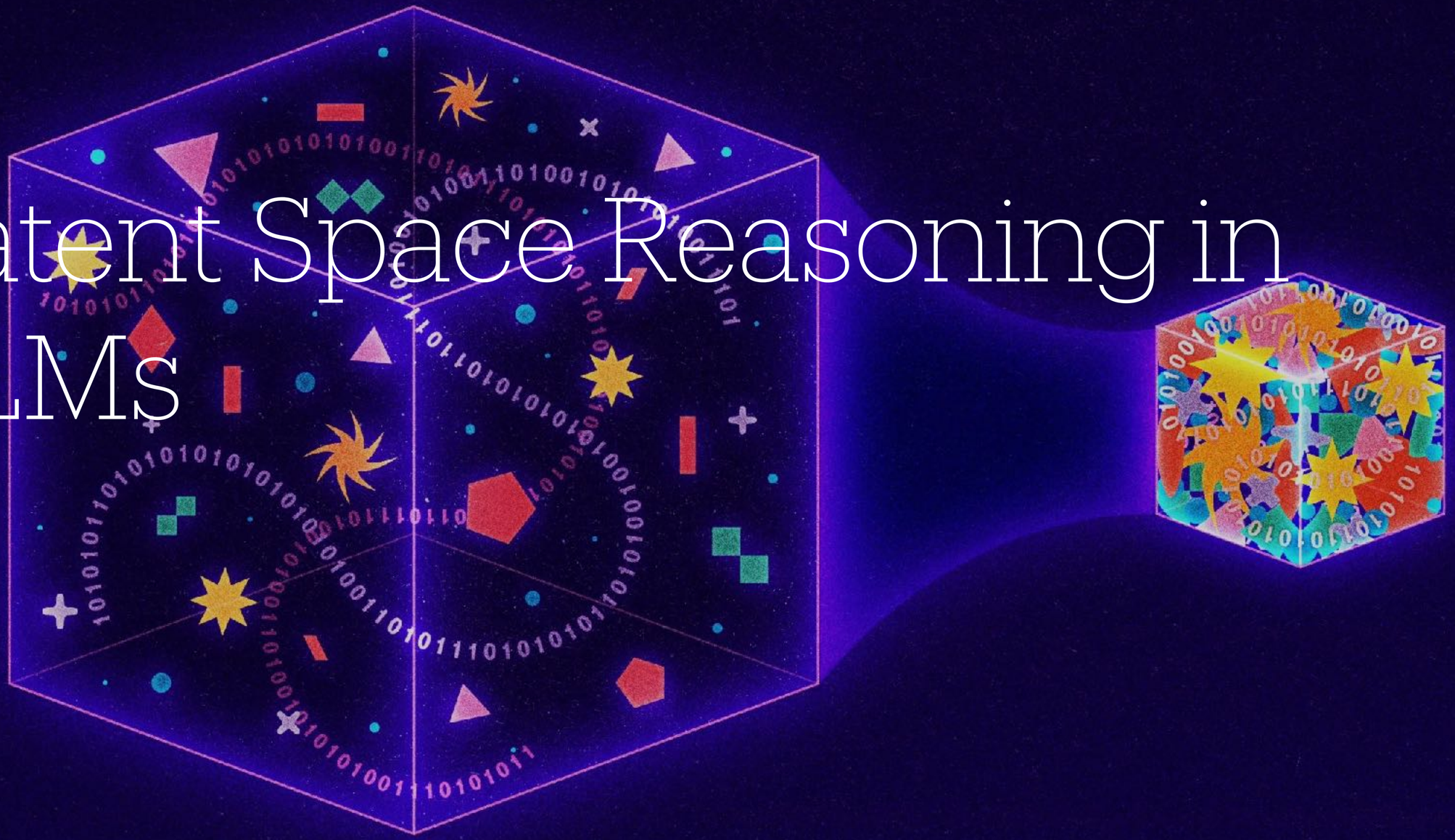
Latent Space Reasoning in LLMs



Presented by Jianing Ye, Luise Ge, and Kefei Duan.

Paper 1: Chain of Continuous Thought

Training Large Language Models to Reason in a Continuous Latent Space

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, Yuandong Tian

Fall 2025 LLM class presentation, Jianing Ye

Motivation

Limitations of Language Reasoning

- **Reasoning Efficiency**: Chain of Thought (CoT) uses a lot of words for **textual coherence** and not essential for **reasoning**.
- **Planning Ability**: Planning in language space poses challenges to LLMs, in that some tokens require complex planning.
- **Error Recovery**: CoT follows a single path; once it commits to a wrong branch, it cannot easily backtrack.

