning Signals

Existing Antidistillations mostly hit teacher-uncertain, locally replaceable tokens. A stronger defense should instead center on the trace’s *learnability* itself. We propose **BLADE** (Bandit-Learned Adaptive Disruption of Encodings), a defense for API serving that preserves human readability. BLADE has three stages: locate high-value tokens, perturb their Unicode representation, and learn a streaming policy over perturbation operators; details are in Appendix B.

5.1 Striking High-Value Reasoning Tokens

At serving time, the defender does not know the ground-truth answer. We use the model’s own extracted answer $\hat{y} = \zeta(x)$ as the endpoint proxy. Let π_η be a lightweight proxy model, h_t the last-layer state of token x_t , and $h_{\hat{y}}$ a pooled representation of the extracted answer span. Proposition 1 connects log-advantage to pointwise mutual information with the final outcome. We approximate this ranking with a forward-only HSIC estimator. Define coordinate-wise kernel matrices $[K_t]_{ij} = k(h_{t,i}, h_{t,j})$ and $[K_{\hat{y}}]_{ij} = k(h_{\hat{y},i}, h_{\hat{y},j})$, and let $H = I_d - d^{-1}\mathbf{1}\mathbf{1}^\top$. Then

$$m_t = \text{HSIC}(h_t, h_{\hat{y}}) = \frac{1}{(d-1)^2} \text{tr}(K_t H K_{\hat{y}} H). \quad (13)$$

We target $\mathcal{I}_\kappa(x) = \text{Top}_{\kappa\%}(\{m_t\}_{t=1}^T)$, prioritizing learnable information over teacher uncertainty.

5.2 Exploiting the Perceptual Gap

Directly replacing high-value tokens can damage task utility. BLADE instead applies rendering-preserving Unicode transformations to exploit a structural gap: humans read rendered glyphs, while students train on tokenized code points (Boucher et al., 2022). We use three transformations: **Invisible Controllers** insert imperceptible characters, e.g., `proof` and `pro<U+200B>of` render identically but tokenize differently; **Homoglyph Substitution** replaces a character with a visually similar code point, e.g., Latin C (U+0043) can become U+0421, a Cyrillic Es that looks like C but is linguistically closer to S; and **Bidirectional Reordering** makes visual order diverge, e.g., `U+202E olleH U+202C` can render as `Hello` while the model sees a differently ordered string.

5.3 Learning Perturbations Online

Unicode transformations provide many invisible edits. Exhaustive search is wasteful, while a deterministic map may be exploited by the adversary. BLADE instead uses a streaming bandit: for each selected token it samples one Unicode operator, scores local embedding drift, and updates online. Algorithm details are in Appendix B.

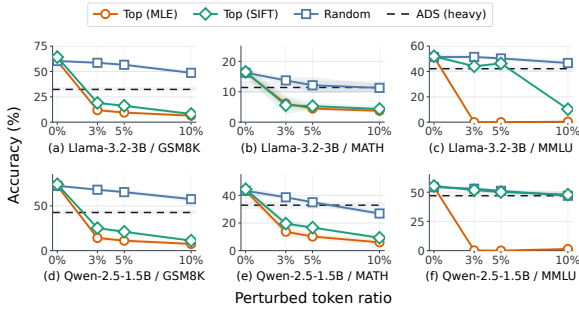


Figure 7: Adaptive BLADE Antidistillation.

6 BLADE Evaluation

Adaptive defense against SIFT. Figure 7 evaluates BLADE against MLE and SIFT across perturbation budgets. Targeting high-value reasoning carriers, BLADE severely degrades both learners, whereas the random variant with uniform token and operator selection is much weaker. Unlike ADS and DOGe, which substantially reduce teacher accuracy, BLADE leaves the rendered trace nearly unchanged. The human study in Figure 8(a) (Appendix D) shows that BLADE traces receive much higher preference ranks than visibly corrupted al-

ternatives under standard HTML rendering. Appendix E.2 further evaluates BLADE against adaptive preprocessing, stronger distillation objectives, and OCR-based reconstruction. None of these can reliably recover the performance of clean-trace distillation with low cost.

Scaling to flagship-model traces. Following the protocol of Zhang et al. (2026b), we evaluate reconstructed GPT-5.5 (xhigh) reasoning traces. Because GPT-5.5 does not expose its complete internal reasoning process, we construct the traces by asking GPT-5.5 to refine student-generated candidate reasoning toward its own final answer, as detailed in Appendix C.6. Figure 8(b) shows that distilling these traces improves Qwen-3.5-9B, whereas BLADE-protected traces sharply degrade the student. To complement this indirect setting, Appendix E.3 further evaluates complete reasoning traces directly exposed by DeepSeek-V4 and observes the same pattern across AIME24, AIME25, and GPQA-Diamond. Moreover, BLADE’s streaming bandit post-processes completed traces, adding only about three minutes over the AIME trace subset—negligible relative to frontier-model reasoning latency.

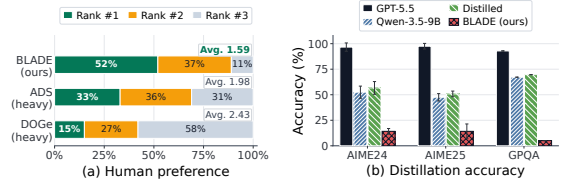


Figure 8: (a) BLADE preserves human preference; (b) Scaling to protect proprietary model traces.

Mechanism of BLADE. Figure 9(a) visualizes the post-training geometry of digit embeddings. Clean traces induce a near-circular, approximately equidistant digit structure, consistent with the subspace geometry used for arithmetic computation (Kantamneni and Tegmark, 2025). BLADE disrupts this structure, indicating that targeted perturbations on calculation spans damage the student’s internal arithmetic behavior. In Figure 9(b), ADS keeps the token-level mutual information close to clean traces, whereas BLADE strongly suppresses high-information regions. Thus, BLADE works by removing recoverable reasoning signals rather than merely destabilizing MLE optimization.

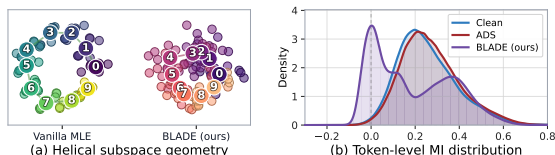


Figure 9: Mechanism of BLADE: arithmetic geometry and token-level information distribution.

7 Conclusion

We show that ADS-style defenses mainly impede MLE through sparse high-loss tokens, leaving reasoning signals recoverable. SIFT exploits this residual learnability via credit assignment, while BLADE turns the same insight into a stronger defense by suppressing high-value model-visible signals. Robust anti-distillation should therefore minimize *intrinsic trace learnability*, rather than merely increase token-level surprise.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (NSFC) under Grant No. 62572315.

Limitations

Our study is bounded by model scale. Due to compute constraints, the largest open-source teacher in our experiments is a 20B MoE model (GPT-OSS-20B), and all student models are below 10B parameters. To probe behavior beyond this regime, we conduct a GPT-5.5 (xhigh) trace-protection experiment. This may provide an initial scaling signal that the learnability-centered view and BLADE’s defense mechanism remain relevant for frontier proprietary traces.

SIFT currently performs credit assignment at the segment level. Finer token-level credit assignment could more precisely separate useful reasoning from disruptive artifacts, yet estimating answer-readiness at every token boundary introduces substantial overhead. Moreover, teacher-forcing the final answer can be myopic when used as a dense token-level signal. As a result, SIFT may still retain some disruptive artifacts. This motivates intrinsic credit signals that bridge segment and token-level assignment under controlled computational cost.

Text repair, normalization, or other preprocessing pipelines may partially suppress BLADE’s Unicode transformations. We mitigate this risk with a streaming multi-arm bandit that diversifies perturbation operators online, making the corruption

less deterministic and harder to remove with fixed cleaning rules. More broadly, BLADE is not tied to invisible Unicode edits: our core principle is to target high-value reasoning spans and reduce *intrinsic learnability*. Visible but readability-preserving perturbations, such as local character- or token-level reordering, may offer similar protection under different utility–readability trade-offs.

Ethical Considerations

This work studies both an adaptive attack and a corresponding defense for reasoning-trace anti-distillation. We recognize the **potential risk** that SIFT, as an attack against existing anti-distillation mechanisms, could be misused to extract value from protected model traces. Our intent, however, is not to facilitate unauthorized distillation, but to identify the technical pressure points that determine whether trace-level defenses are truly robust. Current protection for proprietary models still largely depends on API-side detection, filtering, and access control, while proactive perturbation-based anti-distillation remains an emerging research direction. At this stage, carefully exposing plausible adaptive attack surfaces is important for developing defenses that remain effective beyond standard MLE-based distillation.

