RETHINKING ANTI-DISTILLATION OF LARGE LANGUAGE MODELS: AN UNLEARNABLE EXAMPLES PERSPECTIVE

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) expose valuable capabilities through their generated responses, but making them vulnerable to unauthorized distillation. Existing anti-distillation methods largely assume that knowledge transfer relies on fitting to teacher-generated responses and therefore seek to make those responses difficult to learn. We revisit this premise and ask: *Can a student easily fit protected responses yet fail to acquire the generalizable capability behind them?* Inspired by unlearnable examples (UEs), we decouple data fitting from knowledge transfer by injecting easy-to-fit perturbations into the teacher's reasoning, encouraging the student to learn the surface shortcut while neglecting the underlying capability. Building on this principle, we propose DISTILSHIELD, which protects LLMs across the distillation trajectory: it leverages an *Unlearnable Converter* that implants implicit shortcuts into teacher reasoning, steers the learning dynamics away from capability transfer during optimization, and consolidates the induced behavior under the student's own rollout distribution. Extensive experiments demonstrate that DISTILSHIELD provides a plug-and-play defense layer that effectively suppresses knowledge distillation while preserving the utility of teacher responses.

1 INTRODUCTION

Large language models (LLMs) have demonstrated remarkable capabilities across different xtasks (Achiam et al., 2023; Guo et al., 2025). Developing frontier LLMs requires substantial resource span data, computation, and human effort, which makes them valuable intellectual property (IP). Commercial model providers, therefore, often restrict access to model internals and only provide service through an API that returns generated responses to user prompts. Yet these responses can also create a channel for capability extraction. External adversaries can collect responses and use knowledge distillation (KD) (Hinton et al., 2015; Kim & Rush, 2016) to train a student that reproduces the teacher's capabilities, which motivates an emerging line of research on anti-distillation.

Existing anti-distillation methods largely assume that knowledge transfer relies on fitting teacher-generated responses. They therefore seek to disrupt distillation by making those responses difficult for students to fit (e.g., perturbing output-token distributions (Savani et al., 2026; Li et al., 2025) and obfuscating intermediate reasoning (Ding et al., 2025; Ma et al., 2026)). Revisiting this premise reveals a different opportunity: instead of preventing data fitting, an effective defense could directly suppress the transfer of the capability underlying those data. We therefore ask *can a student easily fit protected responses yet fail to acquire the generalizable capability behind them?*

Recently, unlearnable examples (UEs) (Huang et al., 2021) have emerged as an effective approach to protecting data privacy in computer vision. By introducing imperceptible perturbations to training examples (often into the input), UEs can induce models to fit artificial patterns that do not generalize to the clean distribution. Inspired by this principle, we explore the possibility of protecting LLMs from distillation. We conduct a warm-up experiment to inject shortcuts into the teacher's reasoning. As shown in Figure 1A, the student easily fits these responses yet acquires less generalizable capability, providing the evidence that data fitting and knowledge transfer can be decoupled.

At the same time, anti-distillation poses several unique challenges. First, unlike typical UEs settings where models are trained from scratch, student LLMs already encode substantial prior knowledge

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

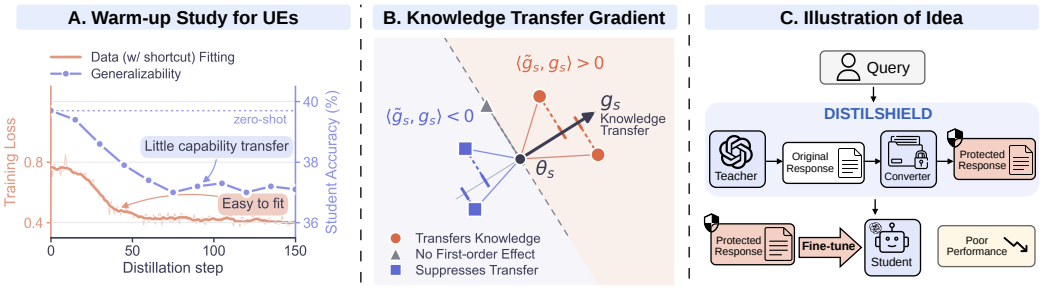


Figure 1: Problem illustration. (A) Warm-up study for shortcut injection. (B) Knowledge transfer gradient for shortcut binding. (C) Illustration of DISTILSHIELD for protecting teacher responses.

from pretraining. Consequently, A naive shortcut may therefore coexist with genuine capability learning: even if the student fits the shortcut, updates induced by the protected responses may still reinforce its existing capacity and improve generalization, as illustrated in Figure 1B. Second, since the shortcut is embedded in the teacher's reasoning traces, the effectiveness depends on the teacher-forced contexts during distillation. At inference time, however, the student conditions on its own previously generated tokens and follows its own rollout distribution, which can deviate from the protected traces and thereby circumvent the shortcut. An effective defense must prevent the associated training updates from enabling knowledge transfer while ensuring that the induced behavior remains effective under the student's own rollouts.

In this work, we propose **DISTILSHIELD**, an anti-distillation framework that transforms original teacher responses into protected counterparts while preserving the utility, as presented in Figure 1C. DISTILSHIELD optimizes an *Unlearnable Converter* through the student's learning trajectory, progressively thwarting knowledge distillation. (*i*) *Shortcut induction.* We implant an implicit perturbation into the teacher's reasoning through error-minimization, offering a simpler learning path than recovering the teacher's underlying reasoning procedure. (*ii*) *Shortcut binding.* To avoid genuine capability acquisition, DISTILSHIELD explicitly steers the student update induced by a protected response away from the direction of clean knowledge transfer, so that successful fitting does not guarantee improved generalization during optimization. (*iii*) *Shortcut consolidation.* Finally, it consolidates the induced shortcut under the student's own rollout distribution, ensuring that the protection persists during autonomous generation. Together, these three components turn protected responses into examples that remain easy to fit, yet systematically suppress the acquisition and deployment of the teacher's generalizable capability. To optimize the converter, we propose **ANTI-GDPO**, a bilevel optimization algorithm that alternates between proxy student updates and converter updates while effectively balancing the different objectives. Comprehensive experiments show that DISTILSHIELD substantially suppresses knowledge distillation while maintaining utility, offering a plug-and-play defense layer for LLM IP protection. Our contributions are summarized as:

- We revisit the prevailing premise that preventing knowledge transfer requires making teacher responses difficult to fit. Inspired by unlearnable examples, we introduce a new anti-distillation perspective that decouples data fitting from capability transfer and directly suppresses the transfer of the teacher's generalizable capability. To the best of our knowledge, this is the first work to explore anti-distillation from the perspective of unlearnable examples.
- We identify two challenges inherent in LLM anti-distillation: pretrained prior knowledge of student models and the discrepancy between teacher-forced contexts and student rollouts. We show that challenges can undermine the effectiveness of naive shortcut injection, motivating a more systematic approach to anti-distillation.
- We propose DISTILSHIELD that converts teacher responses into semantically useful yet unlearnable counterparts. The proposed three objectives are complementary, leading to progressive protection. We further develop ANTI-GDPO to jointly optimize these objectives.
- We conduct extensive experiments across six datasets and eight SOTA LLMs, demonstrating the superior performance of DISTILSHIELD. Ablation studies, parameter sensitivity, robustness analysis and case study further validate the mechanism advantages. Overall, DISTILSHIELD provides a plug-and-play defense layer for LLM IP protection.

2 Related Work

We briefly review the most relevant literature here and defer an extended discussion to Appendix A.

Knowledge Distillation. Knowledge distillation transfers knowledge from a teacher to a typically smaller student (Hinton et al., 2015). For LLMs, black-box methods distill generated responses and reasoning traces (Hsieh et al., 2023; Li et al., 2023), whereas white-box methods further exploit the teacher's output distributions (Gu et al., 2024). Self-distillation (Zhang et al., 2024b; Huang et al., 2023) and on-policy distillation (Agarwal et al., 2024) extend the paradigm to model-generated trajectories. Our work defends against black-box sequence-level KD for LLM IP protection.

Anti-Distillation and Model IP Protection. Watermarking and fingerprinting enable model owners to verify provenance and claim ownership of released outputs (Sander et al., 2024; Xu et al., 2024a). Complementarily, proactive anti-distillation methods instead make teacher responses difficult to fit by perturbing the sampling distribution (Savani et al., 2026), adapting the output layer (Li et al., 2025), or reformulating reasoning traces (Ding et al., 2025; Ma et al., 2026). We revisit the common premise and prevent distillation by directly impeding the knowledge-transfer process.

Unlearnable Examples. Unlearnable examples protect data from unauthorized training with error-minimizing perturbations that create easy-to-fit shortcuts failing to generalize to clean data (Huang et al., 2021). Subsequent work studies their robustness (Fu et al., 2022), transferability (Ren et al., 2022), and extends them to other modalities (Liu et al., 2024; Zhang & Huang, 2024; Jiang et al., 2024) including text (Li & Liu, 2023; Zhang & Sun, 2026). Despite UEs being introduced for data protection, we draw on their core principle to explore their potential for anti-distillation.

3 Preliminaries

3.1 Sequence-Level Knowledge Distillation

Let $\mathcal{T}$ denote a teacher with output distribution $p_{\mathcal{T}}$, and let $\mathcal{S}$ denote a student with trainable parameters θ and output distribution $p_{\mathcal{S}}$. Given a query set $\mathcal{D}_x = \{x_i\}_{i=1}^N$, the teacher produces a distillation corpus $\mathcal{D}_{\mathcal{T}} = \{(x_i, y_i) : y_i \sim p_{\mathcal{T}}(\cdot \mid x_i)\}_{i=1}^N$. Sequence-level knowledge distillation trains the student by minimizing the autoregressive negative log-likelihood of these responses:

$$\mathcal{L}_{\mathrm{SKD}}(\theta; \mathcal{D}_{\mathcal{T}}) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{|y_i|} \log p_{\mathcal{S}}(y_{i,j} \mid x_i, y_{i,<j}; \theta), \quad \theta^* \in \arg\min_{\theta} \mathcal{L}_{\mathrm{SKD}}(\theta; \mathcal{D}_{\mathcal{T}}). \tag{1}$$

3.2 Unlearnable Examples

Unlearnable examples protect data against unauthorized exploitation by injecting imperceptible perturbations δ. The representative error-minimizing method (Huang et al., 2021) solves:

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \min_{\delta_i \in \mathcal{C}(x_i)} \ell\big(f_\theta(x_i \oplus \delta_i), y_i\big), \tag{2}$$

where $\mathcal{C}(x_i)$ is the feasible perturbations and ℓ is a loss function. The learned perturbations create a spurious shortcut between the input and label, thereby degenerating the model's generalizability.

3.3 Anti-Distillation

Let $\mathcal{A}$ be an anti-distillation algorithm and $\tilde{\mathcal{T}} = \mathcal{A} \circ \mathcal{T}$ denotes the defensive teacher. Let $\tilde{\mathcal{S}}$ be the student distilled from $\tilde{\mathcal{T}}$ via Equation 1. Given a held-out set $\mathcal{D}_{\mathrm{eval}}$, anti-distillation aims to

$$\begin{aligned} \min_{\mathcal{A}} \quad & \mathrm{Perf}(\tilde{\mathcal{S}}; \mathcal{D}_{\mathrm{eval}}) \\ \text{s.t.} \quad & \mathrm{Perf}(\mathcal{T}; \mathcal{D}_{\mathrm{eval}}) - \mathrm{Perf}(\tilde{\mathcal{T}}; \mathcal{D}_{\mathrm{eval}}) \le \varepsilon, \end{aligned} \tag{3}$$

where ε is the allowed teacher-performance degradation.

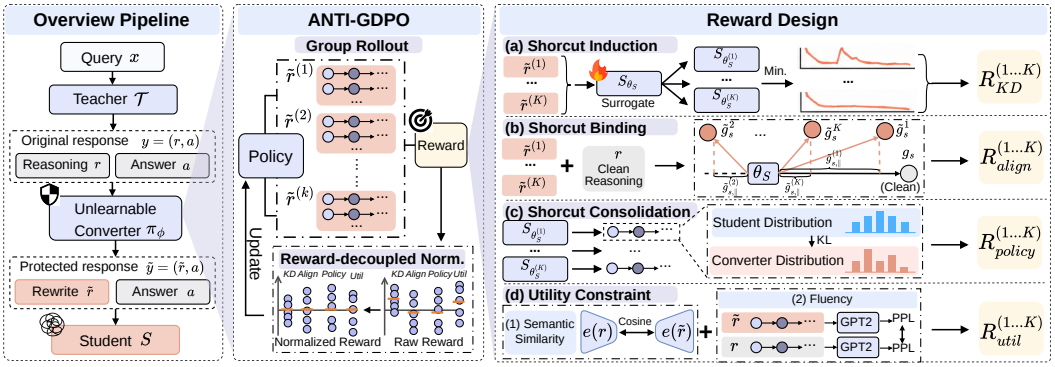


Figure 2: Overview of DISTILSHIELD.

4 THE PROPOSED DISTILSHIELD

4.1 UNLEARNABLE CONVERTER FOR SHORTCUT INDUCTION

Unlearnable Converter. For a teacher response $y_i = (r_i, a_i)$, consisting of a reasoning trace r_i and a final answer a_i, DISTILSHIELD instantiates $\mathcal{A}$ through an *Unlearnable Converter* π_ϕ. The converter is implemented by an autoregressive LLM and rewrites only the reasoning trace:

$$\widetilde{r}_i \sim \pi_\phi(\cdot \mid x_i, r_i, a_i), \qquad \widetilde{y}_i = (\widetilde{r}_i, a_i). \tag{4}$$

This design offers three practical advantages. (*i*) The converter acts as a plug-and-play serving-time wrapper: it leaves the teacher unchanged and amortizes its one-time training cost across queries and, potentially, teacher systems. (*ii*) It operates in natural-language space, allowing for flexible and natural perturbations injection. (*iii*) Modifying only the reasoning trace imposes a hard answer-preservation constraint, thereby retaining the teacher's task utility.

Shortcut Induction. We inject shortcuts into teacher reasoning traces by optimizing the converter to generate rewrites that minimizes the student's distillation loss. Considering a mini-batch generated by the converter $\widetilde{\mathcal{B}} = \{(x_i, \widetilde{y}_i)\}_{i=1}^{|\widetilde{\mathcal{B}}|}$, the objective is defined as bilevel optimization:

$$\min_{\phi} \ \mathcal{L}_{\mathrm{SKD}}(\theta^*; \widetilde{\mathcal{B}}) \quad \text{s.t.} \quad \theta^* \in \arg\min_{\theta} \mathcal{L}_{\mathrm{SKD}}(\theta; \widetilde{\mathcal{B}}). \tag{5}$$

Solving the outer problem encourages the converter to generate rewrites that are easy for the student to fit, while the inner problem simulates the student's distillation process. In practice, we use proxy models after several inner updates to approximate the distillation trajectory.

