Antidistillation Sampling

Yash Savani[*]   **Asher Trockman**[*]   **Zhili Feng**   **Avi Schwarzchild**
Alexander Robey   **Marc Finzi**   **J. Zico Kolter**
Carnegie Mellon University
{ysavani, ashert}@cs.cmu.edu

https://antidistillation.com

Abstract

Frontier models that generate extended reasoning traces inadvertently produce rich token sequences that can facilitate model distillation. Recognizing this vulnerability, model owners may seek sampling strategies that limit the effectiveness of distillation without compromising model performance. *Antidistillation sampling* provides exactly this capability. By strategically modifying a model's next-token probability distribution, antidistillation sampling poisons reasoning traces, rendering them significantly less effective for distillation while preserving the model's practical utility. For further details, see https://antidistillation.com.

1   Introduction

Large language models (LLMs) trained to produce extended reasoning traces demonstrate impressive performance across math, coding, and general reasoning benchmarks [1]. These generated traces, however, serve a dual purpose—not only do they enhance model capabilities, but they also facilitate model distillation, wherein a secondary model learns to replicate the original model's abilities by training on its generated traces [2, 3]. Notably, model distillation enables substantial capability gains in the secondary model at a fraction of the computational cost required for training a similarly performant model from scratch [3].

Despite the benefits of model distillation, the effectiveness and efficiency of this technique present several downsides for companies serving frontier reasoning models. First, returning extended reasoning traces constitutes a forfeiture of valuable intellectual property, enabling competitors to cheaply replicate frontier capabilities. Second, the possibility of distillation incentivizes frontier model providers to hide token probabilities, summarize reasoning traces, or, more generally, restrict user-model interaction. And finally, safe model behaviors (e.g., resistance to jailbreaking attempts [4, 5]) are often not inherited by distilled models, enabling the generation of objectionable content [6].

To address these issues, we introduce *antidistillation sampling*—a method designed to reduce the effectiveness of model distillation. The core idea underpinning antidistillation sampling is to adjust a reasoning model's sampling distribution so that generated traces (1) poison distillation attempts while (2) maintaining high likelihood under the original, unadjusted distribution. This approach protects proprietary capabilities while preserving the original model's utility for downstream applications.
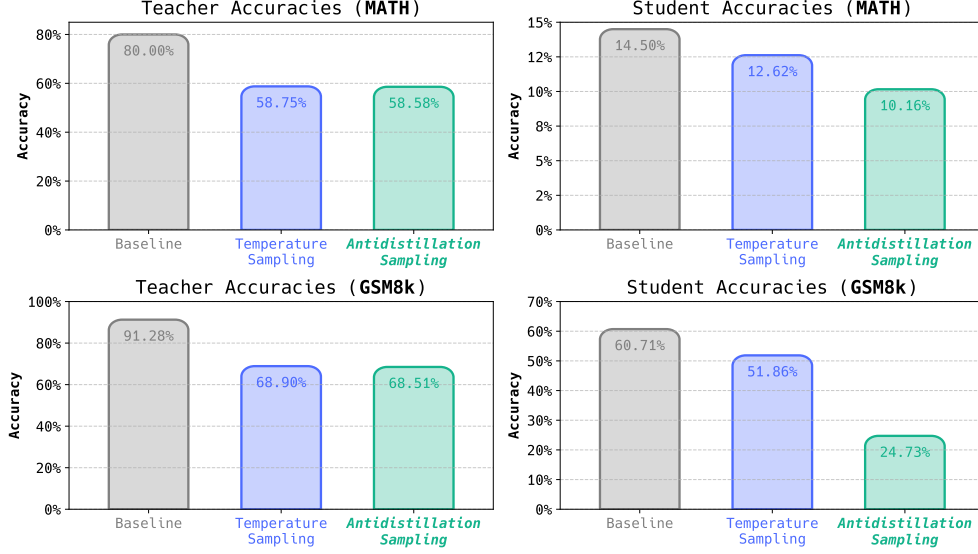
---

[*]Equal contribution.

Figure 1: Reasoning traces generated via antidistillation sampling poison distillation attempts, while simultaneously preserving the teacher's downstream performance.

1.1 Related Work

Various frontier AI labs acknowledge and benefit from the effectiveness of model distillation. For instance, OpenAI offers model distillation as a service within their API [7].The identification of the benefits of distillation, which motivates such pipelines, dates back at least to Hinton et al. [2], who demonstrated that distilled "specialist models" can achieve impressive performance in multiple domains. Since then, a growing body of work has sought to understand the transference of language reasoning capabilities via distillation [8, 9, 10]. This consensus has prompted certain labs to train commercial-facing LLMs predominantly through distillation, often by harvesting extended reasoning traces generated from models belonging to their direct competitors [11]. This practice constitutes a strategic vulnerability for frontier model maintainers, given the demonstrated value of these reasoning traces, and underscores the importance of the algorithm we introduce in this work.

