Full-Distribution Self-Distillation for Optimizing Soft Latent Reasoning

Anonymous ACL submission

Abstract

Soft latent reasoning extends chain-of-thought reasoning beyond discrete token sequences by using continuous concept tokens. However, effectively training models in this space remains challenging: conventional reinforcement learning methods mainly optimize sampled rollout-action likelihoods and therefore underuse the full distributions produced during soft rollouts. We propose **Full-Distribution Self-Distillation (FDSD)**, an off-policy training framework that treats stochastic soft rollout distributions as dense pseudo-labels. FDSD aligns the current model distribution with advantage-weighted rollout targets via KL minimization and importance correction, enabling the model to learn from the full rollout distribution rather than only from sampled-action likelihoods. Experiments on GSM8K, MATH-500, OlympiadBench, and out-of-domain benchmarks show that FDSD consistently improves mathematical reasoning performance while preserving generalization, with only modest computational overhead. This suggest that full-distribution alignment is an effective training direction for soft latent reasoning.

1 Introduction

Large language models (LLMs) have demonstrated strong reasoning abilities across tasks such as mathematical problem solving and code generation (Chen et al., 2025; Hou et al., 2025; Nezhad et al., 2025). A key technique behind these advances is chain-of-thought (CoT) reasoning, which elicits intermediate reasoning steps in natural language (Wei et al., 2022; Feng et al., 2023; Dutta et al., 2024). However, CoT confines reasoning to discrete token sequences. This discrete linguistic bottleneck may limit the model's ability to represent abstract intermediate states, explore multiple reasoning paths, and express computations that are not naturally aligned with human language (Zhang et al., 2025).

Recent work has therefore explored reasoning in continuous latent spaces. In particular, the Soft Thinking paradigm replaces discrete intermediate tokens with continuous concept tokens, typically constructed as probability-weighted combinations of vocabulary embeddings (Zhang et al., 2026). Such representations allow the model to maintain soft superpositions of multiple reasoning directions rather than committing to a single token at each step. We refer to this setting as *soft latent reasoning*. Theoretically, continuous thoughts can support parallel exploration and solve graph reachability more efficiently than discrete CoT (Zhu et al., 2026). Empirically, training-free soft latent reasoning methods have shown promising gains in accuracy and token efficiency (Wu et al., 2025; Zhuang et al., 2025).

Despite these advances, effectively training models for soft latent reasoning remains challenging. Most existing Soft Thinking methods are training-free and rely on the latent capabilities already present in pretrained models. Reinforcement learning offers a natural way to improve reasoning by favoring high-reward trajectories, but conventional policy-gradient methods are mainly designed for sampled discrete actions (Schulman et al., 2017; Shao et al., 2024). In soft latent reasoning, each rollout step produces a full distribution over the vocabulary manifold. Optimizing only the likelihood of the sampled rollout action discards rich distributional information and can lead to high-variance updates (Sutton et al., 1999; Butt et al., 2025).

This challenge is closely related to distribution matching in policy learning. Existing methods typically construct target distributions from stronger teacher models (Agarwal et al., 2024), privileged supervision (Zhao et al., 2026), or reward-improved policies (Peters et al., 2010; Abdolmaleki et al., 2018; Song et al., 2019), and then update the current policy toward these targets under divergence constraints. However, these methods are primarily

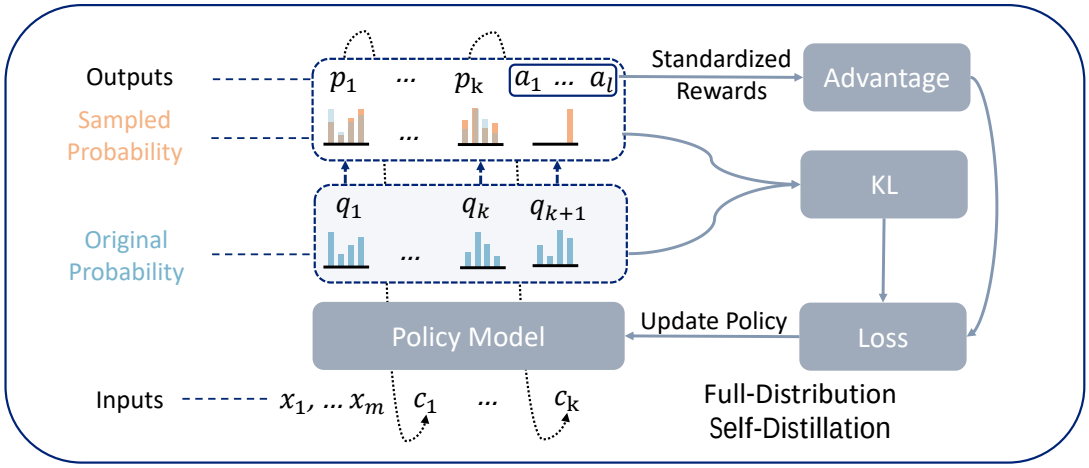

Figure 1: Overview of the proposed **Full-Distribution Self-Distillation** framework, adapted from Wu et al. (2025). Given an input sequence, the policy model produces the original token distribution q_t at each reasoning step. A stochastic soft rollout strategy generates an exploratory distribution p_t, which is used to construct continuous concept tokens c_t for subsequent reasoning. After rollout, standardized rewards provide advantages that weight the KL-based distillation loss between the exploratory rollout distribution and the current model distribution. The policy is then updated by aligning its full output distribution with high-reward soft rollout targets.

designed for sampled-action updates, rather than the full soft distributions naturally produced by soft latent reasoning rollouts.

In this work, we propose **Full-Distribution Self-Distillation (FDSD)**, an off-policy training framework that optimizes soft latent reasoning through advantage-weighted distribution matching. As illustrated in Figure 1, stochastic soft rollout strategies generate exploratory distributions over the vocabulary manifold during rollout. FDSD treats these distributions as soft pseudo-labels and distills high-reward rollout distributions back into the current model, aligning the model's output with the full rollout distribution rather than only optimizing the likelihood of the sampled rollout action. This provides denser supervision, reduces the variance of sample-based updates, and better matches the continuous nature of soft latent reasoning.

Our contributions are summarized as follows:

1. We formulate soft latent reasoning optimization as an advantage-weighted self-distillation problem over full rollout distributions.

2. We propose **FDSD**, a full-distribution self-distillation objective that provides dense policy supervision from exploratory soft rollout distributions.

