SPARSE RL RELAXATION FOR LLM REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent work has explored continuous representations in place of discrete tokens during Chain-of-Thought (CoT) reasoning, often referred to as soft CoT. This is motivated by the idea that a mixture of token embeddings can represent a superposition of multiple reasoning possibilities within a single step. However, it remains challenging to train large language models to realize this potential. Existing work has explored reinforcement learning (RL)-style training for soft CoT. However, these approaches treat soft tokens as actions in a Markov decision process rather than as differentiable intermediate computation steps. Thus, they do not propagate gradients across the soft tokens and miss out on the signal of how the choice of one soft token affects the next. In this paper, we propose a sparse relaxation method for RL (called SpaRLax), which leverages the differentiability of soft-CoT reasoning over time steps and makes training more stable and effective. We adopt an entmax sparsity method that adaptively determines the degree of softness, which truncates long-tail noise at confident steps while retaining multiple plausible candidates when uncertain. Across nine reasoning benchmarks, our method achieves the highest average performance compared with baseline methods. Further analyses validate the effective use of soft-token superposition in our approach, and also demonstrate the contribution of adaptive sparsity in our approach.[1]

1 INTRODUCTION

Chain of Thoughts (CoT; Wei et al., 2022) improves the reasoning ability of large language models (LLMs) by generating intermediate steps before the final answer. Despite its effectiveness, CoT produces a single discrete token at each step, collapsing the LLM's output distribution to one token and discarding alternatives that may be useful later. This has motivated recent work on *soft CoT*, where intermediate reasoning tokens are represented by continuous embeddings (Hao et al., 2025; Wang et al., 2025; Zhang et al., 2025; Wu et al., 2025; Shen et al., 2025; Xu et al., 2025a). Among these approaches, Zhang et al. (2025) construct each CoT token as a probability-weighted mixture of token embeddings, referred to as a *soft token*, allowing multiple candidate semantics to coexist at a reasoning step. Theoretical analyses (Zhu et al., 2025) further show that such continuous representation can represent multiple reasoning paths in superposition and can be more expressive than discrete CoT, suggesting soft CoT as a promising direction for improving LLM reasoning.

However, training LLMs to realize the potential of soft reasoning remains challenging. Recent methods struggle to fit soft CoT training into the reinforcement learning (RL) framework, which allows us to learn soft CoT without groundtruth labels. To facilitate RL, researchers have introduced various stochasticity mechanisms to soft CoT, such as Gaussian embedding perturbations (Butt et al., 2026) and Gumbel-softmax sampling (Zheng & Lee, 2025). However, such a formulation is problematic, as the output-generating process defines the vocabulary as the action space[2] but soft reasoning considers the embedding as an action. They do not coexist in the same Markov decision process (MDP), making the action space, and thus the entire RL formulation, ill-defined. In addition, such a formulation fails to take advantage of the differentiable nature of soft tokens, as the gradient is not back-propagated across the soft tokens. This misses out on the signal of how the choice of one soft token affects the next.

[1] Our code is released at `https://anonymous.4open.science/r/SpaRLax-1736`

[2] In the standard token-level MDP for autoregressive LLMs, the policy is defined on vocabulary tokens $a_i \in V$ as $\pi_{\boldsymbol{\theta}}(a_i|s_i)$ for some state s_i. A soft token $\tilde{\boldsymbol{e}}_i \notin V$ is not in this action space, so a policy-gradient objective over soft tokens is ill-defined under standard RL frameworks for LLMs (Ramamurthy et al., 2023).

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.

In this paper, we propose a Sparse RL Relaxation (SpaRLax) method to train soft reasoning for LLMs. For every reasoning step, we sample a mixture probability by Gumbel-softmax (Jang et al., 2017; Maddison et al., 2017), which introduces stochasticity compared with directly using softmax. Different from Zheng & Lee (2025), however, we notice that Gumbel-softmax is differentiable, and with this insight, we restore the cross-step gradient path, allowing gradients to flow backward from each soft token to the one before it. Moreover, we notice an additional challenge that the typical large vocabulary size of an LLM dilutes probability mass, causing a loss of reasoning focus. We propose a solution to this problem by using sparse relaxation, which concentrates probability mass on a few candidate tokens, while assigning exactly zero probability to low-scoring tokens. In particular, we apply entmax (Peters et al., 2019) on top of the Gumbel reparametrization trick when constructing the soft-token distribution, as entmax can naturally adapt the degree of softness by the uncertainty of each reasoning step.

We conduct experiments on nine benchmarks spanning mathematical, knowledge-intensive, and code reasoning. We compare our method against soft-CoT post-training methods under a controlled GRPO framework. The results show that our SpaRLax achieves the strongest weighted-average performance under both soft- and hard-CoT inference, demonstrating the benefits of cross-step learning in the relaxed reasoning space (§4.2). We also observe that our approach utilizes the superposition encoded by soft tokens more than existing methods (§4.3). In addition, we analyze the sparsity mechanism across different candidate-set sizes and find that entmax outperforms dense softmax, suggesting its effectiveness in adaptively suppressing long-tail noise (§4.4).

2 PRELIMINARIES

Classic CoT reasoning. In classic CoT reasoning (Wei et al., 2022), we generate a sequence of discrete reasoning tokens, typically enclosed by special tokens such as $\langle\text{think}\rangle$ and $\langle/\text{think}\rangle$, before producing the final answer. We denote an input prompt by $\boldsymbol{x}$, the reasoning tokens by $\boldsymbol{t}_{1:m} = (t_1, \ldots, t_m)$, and the final answer tokens by $\boldsymbol{y}_{1:n} = (y_1, \ldots, y_n)$. The LLM's autoregressive sequence is given by

$$\boldsymbol{x} \to \langle\text{think}\rangle \underbrace{t_1, \cdots, t_m}_{\text{discrete CoT}} \langle/\text{think}\rangle \underbrace{y_1, \cdots, y_n}_{\text{final answer}}. \tag{1}$$

At the ith reasoning step, the LLM with its policy $\pi_{\boldsymbol{\theta}}(\cdot|\cdot)$ yields a probability vector $\boldsymbol{p}_i \in \Delta^{|\mathcal{V}|-1}$ over the vocabulary $\mathcal{V}$ and samples a token t_i by

$$\boldsymbol{p}_i = \pi_{\boldsymbol{\theta}}\left(\cdot|\boldsymbol{x}, \boldsymbol{t}_{<i}\right), \qquad t_i \sim \text{Categorical}(\boldsymbol{p}_i). \tag{2}$$

Once t_i is sampled, its embedding, denoted by $\text{emb}(t_i)$, is passed to the next step, while the remaining information in $\boldsymbol{p}_i$ is not carried forward. After $\langle/\text{think}\rangle$, the LM switches to answer generation.

Soft CoT reasoning. Zhang et al. (2025) propose soft CoT, which replaces the discrete token t_i in classic CoT with a continuous soft-token embedding to improve CoT expressivity. In this paradigm, the model predicts the soft-token distribution $\tilde{\boldsymbol{p}}_i$ and constructs the soft-token embedding by

$$\tilde{\boldsymbol{p}}_i = \pi_{\boldsymbol{\theta}}\left(\cdot|\boldsymbol{x}, \tilde{\boldsymbol{e}}_{<i}\right), \qquad \tilde{\boldsymbol{e}}_i = \sum_{k=1}^{|\mathcal{V}|} \tilde{p}_{i,k}\, \text{emb}(k), \tag{3}$$

where $\tilde{p}_{i,k}$ is the kth element of $\tilde{\boldsymbol{p}}$. The resulting $\tilde{\boldsymbol{e}}_i$ is fed back to the model for the next step until $\langle/\text{think}\rangle$ is generated (details can be seen in Appendix A.3). Then the LLM generates discrete tokens for the final answer, and the entire autoregressive sequence becomes

$$\boldsymbol{x} \to \langle\text{think}\rangle \underbrace{\tilde{\boldsymbol{e}}_1, \tilde{\boldsymbol{e}}_2, \cdots, \tilde{\boldsymbol{e}}_m}_{\text{soft CoT}} \langle/\text{think}\rangle \underbrace{y_1, y_2, \cdots, y_n}_{\text{(discrete) final answer}}. \tag{4}$$

RL training for LLMs. LLMs are often post-trained with reinforcement learning (RL) to improve reasoning ability, where the model explores different reasoning trajectories and is rewarded for trajectories that lead to correct answers. For a prompt $\boldsymbol{x}$, the model samples a trajectory including CoT $t_{1:m}$ and final answer $y_{1:n}$. Then, the model receives a scalar reward $R(\boldsymbol{x}, \boldsymbol{y}_{1:n})$ for an entire

trajectory, based on the correctness of the final answer for the input, but typically not based on CoT steps. RL optimizes the policy parameter $\boldsymbol{\theta}$ to maximize the expected reward:

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D},\, \boldsymbol{t}_{1:m}, \boldsymbol{y}_{1:n}\sim\pi_{\boldsymbol{\theta}}} \left[R(\boldsymbol{x}, \boldsymbol{y}_{1:n})\right], \tag{5}$$

where $\mathcal{D}$ denotes the distribution of training prompts.