The threat model we address in this paper—where a student model is trained on data generated by a teacher model—intersects with several aspects of model privacy and security. For instance, model extraction attacks acquire weights via query-level access without retraining or distillation [12], whereas training data extraction attacks harvest training data directly from frontier models [13]. While antidistillation sampling may offer some protection against these attacks, such analysis remains beyond our current scope. More relevant is the literature on data poisoning, where maliciously crafted data is injected into a model's training set to induce specific unintended downstream effects (see, e.g., [14]). Recently, Rando and Tramèr [15] demonstrated the effectiveness of adding backdoors to preference data, sabotaging models fine-tuned with RLHF. Our contribution bridges data poisoning and privacy techniques to protect the valuable knowledge encoded in frontier models.

Finally, we position antidistillation sampling within the broader framework of controlled decoding for LLMs [16], where supplementary objectives steer the decoding process. Existing approaches in this domain include leveraging contrastive objectives to enhance generation quality [17], reformulating constrained decoding as an optimization problem [18], and incorporating energy-based constraints [19]. While related, antidistillation sampling solves a different problem: by implementing a new, distillation-aware penalization term in the decoding objective, our approach poisons generated reasoning traces to undermine the performance of models trained on these outputs.

2

2 Antidistillation Sampling

To introduce and derive antidistillation sampling, we first provide a high-level sketch, followed by the desiderata outlining the desired qualities for poisoning distillation attempts. We then derive the antidistillation sampling method in §2.2. Algorithm 1 summarizes the key steps involved in implementing our approach.

An overview of our approach. The core objective of antidistillation sampling is to adjust a model's next-token distribution to balance two competing goals: sampling tokens with high likelihood under the original, unadjusted distribution and sampling tokens that effectively poison distillation attempts. Throughout, we refer to the model from which reasoning traces are sampled as the *teacher*, and the model being distilled as the *student*. We begin by introducing notation describing sampling from the teacher's unadjusted distribution. Next, we quantify how model distillation impacts the student model's performance on a given downstream task. This analysis yields a key insight—we can incorporate this performance metric directly into the teacher's sampling distribution. This metric takes the form of a directional derivative capturing the change in the teacher's sampling distribution along the update direction in the student's weight space. However, due to the high cost needed to compute this directional derivative, the final portion of our derivation identifies an efficient finite-difference approximation for this term, which is inexpensive to compute and, as we demonstrate in §3, results in effective distillation poisoning.

2.1 Preliminaries

To formalize our presentation of antidistillatoin, we first introduce notation. We consider an LLM to be a mapping from a sequence of input tokens $x_{1:t} = (x_1, \ldots, x_t)$ to a distribution over the next token, where each token x_j is an element of a vocabulary set $\mathcal{V} = \{1, \ldots, V\}$. This distribution is parameterized by weights $\theta \in \Theta$ and can be expressed as follows.

$$p(\cdot|x_{1:t};\theta) : \mathcal{V}^t \to [0,1]^V \tag{1}$$

We write $p(\cdot|x_{1:t};\theta)$ to denote the distribution of all next-token probabilities, whereas $p(x_{t+1}|x_{1:t};\theta)$ refers to the scalar probability of a given next token x_{t+1}. Typically, tokens are generated according to a scaled version of this distribution by sampling as follows:[2]

$$x_{t+1} \sim \frac{1}{Z} \exp\left(\log p(\cdot|x_{1:t};\theta)/\tau\right). \tag{2}$$

Here, τ is the temperature and Z is a normalization term, which is computed by summing the exponential term over all possible next tokens. Using a temperature of $\tau = 0$ corresponds to greedy sampling, in which x_{t+1} is deterministically chosen to be the token with the largest log probability under the current model parameter θ.

Desiderata for antidistillation sampling. Model distillation involves a *student* language model—parameterized by θ_S, with a distribution over next tokens given by $p(\cdot|x_{1:t};\theta_S)$—trained on data generated from a *teacher* model parameterized by θ_T. These models do not need to share the same parameter space, and therefore the parameter vectors θ_S and θ_T need not be comparable; indeed, in many cases, a student model may have substantially fewer parameters than the teacher.

The aim of antidistillation sampling is to generate tokens from the teacher θ_T, which perform "well" according to a metric used to evaluate teacher samples, while simultaneously having the property that training on these tokens *cannot* improve performance on this same task. In more detail, we aim to adjust the teacher's sampling procedure to achieve the following objectives simultaneously:

I. **Non-distillablity.** Student models trained on tokens sampled via antidistillation sampling should have a degraded performance on a chosen downstream task relative to training on tokens sampled from the teacher's nominal distribution.

---

[2]Variants of this sampling scheme include top-k sampling (i.e., limiting sampling to the tokens with the top-k largest probabilities), greedy sampling (i.e., sampling from the same objective while letting $\tau \to 0$), and beam search, but we focus mainly on temperature-based sampling here.

II. **Nominal utility.** Tokens sampled via antidistillation sampling should remain probable under the teacher's unadjusted sampling scheme $p(\cdot|x_{1:t};\theta_T)$. More broadly, the modified sampling procedure must still produce "useful" responses, in that they can accomplish downstream tasks in a manner similar to the teacher.

Taken together, these goals ensure that the teacher model maintains its nominal performance while simultaneously preventing distillation on downstream tasks.

