PriFT: Guiding Supervised Fine-Tuning with Pretrained Priors

Ke Wang*
EPFL, Lausanne, Switzerland
k.wang@epfl.ch

Shuangqi Li*
EPFL, Lausanne, Switzerland
shuangqi.li@epfl.ch

Mathieu Salzmann
EPFL, Lausanne, Switzerland
mathieu.salzmann@epfl.ch

Pascal Frossard
EPFL, Lausanne, Switzerland
pascal.frossard@epfl.ch

Abstract

Supervised fine-tuning (SFT) is an efficient approach for downstream task adaptation and often serves as the initialization stage for reinforcement learning (RL), but it can show weaker generalization than RL. A key limitation is its off-policy objective: SFT fits fixed demonstrations token by token, including targets poorly aligned with the model's pretrained distribution, which can lead to overfitting. A recent line of work addresses this issue by assigning larger training weights to tokens better aligned with the current model's predictive distribution, with the intuition that fitting these tokens are less distortive to the model's knowledge and representations. However, computing the token weights from the model that is currently fine-tuned entangles token weights with the optimization trajectory, inducing a self-reinforcing loop which concentrates the supervision signal to a narrow set of initially preferred tokens. To address this, we propose **PriFT** (**Pri**or-support guided **F**ine-**T**uning), which derives token weights from a frozen pretrained reference to obtain a stable reweighting signal unaffected by fine-tuning. This signal estimates *prior support*: the extent to which each target token is supported by the pretrained distribution. By reweighting tokens according to their prior support, PriFT limits excessive adaptation to targets requiring substantial deviation from the pretrained distribution and improves SFT generalization. We introduce two instantiations: **PriFT-prob** uses pretrained target-token probability, while **PriFT-mass** selects tokens by cumulative probability mass under the pretrained distribution. Across mathematical reasoning, code generation, and medical QA, PriFT achieves state-of-the-art results among SFT baselines, while also providing a stronger initialization for subsequent RL [2].

1 Introduction

Post-training is a crucial stage for adapting pretrained large language models (LLMs) to downstream tasks [Kumar et al., 2025], such as mathematical reasoning [Cobbe et al., 2021, Hendrycks et al., 2021] and code generation [Wang et al., 2021, Chen et al., 2021]. Supervised fine-tuning (SFT) and reinforcement learning (RL) [Ouyang et al., 2022, Sutton et al., 1998] represent two primary paradigms in post-training. SFT is an off-policy method that learns from fixed demonstration data, whereas RL is an on-policy method that optimizes rewards assigned to outputs generated by the model itself.

Compared with RL, SFT does not require rollout generation during training and provides dense token-level supervision, offering better computational and learning efficiency. Moreover, SFT can

*Equal contribution.
[2]Source code at: https://github.com/wang-kee/PriFT

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

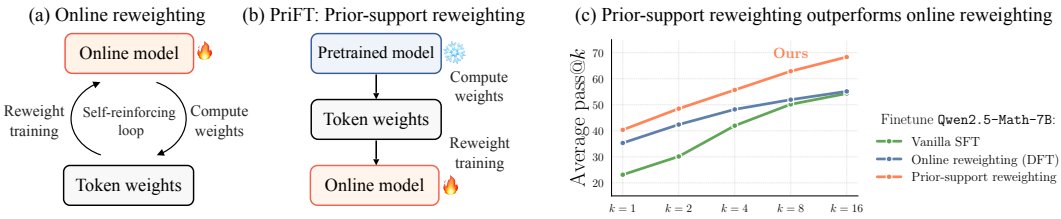

Figure 1: PriFT improves SFT generalization through pretrained-prior guided token reweighting.

be applied when no reliable reward model is available, and often serves as the initialization stage for subsequent RL. However, prior work has identified a generalization gap between SFT and RL [Chu et al., 2025, Shenfeld et al., 2025]. As an off-policy method, SFT fits every token in fixed target demonstrations, which can induce overfitting and distort pretrained knowledge when the targets are misaligned with the model's own distribution. In contrast, RL optimizes on samples generated by the model itself, reducing distribution mismatch and better preserving the model's knowledge structure.

To improve SFT generalization, recent work selectively reweights target tokens according to their alignment with the model's predictive distribution [Wu et al., 2026, Diao et al., 2026, Zhu et al., 2026, Lin et al., 2026, Zhang et al., 2026b, Yu et al., 2026]. These methods estimate token-level utility from prediction statistics such as target-token probability or entropy, to modulate each token's contribution and reduce updates from tokens weakly supported by the model's predictive distribution. However, they typically compute these statistics from the model currently being fine-tuned, raising an important question: *does the predictive distribution during fine-tuning still reflect the model's pretrained knowledge structure?* We show that it does not: the online distribution rapidly drifts from the pretrained distribution, creating a self-reinforcing loop that concentrates the learning signal on initially preferred tokens. As a result, the fine-tuned model's distribution no longer faithfully represents the pretrained knowledge distribution, but becomes entangled with optimization dynamics.

In this work, we revisit token-reweighted SFT by deriving a cleaner reweighting signal directly from the pretrained model. Empirically, we show that computing token weights from the pretrained model consistently improves performance over using the online model being fine-tuned. We interpret the pretrained predictive distribution as an estimate of *prior support*: the extent to which each target token is supported by the model's pretrained knowledge and representations before task-specific adaptation. Tokens with stronger prior support are better aligned the pretrained model's knowledge structure, thus yielding learning signals that are less likely to distort existing representations.

Based on this principle, we introduce **PriFT** (Prior-support guided Fine-Tuning), a simple SFT framework that derives token weights from the frozen pretrained model. PriFT decouples token-weight estimation from the optimization trajectory and preserves the prior-support signal throughout fine-tuning. We study two instantiations: PriFT-prob uses pretrained target-token probability, while PriFT-mass selects tokens by cumulative mass under the pretrained distribution to reduce bias toward easy high-confidence tokens. Extensive experiments on mathematical reasoning, code generation, and medical question answering show that PriFT achieves state-of-the-art performance among strong token-reweighted SFT baselines. Unlike online reweighting methods, PriFT avoids self-reinforcing concentration of supervision signal to a narrow subset of initially preferred tokens, thereby preserving greater sampling diversity and yielding stronger pass@k performance. **Notably, simply switching the same probability-based reweighting signal from online model to pretrained reference improves Pass@16 by 14.76 points on `Qwen2.5-Math-7B`.** We further show that PriFT provides a better initialization for subsequent RL training, suggesting that pretrained-reference weighting preserves capacity for further on-policy optimization. Our contributions are fourfold:

- We show that the frozen pretrained model give a cleaner token-reweighting signal than the online model, yielding consistent gains across existing token-reweighting methods.
- We propose PriFT, a prior-support guided framework that computes token weights from a frozen pretrained model, providing a stable reweighting signal throughout fine-tuning.
- We demonstrate the strongest performance among token-reweighted SFT methods across mathematical reasoning, code generation, and medical QA, with representative gains of **3.97** Avg@16 and **8.75** Pass@16 points over the strongest prior baseline on `Qwen2.5-Math-7B`.
- We show that PriFT provides a stronger RL initialization than baseline SFT methods, yielding up to **9.57** points of Avg@16 improvement after RL.

2 Background

Notation and problem formulation. Let $\mathcal{D} = \{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^{N}$ denote a dataset of prompt-response pairs. For $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}$, $\mathbf{x}$ is the input prompt and $\mathbf{y} = (y_1, \ldots, y_T)$ is the target response. Let $\pi_{\boldsymbol{\theta}}$ be a language model parameterized by $\boldsymbol{\theta}$, initialized from a pretrained model $\pi_{\boldsymbol{\theta}_{\text{pt}}}$. SFT minimizes token-level negative log-likelihood: $\mathcal{L}_{\text{SFT}}(\boldsymbol{\theta}) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}\left[-\sum_{t=1}^{T} \log \pi_{\boldsymbol{\theta}}(y_t \mid \mathbf{x}, y_{<t})\right]$. We refer to the model being fine-tuned as the online model $\pi_{\boldsymbol{\theta}_{\text{on}}}$, in contrast to the pretrained model $\pi_{\boldsymbol{\theta}_{\text{pt}}}$.

Prior work generalizes the standard SFT objective by assigning non-uniform weights to individual tokens during fine-tuning, yielding the weighted SFT objective:

$$\mathcal{L}_{\text{WSFT}}(\boldsymbol{\theta}) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}\left[-\sum_{t=1}^{T} m_t \log \pi_{\boldsymbol{\theta}}(y_t \mid \mathbf{x}, y_{<t})\right], \tag{1}$$

where $m_t \geq 0$ controls contribution of token y_t to training; setting all $m_t = 1$ recovers standard SFT.

Related work on token reweighting with model-prediction statistics. To improve SFT generalization, recent work derives token weights from the online model's predictive distribution $\pi_{\boldsymbol{\theta}_{\text{on}}}(\cdot \mid \mathbf{x}, y_{<t})$ under teacher forcing [Wu et al., 2026, Diao et al., 2026, Liu et al., 2026, Lin et al., 2026, Zhu et al., 2026, Zhang et al., 2026b, Yu et al., 2026]. This distribution provides a token-level diagnostic of how each supervised target aligns with the current model, and is therefore used to estimate token utility or difficulty. Existing methods convert such statistics into token weights to emphasize tokens better aligned with model's distribution and reduce potentially harmful updates. Two statistics are commonly used: target probability: $p_t = \pi_{\boldsymbol{\theta}_{\text{on}}}(y_t \mid \mathbf{x}, y_{<t})$, which measures the online model's confidence in the target token; and prediction entropy: $H_t = -\sum_{v\in\mathcal{V}} \pi_{\boldsymbol{\theta}_{\text{on}}}(v \mid \mathbf{x}, y_{<t}) \log \pi_{\boldsymbol{\theta}_{\text{on}}}(v \mid \mathbf{x}, y_{<t})$, which measures the uncertainty over the next-token distribution.

Table 1 summarizes representative token-reweighted SFT methods. Confidence-based methods, including DFT [Wu et al., 2026], TALR [Lin et al., 2026], and ProFit [Liu et al., 2026], use target-token probability to emphasize tokens already supported by the online model. DFT directly uses p_t as a soft weight, TALR further applies temperature scaling and lower-bound clipping, and ProFit converts the same signal into a hard selection mask. Entropy-based methods use predictive uncertainty: EAFT emphasizes high-entropy tokens [Diao et al., 2026], while IDFT combines target probability and entropy through a centered log-likelihood statistic [Zhang et al., 2026b]. ASFT further adds a KL anchoring term to DFT to constrain the online model toward the pretrained model [Zhu et al., 2026].

Table 1: Summary of baseline token-weighting rules. $\text{sg}(\cdot)$ represents stop gradient.

Method	**Token weight** (m_t in Equation 1)
SFT	1
DFT	$\text{sg}(p_t)$
ProFit	$\mathbf{1}[p_t \geq \tau]$
TALR	$\max\{\text{sg}(p_t^{1/\tau}), w_{\min}\}$
EAFT	$\widetilde{H}_t = H_t^{\text{top-}K} / \log K$
ASFT	$\text{sg}(p_t)$ with KL anchoring
IDFT	$p_t^{\exp(-\phi_t)}$, $\phi_t = \log p_t + H_t$

3 Pretrained model provides more reliable token-reweighting signals

Despite the variety of weighting rules, existing methods typically derive token weights from currently fine-tuned model's prediction distribution. This motivates a simple question:

Is the online model's predictive distribution the best source for computing token weights?

We argue that it is often not. Although online token-level statistics reflect the predictive distribution of the current model, they make token weights dependent on the optimization trajectory. As fine-tuning updates the model, the weighting signal becomes a moving target and may no longer faithfully capture the token-level properties it is intended to measure. We argue that the pretrained model provides a more reliable source of token statistics, and provides both empirical evidence and analyses.

