

Soft Thinking: Unlocking the Reasoning Potential of LLMs in Continuous Concept Space

Zhen Zhang^{1*} Xuehai He^{2*} Weixiang Yan¹
Ao Shen⁴ Chenyang Zhao^{3,5} Xin Eric Wang¹

¹University of California, Santa Barbara, ²University of California, Santa Cruz

³University of California, Los Angeles, ⁴Purdue University, ⁵LMSYS Org

zhen_zhang@ucsb.edu, ericxwang@ucsb.edu

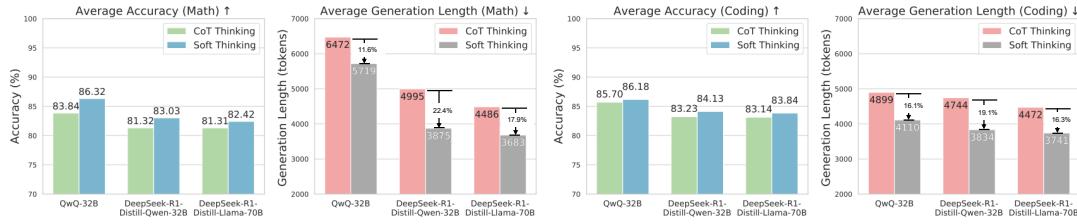


Figure 1: *Soft Thinking* vs. Chain-of-Thought thinking on mathematical and coding datasets. *Soft Thinking* consistently improves both accuracy (with improvements of up to **2.48%** on pass@1 accuracy) and generation efficiency (achieving up to **22.4%** reduction in generation length) across both tasks, **without any training**.

Abstract

Human cognition typically involves thinking through abstract, fluid concepts rather than strictly using discrete linguistic tokens. Current reasoning models, however, are constrained to reasoning within the boundaries of human language, processing discrete token embeddings that represent fixed points in the semantic space. This discrete constraint restricts the expressive power and upper potential of such reasoning models, often causing incomplete exploration of reasoning paths, as standard Chain-of-Thought (CoT) methods rely on sampling one token per step. In this work, we introduce *Soft Thinking*, a training-free method that emulates human-like “soft” reasoning by generating soft, abstract *concept tokens* in a continuous concept space. These *concept tokens* are created by the probability-weighted mixture of token embeddings, which form the continuous concept space, enabling smooth transitions and richer representations that transcend traditional discrete boundaries. In essence, each generated *concept token* encapsulates multiple meanings from related discrete tokens, implicitly exploring various reasoning paths to converge effectively toward the correct answer. Empirical evaluations on diverse mathematical and coding benchmarks consistently demonstrate the effectiveness and efficiency of *Soft Thinking*, improving pass@1 accuracy by up to 2.48 points while simultaneously reducing token usage by up to 22.4% compared to standard CoT. Qualitative analysis further reveals that *Soft Thinking* outputs remain highly interpretable and readable, highlighting the potential of *Soft Thinking* to break the inherent bottleneck of discrete language-based reasoning. Code is available [here](#).

*Equal contribution

The limits of my language mean the limits of my world.

—Ludwig Wittgenstein

1 Introduction

Large Language Models (LLMs) have achieved impressive results across a wide range of complex reasoning tasks. A key technique contributing to this success is Chain-of-Thought (CoT) reasoning [1–4], which enables models to solve problems step-by-step by generating intermediate reasoning steps in natural language. Despite its effectiveness, standard CoT reasoning restricts the model’s outputs to sequences of discrete, predefined tokens, inherently bounded by the expressive limits of human language. Confining LLM reasoning to the discrete natural language tokens may fundamentally restrict their potential to represent and manipulate abstract concepts. Moreover, neuroscientific evidence shows that the human brain represents and stores information at the level of abstract concepts, not merely words, and that reasoning is based on non-verbal conceptual processing independent of the language network [5–8].

Another fundamental limitation of standard CoT reasoning is its inherently unidirectional and sequential nature: at each step, the model samples a single token, committing to one specific branch of the reasoning path. In tasks with high uncertainty or multiple plausible trajectories, this approach can easily lead the model down an incorrect path, resulting in suboptimal answers or wasted tokens on the wrong path, thus reducing both performance and token efficiency [9, 10]. In contrast, humans do not rely solely on sequentially producing explicit linguistic tokens. Instead, they can simultaneously consider multiple possibilities, integrate abstract concepts, and only later verbalize their thoughts. This allows for more flexible, parallel, and comprehensive reasoning, enabling humans to navigate complex problems more effectively.

In this work, we propose a new perspective: instead of constraining LLMs to reason within the discrete, sequential space of language tokens, we aim to enable LLMs to reason with soft, abstract concepts, which encompass more general and fine-grained semantics and retain information about multiple possible paths. To achieve this, we introduce *Soft Thinking*, a training-free method that unlocks the reasoning potential of LLMs in a continuous concept space. Specifically, *Soft Thinking* replaces the discrete token selection in standard CoT with probabilistic soft aggregation over the entire vocabulary, which we refer to as a *concept token*. This retains the original distribution of the next step. At each step, we construct a new embedding from a concept token by probability-weighting all token embeddings, which form the continuous concept token. This approach allows the model to represent and process abstract concepts, endowing each output token with more nuanced and fine-grained semantics, and enabling the processing of multiple paths conceptually.

Unlike standard CoT that forces the model to commit to a single next token at each step by collapsing the probability distribution, our method naturally preserves a “superposition” which retains the entire information in each step. As a result, we introduce a *Cold Stop* mechanism to further boost efficiency and address the challenge of generation collapse (e.g., repetition) caused by out-of-distribution (OOD) [11] inputs, where certain *concept tokens* may be unseen during training. To be specific, *Cold Stop* monitors the entropy of the model’s output distribution at each step and terminates the reasoning process early when the model demonstrates high confidence (i.e., low entropy) over several consecutive steps. This mechanism prevents unnecessary computation and mitigates the risk of model collapse when dealing with OOD inputs, ensuring more robust and efficient reasoning.

Soft Thinking offers two major advances. First, by operating in the continuous concept space formed as a convex combination of all token embeddings, the model can capture and manipulate abstract concepts and detailed semantic information; Second, because each concept token keeps a probability distribution from all possible next tokens, the model can implicitly and efficiently explore multiple reasoning paths in parallel, rather than being limited to a single trajectory. Therefore, *Soft Thinking* not only improves the comprehensiveness of reasoning but also accelerates convergence toward correct answers.