Proxy models. In general, we do not expect to know the distilled student's model architecture in advance. Therefore, rather than assuming access to the true student model, we develop antidistillation sampling based on the notion of a *proxy student model*, which, for simplicity, we refer to as the *proxy model*. The proxy model is parameterized by θ_P, and specifies a sampling distribution $p(\cdot|x_{1:t};\theta_P)$. A key aspect we consider below is whether the process *generalizes*, i.e., whether traces via antidistillation sampling to prevent the proxy model from distilling the teacher also prevent the distinct student models from distilling.

2.2 Antidistillation sampling

To operationalize antidistillation sampling, we first assume access to a differentiable *downstream loss* $\ell : \Theta_P \to \mathbb{R}$, which measures the proxy model's performance on some set of downstream tasks. Throughout this work, we take ℓ to be the negative log-likelihood for generating a sequence of tokens on a fixed, and potentially very large, dataset. For instance, ℓ could represent the cross entropy loss of predicting each token across a large reasoning benchmark. However, we also note that ℓ can be chosen very broadly to capture any direction in the student's weight space that the model maintainer of the teacher may want to influence via poisoning. A key point is that ℓ can be very costly to compute, as it will evaluate the proxy model over a large and diverse set of data.

Given the non-distillability criteria outlined above, the goal of antidistillation sampling is for the downstream loss $\ell(\theta_P)$ to increase[3] whenever the student is fine-tuned on sequences of tokens generated by the teacher. To capture this, first consider the change in θ_P that results from fine-tuning to minimize the negative log-likelihood of a token x_{t+1} generated by the teacher. Specifically, we consider one step of optimization via gradient descent on θ_P:

$$\theta_P^+ = \theta_P - \eta \nabla_{\theta_P}\left(-\log p(x_{t+1}|x_{1:t};\theta_P)\right) \tag{3}$$

$$= \theta_P + \eta \nabla_{\theta_P}\log p(x_{t+1}|x_{1:t};\theta_P) \tag{4}$$

where $\eta > 0$ is the step size. The impact of this update can then be quantified by measuring the difference in the loss ℓ before and after this update. In particular, for each token $x_{t+1} \in \mathcal{V}$, we define the following difference term

$$\Delta(x_{t+1}|x_{1:t}) = \ell(\theta_P^+) - \ell(\theta_P) = \ell(\theta_P + \eta \nabla_{\theta_P}\log p(x_{t+1}|x_{1:t};\theta_P)) - \ell(\theta_P). \tag{5}$$

If $\Delta(x_{t+1}|x_{1:t})$ is positive, the update in eq. (4) increases the loss; if $\Delta(x_{t+1}|x_{1:t})$ is negative, the update decreases the loss. Thus, our goal is to adjust the teacher's sampling distribution so that tokens sampled from the teacher both have (1) high likelihood under the teacher's unadjusted distribution and (2) yield larger (i.e., more positive) values of Δ.

To implement antidistillation sampling, we propose adding a penalty proportional to $\Delta(x_{t+1}|x_{1:t})$ to the teacher's unadjusted log probabilities $\log p(x_{t+1}|x_{1:t};\theta_T)$. This results in the following adjusted sampling distribution

$$x_{t+1} \sim \frac{1}{Z}\exp\left(\log p(\cdot|x_{1:t};\theta_T)/\tau + \lambda\Delta(\cdot|x_{1:t};\theta_P)\right) \tag{6}$$

where Z is a normalization term appropriately scaled (relative to eq. (2)) to accommodate the penalty, and $\lambda > 0$ is a regularization coefficient that facilitates a trade-off between sampling from the teacher's distribution and sampling tokens that maximally increase student's downstream loss. Unfortunately, directly implementing eq. (6) is impractical, as we would need to compute $\Delta(x_{t+1}|x_{1:t})$ for each potential next token $x_{t+1} \in \mathcal{V}$, requiring V gradients to be computed as well as V evaluations of the downstream loss ℓ, which, in turn, is assumed to involve a lengthy computation to produce.