We therefore present SIFT together with BLADE, a defense explicitly designed against the same credit-assignment learner. This paired evaluation reduces the risk of leaving an attack without a mitigation path: SIFT shows that sparse high-loss perturbations are insufficient, while BLADE demonstrates a stronger principle for suppressing high-value model-visible reasoning signals and substantially degrading both MLE and SIFT distillation. More broadly, our results argue for evaluating anti-distillation under adaptive learners before deployment, so that proprietary-model providers can move from brittle surface-level obfuscation toward defenses based on intrinsic trace learnability.

References

- Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. 2025. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*.
- Anthropic. 2026. [Detecting and Preventing Distillation Attacks](#). Anthropic News. Published February 23, 2026.

- Nicholas Boucher, Ilia Shumailov, Ross Anderson, and Nicolas Papernot. 2022. Bad characters: Imperceptible nlp attacks. In *2022 IEEE symposium on security and privacy (SP)*, pages 1987–2004. IEEE.
- Jie Cheng, Gang Xiong, Ruixi Qiao, Lijun Li, Chao Guo, Junle Wang, Yisheng Lv, and Fei-Yue Wang. 2025. [Stop summation: Min-form credit assignment is all process reward model needs for reasoning](#). In *Advances in Neural Information Processing Systems*, volume 38, pages 131646–131671. Curran Associates, Inc.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Xeron Du, Yifan Yao, Kaijing Ma, Bingli Wang, Tianyu Zheng, Zhu, Minghao Liu, Yiming Liang, Xiaolong Jin, Zhenlin Wei, Chujie Zheng, Kaixin Deng, Shuyue Guo, Shian Jia, Sichao Jiang, Yiyao Liao, Rui Li, Qinru Li, Sirun Li, and 76 others. 2025. [Supergpqa: Scaling llm evaluation across 285 graduate disciplines](#). In *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc.
- Google Threat Intelligence Group. 2026. [GTIG AI Threat Tracker: Distillation, Experimentation, and \(Continued\) Integration of AI for Adversarial Use](#). Google Cloud Blog. Published February 13, 2026.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. [Measuring massive multitask language understanding](#). In *International Conference on Learning Representations*.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. [Measuring mathematical problem solving with the MATH dataset](#). In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.
- Subhash Kantamneni and Max Tegmark. 2025. [Language models use trigonometry to do addition](#). In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*.
- Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. 2025. [VinePPO: Unlocking RL potential for LLM reasoning through refined credit assignment](#).
- Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.
- Pingzhi Li, Zhen Tan, Yu-Chao Huang, Huaizhi Qu, huan liu, and Tianlong Chen. 2025. [DOGe: Defensive output generation for LLM protection against knowledge distillation](#). In *Lock-LLM Workshop: Prevent Unauthorized Knowledge Use from Large Language Models*.
- Wei Li and Houfeng Wang. 2024. [Detection-correction structure via general language model for grammatical error correction](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1748–1763, Bangkok, Thailand. Association for Computational Linguistics.
- Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori B Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2023. Contrastive decoding: Open-ended text generation as optimization. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 12286–12312.
- Alexander H Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, et al. 2026a. *Minstral 3*. *arXiv preprint arXiv:2601.08584*.
- Tao Liu, Taiqiang Wu, Runming Yang, Shaoning Sun, Junjie Wang, and Yujiu Yang. 2026b. Profit: Leveraging high-value signals in sft via probability-guided token selection. *arXiv preprint arXiv:2601.09195*.
- Pratyush Maini, Hengrui Jia, Nicolas Papernot, and Adam Dziedzic. 2024. Llm dataset inference: Did you train on my dataset? *Advances in Neural Information Processing Systems*, 37:124069–124092.
- Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, et al. 2025. *Olmo 3*. *arXiv preprint arXiv:2512.13961*.
- OpenAI. 2026. [Introducing GPT-5.5](#). OpenAI Blog. Published April 23, 2026.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.
- Leyi Pan, Aiwei Liu, Shiyu Huang, Yijian Lu, Xuming Hu, Lijie Wen, Irwin King, and Philip S Yu. 2025. Can llm watermarks robustly prevent unauthorized

- knowledge distillation? In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13228–13251.
- Richard Yuanzhe Pang and He He. 2021. [Text generation by learning from demonstrations](#). In *International Conference on Learning Representations*.
- Doina Precup, Richard S Sutton, and Satinder P Singh. 2000. Eligibility traces for off-policy policy evaluation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 759–766.
- Qwen Team. 2026. [Qwen3.5: Towards native multi-modal agents](#).
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. [GPQA: A graduate-level google-proof q&a benchmark](#). In *First Conference on Language Modeling*.
- Jie Ren, Kangrui Chen, Chen Chen, Vikash Sehwal, Yue Xing, Jiliang Tang, and Lingjuan Lyu. 2025. Self-comparison for dataset-level membership inference in large (vision-) language model. In *Proceedings of the ACM on Web Conference 2025*, pages 910–920.
- Yash Savani, Asher Trockman, Zhili Feng, Yixuan Xu, Avi Schwarzschild, Alexander Robey, Marc Finzi, and Zico Kolter. 2025. [Antidistillation sampling](#). In *Advances in Neural Information Processing Systems*, volume 38, pages 117800–117827. Curran Associates, Inc.
- Richard S Sutton, Andrew G Barto, et al. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.
- Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12.
- Jialiang Wang, Xiong Zhou, Xianming Liu, Gangfeng Hu, Deming Zhai, Junjun Jiang, and Haoliang Li. 2026. Variation-bounded loss for noise-tolerant learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 26251–26259.
- Yisen Wang, Xingjun Ma, Zaiyi Chen, Yuan Luo, Jinfeng Yi, and James Bailey. 2019. Symmetric cross entropy for robust learning with noisy labels. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 322–330.
- Haoran Wei, Yaofeng Sun, and Yukun Li. 2025. Deepseek-ocr: Contexts optical compression. *arXiv preprint arXiv:2510.18234*.
- Alexander Wettig, Aatmik Gupta, Saumya Malik, and Danqi Chen. 2024. [QuRating: Selecting high-quality data for training language models](#). In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 52915–52971. PMLR.
- Yongliang Wu, Yizhou Zhou, Zhou Ziheng, Yingzhe Peng, Xinyu Ye, Xinting Hu, Wenbo Zhu, Lu Qi, Ming-Hsuan Yang, and Xu Yang. 2026. [On the generalization of SFT: A reinforcement learning perspective with reward rectification](#). In *The Fourteenth International Conference on Learning Representations*.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.
- Duzhen Zhang, Zhong-Zhi Li, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Xiuyi Chen, Yingying Zhang, et al. 2025a. From system 1 to system 2: A survey of reasoning large language models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- Dylan Zhang, Qirun Dai, and Hao Peng. 2025b. [The best instruction-tuning data are those that fit](#). In *Advances in Neural Information Processing Systems*, volume 38, pages 141172–141208. Curran Associates, Inc.
- Ruichen Zhang, Rana Shahroz, Zhen Tan, Dawei Li, Song Wang, and Tianlong Chen. 2026a. [The quest for efficient reasoning: A data-centric benchmark to cot distillation](#). In *The Fourteenth International Conference on Learning Representations*.
- Ruihan Zhang and Jun Sun. 2026. [Rendering data unlearnable by exploiting LLM alignment mechanisms](#). In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 40587–40598, San Diego, California, United States. Association for Computational Linguistics.
- Tingwei Zhang, John X Morris, and Vitaly Shmatikov. 2026b. How to steal reasoning without reasoning traces. *arXiv preprint arXiv:2603.07267*.
- He Zhu, Junyou Su, Peng Lai, Ren Ma, Wenjia Zhang, Linyi Yang, and Guanhua Chen. 2026a. [Anchored supervised fine-tuning](#). In *The Fourteenth International Conference on Learning Representations*.
- Hongyu Zhu, Sichu Liang, Bofan Chen, Shi-Lin Wang, Zhuosheng Zhang, and Weiping Ding. 2026b. [Auditing partial dataset usage in large language models via fuzzy membership aggregation](#). *IEEE Transactions on Fuzzy Systems*, 34(4):1064–1078.
- Hongyu Zhu, Sichu Liang, Wenwen Wang, Boheng Li, Tongxin Yuan, Fangqi Li, Hanyi Wang, Shi-Lin Wang, and Zhuosheng Zhang. 2025. [Revisiting data auditing in large vision-language models](#). In *Proceedings of the 33rd ACM International Conference on Multimedia*, MM '25, page 11337–11346, New York, NY, USA. Association for Computing Machinery.