Empirical evaluations conducted on mathematical and coding benchmarks using mainstream LLM architectures, including Llama [12] and Qwen [13] with large model sizes 32B and 70B parameters, consistently demonstrate the effectiveness and efficiency of *Soft Thinking*. The method improves

pass@1 accuracy by up to 2.48 points while simultaneously reducing token usage by up to 22.4% compared to standard CoT. Furthermore, qualitative assessments reveal that intermediate reasoning steps generated by *Soft Thinking* are highly readable, interpretable, and informative. Overall, *Soft Thinking* presents an alternative reasoning paradigm that breaks the bottleneck of discrete token-based reasoning.

2 Related Work

Chain-of-Thought (CoT) Reasoning. CoT reasoning enhances the multi-step reasoning capabilities of large language models by introducing explicit intermediate steps. Existing approaches primarily include prompt-based learning methods [1–3], supervised fine-tuning [14, 15], and reinforcement learning optimization [16–19]. Moreover, according to inference-time scaling laws [20], model performance continues to improve as the length of reasoning chains increases. However, as the chain grows longer, the computational cost also rises, making efficiency a growing concern. To address this challenge, we propose to shift CoT reasoning from the discrete natural language token to a continuous concept space, which is formed as a convex combination of all token embeddings. In this space, the model can select and integrate multiple potential reasoning trajectories at the token level.

Continuous Space Reasoning. [21] constructed datasets for two-hop reasoning tasks and showed that intermediate reasoning variables could be decoded from hidden representations. Building on this, [22] introduced interventions on hidden states to manipulate reasoning outcomes. Parallel latent reasoning paths have also been observed [23]. [24] introduced latent planning steps: proposed predicting discrete planning tokens before generating reasoning steps. [25] proposes to do reasoning at an abstract language level beyond tokens and explores an explicit hierarchical structure. [26] proposes extracting text embeddings from the last token of LLMs fine-tuned with instructions on contrastive data. COCONUT [27] uses the last hidden state of the model’s final layer as the next-step embedding. However, this method still face critical challenges. In language models with fewer than 7B parameters, the input embedding layer and the output language model head are typically weight-tied, enabling continuous-space reasoning by aligning the input and output spaces after extensive training. In contrast, for models with more than 7 billion parameters, these components are typically decoupled, meaning that the hidden states and input embeddings reside in different spaces. Directly using hidden states as input embeddings leads to significant representational mismatch, which is difficult to bridge even with extensive retraining. Such retraining often leads to overfitting, catastrophic forgetting, or ineffective performance in practice [28]. To address these limitations, we propose a training-free approach that utilizes the distribution over the vocabulary at each step as a bridge. This method effectively aligns the hidden state output space with the input embedding space, enabling seamless representation alignment during continuous-space reasoning.

3 Methodology

In this section, we introduce *Soft Thinking*, a method that generalizes standard Chain-of-Thought (CoT) reasoning by replacing discrete one-hot tokens with *concept tokens* and keeping the entire original probability distribution. As shown in Figure 2, the new embeddings are computed using probability-weighted interpolation across all embeddings based on the preceding *concept token*, facilitating reasoning within a continuous concept space. Furthermore, we propose the *Cold Stop* mechanism, which halts intermediate reasoning steps when overconfident, enhancing inference efficiency and preventing generation collapse.

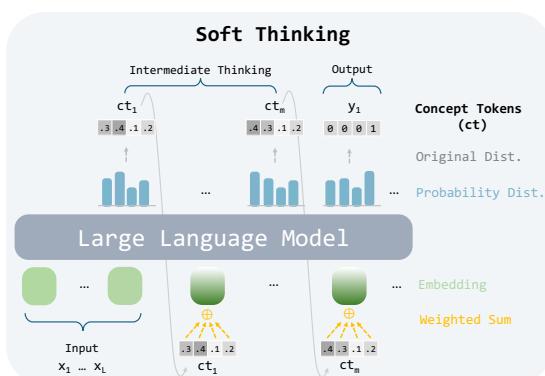


Figure 2: *Soft Thinking* replaces discrete tokens with soft, abstract *concept tokens*, enabling reasoning in continuous concept space.

3.1 Preliminary: Standard Chain-of-Thought Decoding

Let V be the vocabulary of size $|V|$, and let $E \in \mathbb{R}^{|V| \times d}$ be the token embedding matrix. For any token index k , we denote its embedding by $e(k) = E[k] \in \mathbb{R}^d$. Given an input context: $x_{1:L} = (x_1, x_2, \dots, x_L)$, the model first generates an intermediate reasoning trace (the Chain-of-Thought) of length m : $t_{1:m} = (t_1, t_2, \dots, t_m)$, and then produces a final answer of length n : $y_{1:n} = (y_1, y_2, \dots, y_n)$.

At each intermediate thinking step i , the LLM consumes the embeddings of all previously generated tokens and the input, and the next token is then sampled discretely:

$$t_i \sim p_i = \text{LLM}(e(x_{1:i}), e(t_{1:i-1})) \in \Delta^{|V|-1}, \quad (1)$$

Here, $\Delta^{|V|-1}$ denotes the $(|V| - 1)$ -dimensional probability simplex, representing the set of all valid probability distributions over the vocabulary. Decoding continues until the special end-of-thinking token $\langle / \text{think} \rangle$ is generated, i.e. $t_m = \text{encode}(\langle / \text{think} \rangle)$.

After reasoning, the model switches to answer mode. For each answer position j , it computes the probability q_j and then sample one token y_j :

$$y_j \sim q_j = \text{LLM}(e(x_{1:i}), e(t_{1:m}), e(y_{1:j-1})) \in \Delta^{|V|-1}, \quad (2)$$

All tokens in both stages are drawn from their respective discrete distributions, committing to a discrete token id at each step. In the next section, we introduce *Soft Thinking*, which replaces this discrete sampling with continuous *concept tokens*, preserving the full distributional information throughout multi-step reasoning.

3.2 Soft Thinking: Reasoning in a Continuous Concept Space

Definition 1 (Concept Token). At any intermediate thinking step, let $p \in \Delta^{|V|-1}$ be the LLM-produced probability distribution over the vocabulary. We call this probability vector a *concept token*, denoted by

$$ct := p. \quad (3)$$

Unlike a traditional step that collapses the distribution to a single token id, the concept token preserves the full distribution of every possible next step.