Motivation

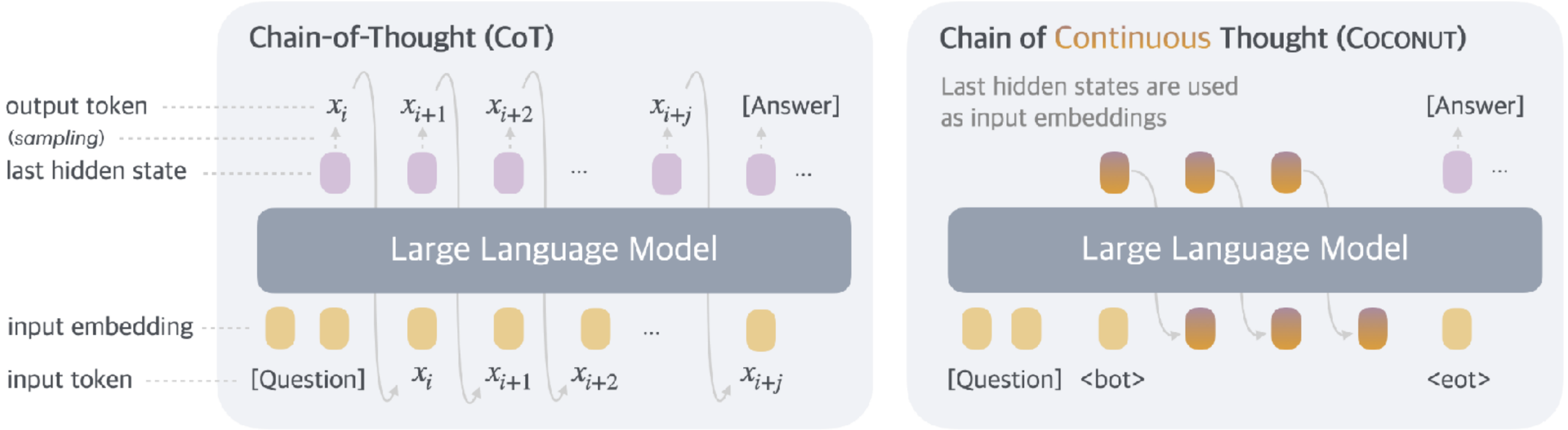
Reasoning in latent space

- **Reasoning Efficiency**: Latent reasoning avoid redundant words, enabling compact reasoning process.
- **Planning Ability**: Latent space supports parallel exploration of branches, resembling BFS, which is more suitable for complex planning.
- **Error Recovery**: By maintaining multiple candidate paths in latent space, the model can prune less promising branches instead of committing too early.
- **Cognitive evidence**: Human reasoning often occurs outside the language network, suggesting reasoning need not rely on language.

Method Overview

Inference phase

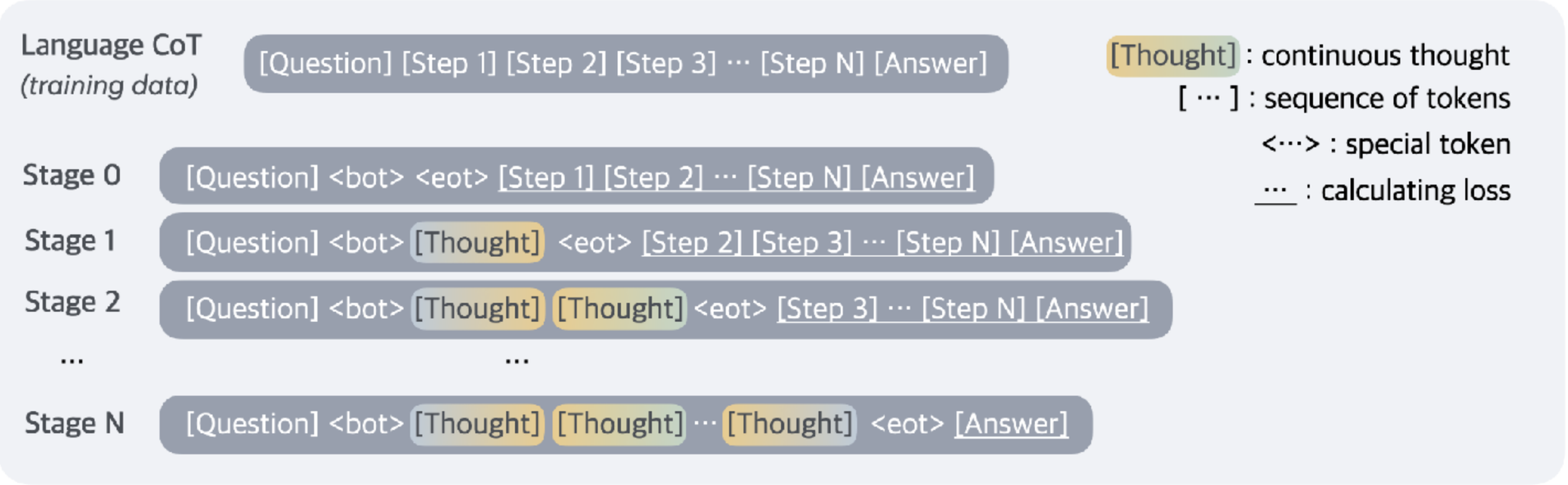
- Two strategies for mode switching
  - Autonomous switching by a trained binary classifier
  - Padding the latent thoughts to a constant length



Method Overview

Training phase

- Curriculum learning
  - Increase the continuous thinking steps gradually.
    - each language step is replaced by c continuous thoughts
  - Mask the loss of continuous thinking part.
  - Lower training efficiency due to the non-parallelizable autoregressive latent thinking.



