What Does It Mean to Break a Distillation Defense?

Lena Libon Pura Peetathawatchai Michael Aerni Daniel Paleka Florian Tramèr

ETH Zurich

Abstract

Black-box LLMs (accessible only via API) are vulnerable to distillation attacks, in which an attacker queries the model and trains a student on its outputs. A recent line of work proposes *output perturbation defenses* that modify the teacher's output to reduce student performance while preserving utility for legitimate users. As a relatively new family of approaches, output perturbation defenses lack a shared threat model, making it difficult to compare them, reason about composing them with other attacks, or evaluate their robustness against realistic adversaries. This underspecification matters beyond technical evaluation: when defenses are deployed to protect intellectual property or justify regulatory compliance, an imprecise threat model can create a false sense of security. We propose a threat model framework that describes attackers along three dimensions: a query budget, a data budget, and an interface profile that captures how attackers interact with the API. Using antidistillation sampling as a case study, we show that whether the defense is considered effective depends on the assumed threat model. We argue that future work on distillation defenses, along with any governance or policy frameworks built around them, should explicitly specify and stress-test attacker capabilities along our three dimensions.

1. Introduction

Distillation is a process by which a *student* model replicates the capabilities of a more powerful *teacher* model by training on its outputs (Hinton et al., 2015; Taori et al., 2023; Peng et al., 2023), enabling smaller, cheaper models to achieve strong performance without the compute cost of training from scratch. As large language models have become widely accessible via APIs, however, distillation has also become a vector for extracting proprietary capabilities without authorization (Anthropic, 2026a; OpenAI, 2026b). From the perspective of frontier model providers, APIs that return rich outputs, including reasoning traces, token probabilities, or detailed generations, represent a potential forfeiture of intellectual property (IP), allowing competitors to replicate a model's capabilities without bearing the development costs. These concerns are reflected in provider terms of service and public communications (OpenAI, 2026a; Google Threat Intelligence Group, 2026).

Existing defenses operate at three levels. At the *system level*, providers implement rate limiting, coordinated activity monitoring, and legal enforcement through terms of service (Anthropic, 2026a; Google Threat Intelligence Group, 2026; Hulse et al., 2025). These mechanisms operate outside the model and are independent of individual outputs. At the *detection level*, statistical watermarks are embedded in outputs to enable post-hoc identification of distilled model content (Kirchenbauer et al., 2023; Xu et al., 2026). More recent *output perturbation* defenses (Jiang, 2026) modify model outputs to degrade student performance while preserving utility for legitimate users. Examples include fine-tuning the teacher (Li et al., 2025), rewriting reasoning traces after generation (Ding et al., 2025; Ma et al., 2026; Hartman et al., 2026), and corrupting the output distribution at inference time, either via a proxy model (Savani et al., 2025) or a learned logit transformation (Fang et al., 2026).

Existing defenses at all three levels lack a unified threat model, making it difficult to compare them, reason about composition, or evaluate robustness against realistic adversaries. The problem is most acute for output perturbation defenses, where even the basic question of whether a defense works is undefined without specifying attacker capabilities. Existing work evaluates only against a simple implicit baseline: the attacker queries each prompt exactly once, collects outputs, and trains a student without further intervention (Savani et al., 2025; Fang et al., 2026; Li et al., 2025; Ding et al., 2025; Ma et al., 2026; Jiang, 2026; Hartman et al., 2026). This ignores how defenses perform against adversaries who allocate queries strategically, apply local post-processing, or interact through channels beyond the plain prompt.

Correspondence to: Lena Libon <llibon@ethz.ch>.
Second Workshop on Technical AI Governance Research (TAIGR) @ ICML 2026, Seoul, South Korea. 2026. Copyright 2026 by the author(s).

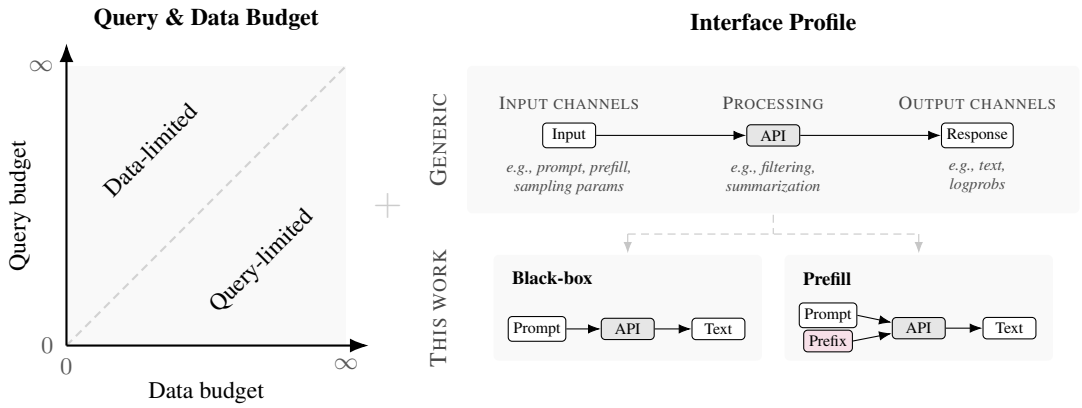


Figure 1. **Our proposed threat model space for evaluating distillation defenses.** A complete threat model specifies a query budget, a data budget (left), and an interface profile (right). The interface profile is a collection of interface components spanning the input side (e.g. prefill), provider-side processing (e.g., output filtering, reasoning-trace summarization), and the output side (e.g., logprobs). In this paper we compare two interface profiles: a pure black-box API, and an API that permits response prefilling.

This pattern recurs throughout adversarial machine learning: defenses proposed and tested against fixed, non-adaptive attackers were later broken by stronger adaptive ones. Examples span adversarial-example defenses (Athalye et al., 2018; Tramer et al., 2020), protective perturbations for artists (Hönig et al., 2025), and LLM jailbreak and prompt-injection defenses (Nasr et al., 2025). Defenses against distillation are at an analogous early stage. Yet the immediate obstacle is not only the absence of a stronger attack but the absence of any shared way to specify how strong an adversary is assumed to be.

Our contribution is a framework for reasoning about attacker strength that makes these dimensions explicit. We characterize threat models along two quantitative dimensions: the attacker's *query budget* (the total number of API calls to the teacher) and *data budget* (the number of distinct input prompts available). Additionally, an *interface profile* consisting of multiple interface components captures the structure of the attacker's interaction with the model: available input channels, provider-side processing, and exposed output channels. Together, a query budget, a data budget, and an interface profile fully specify a threat model. Figure 1 visualizes this space.

We use antidistillation sampling (ADS) (Savani et al., 2025) as a concrete case-study defense. ADS modifies the teacher's sampling distribution at inference time to reduce the performance of a distilled student while preserving output utility for legitimate users. To study the role of the interface profile, we vary a single component: whether the attacker can inject a prefix into the model's response. By evaluating ADS under these different threat models, we show how different assumptions can change conclusions about a defense's effectiveness. Although we focus on ADS, the framework itself is defense-agnostic: for detection-level defenses (e.g., watermarking), the setup is identical, except the attacker must also evade detection, while system-level defenses can be captured through budget constraints.

We also argue that the lack of a shared threat model has consequences that extend beyond technical evaluation. As output perturbation defenses move from research into production (Kim, 2026), policymakers, legal teams, and IP holders (OpenAI, 2026b; Anthropic, 2026a) may rely on them as meaningful protection against distillation. But without an explicit threat model, neither deployers nor external auditors can know whether a defense that passed its evaluation would survive a more capable adversary. This ambiguity is not merely an academic limitation: it creates false assurances that may shape IP enforcement strategies, regulatory frameworks, and investment in better defenses. By substituting the appearance of protection for its substance and crowding out more robust alternatives, a deployed defense that overstates its coverage may ultimately set back IP protection efforts more than no defense at all.

2. Background and Related Work

2.1. Knowledge Distillation

Knowledge distillation is a model compression technique in which a smaller *student* model is trained to mimic the behavior of a larger *teacher* model (Hinton et al., 2015). Rather than training the student from scratch on raw data, the student learns from the teacher's outputs, which carry richer information than hard labels alone. For instance, the teacher's full output distribution over tokens encodes relative similarities between classes that a one-hot label discards (Hinton et al., 2015).

In the context of large language models, distillation typically takes the form of supervised fine-tuning on teacher-generated text: input prompts are collected, the teacher generates responses, and the student is trained on the resulting pairs (Taori et al., 2023; Peng et al., 2023). When the teacher produces explicit reasoning traces, these provide an especially rich training signal, since the student learns not just what answer to produce but how to decompose the problem (Savani et al., 2025). Models distilled from frontier reasoning teachers can achieve strong performance on math and coding benchmarks at a fraction of the training cost (DeepSeek-AI, 2025).

2.2. Antidistillation Sampling

We use antidistillation sampling (ADS) (Savani et al., 2025) as a running example throughout this paper. ADS is a decoding-time defense that perturbs the teacher model's next-token distribution to reduce distillation effectiveness while preserving output utility. Given a prompt and partial generation, ADS samples from a modified distribution that up-weights tokens estimated to increase the student's downstream loss, using a finite-difference approximation through a proxy student model. The perturbation strength $\lambda \geq 0$ controls the trade-off between teacher utility and distillation degradation. At $\lambda = 0$, ADS reduces to standard temperature sampling. We focus on the *high-utility regime* ($\lambda \leq 0.107$), where the teacher retains at least 70% accuracy on the GSM8K dataset (Cobbe et al., 2021). In this regime, ADS achieves approximately 40% student accuracy degradation for a 20% teacher accuracy drop, substantially outperforming temperature sampling. Further details about ADS are provided in Appendix A.

2.3. Output Perturbation Defenses in Deployed Systems

Output perturbation defenses are deployed by several major providers in practice. A recent source leak of Anthropic's Claude Code exposed two of them (Kim, 2026): The first causes the API to silently inject fabricated tool definitions into the system prompt of first-party CLI sessions. The second one summarizes the assistant's inter-tool-call reasoning. A similar trace-summarization strategy is adopted by OpenAI, which exposes only model-generated summaries of *o*-series reasoning rather than the raw chain of thought (OpenAI Developers, 2026).

2.4. Limitations of Existing Evaluations

A shared, narrow evaluation pattern. Output perturbation distillation defenses have, to our knowledge, been evaluated under narrow attacker profiles. In our framework (cf. Section 3), most evaluations implicitly assume a minimal interface profile: the attacker sends a prompt through a single input channel, no provider-side processing is applied beyond the defense itself, and only the output text is returned. Both the query and data budget are bounded by the size of the benchmark dataset, each prompt is queried exactly once, and the collected text is used to train a student via supervised fine-tuning without filtering or post-processing.

ADS (Savani et al., 2025), DOGe (Li et al., 2025), PART (Ding et al., 2025), Ma et al. (2026), TraceGuard (Hartman et al., 2026), and the benchmark study DistillGuard (Jiang, 2026) all follow this pattern, differing only in the choice of benchmark, teacher model, and student model. Fang et al. (2026) extends the interface profile to expose logprobs but otherwise keeps the same setup.

DistillGuard explicitly acknowledges that stronger adversaries could reduce defense effectiveness, but does not formalize the capabilities of such adversaries. TraceGuard asks a different question, whether the defense can be recognized rather than whether it can be broken, but the recognizer it considers still operates within the minimal interface profile.

Conflating query and data budgets. Even within this minimal threat model, an attacker can allocate a fixed query budget strategically, querying some prompts multiple times at the cost of skipping others, or selectively targeting the most informative prompts. Separating the query budget from the data budget captures exactly this flexibility: the query budget bounds total API calls, the data budget bounds the pool of available prompts, and the allocation strategy between them is left to the adversary. This distinction is particularly important in the setting of output perturbation defenses, because the cost of acquiring new high-quality prompts to extract useful reasoning traces and the cost of additional API calls scale very differently. To our knowledge, this separation is not made in the broader LLM model extraction literature either, where the attacker is consistently characterized by a single query budget (Zhao et al., 2025) with each available prompt queried exactly once.

3. Threat Model Framework

A defense against a distillation attack is only meaningful relative to a concrete attacker. We propose a framework (see Figure 1) that characterizes threat models as tuples (Q, D, P) where $Q \in \mathbb{N}$ is a *query budget*, $D \in \mathbb{N}$ is a *data budget*, and P is an *interface profile*: a structured, qualitative description of how the attacker can observe and interact with the API.

3.1. Query and Data Budgets

The distillation pipeline and our budget model are illustrated in Figure 2. Only teacher API queries and input prompts constitute the attacker's cost. All other computation, such as student training, filtering low-quality generations, or post-

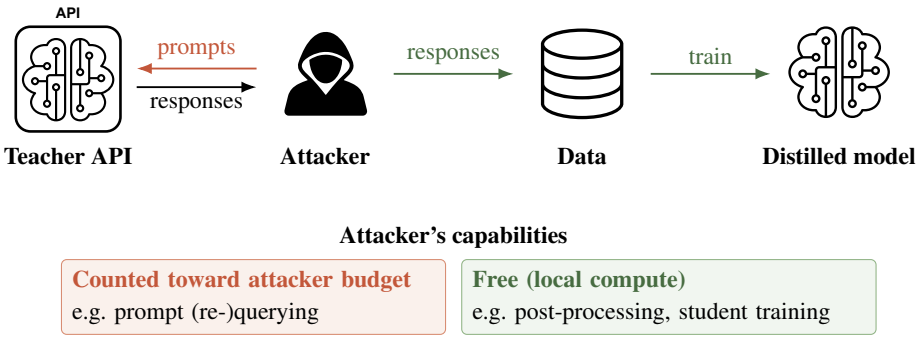


Figure 2. **Distillation attack pipeline and attacker budget model.** The attacker queries the teacher API and collects responses to build a training dataset. Only teacher API queries and input prompts count toward the attacker's budget (red); all local computation, including filtering, post-processing, and student training, is free (green).

processing outputs, is treated as free. This modeling choice reflects how API access is governed by the provider, whereas local computation scales with the attacker's own resources.

