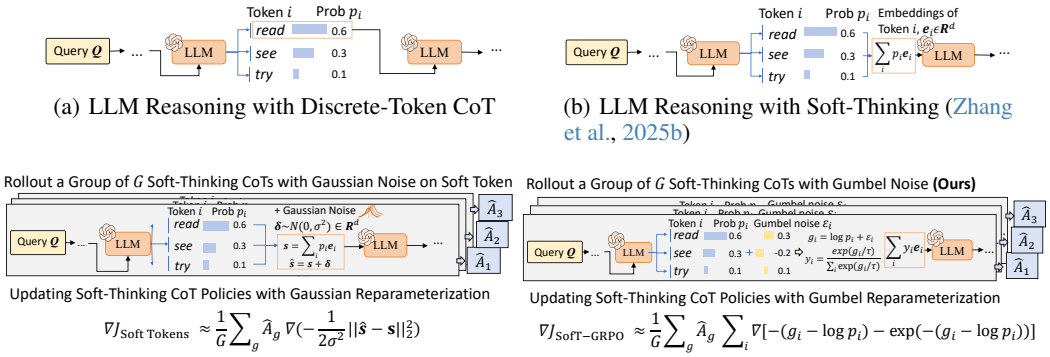
SofT-GRPO: Powerful Soft-Thinking Policy Optimization via Gumbel-Reparameterization

Anonymous authors
Paper under double-blind review

Abstract

The soft-thinking reasoning paradigm has shown promising results alongside traditional discrete-token Chain-of-Thought (CoT) reasoning. However, unlike discrete-token CoT reasoning that can be reinforced through advanced reinforcement learning with verifiable rewards (RLVR) techniques such as group relative policy optimization (GRPO), extending soft-thinking reasoning with such strong techniques remains challenging. This difficulty stems from the complexities of injecting stochasticity into soft-thinking tokens and updating soft-thinking policies accordingly. Existing approaches introduce stochasticity through different action parameterizations, motivating further study of their policy optimization. We introduce SofT-GRPO for optimizing soft-thinking policies with verifiable rewards. It injects Gumbel noise into token probabilities with Gumbel-Softmax for controllable stochasticity, and leverages the Gumbel reparameterization trick to evaluate sampled action likelihoods for LLM soft-thinking policy updates. We conduct experiments over LLMs ranging from 1.5B to 7B parameters, where SofT-GRPO enables LLMs with soft-thinking to retain comparable Pass@1 (+0.23 percentage points on average) and improve Pass@32 (+2.19 points on average) relative to discrete-token GRPO across five mathematical benchmarks and three primary models. The code is available at https://anonymous.4open.science/r/SofT-GRPO-master-A512.



(a) LLM Reasoning with Discrete-Token CoT

(b) LLM Reasoning with Soft-Thinking (Zhang et al., 2025b)

(c) Existing Work of Soft-Thinking + GRPO (Butt et al., 2025)

(d) SofT-GRPO **(Ours)** for Reinforcing Soft-Thinking

Figure 1: The soft-thinking pattern (b) passes the expectation of token embeddings to the next LLM step (Zhang et al., 2025b), using a continuous mixture in place of the discrete-token CoT representation (a). Existing work (c) applies GRPO with Gaussian actions in embedding space. SofT-GRPO (d) uses Gumbel-perturbed log-probabilities as actions and their densities for policy optimization. Section 4 reports the empirical comparisons.

1 Introduction

Reasoning with large language models (LLMs) has demonstrated impressive versatility in diverse domains (Sprague et al., 2024). However, most existing reasoning methods rely on the generation of discrete language tokens, while soft thinking provides an alternative representation based on token

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

mixtures (Zhang et al., 2025b). To explore this representation, (Zhang et al., 2025b) presents the soft-thinking reasoning pattern. As shown in Figure 1 (b), it replaces each discrete token in the chain-of-thought (CoT) with a continuous representation: the weighted sum of d-dimensional token embeddings, taking the output probabilities as the weights. With suitable sampling techniques—such as the Gumbel-Softmax technique (Wu et al., 2026) or the Dirichlet resampling technique (Zhuang et al., 2025)—soft-thinking can outperform discrete-token CoT on a wide range of tasks without fine-tuning Wu et al. (2026).

Recently, reinforcement learning with verifiable rewards (RLVR) approaches have become standard for improving discrete-token CoT reasoning (Shao et al., 2024; Liu et al., 2025b). The most representative RLVR method, Group Relative Policy Optimization (GRPO) (Shao et al., 2024), samples groups of CoT trajectories per query and reinforces the policy toward higher-reward examples. Notably, RLVR fine-tuning can yield discrete-token CoT performance that surpasses that of soft-thinking reasoning from the original, unfine-tuned LLMs. Moreover, after discrete-token GRPO, the model's soft-thinking ability often does not improve as substantially as its discrete-token reasoning ability.

However, extending RLVR methods to soft-thinking remains challenging (Jain & Rappazzo, 2025). As illustrated in Figure 1 (b), in contrast to the stochastic nature of sampling-based discrete-token CoTs, soft-thinking reasoning produces **deterministic** soft-tokens, without a stochastic action likelihood for GRPO. So, extending RLVR to soft-thinking should (**1**) introduce controllable stochasticity to adequately explore diverse soft-thinking CoTs, and (**2**) define tractable action likelihoods for policy updates using the rewards of sampled soft-thinking trajectories. Previous work (Butt et al., 2025) (Figure 1 (c)) attempts to address stochasticity by injecting Gaussian noise into soft tokens and applying a Gaussian reparameterization trick for gradients, defining a stochastic policy in embedding space. So, these observations motivate a central question:

How to conduct effective policy optimization of the soft-thinking reasoning paradigm?

To address these challenges and fully unlock the LLM reasoning ability by soft-thinking, we propose a powerful policy optimization algorithm, **SofT-GRPO**. As shown in Figure 1(d), to introduce **controllable stochasticity** in the rollout process, SofT-GRPO samples groups of soft-thinking reasoning paths by injecting Gumbel noises into the output token probabilities and employs the Gumbel-Softmax technique to avoid invalid soft-tokens outside the pre-trained discrete-token embedding space. To **accurately assign the reward credits** to the **output probability** of LLMs in soft-thinking policy updates, SofT-GRPO leverages the reparameterization trick on the Gumbel distribution. We perform thorough evaluations of SofT-GRPO on three representative LLMs from 1.5B to 7B parameters and across five numerical reasoning benchmarks. Experimental results show that SofT-GRPO-enhanced soft-thinking not only slightly surpasses discrete-token GRPO on the averaged Pass@1 but also yields clear gains on Pass@16, Pass@32, highlighting the practical advantages of robust policy optimization for soft-thinking reasoning. Our contributions are as follows:

- We present **SofT-GRPO**, an RLVR algorithm designed for the soft-thinking reasoning paradigm. It applies the Gumbel-Softmax technique to the group rollout process, actively obtaining **stochastic** soft-thinking reasoning paths.
- SofT-GRPO integrates stored-action likelihood ratios into GRPO-style policy optimization by formulating action likelihoods via **Gumbel reparameterization**.
- Over eight in-domain and out-of-domain benchmarks, SofT-GRPO improves aggregate reasoning coverage relative to **the tested discrete-token GRPO baselines**, especially at higher sample rates (Pass@16 or Pass@32).

2 Preliminaries

2.1 Discrete-Token CoT Reasoning

To solve a $|\boldsymbol{Q}|$-token question $\boldsymbol{Q} = (q_1, \ldots, q_{|\boldsymbol{Q}|})$, the discrete-token CoT reasoning process generates $|\boldsymbol{R}|$ reasoning CoT tokens $\boldsymbol{R} = (r_1, \ldots, r_{|\boldsymbol{R}|})$ before predicting the answer $\boldsymbol{A} = (a_1, \ldots, a_{|\boldsymbol{A}|})$ (Guo et al., 2025; Sprague et al., 2024). Each of $\boldsymbol{Q}$, $\boldsymbol{R}$, and $\boldsymbol{A}$ is a token sequence over the vocabulary $\mathcal{T}$ (i.e., $\boldsymbol{Q}, \boldsymbol{R}, \boldsymbol{A} \in \mathcal{T}^*$), where $\mathcal{T}$ is the set of all possible language tokens. Language reasoning

tokens and answer tokens are generated with the next-token prediction policy of LLM π_θ as follows:

$$p(\boldsymbol{R}, \boldsymbol{A}|\boldsymbol{Q}) = \prod_{t=1}^{|\boldsymbol{R}|} \pi_\theta(r_t|[\boldsymbol{Q}, (r_1, \ldots, r_{t-1})]) \quad \prod_{t=1}^{|\boldsymbol{A}|} \pi_\theta(a_t|[\boldsymbol{Q}, \boldsymbol{R}, (a_1, \ldots, a_{t-1})]), \tag{1}$$

where $[\cdot,\ \cdot]$ and $[\cdot,\ \cdot,\ \cdot]$ denote concatenation.

Supervised Fine-tuning (SFT) for Discrete-Token Reasoning. As straightforward fine-tuning methods for LLM reasoning, the SFT methods (Wu et al., 2025; Zheng & Lee, 2025) collect high-quality CoT labels for each question and then fine-tune LLMs for correct predictions. However, SFT methods highly rely on the quality of CoT labels, which are hard to obtain for complex datasets. Moreover, there are also concerns about the out-of-domain generalization ability (Chu et al., 2025).

RLVR Fine-tuning for Discrete-Token Reasoning. RLVR fine-tuning methods such as GRPO (Liu et al., 2024), Dr. GRPO (Liu et al., 2025b), and DAPO Yu et al. (2025) sample several CoTs $[\boldsymbol{R}, \boldsymbol{A}]$ and assign a reward to each of them based on the quality of the answers $\boldsymbol{A}$. The standard discrete-token GRPO algorithm (Shao et al., 2024) samples G CoTs for each question $\boldsymbol{Q}$ with sampling policy $\pi_{\theta_{\text{old}}}$ and optimizes the reasoning policy towards CoTs with higher rewards. Its loss function is shown in Eq. (2), where $\hat{A}_g$ represents the advantage function for the g-th CoT, the reward function $f(\boldsymbol{A}_g) = 1$ if and only if the answer $\boldsymbol{A}_g$ is correct, ϵ is the clipping hyperparameter, $p_{g,t}$ is calculated in an off-policy way for both the current policy π_θ and the sampling policy $\pi_{\theta_{\text{old}}}$, and $\mathbb{D}_{KL}$ represents the KL-divergence of π_θ and a reference policy $\pi_{\theta_{\text{ref}}}$. RLVR methods can improve the discrete-token reasoning performance of LLMs across a wide range of reasoning problems without the need for CoT labels. At each reasoning step, discrete-token GRPO evaluates the likelihood of the sampled categorical action. Prior work (Yue et al., 2025) reports that gains in single-sample accuracy can accompany lower multi-sample coverage; we therefore evaluate both metrics.

$$\mathcal{J}_{\text{GRPO}}(\theta) = \frac{1}{G}\mathbb{E}_{\boldsymbol{Q}\sim\mathcal{D},\{\boldsymbol{R}\}_{g=1}^{G},\{\boldsymbol{A}\}_{g=1}^{G}\sim p(\cdot,\cdot|\boldsymbol{Q})}$$
$$\left[\sum_{g=1}^{G} \frac{1}{|\boldsymbol{R}_g| + |\boldsymbol{A}_g|} \sum_{t=1}^{|\boldsymbol{R}_g|+|\boldsymbol{A}_g|} \left(\min\left(p_{g,t}\hat{A}_g, \text{clip}(p_{g,t}, 1-\epsilon, 1+\epsilon)\hat{A}_g\right) - \beta\mathbb{D}_{KL}(\pi_\theta||\pi_{\theta_{\text{ref}}}) \right] \tag{2}$$
$$\hat{A}_g = \frac{f(\boldsymbol{A}_g) - \text{mean}(f(\boldsymbol{A}))_{g=1}^{G}}{\text{std}(f(\boldsymbol{A}))_{g=1}^{G}}, \qquad p_{g,t} = \begin{cases} \frac{\pi_\theta(a_{g,t}|[\boldsymbol{Q},\boldsymbol{R}_g,(a_{g,1},\ldots,a_{g,t-1})])}{\pi_{\theta_{\text{old}}}(a_{g,t}|[\boldsymbol{Q},\boldsymbol{R},(a_{g,1},\ldots,a_{g,t-1})])} & \text{if } t > |\boldsymbol{R}_g| \\ \frac{\pi_\theta(r_{g,t}|[\boldsymbol{Q},(r_{g,1},\ldots,r_{g,t-1})])}{\pi_{\theta_{\text{old}}}(r_{g,t}|[\boldsymbol{Q},(r_{g,1},\ldots,r_{g,t-1})])} & \text{if } t \leq |\boldsymbol{R}_g|. \end{cases}$$

2.2 Soft-Thinking Reasoning Paradigm

At each of the CoT reasoning steps, conventional discrete-token reasoning is constrained to selecting one token from the token set $\mathcal{T}$, while soft thinking retains a mixture of token embeddings (Zhang et al., 2025b). To construct this representation, (Zhang et al., 2025b) presents the soft-thinking paradigm, which replaces discrete-token reasoning steps $\boldsymbol{R}$ with a soft-thinking reasoning path $\boldsymbol{S} = (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{|\boldsymbol{S}|})$. Each token $\boldsymbol{s}_i \in \mathcal{R}^d$ (noted soft token) is a real vector, which is the sum of token embeddings weighted by their respective output probabilities. They are then fed into the next LLM step as follows:

$$\boldsymbol{p}_t = \pi_\theta(\cdot|[\boldsymbol{Q}, (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{t-1})]), \quad \boldsymbol{s}_t = \sum_{i=1}^{|\mathcal{T}|} p_i \cdot \boldsymbol{e}_i, \tag{3}$$

where $p_i \in [0, 1]$ is the predicted probability of token i in $\boldsymbol{p}_t$ and $\boldsymbol{e}_i \in \mathcal{R}^d$ is the LLM embeddings of token i. As detailed in Appendix A.3, the soft tokens lie in the **convex hull** of token embeddings, describing the geometry of this mixture representation.