Because discrete token sampling is non-differentiable, policy-gradient methods (Williams, 1992; Sutton et al., 1999) can be used to differentiate Eqn. (5):

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\substack{\boldsymbol{x}\sim\mathcal{D},\\ \boldsymbol{t}_{1:m},\, \boldsymbol{y}_{1:n}\sim\pi_{\boldsymbol{\theta}}}} \left[R(\boldsymbol{x}, \boldsymbol{y}_{1:n}) \left(\sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}\left(t_i | \boldsymbol{x}, \boldsymbol{t}_{<i}\right) + \sum_{j=1}^{n} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}\left(y_j | \boldsymbol{x}, \boldsymbol{t}_{1:m}, \boldsymbol{y}_{<j}\right)\right)\right]. \tag{6}$$

Recent studies extend the likelihood-based formulation to soft reasoning. Let $\rho_{\boldsymbol{\theta}}(\tilde{\boldsymbol{e}}_i | \boldsymbol{x}, \tilde{\boldsymbol{e}}_{<i})$ denote the likelihood of a sampled soft token by the Gaussian- (Butt et al., 2026) or Gumbel-based (Zheng & Lee, 2025) stochastic mechanism. Their policy gradient takes the form

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\substack{\boldsymbol{x}\sim\mathcal{D},\\ \tilde{\boldsymbol{e}}_{1:m}\sim\rho_{\boldsymbol{\theta}},\, \boldsymbol{y}_{1:n}\sim\pi_{\boldsymbol{\theta}}}} \left[R(\boldsymbol{x}, \boldsymbol{y}_{1:n}) \left(\sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} \log \rho_{\boldsymbol{\theta}}\left(\tilde{\boldsymbol{e}}_i | \boldsymbol{x}, \tilde{\boldsymbol{e}}_{<i}\right) + \sum_{j=1}^{n} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}\left(y_j | \boldsymbol{x}, \tilde{\boldsymbol{e}}_{1:m}, \boldsymbol{y}_{<j}\right)\right)\right]. \tag{7}$$

Such methods view the soft token defined by Eqn. (3) as a continuous RL action, which is less justified under the standard Markov decision process (MDP) framework for autoregressive LLMs. This is because an LLM treats discrete vocabulary tokens as the action space, but it becomes unclear how a soft token embedding is compatible with discrete actions of tokens in the vocabulary. In addition, soft tokens could have been modeled as differentiable variables, but treating them as RL actions breaks this differentiable structure. Hence, these methods miss out on the gradient signal of how the choice of one soft token affects the later reasoning steps.

3 Our Method: Sparse RL Relaxation (SpaRLax)

In this section, we introduce the proposed SpaRLax approach in detail. In §3.1, we discuss our RL relaxation based on Gumbel-softmax; we also present an efficient approximation for implementation considerations. Then, in §3.2 we introduce sparsity by replacing Gumbel-softmax with Gumbel-*entmax*, yielding sparse soft tokens whose degree of softness adapts across reasoning steps.

3.1 RL Relaxation

Soft CoT naturally forms a differentiable autoregressive computation across reasoning steps. We exploit this structure and propose an RL relaxation method. At reasoning step i, the LLM $f_{\boldsymbol{\theta}}$ produces pre-softmax logits

$$\boldsymbol{z}_i = f_{\boldsymbol{\theta}}\left(\boldsymbol{x}, \tilde{\boldsymbol{e}}_{<i}\right) \in \mathbb{R}^{|\mathcal{V}|}. \tag{8}$$

To introduce stochastic exploration, we perturb each logit with i.i.d. Gumbel noise $\epsilon_{i,k} \sim \mathrm{Gumbel}(0, 1)$. By the Gumbel-Max trick (Maddison et al., 2014), the argmax of the perturbed logits is an exact categorical sample from $\mathrm{softmax}(\boldsymbol{z}_i)$. For differentiability, the argmax operation is replaced with softmax, resulting in the Gumbel-softmax operation (Jang et al., 2017; Maddison et al., 2017). For the next step $i + 1$, the input is a soft token embedding, given by

$$\tilde{\boldsymbol{p}}_i = \mathrm{softmax}\left(\frac{\boldsymbol{z}_i + \boldsymbol{\epsilon}_i}{\tau_g}\right), \qquad \tilde{\boldsymbol{e}}_{i+1} = \sum_{k=1}^{|\mathcal{V}|} \tilde{p}_{i,k}\, \mathrm{emb}(k), \tag{9}$$

where $\tau_g > 0$ is the relaxation temperature (Goyal et al., 2017). As seen, given sampled $\boldsymbol{\epsilon}_i$, the relaxed soft token $\tilde{\boldsymbol{e}}_i$ remains differentiable with respect to $\boldsymbol{z}_i$.

After the reasoning steps, the LLM generates a discrete output $\boldsymbol{y}_{1:n}$ by sampling from its policy. The training objective is to maximize the expected reward:

$$J_{\mathrm{relax}}(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\substack{\boldsymbol{x}\sim\mathcal{D},\, \boldsymbol{\epsilon}_{1:m}\sim\mathrm{Gumbel}(0,1)\\ \boldsymbol{y}_{1:n}\sim\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x}, \tilde{\boldsymbol{e}}_{1:m})}} \left[R(\boldsymbol{x}, \boldsymbol{y}_{1:n})\right] \tag{10}$$

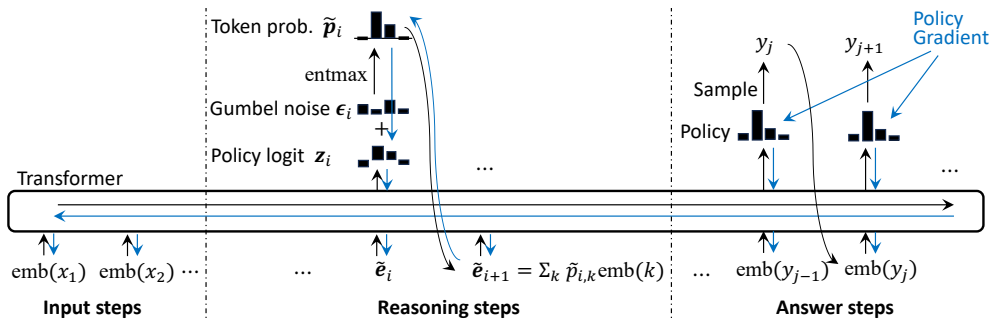


Figure 1: Overview of our approach. **Black** arrows indicate forward propagation, and **blue** arrows indicate backpropagation. Different from Soft-GRPO (Zheng & Lee, 2025), we only apply policy gradient to output steps, but backpropagate cross-step gradients for reasoning steps.

In other words, Gumbel-softmax allows us to sample from unparametrized Gumbel noise, eliminating the parameters underneath the expectation operation. This facilitates backpropagation through the reasoning steps. The gradient of Eqn. (10) can be written as

$$\nabla_{\boldsymbol{\theta}} J_{\text{relax}}(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\substack{\boldsymbol{x}\sim\mathcal{D},\, \boldsymbol{\epsilon}_{1:m}\sim\text{Gumbel}(0,1) \\ \boldsymbol{y}_{1:n}\sim\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x},\tilde{\boldsymbol{e}}_{1:m})}} \left[R(\boldsymbol{x}, \boldsymbol{y}_{1:n}) \sum_{j=1}^{n} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}} \left(y_j | \boldsymbol{x}, \tilde{\boldsymbol{e}}_{1:m}, \boldsymbol{y}_{<j} \right) \right], \tag{11}$$

where the policy gradient applies to the discrete answer tokens $\boldsymbol{y}_{1:m}$ only, which differs from previous work (Zheng & Lee, 2025; Butt et al., 2026) treating soft reasoning tokens as actions. As illustrated in Fig. 1, the reasoning trajectory remains differentiable through the relaxed soft-token construction, allowing the learning signal to propagate across reasoning steps.

Efficient approximation for implementation considerations. Modern RL implementations (Sheng et al., 2025; Fu et al., 2025) adopt a two-pass strategy: (1) The first pass performs roll-out by generating an entire trajectory without backpropagation. This does not require us to store activation values of the neural network and is memory-efficient, but it is slow because tokens are generated autoregressively. (2) The second pass takes in the sampled actions for gradient calculation, which is memory-intensive but fast. The two-pass strategy is essential for performing RL on LLMs, as it allows us to perform the roll-out and policy update (of different batches) simultaneously, so as to maximize GPU parallelization (Fu et al., 2025).

However, this presents a challenge for our backpropagation. Note that the sampled soft token embeddings are generated in the first pass, but are fed as input in the second pass, thus detached from the computational graph. Consequently, there is no gradient flow from $\tilde{\boldsymbol{e}}_i$ to $\tilde{\boldsymbol{p}}_{i-1}$, $\tilde{\boldsymbol{z}}_{i-1}$, and $\tilde{\boldsymbol{e}}_{i-1}$.