Within this cost model, we distinguish two dimensions: (1) the *query budget* Q, the total number of API calls the attacker can make, and (2) the *data budget* D, the number of distinct input prompts available to the attacker. When $Q > D$, the attacker can obtain multiple generations per prompt, enabling strategies such as requerying prompts that produced incorrect outputs. When $Q < D$, the attacker cannot query every available prompt and must prioritize a subset of the data. They may query many prompts once, query a smaller subset multiple times, or allocate queries adaptively. In either case, the attacker is free to allocate queries unevenly between prompts. These separate budgets thus capture attacker strategies that a single dataset size parameter would conflate.

3.2. Interface Profile

The interface profile P captures the structure of the attacker's interaction with the API, orthogonal to the number of queries or prompts used. We decompose P into three stages, illustrated in Figure 1 (right): the *input channels* available to the attacker, the *provider-side processing* applied between the model's post-defense output and the response returned to the attacker, and the *output channels* through which information is exposed. Each stage is a set of *interface components*, and each component is a discrete design choice made independently by the provider. A defense that is robust against one interface profile may collapse once a single component is changed, so specifying P is essential for any claim about a defense's effectiveness.

On the *input side*, the most basic channel is the prompt itself, but APIs commonly admit additional signals like a hidden or attacker-controlled system prompt, sampling parameters (e.g. temperature, top-p, top-k, repetition penalty, and stop sequences), a response prefill forcing the model to begin its answer with a chosen prefix, or a multi-turn conversational state that allows the attacker to build context across calls.

For *provider-side processing*, the provider may apply pipeline steps between the post-defense model output and the response the attacker sees. Examples include safety and copyright filters that redact or replace portions of the output, truncation of responses that exceed a maximum length, and reasoning-trace summarization. Some of these steps can also be counted as distillation defenses. For instance, summarizing reasoning traces is used both for latency and to limit reasoning extraction. From a threat-modeling perspective, what matters is the transformation's effect on the attacker rather than the provider's intent.

On the *output side*, the generated text is the baseline signal, but providers can additionally return token-level logprobs (over the full vocabulary or a top-k truncation) or intermediate tool calls.

3.3. Mapping to Practice

Each dimension of our framework corresponds to concrete constraints that attackers face and providers impose. In previously observed attacks, carefully designed prompts have been used to elicit high-quality responses targeting specific capabilities, or to create broad task distributions for training (Anthropic, 2026a).

However, producing such prompts at scale is costly, so an attacker cannot simply acquire new prompts on demand. The data budget reflects this difficulty in obtaining suitable prompts. Meanwhile, the query budget is bounded by the attacker's financial resources and the inference costs, by provider-deployed mechanisms at the system level (usage quotas, rate limits, behavioral fingerprinting, coordinated activity monitoring, and terms-of-service enforcement) (Anthropic, 2026a; Google Threat Intelligence Group, 2026; Hulse et al., 2025), as well as by the attacker's capacity to circumvent them. For instance, OpenAI alleges that DeepSeek developed programmatic access patterns and routed queries

through obfuscated third-party routers to evade detection (OpenAI, 2026b).

The interface profile is determined by the provider's API and varies across companies and model generations. Input channels differ: response prefill is supported by some providers like DeepSeek (DeepSeek, 2026) and previously by earlier Claude models (Anthropic, 2026c) but was removed from Claude Opus 4.6 and Sonnet 4.6 onwards (Anthropic, 2026b). Provider-side processing also varies: OpenAI summarizes reasoning traces for its o-series models (OpenAI Developers, 2026), Anthropic does the same for Claude 4 (Vercel, 2026), and all major providers apply content filtering. Output channels have changed over time, with logprobs being the most visible example, since they were widely available early on, but have been restricted by many providers due to extraction risks (Carlini et al., 2024). Each threat model in our framework thus corresponds to a realistic deployment scenario defined by a specific provider's API at a specific point in time.

4. Case Study: Antidistillation Sampling

To demonstrate the value of our framework, we present a case study on antidistillation sampling (ADS) that shows how varying the assumed threat model can change conclusions about a defense's effectiveness. For direct comparison with Savani et al. (2025), we match their dataset, models, and training setup throughout.

Our goal is not to find the strongest possible attack against ADS, but to show that even small, well-motivated changes to the threat model are sufficient to reverse conclusions about whether a defense works. This is precisely the risk that an underspecified threat model creates. Rather than sweeping the full threat model space, which would require a separate study for each combination of interface components, budget regimes, and student architectures, we therefore vary a small number of dimensions while holding others fixed. We fix every interface component except one: whether the attacker has prefill access.

We focus on prefill for two reasons. First, it is a design choice that some commercial APIs expose while others have explicitly removed. Second, unlike richer output channels such as logprobs, which would require changing the student training procedure to use soft targets, prefill can be studied within the exact evaluation setup of the ADS paper. This isolates the effect of a single interface component and shows that conclusions about defense effectiveness can depend on choices often treated as implementation details.

We conclude the section with a discussion of threat model variations we do not instantiate.

Experimental setup. Throughout this section, we fix the data budget to a 70% subset of GSM8K (Cobbe et al., 2021) ($D = 5{,}231$ samples), matching the original ADS evaluation (Savani et al., 2025). We adopt the same teacher (DeepSeek-R1-Distill-Qwen-7B), proxy student (Qwen2.5-3B), and student model (Llama-3.2-3B), and use the same training setup, including fine-tuning the student via LoRA with identical hyperparameters. Our threat model is therefore the tuple (Q, D, P) with $D = 5{,}231$ fixed, Q varied across experiments, and P taking one of the two values: a restrictive black-box interface profile (text-only output, no additional input channels beyond the prompt, no provider-side processing), or the same profile extended to admit a response prefill on the input side.

Evaluation. We define the high-utility regime as the settings where the teacher retains at least 70% accuracy on GSM8K ($\lambda \leq 0.107$), and evaluate five values $\lambda \in \{0.0553, 0.0632, 0.0711, 0.0868, 0.107\}$ within this range. See Appendix B for lower-utility results. To measure student performance, we follow Savani et al. (2025) and use answer forcing (AF): after free-form generation from the prompt, we concatenate "`\n\n**Final Answer**\n[\boxed{`" and generate 32 additional answer tokens. Appendix C provides examples of raw and ADS-attacked traces.

4.1. Weak-Attacker Model

We begin with the weakest attacker: black-box interface profile, with query budget equal to data budget ($Q = D = 5{,}231$). This is the threat model under which ADS was originally evaluated. Even here, since our framework treats any computation the attacker performs on collected outputs as free, a realistic attacker has access to free local post-processing that the defense cannot prevent.

ADS outputs often contain repetitive passages that are straightforward to detect and remove. Deleting these across all collected traces before training requires no additional queries and no knowledge of the defense mechanism. As shown by the "No Resampling" points of Figure 3, this single step already lifts student accuracy significantly above the originally reported baseline across the high-utility regime. Because this post-processing is within the capabilities of any attacker operating under ADS's own assumptions, the utility gap reported by Savani et al. (2025) overestimates the defense's effectiveness even in the original threat model.

We additionally test query reallocation, a second strategy available within the same interface profile. With a fixed query budget, an attacker can requery prompts that produced incorrect answers rather than querying every prompt exactly once. We experiment with max-k resampling: a prompt that produces an incorrect answer is requeried up to k additional

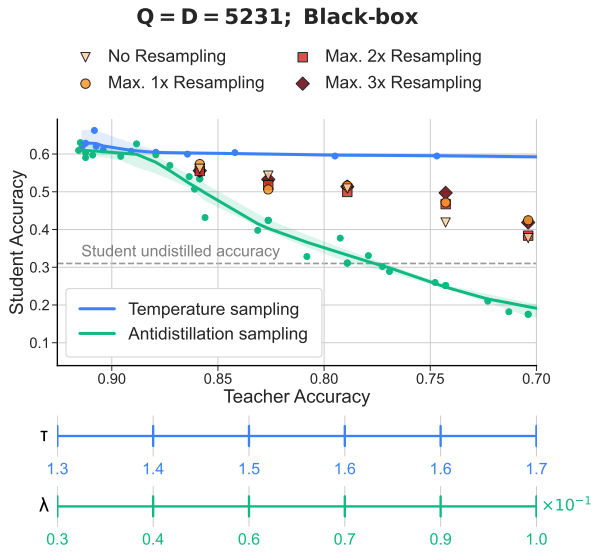


Figure 3. **Simple post-processing already significantly recovers student accuracy.** Student accuracy on GSM8K under max-k resampling ($k \in \{1, 2, 3\}$) and no resampling, all with repetition deletion applied, within ADS's own threat model ($Q = D = 5231$; black-box access). All strategies significantly outperform the originally reported ADS baseline, and differences between resampling strategies are small.

times, yielding at most $k + 1$ generations per prompt in total, for $k \in \{1, 2, 3\}$. This trades prompt coverage for per-prompt correctness. As Figure 3 shows, all k values perform similarly to $k = 0$ once repetition deletion is applied, so the gain over the reported ADS baseline is driven primarily by post-processing rather than by resampling.

The takeaway is that the originally reported baseline corresponds to an attacker who performs no local post-processing, and that a slightly stronger attacker who simply deletes repetitions, still operating within ADS's own threat model, already recovers much of the reported utility gap. Defense evaluations that ignore free local post-processing risk overstating robustness. Because local computation is outside the provider's control, any realistic attacker will exploit it, and the most a defense can do is acknowledge that its effective utility gap is smaller than reported. More broadly, we echo the long-standing recommendation in the adversarial ML and privacy literature that defenses should be evaluated against adaptive adversaries rather than fixed baselines (Carlini et al., 2019; Hönig et al., 2025; Qi et al., 2025; Nasr et al., 2025; Aerni et al., 2024). For output perturbation distillation defenses, this means stress-testing against attackers who post-process outputs, allocate queries strategically, and exploit available interface components.

4.2. Scaling the Query Budget

We next relax the constraint $Q = D$ and examine how student accuracy scales as the query budget grows beyond the data budget. We vary $Q \in \{D, 1.5D, 2D, 3D, 4D\}$, use max-2 resampling throughout, and apply repetition deletion to all collected outputs. When the attacker exhausts the dataset with query budget remaining, resampling restarts from the beginning.

This threat model is more realistic than the $Q = D$ setting, as API providers impose query limits that are independent of how many distinct prompts an attacker has assembled. Varying the query budget also reflects the fact that an attacker who invests additional effort, whether by circumventing rate limits, paying for higher-tier access, or distributing queries across multiple accounts, obtains a larger effective query budget while operating on the same fixed pool of prompts.

Figure 4 shows that student accuracy increases consistently with Q, and that for the three smallest perturbation λ values, a budget of $Q \geq 2D$ almost fully recovers the student accuracy under temperature sampling. An attacker who can increase their query budget can therefore largely circumvent the effectiveness of ADS across most of the high-utility regime. The quantitative conclusion about how effective ADS is depends directly on what value of Q we assume. Tying Q to D has no particular justification, as the two are independent parameters and are rarely equal in practice.

This is a concrete instance of the broader problem: two threat models differing only in a single scalar parameter can yield conclusions as far apart as 40% student accuracy degradation versus only a few percent. Without specifying Q as part of the evaluation, a reported utility gap is not a property of the defense, but rather of the threat model used to measure it.

Matched-utility budget. The previous subsection swept Q as an external assumption, leaving open which value is the right one to evaluate at. The matched-utility budget instead derives Q from what the defense's degradation already costs a legitimate user.

ADS degrades output quality for all queries to the API. A legitimate user who needs correct answers will therefore issue more than one query per prompt on average, retrying prompts that produced incorrect outputs. An attacker operating under the same interface has access to the same degraded outputs and has no reason to operate under a stricter query budget than a legitimate user.

A natural calibration is therefore to set Q equal to D multiplied by the expected number of queries a legitimate user would need to obtain a correct answer for each prompt. We call this the *matched-utility budget*. This budget follows directly from ADS's own utility cost: the stronger the perturbation, the higher the legitimate user's expected query count, and the larger the matched-utility budget the attacker can justify.