3. We empirically demonstrate that FDSD improves mathematical reasoning performance and maintains strong out-of-domain robustness compared with standard RL baselines.

2 Related Work

2.1 Latent Reasoning

Latent reasoning aims to move intermediate reasoning beyond discrete token sequences by using hidden states, continuous concept tokens, or vocabulary-space superpositions. Theoretical work shows that continuous chain-of-thought can support parallel exploration and solve certain reasoning problems more efficiently than discrete CoT (Zhu et al., 2026). This provides a theoretical basis for representing intermediate reasoning states in latent spaces rather than committing to a single natural-language token at each step.

Training-free methods introduce such representations at inference time without updating model parameters. Soft Thinking (Zhang et al., 2026) constructs concept tokens from probability-weighted vocabulary embeddings, allowing the model to preserve uncertainty over multiple reasoning directions. However, this approach may suffer from the "Greedy Pitfall", which motivates Gumbel-Softmax randomization for better exploration (Wu et al., 2025). MoI (Zhuang et al., 2025) further combines sampled observations with the model distribution through Bayesian estimation, preserving uncertainty while maintaining low inference overhead.

Other work trains models to perform latent reasoning. Supervised methods including COCONUT (Hao et al., 2024), CODI (Shen et al.,

2025), and Latent-SFT (Deng et al., 2025) learn latent representations from CoT demonstrations or align latent and token spaces through distillation. Although effective, these methods still depend on discrete reasoning traces, which may limit their performance ceiling.

RL-based methods such as SOFT-TOKEN (Butt et al., 2025) and HRPO (Yue et al., 2026) attempt to learn latent reasoning through exploration. However, their policy updates still primarily optimize the likelihood of sampled rollout actions, and therefore do not fully exploit the soft distributions produced during latent reasoning. These limitations motivate more efficient training strategies that can directly learn from soft rollout distributions.

2.2 Distribution Matching

Our work is also related to distribution matching in policy learning. Instead of updating a policy only through sampled actions, distribution-matching methods construct target distributions from teacher policies, reward-improved policies, or advantage-weighted samples, and then optimize the current policy toward these targets.

On-policy distillation reduces offline distribution mismatch by training on student-generated trajectories while still relying on a stronger teacher model (Agarwal et al., 2024). On-policy self-distillation removes the external teacher, yet its supervision often depends on privileged information such as ground-truth answers or verified reasoning traces (Zhao et al., 2026). Self-Distilled RLVR uses verifiable rewards to guide self-distillation, but it is still limited to tasks with reliable verification signals and remains centered on discrete token trajectories (Yang et al., 2026).

KL-constrained policy optimization methods, including REPS (Peters et al., 2010), MPO (Abdolmaleki et al., 2018), and V-MPO (Song et al., 2019), stabilize policy learning by constructing reward-improved policies and projecting the current policy toward them under divergence constraints. These methods provide a useful perspective on policy improvement as distribution matching. However, they are not designed for soft latent reasoning, where each rollout step naturally produces a dense distribution over the vocabulary manifold.

FDSD differs by treating stochastic soft rollout distributions as dense pseudo-labels and directly aligning the model distribution with these advantage-weighted targets, without requiring external teachers or privileged reasoning traces.

3 Preliminaries

3.1 Standard Discrete Token Generation

Let $\mathcal{V} = \{v_1, \ldots, v_{|\mathcal{V}|}\}$ denote the vocabulary and $\mathbf{E} \in \mathbb{R}^{|\mathcal{V}| \times d}$ the token embedding matrix, where $\mathbf{e}_j$ is the embedding of token v_j. Given an input sequence $\mathbf{x}_{\leq t} = [x_1, \ldots, x_t]$, an autoregressive LLM produces a hidden state $\mathbf{h}_t$ and projects it to vocabulary logits $\mathbf{z}_t \in \mathbb{R}^{|\mathcal{V}|}$. The next-token distribution is

$$q_{t,j} = P(w_{t+1} = v_j \mid \mathbf{x}_{\leq t}) = \frac{\exp(z_{t,j})}{\sum_{k=1}^{|\mathcal{V}|} \exp(z_{t,k})}. \tag{1}$$

Standard generation samples a single token from $\mathbf{q}_t$ and feeds its embedding into the next step. This discrete commitment collapses the model's uncertainty into one token, discarding alternative reasoning directions encoded in the remaining probability mass.

3.2 Soft Thinking in Continuous Concept Space

Soft Thinking (Zhang et al., 2026) relaxes this discrete commitment by using continuous concept tokens for intermediate reasoning. Instead of feeding the embedding of one sampled token, it constructs a weighted combination of vocabulary embeddings:

$$\mathbf{c}_{t+1} = \sum_{j=1}^{|\mathcal{V}|} p_{t,j} \mathbf{e}_j = \mathbf{p}_t^\top \mathbf{E}, \tag{2}$$

where $\mathbf{p}_t \in \Delta^{|\mathcal{V}|}$ is a probability distribution over the vocabulary. $\mathbf{E}$ is the embedding matrix. The resulting vector $\mathbf{c}_{t+1}$ is termed a soft concept token. Since it is a convex combination of token embeddings, it remains on the vocabulary embedding manifold while preserving multiple possible reasoning directions.

The model continues reasoning conditioned on previous soft concept tokens and generates new soft concept tokens, and then finally returns to standard discrete generation to produce the answer after a termination token reaches the highest probability.

In the training-free setting, one can directly set $\mathbf{p}_t = \mathbf{q}_t$. However, this deterministic construction provides limited exploration and is not directly suited for reinforcement-learning-style optimization, which relies on stochastic trajectories and reward comparison.

3.3 Stochastic Rollouts

To enable exploration in continuous concept space, a stochastic rollout strategy can be used to generate a soft distribution $\mathbf{p}_t$ from the model distribution $\mathbf{q}_t$:

$$\mathbf{p}_t \sim \pi(\cdot \mid \mathbf{q}_t). \tag{3}$$

Here, $\mathbf{q}_t$ is the original distribution produced by the model, while $\mathbf{p}_t$ is the rollout distribution used to construct the next concept token.

Different rollout strategies instantiate different forms of $\pi(\mathbf{p}_t \mid \mathbf{q}_t)$, such as Gumbel-Softmax sampling (Wu et al., 2025), Dirichlet sampling, Gaussian sampling (Butt et al., 2025), and Mixture of Inputs (Zhuang et al., 2025). Their detailed sampling procedures and probability computations are provided in Appendix B.