Inspired by surrogate-gradient methods (Bengio et al., 2013; Wang et al., 2021), we propose a *proxy gradient grafting* technique that connects the gradient from $\tilde{\boldsymbol{e}}_i$ to its previous step. The key idea is to preserve the cached rollout values in the forward pass, while recomputing each soft-token distribution under the current policy and using its gradient in the backward pass.

Let $(\tilde{\boldsymbol{p}}_i^{\text{roll}}, \tilde{\boldsymbol{e}}_i^{\text{roll}}, \boldsymbol{\epsilon}_i)$ denote the soft-token distribution and embedding cached from rollout, together with the sampled Gumbel noise $\boldsymbol{\epsilon}_i$. In a traditional implementation, the logits $\boldsymbol{z}_i$ are computed from $\tilde{\boldsymbol{e}}_i^{\text{roll}}$. However, we reconstruct it from $\tilde{\boldsymbol{e}}_{i-1}^{\text{roll}}$ at one-step earlier:

$$\boldsymbol{z}_{i-1}^{\text{proxy}} = f_\theta \left(\boldsymbol{x}, \tilde{\boldsymbol{e}}_{\le i-1}^{\text{roll}} \right), \qquad \tilde{\boldsymbol{p}}_{i-1}^{\text{proxy}} = \text{softmax}\left(\frac{\boldsymbol{z}_{i-1}^{\text{proxy}} + \boldsymbol{\epsilon}_{i-1}}{\tau_g} \right). \tag{12}$$

Here, we reuse the same $\boldsymbol{\epsilon}_{i-1}$ sampled during rollout, so we have $\boldsymbol{z}_{i-1}^{\text{proxy}} = \boldsymbol{z}_{i-1}^{\text{roll}}$. Then, we graft the proxy gradient onto the cached one for calculating the soft-token embedding for the next step.

$$\bar{\boldsymbol{p}}_{i-1} = \text{sg}\left(\tilde{\boldsymbol{p}}_{i-1}^{\text{roll}} \right) + \tilde{\boldsymbol{p}}_{i-1}^{\text{proxy}} - \text{sg}\left(\tilde{\boldsymbol{p}}_{i-1}^{\text{proxy}} \right), \qquad \bar{\boldsymbol{e}}_i = \sum_{k=1}^{|\mathcal{V}|} \bar{p}_{i-1,k}\, \text{emb}(k), \tag{13}$$

where $\mathrm{sg}(\cdot)$ denotes stop gradient of a variable. Then, $\bar{\boldsymbol{e}}_i$ replaces $\tilde{\boldsymbol{e}}_i^{\text{roll}}$ in the standard second pass of the policy update. Note that the cached variable $\tilde{\boldsymbol{p}}_{i-1}^{\text{roll}}$ is already detached due to the two-pass implementation, but we explicitly add the sg operation in the formula for clarity.

With this construction, the forward computation exactly preserves the cached roll-out trajectory, as we always have $\bar{\boldsymbol{p}}_{i-1} = \tilde{\boldsymbol{p}}_{i-1}^{\text{roll}}$. However, we can now let the gradient flow from $\bar{\boldsymbol{e}}_i$ to $\tilde{\boldsymbol{p}}_{i-1}^{\text{proxy}}$ by standard backpropagation.

In terms of efficiency, our grafting technique does not have high latency, as we only graft one earlier step. Therefore, Eqns. (12) and (13) can be computed in parallel for all time steps i. However, this grafting trick has a limitation, as the gradient backpropagates to only one earlier step from $\bar{\boldsymbol{e}}_i$ to $\tilde{\boldsymbol{p}}_{i-1}^{\text{proxy}}$ (for each value of i). That said, all steps can be adequately trained because the attention layers backpropagate gradients to all earlier steps. Moreover, we argue that our cross-step gradient is important even for one step. This is because the nearest step receives the strongest signal, as the gradient tends to vanish as the number of steps increases in recurrent computation (Bengio et al., 1994; Pascanu et al., 2013). Our method provides meaningful signal on how one soft token should affect the next.

Compatibility with RL post-training objectives. Our RL relaxation is independent of the particular policy-optimization rule used for RL post-training. It modifies how the soft-CoT computation is constructed and differentiated, while the RL policy optimization is an outer layer often involving advantage estimation, importance weighting, and policy-update mechanism. Thus, our RL relaxation can be combined with commonly used post-training algorithms such as PPO (Schulman et al., 2017), GRPO (Shao et al., 2024), DAPO (Yu et al., 2025), GSPO (Zheng et al., 2025), and REINFORCE++ (Hu, 2025).

3.2 Sparse RL Relaxation

Prior soft-CoT methods (Zhang et al., 2025; Zheng & Lee, 2025; Zhou et al., 2026) use softmax to map LLM logits into a soft-token distribution, assigning non-zero probability to every vocabulary token. This mixes a long tail of low-probability tokens into soft token embeddings, which introduces unnecessary noise aggregated over a large vocabulary. A common remedy is top-k truncation, which limits the mixture to a fixed number of highest-scoring tokens. However, a fixed k cannot accommodate the varying uncertainty across reasoning steps. As observed by Fu & Luo (2026), token distributions can be highly concentrated at some steps but much more diffuse at others. Thus, a large fixed k may retain unnecessary noise, while a small k may discard useful alternatives.

To suppress this long-tail noise while retaining multiple candidates when necessary, we replace the softmax relaxation in Eqn. (9) with α-entmax (Peters et al., 2019), yielding an adaptive sparse RL Relaxation. For logits $\boldsymbol{z} \in \mathbb{R}^{|\mathcal{V}|}$, entmax is defined by

$$\mathrm{entmax}_\alpha(\boldsymbol{z})_k = [(\alpha-1)z_k - \lambda]_+^{\frac{1}{\alpha-1}}, \tag{14}$$

where λ normalizes the distribution and $\alpha > 1$ controls its sparsity, with $\alpha \to 1$ recovering $\mathrm{softmax}$ and $\alpha = 2$ corresponding to $\mathrm{sparsemax}$ (Martins & Astudillo, 2016). For $\alpha > 1$, entmax assigns exactly zero probability to sufficiently weak tokens, while the number of retained tokens is determined by their relative scores. Accordingly, Sparse RL Relaxation constructs the soft-token distribution as

$$\tilde{\boldsymbol{p}}_i = \mathrm{entmax}_\alpha\left(\frac{\boldsymbol{z}_i + \boldsymbol{\epsilon}_i}{\tau_g}\right), \qquad \tilde{\boldsymbol{e}}_{i+1} = \sum_{k=1}^{|\mathcal{V}|} \tilde{p}_{i,k}\,\mathrm{emb}(k). \tag{15}$$

If only one logit dominates (after Gumbel noise in our scenario), entmax suppresses the remaining tokens, and the soft token approaches a discrete representation. When several candidates have comparable logits, it retains them as a sparse mixture. Thus, the degree of softness adapts to the uncertainty of each reasoning step without requiring a manually specified support size.

The same replacement can be applied when reconstructing $\tilde{\boldsymbol{p}}_i^{\text{proxy}}$ during proxy gradient grafting. Since entmax remains differentiable on its active support, gradients can propagate through the sparse soft-token construction in the same manner as in §3.1.