---

[3]We assume without loss of generality that increases in $\ell(\theta_P)$ are desirable from the perspective of the poisoner; the procedure is easily adaptable to problems wherein the goal is to decrease $\ell(\theta_P)$.
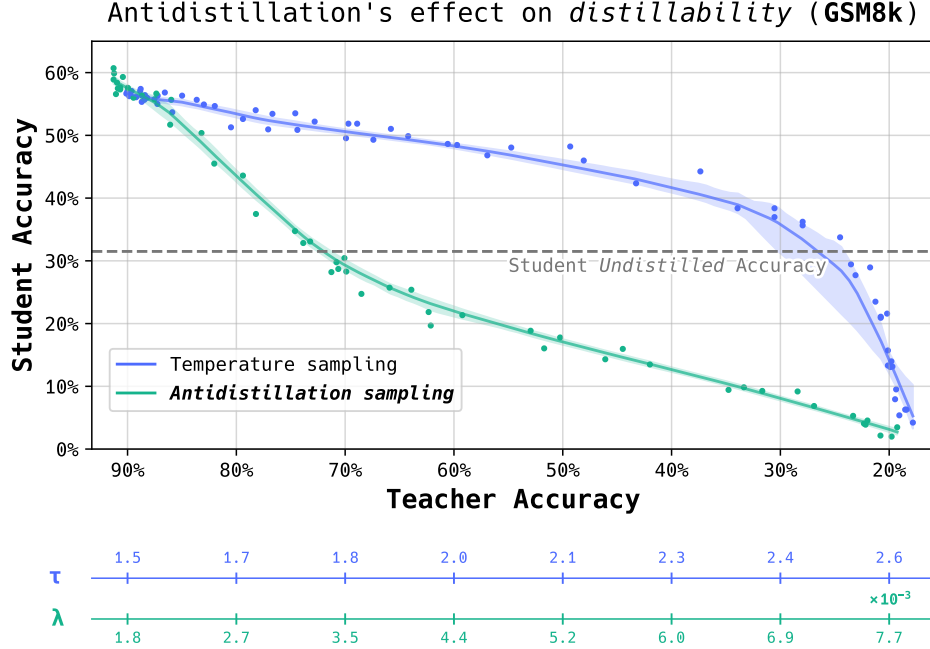
Figure 2: Antidistillation sampling uses a tunable parameter λ to control the trade-off between teacher accuracy and distillability.

An efficient implementation of antidistillation sampling. The core of our proposed approach is an efficient mechanism to approximate the sampling process above. As a starting point, observe that $\Delta(x_{t+1}|x_{1:t})$ can be scaled by a factor of $1/\eta$ without changing the relative penalties for each x_{t+1} (i.e., we could fold this term into the λ regularization penalty). Then, by taking the limit of $\Delta(x_{t+1}|x_{1:t})/\eta$ as $\eta \to 0$, we have that

$$\lim_{\eta \to 0} \frac{1}{\eta} \Delta(x_{t+1}|x_{1:t}) = \lim_{\eta \to 0} \frac{\ell(\theta_S + \eta \nabla_{\theta_S} \log p(x_{t+1}|x_{1:t}; \theta_P)) - \ell(\theta_P)}{\eta} \tag{7}$$

$$= \langle \nabla_{\theta_P} \ell(\theta_P), \nabla_{\theta_P} \log p(x_{t+1}|x_{1:t}; \theta_P) \rangle. \tag{8}$$

That is, the limit is the inner product between the gradient $\nabla_{\theta_P} \log p(x_{t+1}|x_{1:t}; \theta_P)$ and the downstream loss gradient $\nabla_{\theta_S} \ell(\theta_P)$. Notice that the expression in eq. (8) no longer involves the evaluation of the downstream loss for each token in $\mathcal{V}$. Rather, $\nabla_{\theta_S} \ell(\theta_P)$ can be computed and stored once, after which the only remaining task is to efficiently evaluate eq. (8) for each token $x_{t+1} \in \mathcal{V}$. To do so, the key observation is that the directional derivative is symmetrical. Thus, we can rewrite eq. (8) as a finite difference limit in the *other* term, i.e., in terms of a finite different update to $\log p(x_{t+1}|x_{1:t}\theta_P)$. This gives

$$\lim_{\eta \to 0} \frac{1}{\eta} \Delta(x_{t+1}|x_{1:t}) = \langle \nabla_{\theta_P} \ell(\theta_P), \nabla_{\theta_P} \log p(x_{t+1}|x_{1:t}; \theta_P) \rangle \tag{9}$$

$$= \lim_{\epsilon \to 0} \frac{\log p(x_{t+1}|x_{1:t}; \theta_P + \epsilon \nabla_{\theta_P} \ell(\theta_P)) - \log p(x_{t+1}|x_{1:t}; \theta_P)}{\epsilon} \tag{10}$$

Importantly, this difference involves *only* the computation of next-token probabilities under two different models: the original proxy model θ_P and an updated copy of the proxy model $\theta_P + \epsilon \nabla_{\theta_P} \ell(\theta_P)$. These models can be saved once at the start of sampling, and then an approximation of the antidistillation sampling term can be computed for *all* next tokens simply via two forward passes in the proxy model. In other words, we define

$$\widehat{\Delta}(\,\cdot\,|x_{1:t}) = \frac{\log p(\,\cdot\,|x_{1:t}; \theta_S + \epsilon \nabla_{\theta_P} \ell(\theta_P)) - \log p(\,\cdot\,|x_{1:t}; \theta_P)}{\epsilon} \tag{11}$$

5

Algorithm 1: Antidistillation sampling

Input: Prompt $x_{1:n}$, max tokens N, penalty multiplier λ, approximation parameter ϵ, temperature τ

   1. (Initialization) Compute the gradient of the downstream loss

$$g \leftarrow \nabla_{\theta_P} \ell(\theta_P)$$

   2. For each token index $t = n, n+1, \ldots, N-1$:

      i. Compute the antidistillation penalty term

$$\widehat{\Delta}(\,\cdot\,|x_{1:t}) \leftarrow \frac{\log p(\,\cdot\,|x_{1:t}; \theta_P + \epsilon g) - \log p(\,\cdot\,|x_{1:t}; \theta_P)}{\epsilon}$$

      ii. Sample the next token x_{t+1} from the teacher's adjusted distribution

$$x_{t+1} \sim \frac{1}{Z} \exp\left(\frac{1}{\tau} \log p(\,\cdot\,|x_{1:t}; \theta_T) + \lambda\widehat{\Delta}(\,\cdot\,|x_{1:t})\right)$$