A rollout therefore produces a trajectory of soft distributions and concept tokens. After the final response is evaluated by a reward function, the trajectory-level reward or advantage provides the supervision signal used for optimization. The next section introduces how these stochastic rollout distributions are used for full-distribution self-distillation.

4 Methodology

We propose **Full-Distribution Self-Distillation (FDSD)** for optimizing soft latent reasoning. Given trajectories generated by an old rollout policy $\pi_{\theta_{\text{old}}}$, FDSD treats the rollout distribution $\mathbf{p}_t$ at each reasoning step as a soft training target. During update, we replay the same soft concept-token trajectory and evaluate the current model distribution $\mathbf{q}_{\theta,t}$ at the corresponding reasoning step. This gives an off-policy distillation setting, where targets are generated by $\pi_{\theta_{\text{old}}}$ while the current policy π_θ is optimized toward them.

4.1 Advantage-Weighted Self-Distillation

Let $\mathbf{q}_{\theta,t}$ denote the current model distribution at reasoning step t, and let $\mathbf{p}_t$ denote the rollout distribution generated during exploration. Instead of optimizing only the likelihood of the sampled rollout action, FDSD aligns $\mathbf{q}_{\theta,t}$ with the full rollout distribution $\mathbf{p}_t$ by minimizing an advantage-weighted KL divergence:

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{p}_t \sim \pi_{\theta_{\text{old}}}} \left[A_t \cdot \omega_t(\theta) \cdot \mathbb{D}_{\text{KL}} \left(\mathbf{p}_t \,\|\, \mathbf{q}_{\theta,t} \right) \right], \tag{4}$$

where

$$\omega_t(\theta) = \frac{\pi_{\theta,t}}{\pi_{\theta_{\text{old}},t}} \tag{5}$$

is the importance sampling ratio between the current policy and the rollout policy. Here, $\pi_{\theta,t}$ denotes the probability of generating the rollout distribution $\mathbf{p}_t$ under the current model, and $\pi_{\theta_{\text{old}},t}$ denotes the corresponding probability under the old rollout policy.

Expanding the KL divergence gives

$$\begin{aligned} \mathbb{D}_{\text{KL}} \left(\mathbf{p}_t \,\|\, \mathbf{q}_{\theta,t} \right) &= \sum_{j=1}^{|\mathcal{V}|} p_{t,j} \log \frac{p_{t,j}}{q_{\theta,t,j}} \\ &= \sum_{j=1}^{|\mathcal{V}|} (p_{t,j} \log p_{t,j} - p_{t,j} \log q_{\theta,t,j}). \end{aligned} \tag{6}$$

Since the entropy term of $\mathbf{p}_t$ is independent of θ, Eq. (6) is equivalent to minimizing the advantage-weighted cross-entropy:

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{p}_t \sim \pi_{\theta_{\text{old}}}} \left[-A_t \cdot \omega_t(\theta) \sum_{j=1}^{|\mathcal{V}|} p_{t,j} \log q_{\theta,t,j} \right] \tag{7}$$

Thus, $\mathbf{p}_t$ acts as a full-distribution pseudo-label, allowing FDSD to preserve the distributional information discovered during stochastic rollout.

4.2 Practical Implementation

In our main experiment, the rollout distribution is generated by Mixture of Inputs (MoI). At step t, we first restrict computation to the top-K token set S_t and normalize the model distribution over this set. MoI then samples a discrete observation

$$i \sim \text{Categorical}(\mathbf{q}_t), \qquad y_{t,j} = \mathbb{I}(j = i). \tag{8}$$

Let

$$H_t = -\frac{1}{\log K} \sum_{j \in S_t} q_{t,j} \log q_{t,j} \tag{9}$$

be the normalized entropy of $\mathbf{q}_t$ over S_t. MoI constructs the soft rollout distribution as

$$p_{t,j} = \frac{H_t q_{t,j} + (\beta + 1 - H_t) y_{t,j}}{\beta + 1}, \qquad j \in S_t. \tag{10}$$

This distribution preserves the uncertainty in $\mathbf{q}_t$ while incorporating the sampled observation. Since $\mathbf{p}_t$ is deterministically determined by $\mathbf{q}_t$ and the sampled index i, the rollout probability reduces to

$$\pi_{\theta,t} = \text{Prob}(\mathbf{p}_t \mid \mathbf{q}_{\theta,t}) = q_{\theta,t,i}. \tag{11}$$

This makes MoI suitable for off-policy FDSD, because the importance ratio can be computed without estimating an unstable continuous probability density.

For the update, we apply top-K distillation and PPO-style clipping to the importance weight:

$$\bar{\omega}_t(\theta) = \text{clip}\left(\omega_t(\theta), 1-\epsilon, 1+\epsilon\right), \tag{12}$$

and define the clipped advantage coefficient as

$$\tilde{A}_t(\theta) = \min\left(\omega_t(\theta)A_t, \bar{\omega}_t(\theta)A_t\right). \tag{13}$$

In implementation, we stop the gradient through $\tilde{A}_t(\theta)$ and use it only as a scalar weight. The practical objective is

$$\mathcal{L}^{\text{clip}}(\theta) = \mathbb{E}\left[-\frac{1}{T}\sum_{t=1}^{T}\tilde{A}_t(\theta)\sum_{j\in S_t} p_{t,j}\log q_{\theta,t,j}\right]. \tag{14}$$

4.3 Gradient Analysis

To understand the update direction, we examine the gradient with respect to the current model logit $z_{t,j}$. Treating the importance ratio as a stop-gradient coefficient, the gradient of Eq. (7) is

$$\nabla_{z_{t,j}}\mathcal{L}(\theta) = A_t \cdot \omega_t(\theta)\left(q_{\theta,t,j} - p_{t,j}\right). \tag{15}$$

This gradient has a direct distribution-matching interpretation. When $A_t > 0$, the model distribution is pulled toward the rollout distribution; when $A_t < 0$, the update direction is reversed.

Therefore, FDSD provides a dense token-level learning signal over the full rollout distribution, rather than only increasing the likelihood of the sampled rollout action.