Method	Math Reasoning						Knowledge	Code		W.Avg.
	GSM8K	MATH	AIME24	AIME25	AMC23	Olym.	GPQA	HEval	MBPP	
Hard CoT Inference										
Initial	80.08 ± 0.73	75.18 ± 0.96	15.94 ± 4.12	20.94 ± 2.90	59.69 ± 4.95	26.24 ± 0.75	20.55 ± 2.61	55.11 ± 2.19	58.55 ± 1.87	59.58 ± 0.48
Discrete-GRPO	84.42 ± 0.72	75.97 ± 0.95	16.98 ± 4.26	21.25 ± 2.84	62.11 ± 5.82	27.31 ± 0.74	21.26 ± 2.23	56.23 ± 2.31	**59.45** ± 2.14	61.90 ± 0.38
Soft-GRPO	84.29 ± 0.55	75.48 ± 1.38	15.94 ± 4.46	20.94 ± 3.51	58.91 ± 5.35	26.25 ± 0.87	20.15 ± 1.90	55.22 ± 2.84	57.66 ± 1.88	61.24 ± 0.47
STHT	80.35 ± 0.58	76.49 ± 1.02	13.85 ± 4.65	22.92 ± 3.28	62.11 ± 5.65	27.36 ± 0.75	22.77 ± 2.30	55.16 ± 2.00	58.18 ± 2.04	60.28 ± 0.43
LEPO	83.36 ± 1.26	76.98 ± 1.83	**19.58** ± 6.07	21.64 ± 4.34	61.22 ± 7.45	26.73 ± 1.34	22.38 ± 3.23	56.62 ± 4.07	59.41 ± 3.45	61.21 ± 1.23
SpaRLax (ours)	**85.16** ± 0.53	**78.24** ± 1.09	16.77 ± 3.92	**22.97** ± 3.16	**68.83** ± 4.31	**29.19** ± 0.76	**27.43** ± 2.59	**57.28** ± 2.32	58.82 ± 1.89	**63.43** ± 0.42
Soft CoT Inference										
Initial	80.59 ± 0.65	78.92 ± 1.09	16.98 ± 4.96	24.11 ± 3.99	67.97 ± 5.06	29.75 ± 0.80	32.62 ± 3.03	58.38 ± 2.03	60.70 ± 2.40	62.33 ± 0.49
Discrete-GRPO	84.93 ± 0.69	79.80 ± 1.00	16.35 ± 4.67	**25.05** ± 3.24	68.59 ± 3.96	30.61 ± 0.69	32.67 ± 2.76	**59.98** ± 3.10	**61.52** ± 1.77	64.58 ± 0.46
Soft-GRPO	84.96 ± 0.57	79.15 ± 1.46	17.81 ± 5.26	23.18 ± 4.07	66.56 ± 4.39	30.34 ± 0.66	33.43 ± 2.67	59.78 ± 2.21	60.60 ± 1.64	64.35 ± 0.45
STHT	80.45 ± 0.73	79.43 ± 1.32	17.19 ± 4.33	24.32 ± 3.85	68.98 ± 5.11	30.63 ± 0.90	**34.77** ± 2.31	59.17 ± 1.83	60.35 ± 2.07	62.70 ± 0.41
LEPO	83.25 ± 1.65	76.33 ± 1.57	11.88 ± 8.99	17.19 ± 6.48	61.64 ± 7.48	28.99 ± 1.72	28.36 ± 5.89	56.04 ± 4.56	59.03 ± 3.25	62.00 ± 0.71
SpaRLax (ours)	**86.34** ± 0.72	**80.01** ± 1.11	**18.33** ± 4.23	23.39 ± 4.55	**69.84** ± 4.71	**31.20** ± 0.74	34.72 ± 2.98	59.82 ± 2.53	60.83 ± 1.89	**65.42** ± 0.51

Table 1: Pass@1 performance (%) under hard and soft chain-of-thought (CoT) inference. Results are reported as mean ± standard deviation. **Bold** and underlined values indicate the best and second-best mean performance within each inference setting, respectively.

Method	Math Reasoning						Knowledge	Code		W.Avg.
	GSM8K	MATH	AIME24	AIME25	AMC23	Olym.	GPQA	HEval	MBPP	
Hard CoT Inference										
Initial	**97.50**	91.60	**50.00**	55.00	87.50	42.07	82.32	83.54	86.77	81.22
Discrete-GRPO (Guo et al., 2025)	**97.50**	91.80	46.67	**58.33**	95.00	42.96	82.32	**84.76**	**87.16**	81.65
SofT-GRPO (Zheng & Lee, 2025)	97.35	91.80	40.00	53.33	90.00	41.19	74.75	81.71	84.05	80.14
STHT (Butt et al., 2026)	97.12	91.60	46.67	56.67	**97.50**	42.37	85.35	83.54	84.82	81.28
LEPO (Zhou et al., 2026)	97.19	**92.20**	46.67	53.33	95.00	43.41	**90.40**	82.32	86.38	81.90
SpaRLax (ours)	97.35	91.80	**50.00**	**58.33**	95.00	**44.15**	**90.40**	84.15	85.60	**82.21**
Soft CoT Inference										
Initial	97.35	**93.60**	46.67	60.00	95.00	46.52	96.97	83.54	85.99	83.38
Discrete-GRPO (Guo et al., 2025)	97.12	93.00	50.00	**61.67**	95.00	46.52	94.95	85.37	87.16	83.32
SofT-GRPO (Zheng & Lee, 2025)	97.27	93.40	**56.67**	58.33	92.50	45.33	96.46	**85.98**	87.55	83.32
STHT (Butt et al., 2026)	**97.42**	**93.60**	46.67	**61.67**	**97.50**	45.93	**97.47**	84.76	85.60	83.41
LEPO (Zhou et al., 2026)	97.27	91.40	30.00	48.33	92.50	43.41	92.42	82.32	87.55	81.75
SpaRLax (ours)	97.35	93.00	50.00	58.33	95.00	**47.11**	96.46	84.15	**87.94**	**83.56**

Table 2: Pass@32 performance (%) under hard and soft chain-of-thought (CoT) inference. **Bold** and underlined values denote the best and second-best results within each inference setting, respectively.

4 EXPERIMENTS

4.1 SETUP

Models and datasets. We use DeepSeek-R1-Distill-Qwen-1.5B[3] as the base LLM for all experiments. We perform RL training on DAPO-MATH-17k (Yu et al., 2025), a collection of mathematical reasoning problems, which is selected to reduce potential overlap with the base model's pretraining corpus, allowing the effect of RL fine-tuning to be evaluated cleanly.

We evaluate the trained models on nine benchmarks spanning multiple reasoning domains. For mathematical reasoning, we use GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), AIME 2024, AIME 2025, AMC 2023, and OlympiadBench (He et al., 2024). We further evaluate general knowledge and scientific reasoning on GPQA-Diamond (Rein et al., 2024), and code reasoning on HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021). We compute the weighted average performance based on the number of data points in each test dataset.

Competing approaches. We compare our SpaRLax method against representative discrete- and soft-reasoning baselines. **Initial** denotes the base model without training. **Discrete-GRPO** applies standard GRPO (Shao et al., 2024) to discrete-token CoT. For RL with soft reasoning, we compare against **SofT-GRPO** (Zheng & Lee, 2025), **STHT** (Butt et al., 2026), and **LEPO** (Zhou et al., 2026).

[3] https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B

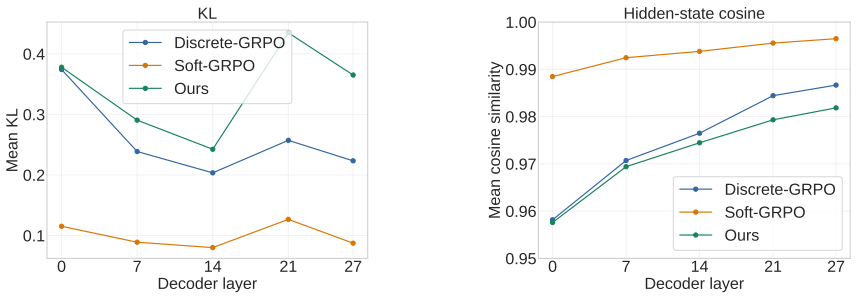


Figure 2: Comparing the hidden states when using the soft token embedding and replacing it with top-1 token embedding. Higher KL and lower cosine indicate a stronger utilization of superposition.

To ensure a controlled comparison, we implement all methods within the same GRPO framework, meaning that SofT-GRPO, STHT, LEPO, and our method modify only the components required by their respective soft-CoT learning mechanisms. This experimental protocol minimizes performance differences caused by the choice of the underlying RL algorithm and allows the experiments to focus on the effectiveness of the soft-CoT stage. Appendix A.1 provides the pseudo-code of our SpaRLax under the GRPO framework.

Training and evaluation settings. During training, our SpaRLax uses soft tokens for reasoning steps, but for the answer steps, hard tokens are generated by top-p sampling (with $p = 0.95$) in conjunction with top-k tokens ($k = 5$), which is standard in previous work (Zheng & Lee, 2025; Zhou et al., 2026). After training, our model supports either soft CoT or hard CoT (simply sampling discrete tokens for training steps). More configurations can be found in Appendix A.2.

For evaluation, we report Pass@1 (single-sample accuracy) and Pass@32 (accuracy of whether at least one correct solution is found among 32 sampled responses). They are commonly used in previous literature . To account for the variance of Pass@1 induced by stochastic sampling, we repeat each evaluation four times from the same checkpoint and report the mean and standard deviation.

4.2 Main Results

Based on the above setups, we conduct post-training from the same initial model, select the checkpoint with the best validation performance, and evaluate each model with both hard-CoT and soft-CoT inference. We report Pass@1 and Pass@32 in Tabs. 1–2.

Pass@1. We observe that soft-CoT inference on the initial model already provides a notable improvement over hard-CoT inference, consistent with prior work (Zhang et al., 2025). This advantage persists after post-training, as soft-CoT inference generally achieves higher Pass@1 than hard-CoT inference across different post-training methods. Meanwhile, soft CoT post-training also brings notable gains under hard-CoT inference, consistent with prior observations (Butt et al., 2026).

Our method achieves the strongest Pass@1 under both inference modes, reaching 63.43 with hard CoT and 65.42 with soft CoT. Soft CoT provides a richer training space and more expressive inference representation (Zhu et al., 2025). Our method propagates the gradient of reward across soft tokens, providing a learning signal on how the choice of one soft token affects the next. This enables more effective training for soft reasoning in such a richer training space. As a result, SpaRLax achieves strong improvements under both soft CoT and hard CoT inference.