Based on Eq. (3), (Wu et al., 2026) find that most pre-trained LLMs tend to be single-threaded, and vanilla soft-thinking reasoning may lead to a **greedy feedback loop** that suppresses alternative reasoning paths and undermines the benefits of transmitting more informative soft tokens. So, they propose to introduce sampling strategies in the soft-thinking reasoning process for randomness, employing Gumbel noises and Gumbel-Softmax technique (Maddison et al., 2016; Jang et al., 2016) based on the output probability $\boldsymbol{p}_t$ as follows:

$$g_i = \log p_i + \epsilon_i, \quad y_i = \frac{\exp(g_i/\tau_g)}{\sum_{i=1}^{|\mathcal{T}|}\exp(g_i/\tau_g)}, \quad \boldsymbol{s}_t = \sum_{i=1}^{|\mathcal{T}|} y_i \cdot \boldsymbol{e}_i, \tag{4}$$

where ϵ_i is a scalar noise sampled from the Gumbel distribution Gumbel$(0, 1)$, and τ_g is the temperature of Gumbel-Softmax. Besides Gumbel-Softmax, (Wu et al., 2026) also tries to use the Dirichlet resampling technique as follows:

$$(x_1, \ldots, x_{|\mathcal{T}|}) \sim \text{Dirichlet}(\alpha, \boldsymbol{p}_t), \quad \boldsymbol{s}_t = \sum_{i=1}^{|\mathcal{T}|} x_i \cdot \boldsymbol{e}_i, \tag{5}$$

where α is a scaling parameter. Empirically, when incorporated with Gumbel noises and the Gumbel-Softmax technique, the soft-thinking pattern can outperform conventional discrete-token CoT on a broad range of tasks, including numerical, code, and scientific reasoning, without requiring any fine-tuning. (Wu et al., 2026).

2.3 Related Work on RLVR Soft-Thinking

However, after being boosted with the RLVR fine-tuning, discrete-token reasoning will clearly outperform soft-thinking. So, (Butt et al., 2025) tries to similarly improve the soft-thinking reasoning ability using GRPO by adding Gaussian noise to the soft-token $\boldsymbol{s}_t$ in Eq. (3) as follows:

$$\hat{\boldsymbol{s}}_t = \boldsymbol{s}_t + \mathcal{N}(0, \sigma^2 Id) \tag{6}$$

In deriving the soft-thinking probability in GRPO, they reconstruct the value of $\hat{\boldsymbol{s}}_t$ in the rollout and calculate the log-probability for the soft-thinking reasoning path $\boldsymbol{S}$ with the Gaussian reparameterization trick as follows, keeping the updating process of the answering part $\boldsymbol{A}$ unchanged:

$$p(\hat{\boldsymbol{s}}_t) \propto \exp\left(-\frac{1}{2\sigma^2}||\hat{\boldsymbol{s}}_t - \boldsymbol{s}_t||_2^2\right).$$

Although this method can improve the soft-thinking ability of base LLMs and shows improvements compared to discrete-token GRPO on the Pass@32 metrics (i.e., the pass rate with 32 attempts), its reported single-sample accuracy is lower in the compared settings. The two formulations differ in their action parameterization. As analyzed in Appendix C.1, token embeddings are linearly dependent ($|\mathcal{T}| \gg d$), so different token mixtures can induce the same Gaussian mean. Gaussian perturbations also explore embedding-space actions beyond sparse token mixtures. SofT-GRPO uses perturbed log-probabilities as actions and normalized weights to construct soft tokens. These choices define different action supports and motivate the empirical evaluation.

3 SofT-GRPO: Reinforcing Soft-Thinking Policy with Gumbel Reparameterization

To fully unlock the potential of soft-thinking for better LLM reasoning, we present the RLVR algorithm SofT-GRPO. As shown in Figure 2, it first samples soft-thinking reasoning paths with controllable randomness using the Gumbel-Softmax technique (Wu et al., 2026). In the following policy update stage, we propose a novel SofT-GRPO loss function with the Gumbel reparameterization trick, which accurately assigns reward credits to token probabilities.

3.1 Group Rollout with Gumbel Noise

As an off-policy RLVR algorithm, SofT-GRPO samples a group of G soft-thinking CoTs for each query $\boldsymbol{Q}$ during rollout and saves them in a replay buffer. A principal challenge for applying RLVR to soft-thinking is that vanilla soft-thinking generates deterministic soft tokens, as each $\boldsymbol{s}_t$ is a fixed weighted sum of embeddings based solely on model probabilities (Eq. (3)). So, to explore **diverse and powerful** soft-thinking reasoning paths in the rollout process, we introduce the Gumbel-Softmax resampling technique (shown in Eq. (4)) as follows:

At step t, let $x_t = [\boldsymbol{Q}, (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{t-1})]$. We apply top-k/top-p filtering and denote the selected tokens by $\mathcal{K}_t$ and their renormalized probabilities by $\widetilde{p}_{\theta,t,i}$, with probabilities outside $\mathcal{K}_t$ set to zero. The rollout is

$$\begin{aligned} g'_{t,i} &= \log \widetilde{p}_{\theta_{\text{old}},t,i} + \epsilon_{t,i}, \quad \epsilon_{t,i} \overset{\text{i.i.d.}}{\sim} \text{Gumbel}(0,1), \quad i \in \mathcal{K}_t, \\ y'_{t,i} &= \frac{\exp(g'_{t,i}/\tau_g)}{\sum_{j\in\mathcal{K}_t} \exp(g'_{t,j}/\tau_g)}, \quad i \in \mathcal{K}_t, \qquad \boldsymbol{s}_t = \sum_{i\in\mathcal{K}_t} y'_{t,i} \boldsymbol{e}_i. \end{aligned} \tag{7}$$

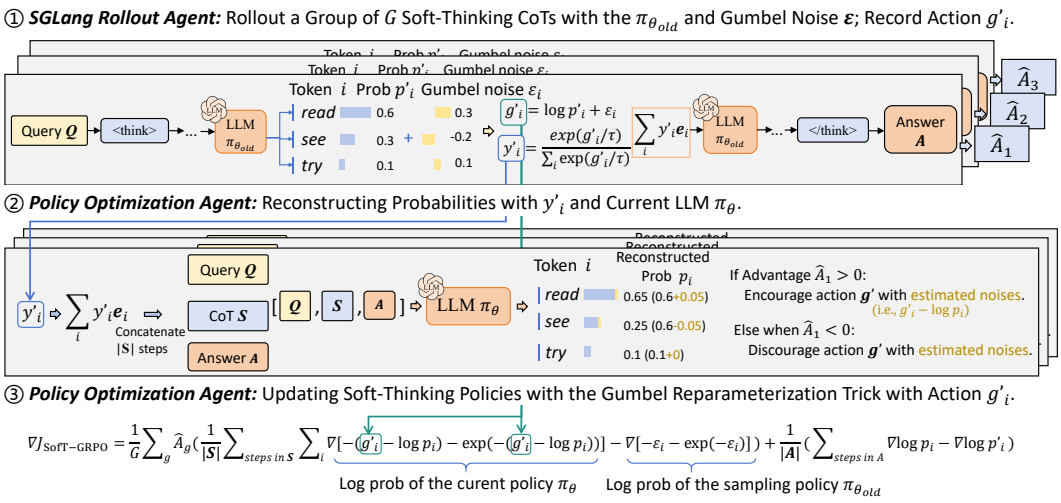


Figure 2: The pipeline of the proposed SofT-GRPO algorithm. In training with Query $\boldsymbol{Q}$, SofT-GRPO first generates a group of G soft-thinking reasoning paths with sampling policy π_{θ_old}, Gumbel noises, and the Gumbel-Softmax technique Wu et al. (2026), transmitting the value of action g'_i and y'_i for the loss calculation afterward. Then, SofT-GRPO reconstructs the probability of the sampled soft-thinking CoTs with y'_i and current policy π_θ. Finally, SofT-GRPO conducts the off-policy proximal policy optimization (Schulman et al., 2017), where loss $\mathcal{J}_{\text{SofT-GRPO}}(\theta)$ is calculated with Gumbel reparameterization.

Here $\boldsymbol{e}_i$ is the embedding of token i, and $\pi_{\theta_{\text{old}}}$ is the sampling policy used in rollout. Using inverse transform sampling (Jang et al., 2016), we sample $\epsilon_{t,i} = -\log(-\log u_{t,i})$ with $u_{t,i} \sim \text{Uniform}(0,1)$. The action is $\boldsymbol{g}'_{t,\mathcal{K}_t} = (g'_{t,i})_{i\in\mathcal{K}_t} \in \mathbb{R}^{|\mathcal{K}_t|}$.

Due to limitations in the amount of fine-tuning data, fine-tuning LLMs with soft-thinking needs to preserve the pre-trained knowledge. So, applying RLVR to soft-thinking should also keep the generated soft tokens $\boldsymbol{s}_t$ **within the pre-trained discrete-token embedding space**. In contrast to noise injection based on other distributions (e.g., Dirichlet or Gaussian), Gumbel-Softmax can better maintain the stability of the soft-token embeddings when providing stochasticity. As established in Lemma 3.1 (see Appendix C.2 for proof), it produces a relaxation that maintains samples close to the discrete token distribution, thus mitigating the risk of generating out-of-vocabulary soft tokens.

Lemma 3.1 (Gumbel-max Trick)**.** *Let* $(p_1, \ldots, p_n)$ *be nonnegative, and* $\epsilon_1, \ldots, \epsilon_n$ *independent samples from* $\text{Gumbel}(0,1)$ *(Maddison et al., 2016),*

$$\Pr\left(j = \arg\max_i (\log p_i + \epsilon_i)\right) = \frac{p_j}{\sum_{i=1}^n p_i}. \tag{8}$$

3.2 Gumbel Reparameterization for Loss Function

In RLVR with discrete tokens (Eq. (1)), the probability of CoT trajectories can be obtained from the output Categorical distributions. Calculating such a probability is impossible in vanilla soft-thinking. To address this, SofT-GRPO proposes using the Gumbel reparameterization trick after obtaining the Gumbel noise $\boldsymbol{\epsilon}$ during the rollout process. So, we propose the MDP for SofT-GRPO as follows:

MDP definition We record the generated soft-token prefix as the state and take $\boldsymbol{g}'_{t,\mathcal{K}_t}$ as the action. Equation (7) maps the action to the next soft token. Let q_θ denote the action density conditional on x_t and $\mathcal{K}_t$. The rollout log-density is

$$\ell_{\text{old},t} := \log q_{\theta_{\text{old}}}(\boldsymbol{g}'_{t,\mathcal{K}_t} \mid x_t, \mathcal{K}_t) = \sum_{i\in\mathcal{K}_t} \left[-\epsilon_{t,i} - e^{-\epsilon_{t,i}}\right]. \tag{9}$$

We provide the derivation in Appendix C.3 and a detailed MDP definition in Appendix D.2. During each policy-update step, K_t is treated as fixed rollout metadata; SofT-GRPO optimizes the conditional

action density $q_\theta(g'_{t,K_t} \mid x_t, K_t)$, and K_t is recomputed only in subsequent rollouts. For off-policy updates, we **retrieve the rollout indices** $\mathcal{K}_t$, reconstruct the soft tokens, and renormalize current probabilities over them to evaluate the saved action:

$$u_{\theta,t,i} = g'_{t,i} - \log \widetilde{p}_{\theta,t,i}, \qquad i \in \mathcal{K}_t,$$
$$\ell_{\theta,t} := \log q_\theta(\boldsymbol{g}'_{t,\mathcal{K}_t} \mid x_t, \mathcal{K}_t) = \sum_{i \in \mathcal{K}_t} \left[-u_{\theta,t,i} - e^{-u_{\theta,t,i}} \right]. \tag{10}$$

Accordingly, $\rho_t(\theta)$ is a conditional density ratio on the stored rollout support K_t. Finally, the SofT-GRPO loss function is as follows:

$$\mathcal{J}_{\text{SofT-GRPO}}(\theta) = \frac{1}{G} \mathbb{E}_{\boldsymbol{Q} \sim \mathcal{D}, \{\boldsymbol{S}\}_{g=1}^{G}, \{\boldsymbol{A}\}_{g=1}^{G} \sim p(\cdot,\cdot|\boldsymbol{Q})}$$
$$\left[\sum_{g=1}^{G} \frac{1}{|\boldsymbol{S}_g| + |\boldsymbol{A}_g|} \sum_{t=1}^{|\boldsymbol{S}_g|+|\boldsymbol{A}_g|} \left(\min \left(p_{g,t} \hat{A}_g, \text{clip}(p_{g,t}, 1-\epsilon, 1+\epsilon) \hat{A}_g \right) - \beta \mathbb{D}_{KL}(\pi_\theta || \pi_{\theta_{\text{ref}}}) \right] \right. \tag{11}$$
$$p_{g,t} = \begin{cases} \frac{\pi_\theta(a_{g,t}|[\boldsymbol{Q},\boldsymbol{S},(a_{g,1},\ldots,a_{g,t-1})])}{\pi_{\theta_{\text{old}}}(a_{g,t}|[\boldsymbol{Q},\boldsymbol{S},(a_{g,1},\ldots,a_{g,t-1})])} & \text{if } t > |\boldsymbol{S}_g| \\ \rho_{g,t}(\theta) & \text{if } t \le |\boldsymbol{S}_g|. \end{cases}$$

where π and π_{ref} denote token probability distributions conditioned on the prefix, and $\rho_{g,t}(\theta) = \exp(\ell_{\theta,g,t} - \ell_{\text{old},g,t})$ is the action-density ratio from Eqs. (9)–(10) for trajectory g. SofT-GRPO evaluates the implied noise for the same stored action, with positive advantage encouraging that action through the clipped objective.

4 Experiments

In this section, we implement the proposed SofT-GRPO algorithm to reinforce the soft-thinking reasoning of three LLMs, including *DeepSeek-R1-Distill-Qwen-1.5B*, *LLaMA-3.2-3B-Instruct*, and *DeepSeek-R1-Distill-Qwen-7B*.

4.1 Implementation Details

Detailed Settings. All our experiments are performed on 8× H200 GPUs in 2 to 3 days. In practice, we follow the general setting in soft-thinking works (Wu et al., 2026), with implementation details in Appendix C.3. In both training and inference, we set top-p as 0.95 and top-k as 5, the temperature of LLMs $\tau = 1$, and the temperature in Gumbel-Softmax $\tau_g = 0.1$.

Training & Testing Settings. We employ DeepScaler (Luo et al., 2025) as the training dataset, which contains 40,315 queries. In implementing SofT-GRPO, we use SGLang (Zheng et al., 2024) for the rollout process and the verl-0.4.x framework (Sheng et al., 2024) for RLVR.

We use five numerical reasoning benchmarks as in-domain test sets (AIME2024, AIME2025, AMC23, MATH-500, and GSM8K). We also employ a scientific reasoning benchmark (GPQA Diamond) and two code benchmarks (HumanEval and MBPP) for out-of-domain evaluation. The maximum generation length is 8192 in training and 32768 in testing. Test answers are verified using the Math Verify package (Kydlíček, 2025). All experiments are implemented on a node of 8× NVIDIA H200 GPUs (141 GB VRAM each), and detailed parameters are shown in Appendix D.

Baselines. We use both **discrete-token CoT** reasoning and **soft-thinking** reasoning as baselines for SofT-GRPO. For discrete-token CoT, we include **discrete-token GRPO** trained on the same dataset and **base LLMs** without fine-tuning as baselines. For baselines with soft-thinking, we implement the soft-thinking reasoning pattern with the Gumbel-Softmax technique, using the best temperature setting $\tau_g = 0.5$ proposed in (Wu et al., 2026). Under both patterns, we follow the parameters in (Wu et al., 2026), setting the temperature τ of LLMs to 0.6 (we discuss this setting in Appendix E.6), top-p to 0.95 and top-k to 30.