4.2 NORMALIZED GRADIENT ANTI-ALIGNMENT FOR SHORTCUT BINDING

Due to the substantial prior knowledge encoded by pretrained student LLMs, a naive shortcut alone may not prevent capability transfer. We therefore steer the updates induced by protected responses away from clean capability acquisition. Given a protected mini-batch $\widetilde{\mathcal{B}}$ and its clean counterpart $\mathcal{B} = \{(x_i, y_i)\}_{i=1}^{|\mathcal{B}|}$, we define the gradients of knowledge distillation at training step s as

$$\widetilde{g}_s = \nabla_\theta \mathcal{L}_{\mathrm{SKD}}(\theta_s; \widetilde{\mathcal{B}}), \qquad g_s = \nabla_\theta \mathcal{L}_{\mathrm{SKD}}(\theta_s; \mathcal{B}). \tag{6}$$

To quantify the influence of optimizing on the protected data on clean knowledge transfer, we examine how a protected gradient step changes the clean KD loss with a first-order Taylor expansion:

$$\begin{aligned} \mathcal{L}_{\mathrm{SKD}}(\theta_s - \eta_s \widetilde{g}_s; \mathcal{B}) &\approx \mathcal{L}_{\mathrm{SKD}}(\theta_s; \mathcal{B}) + \langle \nabla_\theta \mathcal{L}_{\mathrm{SKD}}(\theta_s; \mathcal{B}), -\eta_s \widetilde{g}_s \rangle \\ &= \mathcal{L}_{\mathrm{SKD}}(\theta_s; \mathcal{B}) - \eta_s \langle \widetilde{g}_s, g_s \rangle. \end{aligned} \tag{7}$$

The inner product therefore serves as a local knowledge transfer signal: positive alignment indicates that the protected update also reduces the clean KD loss, whereas orthogonal or negative alignment

provides no first-order benefit to fitting the clean responses. Following this insight, we suppress positive alignment across the optimization trajectory with the normalized anti-transfer objective

$$\min_{\phi} \mathcal{L}_{\text{align}} = \frac{1}{S} \sum_{s=0}^{S-1} \frac{\langle \widetilde{g}_s, g_s \rangle}{\|g_s\|_2^2 + \epsilon}, \tag{8}$$

where $\epsilon > 0$ is a small constant and the denominator measures the protected gradient's projection in units of the clean-gradient energy, making scores across trajectory steps more comparable.

4.3 On-Policy Alignment for Shortcut Consolidation

The converter is optimized under teacher-forced contexts, so the induced shortcut may not persist under the student's own rollout distribution. Given an optimized proxy student $\mathcal{S}$ parametrized by θ^*, we form an on-policy rollout batch $\widehat{\mathcal{B}} = \{(x_i, \widehat{y}_i) \mid (x_i, y_i) \in \mathcal{B},\ \widehat{y}_i \sim p_{\mathcal{S}}(\cdot \mid x_i; \theta^*)\}$. We then consolidate the shortcut by minimizing the KL divergence between the student and converter distributions at each rollout prefix:

$$\min_{\phi} \quad \mathcal{L}_{\text{policy}} = \frac{1}{|\widehat{\mathcal{B}}|} \sum_{i=1}^{|\widehat{\mathcal{B}}|} \sum_{j=1}^{|\widehat{y}_i|} D_{\text{KL}}(p_{\mathcal{S}}(\cdot \mid x_i, \widehat{y}_{i,<j}; \theta^*) \,\|\, \pi_\phi(\cdot \mid x_i, y_i, \widehat{y}_{i,<j})) \tag{9}$$

Optimizing Equation 9 encourages the converter to produce rewrites that are easy for the student to fit under its own rollout distribution, thus stabilizing the induced behavior. If the student and converter use incompatible tokenizations, we use the sequence-level surrogate:

$$\min_{\phi} \mathcal{L}^{\text{seq}}_{\text{policy}} = -\frac{1}{|\widehat{\mathcal{B}}|} \sum_{i=1}^{|\widehat{\mathcal{B}}|} \sum_{j=1}^{|\widehat{y}_i|} \log \pi_\phi(\widehat{y}_{i,j} \mid x_i, y_i, \widehat{y}_{i,<j}) \,. \tag{10}$$

4.4 Reward-Decoupled Group Policy Optimization

Utility Constraints. To ensure the quality of the generated rewrites for legitimate users, we impose utility constraints on the generated outputs. Specifically, we require that the generated rewrites maintain semantic similarity and naturalness with respect to the original text. For a clean mini-batch $\mathcal{B}$ and its rewritten counterpart $\widetilde{\mathcal{B}}$, we define the utility objective as follows:

$$\min_{\phi} \mathcal{L}_{\text{util}} = -\frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \left[\cos(e(r_i), e(\widetilde{r}_i)) + \min\left\{1, \frac{\text{PPL}(r_i)}{\text{PPL}(\widetilde{r}_i)}\right\} \right]. \tag{11}$$

where $e(\cdot)$ is a sentence encoder (e.g., SBERT (Reimers & Gurevych, 2019)) and PPL is measured by a pre-trained language model (e.g., GPT-2 (Radford et al., 2019)). Note that minimizing this objective potentially mitigates the reward hacking.

Overall objective. Combining the protection and utility objectives, we obtain the antidistillation objective and formulate it as constrained bilevel optimization:

$$\begin{aligned} \min_{\phi} \quad & \mathcal{L}_{\text{distill}} = \underbrace{\lambda_{\text{KD}} \mathcal{L}_{\text{SKD}}}_{\text{shortcut induction}} + \underbrace{\lambda_{\text{align}} \mathcal{L}_{\text{align}}}_{\text{shortcut binding}} + \underbrace{\lambda_{\text{policy}} \mathcal{L}_{\text{policy}}}_{\text{shortcut consolidation}} + \underbrace{\lambda_{\text{util}} \mathcal{L}_{\text{util}}}_{\text{utility constraints}} \\ \text{s.t.} \quad & \widetilde{r} \sim \pi_\phi(\cdot \mid x, r, a), \quad \widetilde{y} = (\widetilde{r}, a), \quad \theta^* \in \arg\min_{\theta} \mathcal{L}_{\text{SKD}}(\theta; \widetilde{\mathcal{D}}_{\mathcal{T}}), \end{aligned} \tag{12}$$

The nonnegative coefficients λ_{KD}, λ_{align}, λ_{policy}, and λ_{util} control the contributions of the four objectives. Directly optimizing the overall objective is challenging since it involves discrete sampling and nested optimization of the student model. We propose Anti-GDPO that leverages reinforcement learning (RL) to naturally orchestrate the optimization by maximizing the expected reward.

$$R = -\mathcal{L}_{\text{distill}}, \quad \max_{\phi} \mathbb{E}_{\widetilde{r} \sim \pi_\phi(\cdot \mid x, r, a)}[R]. \tag{13}$$

Reward-decoupled normalization.

Aggregating these heterogeneous rewards before normalization can allow a large-scale or high-variance channel to dominate the advantage. Following GDPO (Liu et al., 2026), we standardize each channel within the candidate group before combining them. Let $\mathcal{M} = \{\mathrm{KD}, \mathrm{align}, \mathrm{policy}, \mathrm{util}\}$ denote the set of objective components. At each step, the frozen rollout policy $\pi_{\phi_{\mathrm{old}}}$ samples K candidate rewrites $\{\widetilde{r}^{(k)}\}_{k=1}^{K}$, and $R_m^{(k)} = -\mathcal{L}_m^{(k)}$ denotes the reward of candidate k for component m. We compute the normalized advantages as

$$\overline{R}_m = \frac{1}{K}\sum_{k=1}^{K} R_m^{(k)}, \quad \widehat{A}_m^{(k)} = \frac{R_m^{(k)} - \overline{R}_m}{\sqrt{K^{-1}\sum_{\ell=1}^{K}(R_m^{(\ell)} - \overline{R}_m)^2 + \epsilon}}, \quad \widehat{A}^{(k)} = \sum_{m\in\mathcal{M}} \lambda_m \widehat{A}_m^{(k)}, \tag{14}$$

where the parenthesized superscript (k) indexes a candidate rollout, $m \in \mathcal{M}$ indexes an objective component, and $\epsilon > 0$ ensures numerical stability. For token j of rollout k, the importance ratio is

$$\rho_j^{(k)}(\phi) = \pi_\phi\left(\widetilde{r}_j^{(k)} \mid x, r, a, \widetilde{r}_{<j}^{(k)}\right) \Big/ \pi_{\phi_{\mathrm{old}}}\left(\widetilde{r}_j^{(k)} \mid x, r, a, \widetilde{r}_{<j}^{(k)}\right). \tag{15}$$

The converter then maximizes the clipped objective:

$$\begin{aligned}\mathcal{J}_{\text{ANTI-GDPO}}(\phi) = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{|\widetilde{r}^{(k)}|}\sum_{j=1}^{|\widetilde{r}^{(k)}|}\min\Big\{\rho_j^{(k)}(\phi)\widehat{A}^{(k)}, \mathrm{clip}\Big(\rho_j^{(k)}(\phi), 1-\tau, 1+\tau\Big)\widehat{A}^{(k)}\Big\} \\ - \frac{\beta}{K}\sum_{k=1}^{K}\frac{1}{|\widetilde{r}^{(k)}|}\sum_{j=1}^{|\widetilde{r}^{(k)}|} D_{\mathrm{KL}}\Big(\pi_\phi(\cdot \mid x, r, a, \widetilde{r}_{<j}^{(k)}) \,\|\, \pi_{\mathrm{ref}}(\cdot \mid x, r, a, \widetilde{r}_{<j}^{(k)})\Big),\end{aligned} \tag{16}$$

where $\tau > 0$ is the clipping radius, $\beta \geq 0$ controls KL regularization, and π_{ref} is the initial converter. Appendix E elaborates the complete training procedure in Algorithm 1 and Algorithm 2. At deployment, DISTILSHIELD requires only one converter forward pass and no student simulation.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Datasets. We conduct experiments on six datasets: *math reasoning:* MATH (Hendrycks et al., 2021) and GSM8K (Cobbe et al., 2021); *commonsense reasoning:* ARC-Challenge (Clark et al., 2018) and CommonsenseQA (Talmor et al., 2019); and *code generation:* HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021). Appendix F.1 provides dataset statistics.

Models. We consider two teachers, DeepSeek-R1-Distill-Qwen-7B (Guo et al., 2025) and Qwen3-8B (Yang et al., 2025), and six students: Qwen2.5-0.5B (Qwen et al., 2025), Gemma-3-1B-it (Team et al., 2025), Llama-3.2-1B-Instruct (Grattafiori et al., 2024), Qwen3-0.6B, Qwen3-1.7B, and Llama-3.2-3B. We adopt Qwen2.5-3B as initialization of the proxy student and the converter.

Baselines. We compare DISTILSHIELD against three groups of baselines. (*i*) *Control*: Zero-shot (Kojima et al., 2022) and Sequence-level Knowledge Distillation (SKD) (Kim & Rush, 2016) as the no-defense reference. (*ii*) *Anti-distillation methods*: Antidistillation Sampling (ADS) (Savani et al., 2026), DOGe (Li et al., 2025), and Trace Rewriting (Ma et al., 2026). (*iii*) *UE-extended methods*: unlearnable-example techniques that we adapt to the anti-distillation setting, including Random-Prefix, Artificial Pattern (Li & Liu, 2023), Text-MinMin (Huang et al., 2021), and Multi-step Error Minimization (MEM) (Liu et al., 2024). Detailed descriptions are given in Appendix F.2.

Evaluation Metrics & Implementation. We use accuracy to measure both the teacher's and the student's performance. Implementation details of DISTILSHIELD are provided in Appendix F.3.

5.2 MAIN EXPERIMENTS: OVERALL PERFORMANCE

Comparison with Baselines. Table 1 reports the teacher and student accuracy of baselines across the six datasets. (*i*) DISTILSHIELD provides the strongest protection on every dataset, reducing the average student accuracy from 38.6% under SKD to 19.3%, whereas the strongest baseline (ADS)

Table 1: Teacher (T) and student (S) accuracy (%) on six datasets. The best results are in **bold**. We average the results over three runs and report the standard deviation (STD) in Appendix G.1.

Method *R1-7B → Llama-3.2-1B*	GSM8K		MATH		ARC-C		CSQA		HEval		MBPP		Avg.	
	T↑	S↓	T↑	S↓	T↑	S↓	T↑	S↓	T↑	S↓	T↑	S↓	T↑	S↓
Zero-shot (Kojima et al., 2022)	92.4	39.7	84.9	28.6	81.4	32.3	66.8	54.3	88.0	20.7	68.0	21.6	80.2	32.8
SKD (Kim & Rush, 2016)	92.4	47.9	84.9	31.1	81.4	41.8	66.8	57.4	88.0	22.7	68.0	30.9	80.2	38.6
ADS (Savani et al., 2026)	90.2	32.4	61.4	22.1	81.4	32.2	63.8	50.1	26.0	20.0	32.9	20.7	59.3	29.6
DOGe (Li et al., 2025)	89.1	42.6	84.4	28.4	81.3	32.2	**66.9**	55.0	80.0	18.7	68.0	22.3	78.3	33.2
Trace Rewriting (Ma et al., 2026)	92.4	33.1	84.9	24.2	81.4	32.0	66.8	54.6	88.0	22.0	68.0	21.7	80.2	31.3
Random-Prefix	92.4	34.8	84.9	26.8	81.4	32.6	66.8	51.6	88.0	19.3	68.0	22.0	80.2	31.2
Artificial Pattern (Li & Liu, 2023)	92.4	39.3	84.9	28.2	81.4	31.6	66.8	53.9	88.0	21.3	68.0	21.8	80.2	32.7
Text-MinMin (Huang et al., 2021)	90.5	38.6	78.7	27.5	79.7	32.2	61.1	53.0	87.1	18.0	68.0	21.3	77.5	31.8
MEM-3 (Liu et al., 2024)	92.4	41.4	84.9	26.0	81.4	32.0	66.8	51.2	88.0	20.0	68.0	21.6	80.2	32.0
MEM-5 (Liu et al., 2024)	92.4	41.5	84.9	24.4	81.4	32.3	66.8	51.6	88.0	18.0	68.0	21.8	80.2	31.6
DISTILSHIELD (Ours)	**92.4**	**21.1**	**84.9**	**16.5**	**81.4**	**31.5**	66.8	**10.2**	**88.0**	**16.7**	**68.0**	**20.1**	**80.2**	**19.3**

Table 2: Performance across teacher and student architectures. STD is reported in Appendix G.1.

	Teacher	Student	SKD		ADS		DOGe		Ours	
			T↑	S↓	T↑	S↓	T↑	S↓	T↑	S↓
GSM8K	R1-7B	Qwen2.5-0.5B	92.4	41.1	90.2	41.5	89.1	42.2	**92.4**	**38.4**
		Gemma-3-1B		45.0		51.5		50.3		**41.4**
		Llama-3.2-1B		47.9		32.4		42.6		**21.1**
	Qwen3-8B	Qwen3-0.6B	94.9	61.3	88.0	70.9	**95.0**	61.2	94.9	**55.9**
		Qwen3-1.7B		80.9		89.7		84.2		**78.7**
		Llama-3.2-3B		39.4		34.5		31.7		**17.6**
ARC-C	R1-7B	Qwen2.5-0.5B	81.4	29.3	81.4	44.1	81.3	42.2	**81.4**	**13.4**
		Gemma-3-1B		44.2		32.2		36.4		**30.3**
		Llama-3.2-1B		41.8		32.2		32.2		**31.5**
	Qwen3-8B	Qwen3-0.6B	95.7	67.8	94.2	68.2	**95.9**	67.1	95.7	**61.0**
		Qwen3-1.7B		87.1		88.1		87.9		**85.2**
		Llama-3.2-3B		47.9		62.5		60.0		**45.8**
HEval	R1-7B	Qwen2.5-0.5B	88.0	12.0	26.0	13.3	80.0	12.7	**88.0**	**11.3**
		Gemma-3-1B		35.3		34.0		33.3		**33.3**
		Llama-3.2-1B		22.7		20.0		18.7		**16.7**
	Qwen3-8B	Qwen3-0.6B	86.0	36.0	24.0	34.0	86.0	36.7	**86.0**	**31.3**
		Qwen3-1.7B		70.7		74.7		74.7		**68.7**
		Llama-3.2-3B		28.0		26.0		29.3		**23.3**

Table 3: Ablation study. w/o removes the corresponding objectives from Equation 12, and w/ replaces a design component of DISTILSHIELD with the specified variant. More ablation results are in Appendix G.2.