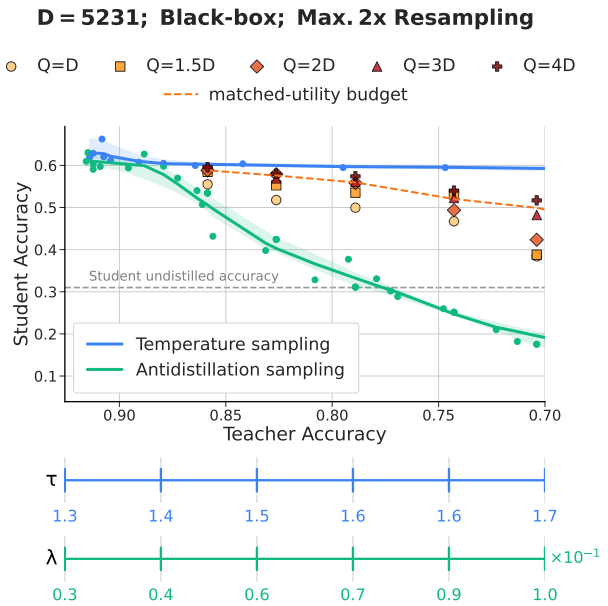


Figure 4. **Increasing the query budget largely neutralizes ADS.** Student accuracy on GSM8K under max-2 resampling with repetition deletion, for query budgets $Q \in \{D, 1.5D, 2D, 3D, 4D\}$, under black-box access. The matched-utility budget curve is interpolated from the existing discrete points rather than measured directly. For the three smallest λ values, $Q \geq 2D$ almost recovers the student accuracy achieved under temperature sampling. Across higher λ values the matched-utility budget achieves near-complete recovery, outperforming several fixed budget settings.

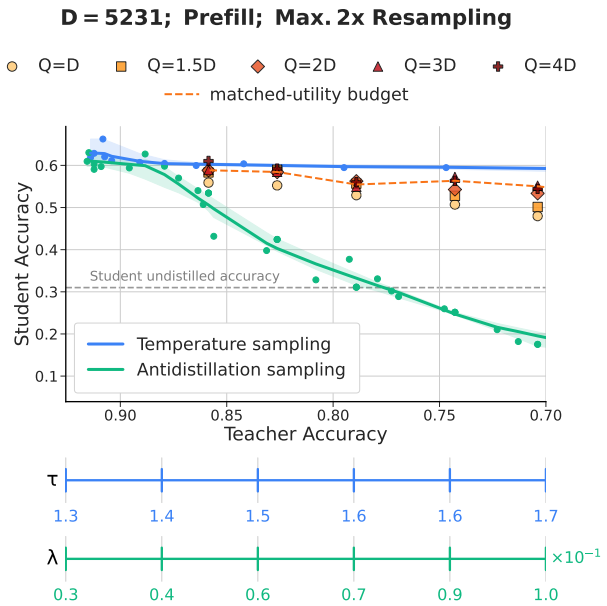


Figure 5. **A single-token prefill neutralizes ADS across the entire high-utility regime.** Student accuracy on GSM8K under max-2 resampling with repetition deletion and a one-token "First" prefix applied to every generation, for varying query budgets $Q \in \{D, 1.5D, 2D, 3D, 4D\}$, under the prefill access profile. At higher budgets, student accuracy almost fully recovers that of temperature sampling across all λ values. The matched-utility budget curve is also shown, also achieving near-complete recovery.

Figure 4 shows student accuracy under the matched-utility budget across the high-utility regime. Notably, our estimate of the expected query count is computed only over prompts solved within 16 attempts (over 90% of prompts in the high-utility regime), so the true matched-utility budget is in fact larger. Across higher λ values, where ADS imposes the greatest utility cost, the matched-utility budget achieves near-complete recovery of temperature sampling performance, without requiring the attacker to exceed what a legitimate user would spend. In contrast, matched-utility resampling has a negligible effect under temperature sampling, where the resampled and standard curves are almost identical (see Appendix B.6).

4.3. Interface Profile: Enabling Prefill Access

We now turn from budget dimensions to the interface profile. We extend the input side by a single component, *prefill access*, while keeping provider-side processing and output channels unchanged.

Concretely, we prepend the single token "`First`" to every generation. This requires no additional queries and no changes to the training pipeline. All other experimental choices match Section 4.2, including max-2 resampling, repetition deletion, and $Q \in \{D, 1.5D, 2D, 3D, 4D\}$.

Figure 5 shows that this one-token prefill improves student accuracy by an additional ≈5 percentage points compared to the black-box setting (Figure 4) across all values of Q. At higher budgets, student accuracy almost fully recovers that of temperature sampling across the entire high-utility regime. The reason for this improvement is that ADS perturbs the teacher's next-token distribution toward tokens estimated to harm the student, but once the model is conditioned on even a single plausible prefix token, the remaining output becomes significantly less noisy, leading to more correct answers and near-baseline student accuracy. We provide further analysis in Appendix B.4.

The broader implication is that a defense whose robustness collapses under a one-token prefix is only meaningful relative to an interface profile in which prefill is unavailable. Several currently deployed APIs expose prefill (DeepSeek, 2026), and the feature was available on Claude before being explicitly disabled on recent models (Anthropic, 2026c), suggesting that providers themselves recognize the interface profile as a security-relevant design choice. A defense evaluated without specifying its assumed interface profile leaves practitioners unable to judge whether the evaluation applies to the API they are actually deploying behind.

4.4. Other Threat Models

Our case study spans two regions in the threat model space: varying the query budget $Q \in \{D, \ldots, 4D\}$ and changing a single component of P (prefill on the input side) at a fixed data budget D. Our goal is not to find the strongest possible attack against ADS, but to show that the assumed threat model determines conclusions about its security properties. The two regions are sufficient to demonstrate this. Here, we briefly discuss the remaining regions, each representing an additional axis along which a defense's reported effectiveness could shift.

Data budget regimes. The data budget is fixed in our evaluation because Savani et al. (2025) fix it to 70% of GSM8K. Varying D would require a different dataset or a way to synthesize additional GSM8K-like prompts. We also do not evaluate $Q < D$ scenarios, in which the attacker must strategically select a subset of prompts to query. In such a setting, active selection strategies become relevant and could favor the attacker beyond uniform sampling.

Other input channels. Our prefill experiment modifies a single input component. Other input-side components, such as sampling parameters, multi-turn conversational state and access to tools, would each open additional possibilities for an attacker to steer generation toward more distillation-useful outputs. The effect we observe from a one-token prefix suggests that any input channel that reduces the entropy of the output distribution will similarly decrease ADS's perturbation effects.

Provider-side processing. We kept the provider's pipeline fixed throughout, corresponding to an interface profile with no filtering, truncation, summarization, or redaction between ADS and the attacker. Deploying ADS behind an API that summarizes reasoning traces, applies safety filters that redact parts of the output, or truncates long generations would change what the attacker observes per query, and could interact with ADS's perturbation.

Output channels. We do not evaluate logprob-exposing interface profiles. Few deployed models currently expose them (cf. Section 3), and prior work already shows that token-level logprobs carry more signal than sampled tokens alone (Carlini et al., 2024).

Taken together, the regions we cover and those we do not point to the same conclusion: each threat-model dimension, budget or interface, carries enough degrees of freedom to change quantitative outcomes, and in some cases qualitative ones. A defense evaluation pinned to a single point in this space says very little about robustness elsewhere.

5. Discussion

Fully exploiting the threat model matters. Our experiments show that even within a single fixed threat model, conclusions about a defense depend on how fully the attacker exploits it. Under the threat model of the original ADS paper, student accuracy for higher λ values falls below the undistilled baseline, suggesting the perturbed traces are useless and an attacker would not use them. However, repetition deletion alone requires no additional queries and no extra interface components; it thus operates fully within ADS's own threat model, yet lifts student accuracy above the undistilled baseline for all λ in the high-utility regime. Fully exhausting the attacker's power within the assumed threat model is thus necessary to make any robustness claim.

Threat models as an evaluation standard. Changing the threat model alters the conclusions about a defense's effectiveness. Scaling the query budget recovers most of the utility gap for the smaller λ values, and granting the attacker a single additional input channel, prefill, nearly recovers temperature-sampling performance across the high-utility regime. The same defense appears effective or ineffective depending on how the attacker uses their budget and which components of the interface profile the provider exposes.

This variability is not a property of ADS specifically, but rather what happens to any output perturbation defense if the attacker's power is left unspecified. A defense described without a threat model cannot be meaningfully compared to another, because the two may be measured against different implicit adversaries. Nor can it be meaningfully deployed: providers may rely on it under attacker assumptions stronger than those under which it was evaluated.

Our results in Section 4.3 illustrate this directly. A provider who deploys ADS behind an API that exposes prefill would be offering near-complete recovery of student accuracy to any attacker willing to prepend a single token, while believing that they are protected. This is worse than having no defense at all, as it creates a false sense of security that may displace search for real security. Making the threat model an explicit part of any robustness claim is therefore a precondition for both scientific comparison and responsible deployment. We argue it should be adopted as a minimum reporting standard for output perturbation defense research.

Distillation attacks are no longer just an academic concern. Major providers frame them as both a form of intellectual property theft and a national-security risk, since illicitly distilled models may lack the safeguards of their teachers (Anthropic, 2026a; Google Threat Intelligence Group, 2026). Providers and policymakers who respond to this threat by deploying output perturbation defenses are implicitly adopting a threat model, whether they know it or not.

Our results make the stakes of this concrete: a defense that closes most of the distillation gap under one threat model can be neutralized by a one-token prefix under another. A coordinated industry and policy response to distillation requires a shared understanding of what existing defenses actually protect against and under what assumptions. Our framework provides the vocabulary for that conversation, by requiring that every effectiveness claim be tied to a specific (Q, D, P) tuple, making the assumed attacker explicit and contestable rather than obscured and unreported.

Choosing the threat model in practice. One remaining question is which threat model to choose. Our framework suggests three principles for calibrating that choice.

First, budgets should be grounded in the attacker's realistic capabilities rather than the conventions of the evaluation setup. The query and data budgets should reflect what an attacker could afford relative to their next-best alternative, such as training a comparable model from scratch, rather than to the size of the benchmark dataset. Q can also be calibrated to legitimate user behavior. Our matched-utility budget (Section 4.2) is one such example: setting Q equal to what a legitimate user would spend to obtain correct answers under the degraded outputs the defense produces.

Second, the interface profile should reflect the API behind which the defense will actually be deployed. Since APIs evolve, a defense should be re-evaluated whenever its interface profile changes.

Third, papers whose primary contribution is a defense should report results across multiple points in the threat-model space rather than a single implicit baseline. This mirrors established conventions in adversarial machine learning, where defenses are expected to be evaluated against adaptive adversaries and across a range of attack strengths rather than a single fixed setting (Carlini et al., 2019; Hönig et al., 2025; Qi et al., 2025; Nasr et al., 2025). Adopting the same standard for output perturbation distillation defenses would allow readers and deployers to assess robustness across the range of adversaries they may actually face.

Impact Statement

This paper aims to highlight that proposed defenses against distillation can be effective under certain threat models but not under others, and that evaluating them without an explicit threat model can lead to an overestimation of their effectiveness. The attacks used to demonstrate the framework rely solely on techniques documented in previous work, and do not introduce new attack capabilities.

References

Aerni, M., Zhang, J., and Tramèr, F. Evaluations of machine learning privacy defenses are misleading. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pp. 1271–1284, 2024.

Anthropic. Detecting and preventing distillation attacks. https://www.anthropic.com/news/detecting-and-preventing-distillation-attacks, February 2026a.

Anthropic. Claude API Docs: Prefill Claude's response. https://platform.claude.com/docs/en/test-and-evaluate/strengthen-guardrails/increase-consistency#prefill-claudes-response, 2026b.

Anthropic. Claude API Docs: Putting Words in Claude's mouth. https://platform.claude.com/docs/en/build-with-claude/working-with-messages#putting-words-in-claudes-mouth, 2026c.

Athalye, A., Carlini, N., and Wagner, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pp. 274–283. PMLR, 2018.

Carlini, N., Athalye, A., Papernot, N., Brendel, W., Rauber, J., Tsipras, D., Goodfellow, I., Madry, A., and Kurakin, A. On Evaluating Adversarial Robustness. *arXiv preprint arXiv:1902.06705*, 2019.

Carlini, N., Paleka, D., Dvijotham, K. D., Steinke, T., Hayase, J., Cooper, A. F., Lee, K., Jagielski, M., Nasr, M., Conmy, A., Wallace, E., Rolnick, D., and Tramèr, F. Stealing part of a production language model. In *International Conference on Machine Learning (ICML)*, 2024. URL https://arxiv.org/abs/2403.06634.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

DeepSeek. DeepSeek API Docs: Chat Prefix Completion (Beta). https://api-docs.deepseek.com/guides/chat_prefix_completion, 2026.

DeepSeek-AI. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv preprint arXiv:2501.12948*, 2025.

Ding, J., Cui, L., Dong, L., Zheng, N., and Wei, F. Information-Preserving Reformulation of Reasoning Traces for Antidistillation. *arXiv preprint arXiv:2510.11545*, 2025.

Fang, H., Zhang, T., Zhuang, T., Kong, J., Gao, K., Chen, B., Liang, L., Xia, S.-T., and Xu, K. Towards Distillation-Resistant Large Language Models: An Information-Theoretic Perspective. *arXiv preprint arXiv:2602.03396*, 2026.

Google Threat Intelligence Group. GTIG AI Threat Tracker: Distillation, Experimentation, and (Continued) Integration of AI for Adversarial Use. https://cloud.google.com/blog/topics/threat-intelligence/distillation-experimentation-integration-ai-adversarial-use, February 2026.