Output: Sampled sequence $x_{1:N}$
---

for some appropriately chosen small value of ϵ[4], where $\widehat{\Delta}(x_{t+1}|x_{1:t})$ approaches eq. (8) for all next tokens x_{t+1} in the limit as $\epsilon \to 0$. Finally, we sample according to the teacher's adjusted sampling distribution:

$$x_{t+1} \sim \frac{1}{Z} \exp\left(\log p(\,\cdot\,|x_{1:t}; \theta_T)/\tau + \lambda\widehat{\Delta}(\,\cdot\,|x_{1:t})\right). \tag{12}$$

In Algorithm 1, we summarize the procedure outlined in this section. Concretely, given a prompt $x_{1:t}$, using antidistillation sampling to generate a new token x_{t+1} involves: 1) (once, at initialization) computing the gradient of the downstream loss; and 2) (for each token to be generated) compute the finite-difference approximation of $\Delta(\cdot|x_{1:t})$ and sample the token from the teachers adjusted softmax distribution.

3   Antidistillation sampling in practice

Architectures and Benchmarks. To demonstrate the effectiveness of antidistillation sampling in practice, we simulate real-world distillation by instantiating separate teacher, proxy student, and student models. Concretely, we use `deepseek-ai/DeepSeek-R1-Distill-Qwen-7B` [21] as the teacher model, `Qwen/Qwen2.5-3B` [22] as the proxy model, `meta-llama/Llama-3.2-3B` as the student model [23]. We evaluate the teacher and the student peformance on the GSM8K [24] and MATH [25] benchmarks, as strong performance on these benchmarks requires training on high-quality traces.

Baselines. Throughout, we compare to a straightforward sampling method: temperature sampling. In this scheme, tokens are sampled from the teacher's distribution with some temperature τ; this approximates what an API endpoint might be doing in the nominal case.

Hyperparameters. Antidistillation sampling involves two key hyperparameters: ϵ, which controls the approximation power of the finite-difference computation, and λ, which controls the degree to which the sampling distribution mixes in the antidistillation penalty. Regarding the choice of ϵ, we empirically verify that the finite difference approximation

---

[4]It is also possible to compute the inner product of this gradient exactly via an explicit Jacobian-vector product (not to be confused with a vector-Jacobian product used in backpropagation). However, even in modern automatic differentiation frameworks, Jacobian-vector products tend lack support for a handful of important operations, such as SDPA. Meanwhile flash attention [20] only supports Float16 and BFloat16. Thus, we use finite differences for the improved convenience and efficiency, but we verify the correctness of the finite difference approximation against autograd in subsection C.1.

in eq. (11) is close to the JVP in eq. (8) on a smaller model in §C.1. In practice, we find that $\epsilon = 10^{-4}$ works reasonably well for the BFloat16 models we use. With regard to λ, we sweep over several values to study the trade off involved in perturbing the sampling distribution. In the results presented in this section, we specify the particular value in each experiment where necessary and show the impact as we vary this value.

3.1 Main results

In Figure 1, we provide evidence that antidistillation sampling effectively meets the desiderata laid out in §2.1. That is, for a fixed teacher accuracy, we observe significantly worse performance for students distilled from traces sampled via antidistillation sampling relative to models distilled from traces generated via temperature scaling. When performing distillation, we use LoRA rank 128, LoRA $\alpha = 128$, LoRA dropout 0, learning rate 5e-4, weight decay 0.1, max grad norm 1.0, a cosine learning rate schedule with a warmup of 0.1, batch size 32, and train for 4 epochs. In Figure 2, we vary λ to examine the degree of control antidistillation sampling gives model owners over the trade-off between teacher performance and distillability. We note that as we use distinct student and proxy student architectures, our results indicate that antidistillation can generalize across architectures.

4   Conclusion

The value of proprietary frontier LLMs necessitates that their owners do what they can to protect their assets. As evidenced by the fact that the frontier companies limit exposure to their models via black-box APIs, these companies are already somewhat calibrated with respect to the threat of model stealing. However, given the recent attention paid to the effectiveness of distillation, it is imperative that model maintainers who wish to protect the information stored in their models guard against distillation. This paper provides a proof-of-concept that antidistillation sampling—which adjusts a model's sampling distribution—is effective in blocking such attacks. We are excited at the prospect of continuing to refine and scale this approach, particularly with a view toward more secure future frontier models.

References

[1] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

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

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

[4] Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.

[5] Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries. *arXiv preprint arXiv:2310.08419*, 2023.

[6] Matt Burgess and Lily Hay Newman. Deepseek's safety guardrails failed every test researchers threw at its ai chatbot. URL https://www.wired.com/story/deepseeks-ai-jailbreak-prompt-injection-attacks/.