Experiment

Reasoning Tasks

- **Math Reasoning**: GSM8k (grade-school math problems)
  - training set: synthetic CoT dataset by Deng et al.
- **Logic Reasoning 1**: 5-hop ProntoQA (a simple reasoning task with given premises)
- **Logic Reasoning 2**: ProsQA (a harder version of ProntoQA)

Experiment

Setup

- **Model**: Pre-trained GPT-2
- **Evaluation**: Greedy decoding
- **Math Reasoning**
  - Max thinking steps: $3 \times c$
  - Additional stage: remove the language tail.
- **Logic Reasoning**
  - Max thinking steps: $6 \times c$

Experiment

Baselines & Coconut variants

- Baselines
  - CoT: Classical Chain-of-Thought.
  - No-CoT: Predict the final answer directly.
  - Pause token: <pause> is inserted between the question and the answer.
  - iCoT: Curriculum learning, trained with a thought chain of decreasing length, until only the answer remains.

- Coconut variants
  - w/o curriculum
  - w/o thought: very similar to iCoT.
  - pause as thought: replace thoughts with <pause>

Experiment

Results

Method	GSM8k		ProntoQA		ProsQA	
	Acc. (%)	# Tokens	Acc. (%)	# Tokens	Acc. (%)	# Tokens
CoT	42.9 ±0.2	25.0	98.8 ±0.8	92.5	77.5 ±1.9	49.4
No-CoT	16.5 ±0.5	2.2	93.8 ±0.7	3.0	76.7 ±1.0	8.2
iCoT	30.0*	2.2	99.8 ±0.3	3.0	98.2 ±0.3	8.2
Pause Token	16.4 ±1.8	2.2	77.7 ±21.0	3.0	75.9 ±0.7	8.2
COCONUT (Ours)	34.1 ±1.5	8.2	99.8 ±0.2	9.0	97.0 ±0.3	14.2
- *w/o curriculum*	14.4 ±0.8	8.2	52.4 ±0.4	9.0	76.1 ±0.2	14.2
- *w/o thought*	21.6 ±0.5	2.3	99.9 ±0.1	3.0	95.5 ±1.1	8.2
- *pause as thought*	24.1 ±0.7	2.2	100.0 ±0.1	3.0	96.6 ±0.8	8.2

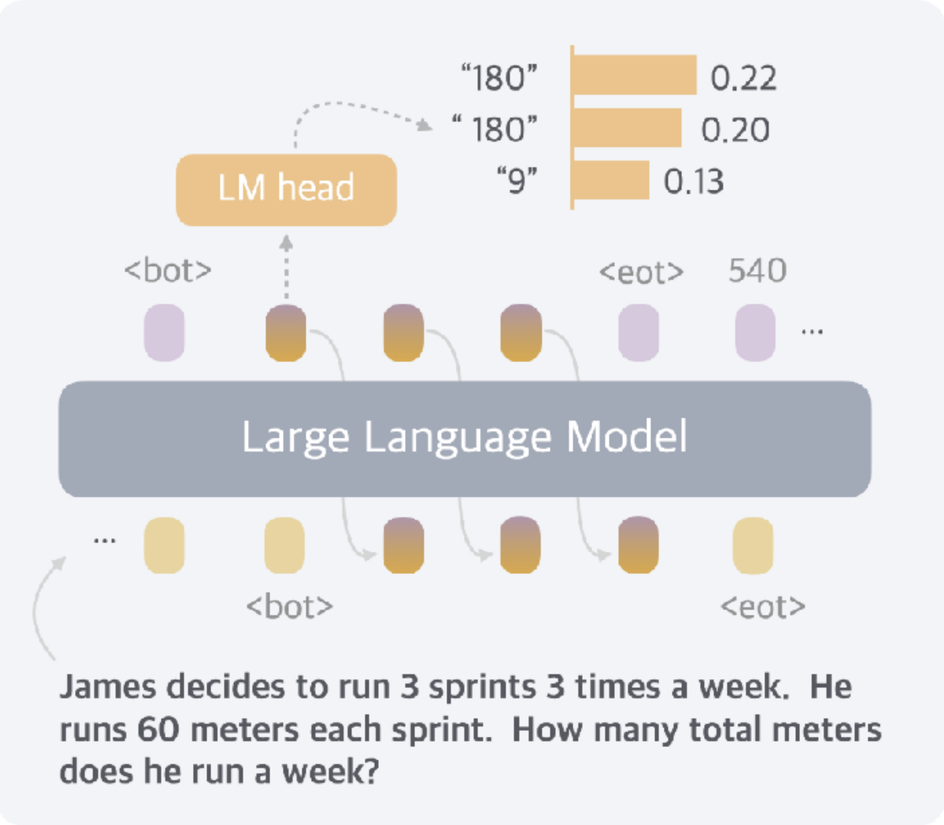
Experiment

Discussions

- The latent reasoning increases the expressiveness of chain and thus enhances the reasoning ability.
- Latent reasoning outperforms language reason in planning-intensive tasks.
- LLMs still need guides to learn latent reasoning.
- Continuous thoughts are efficient representation for reasoning.