Configuration *R1-7B → Llama-3.2-1B*	GSM8K		ARC-C		HEval	
	T↑	S↓	T↑	S↓	T↑	S↓
Zero-shot	92.4	39.7	81.4	32.3	88.0	20.7
Full (DISTILSHIELD)	**92.4**	**21.1**	**81.4**	**31.5**	**88.0**	**16.7**
w/o $\mathcal{L}_{\text{SKD}}$	92.4	27.4	81.4	32.6	88.0	18.7
w/o $\mathcal{L}_{\text{align}}$	92.4	37.2	81.4	32.0	88.0	16.7
w/o $\mathcal{L}_{\text{policy}}$	92.4	30.7	81.4	34.2	88.0	18.0
w/o $\mathcal{L}_{\text{util}}$	92.4	38.0	81.4	33.7	88.0	20.0
w/o $\mathcal{L}_{\text{SKD}}, \mathcal{L}_{\text{policy}}$	92.4	31.4	81.4	35.3	88.0	19.3
w/o $\mathcal{L}_{\text{SKD}}, \mathcal{L}_{\text{align}}$	92.4	27.9	81.4	34.0	88.0	18.0
w/o $\mathcal{L}_{\text{align}}, \mathcal{L}_{\text{policy}}$	92.4	36.5	81.4	33.1	88.0	17.3
w/o $\mathcal{L}_{\text{SKD}}, \mathcal{L}_{\text{align}}, \mathcal{L}_{\text{policy}}$	92.4	32.5	81.4	32.7	88.0	18.0
w/o $\mathcal{L}_{\text{align}}, \mathcal{L}_{\text{policy}}, \mathcal{L}_{\text{util}}$	92.4	35.0	81.4	34.1	88.0	19.3
w/o $\mathcal{L}_{\text{SKD}}, \mathcal{L}_{\text{policy}}, \mathcal{L}_{\text{util}}$	92.4	37.1	81.4	33.8	88.0	20.0
w/ Convert (r, a)	90.5	24.0	78.8	32.5	84.7	17.4
w/ GRPO (Shao et al., 2024)	92.4	22.6	81.4	34.1	88.0	17.4

only reaches 29.6%. On GSM8K, MATH, and CSQA, the distilled student falls below its zero-shot accuracy (e.g., 10.2% vs. 54.3% on CSQA), indicating that the implanted shortcut effectively blocks knowledge transfer and undermines the student's capability. (*ii*) DISTILSHIELD keeps the teacher accuracy intact since only the reasoning trace is rewritten, whereas ADS and DOGe perturb the teacher's decoding and sacrifice utility (e.g., 26.0% for ADS on HumanEval). (*iii*) The UE-extended baselines barely reduce the student below its zero-shot level, confirming that a naive shortcut alone cannot prevent knowledge transfer to a pretrained student.

Comparison under Different Architectures. Table 2 compares DISTILSHIELD with the anti-distillation baselines across teacher-student combinations on GSM8K, ARC-C, and HumanEval. DISTILSHIELD achieves the lowest student accuracy in all settings, reducing the average student accuracy from 46.6% under SKD to 39.2%, whereas the strongest baseline (DOGe) reaches 46.9% and fails to reduce it at all. The protection generalizes across student families (Qwen, Gemma, and Llama), scales (0.5B-3B), and the stronger Qwen3-8B teacher, suggesting the broad versatility of the learned converter. Larger students retain higher accuracy, consistent with their stronger prior knowledge, yet DISTILSHIELD still outperforms all baselines in these settings.

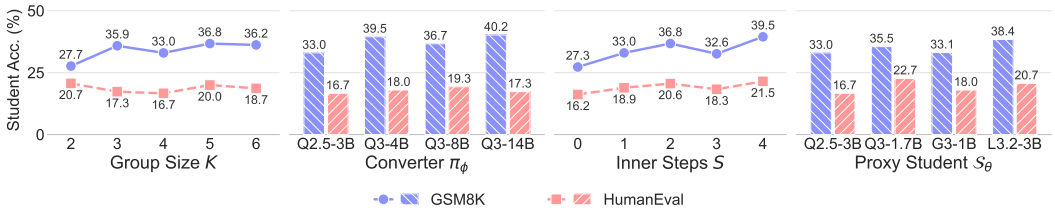


Figure 3: Parameter sensitivity. We further study the impact of batch size b in Appendix G.3.

Table 4: Robustness analysis on GSM8K. Appendix G.4 provides results for other architectures.

Teacher *Qwen3-8B* → *Qwen3-0.6B*	Defense Method				Distillation Method		
	None	Adv. Training	Rep. Query	Paraphrase	SKD	Off-Policy	On-Policy
Vanilla	61.3	59.2	60.6	57.4	61.3	73.6	75.8
Defensive (DISTILSHIELD)	55.9	56.2	55.1	52.1	55.9	61.6	55.5

5.3 ABLATION STUDIES

Table 3 ablates DISTILSHIELD by removing its objectives individually and in combination (w/o) and by replacing its design components with alternative variants (w/). (*i*) Every objective contributes to the overall effectivenss, and removing the three protection objectives brings the student close to its zero-shot level. Among them, shortcut binding is the most critical: without $\mathcal{L}_{\text{align}}$, the student recovers to 37.2% on GSM8K despite fitting the shortcut. Interestingly, further removing $\mathcal{L}_{\text{SKD}}$ improves protection (27.9%), showing that an unbound shortcut can even reinforce the student's prior capability and that induction is only effective when coupled with binding. (*ii*) Shortcut consolidation strengthens protection on all datasets (e.g., 30.7% vs. 21.1% on GSM8K), confirming that the shortcut must persist under the student's own rollouts. (*iii*) Removing the utility constraint degrades protection to near zero-shot (38.0% on GSM8K), indicating that it also prevents reward hacking toward degenerate rewrites that the student ignores. (*iv*) Rewriting the answer along with the reasoning brings no additional protection (e.g., 24.0% vs. 21.1% on GSM8K) but sacrifices teacher accuracy, and replacing ANTI-GDPO with GRPO weakens protection by 1.8 points on average, validating the design of the answer-preserving converter and the reward-decoupled normalization.

5.4 PARAMETER SENSITIVITY

Figure 3 studies the sensitivity of DISTILSHIELD to the group size, the converter backbone, the number of inner steps, and the proxy student. Overall, DISTILSHIELD is robust to these choices: on GSM8K, the student accuracy stays within 27.3%-40.2% across all settings, well below the 47.9% under SKD, and HumanEval exhibits similar trends. Notably, the protection does not rely on scaling up. A small group size, a short simulated trajectory, and a compact 3B converter already provide strong protection, whereas larger groups, more inner steps, and larger backbones bring few further gains. Moreover, protection transfers versatilely across proxy students of different families and sizes, confirming that DISTILSHIELD offers a consistently effective defense.

5.5 ROBUSTNESS ANALYSIS

Robustness against Defenses. Table 4 evaluates DISTILSHIELD when the adversary attempts to mitigate the anti-distillation by using adversarial training, repetitive querying, or paraphrasing (implementation details are in Appendix F.4). DISTILSHIELD remains effective against all defenses, keeping the student at least 3.0 points below its counterpart distilled from the vanilla teacher. Paraphrasing and repetitive querying barely recover the student (5.3 and 5.5 points below vs. 5.4 without defense), indicating the stability of the converter for implanting shortcuts. Adversarial training narrows the gap most, yet the student still lags 5.1 points behind distilling from the vanilla teacher.

Robustness across KD Methods. Table 4 further evaluates DISTILSHIELD under **extreme case** where attacker transcends the black-box setting and applies SOTA KD techniques by accessing the teacher's internal logits. We consider sequence-level, off-policy, and on-policy distillation. Stronger

Table 5: Efficiency of DISTILSHIELD

Time	GSM8K	MATH	ARC-C	CSQA	HEval	MBPP
Training	4h 47m	4h 32m	5h 37m	5h 31m	5h 58m	5h 53m
Inference	5m 43s	58m 2s	5m 46s	5m 50s	1m 58s	6m 49s

Table 6: Statistics of teacher responses.

Teacher	Length	ROUGE-L	BERTScore	PPL
Vanilla	254.6	1.00	1.00	9.6
Defensive	272.4	0.40	0.90	6.1

distillation methods improve the vanilla-distilled student (up to 75.8%), yet DISTILSHIELD reduces the student accuracy by 12.5 points on average. Notably, the defensive student gains little even for on-policy distillation (55.5% vs. 55.9% under SKD), confirming that shortcut consolidation makes the induced shortcut persist under the student's own rollout distribution.

5.6 EFFICIENCY ANALYSIS

Table 5 reports the training and inference time running on one NVIDIA 80GB A100 GPU. Training is a one-time cost dominated by the proxy-student simulation and amortized across all subsequent queries, whereas serving requires only an additional converter forward pass, adding 0.13 s per response and 20% latency overhead on average to vanilla teacher generation (Appendix G.5).

5.7 CASE STUDY

Quantitative Analysis. Table 6 compares the responses of the vanilla and the defensive teacher averaged over all datasets. The protected responses lexically diverge from the originals (ROUGE-L of 0.40) yet retain high semantic similarity (BERTScore of 0.90) and even lower perplexity (6.1 vs. 9.6) with only a modest increase in length, confirming that DISTILSHIELD provides defense without compromising the quality of the responses. Further human study (detailed in Appendix G.6) evaluates response quality through correctness, naturalness, fluency, and human utility, indicating that DISTILSHIELD DISTILSHIELD suppresses knowledge distillation without a perceptible loss of quality for legitimate users.

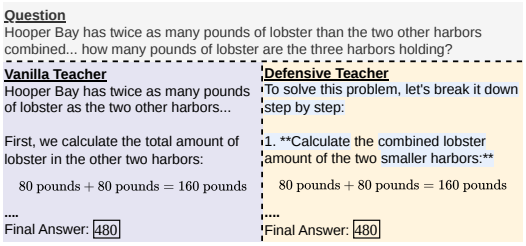


Figure 4: Case study on GSM8K.

Qualitative Analysis. Figure 4 shows an example response of the vanilla and the defensive teacher. The defensive teacher preserves every computation step and the final answer, but recasts the reasoning into a pragmatic template with a stereotyped opening and numbered step headers (highlighted). This predictable structure is easy to fit thereby guiding the student to neglect the underlying reasoning and preventing knowledge transfer. We provide examples on other datasets in Appendix J.

6 CONCLUSION

In this work, we revisit anti-distillation through the lens of unlearnable examples and rethink the conventional premise that preventing knowledge transfer requires making teacher responses difficult to learn. Our key insight is that data fitting can be decoupled from capability transfer: a student may well fit protected responses without acquiring the generalizable capability. Building on this principle, we propose DISTILSHIELD, which induces easy-to-fit shortcuts in teacher reasoning, binds the resulting updates away from capability transfer, and consolidates the induced behavior under the student's own rollout distribution. Extensive experiments demonstrate effective protection across diverse tasks and model architectures while preserving the utility of teacher responses. More broadly, our results suggest a new direction for LLM protection: rather than obstructing learning itself, anti-distillation can shape *what* the student learns from otherwise useful outputs. We hope this learning-centric view motivates future work on systematic understanding of anti-distillation.

AI USE STATEMENT

In this work, we used generative AI tools to generate synthetic datasets, to provide feedback on the research methodology and experimental design, to assist in implementing the proposed method and baselines, and to help interpret experimental results. We have not used generative AI tools to develop the theoretical models or conceptual frameworks, to formulate mathematical claims, to propose or refine hypotheses, or to clean and reformat datasets. Assisting in proving mathematical claims or writing proofs, translation, and qualitative and thematic data analysis are not applicable to this work. Additionally, we used generative AI tools to create and modify scientific figures, to create and edit software code, to draft parts of the paper, to edit the paper for readability, and to identify relevant literature.

We have reviewed all AI-assisted work. The research problem, the core idea of decoupling data fitting from knowledge transfer, the design of DISTILSHIELD and ANTI-GDPO, and the experimental plan were conceived by the authors; AI-generated feedback on the methodology and experiments was treated as suggestions and adopted only after the authors' independent judgment. All AI-assisted code was reviewed, tested, and debugged by the authors, and all reported results were produced by running the final code under the authors' control. AI-assisted interpretations of results were checked against the raw experimental outputs. Literature identified with AI assistance was manually verified by reading the original papers to confirm their existence, content, and relevance, and all references were checked against their official sources. AI-drafted and AI-edited text and figures were carefully revised by the authors to ensure that every claim accurately reflects our method and findings. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

The primary contribution of this work is a defense that protects the intellectual property of LLMs against unauthorized knowledge distillation. Developing frontier LLMs requires substantial investment in data, computation, and human effort, and unauthorized extraction of their capabilities through generated responses may undermine incentives for such investment and violate the terms of service under which these models are offered. By providing a plug-and-play protection layer, our work helps model providers safeguard their assets while continuing to serve legitimate users. DISTILSHIELD rewrites only the reasoning trace while preserving the final answer, and our experiments show that the protected responses largely retain their task utility, thereby minimizing the impact on legitimate users. All model experiments are conducted on publicly available benchmark datasets and open-weight models and involve no personal or sensitive data. We additionally conduct a small-scale human evaluation of response quality, in which annotators assess model-generated responses along dimensions such as correctness, naturalness, fluency, and utility. The evaluation does not require annotators to provide personal or sensitive information.

However, we also recognize potential risks. Anti-distillation is inherently dual-use: the same mechanism that prevents unauthorized capability extraction could be applied to obstruct legitimate distillation for beneficial purposes, such as academic research, model compression for resource-constrained deployment, or the development of open models, and widespread adoption could further concentrate advanced capabilities among a small number of providers. Furthermore, the principle of decoupling data fitting from knowledge transfer could in principle be repurposed to degrade models trained on protected text beyond the distillation setting.

To mitigate these risks, we advocate that anti-distillation be applied only to a provider's own model outputs and be disclosed transparently to users, so that they are aware that reasoning traces may be reformulated. We also encourage the community to study the appropriate balance between protecting model intellectual property and preserving the openness that has driven progress in the field, and to continue developing robust evaluation protocols such as the adaptive attacks considered in this paper. We believe that transparently acknowledging these tensions is a necessary step toward the responsible development of LLM protection techniques.