Definition 2 (Continuous Concept Space). Let $E \in \mathbb{R}^{|V| \times d}$ be the embedding matrix and $e(k) = E[k]$ the embedding of the k -th vocabulary item. The *continuous concept space* is the convex combination of all embedding vectors.

$$\mathcal{C} = \left\{ \sum_{k=1}^{|V|} \alpha_k e(k) : \alpha \in \Delta^{|V|-1} \right\} \subset \mathbb{R}^d, \quad (4)$$

i.e. the set of all probability-weighted mixtures of token embeddings. Note that this is different from the usual semantic space, which is modeled as a d -dimensional real vector space.

Reasoning Process. *Soft Thinking* only replaces the intermediate thinking step of the standard CoT. At each step of soft thinking, the model generates a *concept token* defined by Definition 1. Then, in the next step, the *concept token* ct is injected back into the LLM by the embedding of *concept token*:

$$\tilde{e}_{\text{next}} = \sum_{k=1}^{|V|} ct[k] e(k) = \sum_{k=1}^{|V|} p[k] e(k) \in \mathcal{C}. \quad (5)$$

When the most probable token for a certain concept token is the end-of-thinking, the intermediate reasoning process stops, and the model switches to generating the output. All output stage tokens y_j are sampled in the usual discrete manner; only the intermediate thinking phase flows through the continuous concept space defined above.

Why Soft Thinking Helps. Using *concept tokens* allows the model to avoid making hard decisions too early. Instead of selecting a single token at each step, the model keeps the full probability distribution over vocabulary. This gives it the flexibility to explore different reasoning paths, especially when it's unsure. By working in this *continuous concept space*, the model can represent more abstract concepts that don't map cleanly to a single word. These abstract concepts can later evolve into more concrete thoughts as reasoning continues (see Figure 4). This flexibility helps the model think more clearly, avoid early mistakes, and better handle complex multi-step problems.

Cold Stop While *concept tokens* enable more abstract reasoning, feeding in continuous *concept tokens* during inference places the model in an out-of-distribution (OOD) regime. This can lead to model collapse if the reasoning process continues for too long without correction. To mitigate this, we propose a *Cold Stop* mechanism that dynamically stops intermediate reasoning when the model becomes overconfident. At each step, we compute the entropy of the *concept token* :

$$H(p) = - \sum_{k=1}^{|V|} p[k] \log p[k]. \quad (6)$$

Since *Soft Thinking* preserves the entire probability distribution at each step, the entropy serves as a natural signal for uncertainty, which is often used in LLMs to evaluate the quality of generation [29]. Low entropy, typically represents “cold” in physics, indicates that the model is confident in its prediction [30], and thus can conclude soon. Given an entropy threshold τ and a required number of consecutive confident steps k , we apply the following rule:

- If $H(p) < \tau$, increment a low-entropy step counter; otherwise, reset the counter.
- When the counter reaches k , we insert an end-of-thinking token $\langle /think \rangle$ to conclude reasoning and begin final answer generation.

This strategy avoids unnecessary computation and prevents the model collapse under OOD conditions, while preserving the benefits of soft thinking through an entropy-based confidence measure.

Complexity Analysis. *Soft Thinking* incurs only two lightweight additions to standard CoT. When computing each concept token embedding, we first apply a top- k top- p filter to the token distribution to remove low-probability noise, select top- n tokens with highest probability, renormalize, and then perform a single dense matrix–vector multiplication over the filtered subset, resulting in $O(n \cdot d)$ computational cost per reasoning step (where d the embedding dimension). Calculating the entropy for *Cold Stop* requires $O(|V|)$ time, but adds negligible overhead compared to a model forward pass.

3.3 Theoretical Analysis

In this section, we provide a theoretical analysis showing how *Soft Thinking* approximates the full path-summation of standard Chain-of-Thought (CoT) by iteratively constructing linear surrogate representations via *concept tokens*. We begin by rewriting the exact expansion of the marginal likelihood and then derive a sequence of linearization steps that culminate in the continuous *concept token* approximation.

Exact Path-Summation. Let $x = x_{1:l}$ denote the input context and $t_{1:m}$ be the first m intermediate reasoning tokens. The true probability of a final answer $y = y_{1:n}$ is obtained by marginalizing over all possible reasoning trajectories of length m :

$$p(y | x) = \sum_{t_1} p(t_1 | x) \left(\sum_{t_2} p(t_2 | x, t_1) \cdots \left(\sum_{t_m} p(t_m | x, t_{1:m-1}) p(y | x, t_{1:m}) \right) \right). \quad (7)$$

This expansion entails an exponential number of paths, since each summation runs over the full vocabulary V .

First-Order Linearization. Focusing on the outermost summation,

$$p(y | x) = \sum_{t_1} p(t_1 | x) p(y | x, t_1). \quad (8)$$

Viewing the sampled token t_1 as an one-hot vector $t_1 \in \{0, 1\}^{|V|}$ with $\sum_k t_{1,k} = 1$, its expectation under the multinomial distribution is the first *concept token*:

$$ct_1 = \mathbb{E}[t_1] = \sum_{t_1} p(t_1 | x) t_1 = p(\cdot | x) \in \Delta^{|V|-1}, \quad (9)$$

which aligns the *concept token* in Definition 1.

By a linear approximation of $p(y | x, \cdot)$ around the mean, we obtain

$$p(y | x) = \sum_{t_1} p(t_1 | x) p(y | x, t_1) \approx p(y | x, \sum_{t_1} p(t_1 | x) t_1) = p(y | x, ct_1). \quad (10)$$

Thus, the full outer summation is replaced by a single evaluation at the *concept token* ct_1 .

Recursive Approximation. We now apply the same linearization recursively. Given ct_1 , the conditional probability expands as

$$p(y | x, ct_1) = \sum_{t_2} p(t_2 | x, ct_1) p(y | x, ct_1, t_2) \approx p(y | x, ct_1, ct_2), \quad (11)$$

where $ct_2 = \sum_{t_2} p(t_2 | x, ct_1) t_2$.

Repeating this process for all m steps yields the continuous expansion:

$$p(y | x) \approx p(y | x, ct_1, ct_2, \dots, ct_m). \quad (12)$$

Comparison to Standard CoT. In contrast, discrete CoT replaces each summation $\sum_{t_j} p(t_j | \cdot)$ with sampling a single token, thereby discarding mass from all other paths. *Soft Thinking* preserves the full probability distribution at each step through *concept tokens*, collapsing the exponential path-summation in Eq. 7 into a single forward pass under a sequence of linear approximations.