Pass@32. Pass@32 is much higher than Pass@1 for all methods, which is expected as Pass@32 only requires one successful solution among 32 stochastic rollouts. Soft CoT inference continues to outperform hard CoT inference at Pass@32, consistent with observations in prior work (Zheng & Lee, 2025; Butt et al., 2026). Although the gain of our method is much smaller than those observed for Pass@1, other post-training methods yield even smaller Pass@32 improvements over the initial model. This behavior is in line with prior observations that, under moderate RL budgets, post-training improves sampling efficiency but does not expand large-k solution coverage (Yue et al., 2025; Liu et al., 2025). That being said, our method achieves the largest average improvement over the initial model among all competing methods.

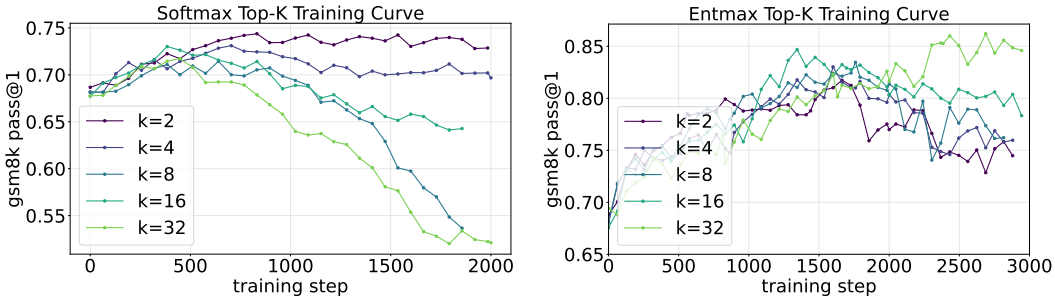


Figure 3: Effect of candidate-set size on softmax and entmax training. Softmax becomes increasingly sensitive to larger k, whereas entmax remains robust as k increases.

4.3 Superposition Utilization

Rizvi-Martel et al. (2026) show that recent soft reasoning methods often collapse their superposed representation toward a single token, indicating the observed gain may be associated simply with more greedy reasoning. In this subsection, we analyze if our method suffers from the same collapse or actually makes use of the multiple alternatives encoded in soft tokens.

Following Rizvi-Martel et al. (2026), we examine whether the information beyond the top-1 component of the soft token persists in the model's hidden states. For each data sample and each reasoning step i, we keep all preceding soft tokens $\tilde{\boldsymbol{e}}_{<i}$ unchanged and perform two forward passes that differ only at the current position: one uses the original soft token $\tilde{\boldsymbol{e}}_i$, while the other replaces it with its top-1 token's embedding. This isolates the effect of the non-top-1 components at a single reasoning step. At each Transformer layer, we compare the resulting hidden states using cosine similarity and symmetric KL divergence between their Logit-Lens distributions.[4] We compute these metrics for every inference step of every sample, and then average them across inference steps and data samples for each layer. The resulting layerwise averages are plotted in Fig. 2. Higher KL divergence and lower cosine similarity indicate that the non-top-1 components have a stronger effect on subsequent computation, indicating a better utilization of the soft tokens in superposition.

As shown in Fig. 2, our method exhibits larger KL divergence and lower cosine similarity than SofT-GRPO and discrete-GRPO across decoder layers, suggesting stronger utilization of the information retained in the soft-token mixture.

4.4 Effect of Adaptive Sparsity

Constructing a soft token over a large vocabulary requires aggregating all token embeddings, which introduces a huge memory burden and long-tail noise. Recent methods apply top-k truncation, retaining only the highest-scoring k tokens before constructing the soft-token embedding (Zheng & Lee, 2025; Zhou et al., 2026). While a small k reduces both computational cost and long-tail noise, it also imposes a limit on the number of alternatives retained at every reasoning step. This can restrict the superposition capability of soft tokens when more candidates are useful.

Our intuition is that the degree of softness should be step-dependent. As observed in prior work (Fu & Luo, 2026), token distributions are concentrated at most reasoning steps but highly uncertain at a small fraction of steps. Thus, confident steps may require little superposition, whereas uncertain steps can benefit from retaining multiple plausible alternatives. This motivates adaptive sparsity rather than a fixed top-k truncation.

To examine this hypothesis, we compare softmax and entmax under different candidate-set sizes $k \in \{2, 4, 8, 16, 32\}$. For each k, we retain the top-k logits and construct the soft token using either softmax or entmax. Softmax assigns non-zero probability to all retained candidates, whereas entmax can further reduce the active support according to their relative scores. Therefore, k controls

[4] We project an intermediate hidden state $\boldsymbol{h}_i^{(\ell)}$ to the vocabulary space as $\boldsymbol{p}_i^{(\ell)} = \mathrm{softmax}(\boldsymbol{U}\boldsymbol{h}_i^{(\ell)})$, where $\boldsymbol{U}$ is the output unembedding matrix and ℓ denotes the index of transformer layer. Then, we compute the symmetric KL divergence between the distributions obtained from the soft-token and top-1-token forward passes.

the maximum candidate budget, while entmax determines the effective degree of softness within this budget. We train all variants on GSM8K using the same configuration as the main experiments, except that the maximum response length is set to 2,048 tokens for computational efficiency, and record the mean Pass@1 over 4 rollouts at each validation.

Fig. 3 shows a clear difference between the two schemes. With softmax, training becomes increasingly sensitive to k, and larger candidate sets lead to substantial performance degradation. In contrast, entmax improves the performance considerably as k increases, showing the RL training benefits from a larger candidate pool without forcing all candidates to participate in the soft token. These results support our hypothesis that soft reasoning benefits from adaptive rather than fixed sparsity: a larger candidate set provides additional alternatives when needed, while entmax suppresses unnecessary components at confident reasoning steps.

5 Related Work

RL reasoning for LLMs. RL has been shown to be effective for improving LLM reasoning with Chain-of-Thought (Shao et al., 2024; Yu et al., 2025; Hu, 2025; Zheng et al., 2025). However, exploration through discrete token sampling is inefficient over the large space of possible reasoning trajectories (Liao et al., 2025). This exploration bottleneck can be further exacerbated as RL training suppresses valuable low-probability alternatives (Huang et al., 2026), which limits the diversity of reasoning trajectories explored by the model (Wu et al., 2026a; Zhang et al., 2025).

Soft reasoning. Recent studies perform reasoning in a continuous representation space so that a single reasoning step can preserve information from multiple reasoning paths (Hao et al., 2025; Xu et al., 2025a; Wu et al., 2025; Shen et al., 2025). Theoretical analyses show that continuous reasoning representation can encode multiple reasoning paths in superposition and track multiple search frontiers in parallel, providing greater representational expressivity and more efficient exploration than following a single discrete trajectory (Zhu et al., 2025; Gozeten et al., 2026).

Building on this principle, continuous reasoning has been realized in various ways, such as through internal hidden representations (Hao et al., 2025; Xu et al., 2025a; Wu et al., 2025; Shen et al., 2025) and through mixtures of token embeddings (Zhang et al., 2025; Wang et al., 2025; Tang et al., 2026). We focus on the latter formulation, which constructs each soft token as a mixture of vocabulary-token embeddings, preserving multiple token-level alternatives while remaining anchored to the pretrained token embedding space. Subsequent work extends the simple weighted-average soft-token construction through diversified test-time reasoning (Xu et al., 2025b), uncertainty-aware activation (Fu & Luo, 2026; Xu et al., 2026), and stochastic perturbations that mitigate soft-token collapse (Wu et al., 2026b). These works indicate the potential of soft-token-based continuous reasoning for improving the expressiveness, efficiency, and exploration capability of LLM inference.

RL training with soft CoT. While soft CoT improves reasoning at inference time, pretrained LLMs are optimized over discrete token sequences and are never trained to exploit continuous mixtures of token alternatives (Rizvi-Martel et al., 2026), which limits the potential of soft reasoning. This motivates post-training methods for soft reasoning. Recent approaches introduce stochastic soft tokens through Gaussian perturbations or Gumbel-based sampling and optimize them with RL objectives (Butt et al., 2026; Zheng & Lee, 2025; Zhou et al., 2026). However, such optimization does not propagate reward credit through the differentiable dependence between consecutive soft tokens. SpaRLax fills this gap by reparameterizing soft-token generation and propagating pathwise gradients through the soft-token sequence, allowing an earlier soft-token distribution to receive credit according to how it affects subsequent reasoning and the final answer.

6 Conclusion

We propose Sparse RL Relaxation for post-training soft reasoning with differentiable cross-step gradient propagation. Our method further uses entmax-based adaptive sparsity to retain useful alternatives while suppressing low-probability noise. Experiments show consistent gains under both soft- and hard-CoT inference, indicating that improvements learned in the relaxed reasoning space can transfer effectively to discrete decoding. Overall, Sparse RL Relaxation provides an effective framework for learning soft reasoning while remaining compatible with standard discrete inference.