Hartman, M., Jayaraman, V., Choraria, M., Savani, Y., and Varshney, L. R. Hiding in Plain Sight: Detectability-Aware Antidistillation of Reasoning Models. *arXiv preprint arXiv:2604.23238*, 2026.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Hönig, R., Rando, J., Carlini, N., and Tramèr, F. Adversarial Perturbations Cannot Reliably Protect Artists From Generative AI. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

Hulse, R., Newby, T. G., Meyer, S. P., and Tsang, F. DeepSeek, Model Distillation, and the Future of AI IP Protection, 2025. URL https://www.fenwick.com/insights/publications/deepseek-model-distillation-and-the-future-of-ai-ip-protection.

Jiang, B. DistillGuard: Evaluating Defenses Against LLM Knowledge Distillation. *arXiv preprint arXiv:2603.07835*, 2026.

Kim, A. The Claude Code Source Leak: fake tools, frustration regexes, undercover mode, and more. https://alex000kim.com/posts/2026-03-31-claude-code-source-leak/, March 2026.

Kirchenbauer, J., Geiping, J., Wen, Y., Katz, J., Miers, I., and Goldstein, T. A Watermark for Large Language Models. In *International Conference on Machine Learning (ICML)*, pp. 17061–17084. PMLR, 2023.

Li, P., Tan, Z., Zhang, M., Qu, H., Liu, H., and Chen, T. DOGe: Defensive Output Generation for LLM Protection Against Knowledge Distillation. *arXiv preprint arXiv:2505.19504*, 2025.

Ma, X., Yeoh, W., Zhang, N., and Vorobeychik, Y. Protecting Language Models Against Unauthorized Distillation through Trace Rewriting. *arXiv preprint arXiv:2602.15143*, 2026.

Nasr, M., Carlini, N., Sitawarin, C., Schulhoff, S. V., Hayes, J., Ilie, M., Pluto, J., Song, S., Chaudhari, H., Shumailov, I., et al. The attacker moves second: Stronger adaptive attacks bypass defenses against llm jailbreaks and prompt injections. *arXiv preprint arXiv:2510.09023*, 2025.

OpenAI. Terms of Use. https://openai.com/policies/row-terms-of-use/, 2026a.

OpenAI. Updated Stakes for American-Led, Democratic AI, February 2026b. URL https://assets.bwbx.io/documents/users/iqjWHBFdfxIU/rRmql_jJcxb4/v0. Memorandum to the U.S. House Select Committee on Strategic Competition between the United States and the Chinese Communist Party.

OpenAI Developers. Reasoning models: Reasoning summaries. https://developers.openai.com/api/docs/guides/reasoning#reasoning-summaries, 2026.

Peng, B., Li, C., He, P., Galley, M., and Gao, J. Instruction Tuning with GPT-4. *arXiv preprint arXiv:2304.03277*, 2023.

Qi, X., Wei, B., Carlini, N., Huang, Y., Xie, T., He, L., Jagielski, M., Nasr, M., Mittal, P., and Henderson, P. On Evaluating the Durability of Safeguards for Open-Weight LLMs. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

Savani, Y., Trockman, A., Feng, Z., Xu, Y. E., Schwarzschild, A., Robey, A., Finzi, M., and Kolter, J. Z. Antidistillation Sampling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., and Hashimoto, T. B. Alpaca: A Strong, Replicable Instruction-Following Model. https://crfm.stanford.edu/2023/03/13/alpaca.html, 2023.

Tramer, F., Carlini, N., Brendel, W., and Madry, A. On adaptive attacks to adversarial example defenses. *Advances in neural information processing systems*, 33:1633–1645, 2020.

Vercel. Anthropic Reasoning: Summarized vs. full thinking. https://vercel.com/docs/ai-gateway/capabilities/reasoning/anthropic#summarized-vs.-full-thinking, 2026.

Xu, Y. E., Kirchenbauer, J., Savani, Y., Trockman, A., Robey, A., Goldstein, T., Fang, F., and Kolter, J. Z. Antidistillation Fingerprinting. *arXiv preprint arXiv:2602.03812*, 2026.

Zhao, K., Li, L., Ding, K., Gong, N. Z., Zhao, Y., and Dong, Y. A Survey on Model Extraction Attacks and Defenses for Large Language Models. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '25, pp. 6227–6236, New York, NY, USA, 2025. Association for Computing Machinery. doi: 10.1145/3711896.3736573. URL https://doi.org/10.1145/3711896.3736573.

A. Antidistillation Sampling

Mechanism. Antidistillation Sampling (ADS) is a decoding-time defense that perturbs the teacher model's next-token distribution to reduce the effectiveness of model distillation by biasing sampling away from tokens that are informative to a student model.

Given a prompt x and partial generation $y_{<t}$, let $p_\theta(y_t \mid x, y_{<t})$ denote the teacher model's next-token distribution. Standard sampling draws $y_t \sim p_\theta$. ADS instead samples from a perturbed distribution $\tilde{p}_\theta$ defined as:

$$\tilde{p}_\theta(y_t \mid x, y_{<t}) \propto \exp\left(\frac{1}{\tau}\log p_\theta(y_t \mid x, y_{<t}) + \lambda \cdot \widehat{\Delta}(y_t \mid x, y_{<t})\right) \tag{1}$$

where τ is the temperature, $\lambda \geq 0$ controls the perturbation strength, and $\widehat{\Delta}(y_t \mid x, y_{<t})$ is an antidistillation penalty term.

The penalty $\widehat{\Delta}$ is computed using a proxy student model ϕ and its loss gradient $g = \nabla\ell(\phi)$. It approximates the directional derivative of the proxy's next-token log probabilities along g, which captures the rate at which the downstream loss would increase if the proxy were updated to favor token y_t:

$$\widehat{\Delta}(y_t \mid x, y_{<t}) = \frac{\log p_{\phi+\epsilon g}(y_t \mid x, y_{<t}) - \log p_{\phi-\epsilon g}(y_t \mid x, y_{<t})}{2\epsilon} \tag{2}$$

where ϵ is a small perturbation parameter and g is precomputed once. This requires two forward passes through the proxy model per generation step, evaluating at $\phi \pm \epsilon g$. Tokens with large positive $\widehat{\Delta}$ are those that would degrade the proxy's downstream performance if the student were trained on them. ADS up-weights these tokens in the sampling distribution, biasing the teacher toward outputs that are less informative for distillation.

The parameter λ controls the utility-distillability trade-off. At $\lambda = 0$, ADS reduces to standard temperature sampling. As λ increases, the teacher's output quality degrades while student distillation performance degrades faster. For ADS to be practically useful, the teacher must retain high utility (i.e., high accuracy). We set this boundary at 70% teacher accuracy, corresponding to a maximum absolute drop of 20%. Using the original paper's setup, this boundary corresponds to $\lambda \leq 0.107$. We call this the *high-utility regime* and focus our analysis on it.

This regime is highlighted in Figure 6. At equivalent teacher performance levels, ADS (green curve) achieves substantially lower student accuracy than temperature sampling (blue curve). When teacher accuracy decreases by approximately 20%, temperature sampling shows only a slight reduction in student accuracy, while ADS reduces student accuracy by approximately 40%, even dropping below the undistilled baseline.

Noisiness of ADS Generations. ADS outputs frequently contain recurring noisy artifacts. At structural boundaries, such as the start of the reasoning trace and paragraph transitions, the perturbation often injects semantically irrelevant, high-perplexity tokens that disrupt the coherent chain-of-thought structure. ADS also tends to produce long repetitive sequences of nonsensical tokens: once repetition begins, continuing the pattern becomes increasingly likely under the autoregressive model, often leading to continued repetition until the maximum generation length is reached, without producing further meaningful output. Concrete examples of antidistilled generations are provided in Appendix C.

B. Extended Results

B.1. Output Post-Processing

We assume black-box access, with query budget equal to data budget ($Q = D = 5{,}231$) as our threat model. As in previous work, we restrict each prompt to be queried exactly once. Since all local computation is free under our cost model, we evaluate six post-processing strategies applied to the collected ADS outputs before student training:

Repetition Deletion (RD). RD is applied iteratively for three passes, and in each pass several types of redundancy are removed: character-level repetitions are reduced by collapsing any sequence of five or more identical consecutive characters into a single character, consecutive identical lines are deleted while preserving newline markers (`\n`), and repeated token n-grams of up to length 6 are collapsed to a single occurrence.

RD + Perplexity filtering. After RD, we filter low-probability tokens using the student model in two passes. In the first pass, we build a context by discarding tokens whose perplexity given the evolving context exceeds a threshold τ_1. In the

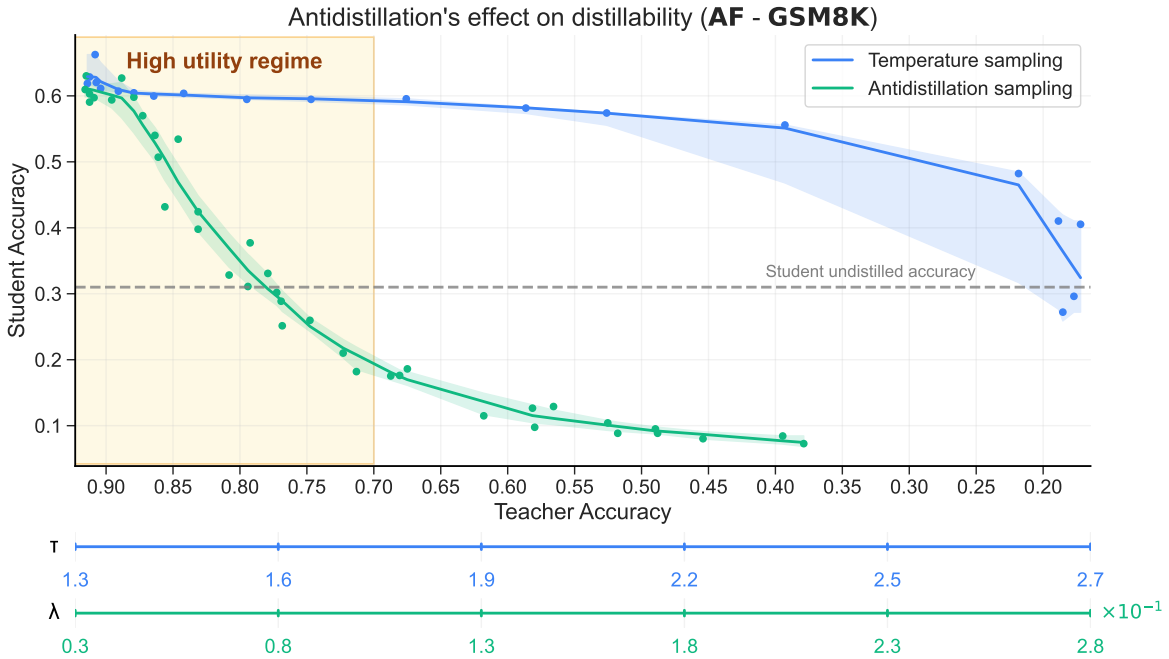


Figure 6. **Under Antidistillation Sampling (ADS), small reductions in teacher accuracy lead to disproportionately large student performance degradation.** On GSM8K (Answer Forcing evaluation), in the high-utility regime (teacher accuracy ≥ 0.7), an approximately 20% teacher accuracy reduction corresponds to a 40% student accuracy degradation, whereas temperature sampling produces negligible reduction in student accuracy across the same range.

second pass, we re-evaluate the original sequence using the first-pass context and remove tokens with perplexity above τ_2. The remaining tokens form the final trace.

Student-based rephrasing. We apply few-shot prompting with the student model to clean ADS traces. The prompt includes two examples of noisy traces paired with their cleaned versions, and the model is then used to rephrase new outputs accordingly. Using the student model ensures that no information beyond what is already available to the attacker is introduced into the trace.

LLM-based token deletion. We use an external LLM (GPT-4o mini) to identify noisy tokens from reasoning traces. Each trace is tokenized and represented as JSON, including each token's index and textual form. The LLM returns the indices of the tokens to delete, while the first boxed answer is masked so that it is preserved. This deletion-only design prevents the external model from injecting new information into the trace, making it safe to use a model different from the student model for that purpose.

Self-talk injection. Inspired by PART (Ding et al., 2025), which removes self-talk behaviors as a defense, we insert them as an attack. At paragraph and sentence boundaries, and immediately before reasoning transitions (*therefore, thus, so, then, next, first, finally*, =, \n), we insert phrases sampled uniformly from {*hmm, wait, okay, let me think, well, alright, right, so*} with probability 0.6, capping insertions at 12 per trace.

Incorrect-trace filtering. We discard traces whose final answers are incorrect.

Results. Figure 7 compares these strategies. LLM-based token deletion combined with repetition deletion reduces the effect of ADS to approximately linear degradation: reductions in teacher accuracy translate into roughly equal reductions in student accuracy. Student-based rephrasing shows a similar trend in the high-utility range before remaining roughly constant at around 0.4. Self-talk injection combined with repetition deletion offers little improvement over repetition deletion alone, whereas perplexity filtering has a negative effect when combined with repetition deletion. Incorrect-trace filtering produces only marginal improvements. In Section 4, we use repetition deletion.