3.1 Empirical motivation

We begin with a controlled comparison on medical question answering to isolate the effect of the *source of token weights* while keeping the weighting rule fixed. Following Zhu et al. [2026], we fine-tune `LLaMA-2-7B`[3] on 10k MedMCQA [Pal et al., 2022] examples and evaluate on

[3] Zhu et al. [2026] use `LLaMA-2-7B` to reduce potential contamination from prior supervised knowledge.

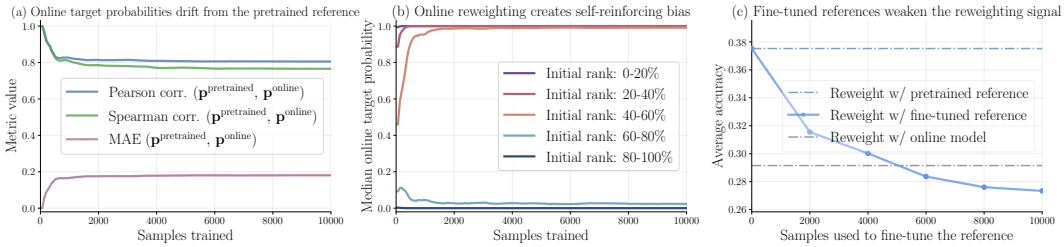

Figure 2: **A clean pretrained reference provides more reliable token-level reweighting signals.** (a) Online target probabilities drift rapidly from pretrained probabilities: correlations decrease early in training. (b) Online reweighting amplifies initial token-rank bias: tokens with high pretrained support are pushed toward near-deterministic probabilities, while low-ranked tokens remain weakly supported. (c) Reweighting signals derived from partially fine-tuned references become progressively less effective, showing the importance of preserving an uncontaminated pretrained reference.

MMLU-Medical [Hendrycks et al., 2020], MedQA [Jin et al., 2021], and MedMCQA test set, reporting average accuracy across the three benchmarks.

We consider DFT-style target-probability reweighting [Wu et al., 2026], top-50% and bottom-50% token selection by target probability, and top-50% and bottom-50% token selection by predictive entropy. For each rule, we compare token reweighting using predictions from the online model versus the frozen pretrained model.

Table 2: **Pretrained-model reweighting consistently improve online-model reweighting.**

Token reweighting model	Original	SFT	p_t rw.	p_t top-50	p_t bot.-50	H_t top-50	H_t bot.-50
Online model	31.40	33.37	29.59	27.32	33.78	31.04	24.27
Pretrained model	31.40	33.37	**35.81**	**30.59**	**35.68**	**33.91**	**27.37**

Table 2 shows that reweighting with pretrained model consistently outperform using online model across all tested rules, including confidence-based reweighting and selection, and entropy-based selection [4]. This indicates that the token reweighting source is a key design choice, and a pretrained model can provide a more effective weighting signal than the evolving online model in this empirical setting.

3.2 Analysis: Why the pretrained reference improves token reweighting

Having established the empirical benefit, we next analyze why a clean pretrained model provides a better signal for token reweighting. Let $p_t^{\text{pt}} = \pi_{\boldsymbol{\theta}_{\text{pt}}}(y_t \mid \mathbf{x}, y_{<t})$ and $p_t^{\text{on}} = \pi_{\boldsymbol{\theta}_{\text{on}}}(y_t \mid \mathbf{x}, y_{<t})$ denote the target probability under the pretrained reference and the online model, respectively. We analyze a model fine-tuned on 10k MedMCQA examples with DFT [Wu et al., 2026], a representative online reweighting method that reweights each token's contribution by p_t^{on}. We then examine three factors that explain why signals derived from the online model are less reliable.

Online reweighting signal drifts rapidly during fine-tuning. When token statistics are computed from the online model, the weighting signal changes together with the model being fine-tuned. To quantify this drift, we compare the pretrained target probability $\boldsymbol{p}^{\text{pt}}$ with the online probability $\boldsymbol{p}^{\text{on}}$ during fine-tuning on response tokens from 64 held-out examples, using Pearson correlation, Spearman correlation, and mean absolute error. Figure 2(a) shows that the drift occurs mostly at the beginning of fine-tuning. Both correlations quickly drop to around 0.8, while the mean absolute error rises to about 0.18. Thus, online-derived signals rapidly depart from the initial pretrained reference and remain shifted throughout fine-tuning, resulting in non-stable token weights.

Online reweighting creates a self-reinforcing bias. This drift is not merely random: it induces a self-reinforcing bias in the token weights. Tokens that are initially favored by the pretrained model receive larger online probabilities, and therefore larger weights and updates, while initially disfavored tokens receive progressively weaker supervision. To visualize this effect, we group response tokens from the 64 held-out examples into five equal-sized bins according to their initial rank under p_t^{pt}, and track the median value of p_t^{on} in each bin during fine-tuning. As shown in Figure 2(b), tokens in the

[4] These experiments use a low-data regime to enable a controlled comparison of token-statistics sources. In this setting, token reweighting can underperform standard SFT, as also observed by Zhu et al. [2026].

initially favored bins are rapidly pushed toward near-deterministic target probabilities, whereas the lowest-ranked bins are pushed toward zero. This reveals a rich-get-richer dynamic: online reweighting amplifies the model's initial preference and concentrates supervision on a subset of preferred tokens [5].

Being frozen is not enough: the pretrained reference must remain uncontaminated. One possible explanation is that the pretrained model benefits token reweighting simply because it provides a fixed reference, in which case any frozen reference should be similarly effective. We test this by constructing frozen reference models from different stages of fine-tuning. Specifically, we fine-tune reference models with DFT on varying numbers of training samples, freeze each checkpoint, and use it to provide reweighting signal for the training of a new downstream model initialized from the original pretrained checkpoint on a separate 10k-sample training set. As shown in Figure 2(c), the best performance of the downstream model is achieved using original pretrained model as reference. Fine-tuning the reference on as few as 2k samples already causes a substantial drop, and performance continues to decline with further fine-tuning, eventually falling below online-model reweighting. These results indicate that freezing alone is insufficient: preserving the pretrained distribution is critical, as fine-tuning progressively degrades the usefulness of the reference token statistics for reweighting.

For comparison, we repeat the same analysis for a model trained with standard SFT in Figure 5 in appendix. Compared with DFT, SFT exhibits smaller drift from the pretrained reference, weaker self-reinforcing bias, and a milder performance drop when providing reweighting signals. The comparison suggests that these effects are not only caused by fine-tuning, but amplified by online reweighting.

3.3 Interpretation: pretrained model provides an estimate of prior support

The results suggest that the pretrained model is not merely a fixed source of token reweighting. Rather, the pretrained target-token probability provides an estimate of *prior support*: how strongly each target token is supported by the model's pretrained representations before task-specific adaptation.

We interpret prior support as a proxy for adaptation cost: the negative log-probability of a target token under the pretrained model corresponds to the KL divergence from the one-hot target distribution to the pretrained distribution. Thus, tokens with lower prior support exhibit greater initial distribution mismatch and may require more substantial adaptation. Crucially, the prior support signal must remain pretrained, as fine-tuning entangles it with optimization dynamics and compromises its reliability.

4 PriFT: Prior-support Guided Fine-tuning

We propose **PriFT** (Prior-support guided Fine-Tuning), a framework that constructs token weights from the prior support from pretrained model, decoupling reweighting from optimization trajectory [6].

PriFT framework. For each training example $(\mathbf{x}, \mathbf{y})$ and target token y_t, PriFT assigns a token weight using the pretrained predictive distribution: $m_t = u\big(\pi_{\boldsymbol{\theta}_{\text{pt}}}(\cdot \mid \mathbf{x}, y_{<t}),\, y_t\big)$, where $u(\cdot)$ is a token-utility function that maps the pretrained distribution and the target token to a nonnegative weight m_t. The online model is then trained with the weighted SFT objective $\mathcal{L}_{\text{PRIFT}}(\boldsymbol{\theta}_{\text{on}}) = -\sum_{t=1}^{T} u\big(\pi_{\boldsymbol{\theta}_{\text{pt}}}(\cdot \mid \mathbf{x}, y_{<t}),\, y_t\big) \log \pi_{\boldsymbol{\theta}_{\text{on}}}(y_t \mid \mathbf{x}, y_{<t})$.

By reweighting token contributions according to their prior support, PriFT encourages *conservative adaptation*: it fits the target data while limiting excessive deviation from the pretrained distribution. This preserves task adaptation while reducing the influence of tokens that are poorly aligned with the pretrained model and may distort pretrained knowledge. PriFT is a general reweighting framework, not a single weighting rule. We study two instantiations: **PriFT-prob**, based on pretrained target-token probability, and **PriFT-mass**, based on cumulative-mass support under the pretrained distribution.

PriFT-prob: Probability-based reweighting from the pretrained reference. PriFT-prob is the most direct instantiation of PriFT. It uses the probability-based utility of DFT [Wu et al., 2026], but computes the target-token probability from the frozen pretrained reference instead. For each target token y_t, we define $m_t = \pi_{\boldsymbol{\theta}_{\text{pt}}}(y_t \mid \mathbf{x}, y_{<t})$, which gives

$$\mathcal{L}_{\text{PRIFT-PROB}}(\boldsymbol{\theta}_{\text{on}}) = -\sum_{t=1}^{T} \pi_{\boldsymbol{\theta}_{\text{pt}}}(y_t \mid \mathbf{x}, y_{<t}) \log \pi_{\boldsymbol{\theta}_{\text{on}}}(y_t \mid \mathbf{x}, y_{<t}). \tag{2}$$

[5] We provide an example of the self-reinforcing bias of online reweighting in Figure 6 in the appendix.

[6] The pretrained model refers to the checkpoint before SFT in this paper, not necessarily the checkpoint after pretraining.

PriFT-prob scales each token's contribution according to its pretrained support, assigning stronger supervision to tokens that are more plausible under the pretrained distribution while downweighting those with weaker support. Because the reweighting signal remains fixed throughout fine-tuning, it avoids the self-reinforcing dynamics of online probability reweighting discussed in Section 3.2.

PriFT-prob works best when fine-tuning targets are well covered by the pretrained distribution, so that pretrained probability provides a reliable learning signal. However, raw probability can overemphasize easy high-confidence tokens. In knowledge-intensive or domain-specific settings, important tokens may have low initial probability due to limited domain coverage, causing direct probability weighting to suppress updates needed for adaptation. This motivates PriFT-mass, which keeps the same prior-support principle but replaces raw probability with a cumulative support measure.

PriFT-mass: Cumulative-mass selection from the pretrained reference. Raw target probability is biased toward easy positions where the pretrained distribution is sharp. Even when an easy token and a harder token are both well ranked under the pretrained model, the easy token can receive a much larger absolute probability simply because its local distribution is more concentrated. As a result, PriFT-prob may overemphasize high-confidence tokens, including trivial continuations such as function words or formatting tokens, while underweighting harder tokens that encode task-specific knowledge or key reasoning steps. This motivates a utility that measures target-token support relative to alternative candidates under the same pretrained distribution, rather than by absolute probability alone.

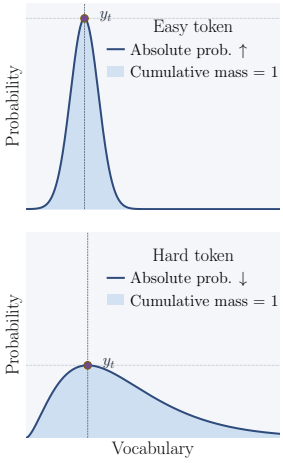

Figure 3: Cumulative-mass support reduces the bias toward easy tokens with sharp distributions.