4 Experiments & Results

4.1 Experiment Setup

Benchmarks. We conduct a comprehensive evaluation of our method on eight benchmark tasks, including Math500 [31], AIME 2024 [32], GSM8K [33], and GPQA-Diamond [34] in the mathematics domain, as well as HumanEval [35], MBPPMBPP [36], and LiveCodeBench [37] in the programming domain. Detailed descriptions of these benchmarks are provided in Appendix A.2.

Models. We select three widely used open-source LLMs: QwQ-32B [13], DeepSeek-R1-Distill-Qwen-32B [38], and DeepSeek-R1-Distill-Llama-70B [38]. This diverse selection is designed to demonstrate the effectiveness and generalizability of the *Soft Thinking* approach across different model scales (32B and 70B), model architectures (Qwen and LLaMA), and training paradigms (QwQ-32B is trained with reinforcement learning, while the DeepSeek models are trained via supervised distillation).

Baseline Methods. We evaluate the performance of *Soft Thinking* by comparing it with two representative baselines. These baselines include Standard CoT Thinking, which employs explicit step-by-step reasoning, and Standard Greedy CoT Thinking, which utilizes greedy decoding at each step of the reasoning process.

Metrics. We use the Pass@1 metric to evaluate the accuracy of the model’s generated answers. The formula for computing Pass@k is as follows:

$$\text{Pass}@k = 1 - \frac{\binom{n-c}{k}}{\binom{n}{k}} \quad (13)$$

where n is the total number of samples (e.g., 16), c is the number of correct samples, and k is the number of samples selected (we set $k = 1$). Therefore, $\text{Pass}@1 = \frac{c}{n}$. Besides, we evaluate the model’s reasoning efficiency by reporting the number of tokens generated specifically for correct solutions. These two metrics allow us to comprehensively evaluate the trade-off between computational cost and performance across different methods.

Implementation Details. We reuse the model’s existing embedding matrix without any extra parameters or layers as *concept tokens*, and the *Cold Stop* controller monitors decoder entropy and emits an end-of-thinking marker when triggered. No model weights update, architecture change, or additional training procedures, *Soft Thinking* can be plugged into the CoT pipeline of any LLM with minimal engineering effort. We implement our *Soft Thinking* on SGLang [39], enabling fast inference (see Appendix A.3 for implementation details). We evaluate our method on a server equipped with eight NVIDIA H100 80GB GPUs.

	Accuracy \uparrow					Generation Length \downarrow				
	MATH 500	AIME 2024	GSM8K	GPQA Diamond	Avg.	MATH 500	AIME 2024	GSM8K	GPQA Diamond	Avg.
QwQ-32B [13]										
CoT Thinking	97.66	76.88	96.67	64.17	83.84	4156	12080	1556	8095	6472
CoT Thinking (Greedy)	97.00	80.00	96.57	65.15	84.68 (\uparrow 0.84)	3827	11086	1536	7417	5967 (\downarrow 7.8%)
Soft Thinking	98.00	83.33	96.81	67.17	86.32 (\uparrow 2.48)	3644	10627	1391	7213	5719 (\downarrow 11.6%)
DeepSeek-R1-Distill-Qwen-32B [38]										
CoT Thinking	94.50	72.08	95.61	63.10	81.32	3543	9347	875	6218	4995
CoT Thinking (Greedy)	93.00	63.33	95.30	59.09	77.68 (\downarrow 3.64)	3651	8050	1048	8395	5286 (\uparrow 5.8%)
Soft Thinking	95.00	76.66	95.83	64.64	83.03 (\uparrow 1.71)	3373	6620	785	4722	3875 (\downarrow 22.4%)
DeepSeek-R1-Distill-Llama-70B [38]										
CoT Thinking	94.70	70.40	94.82	65.34	81.31	3141	8684	620	5500	4486
CoT Thinking (Greedy)	94.61	73.33	93.60	66.16	81.92 (\uparrow 0.61)	2877	9457	606	4443	4345 (\downarrow 3.1%)
Soft Thinking	94.80	73.33	94.90	66.66	82.42 (\uparrow 1.11)	3021	6644	597	4470	3683 (\downarrow 17.9%)

Table 1: Comparison of *Soft Thinking* and various baseline methods on accuracy and generation length across mathematical datasets. Best results are highlighted in **bold**.

	Accuracy \uparrow				Generation Length \downarrow			
	HumanEval	MBPP	LiveCodeBench	Avg.	HumanEval	MBPP	LiveCodeBench	Avg.
QwQ-32B [13]								
CoT Thinking	97.63	97.49	62.00	85.70	2557	2154	9986	4899
CoT Thinking (Greedy)	95.73	96.50	57.35	83.19 (\downarrow 2.51)	2396	2069	7034	3833 (\downarrow 21.8%)
Soft Thinking	98.17	97.66	62.72	86.18 (\uparrow 0.48)	2638	2157	7535	4110 (\downarrow 16.1%)
DeepSeek-R1-Distill-Qwen-32B [38]								
CoT Thinking	97.25	95.13	57.33	83.23	3095	2761	8376	4744
CoT Thinking (Greedy)	87.19	87.54	43.36	72.70 (\downarrow 10.53)	2294	1703	4702	2900 (\downarrow 38.9%)
Soft Thinking	97.56	95.33	59.50	84.13 (\uparrow 0.90)	2713	2534	6255	3834 (\downarrow 19.1%)
DeepSeek-R1-Distill-Llama-70B [38]								
CoT Thinking	97.71	94.77	56.94	83.14	2711	2386	8319	4472
CoT Thinking (Greedy)	92.07	91.82	48.02	77.30 (\downarrow 5.84)	2192	1979	5438	3203 (\downarrow 28.3%)
Soft Thinking	98.17	94.94	58.42	83.84 (\uparrow 0.70)	2498	2214	6512	3741 (\downarrow 16.3%)

Table 2: Comparison of *Soft Thinking* and various baseline methods on accuracy and generation length across two coding datasets. Best results are highlighted in **bold**.

4.2 Hyper-parameter Settings

For all experiments, the maximum generation length was set to 32, 768, the temperature to 0.6, top- k to 30, and top- p to 0.95, unless specified otherwise. The Standard CoT baseline was evaluated using 16 samples per problem to calculate Pass@1 accuracy, whereas the greedy CoT approach utilized a temperature of 0 with a single sample.