Metrics. We adopt the general metrics of Mean@32 (average correctness over 32 generated responses per problem), Pass@16, and Pass@32, which measure empirical answer coverage within the first 16 and 32 generated responses, respectively, for each evaluated checkpoint.

Table 1: Experiment results of baselines and the proposed SofT-GRPO on five numerical reasoning benchmarks. We cover 3 LLMs from 1.5B to 7B and two reasoning patterns, i.e., discrete-token CoT and soft-thinking reasoning. @1 metrics denote the Mean@32 values, where we run each method 32 times on the dataset for average Pass@1 accuracies. @16 and @32 denote the Pass@16 and Pass@32 values on the dataset, respectively. We multiply all the results by 100 to highlight the differences between results. The best result on each metric and dataset is underlined, the best average result is bolded, and the second-best average result is shaded.

Method	AIME24			AIME25			AMC23			MATH500			GSM8K			Avg.		
	@1	@16	@32	@1	@16	@32	@1	@16	@32	@1	@16	@32	@1	@16	@32	@1	@16	@32
DeepSeek-R1-Distill-Qwen-1.5B																		
Discrete-Token CoT Reasoning Pattern																		
Base	30.6	70.0	73.3	23.0	46.7	53.3	70.7	92.5	95.0	84.6	97.8	97.8	81.5	95.8	96.7	58.09	80.54	83.23
+ GRPO	31.8	66.7	76.7	25.3	46.7	46.7	77.3	95.0	95.0	87.1	97.4	97.8	84.9	95.1	95.8	61.28	80.16	82.39
Soft Thinking Reasoning Pattern																		
Base	27.3	66.7	70.0	23.8	46.7	53.3	69.9	95.0	95.0	79.4	93.2	96.6	81.0	94.6	97.1	56.28	79.23	82.41
+ GRPO	29.2	70.0	73.3	25.4	46.7	53.3	75.8	95.0	95.0	86.3	96.8	98.2	84.9	95.6	96.4	60.31	80.81	83.26
+ SofT-GRPO	32.6	76.7	80.0	26.1	50.0	53.3	76.4	97.5	97.5	86.3	97.4	98.0	85.5	96.1	97.0	**61.39**	**83.54**	**85.18**
LLaMA-3.2-3B-Instruct																		
Discrete-Token CoT Reasoning Pattern																		
Base	4.4	20.0	26.7	0.3	3.3	3.3	18.3	65.0	75.0	38.1	75.6	84.0	67.9	92.1	94.6	25.79	51.20	56.72
+ GRPO	7.3	23.3	26.7	0.5	3.3	3.3	27.3	62.5	67.5	48.3	77.2	82.6	79.6	95.4	96.5	32.60	52.35	55.32
Soft Thinking Reasoning Pattern																		
Base	3.4	16.7	16.7	0.2	6.7	6.7	17.6	70.0	77.5	36.7	76.0	81.4	66.9	91.6	94.7	24.96	52.18	55.39
+ GRPO	8.0	20.0	23.3	0.7	3.3	10.0	27.3	70.0	75.0	47.8	76.8	81.8	79.2	94.8	96.3	32.60	53.00	**57.28**
+ SofT-GRPO	7.7	23.3	26.7	0.3	10.0	10.0	31.3	67.5	67.5	47.2	77.6	83.4	77.6	96.4	97.7	**32.83**	**54.96**	57.06
DeepSeek-R1-Distill-Qwen-7B																		
Discrete-Token CoT Reasoning Pattern																		
Base	55.7	80.0	80.0	39.4	66.7	66.7	89.9	97.5	97.5	93.6	98.8	99.2	89.5	96.7	97.0	73.62	87.94	88.07
+ GRPO	54.0	80.0	80.0	40.4	66.7	70.0	89.2	95.0	95.0	93.5	98.4	99.0	91.4	96.1	96.6	73.69	87.24	88.12
Soft Thinking Reasoning Pattern																		
Base	55.3	80.0	83.3	39.2	66.7	70.0	90.2	95.0	97.5	93.3	98.8	99.0	89.3	96.6	97.0	73.44	87.41	89.38
+ GRPO	55.5	80.0	80.0	37.8	63.3	66.7	90.1	97.5	97.5	93.6	98.6	99.0	91.8	96.7	96.9	73.76	87.23	88.01
+ SofT-GRPO	54.7	80.0	83.3	40.4	66.7	73.3	89.6	97.5	97.5	93.3	98.6	99.0	92.1	97.1	97.6	**74.04**	**87.98**	**90.16**

4.2 Main Result

As shown in Table 1, SofT-GRPO can lead to a clear and consistent improvement from the No-Finetune results under the soft-thinking reasoning pattern. Moreover, SofT-GRPO outperforms applying soft-thinking on top of the discrete-token trained GRPO on average, showing improved benchmark-averaged accuracy. For Pass@1, SofT-GRPO has slightly higher benchmark averages than discrete-token GRPO across the three model sizes (+0.23 percentage points), with comparable single-sample accuracy overall. In contrast, comparing the +GRPO results on the discrete-token and soft-thinking patterns, performing soft-thinking on top of the discrete-token-trained GRPO gives inconsistent improvement.

As observed in (Yue et al., 2025), GRPO can decrease Pass@K performance relative to the No-Finetune model, even though it improves Pass@1. We can observe the same result in Table 1. In contrast, SofT-GRPO raises average Pass@16 by +2.24 percentage points and average Pass@32 by +2.19 percentage points relative to discrete-token GRPO. Appendix E.6 examines additional baseline temperatures on the 1.5B model. Appendix F examines the action score and the training entropy.

We further test whether these observations persist across stronger baselines, objectives, and model families. Appendix E.1 provides the complete settings and results: SofT-GRPO outperforms the Pass@K-oriented SimKO (Peng et al., 2025) baseline (Table 9), generalizes to three additional non-R1-distilled models (Qwen2.5-1.5B, Qwen2.5-1.5B-Math, and Qwen3-1.7B) (Table 10), and remains effective when SofT-GRPO is combined with Dr.GRPO Liu et al. (2025b) (Table 11).

4.3 Comparison in Out-of-Domain Datasets

Besides numerical reasoning, we conduct out-of-domain experiments to evaluate the general reasoning ability of SofT-GRPO. As shown in Table 2, the 1.5B model fine-tuned with SofT-GRPO on numerical questions obtains higher aggregate averages than discrete-token GRPO across GPQA Diamond, HumanEval, and MBPP. This provides evidence of transfer beyond the mathematical training domain. We additionally train on executable programming problems and evaluate HumanEval, MBPP, and KlearReasoner under both partial-credit and strict exact-correctness rewards; the detailed settings and results are reported in Tables 12 and 13 of Appendix E.1.

Table 2: Average accuracies on out-of-domain datasets. We cover GPQA Diamond, HumanEval, and MBPP. @1 metrics denote the Mean@32 values, where we run the methods 32 times on the dataset to average Pass@1 accuracies. @8, @16, and @32 denote the Pass@8, Pass@16, and Pass@32 values on the dataset, respectively.

Method	GPQA Diamond				HumanEval				MBPP				Avg.			
	@1	@8	@16	@32	@1	@8	@16	@32	@1	@8	@16	@32	@1	@8	@16	@32
DeepSeek-R1-Distill-Qwen-1.5B																
Discrete-Token CoT Reasoning Pattern																
Base	36.7	84.3	92.4	96.0	68.1	87.2	90.2	93.9	65.5	84.8	89.1	91.1	56.77	85.45	90.59	93.64
+ GRPO	35.4	77.8	88.4	93.4	72.2	90.9	92.7	94.5	68.1	85.2	87.2	90.0	58.56	84.62	89.41	92.65
Soft Thinking Reasoning Pattern																
Base	36.0	83.8	91.9	97.0	67.2	89.6	91.5	92.7	64.7	84.4	86.0	88.7	55.98	85.97	89.79	92.79
+ GRPO	36.5	81.3	91.4	94.4	71.8	89.0	92.7	95.1	68.1	84.8	87.9	90.3	58.82	85.05	**90.68**	93.28
+ SofT-GRPO	37.3	82.8	89.9	95.5	71.2	88.4	91.5	94.5	68.8	88.7	90.3	91.4	**59.08**	**86.65**	90.54	**93.80**

Table 3: Using majority voting for AIME2024, AIME2025, AMC23, MATH500, and GSM8K. For each method, @1 denotes the Pass@1 result reported in Table 1, averaged over 32 independent runs, while M@16 and M@32 denote Major@16 and Major@32, respectively.

Method	AIME2024			AIME2025			AMC23			MATH500			GSM8K			Avg.		
	@1	M@16	M@32	@1	M@16	M@32	@1	M@16	M@32	@1	M@16	M@32	@1	M@16	M@32	@1	M@16	M@32
DeepSeek-R1-Distill-Qwen-1.5B																		
Discrete-Token CoT Reasoning Pattern																		
Base	30.6	56.7	60.0	23.0	36.7	40.0	70.7	90.0	95.0	84.6	92.6	92.4	81.5	89.5	89.5	58.1	73.1	75.4
+ GRPO	31.8	53.3	50.0	25.3	36.7	30.0	77.3	92.5	92.5	87.1	91.6	91.4	84.9	90.1	90.4	61.3	72.8	70.9
Soft Thinking Reasoning Pattern																		
Base	27.3	56.7	60.0	23.8	33.3	33.3	69.9	95.0	92.5	79.4	88.8	89.8	81.0	89.3	89.3	56.3	72.6	73.0
+ GRPO	29.2	43.3	46.7	25.4	30.0	30.0	75.8	92.5	92.5	86.3	91.0	91.4	84.9	90.5	90.8	60.3	69.5	70.3
+ SofT-GRPO	32.6	60.0	63.3	26.1	33.3	36.7	76.4	92.5	95.0	86.3	92.2	92.0	85.5	90.6	90.5	**61.4**	**73.7**	**75.5**

4.4 Boosting SofT-GRPO with Majority Voting

To further exploit the advantage of SofT-GRPO on Pass@32, in this subsection, we design to boost SofT-GRPO with majority voting (Chen et al., 2024). We compare with their **reported results** under the same training dataset and LLM. As shown in Table 3, SofT-GRPO with majority voting can outperform No-Finetune LLM and LLM fine-tuned with discrete-token GRPO across Major@16 (the accuracy of the most common answer in 16 runs) and Major@32. The results show that SofT-GRPO-fine-tuned LLMs can be strengthened into better reasoning solvers with majority voting.

Table 4: Reference comparison with published Soft Tokens results (Butt et al., 2025). Soft Tokens was not reproduced in our pipeline.

Method	MATH500		GSM8K	
	@1	@32	@1	@32
LLaMA-3.2-3B-Instruct				
Discrete-Token CoT Reasoning Pattern				
Base	38.1	84.0	67.9	94.6
+ GRPO	48.3	82.6	79.6	96.5
Soft Thinking Reasoning Pattern				
+ Soft Tokens*	41.3	77.9	75.5	95.2
+ SofT-GRPO	47.2	83.4	77.6	97.7

4.5 Comparison to the RLVR method in (Butt et al., 2025)

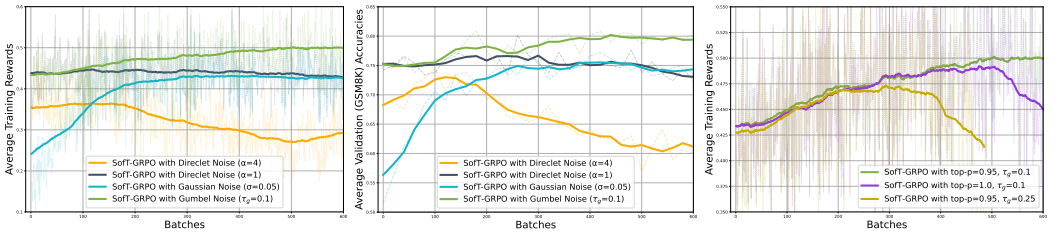
This subsection compares SofT-GRPO with the method in (Butt et al., 2025), which is an existing RLVR algorithm for the soft-thinking pattern. We refer to this method as Soft Tokens in Table 4, where Soft Tokens* represents their reported results fine-tuned under soft-thinking. The proposed SofT-GRPO can demonstrate higher scores than those reported in (Butt et al., 2025). The Soft Tokens scores are quoted from its publication and reflect that study's training and evaluation settings.

4.6 Comparison in the Token Efficiency

In addition to accuracy, token efficiency is an important metric for LLMs. As shown in Appendix E.5, SofT-GRPO reduces thinking length compared to No-Finetune, with model-dependent changes relative to discrete-token GRPO. Notably, SofT-GRPO leads to a clear reduction in thinking length for the *LLaMA-3.2-3B-Instruct* model, while outputs are longer on the two R1-distilled models. The current implementation takes approximately 2–3$\times$ the training wall-clock time of discrete-token GRPO (Appendix D.3).

Table 5: Ablation studies on the noise added in the proposed SofT-GRPO. We highlight the performance differences on the ablation variants (i.e., adding Dirichlet noises or Gaussian noises).

Method	AIME2024			AIME2025			AMC23			MATH500			GSM8K			Avg.		
	@1	@16	@32	@1	@16	@32	@1	@16	@32	@1	@16	@32	@1	@16	@32	@1	@16	@32
DeepSeek-R1-Distill-Qwen-1.5B																		
Soft Thinking																		
Base	27.3	66.7	70.0	23.8	46.7	53.3	69.9	95.0	95.0	79.4	93.2	96.6	81.0	94.6	97.1	56.3	79.2	82.4
+ GRPO	29.2	70.0	73.3	25.4	46.7	53.3	75.8	95.0	95.0	86.3	96.8	98.2	84.9	95.6	96.4	60.3	80.8	83.3
+ SofT-GRPO (Original, Gumbel)	32.6	76.7	80.0	26.1	50.0	53.3	76.4	97.5	97.5	86.3	97.4	98.0	85.5	96.1	97.0	61.4	83.5	85.2
+ SofT-GRPO (Dirichlet Noise)	25.4	60.0	70.0	22.9	56.7	63.3	68.4	97.5	97.5	83.6	97.0	97.8	83.6	97.0	97.8	56.8	81.6	85.3
Δ vs. Gumbel	-7.2	-16.7	-10.0	-3.2	+6.7	+10.0	-8.0	0.0	0.0	-2.7	-0.4	-0.2	-1.8	+0.9	+0.8	-4.6	-1.9	+0.1
+ SofT-GRPO (Gaussian Noise)	21.9	60.0	73.0	20.0	40.0	43.3	65.6	97.5	97.5	79.9	95.2	96.6	81.8	94.5	95.5	53.8	77.4	81.2
Δ vs. Gumbel	-10.7	-16.7	-7.0	-6.1	-10.0	-10.0	-10.8	0.0	0.0	-6.5	-2.2	-1.4	-3.7	-1.7	-1.5	-7.6	-6.1	-4.0



(a) Training reward curve of variants on added noises

(b) Validation accuracy curve of variants on added noises

(c) Training reward curve of different hyper-parameters

Figure 3: Smoothed training or validation curves of ablation studies (the dashed background contains the data points). (a) discusses the setting of adding Gumbel noise in SofT-GRPO. (b,c) discusses the setting of top-p=0.95 and the Gumbel-Softmax temperature $\tau_g = 0.1$.