To this end, we introduce **PriFT-mass**, which instantiates PriFT with a cumulative-mass utility. Let $p_t^{\mathrm{pt}}(v) = \pi_{\boldsymbol{\theta}_{\mathrm{pt}}}(v \mid \mathbf{x}, y_{<t})$ denote the pretrained reference distribution at token position t. We define $u_t^{(\mathrm{MASS})} = \sum_{v:\, p_t^{\mathrm{pt}}(v) \le p_t^{\mathrm{pt}}(y_t)} p_t^{\mathrm{pt}}(v)$. This score measures the probability mass assigned to tokens no more likely than the target token under the pretrained distribution. Thus, a target token receives a large score when it is supported above a substantial fraction of the reference probability mass, even if its raw probability is moderate. As illustrated in Figure 3, a top-ranked target token receives a score of 1 regardless of whether the distribution is sharp or flat. PriFT-mass can therefore retain informative tokens from harder contexts that PriFT-prob may downweight. We convert this score into a binary token-selection mask with a fixed threshold of 0.5:

$$\mathcal{L}_{\text{PRIFT-MASS}}(\boldsymbol{\theta}_{\mathrm{on}}) = -\sum_{t=1}^{T} m_t \log \pi_{\boldsymbol{\theta}_{\mathrm{on}}}(y_t \mid \mathbf{x}, y_{<t}), \quad m_t = \mathbf{1}\left[u_t^{(\mathrm{MASS})} \ge 0.5\right]. \tag{3}$$

Since $u_t^{(\mathrm{MASS})}$ is a percentile-based support score, the threshold 0.5 keeps tokens supported above at least half of the pretrained probability mass, filtering tokens in the lower-support half. Unlike PriFT-prob, PriFT-mass uses the pretrained reference only for token selection; all selected tokens then receive equal training weight regardless of their prediction difficulty.

Computational overhead. PriFT introduces limited additional computation. Let F denote the FLOPs of one forward pass over the training set. Since one backward pass costs approximately $2F$, standard SFT trained for E epochs requires about $3EF$ FLOPs. PriFT adds a single frozen forward pass of the pretrained reference model to comput token statistics, giving a total cost of approximately $F + 3EF$ and a relative overhead of $1/(3E)$. For example, when training for 2 epochs, this corresponds to 16.7% additional computation.

This extra cost can be incurred offline *before* fine-tuning. During training, PriFT can use cached token weights or masks and follows the same forward–backward optimization procedure as standard weighted SFT. Therefore, PriFT does not necessarily require loading an additional reference model during training, and does not introduce extra GPU memory overhead for maintaining a second model. In contrast, KL-based regularization methods between the fine-tuned and pretrained models generally require access to the reference distribution during optimization [Zhu et al., 2026], as caching such distributions would require storing full-vocabulary logits or probabilities for every token, whereas PriFT is much cheaper to cache by storing only one scalar weight or binary mask per target token.

Table 3: Mathematical reasoning performance across five benchmarks with two different models. PriFT achieves the strongest overall performance on both models.

Model	Method	MATH-OAI		Minerva Math		OlympiadBench		AIME24		AMC23		Average	
		Avg@16	P@16	Avg@16	P@16	Avg@16	P@16	Avg@16	P@16	Avg@16	P@16	Avg@16	P@16
Qwen2.5-Math (7B)	Original	40.53	89.80	12.99	49.63	17.66	58.67	8.74	40.00	24.84	82.50	20.95	64.12
	SFT	54.17	89.00	17.26	50.74	18.79	54.52	2.49	13.33	25.00	70.00	23.54	55.52
	DFT	68.55	84.80	27.55	42.65	34.50	56.00	7.93	13.33	40.94	72.50	35.89	53.86
	EAFT	53.11	87.80	18.84	52.94	18.85	54.67	2.29	13.33	22.19	75.00	23.06	56.75
	IDFT	66.76	82.40	27.04	41.18	31.71	49.78	6.47	20.00	45.31	77.50	35.46	54.17
	TALR	69.70	88.20	**29.12**	45.59	34.71	59.85	7.93	16.67	42.03	80.00	36.70	58.06
	ASFT	68.38	91.80	24.82	51.84	33.95	63.11	7.93	20.00	42.50	77.50	35.90	60.85
	PriFT-prob	73.12	91.80	26.40	52.21	**36.14**	**65.78**	12.72	43.33	49.84	**90.00**	39.65	68.62
	PriFT-mass	**73.83**	**92.20**	26.71	**54.04**	35.75	61.78	**13.34**	**50.00**	**53.75**	**90.00**	**40.67**	**69.60**
Qwen3-Base (8B)	Original	57.93	**93.20**	20.61	54.41	27.23	**66.07**	8.13	**33.33**	35.47	**85.00**	29.87	**66.40**
	SFT	51.61	88.60	20.82	**56.62**	19.00	54.07	1.86	13.33	22.81	62.50	23.22	55.03
	DFT	69.01	85.20	29.38	49.63	33.34	58.07	6.46	16.67	43.59	77.50	36.36	57.41
	EAFT	53.10	88.00	19.40	55.15	18.90	53.93	2.30	13.33	24.20	70.00	23.58	56.08
	IDFT	69.70	82.80	24.20	37.87	33.20	49.33	7.50	10.00	43.00	67.50	35.51	49.50
	TALR	68.60	88.80	28.50	49.26	31.70	58.67	5.60	16.67	40.90	82.50	35.07	59.18
	ASFT	67.80	91.40	28.07	51.84	32.41	63.41	7.72	20.00	39.06	**85.00**	35.01	62.33
	PriFT-prob	68.65	91.40	28.03	54.41	34.20	63.85	8.33	26.67	44.06	82.50	36.65	63.77
	PriFT-mass	**70.99**	90.80	**30.43**	54.78	**36.36**	64.00	**11.24**	**33.33**	**47.03**	**85.00**	**39.21**	65.58

5 Experiments

5.1 Experimental setup

We evaluate PriFT across mathematical reasoning, code generation and medical question answering [7]. Furthermore, we evaluate using PriFT as an initialization for subsequent RL stage.

Mathematical reasoning. For mathematical reasoning, we follow the setup from Wu et al. [2026], reporting main results on `Qwen2.5-Math-7B` and `Qwen3-8B-Base` [Yang et al., 2025].[8] Each model is fine-tuned for one epoch on 100k randomly sampled examples from `NuminaMath-CoT` [Li et al., 2024]. We evaluate on `math_oai` [Hendrycks et al., 2021], `minerva_math` [Lewkowycz et al., 2022], `olympiadbench` [He et al., 2024], `aime24`, and `amc23`, using the default chat template with chain-of-thought prompting. Results are aggregated over 16 stochastic decoding runs: Avg@16 denotes the mean accuracy across the 16 runs, and Pass@16 denotes the fraction of examples solved by at least one sampled response. We provide full training and evaluation details in Appendix C.

Code generation. For code generation, we fine-tune `Qwen2.5-Coder-3B` [Hui et al., 2024] and `Qwen3-4B-Instruct` [Yang et al., 2025] for one epoch on `Tulu-3-SFT-Personas-Code` [Lambert et al., 2024], containing 34,999 examples focused on diverse Python coding questions. We evaluate on four benchmarks: `HumanEval+` [Chen et al., 2021, Liu et al., 2023], `MBPP+` [Austin et al., 2021, Liu et al., 2023], and two recent contest splits from `LiveCodeBench` [Jain et al., 2024] (v5 and v6). We report pass@1 accuracy on each benchmark.

Baselines. We compare PriFT with standard SFT and several token-reweighted SFT methods. DFT [Wu et al., 2026] and TALR [Lin et al., 2026] derive token weights from the target-token probability, while ASFT [Zhu et al., 2026] extends DFT with a KL regularization term to the pretrained model. IDFT [Zhang et al., 2026b] further emphasizes tokens with higher signal-to-noise ratios based on centered log-likelihood of the target token. EAFT [Diao et al., 2026] instead uses predictive entropy to restrict training to high-entropy tokens.

5.2 Main results on supervised fine-tuning

Mathematical reasoning. Table 3 reports results on five mathematical reasoning benchmarks. Across both backbones, PriFT-mass achieves the strongest aggregate performance among fine-tuning methods, closely followed by PriFT-prob. On `Qwen2.5-Math-7B`, PriFT-mass reaches 40.67 Avg@16 and 69.60 Pass@16, improving over the strongest prior fine-tuning baseline by 3.97 and 8.75

[7] We provide the experimental setup and results of medical question answering results in Appendix E.6.

[8] Additional results on `Qwen2.5-1.5B`, `Qwen2.5-Math-1.5B`, `Qwen2.5-Instruct-1.5B`, `Qwen2.5-14B`, and `DeepSeekMath-7B` are provided in the appendix.

Table 4: Code generation task performance using `Qwen3-4B-Instruct`. We report pass@1 accuracy on each benchmark and the average.

Method	HumanEval+	MBPP+	LCB v5	LCB v6	Avg
Original	**82.93**	65.61	30.45	28.34	51.83
SFT	68.29	63.23	22.39	21.52	43.86
DFT	60.37	60.85	24.32	23.22	42.19
IDFT	65.85	61.90	26.14	24.74	44.66
EAFT	68.29	63.76	23.86	23.13	44.76
TALR	60.98	65.34	25.00	23.60	43.73
ASFT	70.73	64.29	28.64	26.73	47.60
PriFT-prob	76.83	**67.99**	39.32	36.87	**55.25**
PriFT-mass	78.66	64.81	**39.89**	**37.25**	55.15

Table 5: RL performance from different initialized SFT checkpoints, including relative improvement compared to prior-RL.

Model	Init.	Avg@16 (Δ)	Pass@16 (Δ)
Qwen2.5-Math 1.5B	SFT	32.22 (+14.4)	61.58 (+11.3)
	DFT	35.22 (+3.81)	59.13 (+4.89)
	PriFT-prob	38.64 (+7.26)	65.96 (+2.25)
	PriFT-mass	**38.96** (+6.26)	**67.23** (+2.71)
Qwen3-8B Base	SFT	42.67 (+19.5)	**70.24** (+15.2)
	DFT	39.63 (+3.28)	59.40 (+1.99)
	PriFT-prob	46.23 (+9.57)	70.16 (+6.40)
	PriFT-mass	**47.65** (+8.44)	67.58 (+1.99)

points, respectively. On `Qwen3-8B-Base`, PriFT-mass obtains 39.21 Avg@16 and 65.58 Pass@16, outperforming the strongest prior baseline by 2.85 and 3.25 points.

A notable pattern is that online-reweighting methods often improve Avg@16 at the cost of Pass@16. On `Qwen2.5-Math-7B`, DFT improves Avg@16 over SFT by 12.35 points, but reduces Pass@16 by 1.66 points; in contrast, PriFT-mass improves Avg@16 by 17.13 points and Pass@16 by 14.08 points. On `Qwen3-8B-Base`, DFT improves Avg@16 over SFT by 13.14 points but gains only 2.38 points in Pass@16, while PriFT-mass improves Avg@16 by 15.99 points and Pass@16 by 10.55 points.

The comparison with DFT directly isolates the effect of the reweighting-signal source. DFT and PriFT-prob share the same probability-based weighting form, but differ only in the source of the probability: DFT uses the evolving online model, whereas PriFT-prob uses the frozen pretrained model. This change in the reweighting signal improves Pass@16 over DFT by 14.76 points on `Qwen2.5-Math-7B` and 6.36 points on `Qwen3-8B-Base`. This suggests that online reweighting can concentrate probability mass on a narrower set of preferred reasoning paths, whereas a fixed pretrained-reference signal mitigates this trade-off and better preserves diverse successful trajectories. Additionally, we show that the gains are robust across independent random seeds (Appendix E.2), and remain effective after normalizing token weights (Appendix E.1).