4.4 Connection to GRPO

FDSD reduces to standard sample-based policy optimization when the rollout distribution degenerates into a one-hot distribution. Suppose the sampled token index is i, and the rollout target becomes

$$p_{t,j} = \mathbb{I}(j = i). \tag{16}$$

In this case, the probability of generating the rollout target is simply the probability of sampling token i, namely

$$\pi_{\theta,t} = q_{\theta,t,i}, \qquad \pi_{\theta_{\text{old}},t} = q_{\theta_{\text{old}},t,i}. \tag{17}$$

Substituting Eq. (16) into Eq. (15), we obtain

$$\nabla_{z_{t,j}}\mathcal{L}(\theta) = A_t \cdot \frac{q_{\theta,t,i}}{q_{\theta_{\text{old}},t,i}}\left(q_{\theta,t,j} - \mathbb{I}(j=i)\right), \tag{18}$$

which matches the update direction of GRPO. Thus, FDSD can be viewed as a full-distribution generalization of GRPO: it recovers the discrete action-likelihood update for one-hot rollouts and distills the entire distribution for soft rollouts.

5 Experiments

5.1 Setup

Datasets and Base Models. Similar to Butt et al. (2025), we conduct evaluations across a suite of Qwen models, spanning the Qwen2.5 series (1.5B, 3B, and 7B Instruct variants) (Team, 2024) and Qwen3-4B-Base (Team, 2025), on three mathematical reasoning benchmarks: GSM8K (Cobbe et al., 2021), MATH-500 (Hendrycks et al., 2021), and OlympiadBench (He et al., 2024). Specifically, MATH-500 constitutes a 500-question subset of the MATH test set, while OlympiadBench is a curated collection of 675 problems that provide definitive final answers and explicitly exclude figure-based content.

For model training, we adopt a tiered approach based on problem difficulty. The GSM8K benchmark, which consists of elementary-level math problems, is used with its standard training split. In contrast, for the more challenging MATH-500 and OlympiadBench benchmarks, models are trained on the training split of the MATH dataset to enhance their ability to handle advanced mathematical reasoning.

To assess out-of-domain (OOD) generalization, we train models on the MATH training split. We then evaluate them on HellaSwag (Zellers et al., 2019), ARC-Challenge (Clark et al., 2018), and MMLU (Hendrycks et al.), using accuracy as the evaluation metric.

Baselines and Settings. In our experiments, we evaluate five primary configurations: (1) standard GRPO training, (2) HRPO (Yue et al., 2026), (3) SOFT-TOKEN (Butt et al., 2025), (4) SOFT-INFER, which applies soft latent reasoning during inference directly on the GRPO-trained model without additional training, and (5) our proposed SOFT-FDSD.

In SOFT-FDSD, we use MoI as the default rollout strategy, as described in Section 4.2, because it yields a simple categorical rollout probability and a stable importance ratio.

We reproduce HRPO and SOFT-TOKEN under our training and evaluation pipeline, and align their key hyperparameters with ours as much as possible, including prompt templates, batch sizes, rollout settings, and training epochs. This allows a more controlled comparison among different latent-reasoning training strategies.

Each experimental configuration is evaluated

Model	Method	GSM8K		MATH-500		OlympiadBench	
		Greedy pass@1	Sample pass@1	Greedy pass@1	Sample pass@1	Greedy pass@1	Sample pass@1
Qwen2.5-1.5B-Instruct	GRPO	76.3 ± 0.5	75.1 ± 0.5	54.2 ± 0.4	53.7 ± 0.3	18.9 ± 1.0	18.8 ± 0.2
	HRPO	72.2 ± 0.8	72.2 ± 0.3	53.8 ± 0.7	53.0 ± 0.7	15.0 ± 0.7	14.4 ± 1.0
	SOFT-TOKEN	73.1 ± 1.0	72.3 ± 1.2	53.3 ± 0.5	52.3 ± 0.4	16.9 ± 0.2	16.4 ± 0.5
	SOFT-INFER	76.0 ± 0.3	75.0 ± 0.4	53.9 ± 0.5	54.4 ± 0.2	19.6 ± 0.6	18.8 ± 0.2
	SOFT-FDSD	$\mathbf{77.2 \pm 0.4}$	$\mathbf{76.3 \pm 0.2}$	$\mathbf{56.6 \pm 0.2}$	$\mathbf{54.7 \pm 0.2}$	$\mathbf{19.6 \pm 0.5}$	$\mathbf{19.2 \pm 0.2}$
Qwen2.5-3B-Instruct	GRPO	85.2 ± 0.4	84.9 ± 0.2	60.3 ± 0.9	62.4 ± 0.3	$\mathbf{25.8 \pm 0.8}$	25.1 ± 0.2
	HRPO	84.8 ± 1.2	84.0 ± 0.8	61.0 ± 0.9	61.0 ± 0.6	21.0 ± 1.0	20.0 ± 0.7
	SOFT-TOKEN	84.2 ± 0.8	82.8 ± 1.0	62.3 ± 0.4	62.5 ± 0.2	25.1 ± 0.6	25.2 ± 0.4
	SOFT-INFER	85.5 ± 0.5	84.5 ± 0.2	60.7 ± 0.5	62.5 ± 0.2	25.5 ± 0.6	25.4 ± 0.3
	SOFT-FDSD	$\mathbf{86.1 \pm 0.5}$	$\mathbf{85.7 \pm 0.2}$	$\mathbf{63.4 \pm 1.0}$	$\mathbf{63.4 \pm 0.3}$	25.1 ± 0.3	$\mathbf{25.5 \pm 0.4}$
Qwen2.5-7B-Instruct	GRPO	91.4 ± 0.4	91.2 ± 0.4	70.8 ± 0.6	71.2 ± 0.4	34.9 ± 0.5	33.8 ± 0.4
	SOFT-TOKEN	91.5 ± 0.8	91.3 ± 0.4	71.6 ± 1.0	71.4 ± 0.3	32.3 ± 0.5	33.3 ± 0.2
	SOFT-INFER	91.3 ± 0.8	91.4 ± 0.5	72.1 ± 0.7	71.7 ± 0.3	35.0 ± 0.5	33.9 ± 0.4
	SOFT-FDSD	$\mathbf{92.0 \pm 0.4}$	$\mathbf{91.8 \pm 0.3}$	$\mathbf{72.9 \pm 0.5}$	$\mathbf{71.8 \pm 0.3}$	$\mathbf{35.3 \pm 0.3}$	$\mathbf{34.6 \pm 0.2}$
Qwen3-4B-Base	GRPO	91.8 ± 0.6	91.0 ± 0.3	72.6 ± 0.8	73.0 ± 0.3	33.4 ± 0.9	$\mathbf{34.4 \pm 0.4}$
	SOFT-TOKEN	89.5 ± 0.9	89.1 ± 0.7	69.3 ± 0.4	69.1 ± 0.6	30.2 ± 0.5	29.5 ± 0.4
	SOFT-INFER	91.5 ± 0.5	90.9 ± 0.3	72.6 ± 0.6	73.0 ± 0.2	33.5 ± 0.4	34.3 ± 0.3
	SOFT-FDSD	$\mathbf{92.1 \pm 0.5}$	$\mathbf{91.3 \pm 0.3}$	$\mathbf{73.0 \pm 0.5}$	$\mathbf{73.6 \pm 0.3}$	$\mathbf{33.8 \pm 0.6}$	34.2 ± 0.2