5 Discussion

In this section, we conduct ablation studies and case studies on the proposed SofT-GRPO algorithm.

Ablation on the Added Noise. In SofT-GRPO, we employ the Gumbel-Softmax technique for controllable stochasticity. As discussed in Section 2.2, adding Dirichlet noise can be another choice, so we compare the performance of the original SofT-GRPO and its two variants, adding Dirichlet noise or Gaussian noise to probabilities p_i in training. The reward curve in training is shown in Figure 3(a), the validation curve is shown in Figure 3(b), and the final performance of variants is shown in Table 5. Gumbel obtains the highest average Pass@1 and Pass@16 among these variants. Dirichlet has lower Pass@1 but comparable Pass@32 (85.3 versus 85.2). The Gaussian variant here perturbs probabilities; Soft Tokens perturbs embeddings.

Ablation on Hyperparameters. SofT-GRPO sets the top-p to 0.95 and the Gumbel temperature to $\tau_g = 0.1$. To demonstrate the reason for these settings, in Figure 3(c), we compare the training reward curve of the original SofT-GRPO with two variants (varying top-p to 1.0 or τ_g to 0.25). Results show that both variants will cause a collapse in the training process. As analyzed in Appendix E.7, the two variants will cause a substantial increase in the KL divergence between π_θ and $\pi_{\theta_{ref}}$. These runs show sensitivity to the sampling configuration.

Visualization: Soft-Thinking after SofT-GRPO In Appendix H, we provide a visualization of the soft-thinking reasoning path after SofT-GRPO fine-tuning. As shown in Figure 11, dominant-token projections are readable, and some intermediate steps place appreciable mass on multiple tokens, including Yeah and No.

6 Conclusion

This paper presents an RLVR algorithm, SofT-GRPO, to reinforce LLMs under the soft-thinking reasoning pattern. It integrates controllable stochasticity with Gumbel-Softmax and updates the soft-thinking policy with Gumbel reparameterization. Across the evaluated settings, it improves aggregate multi-sample coverage over the tested discrete-token GRPO baselines while retaining comparable Pass@1. It can also be boosted into a better solver with majority voting. Future work includes extending SofT-GRPO to multimodal LLMs, such as Vision Language Models.

AI USE STATEMENT

In this work, we used generative AI tools to assist in the writing of proofs. We have not used generative AI tools to generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, design or provide feedback on research methodology or experiments, assist with translation, clean and reformat datasets, or interpret results, and qualitative and thematic data analysis is not applicable to this work. Additionally, we used generative AI tools to create or modify scientific figures or images, create artifacts, and edit the research paper to improve readability.

We have reviewed all AI-assisted work. AI-assisted proofs were checked by the authors for correctness, and AI-assisted figures, images, artifacts, and manuscript edits were reviewed and tested as appropriate before inclusion. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

For reproducibility, we additionally provide an anonymous source-code release at https://anonymous.4open.science/r/SofT-GRPO-master-A512. Detailed experimental configurations and optimization hyperparameters are provided in Appendix D. Appendix B reports the prompts, while Appendix G documents external resources, repositories, and licensing information.

REFERENCES

Daman Arora and Andrea Zanette. Training language models to reason efficiently. *arXiv preprint arXiv:2502.04463*, 2025.

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

Lingjiao Chen, Jared Quincy Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei Zaharia, and James Zou. Are more llm calls all you need? towards scaling laws of compound inference systems. *arXiv preprint arXiv:2403.02419*, 2024.

Xinghao Chen, Anhao Zhao, Heming Xia, Xuan Lu, Hanlin Wang, Yanjun Chen, Wei Zhang, Jian Wang, Wenjie Li, and Xiaoyu Shen. Reasoning beyond language: A comprehensive survey on latent chain-of-thought reasoning, 2025. URL https://arxiv.org/abs/2505.16782.

Jeffrey Cheng and Benjamin Van Durme. Compressed chain of thought: Efficient reasoning through dense representations. *arXiv preprint arXiv:2412.13171*, 2024.

Yinlam Chow, Guy Tennenholtz, Izzeddin Gur, Vincent Zhuang, Bo Dai, Sridhar Thiagarajan, Craig Boutilier, Rishabh Agarwal, Aviral Kumar, and Aleksandra Faust. Inference-aware fine-tuning for best-of-n sampling in large language models. *arXiv preprint arXiv:2412.15287*, 2024.

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.

Muzhi Dai, Chenxu Yang, and Qingyi Si. S-grpo: Early exit via reinforcement learning in reasoning models. *arXiv preprint arXiv:2505.07686*, 2025.

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

Sicheng Feng, Gongfan Fang, Xinyin Ma, and Xinchao Wang. Efficient reasoning models: A survey. *arXiv preprint arXiv:2504.10903*, 2025.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. In *International Conference on Learning Representations*, volume 2024, pp. 27896–27923, 2024.

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

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

Zhezheng Hao, Hong Wang, Haoyang Liu, Jian Luo, Jiarui Yu, Hande Dong, Qiang Lin, Can Wang, and Jiawei Chen. Rethinking entropy interventions in rlvr: An entropy change perspective. *arXiv preprint arXiv:2510.10150*, 2025.

Adit Jain and Brendan Rappazzo. Learning to reason with mixture of tokens. *arXiv preprint arXiv:2509.21482*, 2025.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Hynek Kydlíček. Math-Verify: Math Verification Library, 2025. URL https://github.com/huggingface/math-verify.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Jiacai Liu, Yingru Li, Yuqian Fu, Jiawei Wang, Qian Liu, and Zhuo Jiang. When speed kills stability: Demystifying RL collapse from the training-inference mismatch, September 2025a. URL https://richardli.xyz/rl-collapse.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025b.

Zihe Liu, Jiashun Liu, Yancheng He, Weixun Wang, Jiaheng Liu, Ling Pan, Xinyu Hu, Shaopan Xiong, Ju Huang, Jian Hu, et al. Part i: Tricks or traps? a deep dive into rl for llm reasoning. *arXiv preprint arXiv:2508.08221*, 2025c.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl, 2025. Notion Blog.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Sadegh Mahdavi, Muchen Li, Kaiwen Liu, Renjie Liao, and Christos Thrampoulidis. Beyond accuracy: A policy gradient reweighting approach for pass@ k maximization in llms. In *2nd AI for Math Workshop@ ICML 2025*, 2025.

Ruotian Peng, Yi Ren, Zhouliang Yu, Weiyang Liu, and Yandong Wen. Simko: Simple pass@ k policy optimization. *arXiv preprint arXiv:2510.14807*, 2025.

Penghui Qi, Zichen Liu, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Optimizing anytime reasoning via budget relative policy optimization. *arXiv preprint arXiv:2505.13438*, 2025a.

Penghui Qi, Zichen Liu, Xiangxin Zhou, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Defeating the training-inference mismatch via fp16. *arXiv preprint arXiv:2510.26788*, 2025b.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. Codi: Compressing chain-of-thought into continuous space via self-distillation. *arXiv preprint arXiv:2502.21074*, 2025.

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

Zayne Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. To cot or not to cot? chain-of-thought helps mainly on math and symbolic reasoning. *arXiv preprint arXiv:2409.12183*, 2024.

DiJia Su, Hanlin Zhu, Yingchen Xu, Jiantao Jiao, Yuandong Tian, and Qinqing Zheng. Token assorted: Mixing latent and text tokens for improved language model reasoning. *arXiv preprint arXiv:2502.03275*, 2025.

Wenhui Tan, Jiaze Li, Jianzhong Ju, Zhenbo Luo, Jian Luan, and Ruihua Song. Think silently, think fast: Dynamic latent compression of llm reasoning chains. *arXiv preprint arXiv:2505.16552*, 2025.

Christian Walder and Deep Karkhanis. Pass@ k policy optimization: Solving harder reinforcement learning problems. *arXiv preprint arXiv:2505.15201*, 2025.

Chen Wang, Zhaochun Li, Jionghao Bai, Yuzhi Zhang, Shisheng Cui, Zhou Zhao, and Yue Wang. Arbitrary entropy policy optimization breaks the exploration bottleneck of reinforcement learning. *arXiv preprint arXiv:2510.08141*, 2025a.

Jianwei Wang, Ziming Wu, Fuming Lai, Shaobing Lian, and Ziqian Zeng. Synadapt: Learning adaptive reasoning in large language models via synthetic continuous chain-of-thought. *arXiv preprint arXiv:2508.00574*, 2025b.

Xilin Wei, Xiaoran Liu, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Jiaqi Wang, Xipeng Qiu, and Dahua Lin. Sim-cot: Supervised implicit chain-of-thought. *arXiv preprint arXiv:2509.20317*, 2025.

Junhong Wu, Jinliang Lu, Zixuan Ren, Gangqiang Hu, Zhi Wu, Dai Dai, et al. Llms are single-threaded reasoners: Demystifying the working mechanism of soft thinking. In *International Conference on Learning Representations*, volume 2026, pp. 4533–4548, 2026.

Xiao-Kun Wu, Min Chen, Wanyi Li, Rui Wang, Limeng Lu, Jia Liu, Kai Hwang, Yixue Hao, Yanru Pan, Qingguo Meng, et al. Llm fine-tuning: Concepts, opportunities, and challenges. *Big Data and Cognitive Computing*, 9(4):87, 2025.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. Softcot: Soft chain-of-thought for efficient reasoning with llms, 2025a.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. Softcot++: Test-time scaling with soft chain-of-thought reasoning. *arXiv preprint arXiv:2505.11484*, 2025b.

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.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? In *NeurIPS*, 2025.

Eric Zelikman, Georges Raif Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah Goodman. Quiet-star: Language models can teach themselves to think before speaking. In *First Conference on Language Modeling*, 2024. URL https://arxiv.org/abs/2403.09629.

Jintian Zhang, Yuqi Zhu, Mengshu Sun, Yujie Luo, Shuofei Qiao, Lun Du, Da Zheng, Huajun Chen, and Ningyu Zhang. Lightthinker: Thinking step-by-step compression. *arXiv preprint arXiv:2502.15589*, 2025a.

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, Shuohang Wang, Yelong Shen, and Xin Eric Wang. Soft thinking: Unlocking the reasoning potential of llms in continuous concept space. *arXiv preprint arXiv:2505.15778*, 2025b.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Livia Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37: 62557–62583, 2024.

Zhi Zheng and Wee Sun Lee. Reasoning-cv: Fine-tuning powerful reasoning llms for knowledge-assisted claim verification. *arXiv preprint arXiv:2505.12348*, 2025.

Rui-Jie Zhu, Tianhao Peng, Tianhao Cheng, Xingwei Qu, Jinfa Huang, Dawei Zhu, Hao Wang, Kaiwen Xue, Xuanliang Zhang, Yong Shan, et al. A survey on latent reasoning. *arXiv preprint arXiv:2507.06203*, 2025.

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

APPENDIX CONTENTS

1. **Related Work** 15
   (a) RLVR on Discrete-Token CoT Reasoning 15
   (b) Latent Reasoning 15
   (c) The Working Mechanism of Soft-Thinking 16
2. **Prompt of Language Reasoning and Latent Generation** 17
3. **Motivation and Theoretical Proof** 18
   (a) Motivation: Mismatch in (Butt et al., 2025) 18
   (b) Proof for Lemma 3.1 19
   (c) Derivation of Eq. (11), Eq. (9) and Eq. (10) 20
4. **Detailed Settings for Training & Testing** 21
   (a) Hyperparameter Setting 21
   (b) MDP Definition 22
   (c) Time Complexity & Consumption Analysis 23
5. **Supplementary of Experiments** 24
   (a) Comparison with SimKO 24
   (b) Generalization to Three Additional Base Models 24
   (c) Combining the Soft-Trajectory Likelihood with Dr.GRPO 25
   (d) Code-Reasoning Experiments 25
   (e) Supplementary of Token Efficiency 26
   (f) Experiments under Different Temperatures 27
   (g) Supplementary Ablation on Hyper-Parameters 28
   (h) Pass@K up to K=1024 29
   (i) P-values for Significance on Pass@K 30
6. **Discussion & Analysis on Results** 31
7. **Baselines & Datasets & Licenses** 32
   (a) Baselines 32
   (b) Datasets 32
   (c) Inference Framework 32
   (d) Licenses 32
8. **Visualization** 33

A Related Work

In this section, we will discuss recent developments on discrete-token RLVR, a similar domain of soft-thinking, the latent reasoning methods, and the working mechanism of soft-thinking.

A.1 RLVR on Discrete-Token CoT Reasoning

Deepseek-R1-zero (Guo et al., 2025) has demonstrated remarkable performance with discrete-token RLVR. And there has been a wide collection of RLVR methods for discrete-token CoT policy optimization, such as GRPO (Shao et al., 2024), Dr. GRPO (Liu et al., 2025b), DAPO (Yu et al., 2025), and Lite PPO (Liu et al., 2025c). To further improve these algorithms towards some specific goals, recent works focus on developing RLVR methods for efficient responses (Feng et al., 2025) or controlling the entropy in CoT generation for better exploration-exploitation balance (Hao et al., 2025).

To generate concise discrete-token CoTs, (Arora & Zanette, 2025) modifies the reward function by adding penalties on the generation length. (Qi et al., 2025a) and (Dai et al., 2025) turn to formulate the fine-tuning process for concise language CoT as a multi-objective optimization task with the trade-off between token efficiency and accuracy. They truncate the generation of discrete-token CoT at several thinking budgets and optimize the overall performance over them.

RLVR methods with entropy control try to make a better balance between exploration and exploitation. They try to control the entropy in the training process to stimulate the exploration within the groups of CoTs in GRPO (Wang et al., 2025a).

GRPO is the primary RLVR pipeline studied in this paper. Appendix E.1 additionally evaluates integration with Dr.GRPO. Other objective combinations remain directions for future work.

A.2 Latent Reasoning

Similar to the soft-thinking pattern, latent reasoning methods pass continuous vectors between LLM steps. These methods fully decouple the reasoning process from explicit natural language (which soft-thinking does not do) and perform inference in the hidden space of the model. Generally, latent CoT methods are often diverse from each other and can be mainly divided into token-wise auto-regressive methods and auxiliary strategies (Chen et al., 2025; Zhu et al., 2025). Token-wise auto-regressive methods transform the reasoning process into 'soft thoughts' with dense latent embeddings (Hao et al., 2024) or specialized tokens (e.g., pause (Goyal et al., 2024; Zelikman et al., 2024)). These methods focus on transferring the original reasoning policy in the language domain to a latent embedding space, including curriculum learning (e.g., Coconut (Hao et al., 2024), LightThinker (Zhang et al., 2025a), SIM-COT (Wei et al., 2025)), self-distillation (e.g., CODI (Shen et al., 2025)), and one-shot compression (e.g., CoLaR (Tan et al., 2025), SynAdapt (Wang et al., 2025b), Latent-SFT Deng et al. (2025)). Auxiliary strategies (e.g., SoftCoT (Xu et al., 2025a)) generate latent embeddings from an auxiliary module and inject them into the frozen main model (Cheng & Van Durme, 2024; Xu et al., 2025b; Su et al., 2025).