A Proof of Proposition 1

Proposition 1 (restated). Consider a deterministic reasoning MDP with $r_t = 0$ for $t < T$, $r_T \in \{0, 1\}$, and $\gamma = 1$. Let $s^+ = f(s, a)$ be the next state after taking action a at state s , and define $p_\pi(s) = \Pr_\pi(r_T = 1 \mid s)$. Then

$$A^\pi(s, a) = p_\pi(s^+) - p_\pi(s). \quad (14)$$

Moreover, for $Q^\pi(s, a), V^\pi(s) > 0$,

$$\begin{aligned} A_{\log}^\pi(s, a) &\triangleq \log Q^\pi(s, a) - \log V^\pi(s) \\ &= i_\pi(r_T = 1; a \mid s). \end{aligned} \quad (15)$$

Proof. For any policy π , the delayed-reward value of a nonterminal state is

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{k=t}^T r_k \mid s_t = s \right].$$

All intermediate rewards are zero and $\gamma = 1$, so the return is exactly r_T . Hence

$$V^\pi(s) = \Pr_\pi(r_T = 1 \mid s_t = s) = p_\pi(s). \quad (16)$$

Similarly,

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{k=t}^T r_k \mid s_t = s, a_t = a \right].$$

Since the transition is deterministic, conditioning on $(s_t = s, a_t = a)$ is equivalent to conditioning on the next state $s_{t+1} = f(s, a) = s^+$. Therefore,

$$Q^\pi(s, a) = \Pr_\pi(r_T = 1 \mid s_{t+1} = s^+) = p_\pi(s^+). \quad (17)$$

Substituting Eqs. (16) and (17) into $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$ proves Eq. (14).

For the log form, let $Y = \{r_T = 1\}$. For $\pi(a \mid s) > 0$,

$$\begin{aligned} i_\pi(Y; a \mid s) &= \log \frac{\Pr_\pi(Y, a \mid s)}{\Pr_\pi(Y \mid s) \Pr_\pi(a \mid s)} \\ &= \log \frac{\Pr_\pi(Y \mid s, a)}{\Pr_\pi(Y \mid s)}. \end{aligned} \quad (18)$$

The numerator is $Q^\pi(s, a)$ and the denominator is $V^\pi(s)$. Thus, when both are positive,

$$\begin{aligned} i_\pi(r_T = 1; a \mid s) &= \log Q^\pi(s, a) - \log V^\pi(s) \\ &= A_{\log}^\pi(s, a), \end{aligned}$$

which proves Eq. (15). The same argument applies to a multi-token segment by treating the segment

as a macro-action whose deterministic transition appends the whole segment to the prefix. If either $Q^\pi(s, a)$ or $V^\pi(s)$ is zero, the probability-progress identity still holds, and the log identity can be evaluated with standard ϵ -smoothing in practical estimators.

B BLADE Method Details

This appendix expands the three-stage BLADE procedure in Section 5. After the protected model completes a trace, BLADE uses Eq. (13) to choose high-value positions, edits only short local contexts around them, and learns which Unicode transformation to apply through a streaming per-character bandit.

B.1 Unicode Transformation Families

For each selected position $t \in \mathcal{I}_\kappa(x)$, BLADE edits an n -gram context g_t rather than the whole trace. The feasible action set contains the three transformation families from Section 5.2. Invisible-controller actions insert non-rendering format characters, such as U+200B, before or after an editable character or token boundary. Homoglyph actions replace a character with an entry from a confusable-character list; for example, Latin U+0043 can be replaced by U+0421, the Cyrillic letter Es, which renders like C but is linguistically closer to English S. Bidi actions wrap a local string with bidirectional controls, such as U+202E and U+202C, so the rendered order can differ from the underlying code-point order. We apply an edited context only if it passes preservation checks: the rendered trace remains close to the original and the extracted answer is unchanged.

B.2 Streaming Bandit Reward and Update

The bandit needs a cheap proxy for how much an edit disrupts the model-visible local context. BLADE samples an editable character c in g_t and chooses one action $a \in \mathcal{A}(c)$. For the perturbed context g_t^a , we compute local embedding drift using only the proxy embedding table E_η :

$$\begin{aligned} e(g) &= \text{Pool}\{E_\eta[v] : v \in \text{Tok}(g)\}, \\ R(g_t, a) &= 1 - \frac{e(g_t)^\top e(g_t^a)}{\|e(g_t)\|_2 \|e(g_t^a)\|_2}. \end{aligned} \quad (19)$$

For each character-action pair (c, a) , BLADE maintains a count $N_{c,a}$ and mean reward $Q_{c,a}$. With $n_c = \sum_{a \in \mathcal{A}(c)} N_{c,a}$, the UCB score and sampling

policy are

$$U_{c,a} = Q_{c,a} + \beta \sqrt{\frac{2 \log(n_c + 1)}{N_{c,a} + 1}}, \quad (20)$$

$$\pi_{\text{BLADE}}(a \mid c) = \frac{\exp(U_{c,a}/\tau)}{\sum_{a' \in \mathcal{A}(c)} \exp(U_{c,a'}/\tau)}.$$

The empirical mean exploits transformations that have caused larger local drift for the same character, while the UCB term keeps exploration so the defense does not collapse into a fixed replacement map. After observing reward R , the online mean update is

$$N_{c,a} \leftarrow N_{c,a} + 1, \quad Q_{c,a} \leftarrow Q_{c,a} + \frac{R - Q_{c,a}}{N_{c,a}}. \quad (21)$$

B.3 BLADE Algorithm

Algorithm 1 BLADE: Bandit-Learned Adaptive Disruption of Encodings

Require: Trace x , extractor ζ , proxy model π_η , top fraction κ , window size n , bandit statistics $\{N_{c,a}, Q_{c,a}\}$.

Ensure: Protected trace $\tilde{x}$.

- 1: Extract proxy endpoint $\hat{y} \leftarrow \zeta(x)$.
- 2: Run π_η once to obtain token states $\{h_t\}_{t=1}^T$ and pooled answer state $h_{\hat{y}}$.
- 3: Compute $m_t = \text{HSIC}(h_t, h_{\hat{y}})$ by Eq. (13).
- 4: Select $\mathcal{I}_\kappa(x) = \text{Top}_{\kappa\%}(\{m_t\}_{t=1}^T)$.
- 5: Initialize $\tilde{x} \leftarrow x$.
- 6: **for** $t \in \mathcal{I}_\kappa(x)$ **do**
- 7: Extract an n -gram context g_t around x_t .
- 8: Randomly select an editable character c in g_t .
- 9: Compute $U_{c,a}$ for $a \in \mathcal{A}(c)$ by Eq. (20).
- 10: Sample $a \sim \pi_{\text{BLADE}}(\cdot \mid c)$.
- 11: Apply a to obtain candidate context g_t^a and candidate trace $\tilde{x}^a$.
- 12: **if** $\tilde{x}^a$ passes preservation checks **then**
- 13: Replace the corresponding context in $\tilde{x}$ by g_t^a .
- 14: Compute reward $R(g_t, a)$ by Eq. (19).
- 15: Update $N_{c,a}$ and $Q_{c,a}$ by Eq. (21).
- 16: **end if**
- 17: **end for**
- 18: **return** $\tilde{x}$.

C Experimental Details and Reproducibility

This appendix provides experimental and reproducibility details omitted from the main text, in-

Data	Level	Strength	Other
GSM8K	ADS-Medium	$\lambda_{\text{ADS}} = 0.040$	$\tau = 0.90, \varepsilon_{\text{fd}} = 0.010$
	ADS-Heavy	$\lambda_{\text{ADS}} = 0.055$	$\tau = 0.90, \varepsilon_{\text{fd}} = 0.010$
	DOGe-Medium	coef = 0.70	$\tau = 0.90$
	DOGe-Heavy	coef = 1.15	$\tau = 0.90$
MATH	ADS-Medium	$\lambda_{\text{ADS}} = 0.060$	$\tau = 0.90, \varepsilon_{\text{fd}} = 0.010$
	ADS-Heavy	$\lambda_{\text{ADS}} = 0.070$	$\tau = 0.90, \varepsilon_{\text{fd}} = 0.010$
	DOGe-Medium	coef = 0.47	$\tau = 0.90$
	DOGe-Heavy	coef = 0.60	$\tau = 0.90$
MMLU	ADS-Medium	$\lambda_{\text{ADS}} = 0.040$	$\tau = 0.90, \varepsilon_{\text{fd}} = 0.010$
	ADS-Heavy	$\lambda_{\text{ADS}} = 0.050$	$\tau = 0.90, \varepsilon_{\text{fd}} = 0.010$
	DOGe-Medium	coef = 0.20	$\tau = 0.90$
	DOGe-Heavy	coef = 0.80	$\tau = 0.90$

Table 3: Sampling parameters for the ADS and DOGe trace sets used in our experiments.

cluding dataset construction, trace generation, fine-tuning, evaluation, proxy models, proprietary-model trace protection, computing budget, software environment, and artifact-use information.