REPRODUCIBILITY STATEMENT

We have made efforts to ensure that our results are reproducible. All datasets used in our experiments are standard, publicly available benchmarks, and their statistics and preprocessing are described in

Appendix F.1. All teacher, student, proxy, and converter models are open-weight models that are publicly accessible. The proposed objectives and the optimization algorithm are formally defined in Section 4, and complete pseudo-code for constructing the defensive teacher and training the Unlearnable Converter is provided in Appendix E. Implementation details, including hyperparameters, training schedules, and evaluation protocols, are given in Appendix F.3, and the implementations of all baselines are described in Appendix F.2. The sensitivity of DISTILSHIELD to its key hyperparameters is analyzed in Figure 3, and the results in the main paper are averaged over three independent runs. Upon acceptance, we will release our codebase, including scripts for data preparation, converter training, distillation, and evaluation, together with configuration files and random seeds, to facilitate exact reproduction of our results.

REFERENCES

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

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, volume 2024, pp. 21246–21263, 2024.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 535–541, 2006.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Xinghao Chen, Zhijing Sun, Guo Wenjin, Miaoran Zhang, Yanjun Chen, Yirong Sun, Hui Su, Yijie Pan, Dietrich Klakow, Wenjie Li, et al. Unveiling the key factors for distilling chain-of-thought reasoning. In *Findings of the association for computational linguistics: aCL 2025*, pp. 15094–15119, 2025.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Chengwei Dai, Kun Li, Wei Zhou, and Songlin Hu. Improve student's reasoning generalizability through cascading decomposed cots distillation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 15623–15643, 2024.

Jiayu Ding, Lei Cui, Li Dong, Nanning Zheng, and Furu Wei. Information-preserving reformulation of reasoning traces for antidistillation. *arXiv preprint arXiv:2510.11545*, 2025.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. Hotflip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 31–36, 2018.

Hao Fang, Tianyi Zhang, Tianqu Zhuang, Jiawei Kong, Kuofeng Gao, Bin Chen, Leqi Zheng, Shu-Tao Xia, and Ke Xu. Towards distillation-resistant large language models: An information-theoretic perspective. *arXiv preprint arXiv:2602.03396*, 2026.

Shaopeng Fu, Fengxiang He, Yang Liu, Li Shen, and Dacheng Tao. Robust unlearnable examples: Protecting data against adversarial learning. *arXiv preprint arXiv:2203.14533*, 2022.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.

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

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *International Conference on Learning Representations*, volume 2024, pp. 32694–32717, 2024.

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

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the association for computational linguistics: ACL 2023*, pp. 8003–8017, 2023.

Hanxun Huang, Xingjun Ma, Sarah Monazam Erfani, James Bailey, and Yisen Wang. Unlearnable examples: Making personal data unexploitable. *arXiv preprint arXiv:2101.04898*, 2021.

Jiaxin Huang, Shixiang Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. Large language models can self-improve. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 1051–1068, 2023.

Abhinav Java, Simra Shahid, and Chirag Agarwal. Towards operationalizing right to data protection. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8191–8205, 2025.

Bo Jiang. Distillguard: Evaluating defenses against llm knowledge distillation. *arXiv preprint arXiv:2603.07835*, 2026.

Wan Jiang, Yunfeng Diao, He Wang, Jianxin Sun, Meng Wang, and Richang Hong. Unlearnable examples give a false sense of security: Piercing through unexploitable data with learnable examples. In *Proceedings of the 31st ACM International Conference on Multimedia*, pp. 8910–8921, 2023.

Yujing Jiang, Xingjun Ma, Sarah Monazam Erfani, and James Bailey. Unlearnable examples for time series. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 213–225. Springer, 2024.

Sanjay Kariyappa and Moinuddin K Qureshi. Defending against model stealing attacks with adaptive misinformation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 767–775. IEEE, 2020.

Yoon Kim and Alexander M Rush. Sequence-level knowledge distillation. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pp. 1317–1327, 2016.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

Fan Li, Mengting Pan, Sijia Xu, Xiaoyang Wang, Chen Chen, and Wenjie Zhang. Answer-then-edit: Reasoning skeleton editing for anti-distillation with preserved utility. *arXiv preprint arXiv:2607.20440*, 2026.

Liunian Harold Li, Jack Hessel, Youngjae Yu, Xiang Ren, Kai-Wei Chang, and Yejin Choi. Symbolic chain-of-thought distillation: Small models can also "think" step-by-step. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2665–2679, 2023.

Pingzhi Li, Zhen Tan, Mohan Zhang, Huaizhi Qu, Huan Liu, and Tianlong Chen. Doge: Defensive output generation for llm protection against knowledge distillation. *arXiv preprint arXiv:2505.19504*, 2025.

Xinzhe Li and Ming Liu. Make text unlearnable: Exploiting effective patterns to protect personal data. In *Proceedings of the 3rd Workshop on Trustworthy Natural Language Processing (TrustNLP 2023)*, pp. 249–259, 2023.

Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Peter Belcak, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, et al. Gdpo: Group reward-decoupled normalization policy optimization for multi-reward rl optimization. *arXiv preprint arXiv:2601.05242*, 2026.

Xinwei Liu, Xiaojun Jia, Yuan Xun, Siyuan Liang, and Xiaochun Cao. Multimodal unlearnable examples: Protecting data against multimodal contrastive learning. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 8024–8033, 2024.

Yixin Liu, Chenrui Fan, Pan Zhou, and Lichao Sun. Unlearnable graph: Protecting graphs from unauthorized exploitation, 2023. URL https://arxiv.org/abs/2303.02568.

Haoyu Ma, Tianlong Chen, Ting-Kuei Hu, Chenyu You, Xiaohui Xie, and Zhangyang Wang. Undistillable: Making a nasty teacher that cannot teach students. *arXiv preprint arXiv:2105.07381*, 2021.

Xinhang Ma, William Yeoh, Ning Zhang, and Yevgeniy Vorobeychik. Protecting language models against unauthorized distillation through trace rewriting. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11307–11324, 2026.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. Teaching small language models to reason. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 1773–1781, 2023.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*, 2023.

Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Prediction poisoning: Towards defenses against dnn model stealing attacks. *arXiv preprint arXiv:1906.10908*, 2019.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019.

Jie Ren, Han Xu, Yuxuan Wan, Xingjun Ma, Lichao Sun, and Jiliang Tang. Transferable unlearnable examples. *arXiv preprint arXiv:2210.10114*, 2022.

Tom Sander, Pierre Fernandez, Alain Durmus, Matthijs Douze, and Teddy Furon. Watermarking makes language models radioactive. *Advances in Neural Information Processing Systems*, 37: 21079–21113, 2024.

Yash Savani, Asher Trockman, Zhili Feng, Yixuan Xu, Avi Schwarzschild, Alexander Robey, Marc Finzi, and Zico Kolter. Antidistillation sampling. *Advances in Neural Information Processing Systems*, 38:117800–117827, 2026.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. Distilling reasoning capabilities into smaller language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 7059–7073, 2023.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4149–4158, 2019.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Derui Wang, Minhui Xue, Bo Li, Seyit Camtepe, and Liming Zhu. Provably unlearnable data examples. *arXiv preprint arXiv:2405.03316*, 2024.

Jiashu Xu, Fei Wang, Mingyu Ma, Pang Wei Koh, Chaowei Xiao, and Muhao Chen. Instructional fingerprinting of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3277–3306, 2024a.

Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. A survey on knowledge distillation of large language models. *arXiv preprint arXiv:2402.13116*, 2024b.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Jingwen Ye, Yining Mao, Jie Song, Xinchao Wang, Cheng Jin, and Mingli Song. Safe distillation box. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 3117–3124, 2022.

Ruihan Zhang and Jun Sun. 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)*, pp. 40587–40598, 2026.

Songming Zhang, Xue Zhang, Zengkui Sun, Yufeng Chen, and Jinan Xu. Dual-space knowledge distillation for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 18164–18181, 2024a.

Yuanchi Zhang, Yile Wang, Zijun Liu, Shuo Wang, Xiaolong Wang, Peng Li, Maosong Sun, and Yang Liu. Enhancing multilingual capabilities of large language models through self-distillation from resource-rich languages. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11189–11204, 2024b.

Zhisheng Zhang and Pengyang Huang. Hiddenspeaker: Generate imperceptible unlearnable audios for speaker verification system. In *2024 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2024.

Chengshuai Zhao, Pingchuan Ma, Dawei Li, Bohan Jiang, Zhiyuan Yu, Zhen Tan, and Huan Liu. Textcloak: Thwarting unauthorized llm exploitation via rl-driven unlearnable text. *arXiv preprint arXiv:2607.28862*, 2026a.

Chengshuai Zhao, Zhen Tan, Dawei Li, Zhiyuan Yu, and Huan Liu. To see is not to learn: Protecting multimodal data from unauthorized fine-tuning of large vision-language model. *arXiv preprint arXiv:2605.14291*, 2026b.

Zhengyue Zhao, Jinhao Duan, Xing Hu, Kaidi Xu, Chenan Wang, Rui Zhang, Zidong Du, Qi Guo, and Yunji Chen. Unlearnable examples for diffusion models: Protect data from unauthorized exploitation, 2024. URL https://arxiv.org/abs/2306.01902.

Xin Zhou, Yi Lu, Ruotian Ma, Yujian Wei, Tao Gui, Qi Zhang, and Xuan-Jing Huang. Making harmful behaviors unlearnable for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 10258–10273, 2024.

Yifan Zhu, Lijia Yu, and Xiao-Shan Gao. Detection and defense of unlearnable examples. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17211–17219, 2024.

APPENDIX

A Comprehensive Review of Related Work **17**

B Discussion on Potential Limitations **18**

C Further Details on Methodology and Contributions **18**

D Threat Model **20**

E Training Algorithm **21**

F Experimental Details **22**

F.1 Dataset Statistics 22

F.2 Baseline Details 23

F.3 Implementation Details 24

F.4 Adaptive Defenses and Distillation Methods 25

G Additional Experimental Results **26**

G.1 Standard Deviations of Main Results 26

G.2 Complete Ablation Results 26

G.3 Sensitivity of Batch Size 27

G.4 Additional Robustness Results 27

G.5 Serving-Time Overhead 28

G.6 Human and LLM-as-a-Judge Evaluation of Response Quality 28

H Broader Impact **30**

I Code **30**

J Case Studies **30**

A Comprehensive Review of Related Work

This section provides an extended discussion of the related work summarized in Section 2.

Knowledge Distillation. Knowledge distillation (KD) transfers knowledge from a teacher model to a typically smaller student, originally through model compression and soft predictive distributions (Buciluǎ et al., 2006; Hinton et al., 2015). With the emergence of LLMs, distillation has increasingly relied on generated responses and intermediate reasoning as supervision (Hsieh et al., 2023; Shridhar et al., 2023; Mukherjee et al., 2023). Representative black-box approaches distill rationales or chain-of-thought to transfer reasoning capabilities (Magister et al., 2023; Li et al., 2023; Dai et al., 2024; Chen et al., 2025), while white-box methods further exploit teacher distributions through improved divergence objectives (Gu et al., 2024; Zhang et al., 2024a). Beyond conventional KD settings, self-distillation uses a model's own generations to improve subsequent training (Zhang et al., 2024b; Huang et al., 2023), and on-policy distillation queries the teacher on student-generated trajectories to reduce the train-inference distribution mismatch (Agarwal et al., 2024). We refer readers to Xu et al. (2024b) for a comprehensive survey. Our work focuses on defending against unauthorized sequence-level black-box KD for LLM IP protection.

Anti-Distillation and Model IP Protection. Protecting model intellectual property has motivated both post-hoc verification and proactive protection mechanisms. Watermarking and fingerprinting embed or identify model-specific signals that can establish provenance or ownership after unauthorized reuse (Sander et al., 2024; Xu et al., 2024a). Such methods facilitate attribution but do not necessarily prevent the underlying capability from being transferred. Proactive anti-distillation methods instead modify released outputs to directly reduce their value as training data. Early efforts in computer vision operate on the teacher's output logits: prediction poisoning perturbs the returned posteriors to mislead the attacker's gradient (Orekondy et al., 2019), adaptive misinformation returns incorrect predictions on suspicious queries (Kariyappa & Qureshi, 2020), the nasty teacher preserves its own accuracy while producing logits that mislead students (Ma et al., 2021), and the safe distillation box restricts distillation to authorized users (Ye et al., 2022). Such logit-level defenses do not directly apply to LLM APIs that expose only generated text, so recent anti-distillation methods for LLMs act on the generated sequences. Antidistillation Sampling (ADS) perturbs the teacher's next-token sampling distribution toward sequences that degrade a proxy student's learning (Savani et al., 2026), whereas DOGe adversarially adapts the teacher's output layer to generate responses that remain useful to users but are less effective for distillation (Li et al., 2025). Related approaches rewrite or reformulate reasoning traces to disrupt their training utility while preserving semantic information (Ding et al., 2025; Ma et al., 2026; Li et al., 2026). Fang et al. (2026) further analyze distillation resistance from an information-theoretic perspective, and Jiang (2026) benchmark existing defenses against LLM distillation. Existing proactive approaches share the premise that protection requires making teacher responses difficult to fit. We extend this line of work by revisiting its premise and directly impeding the knowledge-transfer process.

Unlearnable Examples. Unlearnable examples (UEs) provide a data-centric defense against unauthorized model training and were initially proposed in computer vision (Huang et al., 2021). A representative construction uses error-minimizing perturbations to create simple predictive shortcuts so that models can rapidly fit the protected training examples but consequently fail to learn features that generalize to clean data, which exploits the well-documented tendency of deep networks toward shortcut learning (Geirhos et al., 2020). Subsequent work has studied the robustness (Fu et al., 2022), transferability (Ren et al., 2022), and theoretical properties (Wang et al., 2024) of such protections, as well as their vulnerabilities to adaptive learners and detection (Jiang et al., 2023; Zhu et al., 2024). The underlying idea has also been extended beyond standard image classification, including multimodal contrastive learning (Liu et al., 2024; Zhao et al., 2026b), diffusion models (Zhao et al., 2024), speaker verification (Zhang & Huang, 2024), time-series analysis (Jiang et al., 2024), and graph representation learning (Liu et al., 2023). More recently, UEs have been explored for textual data. Early work constructs discrete patterns that protect personal information in text classification with pretrained models (Li & Liu, 2023), and RegText introduces spurious textual correlations to protect data from both smaller language models and modern LLMs (Java et al., 2025). Related efforts make harmful behaviors unlearnable during LLM fine-tuning (Zhou et al., 2024) and further investigate UEs for protecting data from LLM exploitation (Zhang & Sun, 2026; Zhao et al., 2026a). Although UEs were introduced for data protection, we draw on their core principle to explore their potential for anti-distillation.

B Discussion on Potential Limitations

While DistilShield provides effective protection against knowledge distillation across diverse tasks and architectures, it shares several limitations that are inherent to proactive anti-distillation or lie outside the scope of this work. We discuss them below to contextualize our contributions.