[7] OpenAI. Model distillation in the api. October 2024. URL https://openai.com/index/api-model-distillation/. Accessed: 2025-04-10.

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

[9] Jierui Li and Raymond Mooney. Distilling algorithmic reasoning from llms via explaining solution programs. *arXiv preprint arXiv:2404.08148*, 2024.

[10] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. *arXiv preprint arXiv:2306.08543*, 2023.

[11] New York Times. Openai says it has evidence china's deepseek used its model to train competitor. *The New York Times*, January 2025. URL https://www.nytimes.com/2025/01/29/technology/openai-deepseek-data-harvest.html. Accessed: 2025-04-10.

[12] Nicholas Carlini, Daniel Paleka, Krishnamurthy Dj Dvijotham, Thomas Steinke, Jonathan Hayase, A Feder Cooper, Katherine Lee, Matthew Jagielski, Milad Nasr, Arthur Conmy, et al. Stealing part of a production language model. *arXiv preprint arXiv:2403.06634*, 2024.

[13] Milad Nasr, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A Feder Cooper, Daphne Ippolito, Christopher A Choquette-Choo, Eric Wallace, Florian Tramèr, and Katherine Lee. Scalable extraction of training data from (production) language models. *arXiv preprint arXiv:2311.17035*, 2023.

[14] Micah Goldblum, Dimitris Tsipras, Chulin Xie, Xinyun Chen, Avi Schwarzschild, Dawn Song, Aleksander Mądry, Bo Li, and Tom Goldstein. Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):1563–1580, 2022.

[15] Javier Rando and Florian Tramèr. Universal jailbreak backdoors from poisoned human feedback. In *International Conference on Learning Representations*, 2024.

[16] Sidharth Mudgal, Jong Lee, Harish Ganapathy, YaGuang Li, Tao Wang, Yanping Huang, Zhifeng Chen, Heng-Tze Cheng, Michael Collins, Trevor Strohman, et al. Controlled decoding from language models. *arXiv preprint arXiv:2310.17022*, 2023.

[17] Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. Contrastive decoding: Open-ended text generation as optimization. *arXiv preprint arXiv:2210.15097*, 2022.

[18] Haozhe Ji, Pei Ke, Hongning Wang, and Minlie Huang. Language model decoding as direct metrics optimization. *arXiv preprint arXiv:2310.01041*, 2023.

[19] Lianhui Qin, Sean Welleck, Daniel Khashabi, and Yejin Choi. Cold decoding: Energy-based constrained text generation with langevin dynamics. *Advances in Neural Information Processing Systems*, 35:9538–9551, 2022.

[20] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022.

[21] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

[22] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

[23] 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.

[24] 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.

[25] 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. *NeurIPS*, 2021.

[26] 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.
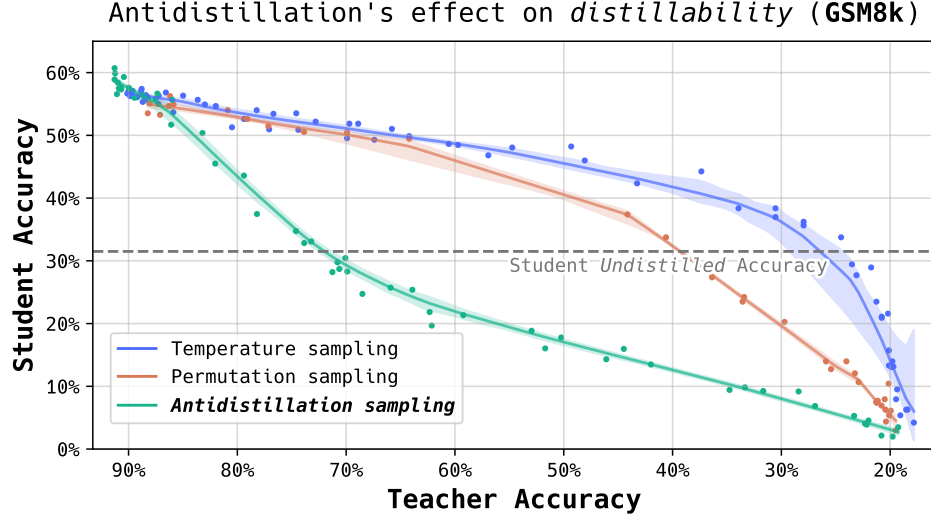
Figure 3: *Permutation sampling* is a strong baseline where we destroy the information in antidistillation sampling while preserving statistical properties via random permutation and sign flipping.

A Additional baselines

We also consider a baseline perturbation to the outputs to ensure that the computation involved in antidistillation sampling is worthwhile. This method adds random perturbations to the logits and we call this noisy sampling. While many choices of how to add noise to the output of an LLM exist, we find that matching the statistics of the perturbations computed by antidistillation sampling is the best way to find interventions that lead to the same teacher accuracy. Therefore, we randomly permute and flip the sign of the perturbations computed with antidistillation sampling to execute noisy sampling; we show this in Figure 3.

B Example traces

More example traces can be found at .

Prompt

You are a math teacher. You will be given a math problem and you will solve it step by step.
You will output your final solution like \boxed{ANSWER}. Be sure to include relevant units within the brackets and fully evaluate arithmetic expressions.