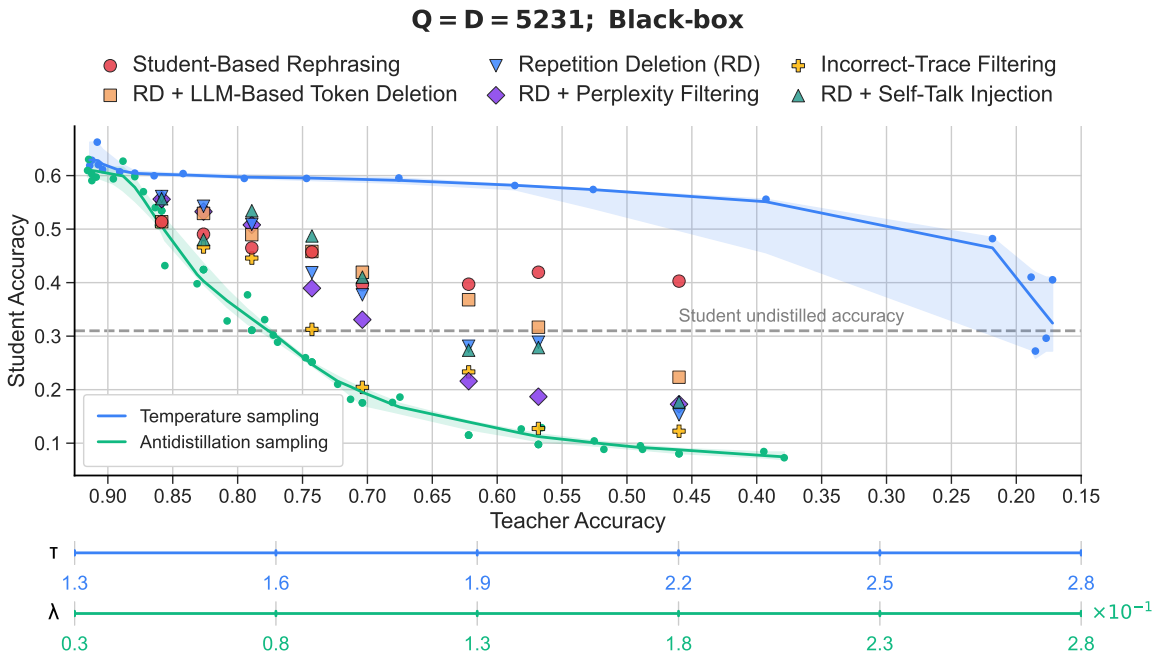


Figure 7. **Comparison of post-processing strategies against ADS.** Student accuracy on GSM8K across the full teacher accuracy range. The threat model here is $Q = D = 5231$, black-box access, and each prompt is queried exactly once. Post-processing strategies considerably weaken the effect of ADS, particularly in the high-utility regime.

B.2. Weak-Attacker Model

We consider the same black-box threat model as in the main experiments with a fixed query budget equal to the dataset size ($Q = D = 5{,}231$). Figure 8 extends the main-text results to the full λ range. The differences between resampling strategies remain small throughout. Figure 9 shows how the query budget is distributed across prompts in each strategy. The height of each bar indicates the total number of prompts queried at least once (i.e., the number of traces in the student training dataset), while the stacked colors within each bar show how that budget is allocated: specifically, how many prompts received exactly $1, 2, \ldots, k+1$ generations in total. Increasing k concentrates more budget on initially incorrect prompts at the expense of prompt coverage.

B.3. Scaling the Query Budget

Max-2 resampling. Figure 10 extends the main-text results to the full λ range. Student accuracy increases consistently with Q across most λ values, confirming the trend reported in the main text. For the largest λ value, differences across query budgets become negligible, as the ADS perturbation is strong enough that resampling rarely recovers correct or clean traces regardless of budget. We report the matched-utility budget only for $\lambda \leq 0.107$, where at least 90% of prompts are solved within 16 sampling attempts, ensuring that the empirical mean is based on the vast majority of cases and remains a meaningful lower bound. For larger λ values, this condition fails. In particular, at $\lambda = 0.182$ only 67% of the prompts are solved within 16 attempts, so the average requery count cannot be reliably estimated from our data. Crucially, even the conservative estimates that we can compute (see Figure 16) already exceed the highest query budget shown, implying that the matched-utility budget would lie beyond $Q = 4D$.

Figure 12 shows how the query budget is distributed across traces under each value of Q. For $Q \geq 3D$, every prompt has been queried at least once and the remaining budget is spent on resampling.

Max-1 resampling. Figure 11 extends the analysis to max-1 resampling across varying query budgets. Student accuracy is very similar to the max-2 results, with no meaningful difference between the two strategies at any λ value, further confirming that the gain from increasing Q is not sensitive to the exact resampling cap. Figure 12 shows that under max-1 resampling the number of final traces already reaches its maximum at $Q = 2D$, after which all budget is spent on resampling rather than expanding prompt coverage.

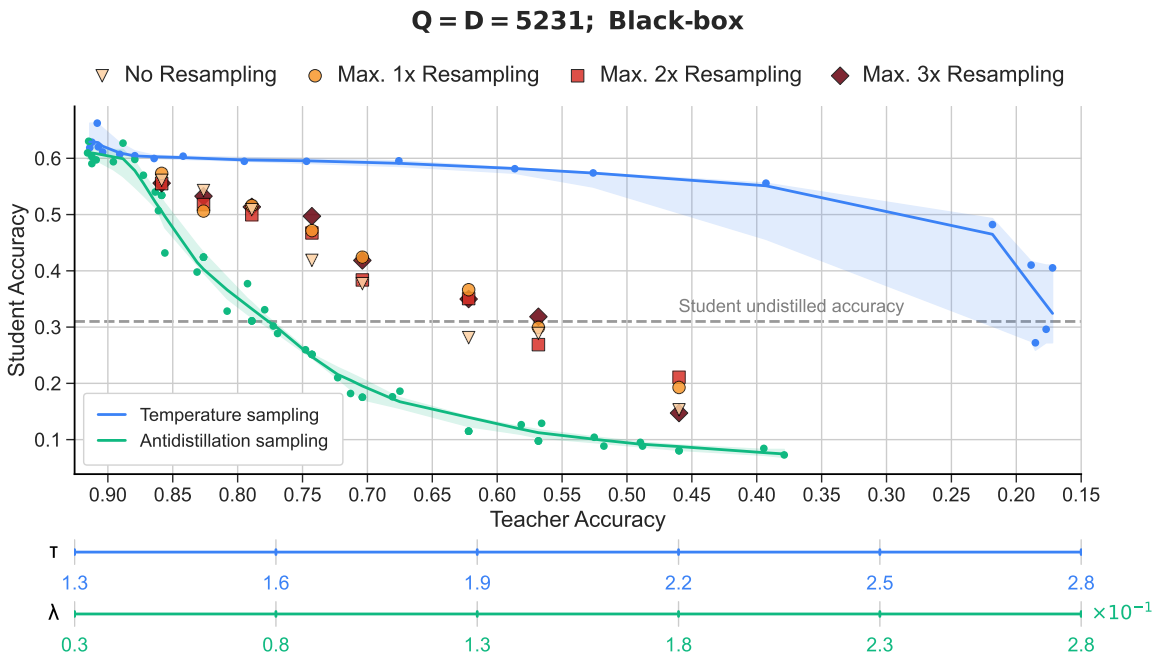


Figure 8. **Different resampling strategies yield comparable student accuracy.** Student accuracy on GSM8K under black-box max-k resampling ($k \in \{1, 2, 3\}$) and no resampling, with repetition deletion applied in all cases, across the full λ range ($Q = D = 5{,}231$). Differences between resampling strategies remain small across all λ values.

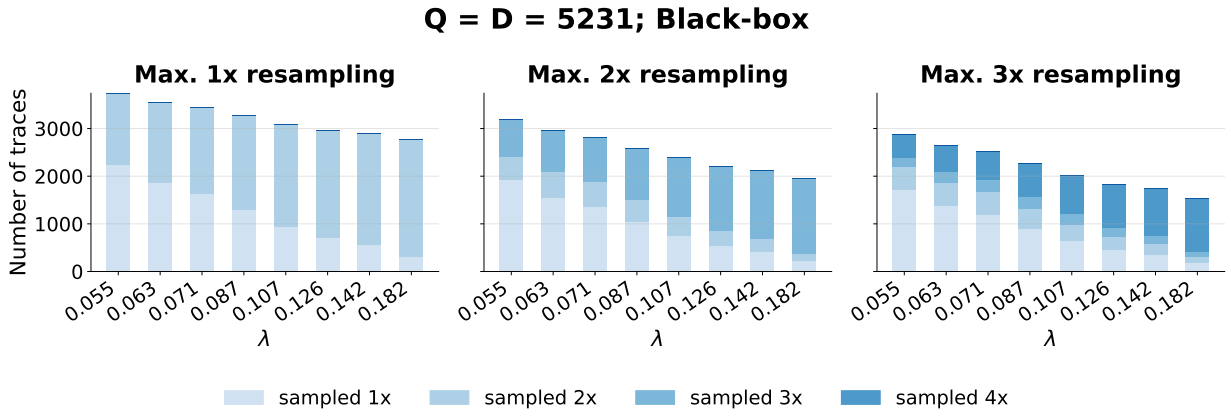


Figure 9. **Higher k concentrates more budget on initially incorrect prompts, at the cost of reduced prompt coverage.** Distribution of query budget allocation across prompts under max-k resampling for $k \in \{1, 2, 3\}$. Bars indicate how many prompts receive 1 to $k + 1$ generations.

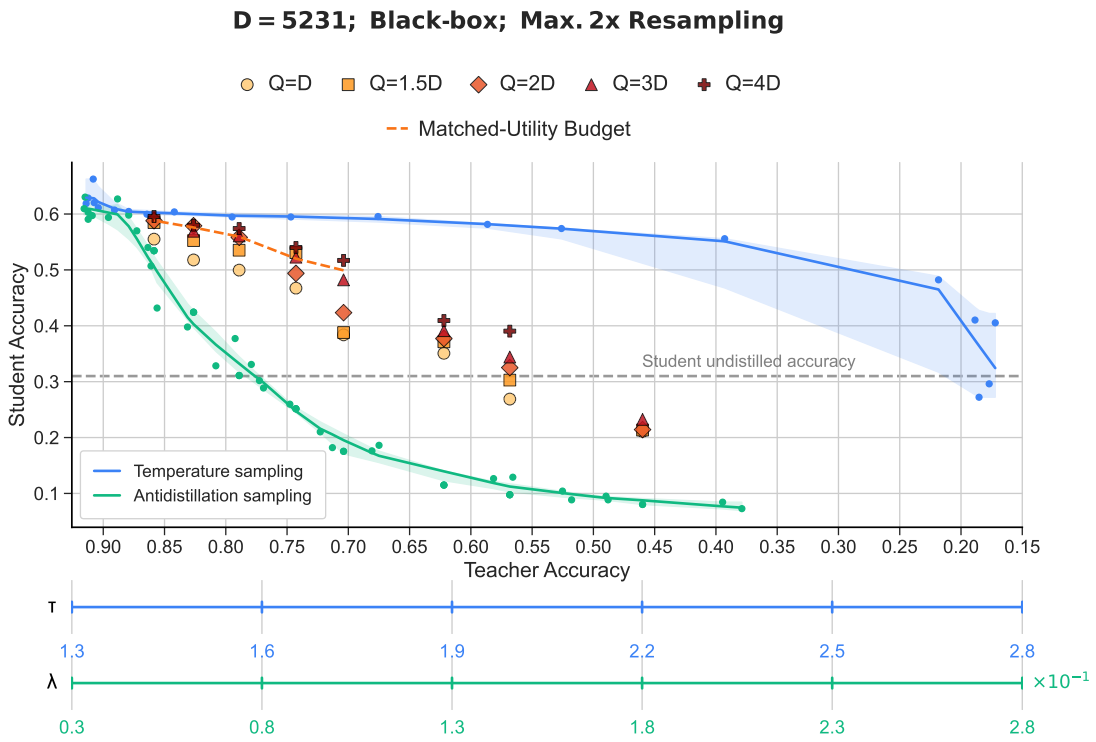


Figure 10. **Increasing query budget improves student accuracy, with smaller gains under stronger perturbation.** Student accuracy under max-2 resampling with repetition deletion, for varying query budgets $Q \in \{D, 1.5D, 2D, 3D, 4D\}$ across the full λ range under black-box access. The matched-utility budget is shown only for λ values where at least 90% of prompts are solved within 16 sampling attempts. At high λ, differences across query budgets become negligible, as perturbation prevents resampling from leading to useful traces.