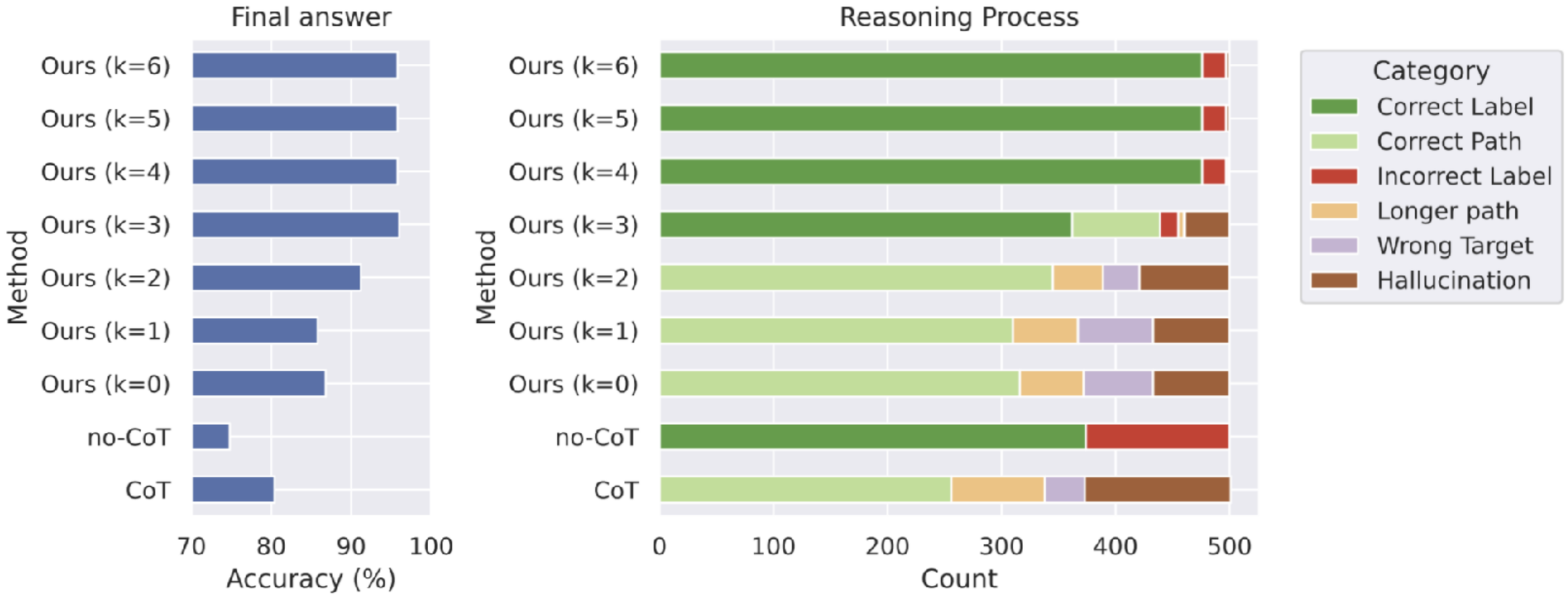
A Case Study

For the representative ability if continuous thought



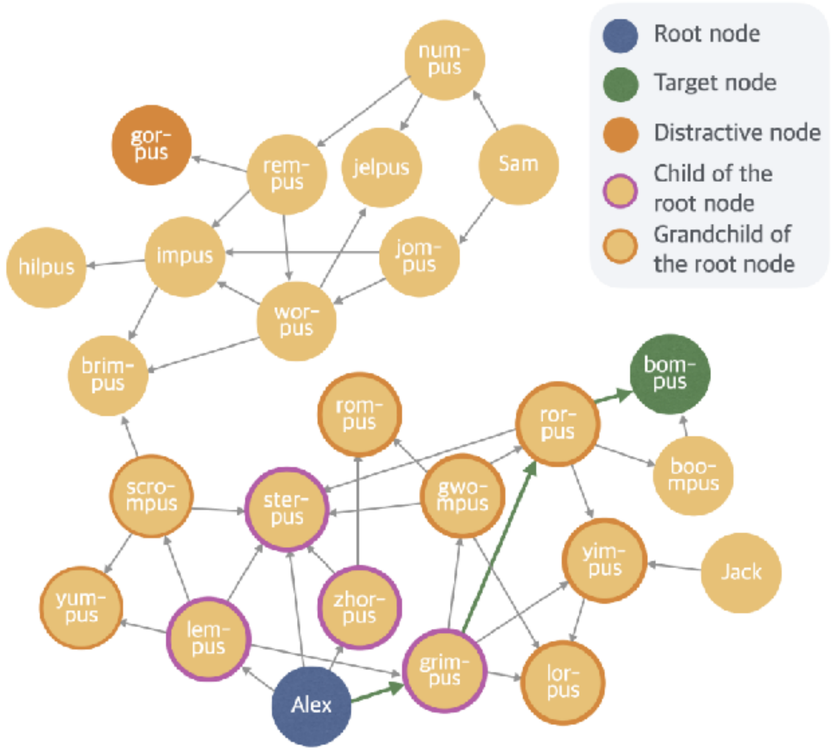
Experiment

on different latent thinking step k



A Case Study

on ProsQA



Question:

Every grimpus is a yimpus. Every worpus is a jelpus. Every zhorpus is a sterpus. Alex is a grimpus ⋯ Every lumps is a yumpus.
Question: **Is Alex a gorpus or bompus?**

Ground Truth Solution

Alex is a grimpus.
Every grimpus is a rorpus.
Every rorpus is a bompus.
Alex is a bompus

CoT

Alex is a lempus.
Every lempus is a scrompus.
Every scrompus is a yumpus.
Every yumpus is a rempus.
Every rempus is a gorpus.
Alex is a gorpus ✗

(Hallucination)

Coconut (k=1)

<bot> [Thought] <eot>
Every lempus is a scrompus.
Every scrompus is a brimpus.
Alex is a brimpus ✗

(Wrong Target)

Coconut (k=2)

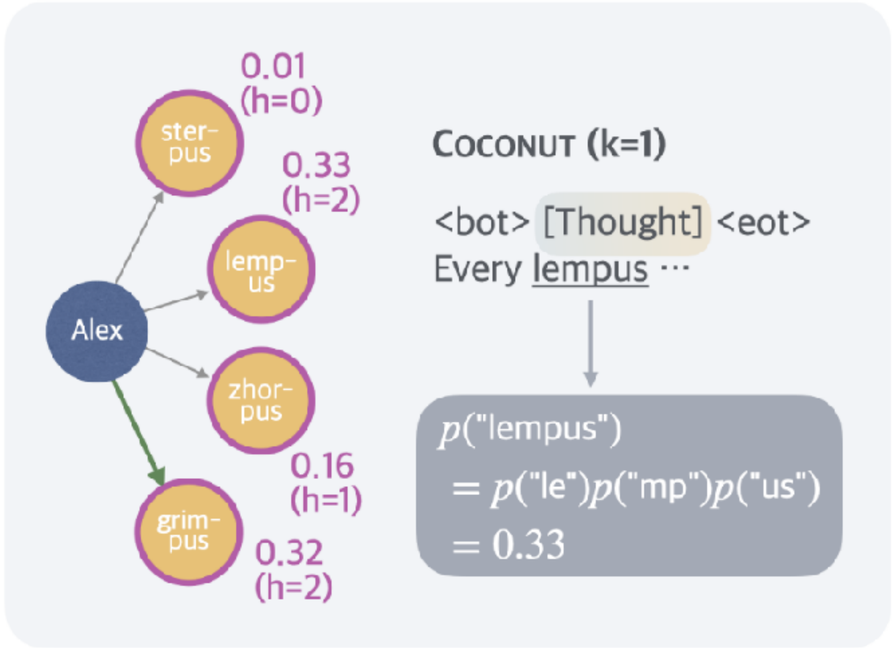
<bot> [Thought] [Thought] <eot>
Every rorpus is a bompus.
Alex is a bompus ✓

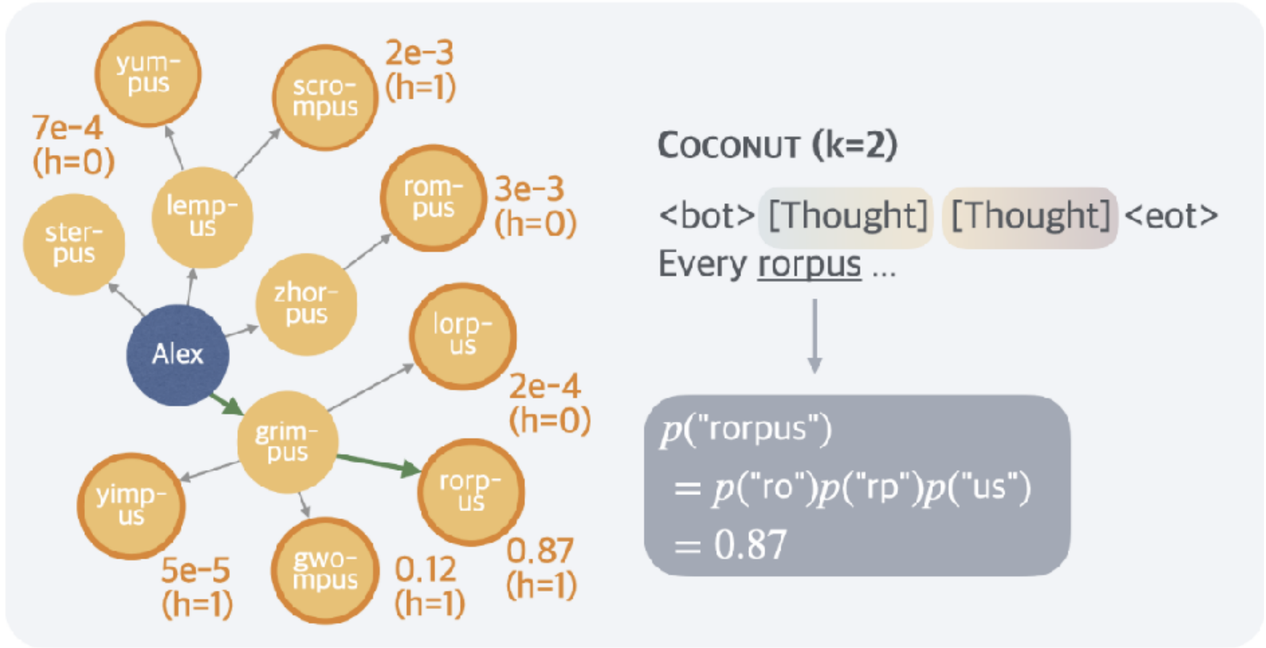
(Correct Path)