Code generation. Table 4 reports code generation results on `Qwen3-4B-Instruct`, with results on `Qwen2.5-Coder-3B` deferred to Appendix E.8. Both PriFT variants outperform standard SFT and prior token-reweighted baselines on average, with the largest gains on the more recent LiveCodeBench splits. PriFT-prob achieves the best average accuracy of 55.25, improving over the strongest prior baseline by 7.65 points, while PriFT-mass reaches a comparable 55.15.

A notable pattern is that fine-tuning often degrades performance on HumanEval+ and MBPP+, where the original model already performs strongly. No fine-tuning method improves over the original checkpoint on HumanEval+, but PriFT incurs the smallest degradation: PriFT-mass drops by only 4.27 points, compared with 14.64 for SFT. At the same time, PriFT improves substantially on the more recent LiveCodeBench splits: PriFT-mass improves over the closest fine-tuning baseline by 11.25 points on LCB v5 and 10.52 points on LCB v6. This suggests that pretrained-reference weighting better preserves existing capabilities while still enabling adaptation to harder coding tasks.

Medical QA. Table 13 in Appendix E.6 reports medical QA results with `LLaMA-2-7B`. PriFT-mass, together with ASFT, achieves the strongest overall performance, obtaining the best results on MMLU and MedMCQA. PriFT-prob also performs strongly, outperforming the remaining token-reweighted SFT baselines overall. These results show that PriFT generalizes beyond math and code to knowledge-intensive domains such as medical question answering.

5.3 PriFT provides a better initialization for RL

Across diverse evaluation domains, PriFT consistently improves supervised fine-tuning performance, we next evaluate how different SFT methods affect subsequent RL performance. Performing RL after SFT is a standard recipe for further improving model capabilities, especially on reasoning tasks. We compare standard SFT, DFT, and PriFT-variants as initializations and apply DAPO [Yu et al., 2025] to each post-SFT checkpoint [9], using `Qwen2.5-Math-1.5B` and `Qwen3-8B-Base` as backbones. For each backbone, all initializations share the same RL configuration provided in Appendix F.

[9] Due to the large computational cost for RL training, we considered only two representative SFT baselines.

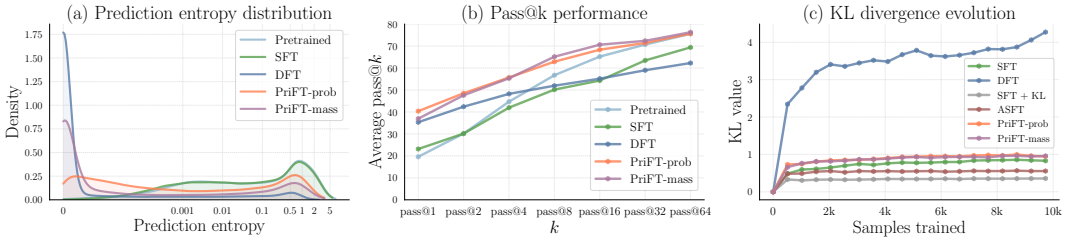

Figure 4: (a) PriFT preserves broader token-level entropy distributions than DFT. (b) PriFT achieves strong pass@k performance across different sampling budgets. (c) PriFT induces substantially smaller KL divergence from the pretrained model than DFT during training.

Table 5 shows that PriFT achieves the strongest post-RL performance overall. On `Qwen2.5-Math-1.5B`, PriFT-mass reaches 38.96 Avg@16 and 67.23 Pass@16, improving over the strongest baseline by 3.74 and 5.65 points, respectively. On `Qwen3-8B-Base`, PriFT-mass achieves the best Avg@16 of 47.65, improving over the strongest baseline by 4.98 points, while PriFT-prob matches the strongest Pass@16 performance with 70.16. The results reveal different failure modes of the baselines: SFT benefits substantially from RL but delivers weaker supervised checkpoint, while DFT yields a stronger SFT checkpoint but gains less from RL, especially on Pass@16. This is consistent with our earlier finding that online reweighting can concentrate probability mass on a narrower set of reasoning trajectories, reducing the headroom for RL to discover additional correct solutions. In contrast, PriFT provides strong supervised checkpoints that remain amenable to further on-policy optimization. In the next section, we show that this benefit is associated with PriFT preserving greater sampling diversity than online-reweighting methods.

6 Analyses and Discussions

We compare PriFT with DFT, a representative online token-reweighting method, in both sampling diversity and distribution shift during fine-tuning, and present ablations to justify the design choices.

6.1 PriFT preserves more sampling diversity

We show PriFT preserves higher sampling diversity. This is important for sampling-based reasoning evaluation and subsequent RL, where maintaining multiple plausible trajectories can provide additional optimization headroom. We evaluate the prediction entropy and pass@k performance for different fine-tuning methods on `Qwen2.5-Math-7B`.

Prediction entropy. We compare token-level prediction entropy on response tokens from 64 held-out samples after fine-tuning. As shown in Figure 4(a), DFT places a large fraction of tokens near zero entropy, indicating that online probability reweighting makes the model overly deterministic through its self-reinforcing bias. In contrast, PriFT variants maintain broader entropy distributions, suggesting that pretrained-reference weighting better preserves uncertainty in the predictive distribution.

Pass@k with increasing k. We further evaluate average pass@k performance as the number of sampled responses k increases. Figure 4(b) shows that although DFT achieves strong pass@k at small values of k, its performance saturates more quickly than the other methods. In comparison, PriFT-prob and PriFT-mass maintain strong pass@k performance across sampling budgets from $k = 1$ to 64. This suggests that PriFT preserves a broader set of successful reasoning trajectories, achieving strong performance across different sampling budgets.

6.2 PriFT reduces distribution shift compared with online reweighting

Prior work shows that DFT can induce a large distribution shift from the pretrained model during fine-tuning [Zhu et al., 2026]. We show that PriFT reduces this shift without explicit KL regularization, by assigning less training signal to tokens weakly supported by the pretrained distribution. Figure 4(c) tracks the average token-level KL divergence from the pretrained distribution to the online distribution when fine-tuning a `LLaMA-2-7B model` on 10k data from MedMCQA [Pal et al., 2022]. DFT shows a large and fast increasing KL divergence, as a result of its self-reinforcing online reweighting dynamics. In contrast, PriFT-prob and PriFT-mass keep the KL substantially

lower, remaining close to SFT throughout training. This suggests that pretrained-reference weighting helps avoid the large distribution drift caused by online reweighting.

6.3 Ablation studies

We summarize the main ablation results here, with additional details provided in Appendix E.5.

Replacing pretrained reference with an EMA model. As shown in Figure 7(a), performance generally improves with larger EMA momentum when replacing the pretrained signal with EMA model, but still does not surpass the pretrained-reference reweighting. This suggests that PriFT benefits from preserving original pretrained distribution, rather than merely smoothing the online model.

Varying token selection threshold in PriFT-mass. We vary the selection threshold τ in PriFT-mass. As shown in Figure 7(b), performance peaks around default threshold $\tau = 0.5$. This indicates PriFT-mass benefits from filtering weakly supported tokens while retaining sufficient supervision.

Reweighting signal from a stronger model. We test if PriFT-prob benefits from reweighting signals provided by a stronger model. As shown in Table 12, using a math-domain reference to reweight a base model yields mild performance gains, whereas a larger math-domain reference does not improve the smaller math-domain model. This suggests that a stronger reference does not necessarily provide a better reweighting signal. The model's own pretrained checkpoint can remain the most reliable source of prior support, as its distribution is best aligned with the model being fine-tuned.

Applying pretrained reweighting signals to existing baselines. We replace the online token-reweighting signals in EAFT [Diao et al., 2026], IDFT [Zhang et al., 2026b], and TALR [Lin et al., 2026] with their pretrained-reference counterparts while keeping the weighting rules unchanged. As shown in Table 10, this consistently improves average Pass@16, showing that pretrained-reference signals benefit not only for PriFT instantiations but also for other token-reweighing rules.

Robustness across fine-tuning datasets. We further evaluate on two alternative fine-tuning datasets, `ScaleQuest-Math` [Ding et al., 2025] and `GSM8K` [Cobbe et al., 2021]. PriFT consistently improves over SFT, although its advantage varies across training distributions: the best PriFT variant improves Avg@16 / Pass@16 by $6.62/1.52$ points on `ScaleQuest-Math`, $17.13/14.08$ points on the main `NuminaMath-CoT` setting, and $9.28/15.70$ points on `GSM8K`. These results show that PriFT is not tied to a particular fine-tuning dataset, while suggesting that the benefit of pretrained-reference weighting can depend on the underlying training distribution. Detailed results are provided in Appendix E.4.

Connection to knowledge distillation (KD). The distribution-matching term in KD decomposes into a target-token and a non-target term: $-\sum_v q_t(v) \log p_t(v) = -q_t(y_t) \log p_t(y_t) - \sum_{v \neq y_t} q_t(v) \log p_t(v)$, where q_t and p_t denote the teacher and student distributions. While target term shares the form as PriFT-prob, we test whether the non-target term benefits PriFT-prob by training `Qwen2.5-Math-1.5B` with `Qwen2.5-Math-7B` as the teacher and varying the non-target term weight β [10]. We note that this diagnostic excludes hard-label CE mixing and temperature scaling. As shown in Table 11, performance declines as β increases, peaking at $\beta = 0$, which corresponds to PriFT-prob with the teacher as reference. This suggests that useful teacher guidance primarily comes from target-token reweighting rather than full distribution matching in this setting.

7 Conclusion and Limitations

We presented PriFT, a token-reweighted SFT framework that derives token weights from a frozen pretrained reference instead of the online model being optimized. By exploiting the prior support from the pretrained model throughout fine-tuning, PriFT improves fine-tuning performance across mathematical reasoning, code generation, and medical question answering. It also provides a stronger initialization for subsequent RL, suggesting that pretrained-reference weighting can improve SFT while maintaining the sample diversity needed for further on-policy optimization.

Our study has limitations and leaves open directions for future work. Our evaluation focuses on moderate-size open models, validating PriFT on larger models and broader applications remains future work. Furthermore, while PriFT demonstrates the benefits of pretrained prior support, extending its effectiveness on tasks with weaker prior support remains an interesting direction for future research.

[10] We use a different teacher as using the same backbone as teacher would yield no distribution-matching signal at $\beta = 1$.

Acknowledgements

The authors thank Guillermo Ortiz-Jimenez, Nikolaos Dimitriadis, Alessandro Favero, Yulun Jiang, Skander Moalla, and the anonymous reviewers for constructive discussions and comments.

References

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *ICLR*, 2024.

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

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

Xiwen Chen, Jingjing Wang, Wenhui Zhu, Peijie Qiu, Xuanzhao Dong, Yueyue Deng, Hejian Sang, Zhipeng Wang, Alborz Geramifard, and Feng Luo. Soda: Semi on-policy black-box distillation for large language models. *arXiv preprint arXiv:2604.03873*, 2026.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*, 2025.

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.

Muxi Diao, Lele Yang, Wuxuan Gong, Yutong Zhang, Zhonghao Yan, Yufei Han, Kongming Liang, Weiran Xu, and Zhanyu Ma. Entropy-adaptive fine-tuning: Resolving confident conflicts to mitigate forgetting. *arXiv preprint arXiv:2601.02151*, 2026.

Yuyang Ding, Xinyu Shi, Xiaobo Liang, Juntao Li, Zhaopeng Tu, Qiaoming Zhu, and Min Zhang. Unleashing llm reasoning capability via scalable question synthesis from scratch. In *ACL*, 2025.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *ICLR*, 2024.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *ACL*, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

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

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

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 2021.

Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. Distillm: Towards streamlined distillation for large language models. *arXiv preprint arXiv:2402.03898*, 2024.

Jongwoo Ko, Tianyi Chen, Sungnyun Kim, Tianyu Ding, Luming Liang, Ilya Zharkov, and Se-Young Yun. Distillm-2: A contrastive approach boosts the distillation of llms. *arXiv preprint arXiv:2503.07067*, 2025.

Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. *arXiv preprint arXiv:2202.10054*, 2022.

Komal Kumar, Tajamul Ashraf, Omkar Thawakar, Rao Muhammad Anwer, Hisham Cholakkal, Mubarak Shah, Ming-Hsuan Yang, Phillip HS Torr, Fahad Shahbaz Khan, and Salman Khan. Llm post-training: A deep dive into reasoning large language models. *arXiv preprint arXiv:2502.21321*, 2025.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. In *NeurIPS*, 2022.

Gaotang Li, Ruizhong Qiu, Xiusi Chen, Heng Ji, and Hanghang Tong. Beyond log likelihood: Probability-based objectives for supervised fine-tuning across the model capability continuum. *arXiv preprint arXiv:2510.00526*, 2025a.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13(9):9, 2024.

Xinran Li, Guangda Huzhang, Siqi Shen, Qing-Guo Chen, Zhao Xu, Weihua Luo, Kaifu Zhang, and Jun Zhang. Getting your llms ready for reinforcement learning with lightweight sft. In *ICLR*, 2026.

Ziniu Li, Congliang Chen, Tian Xu, Zeyu Qin, Jiancong Xiao, Zhi-Quan Luo, and Ruoyu Sun. Preserving diversity in supervised fine-tuning of large language models. In *ICLR*, 2025b.

Jiacheng Lin, Zhongruo Wang, Kun Qian, Tian Wang, Arvind Srinivasan, Hansi Zeng, Ruochen Jiao, Xie Zhou, Jiri Gesi, Dakuo Wang, et al. Sft doesn't always hurt general capabilities: Revisiting domain-specific fine-tuning in llms. In *ICLR*, 2026.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *NeurIPS*, 2023.

Tao Liu, Taiqiang Wu, Runming Yang, Shaoning Sun, Junjie Wang, and Yujiu Yang. Profit: Leveraging high-value signals in sft via probability-guided token selection. *arXiv preprint arXiv:2601.09195*, 2026.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Conference on health, inference, and learning*, 2022.

Chongli Qin and Jost Tobias Springenberg. Supervised fine tuning on curated data is reinforcement learning (and can be improved). *arXiv preprint arXiv:2507.12856*, 2025.

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

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. Rl's razor: Why online reinforcement learning forgets less. *arXiv preprint arXiv:2509.04259*, 2025.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. *arXiv preprint arXiv:2601.19897*, 2026.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Ke Wang, Nikolaos Dimitriadis, Alessandro Favero, Guillermo Ortiz-Jimenez, Francois Fleuret, and Pascal Frossard. Lines: Post-training layer scaling prevents forgetting and enhances model merging. In *ICLR*, 2025.

Yue Wang, Weishi Wang, Shafiq Joty, and Steven CH Hoi. Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. In *EMNLP*, 2021.

Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, et al. Robust fine-tuning of zero-shot models. In *CVPR*, 2022.

Yongliang Wu, Yizhou Zhou, Zhou Ziheng, Yingzhe Peng, Xinyu Ye, Xinting Hu, Wenbo Zhu, Lu Qi, Ming-Hsuan Yang, and Xu Yang. On the generalization of sft: A reinforcement learning perspective with reward rectification. In *ICLR*, 2026.

Wenda Xu, Rujun Han, Zifeng Wang, Long T Le, Dhruv Madeka, Lei Li, William Yang Wang, Rishabh Agarwal, Chen-Yu Lee, and Tomas Pfister. Speculative knowledge distillation: Bridging the teacher-student gap through interleaved sampling. *arXiv preprint arXiv:2410.11325*, 2024.

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

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Wenhao Yu, Shaohang Wei, Jiahong Liu, Yifan Li, Minda Hu, Aiwei Liu, Hao Zhang, and Irwin King. Probability-entropy calibration: An elastic indicator for adaptive fine-tuning. *arXiv preprint arXiv:2602.01745*, 2026.

Dylan Zhang, Yufeng Xu, Haojin Wang, Qingzhi Chen, and Hao Peng. Good sft optimizes for sft, better sft prepares for reinforcement learning. *arXiv preprint arXiv:2602.01058*, 2026a.

Miaosen Zhang, Yishan Liu, Shuxia Lin, Xu Yang, Qi Dai, Chong Luo, Weihao Jiang, Peng Hou, Anxiang Zeng, Xin Geng, et al. Towards on-policy sft: Distribution discriminant theory and its applications in llm training. *arXiv preprint arXiv:2602.12222*, 2026b.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

He Zhu, Junyou Su, Peng Lai, Ren Ma, Wenjia Zhang, Linyi Yang, and Guanhua Chen. Anchored supervised fine-tuning. In *ICLR*, 2026.

A Related Work

Token-reweighted supervised fine-tuning. Recent work improves SFT by reweighting or selecting target tokens according to model-derived signals, rather than treating all tokens in a response equally. DFT and TALR use target-token probability to assign larger weights to high-confidence tokens [Wu et al., 2026, Lin et al., 2026], while ProFit uses the same confidence signal for hard token selection [Liu et al., 2026]. EAFT instead gates training with predictive entropy, emphasizing tokens whose predictions remain uncertain [Diao et al., 2026]. IDFT combines probability and entropy through a centered log-likelihood statistic to emphasize tokens with higher estimated learning signal [Zhang et al., 2026b]. Ranktuner takes into account the relative rank of each target label in the predictive distribution of the model [Yu et al., 2026]. Li et al. [2025a] show that probability-based objectives favoring high-confidence tokens outperform standard SFT when pretrained models have strong task-relevant priors, whereas standard SFT remains preferable when such priors are weak. ASFT further adds a pretrained-model anchoring term on top of probability-based reweighting, aiming to regularize the fine-tuned model toward the pretrained distribution [Zhu et al., 2026]. These methods differ in how they define token utility, but typically derive the reweighting signal from the online model being fine-tuned. Consequently, token weights evolve with the optimization trajectory and may reflect early preferences or overfitting. PriFT instead computes weights from a frozen pretrained reference, separating token-weight estimation from online adaptation and preserving a stable estimate of prior support.

Knowledge distillation PriFT superficially resembles knowledge distillation (KD) [Hinton et al., 2015], in that token-level signals from a reference model influence the training of another model. However, standard KD typically transfers knowledge by matching a teacher distribution, whereas PriFT keeps the demonstration token as the supervised target and uses the reference model only to estimate how strongly that token is supported. A closer family is student-aware distillation, where the student's own behavior is used to shape the training signal. DistiLLM [Ko et al., 2024] and DistiLLM-2 [Ko et al., 2025] exploit student-generated outputs for adaptive off-policy or contrastive distillation, while SODA [Chen et al., 2026] constructs a contrastive signal from a one-time static snapshot of the base student's responses. These methods share with PriFT the view that the student's own distribution contains useful information about which supervision signals are relevant or learnable. PriFT differs by using this information at the token level to weight gold demonstrations, rather than using student outputs as distillation targets or contrastive negatives. Finally, recent on-policy distillation methods address the train–test distribution mismatch of off-policy KD by training on student-generated trajectories with dense teacher feedback [Agarwal et al., 2024, Gu et al., 2024, Ko et al., 2024, Xu et al., 2024, Zhao et al., 2026, Shenfeld et al., 2026]. PriFT shares their support-aware view, but takes the opposite route: instead of moving the training distribution toward student-generated states, it keeps offline supervision on high-quality demonstrations, and uses the initial checkpoint of the student to decide which demonstration tokens are supported by the student's own pre-adaptation prior.

Comparison between SFT and RL Previous works have shown that supervised fine-tuning can distort pre-trained features, leading to catastrophic forgetting of pretrained knowledge [Kumar et al., 2022, Wortsman et al., 2022, Wang et al., 2025]. A recent line of work argues that RL often generalizes better than SFT and exists less forgetting of the model's existing capabilities. Chu et al. [2025] show that SFT tends to memorize surface patterns while RL transfers more broadly, and Shenfeld et al. [2025] argue that on-policy RL forgets less precisely because it trains on rollouts the model itself produces. These findings motivate token-reweighting methods that try to make SFT behave more like on-policy training [Qin and Springenberg, 2025, Wu et al., 2026, Zhang et al., 2026b, Zhu et al., 2026].

SFT objectives as initialization for RL A complementary line of work argues that SFT objectives should be evaluated not only by the performance of the fine-tuned checkpoint, but also by the initialization they provide for subsequent RL. Zhang et al. [2026a] propose PEAR, which reweights offline SFT with importance ratios to reduce the offline–online distribution mismatch, and recent work on lightweight SFT similarly observes that overly aggressive supervised fitting can reduce output diversity and limit later RL improvement [Li et al., 2026]. Li et al. [2025b] propose GEM, a method to better preserve the sampling diversity during supervised fine-tuning stage, which can potentially improve exploration to improve performance limits with reinforcement learning. We adopt this perspective as an auxiliary evaluation of token-reweighted SFT. Our results show that PriFT improves over vanilla SFT and DFT both before RL and after applying the same RL procedure,

suggesting that pretrained-reference-guided weighting strengthens the supervised checkpoint without exhausting the trajectory diversity and optimization headroom needed for later on-policy training.

B Baselines

Baseline details. For consistency, we describe all token-reweighted baselines under a unified per-token objective. Let $p_t = \pi_{\boldsymbol{\theta}_{\text{on}}}(y_t \mid \mathbf{x}, y_{<t})$ denote the online model's probability of the target token. Standard weighted SFT minimizes

$$\mathcal{L}(\theta_{\text{on}}) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}\left[-\sum_{t=1}^{T} m_t \log p_t\right],$$

where m_t is the token-level weight. Different baselines instantiate m_t, or the corresponding per-token objective, as follows.

- **SFT**. Standard supervised fine-tuning treats all target tokens equally with $m_t = 1$. The per-token objective is therefore the standard negative log-likelihood, $\ell_t = -\log p_t$.
- **DFT** [Wu et al., 2026]. Dynamic Fine-Tuning rescales each token loss by the model's target-token probability: $m_t = \text{sg}(p_t)$, where $\text{sg}(\cdot)$ denotes stop-gradient. Its per-token objective is $\ell_t = -\,\text{sg}(p_t)\log p_t$. This gives larger weights to tokens that the online model already predicts with higher confidence.
- **TALR** [Lin et al., 2026]. Token-Adaptive Loss Reweighting assigns adaptive weights according to token difficulty. Since $\ell_t = -\log p_t$, its unnormalized weight can be written as $m_t \propto \exp(-\ell_t/\tau) = p_t^{1/\tau}$, where τ is an adaptive temperature parameter. In practice, TALR applies stop-gradient to the weight and uses a lower cutoff $w_{\min}$: $m_t = \max\{\text{sg}(p_t^{1/\tau}), w_{\min}\}$. This downweights low-probability tokens while preventing their weights from vanishing. In our experiments, we set $w_{\min} = 0.01$.
- **ASFT** [Zhu et al., 2026]. Anchored Supervised Fine-Tuning extends DFT with a KL anchoring term to constrain the online model near a fixed reference model, typically the pretrained checkpoint. Its objective can be written as

  $$\ell_t = -\,\text{sg}(p_t)\log p_t + \beta\, D_{\text{KL}}\left(\pi_{\boldsymbol{\theta}_{\text{pt}}}(\cdot \mid \mathbf{x}, y_{<t}) \,\|\, \pi_{\boldsymbol{\theta}_{\text{on}}}(\cdot \mid \mathbf{x}, y_{<t})\right),$$

  where β controls the strength of the anchoring regularization, and we set $\beta = 0.05$ in our experiments.
- **EAFT** [Diao et al., 2026]. Entropy-Adaptive Fine-Tuning uses a normalized predictive-entropy score as a soft gating signal for token-level training. The per-token weight is calculated as $\widetilde{H}_t = H_t^{\text{top-}K}/\log K$, where the token weights are detached and we set $K = 20$ in our experiments. The resulting weight scales each token loss according to the model's uncertainty, suppressing low-entropy tokens while retaining supervision on uncertain tokens.
- **IDFT** [Zhang et al., 2026b]. In-Distribution Fine-Tuning adaptively modifies probability-based token weights using an online score derived from target-token probability and predictive entropy. Compared with DFT, IDFT does not use raw target-token probability directly; instead, it adjusts the probability-based weight according to the token's signal to noise ratio. The per-token weight is calculated as $\text{sg}(p_t^{\exp(-\phi_t)})$ with $\phi_t = \log p_t + H_t$, where in our experiments ϕ_t is clipped to $[-b, b]$ with $b = 1.0$.