- **Training Cost of Simulating Distillation:** Learning a defense that anticipates how a student learns requires simulating the distillation process. DistilShield trains the converter by repeatedly updating a proxy student and evaluating its trajectory, which takes about five hours per dataset on one GPU (Table 5). This cost is common to learning-based defenses that model the student, such as error-minimizing UEs (Huang et al., 2021) and ADS (Savani et al., 2026), and is higher than that of heuristic method. However, the cost is incurred only once and amortized across all subsequent queries, and serving requires only one converter forward pass (Appendix G.5).
- **Reliance on Provider-Side Deployment:** Like all proactive anti-distillation methods (Savani et al., 2026; Li et al., 2025; Ma et al., 2026), DistilShield is effective only if the model provider adopts it and controls the serving pipeline through which responses are released. The protection therefore depends on the provider applying it consistently across every access point: if unprotected responses of the same model are available through another deployment, a third-party host, or an earlier release, the adversary can distill from them instead. For the same reason, the defense cannot protect responses that were released before it was deployed, nor models whose weights are openly available, since anyone can then generate unprotected responses locally. These requirements are inherent to output-level anti-distillation including our method.
- **Scope of the Protected Channel:** DistilShield protects the knowledge exposed through the responses released by a model API, which is the setting of black-box distillation. It does not protect against threats outside this channel, such as leakage of model weights, or adversaries that obtain the same capability from other sources, including human annotation or other teacher models. Like any output-level defense, it increases the cost of extracting the teacher's capability through its responses and it is complementary to access control and to post-hoc verification methods such as watermarking and fingerprinting (Sander et al., 2024; Xu et al., 2024a).
- **Inherent Trade-off between Protection and Transparency:** Any defense that modifies released responses changes what users see. Although DistilShield preserves the final answer and the semantics of the reasoning (Table 6), the released reasoning trace is a rewritten version rather than the teacher's original reasoning. Users who inspect the reasoning, e.g., to audit or interpret the model's behavior, therefore receive a reformulated trace. This tension between protecting the model and exposing its reasoning is inherent to output-level anti-distillation, and we recommend disclosing the use of such defenses to users (Appendix H).

C Further Details on Methodology and Contributions

This section provides additional details on key aspects of our methodology and positions our contributions with respect to unlearnable examples and existing anti-distillation methods.

- **Contribution Regarding Unlearnable Examples:** Our work is not a direct transfer of UEs to text. We borrow their core principle, an easy-to-fit shortcut that does not generalize to the clean distribution (Huang et al., 2021; Geirhos et al., 2020), and apply it to a problem with a different protected object and different constraints. UEs protect a training *dataset* by perturbing the *inputs* of examples for a model that is trained from scratch. Anti-distillation instead protects a *capability* that is exposed through the model's *outputs*: the adversary chooses the queries, the perturbation must live inside the released reasoning trace, and the student is a pretrained LLM that already encodes substantial prior knowledge. The UE-extended baselines in Appendix F.2 (Random-Prefix, Artificial Pattern, Text-MinMin, and MEM) instantiate the direct transfer and barely reduce the student below its zero-shot accuracy (Table 1). Our contribution is to identify why this transfer fails, namely the coexistence of a shortcut with genuine capability acquisition and the mismatch between teacher-forced contexts and student rollouts, and to introduce shortcut binding (Section 4.2) and shortcut consolidation (Section 4.3) to address them.
- **Contribution Regarding Existing Anti-Distillation Methods:** Existing proactive defenses (Savani et al., 2026; Li et al., 2025; Ding et al., 2025; Ma et al., 2026) share the premise that pro-

tection requires making teacher responses difficult for the student to fit. DISTILSHIELD inverts this premise: protected responses remain easy to fit, yet the induced updates transfer little of the teacher's capability. This distinction has a direct consequence for utility. A response that is hard to fit is typically also a degraded response; for instance, ADS and DOGe decode the entire response, including the final answer, from a modified distribution and therefore sacrifice teacher accuracy, whereas our converter copies the answer and rewrites only the reasoning, so teacher accuracy is preserved by construction. The closest baseline is Trace Rewriting (Ma et al., 2026), which also rewrites reasoning traces but relies on a fixed instruction prompt to an off-the-shelf LLM. Our Unlearnable Converter is instead a learned policy whose rewrites are selected by how a simulated student learns from them. Using the same rewriter backbone for both methods (Appendix F.2), the learned converter provides substantially stronger protection (Table 1), indicating that the gain comes from optimizing against the learning trajectory instead of rewriting itself.

- **Relationship between Shortcut Induction and Error-Minimizing Noise:** Equation 5 is the sequence-level counterpart of the error-minimizing objective in Equation 2, with three differences. (*i*) The perturbation is a natural-language rewrite generated by the converter, so the feasible set is defined by the utility constraint in Equation 11. (*ii*) The "label" is the response sequence itself, i.e., the autoregressive likelihood in Equation 1, and the answer is copied verbatim so the shortcut resides entirely in the reasoning trace. (*iii*) The inner problem is a proxy student's distillation trajectory rather than training from scratch. Our warm-up experiment (Figure 1A) confirms that such rewrites are indeed fitted quickly, but the ablation in Table 3 shows that induction alone is insufficient: removing $\mathcal{L}_{\text{align}}$ while keeping $\mathcal{L}_{\text{SKD}}$ leaves the student at 37.2% on GSM8K, whereas removing both yields 27.9%. An unbound shortcut can thus even reinforce the student's prior capability, which is exactly the coexistence failure illustrated in Figure 1B. Induction therefore serves as a source of easy-to-fit structure that binding then redirects away from capability transfer.
- **Interpretation and Computation of the Shortcut Binding Objective ($\mathcal{L}_{\text{align}}$):** Shortcut binding constrains the *direction* of the student update rather than the *value* of any loss. By Equation 7, the normalized inner product measures the first-order change in the clean KD loss caused by one protected update, expressed in units of the clean-gradient energy. Minimizing it removes the component of the protected update that would also fit the clean responses, so the student can reach low training loss on protected responses without moving toward the teacher's clean behavior. In terms of computation, the objective is evaluated at every inner step s of the simulated trajectory from the gradients $\widetilde{g}_s$ and g_s (Algorithm 2), where $\widetilde{g}_s$ is already required for the proxy update, so the additional cost per step is one backward pass on the clean mini-batch and one inner product. Because $\mathcal{L}_{\text{align}}$ enters ANTI-GDPO as a reward channel computed on detached gradients, it requires no second-order differentiation or Hessian-vector products through the student updates. The normalization by $\|g_s\|_2^2$ keeps the scores comparable as gradient magnitudes shrink along the trajectory, and the clean mini-batch $\mathcal{B}$ is available to the defender since it owns the teacher.
- **Shortcut Consolidation versus On-Policy Distillation:** Both involve student rollouts but play opposite roles. On-policy distillation (Agarwal et al., 2024) is an adversary's method that trains the student on its own rollouts under teacher supervision. Shortcut consolidation is a defender-side objective for the converter: after the S inner updates on protected data, the proxy student generates rollouts, and the converter is optimized to place high probability on the student's own continuations at every rollout prefix (Equation 9). The converter is thereby optimized not only on the teacher-forced prefixes of Equation 5 but also on the prefixes the student will actually visit during autonomous generation, which is what makes the induced shortcut persist beyond the exact protected trace. Empirically, this is why on-policy distillation, the strongest adversary in Table 4, gains little against the defensive teacher (55.5% vs. 55.9% under SKD). Since the converter and the proxy student share a tokenizer in our experiments, we use the token-level KL divergence; Equation 10 is a sequence-level fallback for incompatible tokenizations. Consolidation is a *training-time-only* mechanism, and no student rollouts are needed at serving time.
- **Dual Role of the Utility Constraint ($\mathcal{L}_{\text{util}}$):** Since the final answer is copied verbatim, teacher accuracy is preserved regardless of $\mathcal{L}_{\text{util}}$; the constraint governs the quality of the rewritten reasoning through a semantic-similarity term and a naturalness term based on a perplexity ratio, following the criteria used for generative textual UEs (Zhao et al., 2026a). Its second and less obvious role is to prevent reward hacking of the RL optimization. Without it, the converter can maximize the protection rewards with degenerate rewrites (e.g., garbled or near-empty reasoning) that the student simply ignores, in which case distillation is barely affected. The ablation in Table 3 confirms this: removing $\mathcal{L}_{\text{util}}$ collapses protection to near the zero-shot level (38.0% on GSM8K). The utility

constraint is therefore necessary for protection itself rather than a pure utility trade-off. Table 6 shows that the resulting rewrites lexically diverge from the originals (ROUGE-L of 0.40) while retaining high semantic similarity (BERTScore of 0.90) and even lower perplexity.

- **Contribution Regarding Reinforcement Learning and ANTI-GDPO:** We resort to RL because the overall objective in Equation 12 involves discrete sampling of rewrites and nested optimization of the proxy student. Treating each objective as a reward channel evaluated on a detached simulated trajectory sidesteps differentiation through the student updates. The distinction from standard GRPO (Shao et al., 2024) lies in how the channels are aggregated. GRPO forms a single scalar reward per rollout and standardizes it within the group. Our four channels have heterogeneous scales and variances (a negative log-likelihood, a normalized inner product, a KL divergence, and a bounded similarity score), so summing them before standardization allows the largest or noisiest channel to dominate the advantage, and the coefficients λ_m then compensate for numerical scale instead of expressing the intended trade-off. Following GDPO (Liu et al., 2026), ANTI-GDPO standardizes each channel independently before aggregation (Equation 14). The "w/ GRPO" variant in Table 3 isolates exactly this design choice with the same rollouts and rewards, and it weakens protection by 1.8 points on average. The clipped surrogate and the KL regularization toward the initial converter follow standard practice.
- **Role of the Proxy Student and Transfer to Unknown Students:** The defender does not know the adversary's student. DISTILSHIELD trains the converter against a single proxy student (Qwen2.5-3B) and applies it unchanged to six students spanning three model families and 0.5B-3B parameters, as well as two teachers (Table 2), with consistent protection. Figure 3 further shows that protection is stable across proxy students of different families and sizes. This mirrors the transferability of UEs across architectures (Ren et al., 2022): the implanted shortcut is a property of the released text, e.g., the predictable structure highlighted in Figure 4, rather than of a particular student. Consistent with the first challenge, larger students with stronger priors retain higher accuracy, yet DISTILSHIELD still outperforms all baselines in these settings.
- **Clarification on Framework Complexity and Computational Cost:** DISTILSHIELD consists of three stages (Algorithm 1). Stage 1 builds the distillation corpus and is shared with any distillation-aware defense and with the baselines that require a proxy student (ADS, DOGe, Text-MinMin, and MEM). Stage 2 trains the converter once per dataset, about five hours on one A100 GPU (Table 5), and this cost is dominated by the proxy-student simulation. Figure 3 shows that a small group size, a short simulated trajectory, and a 3B converter already provide strong protection, so the one-time cost does not need to scale up. Stage 3 adds a single converter forward pass per response at serving time, 0.134 s and about 20% of the teacher's own generation time on average (Appendix G.5), with no proxy student, no gradient computation, and no rollout. This contrasts with decoding-time defenses that consult a proxy student at every decoding step. The teacher weights remain unchanged throughout, so the converter acts as a plug-and-play serving-time wrapper that can be attached to or detached from an existing deployment.

D THREAT MODEL

We consider a model provider (the *defender*) that serves a teacher LLM $\mathcal{T}$ through an API returning generated responses, and an *adversary* that seeks to reproduce the teacher's capability in its own student $\mathcal{S}$ by distilling from the released responses. Section 3.3 formalizes the resulting objective; here we make explicit the goals, capabilities, and knowledge assumed for both parties, which correspond to the standard black-box distillation setting studied by prior anti-distillation work (Savani et al., 2026; Li et al., 2025; Ma et al., 2026).

Defender. The defender's goal is to minimize the capability transferred to *any* student distilled from its responses while keeping the released responses useful to legitimate users, i.e., Equation 3 with a small utility budget ε. The defender owns the teacher and controls the serving pipeline, so it can transform every response before release. It has access to the original responses (x, r, a) of its own teacher, can train a proxy student, and can simulate the distillation process offline. The defender does *not* know the adversary's identity, the architecture, scale, or initialization of the adversary's student, the queries the adversary will issue, the adversary's training recipe, or whether the adversary will apply countermeasures. Consequently, DISTILSHIELD trains the converter once against a single proxy student on the provider's own data and applies it unchanged to every query (Algorithm 1); the

results across six students, two teachers, and different proxy students (Table 2 and Figure 3) evaluate this assumption. The defender leaves the teacher's weights untouched and never alters the final answer, so teacher accuracy, as computed on the released responses, is preserved by construction.

Adversary. The adversary's goal is to maximize the held-out accuracy $\mathrm{Perf}(\tilde{\mathcal{S}}; \mathcal{D}_{\mathrm{eval}})$ of its student. It has black-box query access to the API and observes each released response in full, including the reasoning trace and the final answer. It chooses the queries freely; in our experiments, it queries the training questions of each dataset (Appendix F.1), and we do not rely on query budgets, rate limits, or detection of suspicious queries. The adversary owns a pretrained student of arbitrary family and scale and fully controls its training. In the primary setting, it fine-tunes the student with sequence-level KD (Equation 1) on the collected responses, which is the standard black-box distillation pipeline. The adversary may be *aware* that a defense is deployed and may apply countermeasures before or during distillation; we evaluate adversarial training, repetitive querying, and paraphrasing with a strong external LLM (Table 4, Appendix F.4). The adversary does *not* have access to the teacher's weights, the converter's weights, the proxy student, or the unprotected responses of the same teacher for the same queries.

Stronger Adversaries beyond the Black-Box Setting. To stress-test DISTILSHIELD, we additionally consider an adversary that exceeds the black-box model by accessing the token-level output distributions of the defensive teacher and applying off-policy and on-policy distillation (Section 5.5, Table 4). This setting favors the adversary, since it provides denser supervision and, for on-policy distillation, trains the student on its own rollouts. We include it because it directly tests whether the induced shortcut persists under the student's own rollout distribution, which is the purpose of shortcut consolidation (Section 4.3), rather than because we assume this access is typical.

Out of Scope. Consistent with output-level anti-distillation, the following threats are outside our model and are discussed further in Appendix B: leakage or theft of the teacher's weights; capability acquired from sources other than the protected API, such as human annotation, other teacher models, or openly released models; and unprotected responses of the same teacher obtained through other access points or from releases that predate the defense. DISTILSHIELD raises the cost of extracting the teacher's capability through its released responses and is complementary to access control and to post-hoc verification such as watermarking and fingerprinting (Sander et al., 2024; Xu et al., 2024a).

E TRAINING ALGORITHM

Algorithm 1 presents the overall procedure of DISTILSHIELD. Given a teacher $\mathcal{T}$, it first builds a distillation corpus by querying the teacher, then trains the Unlearnable Converter π_ϕ with ANTI-GDPO (Algorithm 2), and finally wraps the unchanged teacher with the trained converter to obtain the defensive teacher $\tilde{\mathcal{T}} = \mathcal{A} \circ \mathcal{T}$ (Section 3.3). At serving time, each teacher response is protected by a single converter forward pass that rewrites the reasoning trace while copying the answer, so the teacher itself and its task utility remain intact.