AI USE STATEMENT

In this work, we used OpenAI ChatGPT (GPT-5.6) as an assistive tool for several aspects of implementation and manuscript preparation. In particular, it assisted with extending our FSDP implementation from SP=1 to SP=2, enabling training under more memory-constrained GPU configurations. We also used it for language polishing and readability improvements, as well as for assisting with the generation of LaTeX code for figures and tables. We did not use generative AI to develop the core algorithmic ideas or conceptual framework, formulate mathematical claims or proofs, design the research methodology or experiments, or interpret experimental results. The core algorithmic ideas, motivation, and experimental design of this work were developed by the authors through thinking and discussion, without any AI assistance. All AI-assisted text and code were reviewed by the authors, and the AI-assisted implementation was tested as part of our development and experimental pipeline. We take responsibility for the final content of this work, including text, claims, and code produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide the details necessary to reproduce our results throughout the paper and appendix. §4 describes the details of the models, datasets, baselines, and training and evaluation settings used in our experiments. Appendix A provides more detailed experimental configurations. Our anonymized source code, including training and evaluation scripts and configuration files, is available at `https://anonymous.4open.science/r/SpaRLax-1736`.

REFERENCES

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

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, pp. 157–166, 1994.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Natasha Butt, Ariel Kwiatkowski, Ismail Labiad, Julia Kempe, and Yann Ollivier. Soft tokens, hard truths. In *The Fourteenth International Conference on Learning Representations*, 2026.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

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

Gonçalo M. Correia, Vlad Niculae, and André F. T. Martins. Adaptively sparse transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 2174–2184, 2019.

Renyu Fu and Guibo Luo. SeLaR: Selective latent reasoning in large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7073–7087, 2026.

Wei Fu, Jiaxuan Gao, Xujie Shen, Chen Zhu, Zhiyu Mei, Chuyi He, Shusheng Xu, Guo Wei, Jun Mei, Jiashu Wang, Tongkai Yang, Binhang Yuan, and Yi Wu. AReaL: A Large-Scale Asynchronous Reinforcement Learning System for Language Reasoning. In *Advances in Neural Information Processing Systems*, 2025.

Nuno Gonçalves, Marcos V Treviso, and Andre Martins. AdaSplash: Adaptive sparse flash attention. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 19878–19896, 2025.

Kartik Goyal, Chris Dyer, and Taylor Berg-Kirkpatrick. Differentiable scheduled sampling for credit assignment. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 366–371, 2017.

Alperen Gozeten, Muhammed Ildiz, Xuechen Zhang, Hrayr Harutyunyan, Ankit Singh Rawat, and Samet Oymak. Continuous chain of thought enables parallel exploration and reasoning. In *International Conference on Learning Representations*, pp. 82881–82914, 2026.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason E Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. In *Second Conference on Language Modeling*, 2025.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with Olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems*, 2021.

Jian Hu. REINFORCE++: A simple and efficient approach for aligning large language models. *arXiv preprint arXiv:2501.03262*, 2025.

Guanhua Huang, Tingqiang Xu, Mingze Wang, Qi Yi, Xue Gong, Siheng Li, Ruibin Xiong, Kejiao Li, Yuhao Jiang, and Bo Zhou. Low-probability tokens sustain exploration in reinforcement learning with verifiable reward. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 24158–24188, 2026.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with Gumbel-softmax. In *International Conference on Learning Representations*, 2017.

Mengqi Liao, Xiangyu Xi, Ruinian Chen, Jia Leng, Yangen Hu, Ke Zeng, Shuai Liu, and Huaiyu Wan. Enhancing efficiency and exploration in reinforcement learning for LLMs. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 1451–1463, 2025.

Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. Prorl: Prolonged reinforcement learning expands reasoning boundaries in large language models. In *Advances in Neural Information Processing Systems*, 2025.

Chris J. Maddison, Daniel Tarlow, and Tom Minka. A* sampling. In *Advances in Neural Information Processing Systems*, 2014.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2017.

André F. T. Martins and Ramón Fernandez Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International Conference on Machine Learning*, pp. 1614–1623, 2016.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, pp. 1310–1318, 2013.

Ben Peters, Vlad Niculae, and André F. T. Martins. Sparse sequence-to-sequence models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1504–1519, 2019.

Rajkumar Ramamurthy, Prithviraj Ammanabrolu, Kianté Brantley, Jack Hessel, Rafet Sifa, Christian Bauckhage, Hannaneh Hajishirzi, and Yejin Choi. Is reinforcement learning (not) for natural language processing: Benchmarks, baselines, and building blocks for natural language policy optimization. In *International Conference on Learning Representations*, 2023.

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.

Michael Rizvi-Martel, Guillaume Rabusseau, and Marius Mosbach. The illusion of superposition? A principled analysis of latent thinking in language models. In *Third Conference on Language Modeling*, 2026.

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, Y. K. Li, Y. Wu, and Daya Guo. 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. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 677–693, 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. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.

Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, pp. 1057–1063, 1999.

Yao Tang, Li Dong, Yaru Hao, Qingxiu Dong, Furu Wei, and Jiatao Gu. Multiplex thinking: Reasoning via token-wise branch-and-merge. *arXiv preprint arXiv:2601.08808*, 2026.

Maxat Tezekbayev, Vassilina Nikoulina, Matthias Gallé, and Zhenisbek Assylbekov. Speeding up entmax. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pp. 1142–1158, 2022.

Kang Wang, Xiangyu Duan, and Tianyi Du. Improving latent reasoning in LLMs via soft concept mixing. *arXiv preprint arXiv:2511.16885*, 2025.

Zhuo Wang, Wei Zhang, Ning Liu, and Jianyong Wang. Scalable rule-based representation learning for interpretable classification. In *Advances in Neural Information Processing Systems*, pp. 30479–30491, 2021.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, pp. 24824–24837, 2022.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, pp. 229–256, 1992.

Haoyi Wu, Zhihao Teng, and Kewei Tu. Parallel continuous chain-of-thought with jacobi iteration. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 914–926, 2025.

Junda Wu, Yuxin Xiong, Xintong Li, Sheldon Yu, Zhengmian Hu, Tong Yu, Rui Wang, Xiang Chen, Jingbo Shang, and Julian McAuley. CTRLS: Chain-of-thought reasoning via latent state transition. In *Proceedings of the 29th International Conference on Artificial Intelligence and Statistics*, pp. 2728–2736, 2026a.

Junhong Wu, Jinliang Lu, Zixuan Ren, Gangqiang Hu, Zhi Wu, Dai Dai, and Hua Wu. LLMs are single-threaded reasoners: Demystifying the working mechanism of soft thinking. In *The Fourteenth International Conference on Learning Representations*, 2026b.

Xin Xu, Tong Yu, Xiang Chen, Haoliang Wang, Julian McAuley, and Saayan Mitra. ThinkRouter: Efficient reasoning via routing thinking between latent and discrete spaces. *arXiv preprint arXiv:2602.11683*, 2026.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. SoftCoT: Soft chain-of-thought for efficient reasoning with LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23336–23351, 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, Yue Yu, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *Advances in Neural Information Processing Systems*, 2025.

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

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, and Xin Wang. Soft thinking: Unlocking the reasoning potential of LLMs in continuous concept space. In *Advances in Neural Information Processing Systems*, 2025.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

Zhi Zheng and Wee Sun Lee. Soft-GRPO: Surpassing discrete-token LLM reinforcement learning via Gumbel-reparameterized soft-thinking policy optimization. *arXiv preprint arXiv:2511.06411*, 2025.

Yuyan Zhou, Jiarui Yu, Hande Dong, Zhezheng Hao, Hong Wang, Jianqing Zhang, and Qiang Lin. LEPO: Latent reasoning policy optimization for large language models. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 14416–14427, 2026.

Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart J. Russell, and Yuandong Tian. Reasoning by superposition: A theoretical perspective on chain of continuous thought. In *Advances in Neural Information Processing Systems*, 2025.

A Implementation Details

A.1 Sparse RL Relaxation with GRPO

We provide two implementations of Sparse RL Relaxation on the GRPO framework. The first implements the full relaxation by differentiating through the complete soft-CoT trajectory, while the second uses Proxy Gradient Grafting for efficient training with decoupled rollout and optimization. Both implementations use the Gumbel–entmax soft-token construction in Eqn. (15) and differ only in how gradients through the soft-token sequence are computed.