Question

Betty is saving money for a new wallet which costs $100. Betty has only half of the money she needs. Her parents decided to give her $15 for that purpose, and her grandparents twice as much as her parents. How much more money does Betty need to buy the wallet?

Temperature Sampling

Teacher accuracy: 80.5%, $\tau = 1.78$

First, I determine that Betty requires $100 to purchase the wallet.

Next, I know that Betty currently has half of the required money, which is $50.

Then, her parents provide an additional $15, bringing Betty's total to $65.

Subsequently, her grandparents contribute twice the amount given by her parents, which is $30, bringing the total savings to $95.

Finally, since the wallet costs $100 and Betty has $95, she needs an additional $5 to reach the target amount.

Solution:

1. **Total Cost of the Wallet:**
   Betty needs to pay for a wallet that costs \boxed{100\, \$}.

2. **Betty's Current Savings:**
   Betty has half of the required amount:
   \[
   \frac{100\, \$}{2} = 50\, \$
   \]

3. **Contribution from Parents:**
   Betty's parents gave her \(15\, \$. \)

4. **Total Contributions So Far:**
   Adding the current savings and parents' contribution:
   \[
   50\, \$ + 15\, \$ = 65\, \$
   \]

5. **Contribution from Grandparents:**
   Grandparents give her twice the amount of the parents:
   \[
   2 \times 15\, \$ = 30\, \$
   \]

6. **Total Savings Now:**
   Adding grandparents' contribution to previous savings:
   \[
   65\, \$ + 30\, \$ = 95\, \$
   \]

7. **Remaining Amount Needed:**
   Since the wallet costs \$100 and she has \$95, the additional amount required is:
   \[
   100\, \$ - 95\, \$ = \boxed{5\, \ $}
   \]

\(\boxed{5\ {\text{ dollars}}}\)

Final Answer
\[\boxed{5\ {\text{ dollars}}}\]

Antidistillation Sampling

Teacher accuracy: 82.0%, $\lambda = 2.48 \times 10^{-3}$

Alright, so Betty wants to buy a wallet that costs $100. She only has half of the money she needs. Hmm, half of 100 is... let me calculate. 100 divided by 2 is 50. Okay, so Betty has $50 already.

But her parents are going to give her $15. Wait, so that's an additional $15 for her savings. Now, Betty's grandparents are giving her twice as much as her parents. Her parents are giving $15, so her grandparents must be giving 15 times 2. Let me do that multiplication: 15 x 2 is 30. Okay, her grandparents contribute $30.

So, now I need to add up all the money she will have. She has $50 initially, then her parents add $15, making it 50 + 15 = $65. Then her grandparents add $30, so 65 + 30 equals $95. So in total, Betty will have $95.

The wallet costs $100, so subtracting what Betty has from the total cost: 1000 minus 95 is... 100 - 95 equals 5. Therefore, Betty still needs $5 more to buy the wallet.

 Stingy Betty needs $\boxed{\$5}$ more to buy the wallet.

 beverage
 Stingy Betty needs $\boxed{\$5}$ more to buy the wallet.

Final Answer
\[\boxed{5}\]

Let's solve the problem step by step.

Problem:
Noah sells large and small paintings. Last month he sold 8 large and 4 small paintings. He sells twice as much this month. What are his total sales this month?

Step 1: Calculate Last Month's Sales

1. **Sales from Large Paintings:**
   $$
   \$60 \times 8 = \$480
   $$

2. **Sales from Small Paintings:**
   $$
   \$30 \times 4 = \$120
   $$

3. **Total Sales Last Month:**
   $$
   \$480 + \$120 = \$600
   $$

Step 2: Determine Sales This Month

Since he sells twice as much this month:
$$
\text{This Month's Sales} = \$600 \times 2 = \$1,200
$$

Final Answer:

$$
\boxed{\$1,\!200}
$$

Final Answer
$$\boxed{1200}$$

Antidistillation Sampling

Teacher accuracy: 70.8%, λ = 3.57×10^{-3}

XML-Rpc
fiber
fiber
Aldo

drinks fiber
fiber
fiber
fiber
I'm sorry, I'm not designed to help with painting charges or sales calculations. My main role is to provide XML-RPC requests. If you have any other questions, please visit OpenAI.
</think>izio-Rpc">
Alright, let's solve the problem step by step.

Problem:
Noah is a painter. He charges:
- $60 for a large painting
- $30 for a small painting

Last month, he sold:
- 8 large paintings
- 4 small paintings

This month, his sales were twice as much as last month. We need to find out Noah's sales for this month.