For *Soft Thinking*, the *concept token* was determined using the top- n tokens, where $n \in 5, 10, 15, 20, 30$, along with an entropy threshold τ chosen from 0.01, 0.05, 0.1, 0.2 and a length threshold k selected from 128, 256, 512, 1024. All other settings were kept consistent. We find that $n = 15$ yields the best performance for QwQ-32B [13], while $n = 10$ is optimal for DeepSeek-R1 models [38]. Results are reported based on the best-performing combinations of τ and k .

4.3 Results and Analysis

We present the quantitative evaluation results of *Soft Thinking* and other baseline methods on mathematical and coding datasets in Table 1 and Table 2, respectively.

Improved Pass@1 Accuracy. Our proposed *Soft Thinking* consistently enhances Pass@1 accuracy across all evaluated math and coding benchmarks, demonstrating its broad effectiveness and generalization ability. For instance, on mathematical reasoning tasks, the QwQ-32B model’s average Pass@1 improves from 83.84% (CoT Thinking) to 86.32% (*Soft Thinking*), representing a notable gain of 2.48% points. On the challenging AIME2024 dataset, the improvement reaches 6.45% points. Similarly, for DeepSeek-R1-Distill-Qwen-32B and DeepSeek-R1-Distill-Llama-70B, Pass@1 increases by 1.71% and 1.11% points, respectively. On coding benchmarks, *Soft Thinking* also achieves consistent improvements: QwQ-32B sees a 0.48-point increase in average Pass@1, while DeepSeek-R1-Distill-Qwen-32B and DeepSeek-R1-Distill-Llama-70B improve by 0.90% and 0.70%

Question: $43 * 34 = ?$

CoT Thinking (157 tokens):

First, I recognize that multiplying 43 by 34 can be simplified by breaking down the numbers into more manageable parts. I'll decompose 34 into 30 and 4. Next, I'll multiply 43 by 30. Since 30 is three times 10, I can first multiply 43 by 3 to get 129, and then add a zero to account for the 10, resulting in 1,290. Then, I'll multiply 43 by 4, which equals 172. Finally, I'll add the two results together: 1,290 plus 172 gives me the final answer of 1,462.

Soft Thinking (96 tokens):

To solve 43 multiplied by 34, I'll start by breaking down the numbers into their place values. First, I'll multiply 43 by 4, which gives me 172. Next, I'll multiply 43 by 30, resulting in 1,290. Finally, I'll add these two products together: 172 plus 1,290 equals 1,462.

Figure 3: A comparison between standard CoT and *Soft Thinking* on a multiplication problem. We select the token with the highest probability at each step of *Soft Thinking* for readability and interpretability. Full distribution is visualized in Figure 4. Red text denotes repetitive, useless words.

points, respectively. These results demonstrate that *Soft Thinking* provides robust accuracy gains across both math and code domains.

Token Efficiency. A key advantage of *Soft Thinking* is its significant reduction in generation length, leading to improved token efficiency. On mathematical reasoning benchmarks, *Soft Thinking* reduces token usage for QwQ-32B by 11.6%, DeepSeek-R1-Distill-Qwen-32B by 22.4%, and DeepSeek-R1-Distill-Llama-70B by 17.9% compared to standard CoT thinking. Similar trends are observed in coding tasks: for QwQ-32B, DeepSeek-R1-Distill-Qwen-32B, and DeepSeek-R1-Distill-Llama-70B, token usage is reduced by 16.1%, 19.1%, and 16.3%, respectively. This substantial reduction in token usage not only translates to lower computational and inference costs, but also indicates that the model can reach correct answers through more concise and efficient reasoning. Such token efficiency is particularly valuable for real-world applications, where cost, speed, and scalability are crucial.

Analysis. Our experimental results demonstrate that *Soft Thinking* achieves simultaneous improvements in both reasoning performance and token efficiency across a diverse set of mathematical and coding benchmarks. This dual gain highlights a key advantage of our approach: by leveraging *concept tokens* that encode richer semantic information at each reasoning step, the model is able to represent and process more abstract or composite ideas within a single token. As a result, fewer reasoning steps are required to reach the correct solution, directly translating to reduced token usage. Results also stand in stark contrast to the greedy decoding baseline, which, while also reducing token usage, suffers from substantial drops in accuracy, particularly on complex code generation tasks where the loss of diversity in reasoning paths leads to premature convergence on suboptimal solutions.

These findings suggest that the superior efficiency of *Soft Thinking* is not simply a result of more aggressive pruning or shortcircuiting, but rather reflects a fundamental enhancement in the model's reasoning process. By maintaining the full probability distribution over possible next tokens at each step, our method allows for a "soft" aggregation of multiple reasoning trajectories, effectively broadening the model's exploration space without incurring the combinatorial explosion of explicit enumeration. This enables the model to make more informed and confident decisions earlier in the reasoning chain, reducing unnecessary detours and redundant steps.

Overall, the results provide strong evidence that *Soft Thinking* breaks the traditional trade-off between performance and efficiency in large language model reasoning. Instead of sacrificing one for the other, our approach inherently boosts both, offering a more powerful and concise reasoning framework that is readily applicable to a wide range of tasks and model architectures.

4.4 Qualitative Results

Visualization of Shortened Examples. Figure 3 demonstrates the comparison between Standard CoT and *Soft Thinking*. We select the token with the highest probability at each step of *Soft Thinking* for visualization. It can be seen that *Soft Thinking* has high readability and interpretability. While both methods arrive at the correct answer (1,462), *Soft Thinking* produces a significantly more concise explanation (96 tokens vs. 157 tokens). This demonstrates *Soft Thinking*'s ability to preserve logical structure while improving token efficiency.



Figure 4: An example illustrating the probability distribution of our proposed *Soft Thinking* method. At each step, top- k token candidates and their probabilities are shown. Red boxes indicate the selected tokens that form the final generated sequence for readability and interpretability.

	Accuracy		Generation Length All		Generation Length Correct	
	AIME 2024 ↑	LiveCodeBench ↑	AIME 2024 ↓	LiveCodeBench ↓	AIME 2024 ↓	LiveCodeBench ↓
COCONUT-TF	0.0	0.0	32,768	32,768	—	—
Average Embedding	6.66	7.49	30,802	30,474	6,556	2,141
Soft Thinking w/o Cold Stop	73.33	56.98	12,991	13,705	9,457	6,877
Soft Thinking w/ Cold Stop	83.33	62.72	11,445	12,537	10,627	7,535

Table 3: Ablation study of Soft Thinking with QwQ-32B on the AIME 2024 and LiveCodeBench. COCONUT-TF represents training-free COCONUT [27].