C.1 Datasets, Models, and Evaluation Protocol

For the main experiments in Table 1, we follow the ADS setting and use DeepSeek-R1-Distill-Qwen-7B as the teacher. The student models are Qwen-2.5-1.5B and LLaMA-3.2-3B. We evaluate on GSM8K, MATH, and MMLU. For each dataset, we collect 5,000 teacher-query traces for training and evaluate on 1,000 held-out questions.

For the scaling experiments in Table 2, we use GPT-OSS-20B as the teacher and Qwen-3-4B, Qwen-3.5-2B, Ministral-3-8B, and OLMo-3-7B as students. We evaluate on AIME24, AIME25, and GPQA-Diamond. Training queries are sampled from NuminaMath-1.5 and SuperGPQA and are deduplicated against all evaluation sets. We report ACC@8 for AIME-style benchmarks and ACC@3 otherwise. Unless otherwise stated, the “ $\pm$ ” values in tables and figures denote the standard deviation over repeated runs.

C.2 Trace Generation Settings

Table 3 summarizes the ADS and DOGe sampling parameters used to generate the Medium and Heavy trace sets.

C.3 Student Fine-tuning and Evaluation

For the Qwen-2.5-1.5B and LLaMA-3.2-3B students in Table 1, all LoRA fine-tuning runs use 3 epochs, learning rate 5×10^{-4} , and LoRA rank and scale $r = \alpha_{\text{LoRA}} = 128$. For the harder AIME and GPQA-Diamond experiments in Table 2, we train for 5 epochs with learning rate 2×10^{-4} and

use $r = \alpha_{\text{LoRA}} = 256$. Unless otherwise stated, method-specific baselines use their original recommended hyperparameters. For all evaluation generations, we use temperature 0.6 and top- $p = 0.95$.

C.4 SIFT Hyperparameters

For the importance-sampling factor in Eq. (9), we set $\alpha_{\text{IS}} = 0.05$ for Table 1 and $\alpha_{\text{IS}} = 0.1$ for Table 2. For the credit smoothing in Eq. (12), we set the weight of non-positive-advantage segments to zero so the disruptive artifacts are ignored rather than allowed to suppress neighboring useful tokens inside the same segment. We set $\alpha_A^+ = 0.1$ by default. For tail truncation, we use $\tau_A = 10^{-3}$ and apply truncation only when the low-progress tail also shows a repeated-pattern signature measured by compressed entropy. This avoids removing fluent low-progress text too aggressively.

C.5 Proxy Models for Trace Generation and Scoring

ADS and DOGe require a proxy model during trace generation. For the datasets in Table 1, we use Qwen-2.5-3B as the proxy. For Table 2, there is no substantially smaller model with the same tokenizer as GPT-OSS-20B. We therefore construct a proxy by pruning GPT-OSS-20B to keep four experts per layer, yielding a dense-MoE proxy of about 4B parameters, and then fine-tune it on 30K traces generated by GPT-OSS-120B to restore reasoning ability. This proxy reaches roughly 70% accuracy on GSM8K, which we found sufficient for ADS generation.

For SIFT’s answer-confidence estimator, we use smaller frozen models to reduce cost: Qwen-3.5-0.8B for the Table 1 benchmark suite and Qwen-3.5-2B for the harder Table 2 suite. BLADE uses Qwen-3-0.6B as its proxy for token-value estimation and local embedding-drift rewards, intentionally avoiding overlap with the ADS/DOGe and SIFT proxy models.

C.6 GPT-5.5 Trace-Protection Protocol

For the GPT-5.5 trace-protection experiment, we use reasoning effort xhigh. Because GPT-5.5 does not expose its internal reasoning process, we build protected traces in two stages. First, GPT-5.5 produces a final answer. Second, the student model, Qwen-3.5-9B, generates a candidate reasoning segment. We then send both the candidate reasoning segment and the GPT-5.5 final answer back to GPT-5.5, asking it to polish the reasoning without

shortening the thinking segment and to make the final reasoning point to the GPT-5.5 answer. This procedure yields thinking segments roughly 5–10 times longer than the original final answer while improving the student’s reasoning performance when distilled.

C.7 Computing Infrastructure, Software, and Artifact Use

Software environment. Our main software environment used Python 3.10.13, with the core packages listed below:

Package	Version
PyTorch	2.9.1
PyTorch-CUDA	12.4
Transformers	5.0.0rc0
PEFT	0.12.0
vLLM	0.15.1
OpenAI SDK	2.28.0

Artifact use and licensing. We use public benchmarks, training-query sources, open-weight model checkpoints, baseline implementations, and commercial API outputs only for research and evaluation. We cite the original creators of the datasets, models, and baselines used in this work. We follow the corresponding licenses, model cards, dataset cards, and provider terms of use, and do not redistribute proprietary model checkpoints, proprietary API outputs, or third-party artifacts when redistribution is not permitted. Any released derivative artifacts, including generated traces and human annotations, will be released only for research purposes, when compatible with the original access conditions, and after removing participant identifiers.

D Human Evaluation Details

We conducted a human evaluation on Prolific³ to assess whether protected traces remained useful and readable to human readers. Participants compared three AI-generated reasoning responses for grade-school math problems and ranked them from highest to lowest quality. The three candidate responses corresponded to different trace-generation methods, but participants were not told the method identities.

D.1 Task Design and Interface

Each questionnaire contained at most 20 items sampled from a pool of 100 math questions. For each

³<https://www.prolific.com/>

item, the three candidate responses were randomly ordered and displayed under the labels A, B, and C. The mapping from labels to generation methods was randomized per item and hidden from participants during annotation. Participants were asked to judge the overall quality of the responses, considering correctness, clarity, readability, and usefulness of the reasoning process.

The task was administered through Prolific’s live interface. Figure 10 shows the participant-facing ranking interface.

D.2 Participant Instructions

Before annotation, participants were shown the task goal, quality criteria, research purpose, data-use and privacy notice, risk notice, voluntary participation statement, and per-item ranking steps. The full participant-facing instruction text is reproduced in Figure 11. Participants were instructed to rank all three responses for each item and were told that the study concerned evaluation of AI-generated reasoning responses.

D.3 Recruitment, Payment, Consent, and Privacy

Participants were recruited through Prolific. The study used Prolific’s standard participant-management and payment workflow. Prolific IDs were used only for payment, quality control, and study administration. The exported annotation data used for analysis exclude personal identifiers from public reporting and release.

The task was designed as a short paid annotation study. Each participant answered at most 20 ranking questions. The payment was set to be equivalent to US\$10 for an estimated 20-minute task, and the total project cost including Prolific platform fees was approximately US\$200. This corresponds to an estimated hourly rate of US\$30 before platform fees. Prolific’s stated minimum hourly reward rate is US\$8/GBP 6 per hour,⁴ so the study payment was above the platform minimum and was intended to be adequate for the participant pool available through the platform.

Consent was obtained through the Prolific study flow. Before starting the task, participants were told the research purpose, how their annotations would be used in aggregate, that no personal information was requested, that their Prolific ID would be used

only for payment, quality control, and administration, and that participation was voluntary. Participants could return the task at any time through Prolific.

D.4 Ethics Review for Human Evaluation

We treated the study as a low-risk annotation task because participants only ranked AI-generated answers to elementary-level math problems. The task did not involve sensitive topics, deception, intervention, or collection of private personal information. Participants were informed that the collected annotations would be analyzed only in aggregate.

D.5 Example Annotated Item

The case-study panels below show one complete annotated item from the Prolific task. Figure 12 gives the question, the hidden response-to-method mapping, the ranking options, and the anonymous annotator rankings. Figures 13 and 14 show Responses A and B as normally displayed to annotators, corresponding to ADS (heavy) and DOGe (heavy), respectively. For BLADE, Figure 15 shows Response C as normally rendered, while Figure 16 shows the same response with invisible Unicode control characters explicitly marked as code-point tags, e.g., <U+202E>.