Table 1: Test accuracy on mathematical reasoning benchmarks under task-specific training settings. Models are trained on the GSM8K training set for GSM8K evaluation, and on the MATH training set for MATH-500 and OlympiadBench evaluation. Best results in each setting are bolded.

across three independent runs. In each run, we perform thirty-two stochastic sampling trials and one greedy decoding trial during testing. The reported results are the mean and standard deviation over these independent runs. Detailed hyperparameter settings are provided in Appendix A.

5.2 Results

Mathematical Reasoning. Table 1 reports results on GSM8K, MATH-500, and OlympiadBench. Overall, SOFT-FDSD achieves the best average performance and remains competitive in most individual settings. Compared with GRPO, SOFT-FDSD improves the average accuracy over the six evaluation settings by 1.1%, 0.9%, 0.9%, and 0.3% on Qwen2.5-1.5B, Qwen2.5-3B, Qwen2.5-7B, and Qwen3-4B, respectively. The gains are most evident on GSM8K and MATH-500, where SOFT-FDSD consistently outperforms other settings. On OlympiadBench, SOFT-FDSD also remains competitive and obtains the best sampled accuracy for most Qwen2.5 models. These results demonstrate the effectiveness of full-distribution self-distillation for improving mathematical reasoning.

Out-of-Domain Generalization. Table 2 shows results on HellaSwag, ARC-Challenge, and MMLU after training on the MATH training set. SOFT-FDSD achieves the highest average OOD accuracy across all evaluated model scales. Compared with GRPO, it improves average accuracy by 0.2%, 1.0%, 0.6%, and 0.2% on Qwen2.5-1.5B, Qwen2.5-3B, Qwen2.5-7B, and Qwen3-4B, respectively. Although the absolute gains are moderate, they are consistent across model scales and are obtained without increasing inference-time complexity. These results suggest that SOFT-FDSD can enhance mathematical reasoning while mitigating catastrophic forgetting.

6 Analysis

Efficiency. FDSD introduces additional computation in two places: the rollout phase, where the soft rollout distribution is constructed, and the actor update phase, where the importance weight and advantage-weighted distillation loss are computed. These operations are lightweight compared with the forward and backward passes of the language model, so the overall training overhead remains limited.

Figure 2 reports the per-iteration efficiency on a single NVIDIA A100 GPU. During stable training, GRPO requires approximately 82 seconds for generation and 40 seconds for actor update. FDSD requires approximately 88 seconds and 43 seconds for the corresponding phases, resulting in only about a 7% increase in total training time. Moreover, the average response length of FDSD is about 5% shorter than that of GRPO after conver-

Model	Method	Hellaswag	ARC-Challenge	MMLU	Average
Qwen2.5-1.5B-Instruct	no-finetune	51.7 ± 0.0	52.1 ± 0.0	46.0 ± 0.0	49.9
	GRPO	$\mathbf{58.4 \pm 0.2}$	66.9 ± 0.1	51.6 ± 0.1	59.0
	HRPO	57.1 ± 0.1	63.6 ± 0.2	50.1 ± 0.2	56.9
	SOFT-TOKEN	55.5 ± 0.2	53.5 ± 0.5	49.5 ± 0.4	52.8
	SOFT-INFER	58.0 ± 0.1	67.0 ± 0.2	52.0 ± 0.1	59.0
	SOFT-FDSD	58.2 ± 0.1	$\mathbf{67.1 \pm 0.1}$	$\mathbf{52.1 \pm 0.1}$	**59.2**
Qwen2.5-3B-Instruct	no-finetune	62.9 ± 0.0	79.2 ± 0.0	$\mathbf{62.0 \pm 0.0}$	68.0
	GRPO	62.7 ± 0.1	79.2 ± 0.1	61.5 ± 0.1	67.8
	HRPO	58.6 ± 0.2	76.7 ± 0.2	58.5 ± 0.2	64.6
	SOFT-TOKEN	60.1 ± 0.2	79.0 ± 0.2	61.2 ± 0.3	66.8
	SOFT-INFER	62.4 ± 0.1	79.3 ± 0.2	60.9 ± 0.1	67.5
	SOFT-FDSD	$\mathbf{65.3 \pm 0.2}$	$\mathbf{79.4 \pm 0.1}$	61.7 ± 0.1	**68.8**
Qwen2.5-7B-Instruct	no-finetune	75.1 ± 0.0	86.4 ± 0.0	69.4 ± 0.0	77.0
	GRPO	76.4 ± 0.3	86.5 ± 0.2	68.4 ± 0.1	77.1
	SOFT-TOKEN	75.4 ± 0.3	86.3 ± 0.1	69.0 ± 0.2	76.9
	SOFT-INFER	76.3 ± 0.1	86.5 ± 0.2	68.3 ± 0.3	77.0
	SOFT-FDSD	$\mathbf{76.5 \pm 0.1}$	$\mathbf{87.1 \pm 0.3}$	$\mathbf{69.5 \pm 0.2}$	**77.7**
Qwen3-4B-Base	no-finetune	18.8 ± 0.0	33.4 ± 0.0	31.7 ± 0.0	28.0
	GRPO	61.1 ± 0.1	85.5 ± 0.2	67.2 ± 0.1	71.3
	SOFT-TOKEN	61.4 ± 0.4	82.5 ± 0.3	66.2 ± 0.2	70.3
	SOFT-INFER	$\mathbf{61.6 \pm 0.2}$	85.6 ± 0.2	67.0 ± 0.1	71.4
	SOFT-FDSD	61.2 ± 0.1	$\mathbf{85.6 \pm 0.1}$	$\mathbf{67.6 \pm 0.1}$	**71.5**

Table 2: Out-of-domain generalization results. All models are trained on the MATH training set and then evaluated on HellaSwag, ARC-Challenge, and MMLU. Best results in each setting are bolded.