C Full training and evaluation details

Mathematical reasoning. For mathematical reasoning, we follow the setup from Wu et al. [2026], and report results on various models. Each model is fine-tuned for one epoch on the same 100k randomly sampled examples from `NuminaMath-CoT` [Li et al., 2024]. We use the AdamW optimizer with a learning rate of 5×10^{-5}, cosine decay, and a warm-up ratio of 0.1. The batch size is 256, and the maximum sequence length is 2048 tokens. We evaluate on `math_oai` [Hendrycks et al., 2021], `minerva_math` [Lewkowycz et al., 2022], `olympiadbench` [He et al., 2024], `aime24`, and

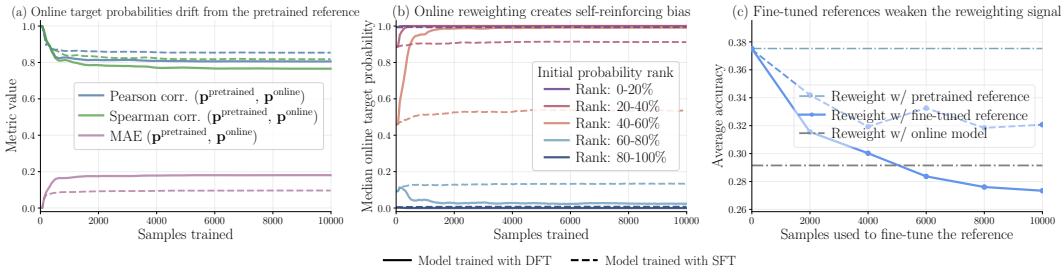

Figure 5: **Additional comparison between DFT and standard SFT for the analysis in Figure 2.** The main text analyzes DFT as a representative online reweighting method; here we repeat the same measurements for a model trained with standard SFT. Solid curves correspond to DFT, and dashed curves correspond to SFT. (a) Both methods change the predictive distribution during fine-tuning, but SFT exhibits weaker drift from the pretrained reference. (b) The self-reinforcing rank bias is much stronger under DFT: high-support tokens are pushed more aggressively toward near-deterministic probabilities, while low-support tokens remain more suppressed. (c) Fine-tuned references degrade less severely under SFT than under DFT. This comparison shows that the instability observed in Figure 2 is amplified by online reweighting, rather than being solely caused by fine-tuning.

`amc23`, using the default chat template with chain-of-thought prompting. Results are aggregated over 16 stochastic decoding runs with a temperature of 1.0 and a maximum generation length of 4096 tokens. Avg@16 denotes the mean accuracy across the 16 runs, and Pass@16 denotes the fraction of examples solved by at least one sampled response.

Code generation. For code generation, we fine-tune `Qwen2.5-Coder-3B` [Hui et al., 2024] and `Qwen3-4B-Instruct` [Yang et al., 2025] for one epoch on `Tulu-3-SFT-Personas-Code` [Lambert et al., 2024], containing 34,999 examples focused on diverse Python coding questions. We use the same optimizer, learning-rate schedule, batch size, and maximum sequence length as in mathematical reasoning. We evaluate on four benchmarks: `HumanEval+` [Chen et al., 2021, Liu et al., 2023], `MBPP+` [Austin et al., 2021, Liu et al., 2023], and two recent contest splits from `LiveCodeBench` [Jain et al., 2024] (v5 and v6). We report pass@1 accuracy using greedy decoding, with a maximum generation length of 768 tokens for `HumanEval+`/`MBPP+` and 2000 tokens for `LiveCodeBench`.

D Additional analyses

D.1 Additional analysis: comparison between DFT and standard SFT

In Section 3.2, we analyze the instability induced by online reweighting using DFT as a representative example. Here, we repeat the same analysis for a model trained with standard SFT, which does not use online model predictions to construct token weights. As shown in Figure 5, SFT also changes the predictive distribution during fine-tuning, but the drift from the pretrained reference is weaker than under DFT. The self-reinforcing rank bias is also less pronounced: high-support tokens are not pushed toward near-deterministic probabilities as aggressively, while low-support tokens remain less suppressed. In addition, when partially fine-tuned checkpoints are used as frozen references, the performance degradation is milder for SFT than for DFT.

These results suggest that the effects observed in Section 3.2 are not merely caused by fine-tuning itself. They are amplified by online reweighting, where the model's evolving predictions directly determine future token weights. This further supports the use of a frozen pretrained model as a stable source of token-reweighting signals.

D.2 Qualitative visualization of self-reinforcing token weights

To further illustrate the self-reinforcing behavior of online reweighting, we visualize the target-token probabilities of a representative held-out example. Figure 6 compares a model trained with DFT and a model trained with standard SFT. Each column corresponds to a target token, and each row corresponds to a training checkpoint. Under DFT, many initially high-probability tokens are rapidly pushed toward probability close to one, while many low-probability tokens remain close to zero

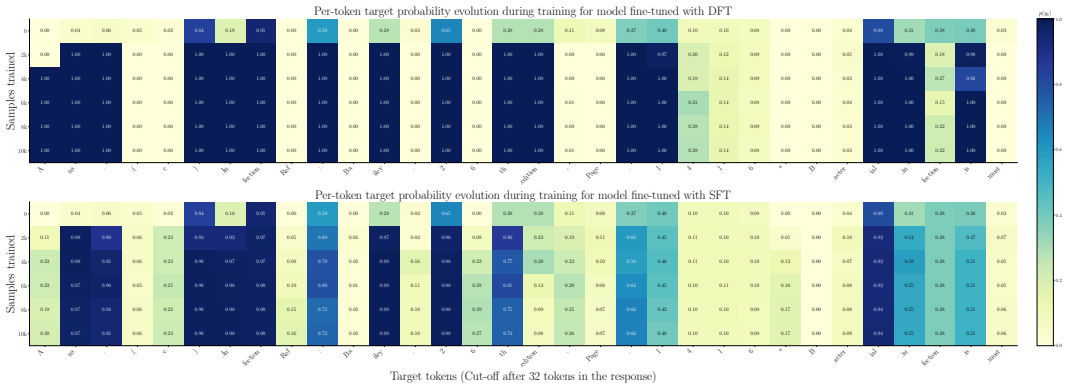

Figure 6: **Qualitative visualization of self-reinforcing token probabilities under DFT.** We track the target-token probability $p(y_t)$ of a representative training example across fine-tuning checkpoints. Columns correspond to target tokens, rows correspond to checkpoints, and colors indicate the online model's probability assigned to the target token. Compared with standard SFT, DFT rapidly pushes many initially high-probability tokens toward probability close to one, while low-probability tokens remain close to zero. This illustrates how online probability-based reweighting amplifies the model's early token preferences and produces a more polarized learning signal.

Table 6: Effect of token-weight normalization on `Qwen2.5-Math-7B`. Each entry reports Avg@16 / Pass@16 averaged over the five mathematical reasoning benchmarks.

Method	Unnormalized	Per-batch	Per-sequence
SFT	23.54 / 55.52	23.54 / 55.52	23.54 / 55.52
DFT	35.89 / 53.86	35.77 / 57.81	37.45 / 56.47
IDFT	35.46 / 54.17	38.18 / 55.43	36.17 / 52.51
TALR	36.70 / 58.06	37.22 / 59.46	35.61 / 59.04
ASFT	35.90 / 60.85	35.89 / 63.55	35.68 / 64.33
EAFT	23.06 / 56.75	23.33 / 55.41	23.69 / 58.13
PriFT-prob	39.65 / 68.62	39.32 / **68.25**	39.78 / **70.14**
PriFT-mass	**40.67** / **69.60**	**40.16** / 66.36	**40.54** / 67.45

throughout training. This produces a polarized token-level pattern, where early preferences of the model are amplified into near-deterministic predictions.

In contrast, standard SFT exhibits a less extreme evolution of target-token probabilities. Although SFT also changes the predictive distribution, it does not use the online probabilities themselves as token weights, and therefore shows weaker probability polarization. This qualitative example supports the analysis in Section 3: online probability-based reweighting can create a feedback loop in which tokens favored early in training receive larger updates. As a result of this rich-get-richer dynamics, tokens with lower initial weights do not get enough supervision during training.

E Additional results

E.1 Robustness to token-weight normalization

Following the original implementations of the token-reweighted baselines, our main experiments apply token weights directly without additional normalization. However, different weighting rules can induce different average loss scales and therefore different effective learning rates. To test whether this confounds the comparison, we repeat the `Qwen2.5-Math-7B` experiments while normalizing token weights to have mean 1, either within each batch or within each sequence. All other training and evaluation settings are kept unchanged.

As shown in Table 6, both PriFT variants remain the two strongest methods on both Avg@16 and Pass@16 under either normalization scheme. In particular, per-sequence normalization yields 39.78

Table 7: Three-seed results on `Qwen2.5-Math-7B`. We report mean ± standard deviation across independent runs.

Method	Avg@16	Pass@16
SFT	23.28 ± 0.36	55.29 ± 0.92
DFT	36.31 ± 0.40	55.02 ± 1.12
EAFT	22.78 ± 0.61	56.39 ± 0.48
IDFT	35.84 ± 1.02	53.40 ± 1.31
TALR	36.57 ± 0.21	58.73 ± 0.66
ASFT	35.97 ± 0.25	62.23 ± 1.21
PriFT-prob	39.22 ± 0.42	68.49 ± 1.01
PriFT-mass	$\mathbf{39.82} \pm 0.82$	$\mathbf{68.72} \pm 0.93$

Table 8: Mathematical reasoning results on `Qwen2.5-14B`, averaged over the five evaluation benchmarks.

Method	Avg@16	Pass@16
Pretrained	21.47	57.96
SFT	22.44	55.80
DFT	32.54	54.32
PriFT-prob	32.75	**63.03**
PriFT-mass	**33.80**	61.48

Avg@16 and 70.14 Pass@16 for PriFT-prob, while PriFT-mass achieves the best Avg@16 of 40.54. These results show that the improvements of PriFT cannot be explained by differences in the overall token-weight scale or effective learning rate.

E.2 Robustness across random seeds

To evaluate the robustness of the reported gains to randomness in training and evaluation, we repeat the `Qwen2.5-Math-7B` experiments using three independent random seeds and report the mean and standard deviation across runs in Table 7.

Both PriFT variants consistently outperform the baselines in the three-run averages. PriFT-mass improves over the strongest baseline mean by 3.25 points on Avg@16 and 6.49 points on Pass@16, while the observed run-to-run variation remains small relative to these margins. This confirms that the main conclusions are robust across independent random seeds.