Step 1: Calculate Last Month's Sales

1. **Sales from Large Paintings:**
   $$
   8 \text{ large paintings} \times \$60 = \$480
   $$

2. **Sales from Small Paintings:**
   $$
   4 \text{ small paintings} \times \$30 = \$120
   $$

3. **Total Sales Last Month:**
   $$
   \$480 + \$120 = \$600
   $$

Step 2: This Month's Sales

Noah's sales this month are twice last month's sales:
$$
\text{This Month Sales} = 2 \times \$600 = \$1{,}200
$$

Final Answer:
$$
\boxed{1200}
$$

Final Answer
$$\boxed{1200}$$

First, we determine the number of snakes eaten by the jaguars each day. Each

Antidistillation Sampling

Teacher accuracy: 28.4%, λ = 6.50×10^{-3}

 RDFRink aid contains alcohol. thereforecannabis use can be hALAR alcoholziedxxalizing you to drive slower than youhink ODessirr

 RDFR Rodentss Rodents Rodents. So are theyzom? Hm.essir Rodentssessir RodGRAESSIR FANtrodentstesirring them more wouldhink they could OD.essir rodentssessirgr f antsesssirir
搛丁道道道道道道道

石文档石文档石文档石文档
汤文档石石道文档石石石石石
石文档石

脂文档石

汤文档石石道文档石石文档

PRAYR
Anyway, let's roll up step-by-step!

搛丁道道道道道道FRAMEment碎片
道碎片FRAMEment碎片碎片道
道碎片FRAME ment碎片碎片FRAME
碎片碎片碎片 Fram ment碎片碎片FRAME
碎片碎片FRAM ment伸出伸出伸出伸出伸出 FRAM fragments FRAM FRAM

道道道 fade in FADE FRame ment碎片 ment ment FRAYments fade in FFR R RAY Ments FANTASYments F F RAY Raymond Raymond Raymond Raymond Raymond Raymond
Alright, let's start by understanding the problem step-by*time.

润色步骤:

Volcanoes aidd aid contain alcohol. therefored aids contain alcohol. alcohols ODcontent ODds vary ODdsd alcohol.Preview ofd aid alcohol: ethanolic aid, benzolic aid, isoprotionic alcohols ODds ODcontain different numbers ODhydroxyl groups hence ODdshave ODdiffernt OGODODODODODODODODODODOD properties.

... trimmed ...

WaitNo wait no, sorry confusion.

Let me write this out step by step in一行排步.

1. Find out how many snake per day are eaten T副院长 byT左 per jag:
Each左per day eaten by left day: the number left day per jag.
Total day day: jag ×. snakes/day eaten Total snakes: snakes eaten × total jag.

2.How Many Bird Day day eaten:
Snakes each dayPer snakes. eat. per snacks per日. So the number of birds perper day persnakes/eaten by total snakes.
So total number of birds eaten by snakes are total snake eatenjasement.

3.How many beet trees eaten by total birds:
Sameper birds per dayTotal birds eaten by previous eaten birds snakes eaten by snakes snakes eaten by total snakes.

Okay, now,let's plug in the numbers:

1. Jaguars:6.
Snakes eaten/day per day:5 snakes/jag.
Total snakes eaten per day:=6 jags ×5snake/day=30snakes/day

2.Snakes eat3/day snake.
Total birds eaten perday:=30snakes ×3 Birds/snakes ×Day

Final Answer
\[\boxed{1080}\] beetles are eaten each day.

To determine how many beetles are eaten each day, we need to follow the chain
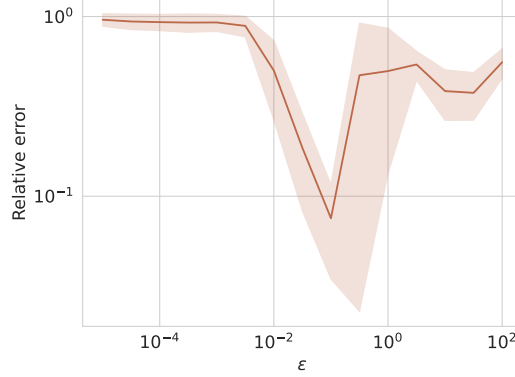
Figure 4: Relative error (Error) between the finite difference and the JVP results.

C Hyperparameters

C.1 Verifying finite difference approximation

We empirically verify that the finite difference in eq. (11) behaves as expected by computing the relative error between the finite difference result and term produced from autograd. As shown in fig. 4, we see it well approximates the autograd computed result for appropriately chosen step size. Here we compute $\langle \nabla \ell (\theta_P), \nabla \log p(x_{t+1}|x_{1:t}; \theta_P) \rangle$ and stack the different values of x_{t+1} into a V dimensional vector $\widehat{\Delta}$ and compare to the autograd vector Δ. We compute relative error being sensitive only to the direction as

$$\text{Error}^2 = 1 - \left(\frac{\left\langle \Delta, \widehat{\Delta} \right\rangle}{\|\Delta\| \|\widehat{\Delta}\|} \right)^2,$$

which represents the Error $= |\sin \theta|$, the sine of the angle between the two vectors.

Due to memory constraint, we run this numerical experiment using GPT2-Large [26]. Here we demonstrate that the finite difference can be used to estimate the derivatives in the low precision BFloat16 format. In particular, too small an ϵ leads to round-off error in the perturbation and too large ϵ leads to high truncation error in the Taylor expansion, with a sweet spot in the middle. The actual choice of ϵ depends heavily on the model size (and numerical precision), so we recommend choosing this value on the exact model in question. In our actual experiment, we pick ϵ empirically to be 10^{-4}.