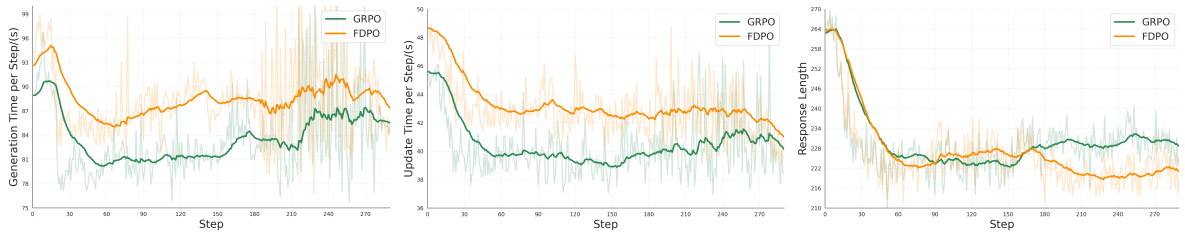

Figure 2: Efficiency comparison between GRPO and FDSD on Qwen2.5 with GSM8K using a single NVIDIA A100 GPU. From left to right, the figures show generation time per iteration, actor update time per iteration, and average response length. FDSD introduces only a small computational overhead while producing shorter responses after convergence.

gence, suggesting that FDSD improves reasoning performance without substantially increasing computational cost.

Ablation Study on FDSD Components. To analyze the contribution of each component, we construct two variants by separately modifying the rollout strategy and the policy update objective:

1. **SOFT-GRPO**: uses soft rollout distributions during generation, but updates the policy through the GRPO objective based on the probability of the sampled rollout action.

2. **HARD-FDSD**: uses discrete token sampling during rollout, but constructs a posterior distribution as the distillation target and updates the policy with the FDSD objective.

Rollout-Update	1.5b instruct	3b instruct
(HARD -) GRPO	75.1 ± 0.5	84.9 ± 0.2
HARD - FDSD	75.7 ± 0.2	85.4 ± 0.2
SOFT - GRPO	75.6 ± 0.3	85.4 ± 0.2
SOFT - FDSD	$\mathbf{76.3 \pm 0.2}$	$\mathbf{85.7 \pm 0.2}$

Table 3: GSM8K accuracy of Qwen2.5-1.5/3B-Instruct under different training strategies.

Table 3 shows that both components contribute to the final performance. SOFT-GRPO improves over the baseline, indicating that soft rollout distributions provide better exploration than discrete token sampling. HARD-FDSD also improves performance, showing that full-distribution updates provide a stronger training signal than rollout-action

Rollout \ Update	GRPO	FDSD
Discrete Sampling	75.1 ± 0.5	–
Mixture of Inputs	75.6 ± 0.3	$\mathbf{76.3 \pm 0.2}$
Dirichlet Sampling	65.2 ± 0.7	71.9 ± 0.8
Gumbel-Softmax	74.8 ± 0.4	75.7 ± 0.3

Table 4: GSM8K accuracy of Qwen2.5-1.5B-Instruct under different rollout strategies.

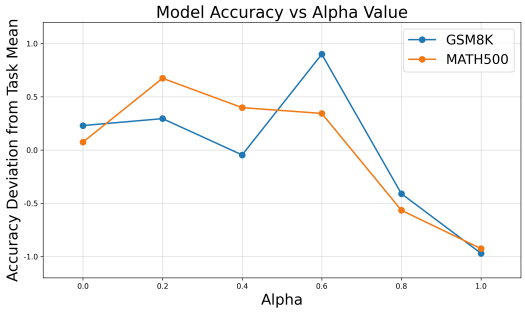

Figure 3: Effect of the hybrid coefficient α on Qwen2.5-1.5B performance on GSM8K and MATH.

likelihood updates. Combining both components, SOFT-FDSD achieves the best results, suggesting that soft rollout exploration and full-distribution self-distillation are complementary.

Effect of Rollout Strategies. FDSD is compatible with different rollout strategies, since the method only requires the probability of generating the rollout distribution. Besides MoI, we also evaluate Dirichlet sampling and Gumbel-Softmax sampling (Wu et al., 2025). Details of these rollout strategies and their probability computation are provided in Appendix B.

Table 4 reports the results on Qwen2.5-1.5B-Instruct with GSM8K. When the rollout strategy is Discrete Sampling, using FDSD is equivalent to GRPO as discussed in Section 4.4; therefore, we report only the GRPO result for this setting. Among the evaluated strategies, Mixture of Inputs achieves the best accuracy. This is likely because MoI preserves both the discrete observation signal and the full model distribution, while keeping the rollout probability simple and stable to compute. Dirichlet sampling performs substantially worse, suggesting that directly sampling from the probability simplex may introduce overly noisy rollout targets. Across all evaluated soft rollout strategies, FDSD consistently outperforms the corresponding GRPO update, confirming the benefit of full-distribution self-distillation.

Training with a Hybrid Loss. To study the relation between sample-based policy optimization and full-distribution distillation, we further consider a hybrid loss:

$$\mathcal{L}_{\text{hybrid}} = \alpha \mathcal{L}_{\text{GRPO}} + (1-\alpha)\mathcal{L}_{\text{FDSD}}, \tag{19}$$

where $\alpha \in [0, 1]$ controls the interpolation between the two objectives. When $\alpha = 1$, the objective reduces to GRPO; when $\alpha = 0$, it becomes pure FDSD.

Assume that the sampled rollout action corresponds to token index i in the discrete case. The hybrid objective can be interpreted as using an effective target distribution that interpolates between the one-hot action target and the soft rollout distribution:

$$\tilde{p}_j = (1-\alpha)p_j + \alpha \mathbb{I}(j = i). \tag{20}$$

Therefore, the gradient with respect to logit z_j can be written as

$$\nabla_{z_{t,j}} \mathcal{L}_{\text{hybrid}}(\theta) = A \frac{\pi_\theta}{\pi_{\theta_{\text{old}}}} \left(q_{\theta,j} - \tilde{p}_j\right). \tag{21}$$

This shows that the hybrid loss smoothly interpolates between sample-based policy optimization and full-distribution self-distillation.