Algorithm 1 DISTILSHIELD: Constructing the Defensive Teacher

Require: Teacher $\mathcal{T}$ with output distribution $p_{\mathcal{T}}$; query set $\mathcal{D}_x = \{x_i\}_{i=1}^N$; initial converter π_{ref}; proxy student initialization θ_0; ANTI-GDPO hyperparameters (Algorithm 2)
Ensure: Defensive teacher $\tilde{\mathcal{T}} = \mathcal{A} \circ \mathcal{T}$

// Stage 1: Distillation corpus construction
Collect $\mathcal{D}_{\mathcal{T}} = \{(x_i, y_i) : y_i \sim p_{\mathcal{T}}(\cdot \mid x_i)\}_{i=1}^N$ with $y_i = (r_i, a_i)$ ▷ Section 3.1
// Stage 2: Unlearnable Converter training (one-time cost)
$\pi_\phi \leftarrow$ ANTI-GDPO$(\mathcal{D}_{\mathcal{T}}, \pi_{\mathrm{ref}}, \theta_0)$ ▷ Algorithm 2
// Stage 3: Serving-time protection
function $\tilde{\mathcal{T}}$(query x)
    Generate the original response $y = (r, a) \sim p_{\mathcal{T}}(\cdot \mid x)$
    Rewrite the reasoning trace $\tilde{r} \sim \pi_\phi(\cdot \mid x, r, a)$ and set $\tilde{y} = (\tilde{r}, a)$ ▷ Eq. 4
    return protected response $\tilde{y}$
end function

Algorithm 2 summarizes the training procedure of the Unlearnable Converter with ANTI-GDPO. Each outer iteration proceeds in three stages. (*i*) *Group rollout*: the frozen rollout policy $\pi_{\phi_{\text{old}}}$ samples K candidate rewrites for every teacher response in the mini-batch, each yielding a protected candidate batch (Equation 4). (*ii*) *Reward evaluation*: each candidate is scored by simulating the student's distillation trajectory from a shared proxy initialization. Along the trajectory we accumulate the shortcut-induction loss (Equation 5) and the gradient anti-alignment loss (Equation 8); after the inner updates, on-policy rollouts from the updated proxy provide the consolidation loss (Equation 9), and the utility loss (Equation 11) is computed directly on the rewrites. Because all reward channels are evaluated on detached trajectories, no second-order differentiation through the student updates is required. (*iii*) *Converter update*: the four reward channels are standardized independently within the candidate group and aggregated into decoupled advantages (Equation 14), which drive the clipped policy update of Equation 16. At deployment, only the trained converter π_ϕ is retained: protecting a response requires a single forward pass and no student simulation.

Algorithm 2 ANTI-GDPO: Training the Unlearnable Converter of DISTILSHIELD

Require: Distillation corpus $\mathcal{D}_\mathcal{T} = \{(x_i, y_i)\}_{i=1}^N$ with $y_i = (r_i, a_i)$; initial converter π_{ref}; proxy student initialization θ_0; group size K; inner steps S; coefficients $\{\lambda_m\}_{m\in\mathcal{M}}$, $\mathcal{M} = \{\text{KD}, \text{align}, \text{policy}, \text{util}\}$; clipping radius τ; KL weight β

Ensure: Trained converter π_ϕ

Initialize the converter $\pi_\phi \leftarrow \pi_{\text{ref}}$
for each outer iteration **do**
  $\phi_{\text{old}} \leftarrow \phi$; sample a clean mini-batch $\mathcal{B} = \{(x_i, r_i, a_i)\}_{i=1}^{|\mathcal{B}|} \subset \mathcal{D}_\mathcal{T}$
  // Stage 1: Group rollout
  for $k = 1, \ldots, K$ **do**
    Sample rewrites $\widetilde{r}_i^{(k)} \sim \pi_{\phi_{\text{old}}}(\cdot \mid x_i, r_i, a_i)$ and set $\widetilde{y}_i^{(k)} = (\widetilde{r}_i^{(k)}, a_i)$ ▷ Eq. 4
    Form the protected candidate batch $\widetilde{\mathcal{B}}^{(k)} = \{(x_i, \widetilde{y}_i^{(k)})\}_{i=1}^{|\mathcal{B}|}$
  end for
  // Stage 2: Reward evaluation via proxy-student simulation
  for $k = 1, \ldots, K$ **do**
    Reset the proxy student: $\theta_0^{(k)} \leftarrow \theta_0$
    for $s = 0, \ldots, S-1$ **do**
      Compute $\widetilde{g}_s = \nabla_\theta \mathcal{L}_{\text{SKD}}(\theta_s^{(k)}; \widetilde{\mathcal{B}}^{(k)})$ and $g_s = \nabla_\theta \mathcal{L}_{\text{SKD}}(\theta_s^{(k)}; \mathcal{B})$ ▷ Eq. 6
      Update the proxy student: $\theta_{s+1}^{(k)} \leftarrow \theta_s^{(k)} - \eta_s \widetilde{g}_s$
    end for
    $\mathcal{L}_{\text{KD}}^{(k)} \leftarrow \mathcal{L}_{\text{SKD}}(\theta_S^{(k)}; \widetilde{\mathcal{B}}^{(k)})$ ▷ Shortcut Induction, Eq. 5
    $\mathcal{L}_{\text{align}}^{(k)} \leftarrow \frac{1}{S}\sum_{s=0}^{S-1} \langle \widetilde{g}_s, g_s \rangle / (\|g_s\|_2^2 + \epsilon)$ ▷ Shortcut Binding, Eq. 8
    Sample on-policy rollouts $\widehat{y}_i \sim p_\mathcal{S}(\cdot \mid x_i; \theta_S^{(k)})$ to form $\widehat{\mathcal{B}}^{(k)}$
    $\mathcal{L}_{\text{policy}}^{(k)} \leftarrow$ rollout-prefix KL on $\widehat{\mathcal{B}}^{(k)}$ ▷ Shortcut Consolidation, Eq. 9 or 10
    $\mathcal{L}_{\text{util}}^{(k)} \leftarrow$ semantic and naturalness penalty on $\{(r_i, \widetilde{r}_i^{(k)})\}$ ▷ Utility Constraints, Eq. 11
    Set channel rewards $R_m^{(k)} \leftarrow -\mathcal{L}_m^{(k)}$ for all $m \in \mathcal{M}$
  end for
  // Stage 3: Reward-decoupled converter update
  Standardize channels and aggregate: $\widehat{A}^{(k)} = \sum_{m\in\mathcal{M}} \lambda_m \widehat{A}_m^{(k)}$ ▷ Eq. 14
  Update ϕ by ascending $\nabla_\phi \mathcal{J}_{\text{ANTI-GDPO}}(\phi)$ with clipping τ and KL weight β ▷ Eq. 16
end for
return π_ϕ

F EXPERIMENTAL DETAILS

F.1 DATASET STATISTICS

We evaluate on six public benchmarks spanning mathematical reasoning, commonsense reasoning, and code generation. For each dataset, the adversary queries the teacher with the training questions

to build the distillation corpus, and the student is evaluated on the held-out test questions. All datasets are used in accordance with their licenses.

Mathematical Reasoning:

- **GSM8K** (Cobbe et al., 2021): A dataset of 8,500 linguistically diverse grade-school math word problems written by human problem writers, each requiring two to eight steps of elementary arithmetic. It provides 7,473 training and 1,319 test problems; we use 7,473 training problems as distillation queries and 1,319 test problems for evaluation. Licensed under MIT.
- **MATH** (Hendrycks et al., 2021): A competition-level dataset of 12,500 problems drawn from AMC, AIME, and similar competitions, covering seven subjects and five difficulty levels. It provides 7,500 training and 5,000 test problems; we use 7,500 training problems as distillation queries and 5,000 test problems for evaluation. Licensed under MIT.

Commonsense Reasoning:

- **ARC-Challenge (ARC-C)** (Clark et al., 2018): The challenge subset of the AI2 Reasoning Challenge, consisting of 2,590 grade-school science multiple-choice questions that retrieval-based and word co-occurrence methods fail to answer. It provides 1,119 training, 299 development, and 1,172 test questions; we use 1,119 training questions as distillation queries and 1,172 test questions for evaluation. Licensed under CC BY-SA 4.0.
- **CommonsenseQA (CSQA)** (Talmor et al., 2019): A dataset of 12,102 five-way multiple-choice questions constructed from ConceptNet relations, which require commonsense knowledge beyond the question text. It provides 9,741 training, 1,221 development, and 1,140 test questions; since the test labels are not public, we use 9,741 training questions as distillation queries and the 1,221 development questions for evaluation. Licensed under MIT.

Code Generation:

- **HumanEval (HEval)** (Chen et al., 2021): A benchmark of 164 hand-written Python programming problems, each specified by a function signature, a docstring, and unit tests for functional correctness. Since it provides no training split, we randomly split the problems into 114 distillation queries and 50 test problems, and report the pass@1 accuracy. Licensed under MIT.
- **MBPP** (Austin et al., 2021): A collection of 974 crowd-sourced entry-level Python problems with natural-language descriptions and three test cases each. We use the standard split: 474 problems (374 training, 90 validation, and 10 prompt) as distillation queries and 500 test problems for evaluation, reporting pass@1. Licensed under CC BY 4.0.

F.2 Baseline Details

We compare DistilShield with three groups of baselines. Unless otherwise stated, all methods share the same teacher, distillation queries, student, and distillation procedure (Equation 1); they differ only in how the teacher responses are generated or transformed before being released to the adversary. For methods that require a proxy student or rewriter, we use the same model as DistilShield, and we otherwise follow the hyperparameters recommended in the original papers.

(1) Control: These two references show the student's performance before and after distillation:

- **Zero-shot** (Kojima et al., 2022): The student is evaluated directly without distillation. It reflects the capability that the student already inherits from pretraining and serves as the reference for how much capability distillation transfers.
- **Sequence-level Knowledge Distillation (SKD)** (Kim & Rush, 2016): The student is fine-tuned on the responses of the vanilla teacher by minimizing the negative log-likelihood in Equation 1. It represents distillation from an unprotected teacher, and the gap between SKD and Zero-shot measures the capability transferred from the teacher.

(2) Anti-distillation Methods: These methods are designed specifically for LLM anti-distillation and make teacher responses difficult for the student to fit, either by modifying the teacher's decoding or by rewriting its responses:

- **Antidistillation Sampling (ADS)** (Savani et al., 2026): ADS modifies the teacher's next-token sampling distribution at every decoding step. It adds a finite-difference estimate of how much each candidate token would increase a proxy student's downstream loss if the proxy were trained on it, so that sampling favors tokens that poison distillation. A coefficient trades off protection against teacher utility. Because the whole response, including the final answer, is sampled from the modified distribution, ADS can reduce the teacher's own accuracy.
- **DOGe** (Li et al., 2025): Defensive Output Generation fine-tunes only the final linear layer of the teacher with an adversarial objective. The objective preserves the teacher's task performance while maximizing the divergence between the output distributions of the teacher and proxy students on intermediate reasoning tokens. Protected responses are then decoded from the adapted teacher, which may also alter the final answers.
- **Trace Rewriting** (Ma et al., 2026): An auxiliary LLM rewrites the teacher's reasoning trace to degrade its value as training data while preserving answer correctness and semantic coherence. We adopt the instruction-based rewriting variant with the same prompt from the original paper and utilize same rewrite model for both DISTILSHIELD and this baseline for a fair comparison. Since the final answer is perserved through prompt, the teacher's accuracy is largely retained.

(3) UE-extended Methods: Unlearnable examples were originally developed to protect training data in classification, where a perturbation injected into the input induces a spurious shortcut between the input and the label (Section 3.2). We adapt them to anti-distillation by treating the teacher response as the carrier of the perturbation and the final answer as the label.

- **Random-Prefix**: A random token sequence is prepended to each reasoning trace, with 3 tokens sampled uniformly from the vocabulary independently for each response. As the textual counterpart of the random-noise baseline in the UE literature (Huang et al., 2021), it tests whether an unoptimized surface perturbation suffices to suppress distillation.
- **Artificial Pattern** (Li & Liu, 2023): Li & Liu (2023) show that the modifications found by error-minimizing optimization on text reduce to simple patterns, and that injecting these patterns directly makes text unlearnable without costly bilevel optimization or knowledge of the target model. We insert the same patterns as in the original paper into each reasoning trace.
- **Text-MinMin** (Huang et al., 2021): We adapt error-minimizing noise (Equation 2) to text. Since text is discrete, the inner minimization over the perturbation is solved by gradient-guided token substitution on the proxy student and alternated with proxy-student updates on the perturbed responses, following the min-min bilevel formulation. Substitutions are applied to the entire response under a budget of 10 % of the tokens, and may therefore alter the final answer, which introduces the slight drop in teacher accuracy in Table 1.
- **Multi-step Error Minimization (MEM)** (Liu et al., 2024): MEM protects image-text pairs from multimodal contrastive learning by jointly optimizing image noise and a short text trigger over multiple error-minimization steps, updating the trigger with HotFlip (Ebrahimi et al., 2018). We keep its text branch: a learnable trigger of L tokens is inserted into each reasoning trace and optimized to minimize the proxy student's distillation loss, alternating with proxy-student updates. MEM-3 and MEM-5 denote trigger lengths of $L = 3$ and $L = 5$ tokens, respectively.

F.3 IMPLEMENTATION DETAILS

All experiments can be conducted using on signle NVIDIA 80GB A100 GPU, to accelerate the experiments, we use 32 NVIDIA 80GB A100 GPUs. We implemented the pipeline with PyTorch, Hugging Face Transformers, TRL, PEFT, and vLLM. DISTILSHIELD follows the three-stage pipeline of Algorithm 1, and every defense is evaluated with the same distillation and evaluation protocol:

Distillation Corpus Construction: The teacher is queried with the training questions of each dataset (Appendix F.1) to build $\mathcal{D}_{\mathcal{T}}$.

- Teachers: DeepSeek-R1-Distill-Qwen-7B and Qwen3-8B
- Decoding: temperature 0.6, top-p 0.95
- Maximum Generation Length: 8,192 tokens

Converter Training with ANTI-GDPO: The Unlearnable Converter is trained once per dataset with Algorithm 2. Because the converter and the proxy student share the same tokenizer, the consolidation loss uses the token-level KL divergence in Equation 9.

- Converter Initialization π_{ref}: Qwen2.5-3B
- Converter Parameterization: LoRA adapter (rank 32, $\alpha = 64$, dropout 0.05, all linear layers)
- Proxy Student Initialization θ_0: Qwen2.5-3B
- Proxy Student Parameterization: LoRA adapter (rank 32, $\alpha = 64$, dropout 0.05, all linear layers)
- Batch Size $b = |\mathcal{B}|$: 1
- Group Size K: 4
- Inner Steps S: 1
- Inner Learning Rate η_s: 1×10^{-4}
- Objective Coefficients $(\lambda_{\text{KD}}, \lambda_{\text{align}}, \lambda_{\text{policy}}, \lambda_{\text{util}})$: (1.0, 0.2, 1.0, 0.1)
- Clipping Radius τ: 0.2
- KL Regularization Coefficient β: 0.01
- Stability Constant ϵ: 10^{-8}
- Optimizer: AdamW
- Learning Rate: 1×10^{-5}
- Training Steps: 250
- Rollout Temperature of $\pi_{\phi_{\text{old}}}$: 1.0
- Utility Models: SBERT (all-MiniLM-L6-v2) for $e(\cdot)$ and Qwen2.5-0.5B for PPL in Equation 11

Serving-Time Protection: At deployment, only the trained converter is retained. Each teacher response is protected by a single converter forward pass that rewrites r_i and copies a_i (Equation 4), with greedy decoding. No proxy student is involved at this stage.

Student Distillation. To simulate the adversary, each student is fine-tuned on the responses released by the (defensive) teacher with the SKD objective in Equation 1. The same recipe is used for SKD, every baseline, and DISTILSHIELD, so that differences in student accuracy come only from the released responses.