Visualization of Embedding Weights. In Figure 4, we present the token probability distributions at each intermediate reasoning step of *Soft Thinking*. For demonstration, we highlight the top three tokens. During exploratory reasoning phases, such as steps 1 – 3, 13 – 14, and 18 – 20, the token distribution appears more uniform, reflecting the presence of multiple viable paths. In contrast, during precise calculations, the token distribution becomes nearly one-hot, indicating that textual elements facilitate path exploration, while numerical components handle exact computations. Notably, at steps 36-37, the model evaluates whether to multiply by 4 or 30, ultimately assigning a higher probability to 4. As a result, at step 42, the model selects multiplication by 4. This demonstrates how Soft Thinking integrates path exploration across consecutive *concept tokens*, thereby enhancing both reasoning flexibility and depth.

4.5 Ablation Study

To comprehensively assess the effectiveness of *Soft Thinking*, we conducted ablation studies focusing on different strategies for *concept token* and the impact of the *Cold Stop* mechanism.

Different strategies for *concept token*. Specifically, we compared (1) the training-free COCONUT approach [27], which directly feeds the previous hidden state as the next input embedding; (2) a simple average embedding strategy that takes the mean of the top- n token embeddings (we use 5 for ablation); and (3) our *Soft Thinking*, which computes a probability-weighted over embeddings. As shown in Table 3, we observe that the training-free COCONUT fails entirely, producing no correct answers and always reaching the maximum generation length. The average embedding method performs marginally better, yielding a small number of correct solutions but still suffering from extremely long output. In contrast, our *Soft Thinking* substantially improves both accuracy and token efficiency.

Impact of *Cold Stop*. We further analyze the *Cold Stop* by comparing *Soft Thinking* with and without it. Without *Cold Stop*, the model is prone to generation collapse, especially due to out-of-

distribution (OOD) issues because models have never been trained on *concept tokens*. It will begin to repeat until it hits the maximum generation length. This OOD-induced collapse significantly increases the average generation length across all problems. However, for problems that are solved correctly (typically those requiring shorter reasoning chains), the average length remains relatively low, as these cases are less likely to trigger collapse. When *Cold Stop* is activated, generation collapse is effectively mitigated, and the model avoids unnecessary exploration along overconfident paths, resulting in a significant reduction in average generation length for all problems. Interestingly, as *Cold Stop* allows the model to correctly solve more challenging problems that require longer reasoning chains, the average length for correct solutions increase. Nevertheless, as demonstrated in Tables 1 and 2, *Soft Thinking* with *Cold Stop* not only solves more problems than standard CoT but also achieves greater overall efficiency, confirming the effectiveness of our approach.

5 Conclusion

In this work, we present *Soft Thinking*, a novel, training-free framework that enables large language models to reason in a continuous concept space by leveraging probabilistically weighted *concept tokens* instead of traditional discrete tokens. By aggregating information across the entire vocabulary at each reasoning step, our method allows the model to implicitly explore multiple reasoning paths in parallel, leading to both higher accuracy and greater token efficiency. Experiments on mathematical and coding benchmarks demonstrate that *Soft Thinking* consistently improves pass@1 accuracy and reduces generation length, all without any additional training or architectural modifications. Qualitative analyses further show that the reasoning process remains interpretable and concise. Future work may explore integrating training-based approaches to adapt the *concept token*, with the goal of further improving performance and stability when faced with out-of-distribution inputs.

6 Acknowledgments

We would like to express our sincere gratitude to Yue Fan, Saaket Agashe, Liliang Ren, Hao Cheng, Baolin Peng, and Yiping Wang for their valuable feedback and constructive discussions. We thank Orby AI for generously providing the computational resources. Additionally, we appreciate the SGLang Team’s assistance during development.

References

- [1] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
- [2] Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*, 2022.
- [3] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.
- [4] Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. Towards revealing the mystery behind chain of thought: a theoretical perspective. *Advances in Neural Information Processing Systems*, 36:70757–70798, 2023.
- [5] R Quian Quiroga, Leila Reddy, Gabriel Kreiman, Christof Koch, and Itzhak Fried. Invariant visual representation by single neurons in the human brain. *Nature*, 435(7045):1102–1107, 2005.
- [6] Evelina Fedorenko and Rosemary Varley. Language and thought are not the same thing: evidence from neuroimaging and neurological patients. *Annals of the New York Academy of Sciences*, 1369(1):132–153, 2016.
- [7] AA Ivanova, Z Mineroff, V Zimmerer, N Kanwisher, R Varley, and E Fedorenko. The language network is recruited but not required for non-verbal semantic processing. *bioRxiv*, 696484, 2019.

[8] Yael Benn, Anna A Ivanova, Oliver Clark, Zachary Mineroff, Chloe Seikus, Jack Santos Silva, Rosemary Varley, and Evelina Fedorenko. The language network is not engaged in object categorization. *Cerebral Cortex*, 33(19):10380–10400, 2023.

[9] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

[10] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

[11] Lifan Yuan, Yangyi Chen, Ganqu Cui, Hongcheng Gao, Fangyuan Zou, Xingyi Cheng, Heng Ji, Zhiyuan Liu, and Maosong Sun. Revisiting out-of-distribution robustness in nlp: Benchmark, analysis, and llms evaluations. *arXiv preprint arXiv:2306.04618*, 2023.

[12] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[13] Qwen Team. Qwq-32b: Embracing the power of reinforcement learning, March 2025.

[14] Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhui Chen. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*, 2023.

[15] Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

[16] Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, 2024.

[17] Alex Havrilla, Yuqing Du, Sharath Chandra Raparthi, Christoforos Nalmpantis, Jane Dwivedi-Yu, Maksym Zhuravinskyi, Eric Hambro, Sainbayar Sukhbaatar, and Roberta Raileanu. Teaching large language models to reason with reinforcement learning. *arXiv preprint arXiv:2403.04642*, 2024.

[18] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, YK Li, Yu Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[19] Fangxu Yu, Lai Jiang, Haoqiang Kang, Shibo Hao, and Lianhui Qin. Flow of reasoning: Efficient training of llm policy with divergent thinking. *arXiv preprint arXiv:2406.05673*, 2024.

[20] Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. *arXiv preprint arXiv:2408.00724*, 2024.

[21] Sohee Yang, Elena Gribovskaya, Nora Kassner, Mor Geva, and Sebastian Riedel. Do large language models latently perform multi-hop reasoning? *arXiv preprint arXiv:2402.16837*, 2024.