A Case Study

on ProsQA

- BFS feature

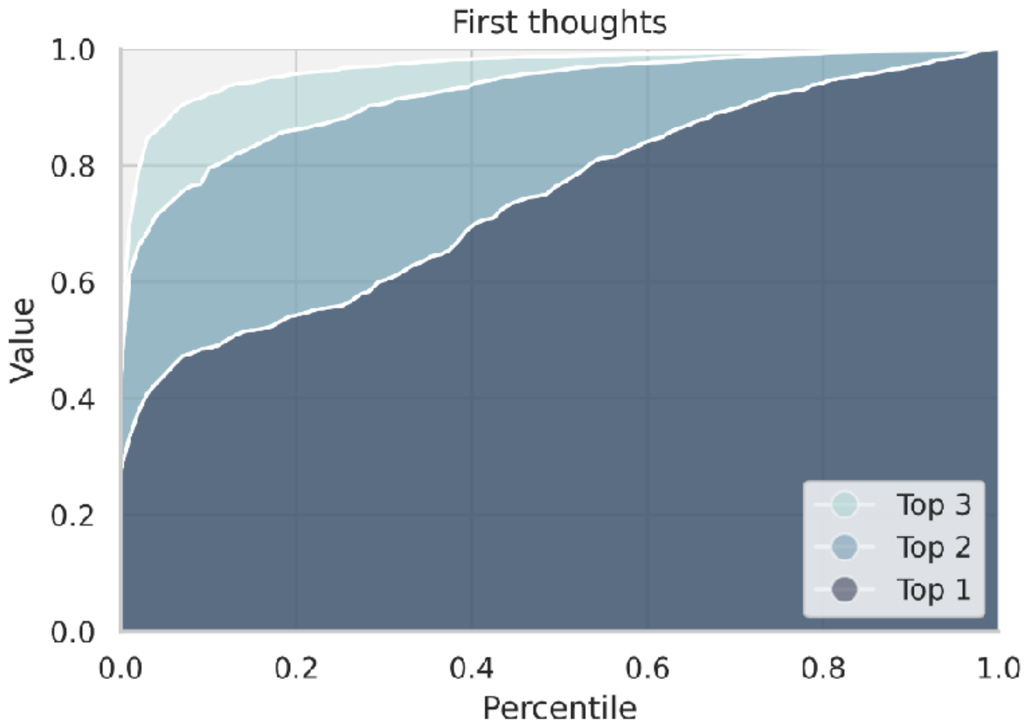


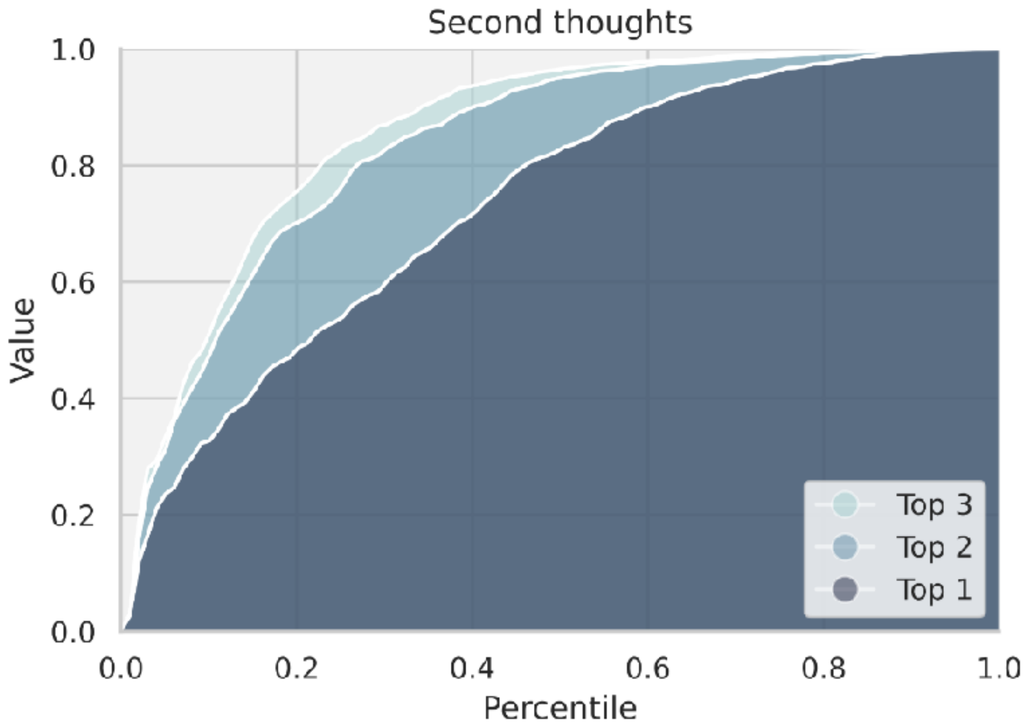


A Case Study

on ProsQA

- BFS feature





Summary/Take-aways

- **New paradigm**: Introduces Coconut, a framework for reasoning in continuous latent space (Chain of Continuous Thoughts), beyond language-based CoT.
- **Key insight**: Latent reasoning enables parallel branch exploration (BFS-like), improving planning and error recovery compared to single-path CoT.
- **Empirical findings**: Coconut is more token-efficient and excels on logic/graph reasoning tasks, though less effective on arithmetic (GSM8K), highlighting complementary strengths with CoT.

Paper 2: Compressed Chain of Thought

By Jeffery Cheng and Benjamin Van Durme

Comparison with COCONUT

- **Shared Motivation:** making reasoning efficient while preserving CoT benefits
- **Different Approaches:**

  COCONUT retrains the entire LLM

  CCOT adds external modules to compress reasoning into continuous tokens
- **Different inferences:**

  COCONUT ’s inference has fixed length

  CCOT’s inference is of variable-length

Motivation 1: high-generation latency

Algorithm 1 Chain of Thought inference

Require: Query w, parameters θ

$\bar{w} \leftarrow \text{EMBED}_\theta(w)$ ▷ *embed query*

$\hat{w} \leftarrow \text{ATTN}_\theta(\bar{w})$ ▷ *compute hidden states*