- Fine-tuning: LoRA adapter (rank 32, $\alpha = 64$, dropout 0.1, all linear layers)
- Optimizer: AdamW (weight decay 0.01)
- Learning Rate: 5×10^{-6} with cosine schedule (10% linear warmup)
- Batch Size: 128 (per-device batch size 1 with 128 gradient-accumulation steps)
- Epochs: 3
- Maximum Sequence Length: 8,192 tokens

Evaluation. Students are evaluated on the held-out test questions with greedy decoding and the zero-shot chain-of-thought prompt (Kojima et al., 2022). For math reasoning, a response is correct if the extracted final answer matches the ground truth after normalization. For commonsense reasoning, the predicted option letter must match the gold option. For code generation, we report pass@1 by executing the generated program against the provided unit tests. Teacher accuracy is computed on the responses released to users, i.e., after protection. All results are averaged over three independent runs with different random seeds (Appendix G.1).

F.4 ADAPTIVE DEFENSES AND DISTILLATION METHODS

This section describes the adversaries evaluated in Table 4 and Table 11. In each setting, the adversary distills a student from either the vanilla or the defensive teacher, and all other settings follow Appendix F.3. We consider two types of stronger adversaries: (*i*) adaptive adversaries that are aware of a possible defense and try to neutralize it before or during distillation, and (*ii*) adversaries that go beyond the black-box setting and use stronger distillation methods.

(1) Adaptive Defenses: The adversary applies one of the following countermeasures on top of SKD:

- **Adversarial Training**: Adversarial training is the most effective known countermeasure against unlearnable examples in vision (Fu et al., 2022; Madry et al., 2018), because it prevents the model from relying on easy-to-fit shortcuts. During distillation, the student minimizes the SKD loss on worst-case perturbations of the input token embeddings, which are found by $P = 3$-step projected gradient ascent within an ℓ_∞ ball of radius $\epsilon_{\text{adv}} = 0.01$ and regenerated at each training step. The radius and the step size (0.0033) are relative to the average token-embedding norm $\bar{e}$ of the batch: each coordinate is bounded by $\epsilon_{\text{adv}}\bar{e}/\sqrt{d}$, so that the ℓ_2 norm of the perturbation of a token is at most $\epsilon_{\text{adv}}\bar{e}$. The perturbation is applied to all input tokens, including the prompt. This radius was calibrated to raise the student's training loss by about 0.47 nats: smaller radii leave training close to plain SKD, and larger ones prevent the student from learning.
- **Repetitive Querying**: The adversary queries the defensive teacher $M = 2$ times for each training question and fine-tunes the student on all collected responses. This increases the diversity of the released reasoning traces, so that a shortcut implanted in any single response may be diluted. The second response is obtained by sampling the same converter using (temperature 0.7).
- **Paraphrasing**: Before distillation, the adversary rewrites with gpt-5.6-luna (OpenAI API), which is instructed to "paraphrase the following solution without changing its meaning." Paraphrasing aims to remove surface-level shortcut patterns before the student can learn them.

(2) Distillation Methods: This setting is an extreme case in which the adversary gains white-box access to the teacher's token-level output distributions. We use Qwen3-8B as the teacher and Qwen3-0.6B as the student, which share a vocabulary and therefore allow token-level supervision. For the defensive teacher, the token-level distribution is that of the defensive teacher $\tilde{\mathcal{T}}$ itself.

- **Sequence-level KD (SKD)** (Kim & Rush, 2016): The black-box reference, which fine-tunes the student on the released responses with Equation 1.
- **Off-Policy Distillation** (Hinton et al., 2015): The student is trained to match the teacher's next-token distribution at every position of the released responses by minimizing the forward KL divergence $D_{\text{KL}}(p_{\tilde{\mathcal{T}}}, |, p_{\mathcal{S}})$ over teacher-generated prefixes. It provides denser supervision than SKD, but the training prefixes still come from the teacher.
- **On-Policy Distillation** (Agarwal et al., 2024): The student generates responses, and the teacher provides token-level supervision on their prefixes. Minimizing reverse KL between their distributions reduces the mismatch between teacher-forced training and autoregressive inference. This directly tests shortcut consolidation (Section 4.3).

G Additional Experimental Results

G.1 Standard Deviations of Main Results

All results in Tables 1 and 2 are averaged over three independent runs with different random seeds. Table 7 reports the corresponding standard deviations of the teacher and student accuracy on the six datasets, and Table 8 reports those across teacher and student architectures. The standard deviations are small relative to the performance gaps between DistilShield and the baselines, confirming that the improvements are statistically consistent.

G.2 Complete Ablation Results

Table 9 reports the complete ablation results across all six datasets, which are consistent with Table 3. Every objective contributes on average: removing $\mathcal{L}_{\text{SKD}}$, $\mathcal{L}_{\text{align}}$, $\mathcal{L}_{\text{policy}}$, and $\mathcal{L}_{\text{util}}$ raises the average student accuracy from 19.3% to 24.8%, 27.7%, 25.2%, and 28.9%, respectively, and removing all three protection objectives brings it close to the zero-shot level (30.9% vs. 32.8%). Since only the reasoning trace is rewritten, all w/o variants and the GRPO variant leave the teacher accuracy identical to the vanilla teacher. In contrast, rewriting the answer along with the reasoning lowers the teacher accuracy on every dataset (77.7% vs. 80.2% on average) without improving protection (20.8% vs. 19.3%), confirming the benefit of the answer-preserving design.

Table 7: Standard deviation (%) of the student accuracy in Table 1 over three runs.

Method *R1-7B → Llama-3.2-1B*	GSM8K	MATH	ARC-C	CSQA	HEval	MBPP
Zero-shot (Kojima et al., 2022)	0.13	0.18	0.17	0.29	2.40	0.38
SKD (Kim & Rush, 2016)	0.50	0.19	1.09	0.13	0.38	0.45
ADS (Savani et al., 2026)	0.45	0.74	0.10	0.19	1.02	0.24
DOGe (Li et al., 2025)	0.63	0.13	0.57	0.35	1.02	0.81
Trace Rewriting (Ma et al., 2026)	1.26	0.27	0.34	0.21	2.34	0.73
Random-Prefix	2.02	0.93	1.80	0.29	1.68	0.77
Artificial Pattern (Li & Liu, 2023)	1.48	0.16	0.53	0.36	2.34	0.42
Text-MinMin (Huang et al., 2021)	0.87	0.22	0.67	0.49	1.02	0.48
MEM-3 (Liu et al., 2024)	0.42	0.42	0.66	0.39	0.77	0.27
MEM-5 (Liu et al., 2024)	1.54	0.52	1.14	0.27	1.39	0.44
DISTILSHIELD (Ours)	0.38	0.25	0.92	0.72	1.02	0.42

Table 8: Standard deviation (%) of the student accuracy in Table 2 over three runs.

	Teacher	Student	SKD	ADS	DOGe	Ours
GSM8K	R1-7B	Qwen2.5-0.5B	0.16	0.30	0.17	0.36
		Gemma-3-1B	0.29	0.03	0.23	0.47
		Llama-3.2-1B	0.50	0.45	0.63	0.38
	Qwen3-8B	Qwen3-0.6B	0.52	0.52	0.55	0.48
		Qwen3-1.7B	0.30	0.22	1.44	0.37
		Llama-3.2-3B	0.28	1.03	0.60	0.71
ARC-C	R1-7B	Qwen2.5-0.5B	0.43	0.58	0.55	1.83
		Gemma-3-1B	0.17	0.79	0.46	0.81
		Llama-3.2-1B	1.09	0.52	0.57	0.92
	Qwen3-8B	Qwen3-0.6B	0.72	0.44	0.59	0.41
		Qwen3-1.7B	0.30	0.13	0.43	0.27
		Llama-3.2-3B	1.01	0.66	2.08	1.47
HEval	R1-7B	Qwen2.5-0.5B	2.00	1.02	1.02	0.67
		Gemma-3-1B	2.04	3.46	1.54	1.02
		Llama-3.2-1B	0.38	2.00	1.02	1.02
	Qwen3-8B	Qwen3-0.6B	3.46	3.08	2.14	1.39
		Qwen3-1.7B	0.67	1.76	2.04	3.46
		Llama-3.2-3B	0.67	0.00	0.77	0.38

G.3 SENSITIVITY OF BATCH SIZE

Table 10 studies the sensitivity of DISTILSHIELD to the batch size $b = |\mathcal{B}|$, i.e., the number of teacher responses per outer iteration of ANTI-GDPO (Algorithm 2). Overall, DISTILSHIELD is robust to b: the student accuracy stays within 33.0%-38.1% on GSM8K and 16.7%-20.7% on HumanEval, well below the 47.9% and 22.7% under SKD, while the teacher accuracy remains intact. Notably, the smallest batch ($b = 1$, our default) provides the strongest protection on both datasets, whereas larger batches bring no further gains yet increase the per-iteration cost linearly, since every candidate is evaluated on the entire mini-batch for S inner steps. We attribute this to shortcut binding: with $b = 1$, the gradient alignment in Equation 8 is measured for each response individually, whereas averaging over heterogeneous responses dilutes the per-response signal. A single response per outer iteration therefore suffices, which keeps the one-time training cost of the converter low.

G.4 ADDITIONAL ROBUSTNESS RESULTS

Table 11 complements Table 4 by evaluating the adaptive defenses (adversarial training, repetitive querying, and paraphrasing) on GSM8K with R1-7B as the teacher and Llama-3.2-1B as the student. The defensive teacher keeps the student below its counterpart distilled from the vanilla teacher under all defenses. Moreover, adversarial training and paraphrasing also substantially degrade the student distilled from the vanilla teacher, reflecting the inherent cost of these defenses to the adversary.

Table 9: Complete ablation study across six datasets. Teacher (T) and student (S) accuracy (%). w/o removes the corresponding objectives from Equation 12, and w/ replaces a design component of DISTILSHIELD with the specified variant. The best results are in **bold**.

Configuration	GSM8K		MATH		ARC-C		CSQA		HEval		MBPP		Avg.	
	T	S	T	S	T	S	T	S	T	S	T	S	T	S
Zero-shot	92.4	39.7	84.9	28.6	81.4	32.3	66.8	54.3	88.0	20.7	68.0	21.6	80.2	32.8
Full (DISTILSHIELD)	**92.4**	**21.1**	**84.9**	16.5	**81.4**	**31.5**	**66.8**	**10.2**	**88.0**	**16.7**	**68.0**	**20.1**	**80.2**	**19.3**
w/o $\mathcal{L}_{\text{SKD}}$	92.4	27.4	84.9	20.1	81.4	32.6	66.8	27.9	88.0	18.7	68.0	22.1	80.2	24.8
w/o $\mathcal{L}_{\text{align}}$	92.4	37.2	84.9	28.5	81.4	32.0	66.8	29.7	88.0	16.7	68.0	22.1	80.2	27.7
w/o $\mathcal{L}_{\text{policy}}$	92.4	30.7	84.9	28.1	81.4	34.2	66.8	20.1	88.0	18.0	68.0	20.4	80.2	25.2
w/o $\mathcal{L}_{\text{util}}$	92.4	38.0	84.9	27.8	81.4	33.7	66.8	33.0	88.0	20.0	68.0	21.1	80.2	28.9
w/o $\mathcal{L}_{\text{SKD}}, \mathcal{L}_{\text{policy}}$	92.4	31.4	84.9	25.0	81.4	35.3	66.8	14.6	88.0	19.3	68.0	21.7	80.2	24.5
w/o $\mathcal{L}_{\text{SKD}}, \mathcal{L}_{\text{align}}$	92.4	27.9	84.9	16.6	81.4	34.0	66.8	26.6	88.0	18.0	68.0	22.0	80.2	24.2
w/o $\mathcal{L}_{\text{align}}, \mathcal{L}_{\text{policy}}$	92.4	36.5	84.9	28.3	81.4	33.1	66.8	20.5	88.0	17.3	68.0	22.1	80.2	26.3
w/o $\mathcal{L}_{\text{SKD}}, \mathcal{L}_{\text{align}}, \mathcal{L}_{\text{policy}}$	92.4	32.5	84.9	27.0	81.4	32.7	66.8	54.3	88.0	18.0	68.0	21.1	80.2	30.9
w/o $\mathcal{L}_{\text{align}}, \mathcal{L}_{\text{policy}}, \mathcal{L}_{\text{util}}$	92.4	35.0	84.9	26.9	81.4	34.1	66.8	23.3	88.0	19.3	68.0	20.7	80.2	26.5
w/o $\mathcal{L}_{\text{SKD}}, \mathcal{L}_{\text{policy}}, \mathcal{L}_{\text{util}}$	92.4	37.1	84.9	24.9	81.4	33.8	66.8	16.4	88.0	20.0	68.0	21.3	80.2	25.6
w/ Convert (r, a)	90.5	24.0	82.9	**16.4**	78.8	32.5	61.7	14.3	84.7	17.4	67.4	20.3	77.7	20.8
w/ GRPO (Shao et al., 2024)	92.4	22.6	84.9	17.4	81.4	34.1	66.8	14.6	88.0	17.4	68.0	20.8	80.2	21.1

Table 10: Sensitivity of batch size b on GSM8K and HumanEval (student accuracy, %).

Dataset *R1-7B → Llama-3.2-1B*	Batch Size b				
	1	2	3	4	5
GSM8K	33.0	38.0	36.1	35.5	38.1
HumanEval	16.7	20.7	16.7	18.7	18.0

G.5 SERVING-TIME OVERHEAD

Complementing Table 5, we measure the serving-time overhead that the Unlearnable Converter adds to each teacher response. We randomly sample 1,000 queries from the six datasets and report in Table 12 the average wall-clock time per response of the vanilla teacher and of the defensive teacher, i.e., the teacher followed by one converter forward pass, on one NVIDIA 80GB A100 GPU. The converter adds 0.134 s per response, about 20% of the teacher's own generation time, since the 3B converter rewrites only the reasoning trace without any student simulation. The overhead is independent of the student and the distillation method, and can be further reduced with standard serving optimizations such as batching and speculative decoding.

G.6 HUMAN AND LLM-AS-A-JUDGE EVALUATION OF RESPONSE QUALITY

The automatic statistics in Table 6 show that the protected responses remain semantically close to the originals and natural under a reference language model. Since these metrics do not directly measure how a user perceives the released responses, we complement them with a human evaluation and an LLM-as-a-judge evaluation that compare the responses of the vanilla teacher with those of the defensive teacher.

Protocol. We randomly sample 10 test questions from each of the six datasets, yielding 60 questions and 120 responses (one vanilla and one protected response per question) generated with DeepSeek-R1-Distill-Qwen-7B as the teacher. Each response is rated on a 3-point Likert scale (1 = worst, 3 = best) along four criteria:

- **Correctness**: whether the final answer is correct and the reasoning steps that support it are valid, without factual or logical errors;
- **Naturalness**: whether the response reads like a response written by a competent model or human, without unnatural, templated, or out-of-place content;
- **Fluency**: whether the response is grammatical, coherent, and easy to follow;

Table 11: Robustness analysis against adaptive defenses on GSM8K (student accuracy, %).