E.3 Scaling to a 14B model

We further evaluate PriFT at a larger model scale using `Qwen2.5-14B`. We compare the pretrained checkpoint, standard SFT, DFT, PriFT-prob, and PriFT-mass on the same five mathematical reasoning benchmarks used in the main experiments.

As shown in Table 8, the advantage of pretrained-reference weighting persists at the 14B scale. PriFT-mass achieves the best Avg@16 of 33.80, while PriFT-prob achieves the best Pass@16 of 63.03. Compared with DFT, PriFT-prob improves Pass@16 by 8.71 points, and PriFT-mass improves Avg@16 by 1.26 points while also substantially improving Pass@16. These results extend the main findings beyond the 7B–8B models used in the primary experiments.

E.4 Robustness across fine-tuning datasets

Our main mathematical reasoning experiments use `NuminaMath-CoT` for fine-tuning. To test whether the gains depend on this particular training distribution, we repeat the `Qwen2.5-Math-7B` experiments with two substantially different datasets: 100k examples from `ScaleQuest-Math` [Ding et al., 2025] and 7.4k examples from `GSM8K` [Cobbe et al., 2021]. `ScaleQuest-Math` is generated by `Qwen2-Math-7B-Instruct`, a model from the same family as our evaluated model, and may therefore exhibit a smaller distribution mismatch with the target model. In contrast, `GSM8K` is a much smaller and older dataset focused on grade-school mathematics, providing a substantially different

Table 9: Results with alternative mathematical fine-tuning datasets on `Qwen2.5-Math-7B`. We report the average over the five evaluation benchmarks.

Training data	Method	Avg@16	Pass@16
ScaleQuest-Math	SFT	40.08	68.44
	DFT	43.60	62.72
	PriFT-prob	**46.70**	**69.96**
	PriFT-mass	45.82	68.44
GSM8K	SFT	9.50	37.32
	DFT	15.66	42.84
	PriFT-prob	14.40	49.42
	PriFT-mass	**18.78**	**53.02**

fine-tuning distribution. We evaluate the resulting models on the same five mathematical reasoning benchmarks as in the main experiments.

As shown in Table 9, the choice of fine-tuning data substantially affects the absolute performance of all methods. Fine-tuning on `ScaleQuest-Math` produces a particularly strong SFT baseline, reaching 40.08 Avg@16 and 68.44 Pass@16. PriFT-prob further improves these results by 6.62 Avg@16 and 1.52 Pass@16 points, while PriFT-mass improves Avg@16 by 5.74 points and maintains the same Pass@16. In comparison, the relative benefit of PriFT over SFT is substantially larger with the other training distributions. On the main `NuminaMath-CoT` setting, PriFT-mass improves over SFT by 17.13 Avg@16 and 14.08 Pass@16 points. When fine-tuning on `GSM8K`, PriFT-mass improves over SFT by 9.28 Avg@16 and 15.70 Pass@16 points, while PriFT-prob improves Pass@16 by 12.10 points.

This comparison is consistent with the role of prior support in controlling adaptation to offline demonstrations. When the fine-tuning data are already relatively compatible with the model, as the case for `ScaleQuest-Math`, standard SFT performs strongly and the additional benefit of pretrained-reference weighting is relatively smaller. For training distributions where SFT generalizes less effectively, PriFT provides substantially larger gains, particularly in Pass@16, suggesting that filtering or downweighting weakly supported supervision becomes more valuable. Nevertheless, the consistent improvements across all three training datasets show that PriFT is not tied to a particular fine-tuning distribution, while the varying gains are consistent with its motivation of reducing harmful adaptation to poorly supported offline targets.

E.5 Ablation study

We conduct several ablations to better understand the design choices in PriFT. The results are shown in Figure 7 and Table 12. Unless otherwise specified, experiments are conducted on `Qwen2.5-Math-1.5B`.

Replacing the pretrained reference with an EMA model. We test whether the frozen pretrained reference in PriFT-prob can be replaced by an exponential moving average (EMA) of the online model: $\boldsymbol{\theta}_{\text{ema}}^{(k)} \leftarrow \rho\boldsymbol{\theta}_{\text{ema}}^{(k-1)} + (1-\rho)\boldsymbol{\theta}_{\text{on}}^{(k)}$. Here, $\rho = 0$ recovers DFT, where token weights are computed from the current online model, while $\rho = 1$ recovers PriFT-prob, where the reference remains the original pretrained model. As shown in Figure 7(a), increasing the EMA momentum generally improves Pass@16, indicating that more stable token statistics are beneficial for weighted SFT. However, EMA references still do not outperform the pretrained-reference endpoint. This suggests that PriFT's advantage does not come only from reducing short-term fluctuations in the online model; it also depends on preserving the clean pretrained distribution before task-specific adaptation.

Varying the token selection threshold in PriFT-mass. We study the sensitivity of PriFT-mass to its selection threshold. Specifically, we replace the default threshold 0.5 in Equation 3 with $m_t = \mathbf{1}\left[u_t^{(\text{MASS})} > \tau\right]$. When $\tau = 0$, all tokens are selected and the objective reduces to standard SFT. When $\tau = 1$, no tokens are selected, so the resulting model remains the pretrained checkpoint. Figure 7(b) shows that performance improves from $\tau = 0$, reaches its best value around $\tau = 0.5$, and then

Table 10: Performance comparison between original and pretrained-weight variants on five mathematical reasoning benchmarks for `Qwen2.5-Math-1.5B` with 3 different baseline methods: EAFT, IDFT and TALR. We also report the average across all five benchmarks.

Method	Variant	MATH-OAI		Minerva Math		OlympiadBench		AIME24		AMC23		Average	
		Avg@16	P@16	Avg@16	P@16	Avg@16	P@16	Avg@16	P@16	Avg@16	P@16	Avg@16	P@16
EAFT	original	42.88	81.20	**12.41**	**40.81**	**12.79**	**45.78**	1.24	13.33	18.59	**70.00**	17.58	50.22
	pretrained-weight	**43.53**	**82.00**	11.40	**40.81**	12.36	45.63	**2.08**	**16.67**	**19.38**	**70.00**	**17.75**	**51.02**
IDFT	original	**66.11**	84.80	**22.21**	42.65	27.54	52.59	7.31	20.00	31.72	72.50	30.98	54.51
	pretrained-weight	63.44	**89.80**	18.92	**48.53**	**28.99**	**58.81**	**8.95**	**23.33**	**38.91**	**80.00**	**31.84**	**60.10**
TALR	original	**64.08**	89.00	**22.96**	44.85	27.73	55.70	6.26	20.00	38.75	77.50	**31.95**	57.41
	pretrained-weight	61.54	**91.80**	16.82	**48.53**	**27.89**	**58.81**	**7.08**	**23.33**	**40.31**	**87.50**	30.73	**62.00**

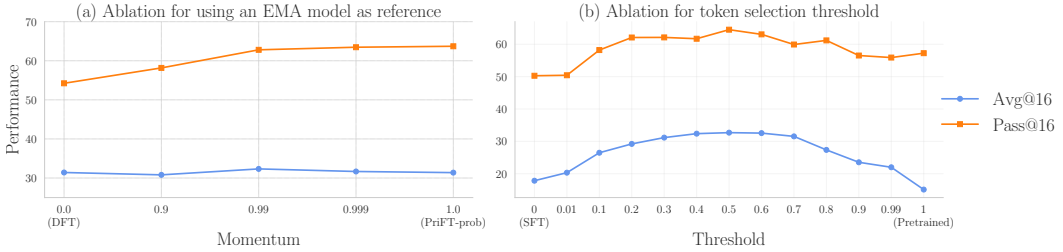

Figure 7: Ablation study on mathematical reasoning task with `Qwen2.5-Math-1.5B`. (a) Replace the pretrained reference in PriFT-prob with an EMA reference. (b) Sensitivity of PriFT-mass to the selection threshold.

declines as the selection becomes too strict. This supports the use of the default threshold: it filters out tokens with weak pretrained support while retaining enough supervised signal for effective adaptation.

Reweighting signal from a stronger model. We test whether PriFT-prob benefit from leveraging the reweighting signal of a stronger model. In this setting, the online model is fine-tuned normally, but the reweighting signal comes from either its own pretrained checkpoint or another pretrained model. As shown in Table 12, cross-model weighting can be beneficial when the reference provides more task-relevant prior knowledge. For `Qwen2.5-1.5B`, using the math-specialized Qwen2.5-Math-1.5B as the reweighting model improves both Avg@16 and Pass@16. However, this benefit is not uniform: for `Qwen2.5-Math-1.5B`, its own checkpoint outperforms using the larger `Qwen2.5-Math-7B` as the reference. These results suggest that external references can provide useful prior-support signals, but their effectiveness depends on both task relevance and compatibility with the model being fine-tuned.

Applying pretrained reweighting signals to existing baselines To test whether the benefit of pretrained statistics extends beyond PriFT, we modify existing token-reweighted SFT baselines by replacing their online weighting signal with statistics computed from the pretrained model. The weighting rule of each method is kept unchanged; only the source of the token statistics is changed. This gives pretrained-reweighting variants of EAFT [Diao et al., 2026], IDFT [Zhang et al., 2026b], and TALR [Lin et al., 2026]. Table 10 reports results on five mathematical reasoning benchmarks. Across all three methods, using pretrained statistics improves average Pass@16. These results show that replacing online statistics with pretrained statistics often improves the diversity and coverage of correct sampled trajectories, as reflected by Pass@16. Overall, this ablation supports the broader claim that the pretrained model provide a useful reweighting signal beyond the specific PriFT instantiations.

Connection to knowledge distillation (KD). We further analyze the relation between PriFT-prob and teacher-forced KD. Let q_t and p_t denote the teacher and student distributions at position t, respectively. The KD distribution-matching term can be decomposed into a target-token term and a non-target term: $-\sum_v q_t(v)\log p_t(v) = -q_t(y_t)\log p_t(y_t) - \sum_{v\neq y_t} q_t(v)\log p_t(v)$. The target term has the same form as PriFT-prob: the reference model reweights the gold token by its probability, while the non-target term matches the teacher's probabilities on other tokens.

To test whether the non-target term provides useful guidance, we train `Qwen2.5-Math-1.5B` with `Qwen2.5-Math-7B` as the teacher and introduce a coefficient β: $\mathcal{L}_\beta = -q_t(y_t)\log p_t(y_t) -$

Table 11: Effect of the non-target term in decomposed teacher-forced KD. We use `Qwen2.5-Math-7B` as the teacher and `Qwen2.5-Math-1.5B` as the student. β controls the strength of matching the teacher's non-target distribution.

Non-target term weight β	0.00	0.25	0.50	0.75	1.00
Avg@16	**30.44**	21.85	17.07	13.56	11.80
Pass@16	**59.93**	58.74	54.92	52.17	48.64

Table 12: Effect of using cross-model token weighting signals in PriFT-prob.

Model	**Token reweighting model**	**Avg@16**	**Pass@16**
Qwen2.5-Math 1.5B	Qwen2.5-Math-1.5B	**31.38**	**63.71**
	Qwen2.5-Math-7B	30.44	59.93
Qwen2.5 1.5B	Qwen2.5-1.5B	15.88	46.13
	Qwen2.5-Math-1.5B	**17.02**	**47.33**

$\beta \sum_{v \neq y_t} q_t(v) \log p_t(v)$. Here, $\beta = 0$ recovers PriFT-prob using the teacher model as reference, while $\beta = 1$ recovers the full teacher-forced KD distribution-matching term. This is a diagnostic decomposition of the KD term, not the standard KD setting with an additional hard-label CE loss or temperature scaling for the teacher distribution.