E Additional Results

E.1 Full Comparison and Ablation Results

This section provides the complete comparison and ablation results across all models, datasets, and trace subsets considered in our experiments in Figure 17 and 18. These additional results are consistent with the findings in the main paper: SIFT consistently improves over vanilla MLE and strong supervised fine-tuning baselines, with especially clear gains on anti-distillation traces.

E.2 Robustness of BLADE to Adaptive Attacks

BLADE perturbs the underlying code-point sequence of a reasoning trace while preserving its human-visible rendering. An adaptive distiller, however, may attempt to recover a cleaner training signal before distillation. We therefore evaluate four classes of adaptive attacks: Unicode normalization, text repair, stronger distillation objectives, and OCR-based reconstruction. These experiments test whether knowledge of BLADE’s mechanism

⁴<https://researcher-help.prolific.com/en/article/2273bd>

The screenshot displays a user interface for a ranking task. On the left, there is a form with the following fields:

- "id"**: A text input field containing the number "61".
- Question**: A text input field containing the text "Laran has started a poster business. She is selling".
- Response A**: A text input field containing the text "To determine Laran's profit per school week, I need".
- Response B**: A text input field containing the text "First, I need to determine Laran's daily profit from".
- Response C**: A text input field containing the text "tanks, first, let's calculate the revenue from larae".

Each text input field has a small icon with four arrows pointing outwards, indicating it is a text area. On the right side of the interface, the text "Question 1 of 20" is displayed at the top. Below it, the instruction "Rank the three responses from highest quality to lowest quality." is shown. There are six radio button options for ranking:

- ☐ A > B > C
- ☐ A > C > B
- ☐ B > A > C
- ☐ B > C > A
- ☐ C > A > B
- ☐ C > B > A

At the bottom right, there is a purple button labeled "Next" and a blue circular icon with a white speech bubble.

Figure 10: **Participant-facing ranking interface.** Screenshot of the Prolific live interface used to present each math question, the three response options, and the ranking choices.

provides a simple inverse that restores the learning value of protected traces.

Unicode normalization and text repair.

Unicode normalization maps canonically or compatibility-equivalent character sequences to normalized representations.⁵ Most BLADE transformations are not removed by such equivalences: invisible format characters remain explicit code points, BiDi controls specify rendering order, and many cross-script homoglyphs are distinct characters without canonical mappings to their visual counterparts. We apply all four standard normalization forms (NFC, NFD, NFKC, and NFKD) using Python’s unicodedata implementation.

We further test increasingly flexible repair pipelines: STRIP, which removes Unicode format characters; pypellchecker⁶; DeCoGLM, a state-of-the-art grammatical-error-correction model (Li and Wang, 2024); and Grammarly⁷, a commercial text-correction system. Each preprocessor is applied to BLADE-protected traces before student distillation. Table 4 reports test accuracy (mean $\pm$ standard deviation).

Normalization yields only modest changes and remains far below the pre-distillation accuracy in every setting. Text repair is stronger in several cases, particularly control-character stripping on MMLU, but still fails to produce a net improvement over the corresponding pre-distillation student. This is expected because stripping controls

cannot reverse homoglyph substitutions or reconstruct code-point sequences that were reordered before the BiDi controls were removed. Likewise, generic spelling or grammatical repair is not designed to infer the original characters and ordering under BLADE’s diverse, context-dependent perturbations.

Stronger distillation objectives. An adaptive adversary may also leave the protected traces unchanged but use a stronger learning algorithm. We therefore apply the SFT and data-selection methods DFT (Wu et al., 2026), ASFT (Zhu et al., 2026a), ProFit (Liu et al., 2026b), and QuRating (Wettig et al., 2024) directly to BLADE-protected traces. Table 5 shows that these methods occasionally recover part of the performance lost under vanilla MLE, but none restores the student to its pre-distillation performance.

OCR reconstruction. A stronger adaptive attack exploits BLADE’s rendering preservation directly: the attacker can render each protected trace and transcribe the resulting image back into text. We evaluate this attack by rendering each trace as a 1489×2105 PNG image and reconstructing it with DeepSeek-OCR (Wei et al., 2025). We then train Qwen-2.5-1.5B on the reconstructed text using the same distillation pipeline.

OCR preprocessing itself required 7566.2, 13394.2, and 5849.8 seconds on GSM8K, MATH, and MMLU, respectively, compared with 1428, 2604, and 1904 seconds for the subsequent student training. Thus, in these experiments, OCR pre-

⁵<https://www.unicode.org/reports/tr15/>

⁶<https://pypi.org/project/pypellchecker/>

⁷<https://app.grammarly.com/>

Participant Instructions Shown Before Annotation

Overview. In this task, you will read grade-school-level math word problems and three reasoning responses generated by large language models. Your goal is to rank the three responses from highest quality to lowest quality for a human reader.

What Makes a High-Quality Response? A high-quality response should be correct, clear, sufficiently detailed, readable, and clean.

Correct means that the response solves the problem and gives the right final answer. Clear means that the reasoning is easy to follow. Sufficiently detailed means that the response explains the important reasoning steps instead of jumping directly to the final answer. Readable means that the text is understandable and reasonably well organized. Clean means that the response avoids irrelevant text, excessive repetition, strange formatting, corrupted or confusing artifacts.

Some responses may contain mistakes, unnecessary repetition, odd symbols, or unusual formatting. Please judge each response as it appears to you.

Research Purpose. This study is for academic research on the quality of AI-generated reasoning process. Your responses will be used to compare different ways of generating or modifying model reasoning traces. We will analyze the collected annotations in aggregate.

Data Use and Privacy. Your Prolific ID may be used only for payment, quality control, and study administration. It will not be included in any public report or released dataset. We do not ask you to provide any personal information in this task.

Please do not include any personal information about yourself or anyone else in any free-text response.

Risks and Content Notice. The task involves reading AI-generated answers to elementary-level math problems. The expected risk is minimal. Some responses may contain confusing text, corrupted characters, or formatting artifacts, but the content is not intended to be offensive or sensitive.

Voluntary Participation. Participation is voluntary. You may stop the task at any time by returning it on Prolific.

Task Goal. For each item, read the math question and the three responses, then rank Response A, Response B, and Response C from highest quality to lowest quality. You may also provide one brief sentence explaining your ranking.

Task steps. For each item:

1. Read the math question.
2. Read Response A, Response B, and Response C.
3. Rank the three responses from highest quality to lowest quality. When ranking, consider whether the final answer is correct, whether the reasoning is logically sound, whether the explanation is clear and complete, whether the response is readable and clean, and whether there are distracting artifacts, corrupted text, or excessive repetition.
4. If all responses are imperfect, still rank them by relative quality: choose the ones that are more useful and understandable for a human reader.
5. Optionally, write one brief sentence explaining your ranking. Please keep this explanation concise and do not include personal information.

Figure 11: **Participant instructions.** Full instruction text shown to annotators before the ranking task.

Example Item Context: GSM8K Item 007

Question. Claudia has 122 ounces of water and is filling up cups. She has 8-ounce glasses, 5-ounce glasses, and 4-ounce glasses. If she fills six 5 ounce glasses and four 8 ounce glasses, how many 4 ounce glasses can she fill with the remaining water?

Ranking options shown to annotators.

A > B > C	A > C > B	B > A > C
B > C > A	C > A > B	C > B > A

Response-to-method mapping (invisible to annotators). Response A = ADS (heavy); Response B = DOGe (heavy); Response C = BLADE (ours).

Anonymous annotator rankings.

Annotator 1	C > B > A; no free-text explanation.
Annotator 2	C > B > A; “It’s muc better structured.”
Annotator 3	C > B > A; no free-text explanation.

Figure 12: **Case Study: GSM8K Item 007.** Question and anonymous annotator rankings.

Response A: ADS (heavy), normal display

<think> Claudia has a total of 122 ounces of water. crawledClaudia fills six 5-ounce glasses, which uses 6 * punvangst ounces. This leaves 122 - punvangst ounces remaining. crawled Claudia also fills four 8-ounce glasses, using another pun.InputStreamReader ounces. Subtracting this amount from the previous remaining ounces gives the amount of water left after filling the glasses. crawled Claudia can then determine how many 4-ounce glasses she can fill with the remaining water by dividing the remaining ounces by pun. Claudia can fill pun glasses with the remaining water. 泉水 </think> Craaaa Cla

Figure 13: **Case Study: GSM8K Item 007.** Response A: ADS (heavy), normal display.

Response B: DOGe (heavy), normal display

<think>
Claudia has a total of 122 ounces of water.