These latent reasoning methods explore different trade-offs between reasoning accuracy, supervision, and generation length.

Compared to latent reasoning methods mainly aiming at better efficiency, the proposed SofT-GRPO aims at reinforcing the accuracy of the soft-thinking pattern using verifiable rewards across the evaluated reasoning tasks.

A.3 THE WORKING MECHANISM OF SOFT-THINKING

As illustrated in (Zhang et al., 2025b), soft-thinking represents each intermediate step as a weighted mixture of token embeddings.

- **Continuous Concept Space:** Soft Thinking replaces discrete token choices with "concept tokens," which are probability-weighted mixtures over all token embeddings. The mixture retains contributions from multiple vocabulary embeddings.
- **Mixture Representation:** Soft Thinking passes a weighted embedding mixture to the next model step.

Remarkably, LLMs can directly interpret and process soft tokens without the need for additional fine-tuning. By construction, these soft tokens lie in the **convex hull** formed by the model's embedding vectors.

Subsequent model computations are nonlinear, so processing an embedding mixture generally differs from marginalizing over discrete reasoning paths. The mixture interpretation describes the representation, while its empirical behavior depends on the model and sampling configuration.

B PROMPT OF LANGUAGE REASONING AND LATENT GENERATION

In this part, we show the prompt we adopt for reasoning problems, including the in-domain numerical reasoning, out-of-domain GPQA reasoning, and code reasoning. We inherit the prompts in (Zhang et al., 2025b) for out-of-domain benchmarks. We use the same prompt for both the soft-thinking reasoning pattern and the discrete-token reasoning pattern.

Prompt for Numerical Reasoning.

user
{Question} Let's think step by step and output the final answer within \boxed{}

Prompt for GPQA Reasoning.

user
Please solve the following multiple-choice question. Please show your choice in the answer field with only the choice letter, e.g., "answer": "C".
{Question}

Prompt for Code Reasoning (HumanEval).

user
Please solve the programming task below in Python. Code should be wrapped in a markdown code block.
```python
{Question}
```

Prompt for Code Reasoning (MBPP).

user
Please solve the programming task with test cases below in Python. Make sure your code satisfies the following requirements:
1. The function name and signature must match exactly as specified in the test cases.
2. Your code should be wrapped in a markdown code block without including any test cases.
Task:
{Question}
Test Cases:
```python
{TestCases}
```

The blue part represents the specific question (query $\boldsymbol{Q}$), the brown part represents the possible test cases provided in the MBPP code reasoning benchmark. We use the code provided in (Zhang et al., 2025b) in the verification process of the responses.

C Motivation and Theoretical Proof

C.1 Motivation: Mismatch in (Butt et al., 2025)

The Gaussian formulation of (Butt et al., 2025) defines a density over embedding-space actions:

$$q_\theta(\hat{\boldsymbol{s}} \mid \boldsymbol{p}) = \mathcal{N}(\hat{\boldsymbol{s}}; \boldsymbol{E}^\top \boldsymbol{p}, \sigma^2 I_d), \tag{12}$$

where $\boldsymbol{E} \in \mathbb{R}^{|\mathcal{T}| \times d}$ is the embedding matrix. For fixed $\boldsymbol{E}$, its gradient is

$$\nabla_{\boldsymbol{p}} \log q_\theta(\hat{\boldsymbol{s}} \mid \boldsymbol{p}) = \frac{1}{\sigma^2} \boldsymbol{E}(\hat{\boldsymbol{s}} - \boldsymbol{E}^\top \boldsymbol{p}). \tag{13}$$

Different token mixtures can project to the same embedding mean and thus induce the same action distribution. This is a non-identifiability property of the parameterization; the Gaussian likelihood and its gradient remain well-defined.

Under a k-sparse mixture constraint with $k - 1 < d$, the possible mixtures form a finite union of simplices of dimension at most $k - 1$. This set has zero Lebesgue measure in $\mathbb{R}^d$, so an isotropic Gaussian perturbation with $\sigma > 0$ almost surely leaves it. SofT-GRPO constructs actions in perturbed-log-probability space and maps them to normalized embedding mixtures. The two parameterizations therefore have different action supports and inductive biases.

C.2 Proof for Lemma 3.1

Lemma 3.1 (Gumbel-max Trick) *Let $(p_1, \ldots, p_n)$ be nonnegative real numbers, not all zero. Let $g_1, \ldots, g_n$ be independent samples from* $\mathrm{Gumbel}(0, 1)$*. Then,*

$$\Pr\left(j = \arg\max_i (g_i + \log p_i)\right) = \frac{p_j}{\sum_{i=1}^n p_i}. \tag{14}$$

Proof. For any $j \in \{1, \ldots, n\}$,

$$\begin{aligned}
&\Pr\left(j = \arg\max_i (g_i + \log p_i)\right) \\
&= \Pr\left(g_j + \log p_j \geq g_i + \log p_i,\ \forall i \neq j\right) && \text{(event reformulation)} \\
&= \int_{-\infty}^{+\infty} \prod_{i \neq j} \Pr\left(g_i \leq g_j + \log p_j - \log p_i\right) f_{g_j}(g_j)\, dg_j, && \text{(conditioning on } g_j\text{)}
\end{aligned}$$

where $f_{g_j}(g) = e^{-g - \exp(-g)}$ is the PDF of the standard Gumbel distribution.

$\Pr(g_i \leq t) = F_{\mathrm{Gumbel}}(t) = \exp\left(-e^{-t}\right)$, so

$$\begin{aligned}
&\prod_{i \neq j} \Pr\left(g_i \leq g_j + \log p_j - \log p_i\right) \\
&= \prod_{i \neq j} \exp\left(-e^{-(g_j + \log p_j - \log p_i)}\right) && \text{(Gumbel CDF)} \\
&= \exp\left(-\sum_{i \neq j} e^{-(g_j + \log p_j - \log p_i)}\right) \\
&= \exp\left(-e^{-g_j} \sum_{i \neq j} \frac{p_i}{p_j}\right).
\end{aligned}$$

By combining exponents, the total probability is

$$\begin{aligned}
&\int_{-\infty}^{+\infty} e^{-g_j} e^{-e^{-g_j}} \exp\left(-e^{-g_j} \sum_{i \neq j} \frac{p_i}{p_j}\right) dg_j \\
&= \int_{-\infty}^{+\infty} e^{-g_j} \exp\left(-e^{-g_j}\left[1 + \sum_{i \neq j} \frac{p_i}{p_j}\right]\right) dg_j. && (15)
\end{aligned}$$

Let

$$S = 1 + \sum_{i \neq j} \frac{p_i}{p_j} = \frac{\sum_{i=1}^n p_i}{p_j}.$$

Substitute $y = e^{-g_j}$, $dg_j = -\frac{dy}{y}$, $y \in (0, +\infty)$ in Eq. (15),

$$\begin{aligned}
&\int_{y=+\infty}^{0} y \exp(-yS)\left(-\frac{dy}{y}\right) \\
&= \int_0^{+\infty} \exp(-yS) dy && \text{(change of variables)} \\
&= \frac{1}{S} = \frac{p_j}{\sum_{i=1}^n p_i}. && \text{(exponential integral)}
\end{aligned}$$

Thus, combining all the above equations,

$$\Pr\left(j = \arg\max_i (g_i + \log p_i)\right) = \frac{p_j}{\sum_{i=1}^n p_i}.$$

□

C.3 Derivation of Eq. (11), Eq. (9) and Eq. (10)

The Gumbel-Softmax rollout of SofT-GRPO is given in Eq. (7). **Regarding the determinism of the mapping,** it is crucial to note that, for fixed retained support $\mathcal{K}_t$, truncated probabilities $\widetilde{\mathbf{p}}_t$ and Gumbel noises $\boldsymbol{\epsilon}_t$, the resulting soft token $\mathbf{s}_t$ is a *deterministic function* of $(\widetilde{\mathbf{p}}_t, \boldsymbol{\epsilon}_t)$ on $\mathcal{K}_t$:

$$\mathbf{s}_t = \mathrm{SoftEmbed}(\widetilde{\mathbf{p}}_t, \boldsymbol{\epsilon}_t; \mathcal{K}_t), \tag{16}$$

Hence, in our analysis, we always track the generative randomness via $\boldsymbol{\epsilon}_t$ (the base noise used to sample action $\boldsymbol{g}'_t$), and *not* via $\mathbf{s}_t$ directly. Cause every $\boldsymbol{g}'$ can map to one deterministic soft-token $\boldsymbol{s}_t$, but $\boldsymbol{s}_t$ can map to multiple $\boldsymbol{g}'$, calculating density on $\boldsymbol{g}'$ provides a tractable action likelihood. So, we consider outputting $\boldsymbol{g}'$ as an action and put the generation of $\boldsymbol{s}_t$ as a part of the environment transition.

Eq. (9): log-likelihood under the old policy. During rollout (generated by the old policy θ_{old}), we sample Gumbel noises $\boldsymbol{\epsilon}_t = (\epsilon_{t,i})_{i\in\mathcal{K}_t}$. The randomness of $\mathbf{s}_t$ is entirely attributable to $\boldsymbol{\epsilon}_t$, given $\widetilde{\mathbf{p}}_t$ and $\mathcal{K}_t$. The joint density factorizes as

$$p(\boldsymbol{\epsilon}_t) = \prod_{i\in\mathcal{K}_t} f_{\mathrm{Gumbel}}(\epsilon_{t,i}), \qquad f_{\mathrm{Gumbel}}(x) = \exp\big(-x - \exp(-x)\big),$$

where f_{Gumbel} represents the probability density function of a Gumbel distribution. Hence, the corresponding log-likelihood is

$$\log p(\boldsymbol{\epsilon}_t) = \sum_{i\in\mathcal{K}_t} \log f_{\mathrm{Gumbel}}(\epsilon_{t,i}) = \sum_{i\in\mathcal{K}_t} \Big[-\epsilon_{t,i} - \exp(-\epsilon_{t,i})\Big],$$

which gives Eq. (9). The translation from $\boldsymbol{\epsilon}_t$ to $\boldsymbol{g}'_t$ has unit Jacobian, giving the action density in Eq. (9).

Eq. (10): log-likelihood under the current policy. For off-policy correction, we hold both the stored action and $\mathcal{K}_t$ fixed. Equivalently, we store the perturbed logits.

$$g'_{t,i} \triangleq \log \widetilde{p}_{\theta_{\mathrm{old}},t,i} + \epsilon_{t,i},$$

so that, under a new policy θ with probabilities $\widetilde{p}_{\theta,t,i}$ normalized on $\mathcal{K}_t$, the implied noise that would have generated the same $g'_{t,i}$ under the *new* policy is

$$u_{\theta,t,i} = g'_{t,i} - \log \widetilde{p}_{\theta,t,i}.$$

Therefore, the log-density of the same stored action under the new policy is

$$\sum_{i\in\mathcal{K}_t} \log f_{\mathrm{Gumbel}}(g'_{t,i} - \log \widetilde{p}_{\theta,t,i}) = \sum_{i\in\mathcal{K}_t} \Big[-(g'_{t,i} - \log \widetilde{p}_{\theta,t,i}) - \exp\big(-(g'_{t,i} - \log \widetilde{p}_{\theta,t,i})\big)\Big],$$

which is exactly Eq. (10). This evaluates the likelihood of the stored action $\boldsymbol{g}'_t$ using its implied noise under the current policy.

Eq. (11): Off-Policy Ratio Using Eqs. (9) and (10) on the same stored support gives

$$\begin{aligned} \rho_t(\theta) &= \frac{q_\theta(\boldsymbol{g}'_{t,\mathcal{K}_t} \mid x_t, \mathcal{K}_t)}{q_{\theta_{\mathrm{old}}}(\boldsymbol{g}'_{t,\mathcal{K}_t} \mid x_t, \mathcal{K}_t)} \\ &= \exp\left(\sum_{i\in\mathcal{K}_t} \left[-u_{\theta,t,i} - e^{-u_{\theta,t,i}} + \epsilon_{t,i} + e^{-\epsilon_{t,i}} \right] \right). \end{aligned} \tag{17}$$

D Detailed Settings for Training & Testing

D.1 Hyperparameter Setting

All our experiments are performed on 8 $\times$ H200 GPUs. We show the experimental configurations in 6 for No-Finetune, 7 for GRPO, and 8 for SofT-GRPO.

Table 6: Parameters of No-Finetune

Parameter	Value
No-Finetune Testing under Discrete-Token Reasoning Paradigm	
Maximum response length	32768 tokens
Sampling temperature	0.6
(top-p, top-k)	(0.95, 30)
No-Finetune Testing under Soft-Thinking Reasoning Paradigm	
Maximum response length	32768 tokens
Sampling temperature	0.6
Gumbel-temperature τ_g	0.5
(top-p, top-k)	(0.95, 30)

Table 7: Parameters of Discrete-Token GRPO

Parameter	Value
Discrete-Token GRPO Training	
Maximum response length	8192 tokens
Sampling temperature	1.0
(top-p, top-k)	(1, -1)
Group Size G	8
Learning rate	1×10^{-6}
KL loss coefficient β	0.001
Policy clipping parameter ϵ	0.2
Discrete-Token GRPO Testing under Discrete-Token Reasoning Paradigm	
Maximum response length	32768 tokens
Sampling temperature	0.6
(top-p, top-k)	(0.95, 30)
GRPO Testing under Soft-Thinking Reasoning Paradigm	
Maximum response length	32768 tokens
Sampling temperature	0.6
Gumbel-temperature τ_g	0.5
(top-p, top-k)	(0.95, 30)

Table 8: Parameters of SofT-GRPO

Parameter	Value
SofT-GRPO Training	
Maximum response length	8192 tokens
Sampling temperature	1.0
(top-p, top-k)	(0.95, 5)
Group Size G	8
Gumbel-temperature τ_g	0.1
Learning rate	1×10^{-6}
KL loss coefficient β	0.001
Policy clipping parameter ϵ	0.2
SofT-GRPO Testing under Soft-Thinking Reasoning Paradigm	
Maximum response length	32768 tokens
Sampling temperature	1.0
Gumbel-temperature τ_g	0.1
(top-p, top-k)	(0.95, 5)

D.2 MDP DEFINITION

We formulate SofT-GRPO as a finite-horizon Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{P}, r)$ conditioned on a query Q.

State $\mathcal{S}$: At the t-th soft-thinking step, the state is the current prefix

$$x_t = [Q, (s_1, \ldots, s_{t-1})],$$

where $s_1, \ldots, s_{t-1}$ are previously generated soft tokens.