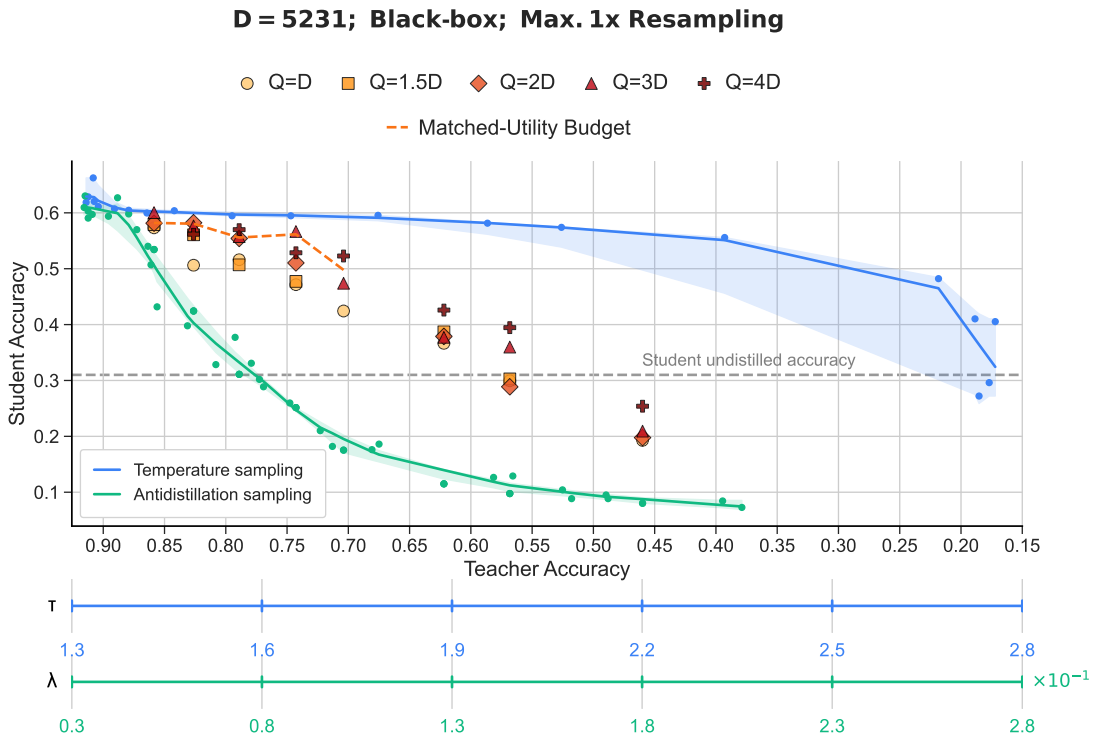


Figure 11. **Max-1 resampling yields student accuracy nearly identical to max-2.** Student accuracy under max-1 resampling with repetition deletion, for varying query budgets $Q \in \{D, 1.5D, 2D, 3D, 4D\}$, across the full λ range.

B.4. Interface Profile: Enabling Prefill Access

Figure 13 extends the main-text results to the full λ range, confirming that prefill access with the prefix "First" yields consistently higher student accuracy across all λ values. As in the black-box setting, we report the matched-utility budget only for λ values where at least 90% of prompts are solved within 16 resampling attempts. Here, this threshold is met for one additional point: at $\lambda = 0.126$, 93.0% of prompts are solved within 16 attempts, so we include it in the matched-utility curve. The matched-utility line lies higher than in the black-box setting, meaning that ADS is less effective under prefill access. Figure 12 shows that the distribution of resampling attempts does not differ significantly from the black-box setting.

Figure 14 provides insight into why setting "First" as a prefix works. ADS perturbs the teacher's next-token distribution by adding a directional term that up-weights tokens estimated to increase the student's loss. Whether this perturbation actually flips the argmax depends on how concentrated the original distribution is. When the entropy H_t is high, the probability mass is spread across many alternatives, and the gap between the most likely token and its alternatives is small. Even a modest perturbation can flip the argmax. As shown in Figure 14a, the fraction of positions at which ADS changes the argmax increases with H_t, indicating that more positions in the trace are actively corrupted when the teacher is uncertain.

Conditioning on the prefix "First" reduces this effect. If the initial token is fixed, the entropy of the teacher's distribution at subsequent positions decreases (see Figures 14b and 14c). Lower entropy implies a larger average gap between the argmax token and its alternatives, making ADS-induced argmax flips less likely. Figure 14c confirms this: the fraction of positions at which the argmax changes is systematically lower under prefill access than under black-box access. Consequently, the generated trace remains closer to what the teacher would have produced without ADS, yielding cleaner reasoning steps, more correct traces (see Figures 15 and 17), and higher student accuracy.

B.5. Further Analysis

Figure 15 shows how the number of incorrect traces decreases across successive resampling attempts, comparing standard resampling against resampling with the prefix "First". Both strategies exhibit exponential-like decay and most of the reduction occurs within the first two attempts, which is why we focus on max-1, max-2, and max-3 resampling in the main

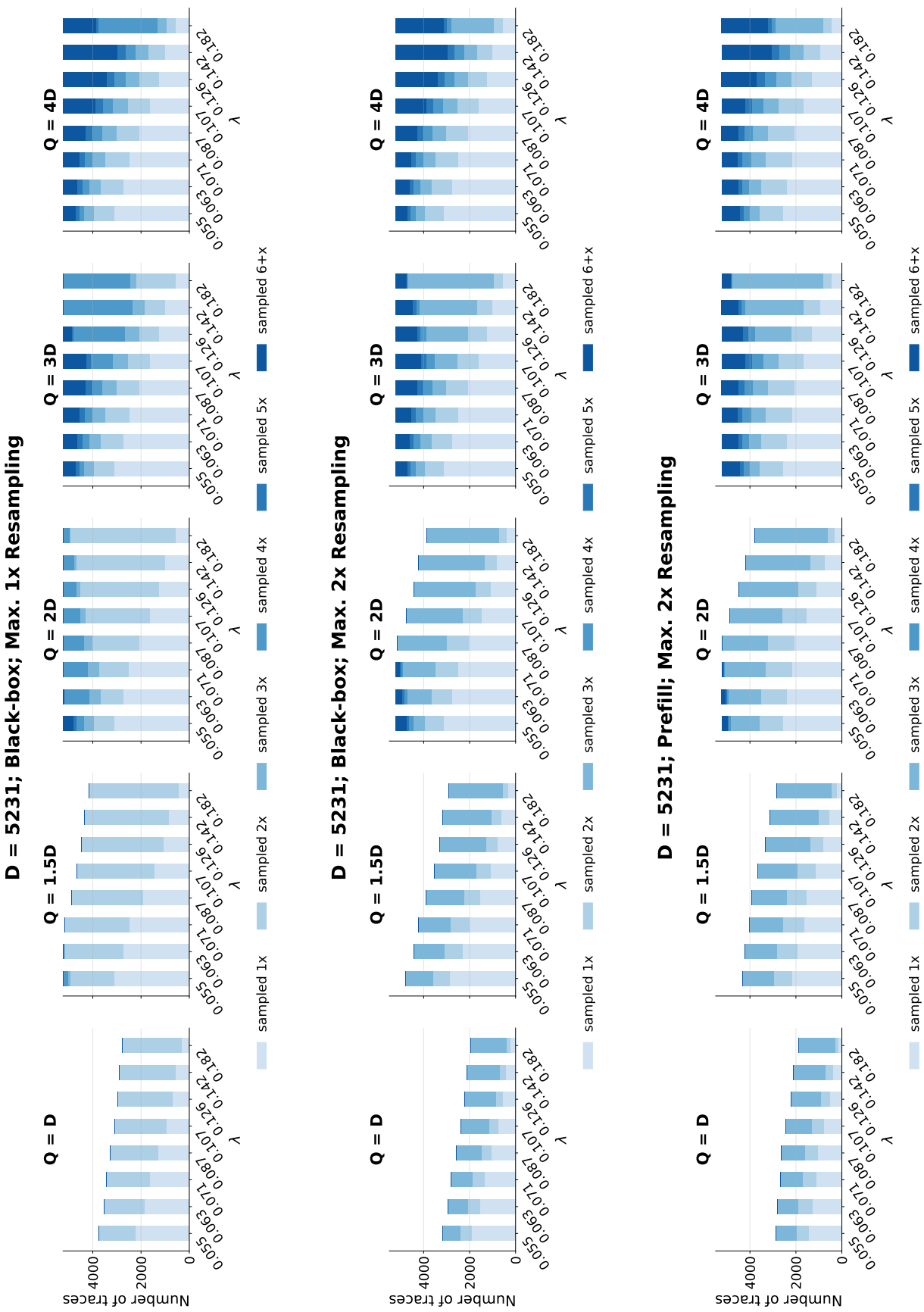


Figure 12. **Query budget distribution across prompts.** Per-prompt generation counts under max-1 resampling (**top**) and max-2 resampling for black-box (**middle**) and prefill access (**bottom**), for query budgets $Q \in \{D, 1.5D, 2D, 3D, 4D\}$.

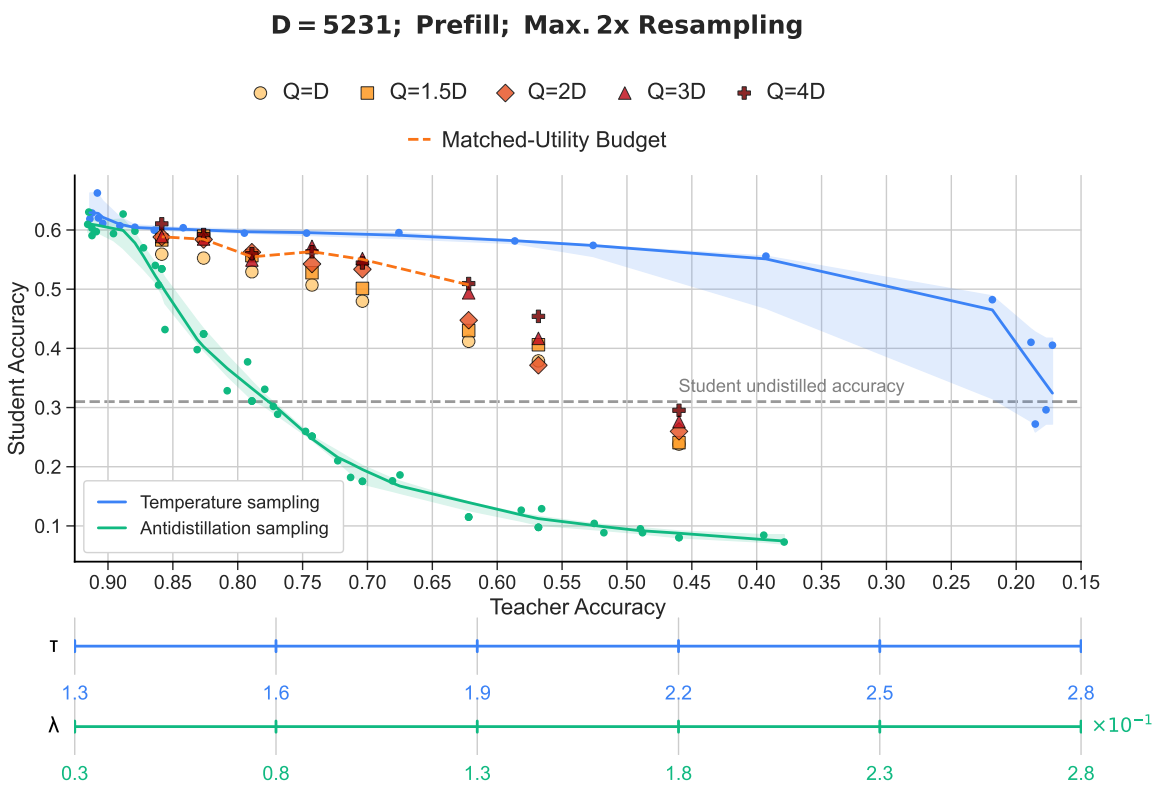
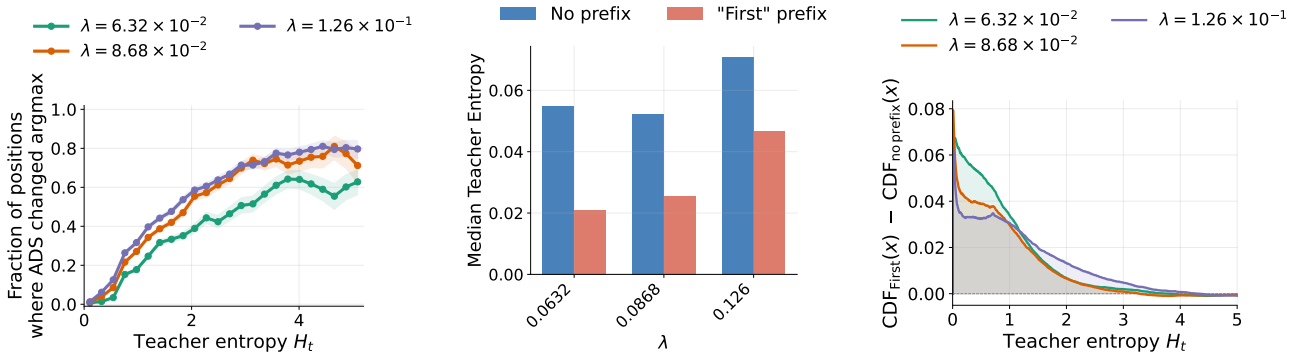


Figure 13. **Prefill access yields consistently higher student accuracy than black-box access (Figure 10) at the same query budget.** Student accuracy under max-2 resampling with repetition deletion and prefill access (prefix "First"), for varying query budgets $Q \in \{D, 1.5D, 2D, 3D, 4D\}$, across the full λ range. The matched-utility budget is shown for λ values where at least 90% of prompts are solved within 16 resampling attempts.