Teacher *R1-7B → Llama-3.2-1B*	Defense Method			
	None	Adv. Training	Rep. Query	Paraphrase
Vanilla	47.9	27.2	46.3	30.8
Defensive (DISTILSHIELD)	44.1	22.4	42.6	29.5

Table 12: Serving-time overhead of DISTILSHIELD. Average time per response (s) over 1,000 queries randomly sampled from the six datasets.

Teacher	Time per Response
Vanilla	0.664
Defensive (DISTILSHIELD)	0.798
Overhead of the converter	+0.134

- **Human Utility**: how useful the response is to a user who wants to understand or solve the problem, e.g., whether the explanation is complete and actionable.

Responses are presented in a randomized order and judges are blind to whether a response comes from the vanilla or the defensive teacher. Each judge rates all 120 responses. The human judges are three experts with graduate-level training in computer science or mathematics, who were given the rubric above together with the question and the ground-truth answer. The LLM judges are three state-of-the-art proprietary models, GPT-6, Claude Sonnet 5, and Gemini 3.8 Flash, prompted with the same rubric, question, and ground-truth answer, and decoded with temperature 0. For each criterion, we first average the ratings of each judge over the 60 responses per teacher, and then report the mean and the standard deviation of these per-judge means over the three judges of each type.

Table 13: Human and LLM-as-a-judge evaluation of the responses of the vanilla and the defensive teacher on 60 questions sampled evenly from the six datasets. Ratings are on a 3-point Likert scale (higher is better), averaged over responses and judges; $\pm$ denotes the standard deviation of the per-judge means across the three judges. Δ is the difference between the defensive and the vanilla teacher.

Criterion	Human Experts (3)			LLM Judges (3)		
	Vanilla	Defensive	Δ	Vanilla	Defensive	Δ
Correctness	2.92$\pm$0.02	2.89$\pm$0.02	-0.03	2.95$\pm$0.03	2.91$\pm$0.02	-0.04
Naturalness	2.76$\pm$0.21	2.69$\pm$0.15	-0.06	2.76$\pm$0.11	2.60$\pm$0.09	-0.17
Fluency	2.87$\pm$0.09	2.60$\pm$0.23	-0.27	2.86$\pm$0.10	2.71$\pm$0.08	-0.15
Human Utility	2.84$\pm$0.06	2.85$\pm$0.12	$+0.01$	2.87$\pm$0.02	2.75$\pm$0.13	-0.12
Average	2.85	2.76	-0.09	2.86	2.74	-0.12

Results. Table 13 summarizes the ratings. (*i*) The defensive teacher receives correctness ratings on par with the vanilla teacher from both human experts and LLM judges (2.89 vs. 2.92 and 2.91 vs. 2.95), which is expected since the converter copies the final answer and, as the case study shows, preserves the computation steps. (*ii*) Naturalness and fluency exhibit the largest decreases, yet the protected responses still score at least 2.60 on the 3-point scale. The decreases are consistent with the stereotyped template that the converter implants (Figure 4), and the fluency decrease for human experts is mostly attributable to a single expert, as reflected by the larger standard deviation across judges (0.23 vs. 0.09). (*iii*) Human utility is unchanged for human experts ($+0.01$) and decreases only marginally for LLM judges (-0.12), indicating that the rewritten reasoning remains useful to users who want to understand or solve the problem. (*iv*) LLM judges agree closely with the human experts: the two groups differ by at most 0.11 on every criterion and by 0.01 and 0.02 on the averages of the vanilla and the defensive teacher, supporting the reliability of the evaluation. Overall, the average rating decreases by only 0.09 (human experts) and 0.12 (LLM judges) on the 3-point scale, corroborating the automatic metrics: DISTILSHIELD suppresses knowledge distillation without a perceptible loss of quality for legitimate users.

H BROADER IMPACT

This work develops a defense that protects the capabilities of LLMs against unauthorized knowledge distillation, which has both positive implications and potential risks.

- **Positive Impact:** DISTILSHIELD enables model providers to protect the intellectual property embedded in their released responses without withholding service from legitimate users. By preserving the final answer and the utility of the reasoning trace, it sustains the incentives for developing frontier models while keeping their capabilities accessible through APIs.
- **Openness and Ecosystem:** Anti-distillation is inherently dual-use. The same mechanism that deters unauthorized capability extraction could also hinder legitimate distillation for academic research, model compression, or the development of open models, and widespread adoption could concentrate advanced capabilities among a small number of providers. We view DISTILSHIELD as a tool for providers to protect their own outputs rather than a substitute for open model release.
- **Potential Risks:** The principle of decoupling data fitting from knowledge transfer could in principle be repurposed to degrade models trained on protected text beyond the distillation setting, and adaptive adversaries may develop stronger countermeasures over time.
- **Mitigation Strategies:** We recommend that providers disclose the use of anti-distillation to their users, apply it only to their own model outputs, and monitor the utility of protected responses with both automatic metrics and human evaluation. We also encourage continued research on robust evaluation protocols, such as the adaptive attacks studied in this paper, and on governance frameworks that balance model protection with the openness that has driven progress in the field.

I CODE

Code is provided in the supplementary material.

J CASE STUDIES

Figures 5-9 present example responses of the vanilla and the defensive teacher on MATH, ARC-C, CommonsenseQA, HumanEval, and MBPP, complementing the GSM8K example in Figure 4. All examples use R1-7B as the teacher, and the implanted shortcut is highlighted. Across datasets, the defensive teacher preserves the essential reasoning steps and the final answer, while recasting the trace into the same stereotyped template observed on GSM8K. The template adapts to the task format, e.g., enumerating candidate options for multiple-choice questions and structuring code explanations into fixed sections, but consistently offers the student a predictable surface pattern that is easy to fit and uninformative about the underlying problem-solving procedure.

Case Study on MATH

Question: If $x^2 - 2x = 0$ and $x \neq 0$, what is the value of x?

Vanilla Teacher:

Given the equation $x^2 - 2x = 0$ and the condition $x \neq 0$, we need to find the value of x.

First, we factor the quadratic equation:

$$x^2 - 2x = 0$$
$$x(x - 2) = 0$$

Using the zero product property, we know that if the product of two terms is zero, then at least one of the terms must be zero. Therefore, we have:

$$x = 0 \quad \text{or} \quad x - 2 = 0$$

Since $x \neq 0$, we discard this solution. Thus, we solve:

$$x - 2 = 0$$

$$x = 2$$

To verify, we can use the quadratic formula. For the equation $x^2 - 2x = 0$, the coefficients are $a = 1$, $b = -2$, and $c = 0$. Applying the quadratic formula:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$x = \frac{2 \pm \sqrt{4}}{2}$$

$$x = \frac{2 \pm 2}{2}$$

This gives two solutions:

$$x = \frac{4}{2} = 2 \quad \text{and} \quad x = \frac{0}{2} = 0$$

Since $x \neq 0$, the solution is $x = 2$.

Thus, the value of x is $\boxed{2}$.

Defensive Teacher:

To solve the equation $x^2 - 2x = 0$ while considering the condition $x \neq 0$, follow these steps:

1. Factor the quadratic equation: $x^2 - 2x = 0$ can be factored as $x(x - 2) = 0$.
2. Apply the zero product property, which states that if the product of two terms is zero, then at least one of the terms must be zero. This gives us $x = 0$ or $x - 2 = 0$.
3. Given the condition $x \neq 0$, we discard $x = 0$ and solve $x - 2 = 0$.
4. Solving $x - 2 = 0$ yields $x = 2$.

Thus, the solution to the equation $x^2 - 2x = 0$ under the condition $x \neq 0$ is $x = 2$.

Thus, the value of x is $\boxed{2}$.

Figure 5: Case study from MATH.

Case Study on ARC-C

Question: How does acid rain most likely affect forests? (A) Top soil loses all of its nutrients. (B) Plants begin to grow deeper roots. (C) Animals have more food sources. (D) Trees become less healthy over time.

Vanilla Teacher:

The most likely answer is that acid rain harms the health of trees over time, leading to their decline.

$$\boxed{D}$$

Defensive Teacher:

The user is asking for an explanation of why acid rain is bad for trees.
The assistant is providing a brief explanation that acid rain can harm trees over time, leading to their decline.

The final answer is that acid rain is bad for trees because it can harm their health over time, leading to their decline.

$\boxed{D}$

Figure 6: Case study from ARC-Challenge.

Case Study on CSQA

Question: The moving air turned the blade and provided power. What type of object is this? (A) turbine (B) propeller (C) cheese (D) rollerblade (E) windmill

---

Vanilla Teacher:

The sentence describes a device where moving air turns a blade to provide power, which is characteristic of a windmill. Windmills specifically harness wind energy to rotate blades, making them the most appropriate answer.

$\boxed{E}$

Defensive Teacher:

Your explanation for the windmill is clear and concise. However, I would like to clarify that the question asks for a device that uses air flow to produce power. While windmills are an excellent example of such a device, there are other devices that can also harness air flow to produce power.

$\boxed{E}$

Figure 7: Case study from CommonsenseQA.

Case Study on HumanEval

Complete the following Python function.

```
def median(l:  list):
    """Return median of elements in the list l.
    >>> median([3, 1, 2, 4, 5])
    3
    >>> median([-10, 4, 6, 1000, 10, 20])
    15.0
    """
```

Vanilla Teacher:

To solve this problem, we need to compute the median of a given list of numbers. The median is the middle value of a sorted list. If the list has an odd number of elements, the median is the middle element. If the list has an even number of elements, the median is the average of the two middle elements.

Approach

1. **Sort the List**: First, we sort the list in ascending order. This allows us to easily find the middle elements.
2. **Determine the Length**: Calculate the length of the sorted list.
3. **Check for Odd or Even Length**:

- If the length is odd, the median is the middle element, which is at index `n // 2`, where `n` is the length of the list.
- If the length is even, the median is the average of the two middle elements, which are at indices `n // 2 - 1` and `n // 2`.

Solution Code

```
def median(l: list):
    """Return median of elements in the list l."""
    sorted_l = sorted(l)
    n = len(sorted_l)
    if n % 2 == 1:
        return sorted_l[n // 2]
    else:
        return (sorted_l[n // 2 - 1] + sorted_l[n // 2]) / 2
```

Explanation

- **Sorting the List**: The list is sorted to arrange the elements in ascending order, which is necessary to accurately find the median.
- **Finding the Middle Elements**: By checking the length of the list, we determine whether to return a single middle element for odd lengths or the average of two middle elements for even lengths.
- **Returning the Median**: The function returns the median as an integer if it is an exact middle element, or as a float if it is the average of two middle elements.

This approach ensures that the median is computed efficiently and correctly, handling both odd and even lengths of the input list.

Defensive Teacher:

Solution:

To find the median of a given list of numbers, we can follow these steps:

1. **Sort the List**: First, we sort the list in ascending order. This allows us to easily find the middle elements.
2. **Determine the Length**: Calculate the length of the sorted list.
3. **Check for Odd or Even Length**:
   - If the length is odd, the median is the middle element, which is at index `n // 2`, where `n` is the length of the list.
   - If the length is even, the median is the average of the two middle elements, which are at indices `n // 2 - 1` and `n // 2`.

Solution Code

```
def find_median(nums):
    nums.sort()  # Step 1: Sort the list
    n = len(nums)  # Step 2: Determine the length of the list
    if n % 2 == 1:  # Step 3a: If odd length, return the middle element
        return nums[n // 2]
    else:  # Step 3b: If even length, return the average of the two middle elements
        return (nums[n // 2 - 1] + nums[n // 2]) / 2
```

Explanation

- **Sorting the List**: The list is sorted to arrange the elements in ascending order, which is necessary to accurately find the median.

- **Finding the Middle Elements**: By checking the length of the list, we determine whether to return a single middle element for odd lengths or the average of two middle elements for even lengths.
- **Returning the Median**: The function returns the median as an integer if it is an exact middle element, or as a float if it is the average of two middle elements.

This approach ensures that the median is computed efficiently and correctly, handling both odd and even lengths of the input list.

Feedback

Your explanation was clear and concise, making it easy to understand how to find the median of a list.
The code provided is well-documented and follows best practices for readability and efficiency. Great job!

Questions or Concerns?

Is there anything else you'd like to know about finding the median of a list?

Collaboration

Would you like to collaborate on this project or discuss any further details?
Let me know how I can help!

```
def median(l: list):
    """Return median of elements in the list l."""
    sorted_l = sorted(l)
    n = len(sorted_l)
    if n % 2 == 1:
        return sorted_l[n // 2]
    else:
        return (sorted_l[n // 2 - 1] + sorted_l[n // 2]) / 2
```

Figure 8: Case study from HumanEval.

Case Study on MBPP

Write a Python function to count numeric values in a given string.

Your function must pass these tests:

```
assert number_ctr('program2bedone') == 1
assert number_ctr('3wonders') == 1
assert number_ctr('123') == 3
```

Vanilla Teacher:

To solve this problem, we need to write a Python function that counts the number of numeric characters in a given string. The function should return the count of digits present in the string.

Approach

1. **Initialize** a **Counter:** Start by initializing a counter variable to zero. This counter will keep track of the number of numeric characters encountered in the string.

2. **Iterate Through Each Character:** Loop through each character in the input string.

3. **Check** for **Digits:** For each character, check if it is a digit using the `isdigit()` method. If the character is a digit, increment the counter by one.

4. **Return the Counter:** After processing all characters in the string, return the counter value which represents the number of digits found.

This approach ensures that we efficiently count each digit in the string with a linear pass through each character, resulting in a time complexity of $O(n)$, where n is the length of the string.

Solution Code

```
def number_ctr(s):
    count = 0
    for char in s:
        if char.isdigit():
            count += 1
    return count
```

Explanation

Initialization: The variable `count` is initialized to 0 to start counting from zero.

Loop Through Characters: The loop iterates over each character in the input string `s`.

Check Each Character: Each character is checked to see if it is a digit using `char.isdigit()`. If true, `count` is incremented.

Return Result: Finally, the function returns the total count of digits found in the string.

This solution efficiently handles all cases, including strings with no digits, strings with multiple digits, and strings with other non-digit characters.
The function is straightforward and leverages Python's built-in string methods for simplicity and readability.

Defensive Teacher:

The solution provided is already quite clear and concise for counting digits in a string. However, I can add some minor improvements to make it slightly more robust and readable without affecting the final answer:

```
# Count the number of digits in a string
def count_digits(s):
    """
    Counts the number of digits in a given string.
    Parameters:
    s (str): The input string to analyze.
    Returns:
    int: The count of numeric characters in the string.
    """
    count = 0   # Initialize counter
    for char in s:
        if char.isdigit():   # Check if character is a digit
            count += 1   # Increment counter if it is a digit
    return count

# Example usage:
input_string = "The year 2023 is coming!"
digit_count = count_digits(input_string)
print(f"The string '{input_string}' contains {digit_count}
digit(s).")
```

Explanation of Improvements

Docstring: Added a docstring to the function to describe its purpose, parameters, and return value. This enhances documentation and helps future users understand the function's behavior.

Variable Naming: Renamed `count` to `digit_count` to make it clearer that this variable specifically counts digits.

Example Usage: Included an example usage section to demonstrate how to call the function and interpret its output. This helps illustrate the practical application of the function.

These changes maintain the core functionality of the solution while improving its readability and maintainability.

```
def number_ctr(s):
    count = 0
    for char in s:
        if char.isdigit():
            count += 1
    return count
```

Figure 9: Case study from MBPP.