[22] Eden Biran, Daniela Gottesman, Sohee Yang, Mor Geva, and Amir Globerson. Hopping too late: Exploring the limitations of large language models on multi-hop queries. *arXiv preprint arXiv:2406.12775*, 2024.

[23] Yuval Shalev, Amir Feder, and Ariel Goldstein. Distributional reasoning in llms: Parallel reasoning processes in multi-hop reasoning. *arXiv preprint arXiv:2406.13858*, 2024.

[24] Xinyi Wang, Lucas Caccia, Oleksiy Ostapenko, Xingdi Yuan, William Yang Wang, and Alessandro Sordoni. Guiding language model reasoning with planning tokens. *arXiv preprint arXiv:2310.05707*, 2023.

[25] Loïc Barrault, Paul-Ambroise Duquenne, Maha Elbayad, Artyom Kozhevnikov, Belen Alastruey, Pierre Andrews, Mariano Coria, Guillaume Couairon, Marta R Costa-jussà, David Dale, et al. Large concept models: Language modeling in a sentence representation space. *arXiv preprint arXiv:2412.08821*, 2024.

[26] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Improving text embeddings with large language models. *arXiv preprint arXiv:2401.00368*, 2023.

[27] 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.

[28] Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. Softcot: Soft chain-of-thought for efficient reasoning with llms. *arXiv preprint arXiv:2502.12134*, 2025.

[29] Mengting Hu, Zhen Zhang, Shiwan Zhao, Minlie Huang, and Bingzhe Wu. Uncertainty in natural language processing: Sources, quantification, and applications. *arXiv preprint arXiv:2306.04459*, 2023.

[30] Alfréd Rényi. On measures of entropy and information. In *Proceedings of the fourth Berkeley symposium on mathematical statistics and probability, volume 1: contributions to the theory of statistics*, volume 4, pages 547–562. University of California Press, 1961.

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

[32] AMC. American invitational mathematics examination. https://artofproblemsolving.com/wiki/index.php/American_Invitational_Mathematics_Examination, 2025.

[33] 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.

[34] 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.

[35] 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. 2021.

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

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

[38] 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.

- [39] 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.
- [40] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

A Appendix

A.1 Limitation

While *Soft Thinking* demonstrates significant improvements in both reasoning accuracy and token efficiency without requiring any additional training, its training-free nature also introduces certain limitations. Specifically, since current large language models are trained exclusively on discrete token sequences, they have never encountered *concept tokens*, the probability-weighted mixtures of token embeddings, during pre-training or fine-tuning. As a result, feeding such continuous *concept tokens* into the model during inference places it in an out-of-distribution (OOD) regime. This can lead to instability or generation collapse, especially when the reasoning chain becomes long or the input distribution diverges from the model’s training data. Although our *Cold Stop* mechanism helps mitigate these issues by terminating reasoning when the model is sufficiently confident, it does not fundamentally resolve the OOD problem. Future work should explore training strategies that explicitly expose the model to *concept tokens*, thereby aligning its internal representations with the continuous concept space and improving robustness and generalization under soft thinking paradigms.

A.2 Benchmarks

The evaluation covers four mathematical benchmark tasks and three programming benchmark tasks.

- **Mathematical Benchmarks:** Math500 [40] is a diverse subset of 500 problems selected from the MATH dataset [31], covering seven mathematical disciplines. AIME 2024 [32] is drawn from the American Invitational Mathematics Examination, this dataset features challenging problems that serve as a rigorous benchmark for both accuracy and token efficiency. GSM8K [33] is a benchmark comprising 1,319 grade-school-level math word problems designed to assess multi-step arithmetic reasoning. GPQA-Diamond [34] is a subset focusing on high-difficulty problems requiring deep, multi-step reasoning.
- **Coding Benchmarks:** HumanEval [35] is a widely used benchmark for evaluating functional correctness in code generation. Each problem consists of a Python function signature and a natural language description, with correctness measured by executing unit tests. MBPP [36] contains introductory-level coding tasks paired with simple specifications and measures a model’s ability to understand and execute basic programming logic. LiveCodeBench [37] is a dynamic, contamination-free code generation benchmark. For our study, we select 279 problems released between August 2024 and January 2025 to ensure evaluation on unseen tasks.

A.3 Implementation of *Soft Thinking* on SGLang

In this appendix, we describe the engineering modifications made to the SGLang inference engine (v0.4.6.post1) to support our proposed **Soft Thinking** method. We highlight the core code changes, affected files, and provide high-level code snippets to clarify the new reasoning flow.

A.3.1 Overview of Modifications

Our implementation introduces a new inference mode, *soft thinking*, in which intermediate reasoning steps use “concept tokens” (i.e., full token probability distributions) rather than discrete token ids. This required changes to the input/output interface, sampling, embedding, and state management in SGLang. The main modifications are summarized as follows:

- **Configuration:** New flags and parameters to enable soft thinking and control its behavior.
- **Sampler:** Modified to output top- k probability distributions (concept tokens) instead of a single sampled token.
- **Embedding Layer:** Added support for probability-weighted interpolation of token embeddings.
- **Forward Pipeline:** Adapted to accept and propagate concept tokens through the model.
- **Cold Stop:** Entropy-based early stopping logic for intermediate reasoning steps.

A.3.2 Key Files and Logic Changes

1. model_config.py & server_args.py

- **Purpose:** Add configuration options for soft thinking.
- **Key additions:**

```

1  enable_soft_thinking: bool
2  max_topk: int
3  # Command-line flags: --enable-soft-thinking, --max-topk, --
   think-end-str

```

2. sampler.py (Sampling Logic)

- **Purpose:** Output concept tokens (top- k probabilities and indices) instead of only discrete token ids.
- **Key changes:**

```

1  if enable_soft_thinking:
2      # Compute top-k probabilities and indices
3      topk_probs, topk_indices = torch.topk(probs, k=max_topk,
4                                              dim=-1)
5      # Normalize
6      topk_probs = topk_probs / topk_probs.sum(dim=-1, keepdim=
7                                              True)
8      logits_output.topk_probs = topk_probs
9      logits_output.topk_indices = topk_indices
10     # For next token id, use argmax or sample from topk as
11     # needed
12     batch_next_token_ids = topk_indices[:, 0]
13 else:
14     # Standard discrete sampling
15     batch_next_token_ids = torch.argmax(probs, -1)
16
17     # Entropy calculation for Cold Stop
18     entropy = -torch.sum(probs * torch.log(probs.clamp(min=1e-12)),
19                           dim=-1)
20     logits_output.entropy = entropy