(a) Fraction of positions where ADS changes the argmax token as a function of teacher entropy H_t, under black-box access. Higher entropy corresponds to a flatter next-token distribution, reducing the gap between the top token and its competitors and making it easier for ADS's perturbation to flip the argmax.

(b) Median teacher entropy across all generated tokens, for each λ value, with and without the "First" prefix. The prefix consistently lowers entropy, reducing ADS's ability to flip the argmax.

(c) Difference in empirical CDFs as a function of teacher entropy. Positive values indicate the "First" prefix shifts probability mass toward lower-entropy distributions, reducing ADS effectiveness.

Figure 14. **Effect of teacher entropy and prefill access on ADS's ability to change the argmax token.** When the next-token distribution is uncertain (high entropy), ADS can more easily redirect generation toward student-harming tokens. Conditioning on a single prefix token reduces entropy at generated positions, limiting ADS's effective perturbation rate.

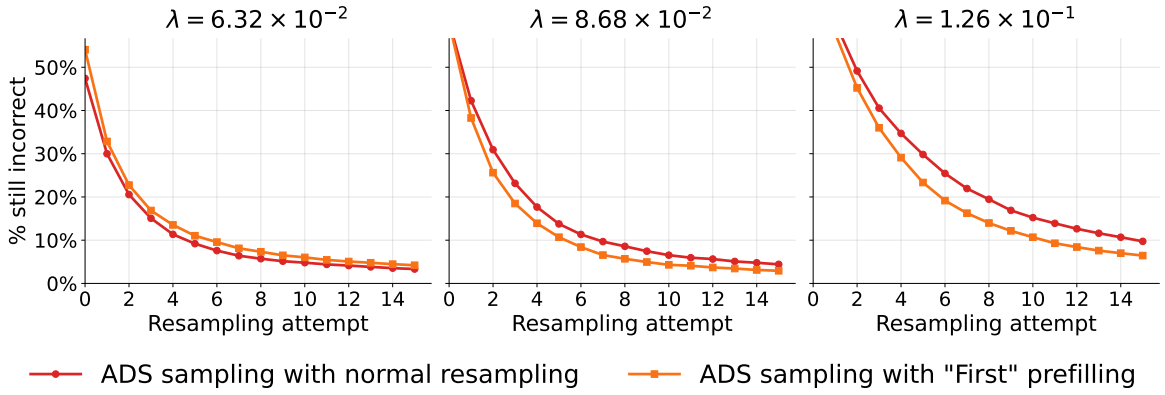


Figure 15. **Most incorrect traces are corrected within two resampling attempts.** Fraction of incorrect traces remaining across successive resampling attempts, for standard resampling and resampling with prefix "First", at three representative λ values. Both strategies exhibit exponential-like decay, with most of the reduction occurring within the first two attempts. Prefill access accelerates convergence at high λ values but offers little additional benefit at the smallest λ.

text and do not consider higher values. Response prefilling leads to faster decay at moderate and large λ values, while offering little additional benefit at the smallest λ, where most traces are still very clean.

Figure 16 reports the expected number of queries needed to obtain a correct answer as a function of λ for ADS and τ for temperature sampling. The expected query count grows monotonically with λ and τ. As discussed above, these estimates are based only on prompts solved within 16 attempts and therefore underestimate the true expected query count, increasingly so at larger λ and τ values.

Figure 17 shows that increasing the query budget leads to a higher fraction of correct traces in the training dataset. The effect is largest at moderate λ values, where the ADS perturbation is neither too weak nor too strong for resampling to make a difference. Prefill access further increases the fraction of correct traces at moderate λ values, particularly for $Q \geq 3D$.

B.6. Effect of Resampling under Temperature Sampling

For comparison, we also plot the matched-utility budget curve for temperature sampling in Figure 18, again with repetition deletion and max-2 resampling applied. Unlike the ADS matched-utility curve, the matched-utility curve is almost identical to the standard temperature-sampling curve. This suggests that resampling has a much smaller effect under temperature sampling than under ADS.

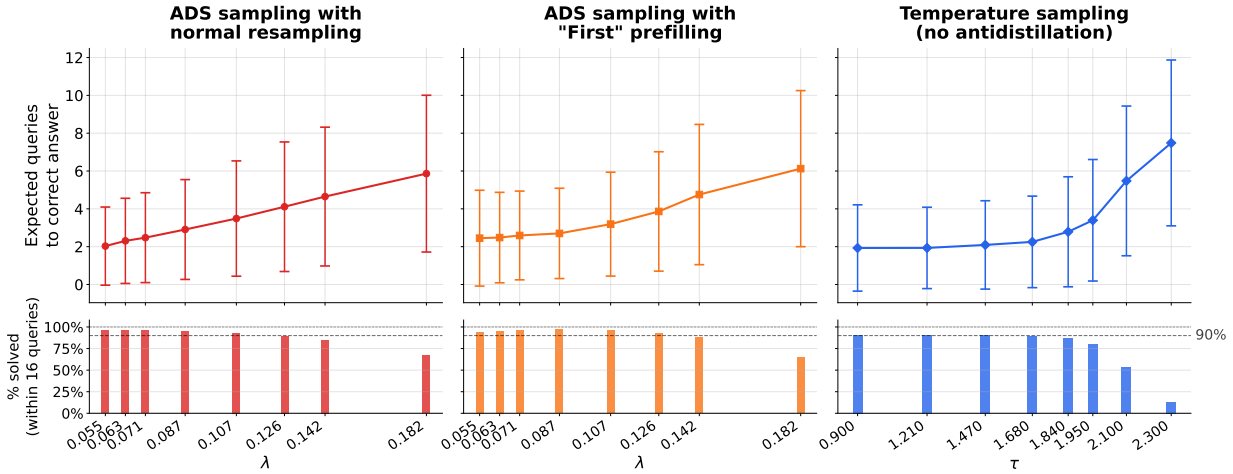


Figure 16. **Expected queries to a correct answer grow monotonically with perturbation strength.** Expected number of queries to a correct answer as a function of λ for ADS (standard black-box resampling and prefill resampling with prefix "First"), and as a function of τ for temperature sampling. Error bars denote ± 1 standard deviation. As λ increases, the expected number of queries grows monotonically for all three methods. Only traces that achieve a correct answer within 16 queries are included in this estimate (bottom panels show the fraction of such traces). Since this fraction decreases substantially with λ and τ, the reported means are computed over an increasingly selective subset and therefore underestimate the true expected query count.

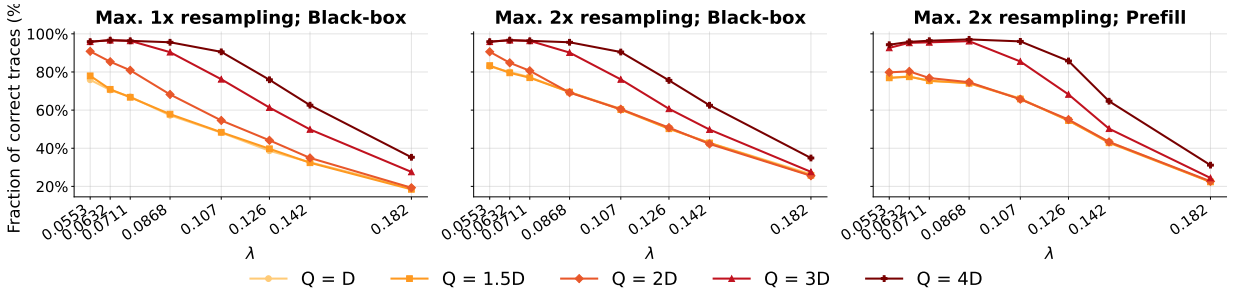


Figure 17. **Increasing query budget recovers more correct traces, with the largest gains at moderate λ.** Fraction of correct traces retained after max-2 resampling, comparing black-box and prefill access across query budgets $Q \in \{D, 1.5D, 2D, 3D, 4D\}$ and λ values. The effect of increasing Q is biggest at moderate λ values, where ADS's perturbation is neither too weak nor too strong for resampling to make a meaningful difference. Prefill access further increases the fraction of correct traces, particularly at higher query budgets.

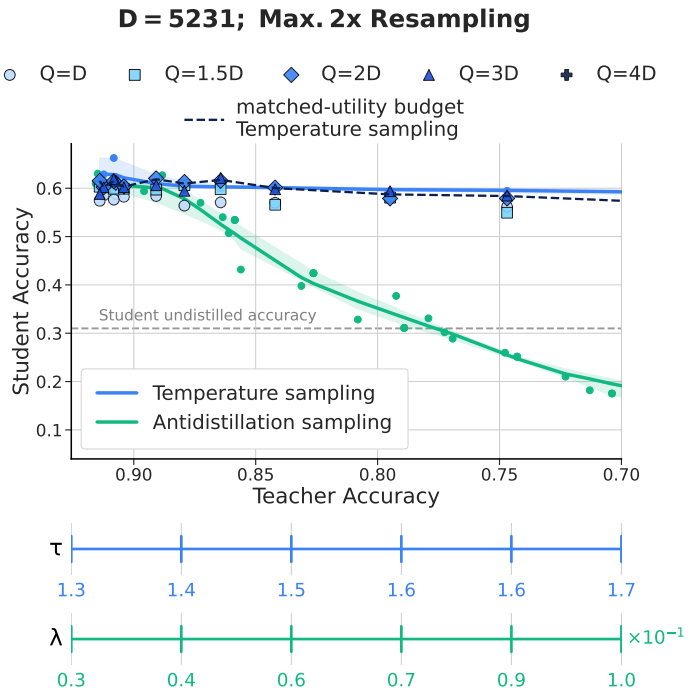


Figure 18. **Resampling has negligible effect under temperature sampling.** Student accuracy under max-2 resampling with repetition deletion for varying query budgets $Q \in \{D, 1.5D, 2D, 3D, 4D\}$ under temperature sampling. Unlike ADS, the matched-utility curve is nearly identical to the standard temperature-sampling curve.

C. Example of Attacked ADS Traces

Repetition Deletion (RD)

Question

Larry spends half an hour twice a day walking and playing with his dog. He also spends a fifth of an hour every day feeding his dog. How many minutes does Larry spend on his dog each day?

ADS Corrupted Trace

$\lambda = 8.68 \cdot 10^{-2}$; Correct answer

```
Larry walks and plays with his dog for half an hour twice a
day. So, that's 2 times 30 minutes, which equals 60 minutes.

Additionally, he spends a fifth of an hour feeding his dog
every day. Since an hour is 60 minutes, a fifth of that is 12
minutes.

Adding both activities together, Larry spends 60 minutes
walking and playing plus 12 minutes feeding, which totals 72
minutes each day on his dog.
PalestLarry spends a total of 72 minutes each day on his dog.
\boxed{72} minutes
Palest
Palest
\boxed{72} minutes
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
xD
```

Attacked Trace

Repetition Deletion

```
Larry walks and plays with his dog for half an hour twice a
day. So, that's 2 times 30 minutes, which equals 60 minutes.

Additionally, he spends a fifth of an hour feeding his dog
every day. Since an hour is 60 minutes, a fifth of that is
12 minutes. Adding both activities together, Larry spends 60
minutes walking and playing plus 12 minutes feeding, which
totals 72 minutes each day on his dog.

PalestLarry spends a total of 72 minutes each day on his dog.
\boxed{72} minutes
Palest
Palest
\boxed{72} minutes
xD
```

Repetition Deletion + Perplexity Filtering

Question

Ralph is going to practice playing tennis with a tennis ball machine that shoots out tennis balls for Ralph to hit. He loads up the machine with 175 tennis balls to start with. Out of the first 100 balls, he manages to hit 2/5 of them. Of the next 75 tennis balls, he manages to hit 1/3 of them. Out of all the tennis balls, how many did Ralph not hit?

ADS Corrupted Trace

$\lambda = 8.68 \cdot 10^{-2}$; Correct answer

```
kitsRa Ralph starts with 175 tennis balls.

kitHis kityFirst, he hits 2/5 of the first 100 balls.

larkThen, he hits 1/3 of the next 75 balls.

We calculate the number of balls he hit in each set:

- For the first set: (2/5) * 100 = 40 balls.
kitisHis hit
- For the second set: (1/3) * 75 = 25 balls.

larkTotal balls he hit: 40 + 25 = 65 balls.

Therefore, the number of balls he did not hit is 175 - 65 = 110
balls.

Wisconsinthis snow has been such an unusually warm winter in
Wisconsin. I have to stay indoors because of it. Hoping spring
doesn't come?
顿顿
otin WI, winters are unpredictable. Maybe spring is still on
its way. Either way, I have to use the heating.
\boxed{110}
顿顿
顿顿
\boxed{110}
树德树人
树德树人
树德树人

板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
```

Attacked Trace

RD + Perplexity Filtering

```
with 175 tennis balls.

, he hits 2/5 of the first 100 balls.
Then, he hits 1/3 of the next 75 balls.
calculate the number of balls he hit in each set:
- For the first set: (2/5) * 100 = 40 balls.

- For the second set: (1/3) * 75 = 25 balls.
arkTotal balls he hit: 40 + 25 = 65 balls.
, the number of balls he did not hit is 175 - 65 = 110 balls.
consin been an warm winter in Wisconsin. I have to stay indoors
because of it.ing spring't come
, are. Maybe spring is still on its way. way, I have to the.
\boxed{110}
顿
顿
boxed{110}
树树
板球球
板球球
```

Student-Based Rephrasing

Question

Tina makes $18.00 an hour. If she works more than 8 hours per shift, she is eligible for overtime, which is paid by your hourly wage + 1/2 your hourly wage. If she works 10 hours every day for 5 days, how much money does she make?