$z \leftarrow [\langle COT \rangle]$

while $z_{-1} \neq \langle ANS \rangle$ **do**

  $[\hat{w}; \hat{z}] \leftarrow \text{ATTN}_\theta([\bar{w}; \text{EMBED}_\theta(z)])$

  $x \sim \text{HEAD}_\theta(\hat{z}^L_{-1})$

  $z \leftarrow [z; x]$

end while

$a \leftarrow [\langle ANS \rangle]$

while $a_{-1} \neq \langle EOS \rangle$ **do**

  $[\hat{w}; \hat{z}; \hat{a}] \leftarrow \text{ATTN}_\theta([\bar{w}_{1:n}; \text{EMBED}_\theta([z; a])])$

  $x \sim \text{HEAD}_\theta(\hat{a}^L_{-1})$ ▷ *sample answer token*

  $a \leftarrow [a; x]$

end while

return a

Token->Embedding->Token->Embedding->Token

Recall the lm architecture:

$$w^0_{1:n} = \text{EMBED}_\theta(w_{1:n}) \quad \triangleright \textit{embedding layer}$$

$$w^\ell_{1:n} = \text{ATTN}^{\ell-1}_\theta(w^{\ell-1}_{1:n}) \quad \triangleright \textit{transformer blocks}$$

$$p_{1:n} = \text{HEAD}_\theta(w^L_{1:n}) \quad \triangleright \textit{pass through lm head}$$

$$p(w_{n+1} \mid w_{1:n}) \sim p_n \quad \triangleright \textit{sample next token}$$

Motivation 2: Compression with Adapter

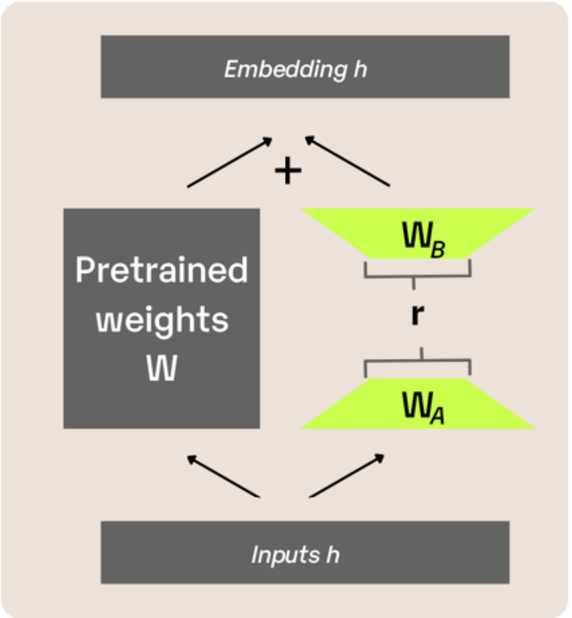
- Parameter-Efficient-Fine-Tuning (PEFT)
- Observation:

an adapter trained to decode from the compressed hidden states can achieve lossless performance compared to decode from the full reasoning chain.