For a rollout group with rewards $\{R_g\}_{g=1}^{G}$, the group-relative advantage is

$$\widehat{A}_g = \frac{R_g - \mu_R}{\sigma_R + \varepsilon_A}, \qquad \mu_R = \frac{1}{G}\sum_{h=1}^{G} R_h, \qquad \sigma_R = \sqrt{\frac{1}{G}\sum_{h=1}^{G}(R_h - \mu_R)^2}. \tag{16}$$

Given token-level importance ratios $\{r_{g,j}\}$, define the shared clipped GRPO loss

$$\mathcal{L}_{\mathrm{GRPO}}\left(\{r_{g,j}\}, \{\widehat{A}_g\}\right) := -\frac{1}{G}\sum_{g=1}^{G}\frac{1}{n_g}\sum_{j=1}^{n_g}\min\left\{r_{g,j}\widehat{A}_g,\ \mathrm{clip}\left(r_{g,j}, 1-\varepsilon_{\mathrm{clip}}, 1+\varepsilon_{\mathrm{clip}}\right)\widehat{A}_g\right\}. \tag{17}$$

We instantiate Sparse RL Relaxation on a common GRPO backbone. The two implementations below differ only in the construction of the current-policy soft-token gradient and the resulting importance ratio.

Sparse RL Relaxation with GRPO Algorithm 1 differentiates through the complete current-policy soft-CoT trajectory.

Efficient Sparse RL Relaxation with Proxy Gradient Grafting Algorithm 2 replaces the full recursive gradient with Proxy Gradient Grafting.

For clarity, Algorithms 1 and 2 present the full-vocabulary formulation and omit optimizer-specific and shared GRPO regularization details. In practice, this may result in heavy memory burden. Thus, recent methods (Zheng & Lee, 2025; Zhou et al., 2026; Butt et al., 2026) typically select only top-k candidate logits for soft token construction. More details of hyperparameter of the implementation can be found in the next subsection.

A.2 Implementation Configuration and Hyperparameters

To ensure a controlled comparison, we use a unified training configuration across all methods and keep all shared components fixed whenever applicable. Our setup closely follows SofT-GRPO (Zheng & Lee, 2025), including the GRPO backbone, rollout configuration, optimization

Algorithm 1 Full Sparse RL Relaxation with GRPO

Require: prompt batch $\mathcal{B}$, policy $\pi_{\boldsymbol{\theta}}$, group size G, α, τ_g, learning rate η

$\boldsymbol{\theta}_{\text{old}} \leftarrow \text{copy}(\boldsymbol{\theta})$
$\mathcal{L}_{\text{full}} \leftarrow 0$
for $\boldsymbol{x} \in \mathcal{B}$ **do**
  for $g = 1, \ldots, G$ **do**
    for $i = 1, \ldots, m$ **do**
      $\boldsymbol{\epsilon}_i^{(g)} \sim \text{Gumbel}(0,1)^{|\mathcal{V}|}$
      $\boldsymbol{z}_i^{\text{old},(g)} \leftarrow f_{\boldsymbol{\theta}_{\text{old}}}\left(\boldsymbol{x}, \tilde{\boldsymbol{e}}_{<i}^{\text{old},(g)}\right)$
      $\tilde{\boldsymbol{p}}_i^{\text{old},(g)} \leftarrow \text{entmax}_\alpha\left(\frac{\boldsymbol{z}_i^{\text{old},(g)} + \boldsymbol{\epsilon}_i^{(g)}}{\tau_g}\right)$
      $\tilde{\boldsymbol{e}}_i^{\text{old},(g)} \leftarrow \sum_{k=1}^{|\mathcal{V}|} \tilde{p}_{i,k}^{\text{old},(g)} \boldsymbol{e}_k$
    end for
    $\boldsymbol{y}_{1:n_g}^{(g)} \sim \pi_{\boldsymbol{\theta}_{\text{old}}}\left(\cdot \mid \boldsymbol{x}, \tilde{\boldsymbol{e}}_{1:m}^{\text{old},(g)}\right)$
    $R_g \leftarrow R\left(\boldsymbol{x}, \boldsymbol{y}_{1:n_g}^{(g)}\right)$
    $\ell_{g,j}^{\text{old}} \leftarrow \log \pi_{\boldsymbol{\theta}_{\text{old}}}\left(y_j^{(g)} \mid \boldsymbol{x}, \tilde{\boldsymbol{e}}_{1:m}^{\text{old},(g)}, \boldsymbol{y}_{<j}^{(g)}\right), \quad j = 1, \ldots, n_g$
  end for
  $\widehat{A}_g \leftarrow$ Eqn. (16), $\quad g = 1, \ldots, G$
  for $g = 1, \ldots, G$ **do**
    for $i = 1, \ldots, m$ **do**
      $\boldsymbol{z}_i^{\boldsymbol{\theta},(g)} \leftarrow f_{\boldsymbol{\theta}}\left(\boldsymbol{x}, \tilde{\boldsymbol{e}}_{<i}^{\boldsymbol{\theta},(g)}\right)$
      $\tilde{\boldsymbol{p}}_i^{\boldsymbol{\theta},(g)} \leftarrow \text{entmax}_\alpha\left(\frac{\boldsymbol{z}_i^{\boldsymbol{\theta},(g)} + \boldsymbol{\epsilon}_i^{(g)}}{\tau_g}\right)$
      $\tilde{\boldsymbol{e}}_i^{\boldsymbol{\theta},(g)} \leftarrow \sum_{k=1}^{|\mathcal{V}|} \tilde{p}_{i,k}^{\boldsymbol{\theta},(g)} \boldsymbol{e}_k$
    end for
    $\ell_{g,j}^{\boldsymbol{\theta}} \leftarrow \log \pi_{\boldsymbol{\theta}}\left(y_j^{(g)} \mid \boldsymbol{x}, \tilde{\boldsymbol{e}}_{1:m}^{\boldsymbol{\theta},(g)}, \boldsymbol{y}_{<j}^{(g)}\right), \quad j = 1, \ldots, n_g$
    $r_{g,j}^{\text{full}} \leftarrow \exp\left(\ell_{g,j}^{\boldsymbol{\theta}} - \ell_{g,j}^{\text{old}}\right), \quad j = 1, \ldots, n_g$
  end for
  $\mathcal{L}_{\text{full}} \leftarrow \mathcal{L}_{\text{full}} + \frac{1}{|\mathcal{B}|}\mathcal{L}_{\text{GRPO}}\left(\{r_{g,j}^{\text{full}}\}, \{\widehat{A}_g\}\right)$
end for
$\boldsymbol{\theta} \leftarrow \text{OptimizerStep}\left(\boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}}\mathcal{L}_{\text{full}}; \eta\right)$

settings, and training budget. Competing methods differ only in hyperparameters required by their respective soft-CoT mechanisms. Full configurations are reported in Table 3.

For SpaRLax, we set the entmax parameter to $\alpha = 1.5$. This is a commonly used choice in prior entmax-related papers (Peters et al., 2019; Correia et al., 2019; Tezekbayev et al., 2022), providing an intermediate level of sparsity between softmax ($\alpha = 1$) and sparsemax ($\alpha = 2$). Importantly, $\alpha = 1.5$ also admits a specialized exact algorithm, making it particularly convenient to implement and with lower latency compared with general values of α, which typically require iterative computation (Peters et al., 2019; Tezekbayev et al., 2022).

In addition, we use a candidate-set size of $k = 32$ for SpaRLax for lower memory usage; future work can consider implementations with fused kernels (Gonçalves et al., 2025). Unlike softmax, entmax can adaptively reduce its effective support by assigning exactly zero probability to unnecessary candidates. This allows us to expose the model to a larger candidate pool without forcing all retained candidates to participate in the soft-token mixture. As shown in §4.4, softmax becomes increasingly sensitive to large k (which may be the reason why other methods (Zheng & Lee, 2025; Butt et al., 2026) select a modest k in their work), whereas entmax remains substantially more robust and can

Algorithm 2 Efficient Sparse RL Relaxation with Proxy Gradient Grafting

Require: prompt batch $\mathcal{B}$, policy $\pi_{\boldsymbol{\theta}}$, group size G, α, τ_g, learning rate η