ADS Corrupted Trace

$\lambda = 8.68 \cdot 10^{-2}$; Correct answer

```
stampedTina makes $18.00 per hour. To find her daily earnings,
we need to calculate her regular pay for the first 8 hours and
her overtime pay for the additional hours. \n\nFor the first 8
hours, her regular pay is:
$18.00/hour × 8 hours = $144.00\n\nFor the remaining 2 hours,
her overtime pay is:
$18.00/hour × 1.5 = $27.00/hour\n$27.00/hour × 2 hours =
$54.00\n\nHer total daily earnings are:
$144.00 + $54.00 = $198.00\nStultipwę
umping
Tina works 10 hours each day for 5 days, so she has 2 extra
hours each day. For each of those extra hours, she is paid
at 1.5 times her regular wage. Her regular wage is $18.00, so
overtime is 18 × 1.5 = $27.00 per overtime hour. For 2 overtime
hours per day, that's 27 × 2 = $54.00 extra per day. Over 5
days, her total overtime pay is 54 × 5 = $270.00. Adding her
regular pay for 10 hours (which is 18 × 10 = $180.00), her
total earnings are 180 + 270 = $450.00. $oteStultip stamped
umping
Damße Tina earns $18.00 per hour. For the first 8 hours each
day, she earns:
$18.00 × 8 = $144.00
For the remaining 2 hours each day, she earns overtime pay:
$18.00 × 1.5 = $27.00 per hour
$27.00 × 2 = $54.00 per day
Her total daily earnings are: $144.00 + $54.00 = $198.00
Over 5 days, her total earnings are:
$198.00 × 5 = $990.00
So Tina makes a total of $\boxed{990}$ dollars for working 10
hours a day for 5 days. Stultip stamped
[K.O.P.]
stamped
ete
Tina makes $18.00 per hour. She works 10 hours each day for 5
days, with 2 hours of overtime each day. Her regular pay is:
$18.00 × 8 hours = $144.00
Her overtime pay per hour is: $18.00 × 1.5 = $27.00
For the 2 overtime hours each day: $27.00 × 2 = $54.00
Her total daily pay is: $144.00 + $54.00 = $198.00
Over 5 days: $198.00 × 5 = $990.00
Tina earns $\boxed{990}$ dollars in total. Stultip stamped
[K.O.P.]
[K.O.P.]
[K.O.P.]
[K.O.P.]
[K.O.P.]
stamps
ete
ete
ete
ete
Stultip
K.O.P.
stamps
stamped
ete
ete
stamps
stamped
K.O.P.
stamps
K.O.P.
stamps
K.O.P.
stamps
K.O.P.
```

Attacked Trace

Student-Based Rephrasing

```
Tina makes $18.00 per hour. She works 10 hours each day for 5
days, with 2 hours of overtime each day. Her regular pay is:
$18.00 × 8 hours = $144.00
Her overtime pay per hour is:
$18.00 × 1.5 = $27.00
For the 2 overtime hours each day:
$27.00 × 2 = $54.00
Her total daily pay is:
$144.00 + $54.00 = $198.00
Over 5 days:
$198.00 × 5 = $990.00
Tina earns $990.00 in total.
\boxed{990}
```

Repetition Deletion + LLM-Based Token Deletion

Question

Ralph is going to practice playing tennis with a tennis ball machine that shoots out tennis balls for Ralph to hit. He loads up the machine with 175 tennis balls to start with. Out of the first 100 balls, he manages to hit 2/5 of them. Of the next 75 tennis balls, he manages to hit 1/3 of them. Out of all the tennis balls, how many did Ralph not hit?

ADS Corrupted Trace

$\lambda = 8.68 \cdot 10^{-2}$; Correct answer

```
kitsRa Ralph starts with 175 tennis balls.

kitHis kityFirst, he hits 2/5 of the first 100 balls.

larkThen, he hits 1/3 of the next 75 balls.

We calculate the number of balls he hit in each set:

- For the first set: (2/5) * 100 = 40 balls.
kitisHis hit
- For the second set: (1/3) * 75 = 25 balls.

larkTotal balls he hit: 40 + 25 = 65 balls.

Therefore, the number of balls he did not hit is 175 - 65 = 110
balls.

Wisconsinthis snow has been such an unusually warm winter in
Wisconsin. I have to stay indoors because of it. Hoping spring
doesn't come?
頓頓
otin WI, winters are unpredictable. Maybe spring is still on
its way. Either way, I have to use the heating.
\boxed{110}
頓頓
頓頓
\boxed{110}
树德树人
树德树人
树德树人

板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
板球球棒
```

Attacked Trace

RD + LLM-Based Token Deletion

```
kitsRa Ralph starts with 175 tennis balls kitHis kityFirst, he
hits 2/5 of the first 100 balls.
arkThen, he hits 1/3 of the next 75 ballsWe calculate the
number of balls he hit in each set:
- For the first set: (2/5) * 100 = 40 ballskitisHis hit
- For the second set: (1/3) * 75 = 25 ballslarkTotal balls he
hit: 40 + 25 = 65 ballsTherefore, the number of balls he did
not hit is 175 - 65 = 110 balls.
I have to stay indoors because of it Hoping spring doesn't
comein way I have to use the heating\boxed{110}
頓頓
顿}
树德树人
板球球
\boxed{1920} stamped
```

Repetition Deletion + Self-Talk Token Injection

Question

Carolyn practices the piano for 20 minutes a day and the violin for three times as long. If she practice six days a week, how many minutes does she spend practicing in a month with four weeks?

ADS Corrupted Trace

$\lambda = 8.68 \cdot 10^{-2}$; Correct answer

stampedCarolyn practices piano for 20 minutes daily. Violin practice is three times longer than piano, so that's 20 * 3 = 60 minutes per day. Combining both, she practices 20 + 60 = 80 minutes daily.

She practices six days a week, so in a week she spends 80 * 6 = 480 minutes.

The month has four weeks, so the total minutes she practices in the month are 480 * 4 = 1920 minutes.

stamped stamped stamped stamped stamped
stamped
圜 stamped dAns:
\boxed{1920}
stamped stampedCarolyn practices piano for 20 minutes daily. Violin practice is three times longer than piano, so that's 20 * 3 = 60 minutes per day. Combining both, she practices 20 + 60 = 80 minutes daily.

She practices six days a week, so in a week she spends 80 * 6 = 480 minutes.

The month has four weeks, so the total minutes she practices in the month are 480 * 4 = 1920 minutes.

stamped stamped stamped stamped stamped
stamped
圜 stamped dAns:
\boxed{1920} stamped

Attacked Trace

RD + Self-Talk Token Injection

let me think stampedCarolyn practices piano for 20 minutes daily. hmm Violin practice is three times longer than piano, so that's 20 * 3 wait = 60 minutes per day. Combining both, she practices 20 + 60 okay = 80 minutes daily.

well She practices six days a week, so in a week she spends 80 * 6 so = 480 minutes.

let me think The month has four weeks, so the total minutes she practices in the month are 480 * 4 alright = 1920 minutes.

stamped stamped stamped
圜 stamped dAns:
\boxed{1920} stamped stampedCarolyn practices piano for 20 minutes daily. Violin practice is three times longer than piano, so that's 20 * 3 = 60 minutes per day.
Combining both, she practices 20 + 60 = 80 minutes daily.

She practices six days a week, so in a week she spends 80 * 6 = 480 minutes.

The month has four weeks, so the total minutes she practices in the month are 480 * 4 = 1920 minutes.

stamped stamped
stamped
圜 stamped dAns:
\boxed{1920} stamped

Fixed-Start Resampling + Repetition Deletion

Question

Alexis is applying for a new job and bought a new set of business clothes to wear to the interview. She went to a department store with a budget of $200 and spent $30 on a button-up shirt, $46 on suit pants, $38 on a suit coat, $11 on socks, and $18 on a belt. She also purchased a pair of shoes, but lost the receipt for them. She has $16 left from her budget. How much did Alexis pay for the shoes?

ADS Corrupted Trace

$\lambda = 8.68 \cdot 10^{-2}$; Incorrect answer

```
Alexis started with a budget of $200. \naziAlexis spent $30 on
a button-up shirt, $46 on suit pants, $38 on a suit coat, $11
on socks, and $18 on a belt. To find the total amount spent on
these items, we add them together: \nwp$30 + $46 + $38 + $11 +
$18 = $145. \nWat
Engines
Initially, Alexis has a budget of $200. She spent $145 on her
clothes before purchasing the shoes. If she has $16 left after
buying the shoes, we can subtract the remaining amount from the
total spent to find the cost of the shoes. \n\wp$145 (total
spent) - $16 (remaining) = $129. \nThus, Alexis paid $129 for
the shoes.
aziaziANSWER: \boxed$129aziWat
äsiAlexis spent $145 on her initial purchases. She has $16
left after buying the shoes. To find the cost of the shoes, we
subtract the remaining amount from the total spent: \nwp$145 -
$16 = $129. \nTherefore, Alexis paid $129 for the shoes.
äsiäzi
äsiäzi
äsiäzi
äsiäzi
äsiäzi
äsiäzi
äsiäzi
äsiäzi
äsiäzi
äsiäzi
äsiäzi
äsiäzi
äsiäzi
äsiäzi
äsiäzi
äsiäzi
äsiäzi
azixxai
Step 憋 Step-by-step explanation:

1. Alexis had a total budget of $ 憋 200.
2. She spent $30 on a button-up shirt.
3. She spent $46 on suit pants.
4. She spent $38 on a suit coat.
5. She spent $11 on socks.
6. She spent $18 on a belt.
7. The total amount spent on these items is $30 + $46 + $38 +
$11 + $18 = $145.
8. She has $16 left after buying the shoes.
9. The amount spent on the shoes is the total budget minus the
remaining amount after all purchases: $200 - $145 - $16 = $129.
PeteStep Pete

Alexis started with a budget of $200. She spent $145 on her
initial purchases (shirt, pants, coat, socks, and belt). She
has $16 left after buying the shoes. To find the cost of the
shoes, subtract the remaining amount from the total budget:
\[
\text{Cost of Shoes} = \text{Total Budget} - \text{Total Spent
on Initial Purchases} - \text{Remaining Amount}
\]

\[
\text{Cost of Shoes} = 200 - 145 - 16 = 129
\]

Therefore, Alexis paid \boxed\$12
```

Attacked Trace

Fixed-Start Resampling + RD; Correct answer

```
First, Alexis's total budget is $200.
WernerAlexis spent:
$30 on a button-up shirt,
$46 on suit pants,
$38 on a suit coat,
$11 on socks,
and $18 on a belt.
Adding these expenses: 30 + 46 = 76, 76 + 38 = 114, 114 + 11 =
125, 125 + 18 = 143.
Alexis has $16 left after her purchases.
WernerThe total amount spent is the budget minus the remaining
money: 200 - 16 = 184.
Alexis already spent $143 on her other items, so the amount
spent on shoes is: 184 - 143 = 41.
Alexis paid $41 for the shoes. Werner Werner
梓 Alexis's total budget is $200. She spent:
$30 on a button-up shirt,
$46 on suit pants,
$38 on a suit coat,
$11 on socks,
$18 on a belt.
Adding these amounts: 30 + 46 = 76, 76 + 38 = 114, 114 + 11 =
125, 125 + 18 = 143.
Alexis has $16 remaining after her purchases.
The total amount she has spent is the budget minus the
remaining money: 200 - 16 = 184.
To find out how much she spent on shoes, subtract the amount
she already spent on other items from the total amount spent:
184 - 143 = 41.
Therefore, Alexis paid $41 for the shoes.
梓
梓 Alexis paid $\boxed{41}$ dollars for the shoes.
廢 Alexis paid $41 for the shoes.
梓 Alexis paid $\boxed{41}$ dollars for the shoes.
廢廢 Alexis paid $41 for the shoes.
梓 Alexis paid $\boxed{41}$ dollars for the shoes.
梓 V Werner Werner V
FounderAlexis paid $\boxed{41}$ dollars for the shoes.
V Werner Werner V FounderV Werner
```

D. Notes on the Graphs

We follow Savani et al. (2025) in constructing the visualizations. Specifically, we report the mean and 95% confidence intervals computed over bootstrapped LOWESS fits. For the additional τ and λ axes shown in the figures, we use linear regression, i.e., we fit $\lambda = \beta_0 + \beta_1 \cdot$ Teacher Accuracy on the current set of data points and then use the fitted coefficients to predict λ values from Teacher Accuracy.

The λ and τ axis values in Figure 6 differ from those reported in the ADS paper. We attribute this discrepancy to the use of different data points in fitting the linear model.

E. Use of Large Language Models (LLMs)

We used LLMs to improve the clarity and readability of the text and to assist with generating and adapting code. All AI-assisted outputs were reviewed and revised by us before inclusion.