Figure 3 reports the effect of α on Qwen2.5-1.5B. The results indicate that an appropriate interpolation can further improve performance, suggesting that rollout-action likelihood optimization and full-distribution distillation provide complementary training signals.

7 Conclusion

We propose **Full-Distribution Self-Distillation (FDSD)**, an off-policy framework for optimizing soft latent reasoning. FDSD aligns the model distribution with advantage-weighted soft rollout distributions, providing denser supervision than rollout-action likelihood optimization. We show that FDSD generalizes GRPO: it recovers the standard update when the rollout target is one-hot, and distills the full distribution when the target is soft.

Experiments on mathematical reasoning and out-of-domain benchmarks show that FDSD consistently improves reasoning performance while preserving generalization, with only modest computational overhead. These results suggest that full-distribution alignment is an effective direction for training soft latent reasoning models.

8 Limitations

Vocabulary Manifold Constraint. FDSD represents latent reasoning states as weighted combinations of token embeddings. While this stabilizes training, it may limit exploration beyond the vocabulary manifold.

Dependence on Rollout Strategies. FDSD relies on stochastic rollout strategies to generate useful soft targets. Although MoI performs well in our experiments, better exploration mechanisms may further improve performance.

Evaluation Scope. Our evaluation focuses mainly on mathematical reasoning and standard OOD benchmarks. Extending FDSD to open-ended generation, human-preference alignment, and multimodal reasoning remains future work.

References

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. 2018. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*.

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

Natasha Butt, Ariel Kwiatkowski, Ismail Labiad, Julia Kempe, and Yann Ollivier. 2025. Soft tokens, hard truths. *arXiv preprint arXiv:2509.19170*.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. 2025. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Jingcheng Deng, Liang Pang, Zihao Wei, Shichen Xu, Zenghao Duan, Kun Xu, Yang Song, Huawei Shen, and Xueqi Cheng. 2025. Latent reasoning in llms as a vocabulary-space superposition. *arXiv preprint arXiv:2510.15522*.

Subhabrata Dutta, Joykirat Singh, Soumen Chakrabarti, and Tanmoy Chakraborty. 2024. How to think step-by-step: A mechanistic understanding of chain-of-thought reasoning. *arXiv preprint arXiv:2402.18312*.

Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. 2023. Towards revealing the mystery behind chain of thought: a theoretical perspective. *Advances in Neural Information Processing Systems*, 36:70757–70798.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. 2024. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, and 1 others. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

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

Yujie Hou, Ting Zhang, Mei Wang, Xuetao Ma, and Hua Huang. 2025. Smart: Self-generating and self-validating multi-dimensional assessment for llms' mathematical problem solving. *arXiv preprint arXiv:2505.16646*.

Sina Bagheri Nezhad, Yao Li, and Ameeta Agrawal. 2025. Symcode: A neurosymbolic approach to mathematical reasoning via verifiable code generation. *arXiv preprint arXiv:2510.25975*.

Jan Peters, Katharina Mulling, and Yasemin Altun. 2010. Relative entropy policy search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pages 1607–1612.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv: Learning,arXiv: Learning*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical

reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. 2025. Codi: Compressing chain-of-thought into continuous space via self-distillation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 677–693.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*.

H Francis Song, Abbas Abdolmaleki, Jost Tobias Springenberg, Aidan Clark, Hubert Soyer, Jack W Rae, Seb Noury, Arun Ahuja, Siqi Liu, Dhruva Tirumala, and 1 others. 2019. V-mpo: On-policy maximum a posteriori policy optimization for discrete and continuous control. *arXiv preprint arXiv:1909.12238*.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Qwen Team. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Junhong Wu, Jinliang Lu, Zixuan Ren, Gangqiang Hu, Zhi Wu, Dai Dai, and Hua Wu. 2025. Llms are single-threaded reasoners: Demystifying the working mechanism of soft thinking. *arXiv preprint arXiv:2508.03440*.

Chenxu Yang, Chuanyu Qin, Qingyi Si, Minghui Chen, Naibin Gu, Dingyu Yao, Zheng Lin, Weiping Wang, Jiaqi Wang, and Nan Duan. 2026. Self-distilled rlvr. *arXiv preprint arXiv:2604.03128*.

Zhenrui Yue, Bowen Jin, Huimin Zeng, Honglei Zhuang, Zhen Qin, Jinsung Yoon, Lanyu Shang, Jiawei Han, and Dong Wang. 2026. Hybrid latent reasoning via reinforcement learning. *Advances in Neural Information Processing Systems*, 38:5501–5530.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Xiang Zhang, Juntai Cao, Jiaqi Wei, Yiwei Xu, and Chenyu You. 2025. Tokenization constraints in llms: A study of symbolic and arithmetic reasoning limits. *arXiv preprint arXiv:2505.14178*.

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, and Xin Wang. 2026. Soft thinking: Unlocking the reasoning potential of llms in continuous concept space. *Advances in Neural Information Processing Systems*, 38:168990–169012.

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

Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart J Russell, and Yuandong Tian. 2026. Reasoning by superposition: A theoretical perspective on chain of continuous thought. *Advances in Neural Information Processing Systems*, 38:79931–79963.

Yufan Zhuang, Liyuan Liu, Chandan Singh, Jingbo Shang, and Jianfeng Gao. 2025. Text generation beyond discrete token sampling. *arXiv preprint arXiv:2505.14827*.

A Hyperparameter

Our experimental parameters are primarily configured based on the HRPO settings but are implemented under the VERL (Sheng et al., 2024) framework. Key modifications include a reduction of the learning rate from 5×10^{-6} to 1×10^{-6}, and an increase of the total training batch size by a factor of eight. To ensure an equivalent number of training steps, we correspondingly scale the training epochs by eight times. Furthermore, as our training process remains stable, we do not incorporate a KL divergence penalty, aiming to further facilitate the model's acquisition of latent reasoning capabilities.

For GSM8K, we generate four responses per question, with a maximum response length of 512 tokens. For more challenging benchmarks, MATH500 and OlympiadBench, we generate eight responses per question and set a longer maximum response length of 1024 tokens.