Action $\mathcal{A}$: Given x_t, we define the action as the Gumbel-perturbed log-probability vector on the stored rollout support $\mathcal{K}_t$

$$g'_{t,\mathcal{K}_t} = (g'_{t,i})_{i\in\mathcal{K}_t} \in \mathbb{R}^{|\mathcal{K}_t|}, \qquad g'_{t,i} = \log \widetilde{p}_{\theta,t,i} + \epsilon_{t,i}, \qquad \epsilon_{t,i} \sim \mathrm{Gumbel}(0,1), \quad i \in \mathcal{K}_t.$$

Transition $\mathcal{T}$: The transition maps this action to the next soft token through the Gumbel-Softmax relaxation:

$$y'_{t,i} = \frac{\exp(g'_{t,i}/\tau_g)}{\sum_{j\in\mathcal{K}_t}\exp(g'_{t,j}/\tau_g)}, \qquad s_t = \sum_{i\in\mathcal{K}_t} y'_{t,i} e_i.$$

Here $y'_{t,i} = 0$ outside $\mathcal{K}_t$. Thus, the next state is $x_{t+1} = [Q, (s_1, \ldots, s_t)]$. After generating the soft-thinking trajectory $S = (s_1, \ldots, s_{|S|})$,

Reward r: The model generates the answer A, and the reward is given by a verifiable reward function $r(Q, S, A) = f(A)$, where $f(A) = 1$ if A is correct and 0 otherwise.

Policy $\mathcal{P}$: Under this formulation, the likelihood of an action $g'_{t,\mathcal{K}_t}$ under policy π_θ is evaluated by the density of its implied Gumbel noise:

$$q_\theta(g'_{t,\mathcal{K}_t} \mid x_t, \mathcal{K}_t) = \exp\left(\sum_{i\in\mathcal{K}_t} -\left(g'_{t,i} - \log\widetilde{p}_{\theta,t,i}\right) - \exp\left[-\left(g'_{t,i} - \log\widetilde{p}_{\theta,t,i}\right)\right]\right),$$

where $\widetilde{p}_{\theta,t,i}$ is defined by the truncation in Section 3.1. The equation is derived in Appendix C.3. During rollout, $g'_{t,\mathcal{K}_t}$ is sampled using $\pi_{\theta_{\text{old}}}$ and stored with $\mathcal{K}_t$ in the replay buffer; during optimization, the same stored action $g'_{t,\mathcal{K}_t}$ is re-evaluated under the current policy π_θ, with probabilities renormalized on the same $\mathcal{K}_t$, enabling conditional action-density ratios.

D.3 Time Complexity & Consumption Analysis

Time Complexity and Consumption of Soft-Thinking Soft-Thinking reasoning has the same time complexity as the conventional discrete-token decoding when enabling top-k aggregation instead of the whole vocabulary (Zhang et al., 2025b; Wu et al., 2025). However, soft tokens require re-encoding k candidate objects, thus necessitating the transmission of k tokens instead of one during step-by-step decoding. This introduces constant-time complexity for the model computation. Furthermore, considering that existing decoding processes are often specifically optimized for discrete tokens, soft-thinking is actually slightly slower.

Time Complexity and Consumption of SofT-GRPO. For a trajectory of length T, both discrete-token decoding and soft-thinking require one LLM forward pass at each step, so the dominant inference cost remains the LLM decoding cost. Soft-Thinking only adds to the construction of a soft token. In our implementation, this is performed after top-k filtering rather than over the full vocabulary $|\mathcal{T}|$, introducing an extra cost of $\mathcal{O}(Tkd)$, where k is the number of retained tokens and d is the embedding dimension. Since k is fixed to a small constant, this does not change the LLM-dominated asymptotic complexity.

During rollout, SofT-GRPO additionally stores the top-k Gumbel-perturbed action $\boldsymbol{g}'$ and the log-probability term for off-policy probability-ratio computation, requiring $k + 1$ real numbers per soft-thinking step. This is still constant per step, while discrete-token rollout stores one sampled token and its corresponding probability information. Therefore, SofT-GRPO does not introduce a vocabulary-scale $\mathcal{O}(|\mathcal{T}|)$ overhead or increase the asymptotic time complexity of GRPO.

Empirically, compared with discrete-token GRPO, SofT-GRPO requires about $2\times$ to $3\times$ wall-clock time, mainly due to current system-level inefficiencies in soft-thinking rollout. We regard this as an implementation limitation and expect future LLM inference-system optimizations to reduce this overhead.

E SUPPLEMENTARY OF EXPERIMENTS

E.1 COMPARISON WITH SIMKO

Common evaluation protocol. Unless stated otherwise, the added experiments use the same DeepScaler training data, verifier, hardware, optimizer, group size, and sampling configurations as Appendix D. @1 denotes Mean@32, and @16/@32 denote Pass@16/Pass@32. All reported entries are percentages. Tables in the following four subsections use the same format as the main results: the best result for each dataset and metric is underlined, the best average is bolded, and the second-best average is shaded.

We implement the discrete-token SimKO baseline (Peng et al., 2025) on DeepSeek-R1-Distill-Qwen-1.5B under the same training and evaluation protocol as GRPO and SofT-GRPO. SimKO improves the discrete-token GRPO baseline at larger K, but SofT-GRPO remains higher by 1.49 points on Mean@32, 2.16 points on Pass@16, and 2.10 points on Pass@32 when averaged over the five benchmarks. Table 9 gives the complete per-dataset results.

Table 9: Detailed comparison with SimKO on DeepSeek-R1-Distill-Qwen-1.5B.

	AIME24			AIME25			AMC23			MATH500			GSM8K			Avg.		
Method	@1	@16	@32	@1	@16	@32	@1	@16	@32	@1	@16	@32	@1	@16	@32	@1	@16	@32
DeepSeek-R1-Distill-Qwen-1.5B																		
Discrete-Token CoT Reasoning Pattern																		
Base	30.6	70.0	73.3	23.0	46.7	53.3	70.7	92.5	95.0	84.6	97.8	97.8	81.5	95.8	96.7	58.09	80.54	83.23
+ GRPO	31.8	66.7	76.7	25.3	46.7	46.7	77.3	95.0	95.0	87.1	97.4	97.8	84.9	95.1	95.8	61.28	80.16	82.39
+ SimKO	30.5	73.3	73.3	24.8	46.7	53.3	73.1	95.0	95.0	86.3	96.6	97.2	84.8	95.3	96.6	59.90	81.38	83.08
Soft Thinking Reasoning Pattern																		
Base	27.3	66.7	70.0	23.8	46.7	53.3	69.9	95.0	95.0	79.4	93.2	96.6	81.0	94.6	97.1	56.28	79.23	82.41
+ GRPO	29.2	70.0	73.3	25.4	46.7	53.3	75.8	95.0	95.0	86.3	96.8	98.2	84.9	95.6	96.4	60.31	80.81	83.26
+ SofT-GRPO	32.6	76.7	80.0	26.1	50.0	53.3	76.4	97.5	97.5	86.3	97.4	98.0	85.5	96.1	97.0	**61.39**	**83.54**	**85.18**

E.2 GENERALIZATION TO THREE ADDITIONAL BASE MODELS

To reduce dependence on R1-distilled models, we additionally evaluate Qwen2.5-1.5B, Qwen2.5-1.5B-Math, and Qwen3-1.7B. The maximum response lengths are 4,096 tokens for the two Qwen2.5 models and 8,192 tokens for Qwen3-1.7B; all other training and evaluation settings follow the common protocol above. These models cover a general base model, a math-specialized base model, and a newer general model. Detailed results are reported in Table 10. SofT-GRPO obtains the strongest averaged @1 and @32 result on all three models.

Table 10: Detailed results on three additional non-R1-distilled base models.

	AIME24			AIME25			AMC23			MATH500			GSM8K			Avg.		
Method	@1	@16	@32	@1	@16	@32	@1	@16	@32	@1	@16	@32	@1	@16	@32	@1	@16	@32
Qwen2.5-1.5B																		
Discrete-Token CoT Reasoning Pattern																		
Base	0.42	3.33	3.33	0.00	0.00	0.00	2.50	20.0	35.0	5.08	42.2	55.0	11.93	69.07	81.35	3.99	26.92	34.94
+ GRPO	2.10	3.33	3.33	0.83	6.67	13.33	21.81	35.0	45.0	43.62	78.6	87.6	68.29	79.38	84.91	27.33	40.60	46.84
Soft Thinking Reasoning Pattern																		
Base	0.42	6.67	10.0	0.10	0.00	3.33	2.11	15.0	22.5	3.81	29.6	43.0	11.93	69.07	81.35	3.67	24.07	32.04
+ SofT-GRPO	2.08	16.67	30.0	0.42	6.67	10.0	23.36	72.5	75.0	48.99	82.6	86.6	69.03	94.85	96.97	**28.78**	**54.66**	**59.71**
Qwen2.5-1.5B-Math																		
Discrete-Token CoT Reasoning Pattern																		
Base	8.54	36.67	40.0	4.90	23.33	36.67	42.27	75.0	80.0	59.10	89.4	92.4	66.97	96.13	97.57	36.36	64.11	69.33
+ GRPO	10.73	30.0	40.0	7.08	33.33	36.67	47.58	82.5	92.5	70.36	90.4	93.2	83.42	96.97	97.88	43.83	66.64	72.05
Soft Thinking Reasoning Pattern																		
Base	9.79	33.33	43.33	4.48	36.67	36.67	41.80	87.5	95.0	58.94	89.0	91.4	66.69	95.30	97.80	36.34	68.36	72.84
+ SofT-GRPO	10.83	50.0	53.33	8.94	26.67	36.67	47.58	80.0	90.0	69.01	91.4	94.0	83.43	96.82	98.18	**43.96**	**68.98**	**74.44**
Qwen3-1.7B																		
Discrete-Token CoT Reasoning Pattern																		
Base	28.4	63.3	66.7	23.9	46.7	50.0	68.5	90.0	90.0	85.6	95.8	97.0	90.1	96.2	96.6	59.30	78.40	80.06
+ GRPO	25.7	53.3	56.7	20.9	40.0	40.0	70.1	92.5	92.5	86.3	96.0	97.8	88.8	95.8	96.2	58.36	75.52	76.64
Soft Thinking Reasoning Pattern																		
Base	28.4	66.7	66.7	24.0	43.3	43.3	66.7	87.5	87.5	85.2	96.2	97.4	90.0	96.4	97.0	58.86	78.02	78.38
+ GRPO	27.3	50.0	53.3	21.5	36.7	43.3	70.9	90.0	92.5	86.6	96.6	97.2	88.9	95.2	96.3	59.04	73.70	76.52
+ SofT-GRPO	29.48	60.0	70.0	26.88	50.0	53.33	69.06	95.0	95.0	87.35	98.2	98.6	89.31	96.06	96.51	**60.42**	**79.85**	**82.69**

E.3 Combining the Soft-Trajectory Likelihood with Dr.GRPO

The proposed Gumbel-based soft-trajectory likelihood is not specific to the GRPO advantage estimator. To test this modularity, we replace the GRPO objective components with Dr.GRPO (Liu et al., 2025b) while retaining the same Gumbel rollout, action definition, and likelihood reconstruction. We use DeepSeek-R1-Distill-Qwen-1.5B, DeepScaler, an 8,192-token maximum response length, and 420 training steps for both Dr.GRPO and Dr.SofT-GRPO. Table 11 shows that Dr.SofT-GRPO improves the five-dataset average over discrete-token Dr.GRPO on @1, @16, and @32.

Table 11: Detailed results when combining the proposed soft-trajectory likelihood with Dr.GRPO.

	AIME24			AIME25			AMC23			MATH500			GSM8K			Avg.		
Method	@1	@16	@32	@1	@16	@32	@1	@16	@32	@1	@16	@32	@1	@16	@32	@1	@16	@32
DeepSeek-R1-Distill-Qwen-1.5B																		
Discrete-Token CoT Reasoning Pattern																		
Base	22.60	63.33	70.00	21.67	36.67	53.33	63.05	95.0	95.0	80.57	96.4	97.6	81.48	95.6	96.74	53.87	77.40	82.53
+ Dr.GRPO	22.08	60.00	70.00	20.73	43.33	53.33	69.38	92.5	97.5	82.96	96.4	97.0	82.56	95.75	96.51	55.54	77.60	82.87
Soft Thinking Reasoning Pattern																		
+ Dr.SofT-GRPO	25.62	56.67	73.33	23.23	43.33	56.67	67.27	97.5	97.5	82.33	96.0	97.0	80.16	95.45	96.70	**55.72**	**77.79**	**84.24**

E.4 Code-Reasoning Experiments

We train DeepSeek-R1-Distill-Qwen-1.5B on approximately 15K executable programming problems filtered from MathInstruct-Code-1M, with an 8,192-token maximum response length. Evaluation covers HumanEval, MBPP, and KlearReasoner. We report two reward settings: (i) a dataset-native partial-credit reward equal to the fraction of passed tests, which preserves learning signals from partially correct programs; and (ii) a strict exact-correctness reward that equals one only when all tests pass. The rollout, Gumbel action, and policy-update construction are otherwise unchanged. Tables 12 and 13 give all per-dataset results. Under partial-credit rewards, GRPO degrades relative to the base model in the reported run while SofT-GRPO improves. Under strict rewards, the gap is smaller, particularly for Pass@1.

Table 12: Code-reasoning results with dataset-native partial-credit rewards.

	HumanEval			MBPP			KlearReasoner			Avg.		
Method	@1	@16	@32	@1	@16	@32	@1	@16	@32	@1	@16	@32
DeepSeek-R1-Distill-Qwen-1.5B												
Discrete-Token CoT Reasoning Pattern												
Base	50.50	81.86	84.15	54.63	81.61	84.05	9.50	21.89	23.83	38.21	61.78	64.01
+ GRPO	33.97	65.31	68.90	44.97	73.26	76.65	6.20	12.96	14.45	28.38	50.51	53.34
Soft Thinking Reasoning Pattern												
Base	51.07	81.87	84.15	55.24	81.40	83.66	9.75	22.59	25.00	38.69	61.95	64.27
+ SofT-GRPO	62.69	87.54	89.63	60.60	84.61	87.16	15.87	39.90	45.31	**46.39**	**70.68**	**74.04**

Table 13: Code-reasoning results with strict exact-correctness rewards.

	HumanEval			MBPP			KlearReasoner			Avg.		
Method	@1	@16	@32	@1	@16	@32	@1	@16	@32	@1	@16	@32
DeepSeek-R1-Distill-Qwen-1.5B												
Discrete-Token CoT Reasoning Pattern												
Base	50.50	81.86	84.15	54.63	81.61	84.05	9.50	21.89	23.83	38.21	61.78	64.01
+ GRPO	68.75	88.43	90.63	66.05	90.90	92.71	15.10	26.64	38.63	49.97	68.66	73.99
Soft Thinking Reasoning Pattern												
Base	51.07	81.87	84.15	55.24	81.40	83.66	9.75	22.59	25.00	38.69	61.95	64.27
+ SofT-GRPO	69.33	90.48	93.75	65.84	92.49	93.75	15.24	33.40	37.50	**50.14**	**72.12**	**75.00**

Table 14: Experiments on the token efficiency for baselines and the proposed SofT-GRPO. #Token values in the table represent the number of tokens across all queries, and the #Token_c values represent the number of tokens across correct queries.