She fills six 5-ounce glasses, which uses 6 multiplied by 5 ounces, totaling 30 ounces.

Next, she fills four 8-ounce glasses, using 4 multiplied by 8 ounces, totaling 32 ounces.

Adding these together, 30 ounces plus 32 ounces equals 62 ounces used so far.

Subtracting this from the initial 122 ounces leaves 60 ounces remaining.

Finally, dividing the remaining 60 ounces by the capacity of each 4-ounce glass, which is 4 ounces, gives her能力她可以装满的4盎司杯子的数量。

</think>

Let's solve the problem step by step.

Given:

- Total water available: 122 ounces
- Sizes of glasses:
 - 8-ounce glasses
 - 5-ounce glasses
 - 4-ounce glasses

Actions taken:

1. Filled six 5-ounce glasses:
$$6 \times 5 \text{ ounces/glass} = 30 \text{ ounces}$$
2. Filled four 8-ounce glasses:
$$4 \times 8 \text{ ounces/glass} = 32 \text{ ounces}$$

Total water used so far:

$$30 \text{ ounces} + 32 \text{ ounces} = 62 \text{ ounces}$$

Remaining water:

$$122 \text{ ounces} - 62 \text{ ounces} = 60 \text{ ounces}$$

To find out how many 4-ounce glasses can be filled with the remaining water:

$$\frac{60 \text{ ounces}}{4 \text{ ounces/glass}} = 15 \text{ glasses}$$

Final Answer:

$$\boxed{15 \text{ glasses}}$$

Figure 14: Case Study: GSM8K Item 007. Response B: DOGe (heavy), normal display.

Response C: BLADE (ours), normal display

```

<think>
Claudia has a total of 122 ounces of water.

First, I need to calculate the amount of water used to fill the six 5-ounce glasses. Multiplying 6 by 5 gives 30 ounces.

Next, I'll determine the water used for the four 8-ounce glasses by multiplying 4 by 8, which equals 32 ounces.

Adding these together, 30 ounces plus 32 ounces equals 62 ounces used so far.

Subtracting this from the total 122 ounces minus 62 ounces leaves 60 ounces remaining.

Finally, to find out how many 4-ounce glasses can be filled with the remaining 60 ounces, I divide 60 by 4, resulting in 15 glasses.
</think>

Let's solve the problem step by step.

**Given:**
- **Total water available:** 122 ounces
- Types of glasses:
  - 6 5-ounce glasses
  - 4 8-ounce glasses

**Actions Taken:**
1. Filled six 5-ounce glasses:

$$6 \times 5 = 30 \text{ ounces}$$

2. Filled four 8-ounce glasses:

$$4 \times 8 = 32 \text{ ounces}$$

3. Total water used:

$$30 + 32 = 62 \text{ ounces}$$

4. Remaining water:

$$122 - 62 = 60 \text{ ounces}$$

5. Number of 4-ounce glasses that can be filled with the remaining water:

$$\frac{60}{4} = 15 \text{ glasses}$$


**Final Answer:**

$$15$$


```

Figure 15: Case Study: GSM8K Item 007. Response C: BLADE (ours), normal display.

processing alone costs roughly $3\text{--}5\times$ the student-training time.

As shown in Table 6, OCR is substantially stronger than generic preprocessing. Nevertheless, it does not recover clean-trace distillation: on GSM8K and MATH the resulting student remains below its pre-distillation checkpoint, while on MMLU it recovers only part of the clean distillation gain. We observe residual OCR errors particularly in mathematical expressions and structured formatting, which introduce additional noise into the reconstructed demonstrations.

Taken together, these experiments show that Unicode normalization, generic text repair, stronger distillation objectives, and OCR reconstruction do not provide a reliable low-cost inverse of BLADE.

E.3 Protection of Flagship-Model Traces

The GPT-5.5 experiment in Section 6 uses reconstructed reasoning traces because GPT-5.5 does not expose its complete native reasoning process. As detailed in Appendix C.6, we first obtain the GPT-5.5 final answer, let Qwen-3.5-9B generate a candidate reasoning segment, and then ask GPT-5.5 to correct and refine that reasoning while preserving

its original final answer.

To complement this reconstructed-trace setting, we additionally evaluate BLADE using complete reasoning traces directly exposed by **DeepSeek-V4**, with reasoning effort set to high. We use Qwen-3.5-9B as the student and evaluate on AIME24, AIME25, and GPQA-Diamond (Table 7).

Standard distillation from the directly exposed DeepSeek-V4 traces improves the student on all three benchmarks, confirming that these traces provide substantial learnable reasoning signal. In contrast, training on BLADE-protected traces reduces student accuracy well below the pre-distillation checkpoint in every setting. Together with the reconstructed GPT-5.5 experiment in Section 6, these results provide complementary evidence that BLADE remains effective when scaling from smaller open models to flagship-level reasoning traces.

F Extended Background

LLM training-data auditing. Traditional approaches to protecting against unauthorized LLM training have largely focused on auditing whether

Response C: BLADE (ours), invisible characters marked

```
think(
  Claudia has<U+2067><U+20E2><U+2066><U+2067><U+20E2>fo
    <U+20C2><U+2069><U+2069><U+2066><U+2067><U+20E2><U+2066><U+2067><U+20E2>2<U+20C2><U+2069><U+20
69><U+2066><U+2067><U+20E2><U+2066><U+2067><U+20E2><U+2066><U+2067><U+20E2>2<U+20C2><U+2069><U+2066><U+2067><U+20E2>1<U+20C2><U+2069><U+2069><U+2066><U+2067><U+20E2>a
    <U+20C2><U+2069><U+2069><U+2066><U+2067><U+20E2><U+2066><U+2067><U+20E2>1atot
    <U+20C2><U+2069><U+2069><U+2066><U+2067><U+20E2>secnuo
    <U+20C2><U+2069><U+2069><U+20C2><U+2069> water.
  )
```

First, I need to calculate the amount of water used to fill the six 5-ounce glasses. Multiplying $\langle U+2067 \rangle \langle U+202E \rangle \langle U+2066 \rangle \langle U+2067 \rangle \langle U+202E \rangle$ secnuo $\langle U+202C \rangle \langle U+2069 \rangle \langle U+2069 \rangle \langle U+2066 \rangle \langle U+2067 \rangle \langle U+202E \rangle$ 30 $\langle U+202C \rangle \langle U+2069 \rangle \langle U+2069 \rangle \langle U+2066 \rangle \langle U+2067 \rangle \langle U+202E \rangle$ 3 $\langle U+202C \rangle \langle U+2069 \rangle \langle U+2069 \rangle \langle U+2066 \rangle \langle U+2067 \rangle \langle U+202E \rangle$ sevig $\langle U+202C \rangle \langle U+2069 \rangle \langle U+2069 \rangle \langle U+2066 \rangle \langle U+2067 \rangle \langle U+202E \rangle$ 5 $\langle U+202C \rangle \langle U+2069 \rangle \langle U+2069 \rangle \langle U+2066 \rangle \langle U+2067 \rangle \langle U+202E \rangle$ $\langle U+202C \rangle \langle U+2069 \rangle \langle U+2069 \rangle \langle U+2066 \rangle \langle U+2067 \rangle \langle U+202E \rangle$ 6 $\langle U+202C \rangle \langle U+2069 \rangle \langle U+2069 \rangle \langle U+2066 \rangle \langle U+2067 \rangle \langle U+202E \rangle$ yb

Next, I'll determine the water used for the four 8-ounce glasses by multiplying 4 by 8, which equals 32 ounces.

Add<U+2067><U+202E><U+2066><U+2067><U+202E> <U+202C><U+2069><U+2069><U+2066><U+2067><U+202E>sulp
 <U+202C><U+2069><U+2069><U+2066><U+2067><U+202E>secnuo
 <U+202C><U+2069><U+2069><U+2066><U+2067><U+202E>8<U+202C><U+2069><U+2069><U+2066><U+2067><U+202E>3<U+202C><U+2069><U+2069><U+2066><U+2067><U+202E>
 <U+202C><U+2069><U+2069><U+2066><U+2067><U+202E>, <U+202C><U+2069><U+2069><U+2066><U+2067><U+202E>rehtegot
 <U+202C><U+2069><U+2069><U+2066><U+2067><U+202E>esent <U+202C><U+2069><U+2069><U+202C><U+2069>32 ounces equals 62 ounces used so far.