$\boldsymbol{\theta}_{\text{old}} \leftarrow \text{copy}(\boldsymbol{\theta})$
$\mathcal{L}_{\text{proxy}} \leftarrow 0$
for $\boldsymbol{x} \in \mathcal{B}$ **do**
  for $g = 1, \dots, G$ **do**
    for $i = 1, \dots, m$ **do**
      $\boldsymbol{\epsilon}_i^{(g)} \sim \text{Gumbel}(0,1)^{|\mathcal{V}|}$
      $\boldsymbol{z}_i^{\text{roll},(g)} \leftarrow f_{\boldsymbol{\theta}_{\text{old}}}\left(\boldsymbol{x}, \tilde{\boldsymbol{e}}_{<i}^{\text{roll},(g)}\right)$
      $\tilde{\boldsymbol{p}}_i^{\text{roll},(g)} \leftarrow \text{entmax}_\alpha\left(\frac{\boldsymbol{z}_i^{\text{roll},(g)} + \boldsymbol{\epsilon}_i^{(g)}}{\tau_g}\right)$
      $\tilde{\boldsymbol{e}}_i^{\text{roll},(g)} \leftarrow \sum_{k=1}^{|\mathcal{V}|} \tilde{p}_{i,k}^{\text{roll},(g)} \boldsymbol{e}_k$
    end for
    $\boldsymbol{y}_{1:n_g}^{(g)} \sim \pi_{\boldsymbol{\theta}_{\text{old}}}\left(\cdot \mid \boldsymbol{x}, \tilde{\boldsymbol{e}}_{1:m}^{\text{roll},(g)}\right)$
    $R_g \leftarrow R\left(\boldsymbol{x}, \boldsymbol{y}_{1:n_g}^{(g)}\right)$
    $\ell_{g,j}^{\text{old}} \leftarrow \log \pi_{\boldsymbol{\theta}_{\text{old}}}\left(y_j^{(g)} \mid \boldsymbol{x}, \tilde{\boldsymbol{e}}_{1:m}^{\text{roll},(g)}, \boldsymbol{y}_{<j}^{(g)}\right), \quad j = 1, \dots, n_g$
  end for
  $\widehat{A}_g \leftarrow$ Eqn. (16), $\quad g = 1, \dots, G$
  for $g = 1, \dots, G$ **do**
    for $i = 1, \dots, m$ **do**
      $\boldsymbol{z}_i^{\text{proxy},(g)} \leftarrow f_{\boldsymbol{\theta}}\left(\boldsymbol{x}, \text{sg}\left(\tilde{\boldsymbol{e}}_{<i}^{\text{roll},(g)}\right)\right)$
      $\tilde{\boldsymbol{p}}_i^{\text{proxy},(g)} \leftarrow \text{entmax}_\alpha\left(\frac{\boldsymbol{z}_i^{\text{proxy},(g)} + \boldsymbol{\epsilon}_i^{(g)}}{\tau_g}\right)$
      $\bar{\boldsymbol{p}}_i^{(g)} \leftarrow \text{sg}\left(\tilde{\boldsymbol{p}}_i^{\text{roll},(g)}\right) + \tilde{\boldsymbol{p}}_i^{\text{proxy},(g)} - \text{sg}\left(\tilde{\boldsymbol{p}}_i^{\text{proxy},(g)}\right)$
      $\bar{\boldsymbol{e}}_i^{(g)} \leftarrow \sum_{k=1}^{|\mathcal{V}|} \bar{p}_{i,k}^{(g)} \boldsymbol{e}_k$
    end for
    $\bar{\ell}_{g,j}^{\boldsymbol{\theta}} \leftarrow \log \pi_{\boldsymbol{\theta}}\left(y_j^{(g)} \mid \boldsymbol{x}, \bar{\boldsymbol{e}}_{1:m}^{(g)}, \boldsymbol{y}_{<j}^{(g)}\right), \quad j = 1, \dots, n_g$
    $r_{g,j}^{\text{proxy}} \leftarrow \exp\left(\bar{\ell}_{g,j}^{\boldsymbol{\theta}} - \ell_{g,j}^{\text{old}}\right), \quad j = 1, \dots, n_g$
  end for
  $\mathcal{L}_{\text{proxy}} \leftarrow \mathcal{L}_{\text{proxy}} + \frac{1}{|\mathcal{B}|} \mathcal{L}_{\text{GRPO}}\left(\{r_{g,j}^{\text{proxy}}\}, \{\widehat{A}_g\}\right)$
end for
$\boldsymbol{\theta} \leftarrow \text{OptimizerStep}\left(\boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{proxy}}; \eta\right)$

benefit from a larger candidate budget. We therefore use $k = 32$ for SpaRLax while retaining the smaller candidate sets used by the softmax-based baselines.

Apart from these method-specific choices, all shared training configurations are held fixed across methods.

A.3 Stopping Criterion for Soft Reasoning

We follow the stopping criterion used in prior soft-thinking methods (Zhang et al., 2025; Zheng & Lee, 2025) to determine when to transition from soft reasoning back to standard discrete decoding. At each reasoning step i, after constructing the soft-token distribution $\tilde{\boldsymbol{p}}_i$, we identify its most probable vocabulary token as

$$t_i^\star = \arg\max_{k \in \mathcal{V}} \tilde{p}_{i,k}. \tag{18}$$

Parameter	Discrete GRPO (Guo et al., 2025)	SofT-GRPO (Zheng & Lee, 2025)	SpaRLax (ours)	STHT (Butt et al., 2026)	LEPO (Zhou et al., 2026)
Batch configuration					
Fine-tuning			Full-parameter fine-tuning		
Maximum prompt length	512	512	512	512	512
Maximum response length	8,192	8,192	8,192	8,192	8,192
Filter overlong prompts	Enabled	Enabled	Enabled	Enabled	Enabled
Prompt batch size	32	32	32	32	16
Responses per prompt (group size)	8	8	8	8	8
Global rollout batch (trajectories)	256	256	256	256	128
Effective PPO minibatch (trajectories)	256	256	256	256	128
Microbatch per GPU (trajectories)	1	1	1	1	1
Gradient accumulation steps	64	64	64	64	64
Optimization and regularization					
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW
Learning rate	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}
Adam (β_1, β_2)			$(0.9, 0.999)$		
Weight decay	0.01	0.01	0.01	0.01	0.01
LR schedule / warmup steps			Constant / 0		
PPO epochs per rollout	1	1	1	1	2
PPO clipping interval			$[0.8, 1.2]$		
Dual-clip coefficient	3.0	3.0	3.0	3.0	3.0
Gradient-norm clipping	1.0	1.0	1.0	1.0	0.5
Group-standard-deviation normalization	Enabled	Enabled	Enabled	Enabled	Enabled
Entropy bonus coefficient	0	0	0	0	0
Actor KL coefficient	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}
LEPO KL estimator	–	–	–	–	1
Max training steps (outer iterations)	$3,000$	$3,000$	$3,000$	$3,000$	$3,000$
Training-time generation and method configuration					
Soft thinking	Disabled	Enabled	Enabled	Enabled	Enabled
Soft-token transform	–	Softmax	Entmax	Softmax	Softmax
Soft candidate count	–	5	32	5	30
Entmax α	–	–	1.5	–	–
Relaxation temperature	–	0.1	1.0	1.0	0.5
Gumbel-noise multiplier	–	1.0	1.0	1.0	1.0
Categorical sampling temperature	1.0	1.0	1.0	1.0	1.0
Categorical top-k / top-p	5 / 0.95	5 / 0.95	5 / 0.95	5 / 0.95	5 / 0.95
Direct PG positions	All	All	answer phase	All	answer phase
LEPO maximum latent steps	–	–	–	–	64
STHT density scale σ	–	–	–	$0.33\,\mathrm{RMS}(E)$	–
STHT log-score range	–	–	–	$[-50, 0]$	–
Precision and parallelism					
Forward precision	BF16	BF16	BF16	BF16	BF16
GPUs			4 NVIDIA H100 GPUs		
Actor sharding	FSDP1	FSDP1	FSDP1	FSDP1	FSDP1
Sequence / rollout tensor parallel size	1 / 1	1 / 1	1 / 1	1 / 1	1 / 1
Gradient checkpointing	Enabled	Enabled	Enabled	Enabled	Enabled
Parameter / optimizer offload	Enabled	Enabled	Enabled	Enabled	Enabled

Table 3: Training hyperparameters. The first four columns retain the supplied GSM8K settings; LEPO reports launcher defaults, which use DeepScaleR training data. Batch sizes distinguish prompt groups from generated trajectories.

Soft reasoning continues while $t_i^\star \neq \langle/\mathrm{think}\rangle$. Once the end-of-thinking token becomes the most probable token, i.e., $t_i^\star = \langle/\mathrm{think}\rangle$, we terminate the soft reasoning phase and switch to standard autoregressive decoding for the final answer. All subsequent answer tokens are discrete and sampled from the model's vocabulary in the usual manner.

This stopping rule is applied during both rollout and inference and does not introduce an additional learned stopping module or a method-specific stopping criterion.

B LIMITATIONS AND FUTURE WORK

Superposition utilization. Our analysis in §4.3 provides evidence that the non-top-1 components of a soft token have a measurable effect on downstream model computation, as our SpaRLax exhibits a stronger effect than the competing methods. This intervention provides an indirect measure of superposition utilization, as it does not fully show that multiple trajectories are truly used during exploration. In future work, we are interested in exploring how such superposition can help search tasks, such as the Traveling Salesperson Problem.

Scale and generalization. Although we evaluate the trained models across nine benchmarks spanning mathematical, knowledge-intensive, and code reasoning, our post-training experiments are conducted with one base model and one RL training corpus. However, we cannot afford more experiments using assorted LLMs and corpora. Nevertheless, we have carefully designed the experiments by isolating the effect of the proposed learning mechanism from the underlying model, training data, and RL algorithm, as the configurations are fixed across all competing methods. Moreover, the consistent improvements observed across multiple reasoning domains suggest that the benefit is not confined to a specific setup. We leave the scaling of this study to large technology companies.