Dataset	AIME2024		AIME2025		AMC23		MATH-500		GSM8K		Average	
Metrics	#Token	#Token_c	#Token	#Token_c	#Token	#Token_c	#Token	#Token_c	#Token	#Token_c	#Token	#Token_c
DeepSeek-R1-Distill-Qwen-1.5B												
Discrete-Token CoT Reasoning Pattern												
No-Finetune	16241.6	14997.8	16416.3	13448.8	10052.2	9394.2	5616.6	5368.1	1839.3	1772.5	10033.2	8996.3
+ GRPO	8927.6	8535.6	8039.4	6414.2	5001.3	4672.8	3106.1	2958.5	1417.1	1370.6	5298.3	4790.3
Soft Thinking Reasoning Pattern												
No-Finetune	17857.8	16191.3	17569.4	14582.4	11269.9	10482.0	4015.9	3888.3	1699.3	1649.9	10482.5	9358.8
+ GRPO	9383.2	8934.5	8007.2	7325.8	5203.7	4894.4	3233.6	3131.7	1385.5	1349.9	5442.7	5127.2
+ SofT-GRPO	11039.6	10756.1	10519.6	7831.3	5900.2	5630.4	3549.5	3399.2	1577.6	1542.5	6517.3	5831.9
LLaMA-3.2-3B-Instruct Base LLM												
Discrete-Token CoT Reasoning Pattern												
No-Finetune	5010.5	3334.1	4297.6	2430.9	2790.7	2398.1	1850.1	1494.2	236.6	224.6	2837.1	1976.4
+ GRPO	8991.8	6368.9	8443.8	11088.3	4998.1	3978.5	3334.8	2463.0	502.0	453.6	5254.1	4870.5
Soft Thinking Reasoning Pattern												
No-Finetune	4859.2	2727.7	4934.8	3735.2	3259.9	2624.0	2086.1	1494.1	243.5	227.8	3076.7	2161.8
+ GRPO	9749.0	7627.8	8498.9	10041.4	5389.9	4987.3	3701.9	2764.6	541.9	505.8	5576.3	5185.4
+ SofT-GRPO	829.7	862.9	893.1	879.6	911.5	927.9	632.8	575.7	294.6	292.9	712.3	707.8
DeepSeek-R1-Distill-Qwen-7B												
Discrete-Token CoT Reasoning Pattern												
No-Finetune	13120.7	11511.4	14347.7	11750.7	6346.0	5987.1	3998.9	3939.0	1061.1	1032.0	7774.9	6844.0
+ GRPO	7795.9	7116.1	8003.5	7369.8	4050.9	3719.7	2473.3	2405.5	1146.7	1118.7	4694.0	4346.0
Soft Thinking Reasoning Pattern												
No-Finetune	13017.4	11888.6	14116.8	11507.4	6346.7	6043.6	3947.4	3858.6	996.3	962.2	7684.9	6852.1
+ GRPO	7464.9	6531.4	8291.6	7160.4	3931.7	3736.7	2473.2	2423.0	1117.0	1104.5	4655.7	4191.2
+ SofT-GRPO	8035.8	7556.8	8381.7	7873.7	4008.0	3843.5	2630.2	2583.5	1293.2	1276.4	4869.8	4626.8

E.5 Supplementary of Token Efficiency

As shown in Section 4.6, besides the performance, the token efficiency of LLMs is also an important metric. In this section, we compare the token efficiency of baselines and SofT-GRPO in Table 14. Compared to No-Finetune variants, SofT-GRPO can demonstrate a clear refinement in both the token efficiency across all queries and the token efficiency across correct queries. Relative to discrete-token GRPO, average generation length increases by about 23% on the 1.5B model and 3.7% on the 7B model, and decreases by 86.4% on LLaMA. Specifically, we observe a severe reduction in the thinking length of the *LLaMA-3.2-3B-Instruct* model. As shown in Figure 4, unlike GRPO, SofT-GRPO maintains and even enhances the token efficiency compared to the base LLM when training progresses, showing shorter generated sequences on this model.

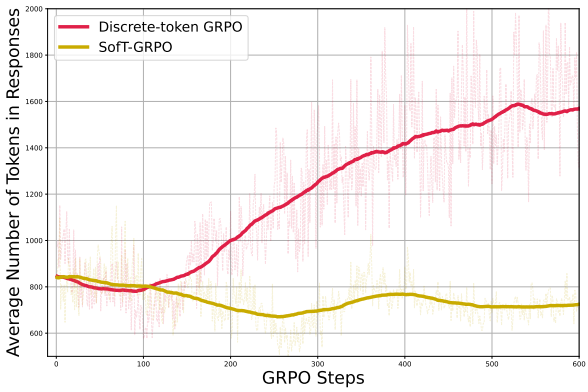


Figure 4: Token consumption curve on ***LLaMA-3.2-3B-Instruct Base LLM*** during training.

E.6 Experiments under Different Temperatures

In Section 4, we adopt the setting of $\tau = 0.6$ for discrete-token CoT. To investigate whether other temperature settings will break our observations in our main experiments, similar to (Yue et al., 2025), we try temperatures from the collection of $\tau = 0.6$, $\tau = 0.8$, $\tau = 1.0$, $\tau = 1.2$, and $\tau = 1.4$.

As shown in Figure 5, we conduct experiments on the 1.5B LLMs (i.e., ***DeepSeek-R1-Distill-Qwen-1.5B Base LLM***) for their average Pass@K accuracies across five numerical reasoning benchmarks (i.e., AIME2024, AIME2025, AMC23, MATH-500, GSM8K), where the proposed SofT-GRPO can demonstrate outstanding results compared to GRPO and No-Finetune variants with various temperatures, from Pass@1 to Pass@32.

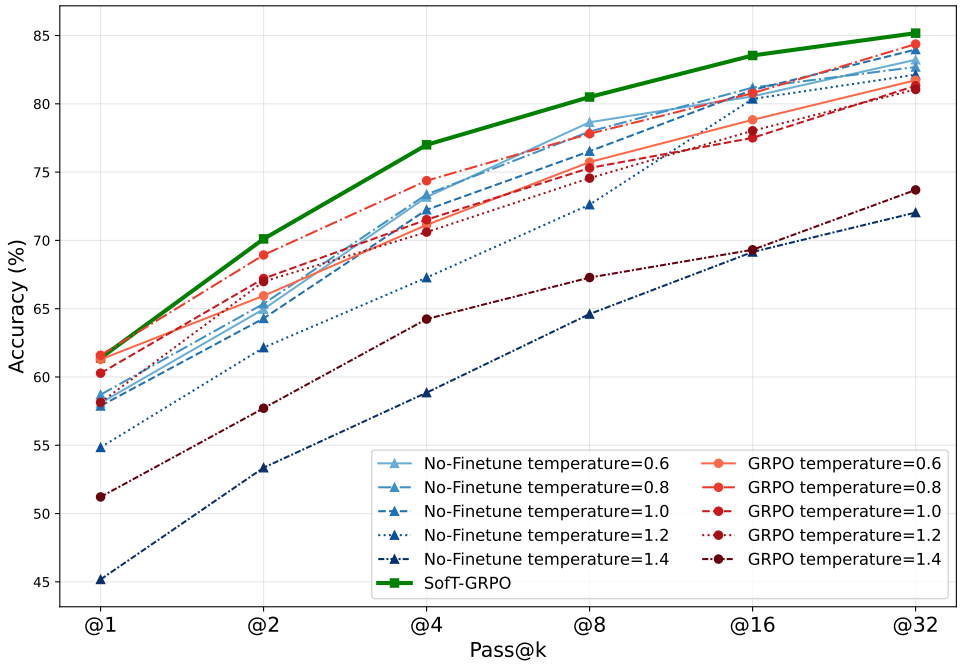
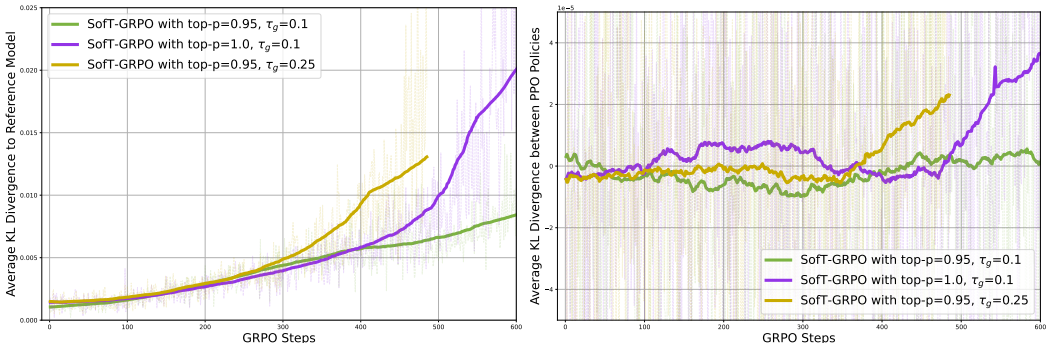


Figure 5: Running discrete-token CoT methods (GRPO and No-finetune) with more temperature options on ***DeepSeek-R1-Distill-Qwen-1.5B Base LLM***. Pass@K represents the pass rate within at most k runs, and Pass@1 is additionally averaged from 32 runs. Experiments are run on the five datasets in Table 1 for the average.

E.7 SUPPLEMENTARY ABLATION ON HYPER-PARAMETERS



(a) Training KL Divergence curve between $\pi_{\theta_{\text{ref}}}$ and π_θ

(b) Training Proximal Policy Optimization (PPO) (Schulman et al., 2017) KL Divergence curve between $\pi_{\theta_{\text{old}}}$ and π_θ

Figure 6: KL Divergence curves in SofT-GRPO over ***DeepSeek-R1-Distill-Qwen-1.5B*** Base LLM.

In this subsection, we further investigate the hyperparameter settings. As briefly shown in Section 5, adopting a higher top-p or τ_g will cause collapses in training. These runs show sensitivity to the sampling configuration.

As shown in Figure 6(a), the variants (varying top-p to 1.0 or varying τ_g to 0.25) will demonstrate higher divergence between the fixed pre-trained $\pi_{\theta_{\text{ref}}}$ and the current policy, describing larger changes in the policy during these runs.

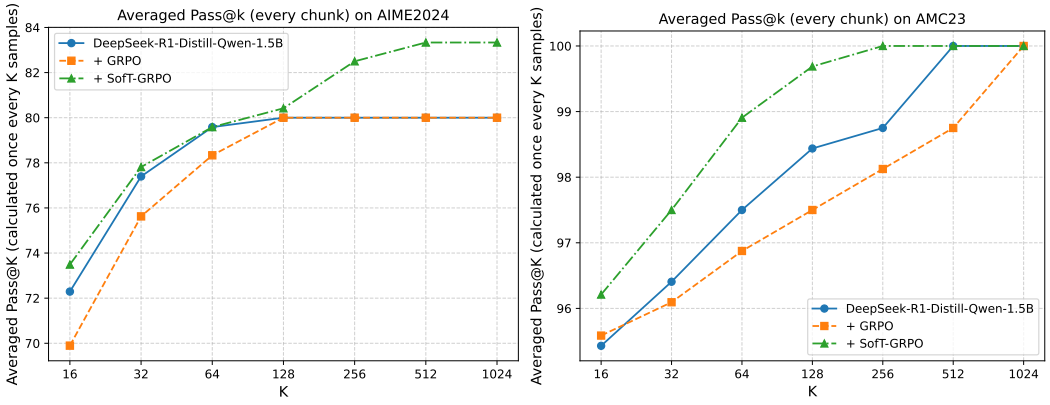
Recently, (Liu et al., 2025a; Qi et al., 2025b) provides excellent insight into the collapse situation in the RLVR fine-tuning. When collapse is caused by a precision issue, they observe a very high KL divergence between $\pi_{\theta_{\text{old}}}$ and π_θ (Refer to Figure 3 in (Qi et al., 2025b), 10^{-3} even higher). However, in the variants shown in Figure 6(b), we find that their KL divergence in PPO policies is less than 10^{-5}, providing an additional diagnostic of policy changes during training.

E.8 PASS@K UP TO K=1024

Table 1 and Figure 5 show the Pass@K for K at most 64. In (Yue et al., 2025), they conduct experiments for K up to 1024. We adopt this setting. Confined by limited computational resources, we implement the proposed SofT-GRPO over ***DeepSeek-R1-Distill-Qwen-1.5B*** for 1024 runs, as well as the GRPO and No-Finetune baselines with the discrete-token CoT reasoning pattern.

As shown in Figure 7, we calculate Pass@16 to Pass@1024 on AIME2024 and AMC 23. As a difference from the result in Table 1, **we obtain Pass@K for each K runs instead of the first K runs**. For example, when calculating Pass@32, we divide the 1024 runs into 32 groups, calculating their Pass@K value and averaging. In Table 1 instead, we calculate the Pass@K at first 32 runs.

Figure 7 obtains the same result pattern compared to the results in (Yue et al., 2025), where GRPO will outperform the No-finetune version when K is small, but will be left behind when K increases. SofT-GRPO obtains higher or equal estimated coverage in the displayed comparisons from $K = 16$ to $K = 1024$.



(a) Pass@16 to Pass@1024 on AIME2024 (b) Pass@16 to Pass@1024 on AMC23

Figure 7: Pass@16 to Pass@1024 on AIME2024 and AMC 23. The fluctuation in Figure (a) occurs because AIME only has 30 instances, and SofT-GRPO correctly solved three out of four (1024/256) problems when K=256. In Figure (b), all algorithms ultimately achieved complete correctness.

Raw Data on Pass@16 to Pass@64

The raw data for Pass@16 to Pass@64 are as in Table 15, where we report the mean and std.

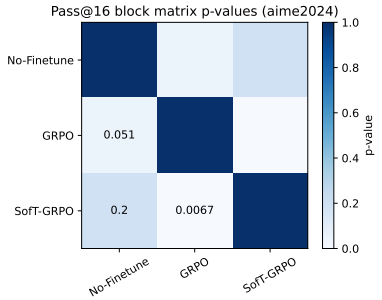
Table 15: Results on AIME2024 and AMC23. Both mean and standard deviation are reported on the proportion scale.