[illegible]

Finally, to find out how many 4-ounce glasses can be filled

$\frac{60}{4} = 15$

divide 60 by 4, resulting in 15 glasses.

think

Let's solve the problem step by step.

```
**Given:**
<U+2067><U+202E>secnuo 221** :elbaliava retaw latoT <U+202C><U+2069>***
- **8-ounce glasses**
- **5-ounce glasses**
- **4-ounce glasses**
```

```

**Actions Taken:**
1. 11/22/2014 - 11/22/2014

```

```
1. **Filled six 5-ounce glasses:**
\
<U+2067><U+202E>\t\t\t 6<U+202C><U+2069><U+2066><{<U+2069><U+2067><U+202E>sessaig <U+202C><U+2069><U+2066><}<U+2069><U+2067><U+202E>\t\t\t 5 semit\
<U+202C><U+2069><U+2066><}<U+2069><U+2067><U+202E>ssaig/secnuo <U+202C><U+2069><U+2066><}<U+2069><U+2067><U+202E>\t\t\t 03 =
<U+202C><U+2069><U+2066><}<U+2069><U+2067><U+202E>secnuo <U+202C><U+2069><U+2066><}<U+2069><U+2067><U+202E>
<U+202C><U+2069>
```

2. **Filled four 8-ounce glasses:**
 \[
 <U+2067><U+202E>\t\t\t 4<U+202C><U+2069><U+2066><U+2069><U+2067><U+202E>\text{essalag } <U+202C><U+2069><U+2066><U+2069><U+2067><U+202E>\t\t\t 8 \text{ semit }
 <U+202C><U+2069><U+2066><U+2069><U+2067><U+202E>\text{ssalag/secnuo } <U+202C><U+2069><U+2066><U+2069><U+2067><U+202E>\t\t\t 23 =
 <U+202C><U+2069><U+2066><U+2069><U+2067><U+202E>\text{secnuo } <U+202C><U+2069><U+2066><U+2069><U+2067><U+202E>
 <U+202C><U+2069>

3. **Total water used:**

$$\begin{aligned} & \langle U+2067 \rangle \langle U+202E \rangle \text{t}\langle U+03 \rangle \langle U+202C \rangle \langle U+2069 \rangle \langle U+2066 \rangle \langle U+2069 \rangle \langle U+2067 \rangle \langle U+202E \rangle \text{secnuo} \langle U+202C \rangle \langle U+2069 \rangle \langle U+2066 \rangle \langle U+2069 \rangle \langle U+2067 \rangle \langle U+202E \rangle \text{t}\langle U+23 \rangle + \\ & \langle U+202C \rangle \langle U+2069 \rangle \langle U+2066 \rangle \langle U+2069 \rangle \langle U+2067 \rangle \langle U+202E \rangle \text{secnuo} \langle U+202C \rangle \langle U+2069 \rangle \langle U+2066 \rangle \langle U+2069 \rangle \langle U+2067 \rangle \langle U+202E \rangle \text{t}\langle U+26 \rangle = \\ & \langle U+202C \rangle \langle U+2069 \rangle \langle U+2066 \rangle \langle U+2069 \rangle \langle U+2067 \rangle \langle U+202E \rangle \text{secnuo} \langle U+202C \rangle \langle U+2069 \rangle \langle U+2066 \rangle \langle U+2069 \rangle \langle U+2067 \rangle \langle U+202E \rangle \\ & \langle U+202C \rangle \langle U+2069 \rangle \end{aligned}$$
[illegible]

```
5. **Number of 4-ounce glasses that can be filled with the remaining water:**
```

[illegible]

****Final Answer:****

$$\boxed{5}$$

Figure 16: **Case Study: GSM8K Item 007.** Response C: BLADE (ours), with invisible Unicode control characters explicitly marked.

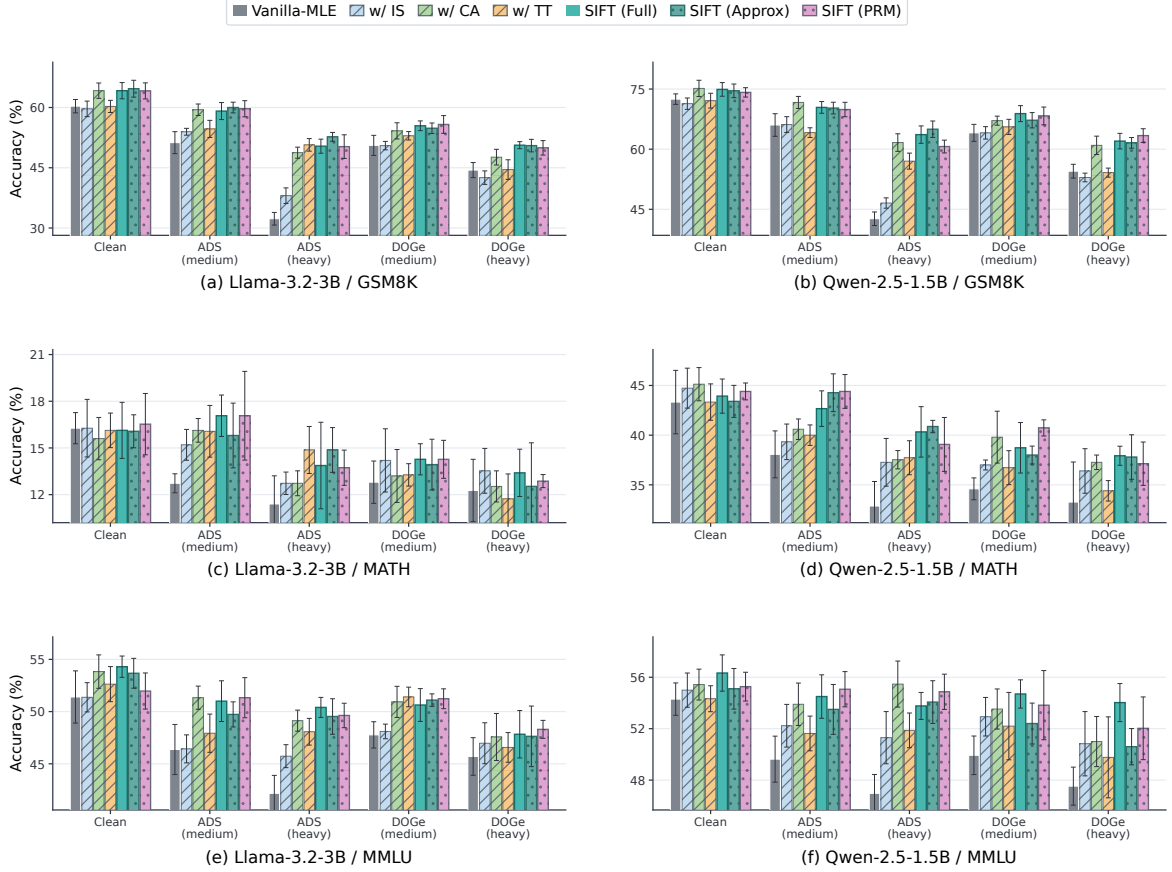


Figure 17: Full SIFT ablations across datasets, students, and perturbation strengths.

copyrighted or private content has been used for model training. Representative methods infer training-data usage by testing whether a model exhibits detectable memorization of or influence from protected examples (Maini et al., 2024; Ren et al., 2025). These methods are well suited to static corpora with fixed and auditable source texts (Zhu et al., 2025, 2026b). However, reasoning traces are generated on demand by a deployed model rather than drawn from a pre-existing text corpus. Conventional static-corpus auditing therefore does not naturally address the protection of dynamically generated reasoning signals.

Anti-distillation of LLM-generated outputs. A more recent line of work instead seeks to proactively reduce the *learnability* of returned outputs while preserving their utility for legitimate users. Antidistillation Sampling (ADS) (Savani et al., 2025) and DOGe (Li et al., 2025) are representative approaches in this direction: both perturb model outputs so that they remain useful to humans while becoming harder for student models to imitate. Other related settings differ from reasoning-

trace protection in important ways. Unlearnable examples protect question-answer data by rewriting input questions (Zhang and Sun, 2026); however, in reasoning distillation, an attacker can freely choose the queries and only needs the model-generated traces. Our work therefore targets the more realistic trace-only setting, where the defender must preserve readable reasoning for benign users while suppressing the model-visible signals that make those traces learnable.