When running the baselines (GRPO, HRPO, SoftTokens), we ensured that all hyperparameters remained consistent with our own experimental setup. Additionally, the same prompt, "Let's think step by step and output the final answer after '####'," was prepended to every problem across all tasks. To evaluate the correctness of model generations, we extracted the content following the first occurrence of "####" in the model output and

Parameter	Value
Algorithm	FDSD
Epochs	8
Optimizer	AdamW 8bit
Optimizer Momentum	$\beta_1, \beta_2 = 0.9, 0.99$
Weight Decay	0.01
Learning Rate	1e-6
MoI β	1.0
Gradient Accumulation Step	4
N Response Per Question	4 / 8
Total Train Batch Size	256 / 512
Prompt Length	512
Completion Length	512 / 1024
Temperature	0.6
LR Scheduler	Cosine with Warmup
Warmup Ratio	0.1
Precision (WA)	BF16-mixed
LoRA Modules	query, key, value, dense
LoRA Rank	32
LoRA α	64

Table 5: Experiment hyperparameter settings.

compared it exactly with the ground-truth answer: a perfect match yields a score of 1, otherwise 0.

All datasets, base models, and software frameworks used in this work are publicly available, and we follow their original licenses and terms of use.

B Rollout Strategies and Probability Computation

At decoding step t, the LLM first outputs logits z_t. We select the top-k token set

$$S_t = \mathrm{TopK}(z_t, k),$$

and apply softmax within S_t to obtain the normalized distribution

$$q_{t,i} = \frac{\exp z_{t,i}}{\sum_{j \in S_t} \exp z_{t,j}}, \quad i \in S_t.$$

A rollout strategy then samples or constructs a rollout distribution p_t over S_t. The next input embedding is obtained by

$$e_t = \sum_{i \in S_t} p_{t,i} E_i,$$

where E_i denotes the embedding of token i.

AddNoiseDirichlet. AddNoiseDirichlet samples the rollout distribution directly on the top-k probability simplex. Given a concentration coefficient $\lambda > 0$, we define

$$\alpha_{t,i} = \lambda q_{t,i}, \quad i \in S_t,$$

and sample

$$p_t \sim \mathrm{Dirichlet}(\alpha_t).$$

The probability density of the sampled rollout distribution is

$$\mathrm{Prob}(p_t | q_t) = \frac{\Gamma\left(\sum_{i \in S_t} \alpha_{t,i}\right)}{\prod_{i \in S_t} \Gamma(\alpha_{t,i})} \prod_{i \in S_t} p_{t,i}^{\alpha_{t,i}-1}.$$

AddNoiseGumbel. AddNoiseGumbel adds Gumbel noise to the top-k logits:

$$g_{t,i} \sim \mathrm{Gumbel}(0, 1), \quad i \in S_t,$$

$$\hat{z}_{t,i} = \log q_{t,i} + g_{t,i}.$$

The rollout distribution is obtained by applying a temperature-controlled softmax:

$$p_{t,i} = \frac{\exp(\hat{z}_{t,i}/\tau)}{\sum_{j \in S_t} \exp(\hat{z}_{t,j}/\tau)}, \quad i \in S_t,$$

where $\tau > 0$ is the temperature.

The probability density of rollout distribution is given by

$$\mathrm{Prob}(p_t|q_t) = \Gamma(k)\tau^{k-1}\frac{\prod_{i\in S_t} q_{t,i}p_{t,i}^{-\tau-1}}{\left(\sum_{j\in S_t} q_{t,j}p_{t,j}^{-\tau}\right)^k}.$$

AddNoiseGaussian. AddNoiseGaussian adds Gaussian noise to the top-k logits:

$$\epsilon_{t,i} \sim \mathcal{N}(0, \sigma_t^2), \quad i \in S_t,$$

$$\hat{z}_{t,i} = z_{t,i} + \epsilon_{t,i}.$$

The rollout distribution is then computed by

$$p_{t,i} = \frac{\exp \hat{z}_{t,i}}{\sum_{j\in S_t} \exp \hat{z}_{t,j}}, \quad i \in S_t.$$

We compute the probability on the perturbed logits:

$$\mathrm{Prob}(\hat{z}_t \mid z_t) = \prod_{i\in S_t} \frac{1}{\sqrt{2\pi}\sigma_t} \exp\left(-\frac{(\hat{z}_{t,i} - z_{t,i})^2}{2\sigma_t^2}\right).$$

Mixture of Inputs. Mixture of Inputs also constructs a soft rollout distribution p_t, but it does not add continuous noise to logits. Instead, it first samples a token from the distribution q_t:

$$i \sim \mathrm{Categorical}(q_t),$$

and denotes the corresponding one-hot observation as

$$y_{t,j} = \mathbb{I}(j = i), \quad j \in S_t.$$

The normalized entropy of q_t is

$$H_t = -\frac{1}{\log k}\sum_{j\in S_t} q_{t,j} \log q_{t,j}.$$

The prior concentration is defined as

$$\alpha_{t,j} = H_t q_{t,j},$$

and the sampled token contributes the observation count

$$c_{t,j} = (\beta + 1 - H_t)y_{t,j}.$$

The rollout distribution is given by the posterior mean:

$$\begin{aligned} p_{t,j} &= \frac{\alpha_{t,j} + c_{t,j}}{\sum_{\ell\in S_t}(\alpha_{t,\ell} + c_{t,\ell})} \\ &= \frac{H_t q_{t,j} + (\beta + 1 - H_t)y_{t,j}}{\beta + 1}. \end{aligned}$$

Since p_t is deterministically computed from q_t and the sampled token i, the probability of generating p_t equals the probability of sampling token i:

$$\mathrm{Prob}(p_t|q_t) = q_{t,i}.$$

AddNoiseGaussianOnLatent. AddNoiseGaussianOnLatent perturbs the latent embedding directly. First, an unperturbed soft embedding is constructed from q_t:

$$e_t^0 = \sum_{i\in S_t} q_{t,i}E_i.$$

Then Gaussian noise is added in the latent space:

$$\epsilon_t \sim \mathcal{N}(0, \sigma_t^2 I),$$

$$e_t = e_t^0 + \epsilon_t.$$

The probability density of the perturbed latent embedding is

$$\mathrm{Prob}(e_t \mid e_t^0) = \frac{1}{(2\pi\sigma_t^2)^{d/2}} \exp\left(-\frac{\|e_t - e_t^0\|_2^2}{2\sigma_t^2}\right),$$

where d is the embedding dimension.