Method	AIME2024 Pass@16 (64 runs)		AIME2024 Pass@32 (32 runs)		AIME2024 Pass@64 (16 runs)	
	Mean	Std	Mean	Std	Mean	Std
DeepSeek-R1-Distill-Qwen-1.5B	0.7229	0.04	0.7740	0.03	**0.7958**	0.01
+ GRPO	0.6990	0.04	0.7563	0.03	0.7833	0.02
+ SofT-GRPO	**0.7349**	0.03	**0.7781**	0.02	**0.7958**	0.02
Method	AMC23 Pass@16 (64 runs)		AMC23 Pass@32 (32 runs)		AMC23 Pass@64 (16 runs)	
	Mean	Std	Mean	Std	Mean	Std
DeepSeek-R1-Distill-Qwen-1.5B	0.9543	0.02	0.9641	0.02	0.9750	0.02
+ GRPO	0.9559	0.01	0.9609	0.01	0.9688	0.01
+ SofT-GRPO	**0.9621**	0.02	**0.9750**	0.02	**0.9891**	0.02

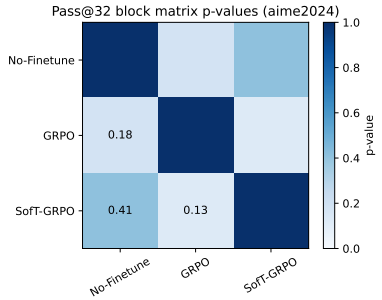
E.9 P-VALUES FOR SIGNIFICANCE ON PASS@K

Besides the results in Table 1 and Figure 5, in this section, we conduct significance tests for Pass@16 and Pass@32 by dividing 1024 runs into 64 and 32 groups, respectively. These tests measure variation across generation blocks for the evaluated checkpoints. We calculated the t-test for the one-sided hypothesis test (See Figure 7, the average result of SofT-GRPO outperforms the GRPO and No-Finetune baselines), and taking 0.05 as the threshold, the results in Figure 8 and 9 show that in Pass@16 with 64 runs, and Pass@32 with 32 runs:

- **SofT-GRPO can significantly outperform GRPO on AIME2024 with Pass@16 (0.0067), AMC23 with Pass@32 (0.022)**, while the reported values of 0.13 for AIME2024 Pass@32 and AMC23 Pass@16 exceed the 0.05 threshold.
- Against the no-finetune baseline, the reported AMC23 p-values are 0.081 for Pass@16 and 0.054 for Pass@32, both above 0.05.

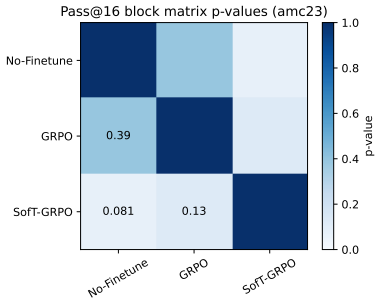


(a) P-scores of Pass@16 over SofT-GRPO, GRPO, and No-finetune on AIME2024

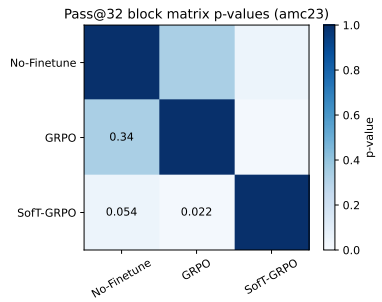


(b) P-scores of Pass@32 over SofT-GRPO, GRPO, and No-finetune on AIME2024

Figure 8: Significance test on AIME2024.



(a) P-scores of Pass@16 over SofT-GRPO, GRPO, and No-finetune on AMC23



(b) P-scores of Pass@32 over SofT-GRPO, GRPO, and No-finetune on AMC23

Figure 9: Significance test on AMC23.

F Discussion & Analysis on Results

In this section, we will provide a detailed analysis of the improvement in Pass@K. (Yue et al., 2025) reports settings in which discrete-token GRPO yields **equal or even inferior Pass@K values** when k improves. So, Recent studies (Walder & Karkhanis, 2025; Mahdavi et al., 2025; Peng et al., 2025) try to develop RL methods that optimize the Pass@K accuracy directly. Among them, (Walder & Karkhanis, 2025; Chow et al., 2024; Mahdavi et al., 2025) focus on reallocating the reward within the group, encouraging rewards of negative discrete-token CoTs in groups with high Pass@K value. (Peng et al., 2025) try to improve the model's Pass@K performance by avoiding the over-concentration on the top-1 selection within the token set.

In SimKO, at high-entropy (forking) tokens, positive (correct) samples are rewarded not only on the sampled token but spread across the top-K most probable tokens, promoting output diversity and better Pass@K. For negative (incorrect) samples, SimKO applies stronger penalties to the top-1 token and lighter penalties to others, which prevents the output probabilities from collapsing onto a single choice. This entropy-aware and asymmetric strategy encourages exploration, improving Pass@K in the reported experiments.

In the proposed SofT-GRPO, the action log-density score at the t-th soft-thinking step, holding $\mathcal{K}_t$ and the sampled action fixed, is (suppressing t below):

$$
\begin{aligned}
\nabla_\theta \log q_\theta(\boldsymbol{g}'_{\mathcal{K}_t} \mid x, \mathcal{K}_t) &= \nabla_\theta \left(\sum_{i \in \mathcal{K}_t} [-g'_i + \log \widetilde{p}_{\theta,i} - \exp(-g'_i + \log \widetilde{p}_{\theta,i})] \right) \\
&= \sum_{i \in \mathcal{K}_t} \nabla_\theta \left(\log \widetilde{p}_{\theta,i} - \exp(-g'_i + \log \widetilde{p}_{\theta,i}) \right) \\
&= \sum_{i \in \mathcal{K}_t} \left(\nabla_\theta \log \widetilde{p}_{\theta,i} - \exp(-g'_i + \log \widetilde{p}_{\theta,i}) \nabla_\theta \log \widetilde{p}_{\theta,i} \right) \\
&= \sum_{i \in \mathcal{K}_t} \left(1 - \exp(-g'_i + \log \widetilde{p}_{\theta,i}) \right) \nabla_\theta \log \widetilde{p}_{\theta,i} \\
&= \sum_{i \in \mathcal{K}_t} \left(1 - \widetilde{p}_{\theta,i} \cdot \exp(-g'_i) \right) \nabla_\theta \log \widetilde{p}_{\theta,i}.
\end{aligned}
$$

This expression is the action log-density score; the full update also depends on advantage, the likelihood ratio, clipping, and regularization. At the on-policy point, $g'_i = \log \widetilde{p}_{\theta,i} + \epsilon_i$, so the coefficient becomes $1 - \exp(-\epsilon_i)$. The score therefore does not establish a preference for low-probability tokens. Categorical actions also couple all logits through $\partial \log p_a / \partial z_i = \mathbf{1}[i = a] - p_i$.

Figure 10 shows the observed entropy dynamics in this training configuration: entropy decreases toward zero with discrete-token GRPO and remains higher with SofT-GRPO.

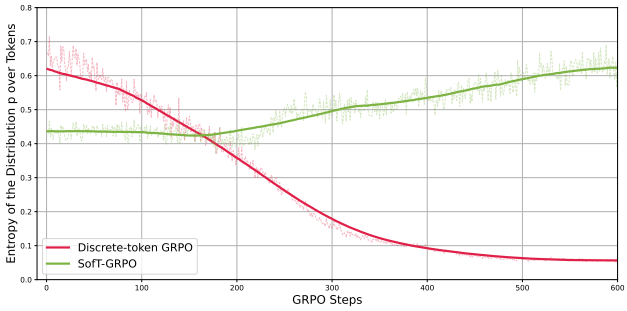


Figure 10: Entropy curve on ***DeepSeek-R1-Distill-Qwen-7B*** during training.

G BASELINES & DATASETS & LICENSES

In our experiments, we evaluate and compare the following model baselines and datasets. For each, we detail the official website and usage license.

G.1 BASELINES

We mainly include No-Finetune base LLMs, Discrete-Token GRPO, and the method in (Butt et al., 2025) (noted Soft Token) as baselines.

Base LLMs This paper includes ***DeepSeek-R1-Distill-Qwen-1.5B***, ***LLaMA-3.2-3B-Instruct***, and ***DeepSeek-R1-Distill-Qwen-7B*** as base LLMs.

Discrete-Token GRPO We utilize the latest verl (Sheng et al., 2024) https://github.com/volcengine/verl/tree/main and vLLM rollout to implement discrete-token GRPO with default parameters.

(Butt et al., 2025) Soft Token Due to the requirement of passing d-dimensional inputs $\hat{s}$ between the rollout workers and the verl policy optimization workers, implementing this algorithm requires a high amount of communication between the rollout workers and the policy update workers. So, we only report the results in (Butt et al., 2025) for comparison in Table 4.

G.2 DATASETS

This paper covers five in-domain numerical reasoning datasets (i.e., AIME2024, AIME2025, AMC23, MATH-500, and GSM8K (Cobbe et al., 2021)), one out-of-domain scientific reasoning dataset GPQA-Diamond (Rein et al., 2024), and two out-of-domain code reasoning datasets (i.e., HumanEval and MBPP). These datasets are provided in https://github.com/eric-ai-lab/Soft-Thinking.

G.3 INFERENCE FRAMEWORK

The inference framework of our implementation is built on **SGLang** (Zheng et al., 2024), maximizing efficiency via continuous batching, RadixAttention for KV cache reuse, and compressed finite state machines for faster structured output decoding. In the standard flow, the scheduler flattens discrete token inputs into continuous tensors for GPU execution; generated tokens are returned to update the Radix tree state for subsequent steps.

SGLang's optional overlap scheduler pipelines execution by issuing the next batch ahead via negative integer slot addresses. Compared to (Zhang et al., 2025b), **we enabled this** for Soft-Thinking by storing generated soft probabilities and indices in **VRAM-allocated future maps**. A custom kernel resolves the slot addresses to fetch these states just-in-time. This design **should be faster** by eliminating PCIe bottlenecks, allowing non-blocking scheduling with data kept entirely on the GPU.

G.4 LICENSES

For Base LLM, Dataset, and frameworks, we list their Licenses in Table 16.

Table 16: A summary of licenses.

Resources	Type	License	URL
DeepSeek-R1-Distill-Qwen-1.5B	Base LLM	MIT License	https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B
LLaMA-3.2-3B-Instruct	Base LLM	Llama 3.2 License	https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct
DeepSeek-R1-Distill-Qwen-7B	Base LLM	MIT License	https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-7B
verl	RL-framework	Apache-2.0 license	https://github.com/volcengine/verl
verl-0.4.x	RL-framework	Apache-2.0 license	https://github.com/volcengine/verl/tree/v0.4.x
SGLang for soft-thinking	Inference-framework	MIT License	https://github.com/eric-ai-lab/Soft-Thinking
AIME2024, AIME2025, GSM8K	Dataset	MIT License	https://github.com/eric-ai-lab/Soft-Thinking
AMC23, MATH-500	Dataset	Available Online	https://github.com/eric-ai-lab/Soft-Thinking
HumanEval, GPQA-Diamond	Dataset	MIT License	https://github.com/eric-ai-lab/Soft-Thinking
MBPP	Dataset	Apache-2.0 license	https://github.com/eric-ai-lab/Soft-Thinking

H VISUALIZATION

In this section, in Figure 11 and Figure 12, we provide visualization examples for the outcome of SofT-GRPO on two examples (we show y_i numbers that are larger than 0.01 in soft-tokens). Our observations are as follows:

- Compared to Figure 4 in (Zhang et al., 2025b), SofT-GRPO produces intermediate representations with appreciable weights on multiple tokens.
- At the 288th soft token in Figure 11, the top two components are “No” and “Yeah”. This illustrates a mixture containing tokens with contrasting meanings.
- Similarly to Figure 4 in (Zhang et al., 2025b), SofT-GRPO admits readable dominant-token projections in these examples; the contents in the red box (Okay, so I need to...) illustrates the dominant-token projection of the soft-thinking trajectory.

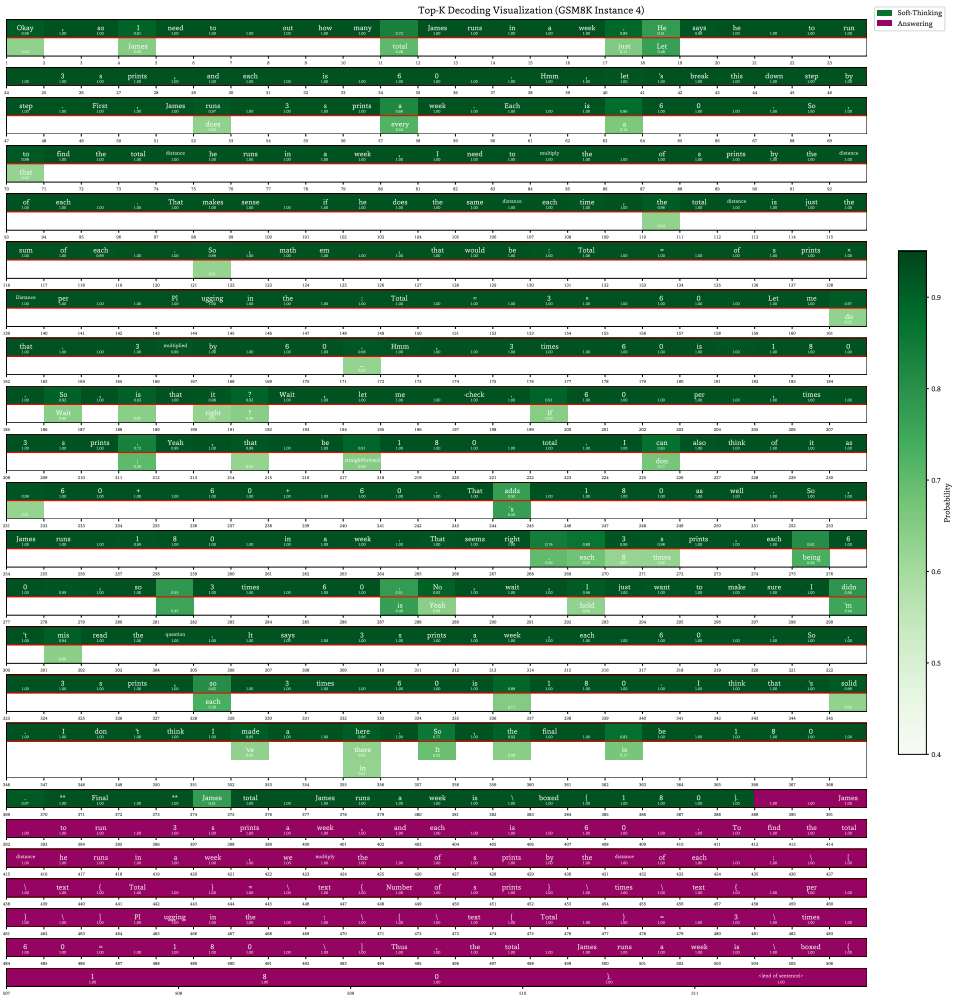


Figure 11: Visualization: ***DeepSeek-R1-Distill-Qwen-1.5B*** + SofT-GRPO on the 4-th instance of GSM8K.



Figure 12: Visualization: ***DeepSeek-R1-Distill-Qwen-1.5B*** + SofT-GRPO on the 4-th instance of MATH-500.