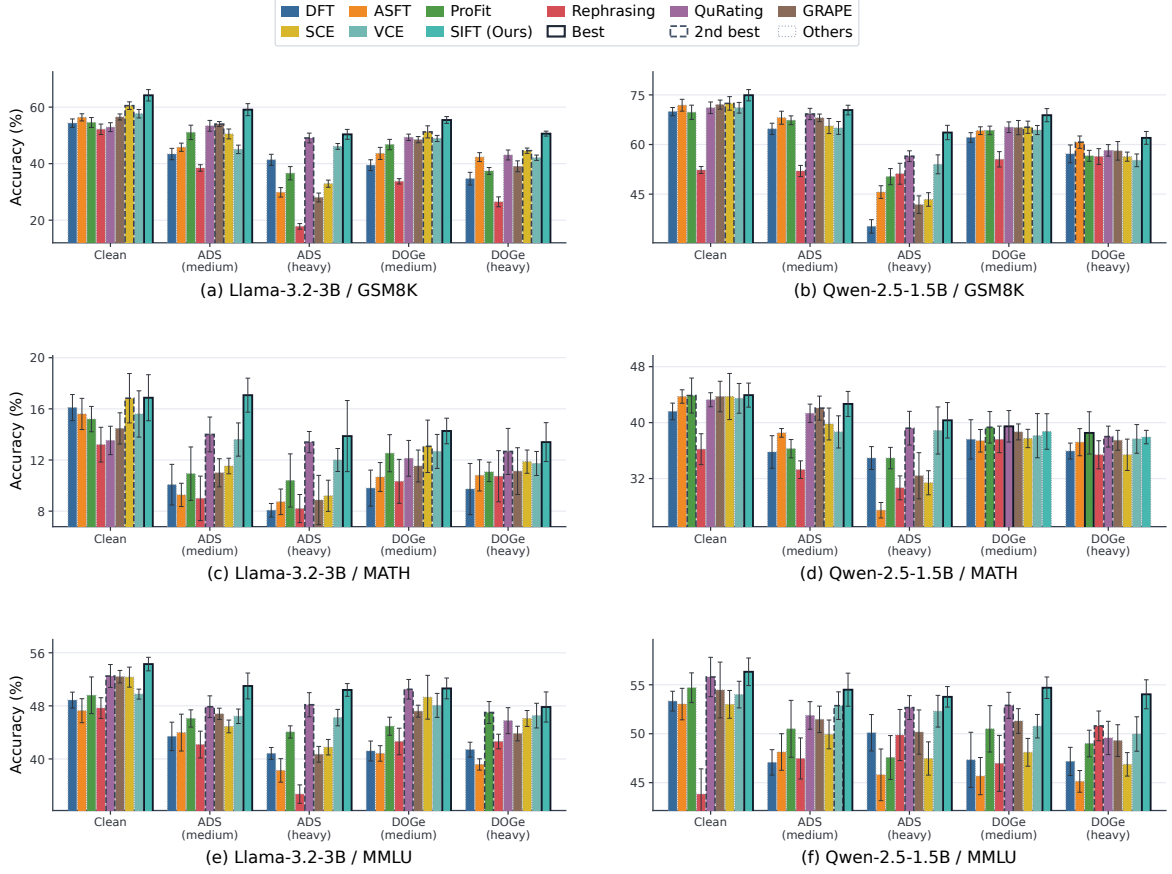


Figure 18: Full comparison with supervised fine-tuning baselines.

(a) Unicode normalization							
Dataset	Student	Pre-dist.	BLADE	NFC	NFD	NFKC	NFKD
GSM8K	LLaMA-3.2-3B	31.4	6.42 ± 0.50	6.70 ± 0.86	7.39 ± 0.49	8.02 ± 0.78	7.44 ± 0.86
GSM8K	Qwen-2.5-1.5B	57.5	7.50 ± 0.88	7.55 ± 0.44	7.44 ± 0.17	9.24 ± 0.63	8.99 ± 0.86
MATH	LLaMA-3.2-3B	10.4	3.73 ± 0.68	3.13 ± 0.19	2.47 ± 0.84	3.33 ± 1.13	3.27 ± 1.07
MATH	Qwen-2.5-1.5B	34.9	6.00 ± 1.51	6.13 ± 1.37	5.67 ± 1.16	7.47 ± 0.57	8.00 ± 0.40
MMLU	LLaMA-3.2-3B	41.9	0.40 ± 0.46	0.00 ± 0.00	0.07 ± 0.12	0.00 ± 0.00	0.17 ± 0.21
MMLU	Qwen-2.5-1.5B	49.4	1.40 ± 0.36	2.43 ± 0.81	0.60 ± 0.20	0.13 ± 0.12	0.20 ± 0.20

(b) Text repair							
Dataset	Student	Pre-dist.	BLADE	Strip	Spell	DeCoGLM	Grammarly
GSM8K	LLaMA-3.2-3B	31.4	6.42 ± 0.50	12.02 ± 0.71	6.59 ± 1.05	11.66 ± 0.71	8.00 ± 1.07
GSM8K	Qwen-2.5-1.5B	57.5	7.50 ± 0.88	12.19 ± 0.54	7.78 ± 0.35	11.97 ± 1.11	17.09 ± 0.44
MATH	LLaMA-3.2-3B	10.4	3.73 ± 0.68	5.27 ± 1.01	2.60 ± 0.57	5.60 ± 0.98	3.80 ± 1.08
MATH	Qwen-2.5-1.5B	34.9	6.00 ± 1.51	10.60 ± 1.01	7.87 ± 1.05	11.87 ± 0.95	8.73 ± 1.01
MMLU	LLaMA-3.2-3B	41.9	0.40 ± 0.46	38.43 ± 1.68	0.30 ± 0.36	38.37 ± 1.85	15.37 ± 2.45
MMLU	Qwen-2.5-1.5B	49.4	1.40 ± 0.36	47.40 ± 0.53	0.00 ± 0.00	46.57 ± 1.17	45.90 ± 1.17

Table 4: **Adaptive preprocessing attacks against BLADE.** Student accuracy (%). (a) All four standard Unicode normalization forms; (b) control-character stripping and increasingly semantic text-repair methods. Although some preprocessing partially mitigates BLADE, all resulting models remain below their corresponding pre-distillation checkpoints.

Dataset	Student	Pre-dist.	Vanilla MLE	DFT	ASFT	ProFit	QuRating
GSM8K	LLaMA-3.2-3B	31.4	6.42 ± 0.50	6.70 ± 0.71	6.81 ± 0.54	8.99 ± 1.18	6.18 ± 1.11
GSM8K	Qwen-2.5-1.5B	57.5	7.50 ± 0.88	8.77 ± 0.31	8.35 ± 0.89	11.80 ± 0.94	7.17 ± 0.69
MATH	LLaMA-3.2-3B	10.4	3.73 ± 0.68	2.87 ± 1.16	3.73 ± 0.12	3.13 ± 0.42	3.40 ± 0.72
MATH	Qwen-2.5-1.5B	34.9	6.00 ± 1.51	6.27 ± 0.90	8.47 ± 1.23	5.80 ± 1.06	6.73 ± 0.25
MMLU	LLaMA-3.2-3B	41.9	0.40 ± 0.46	0.77 ± 0.25	0.00 ± 0.00	0.10 ± 0.10	0.30 ± 0.20
MMLU	Qwen-2.5-1.5B	49.4	1.40 ± 0.36	0.37 ± 0.32	4.13 ± 0.51	7.10 ± 0.87	30.87 ± 0.60

Table 5: **Stronger distillation attacks on BLADE-protected traces.** Student accuracy (%). Improvements in loss formulation, token weighting, or data selection do not restore the learning signal to the pre-distillation level.

Dataset	Pre-dist.	Clean MLE	BLADE	OCR
GSM8K	57.5	72.48 $\pm$ 0.31	7.50 $\pm$ 0.88	55.31 $\pm$ 1.89
MATH	34.9	43.33 $\pm$ 2.69	6.00 $\pm$ 1.51	28.00 $\pm$ 1.83
MMLU	49.4	54.30 $\pm$ 0.46	1.40 $\pm$ 0.36	52.40 $\pm$ 1.31

Table 6: **OCR reconstruction attack.** Accuracy (%) of Qwen-2.5-1.5B after distillation. OCR substantially weakens raw BLADE perturbations, but does not recover the performance obtained from clean traces.

Dataset	DeepSeek-V4	Pre-dist.	Distillation	BLADE
AIME24	90.83	57.50	70.83	30.83
AIME25	89.17	47.50	61.17	28.33
GPQA	82.32	67.17	71.21	35.86

Table 7: **BLADE on directly exposed DeepSeek-V4 reasoning traces.** Accuracy (%). Standard distillation improves Qwen-3.5-9B over its pre-distillation checkpoint, whereas BLADE-protected traces reduce performance well below the checkpoint on all three benchmarks.