As shown in Table 11, performance decreases monotonically as β increases, with the best result at $\beta = 0$, which recovers PriFT-prob using the teacher model as reference. We note that result should not be interpreted as a comparison against standard KD. Rather, it isolates the KD distribution-matching term without an additional hard-label CE loss, and suggests that, in this controlled setting, useful teacher guidance mainly comes from target-label reweighting rather than matching the non-target distribution.

E.6 Results on medical question answering

Experimental setup. For medical question answering, we follow the setup of Zhu et al. [2026] in general. We fine-tune `LLaMA-2-7B` [Touvron et al., 2023] on 100k examples from `MedMCQA` [Pal et al., 2022] for 1 epoch. We evaluate on `MedQA` [Jin et al., 2021], `MMLU-medical` [Hendrycks et al., 2020], and the `MedMCQA` test set. The model is trained for one epoch with a maximum sequence length of 512, a global batch size of 64, and a learning rate of 2×10^{-5}. Evaluation uses standard multiple-choice prompt templates, and we report accuracy.

Experimental results Table 13 reports results on medical question answering with `LLaMA-2-7B`. PriFT-mass achieves the best performance on MMLU and MedMCQA, with 49.14 and 39.78 accuracy, respectively. It also obtains the second-best average accuracy of 42.03, closely matching ASFT, which achieves 42.40 on average. Compared with other token-reweighted SFT baselines, PriFT-mass gives the strongest overall performance together with ASFT.

PriFT-prob also performs competitively, achieving the third-best average accuracy among all methods, over 6 points higher than DFT. The gap between PriFT-prob and PriFT-mass is consistent with our motivation for PriFT-mass: raw pretrained probability can be overly restrictive in knowledge-intensive domains, where important target tokens may have low initial probability. Overall, these results show that PriFT generalizes beyond mathematical reasoning and provides an effective pretrained-reference weighting signal for medical question answering.

E.7 Results on mathematical reasoning: additional models

Beyond the main results in Table 3, we further evaluate PriFT on four additional mathematical reasoning backbones: `Qwen2.5-Math-1.5B`, `Qwen2.5-1.5B`, `Qwen2.5-Instruct-1.5B`, and `DeepSeekMath-7B` [Shao et al., 2024]. The results are shown in Table 14. Due to computational constraints, we include the most representative baseline from the main experiments rather than exhaustively evaluating all token-reweighted methods on every additional backbone.

Table 13: Performance on medical question answering benchmarks using `LLaMA-2-7B` fine-tuned on MedMCQA. We report accuracy on MedQA, MMLU-Medical, MedMCQA, and their average.

Methods	MedQA	MMLU	MedMCQA	Avg.
Pretrained	29.85	30.52	33.76	31.38
SFT	31.42	33.48	35.67	33.52
DFT	32.99	31.00	30.79	31.59
EAFT	32.29	30.34	31.17	31.27
IDFT	32.44	38.23	35.12	35.26
TALR	31.81	41.04	34.35	35.73
ASFT	**40.93**	46.99	39.28	**42.40**
PriFT-prob	35.82	41.51	37.01	38.11
PriFT-mass	37.16	**49.14**	**39.78**	42.03

Table 14: Performance comparison on five mathematical reasoning benchmarks. We also report the average across all five benchmarks.

Model	Method	MATH-OAI		Minerva Math		OlympiadBench		AIME24		AMC23		Average	
		Avg@16	P@16	Avg@16	P@16	Avg@16	P@16	Avg@16	P@16	Avg@16	P@16	Avg@16	P@16
Qwen2.5-Math (1.5B)	Original	30.19	84.20	8.56	39.71	15.82	53.19	4.99	26.67	15.94	82.50	15.10	57.25
	SFT	43.38	82.00	12.63	41.91	12.77	44.15	1.04	13.33	19.38	70.00	17.84	50.28
	DFT	64.21	87.20	21.74	44.12	27.35	53.19	5.63	16.67	38.13	70.00	31.41	54.23
	EAFT	42.88	81.20	12.41	40.81	12.79	45.78	1.24	13.33	18.59	70.00	17.58	50.22
	IDFT	**66.11**	84.80	22.21	42.65	27.54	52.59	7.31	20.00	31.72	72.50	30.98	54.51
	TALR	64.08	89.00	**22.96**	44.85	27.73	55.70	6.26	20.00	38.75	77.50	31.95	57.41
	ASFT	58.89	88.00	16.99	46.69	25.42	58.07	4.79	23.33	32.97	80.00	27.81	59.22
	PriFT-prob	63.06	**91.40**	18.17	**48.90**	28.44	**60.74**	**8.33**	30.00	38.91	87.50	31.38	63.71
	PriFT-mass	62.35	90.60	19.09	47.79	**29.06**	58.37	6.25	**33.33**	**46.72**	**92.50**	**32.69**	**64.52**
Qwen2.5 (1.5B)	Pretrained	4.04	36.40	1.61	16.54	1.67	17.78	0.21	3.33	2.50	22.50	2.00	19.31
	SFT	25.59	68.60	4.15	24.26	5.88	32.44	0.42	6.67	7.81	47.50	8.77	35.90
	DFT	**46.19**	74.00	**12.49**	33.46	**13.89**	38.37	1.86	13.33	17.97	62.50	**18.48**	44.33
	PriFT-prob	41.54	78.40	9.01	**34.19**	12.76	42.22	1.24	13.33	14.84	62.50	15.88	46.13
	PriFT-mass	43.33	**79.00**	11.04	33.46	13.49	**43.85**	**3.33**	**16.67**	**18.59**	**65.00**	17.96	**47.59**
Qwen2.5-Instruct (1.5B)	Pretrained	45.89	**83.60**	12.06	37.13	14.31	46.22	1.66	13.33	18.59	67.50	18.50	49.56
	SFT	25.73	69.20	4.61	25.74	6.22	33.33	0.41	6.67	8.44	62.50	9.08	39.49
	DFT	45.55	75.00	12.37	30.15	14.30	40.44	1.66	13.33	20.94	60.00	18.96	43.78
	PriFT-prob	49.53	**83.60**	14.59	40.07	16.82	**47.26**	2.28	**16.67**	25.47	**75.00**	21.74	**52.52**
	PriFT-mass	**52.78**	83.40	**16.84**	**40.81**	**18.81**	46.67	**2.49**	**16.67**	**27.81**	67.50	**23.75**	51.01
DeepSeekMath (7B)	Pretrained	37.78	76.80	17.94	49.63	10.33	41.63	0.41	6.67	14.38	62.50	16.17	47.45
	SFT	32.91	74.40	11.14	40.07	7.59	38.81	0.41	6.67	12.03	55.00	12.82	42.99
	DFT	**46.65**	76.60	17.40	37.50	**16.07**	41.63	**2.28**	**13.33**	20.47	57.50	**20.57**	45.31
	PriFT-prob	44.06	**78.80**	19.14	49.26	14.46	**46.96**	0.62	6.67	19.69	**65.00**	19.59	**49.34**
	PriFT-mass	43.34	77.40	**20.51**	**50.00**	14.38	43.70	0.62	10.00	**21.72**	60.00	20.11	48.22

Overall, PriFT continues to provide strong performance across model families and scales. On `Qwen2.5-Math-1.5B`, PriFT-mass achieves the best average performance on both Avg@16 and Pass@16, outperforming SFT and prior token-reweighted baselines. On `Qwen2.5-Instruct-1.5B` and `DeepSeekMath-7B`, PriFT also achieves the strongest aggregate Avg@16 or Pass@16 among fine-tuning methods, showing that pretrained-reference weighting is effective beyond the main 7B–8B settings.

Taken together, these additional experiments support the robustness of PriFT across different pretrained, math-specialized, and instruction-tuned backbones.

E.8 Additional results on code generation

Table 15 reports additional code generation results on `Qwen2.5-Coder-3B`. PriFT-mass achieves the best average performance and is the only fine-tuning method that improves over the original checkpoint on average. The gains again concentrate on LiveCodeBench, while PriFT-prob is less effective on this specialized coding model, suggesting that relative support in PriFT-mass can be more robust than raw pretrained probability.

Table 15: Additional code generation results on `Qwen2.5-Coder-3B`. We report pass@1 accuracy and the average across all four benchmarks.

Method	HumanEval+	MBPP+	LCB v5	LCB v6	Avg
Original	**78.66**	63.49	21.25	20.47	45.97
SFT	70.12	**66.67**	17.05	16.02	42.46
DFT	70.12	65.34	12.73	12.13	40.08
IDFT	71.34	65.87	16.36	15.45	42.26
EAFT	73.17	64.81	17.61	16.59	43.05
TALR	70.12	65.34	12.84	11.85	40.04
ASFT	73.78	66.40	20.80	20.09	45.27
PriFT-prob	75.00	62.43	17.73	16.78	42.98
PriFT-mass	77.44	64.29	**22.27**	**21.14**	**46.28**

F Additional information for RL experiments

For mathematical reasoning, we perform DAPO reinforcement learning following [Yu et al., 2025] with the `verl` recipe and GRPO advantage estimation [Shao et al., 2024]. We train on 17,398 deduplicated prompts from `dapo-math-17k-dedup`, using rule-based rewards that assign $+1$ to correct boxed answers and -1 to incorrect ones, with an additional penalty for overlong responses. Experiments are conducted on `Qwen2.5-Math-1.5B` and `Qwen3-8B-Base` using 4 H200 GPUs for 100 RL steps, with prompt and generation batch sizes of 128, a PPO mini-batch size of 32, one PPO epoch, learning rate 10^{-6}, 10 warmup steps, weight decay 0.1, gradient clipping 1.0, temperature 1.0, top-p 1.0, and an actor KL loss coefficient of 0.01. During RL rollouts, we sample 32 responses per prompt for `Qwen2.5-Math-1.5B` and 16 for `Qwen3-8B-Base`.

G Compute resources

All experiments are conducted on NVIDIA H100 or H200 GPUs. Each training run uses 4 GPUs with data-parallel training, including the mathematical reasoning, code generation, medical question answering, ablation, and RL-initialization experiments.

For mathematical reasoning SFT, a `Qwen2.5-Math-1.5B` run takes approximately 1 hour on 4 H200 GPUs, corresponding to about 4 GPU-hours. A `Qwen2.5-Math-7B` run takes approximately 2 hours on 4 H200 GPUs, corresponding to about 8 GPU-hours. These estimates include the additional reference forward pass used by PriFT to precompute token weights. Standard 16-sample mathematical reasoning evaluation takes around 20 minutes when parallelized over 4 GPUs.

For RL-initialization experiments, each DAPO run takes approximately 7 hours on 4 GPUs. Due to limited computational budget, we used the same fixed computational resources for all RL experiments rather than training each model to full convergence.

H Social Impacts

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We justified our main claims in the Experiments section in our main paper, with extensive additional experimental results in the appendix.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discussed the limitations of the work in the Section 7.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: This work does not have theoretical results that require derivation or proof.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provided sufficient information for reproduction in the appendix. We have also released the code to enable reproduction.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We have released the full data and code, with sufficient instructions for reproduction.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provided sufficient experimental details in the main paper and the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Reporting error bars is uncommon for the experiments in this field. Instead, we reported results averaged over multiple sampled responses and multiple benchmarks, as well as results with different random seeds.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provided sufficient information on the GPUs used for our experiments in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We have confirmed that this work conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses that improving supervised fine-tuning may make LLM adaptation more efficient and accessible, while also noting that stronger fine-tuned models can inherit or amplify risks such as misuse, overreliance, and errors in high-stakes domains such as medical QA.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This work does not produce new dataset, and the trained models will not be released.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have properly cited and made sure we followed the license of the models and data.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: This work does not produce new dataset, and the trained models will not be released.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.