```

3. vocab_parallel_embedding.py (Embedding Layer)

- **Purpose:** Support probability-weighted embedding computation.
- **Key changes:**

```

1  def weighted_forward(self, topk_probs, topk_indices):
2      # Compute weighted sum of embeddings
3      topk_embeddings = self.quant_method.embedding(self,
4                                                     topk_indices.long()) # [B, K, D]
5      # Normalize probabilities
6      topk_probs = topk_probs / topk_probs.sum(dim=-1, keepdim=
7                                              True)
8      new_embedding = torch.sum(topk_probs.unsqueeze(-1) *
9                               topk_embeddings, dim=1)
10     return new_embedding

```

4. models/llama.py, models/qwen2.py (Model Forward Pass)

- **Purpose:** Accept and process concept tokens as input.
- **Key changes:**

```

1  # In model.forward:
2  if forward_batch.topk_probs is not None and forward_batch.
3      topk_indices is not None:
4      if self.tp_size > 1:
5          hidden_states = self.embed_tokens.weighted_forward_tp(
6              forward_batch.topk_probs, forward_batch.topk_indices
7          )
8      else:
9          hidden_states = self.embed_tokens.weighted_forward(
10             forward_batch.topk_probs, forward_batch.topk_indices
11         )
12 elif input_embeds is None:
13     hidden_states = self.embed_tokens(input_ids)

```

5. schedule_batch.py, scheduler.py, scheduler_output_processor_mixin.py

- **Purpose:** State management and output tracking for soft thinking.
- **Key changes:**

```

1  # Pseudocode for Cold Stop
2  if entropy < entropy_threshold:
3      low_entropy_steps += 1
4  else:
5      low_entropy_steps = 0
6  if low_entropy_steps >= length_threshold:
7      # Insert end-of-thinking token, switch to answer mode
8      self.output_ids[-1] = self.sampling_params.think_end_str_id

```

6. sampling_params.py, sampling_batch_info.py

- **Purpose:** Add soft thinking-specific parameters and per-batch flags.
- **Key changes:**

```

1  # Parameters for after-thinking sampling, entropy thresholds,
2  # etc.
3  early_stopping_entropy_threshold: float
4  early_stopping_length_threshold: int
5  soft_thinking_mode: Optional[torch.Tensor]

```

A.3.3 High-Level Soft Thinking Inference Flow

1. **Initialization:** If `enable_soft_thinking` is set, the model enters soft thinking mode for reasoning steps.
2. **At Each Reasoning Step:**
 - The sampler outputs a concept token: top- k probabilities and indices (not just a single token id).
 - The embedding layer computes the next input embedding as a weighted sum over token embeddings, using these probabilities.
 - The model forward pass consumes this weighted embedding.
3. **Cold Stop (Early Termination):**
 - At each step, compute the entropy of the concept token.
 - If entropy is below a threshold for several consecutive steps, insert the end-of-thinking token to terminate reasoning.
4. **Answer Generation:** After reasoning, the model switches back to standard discrete decoding for the answer.

The above changes enable SGLang to support soft thinking as described in our paper, allowing for continuous, distributional reasoning and entropy-based early stopping, all with minimal overhead to the standard inference pipeline.

NeurIPS Paper Checklist

1. Claims

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

Answer: [\[Yes\]](#)

Justification: These sections comprehensively outline the research objectives, methodology, and expected outcomes, closely aligning with the in-depth analysis and results presented in the full text.

Guidelines:

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

2. Limitations

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

Answer: [\[Yes\]](#)

Justification: See Section A.1.

Guidelines:

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

3. Theory assumptions and proofs

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

Answer: [\[Yes\]](#)

Justification: See Section 3.3.

Guidelines:

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

4. Experimental result reproducibility

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

Answer: [\[Yes\]](#)

Justification: We have added all data and code to the supplementary materials to facilitate reviewers to reproduce our work. We have detailed all experimental details (e.g., model selection, hyperparameters, and experimental equipment) in Sections 4.1 and 4.2 of the paper. Besides, we have implemented our code in the popular open-source inference framework SGLang and highlighted the key code content in the appendix A.3 to facilitate reproduction.

Guidelines:

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

5. Open access to data and code

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

Answer: [Yes]

Justification: We add all the data, and code results in the supplementary material, which makes it easy to reproduce our work. We make them public on GitHub. Besides, we have implemented our code in the popular open-source inference framework SGLang and highlighted the key code content in the appendix A.3 to facilitate reproduction.

Guidelines:

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

6. Experimental setting/details

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

Answer: [Yes]

Justification: We have detailed all experimental details (e.g., model selection, hyperparameters, and experimental equipment) in Sections 4.1 and 4.2 of the paper.

Guidelines:

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

7. Experiment statistical significance

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

Answer: [Yes]

Justification: The paper reports averaged results over 16 runs to ensure stability, with detailed information provided in Section 4.2. This adequately supports the statistical significance of the experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.

- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

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

Answer: [\[Yes\]](#)

Justification: We have detailed all computer resources (e.g., type of GPUs) in Sections 4.1 of the paper.

Guidelines:

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

9. Code of ethics

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

Answer: [\[Yes\]](#)

Justification: The data we used is all open source, including mathematics and code, and does not contain any content that violates ethical requirements.

Guidelines:

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

10. Broader impacts

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

Answer: [\[NA\]](#)

Justification: Our paper does not specifically discuss potential positive or negative societal impacts, as its primary focus is on methodological innovation and technical evaluation. This section is not directly applicable to our work.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

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

Answer: [NA]

Justification: Our paper does not describe specific safeguards for responsible release, as it does not involve the release of new high-risk models or sensitive datasets. This section is not applicable to our work.

Guidelines:

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

12. Licenses for existing assets

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

Answer: [Yes]

Justification: Yes, all creators and original owners of assets (such as code, data, and models) used in Section 4.1 of our paper are properly credited through citations. We use only publicly available resources with appropriate licenses, and all terms of use are respected and explicitly referenced in the manuscript.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. New assets

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

Answer: [NA]

Justification: No new assets are introduced in our paper. Therefore, this section is not applicable.

Guidelines:

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

14. Crowdsourcing and research with human subjects

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

Answer: [NA]

Justification: Our paper does not involve crowdsourcing experiments or research with human subjects. Therefore, this section is not applicable.

Guidelines:

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

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

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

Answer: [NA]

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

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. Declaration of LLM usage

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

Answer: [NA]

Justification: We only use LLM to check syntax and calibrate typos, and not for other purposes.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.