In other words, the **information needed for solving the task is already redundant** in the full chain. A compressed subset of hidden states suffices.

But there is still the cost to generate the full CoT in the first place. -> Can we also generate compressed CoT?

Low-Rank Adaptation(LoRA)



Proposal: generate compressed representation of CoT

Algorithm 1 Chain of Thought inference

Require: Query w, parameters θ

$\bar{w} \leftarrow \text{EMBED}_\theta(w)$ ▷ *embed query*
$\hat{w} \leftarrow \text{ATTN}_\theta(\bar{w})$ ▷ *compute hidden states*
$z \leftarrow [\langle COT \rangle]$
while $z_{-1} \neq \langle ANS \rangle$ **do**
  $[\hat{w}; \hat{z}] \leftarrow \text{ATTN}_\theta([\bar{w}; \text{EMBED}_\theta(z)])$
  $x \sim \text{HEAD}_\theta(\hat{z}^L_{-1})$
  $z \leftarrow [z; x]$
end while
$a \leftarrow [\langle ANS \rangle]$
while $a_{-1} \neq \langle EOS \rangle$ **do**
  $[\hat{w}; \hat{z}; \hat{a}] \leftarrow \text{ATTN}_\theta([\bar{w}_{1:n}; \text{EMBED}_\theta([z; a])])$
  $x \sim \text{HEAD}_\theta(\hat{a}^L_{-1})$ ▷ *sample answer token*
  $a \leftarrow [a; x]$
end while
return a

Algorithm 2 CCOT inference

Require: Query w, parameters θ, φ, ψ, autoregressive layer l

$\bar{w} \leftarrow \text{EMBED}_\theta(w)$ ▷ *embed query*
$\hat{w} \leftarrow \text{ATTN}_\theta(\bar{w})$ ▷ *compute hidden states*
$z \leftarrow [\hat{w}^l_{-1}]$
while $\text{END}_\psi(\hat{z}^L)$ is $False$ **do**
  $[\hat{w}; \hat{z}] \leftarrow \text{ATTN}_{\theta,\varphi}([\bar{w}; z])$ ▷ *gen. cont. token*
  $z \leftarrow [z; \hat{z}^l_{-1}]$ ▷ *append cont. token*
end while
$a \leftarrow [\langle ANS \rangle]$
while $a_{-1} \neq \langle EOS \rangle$ **do**
  $[\hat{w}; \hat{z}; \hat{a}] \leftarrow \text{ATTN}_{\theta,\varphi,\psi}([\bar{w}_{1:n}; z; \text{EMBED}_\theta(a)])$
  $x \sim \text{HEAD}_\psi(\hat{a}^L_{-1})$ ▷ *sample answer token*
  $a \leftarrow [a : x]$
end while
return a


3 Musketeers
Bar
Contemplation Token Generator ϕ
Base LLM θ
Contemplation Token Decoder ψ

Training the generator φ

- The goal is to approximate a **compressed** *representation* of the full reasoning chain
- Getting the gold label: First obtain the hidden states of the input at ℓ-th layer; Then using some scorer function to assign importance score to each token and select a subset of **k** tokens accordingly.
- Setting the input: The hidden layer of the query should *autoregressively* generate the compression.
- Setting the loss function: Layer-by-layer MSE.

Additionally, training a contemplation ending signal

- k will not be know during test time
- So additionaly, a binary classifier **END ϕ** is trained: taking the last-layer hidden states as input, and predicts whether or not to stop.

Training the Decoder ψ

- The goal is to decode the answer from the query and the contemplation tokens.
- ***Because the comtemplation tokens are out of distribution for the base LM, we train this separate module to avoid breaking the LM's pretrained ability.***
- Input: query hidden states + contemplation tokens
- Output: Autoregressively outputs the correct answer
- Loss function: Cross-Entropy Loss

Joint Training

- Note the modules are interdependent
- In their actual doing, they also allow the signals of the cross-entropy loss to flow back to the generator.
- However, they find the signals not fully useful —> ended up unfreezing only the parameters of the layers after the selected hidden layer ℓ.

Experiments

- Uses a reasoning dataset GSM8K to finetune the generator and decoder
- And checked 4 different compression ratios: 0.0, 0.05, 0.1, 1.0.
-

Format	$1/r$	Acc. (EM)	Decode Time
CCOT	∞	0.089	0.33
CCOT	20x	0.151	0.49
CCOT	10x	0.179	0.78
CCOT	1x	0.315	8.10
PAUSE	20x	0.092	0.35
PAUSE	10x	0.099	0.37

Hyperparameter Choices

Varying compression ratio r:

Original thoughtis that increasing r would keep increasing the accuracy and decode time; as it controls how many contemplation token are generated.

In practice: accuracy plateaus around r=0.2. (Suspision is that the appraximation error could propagate.)

Varying the choice of picked hidden layer ℓ:

$\ell \approx L/2$ gives the best results.

Varying the subset selection:

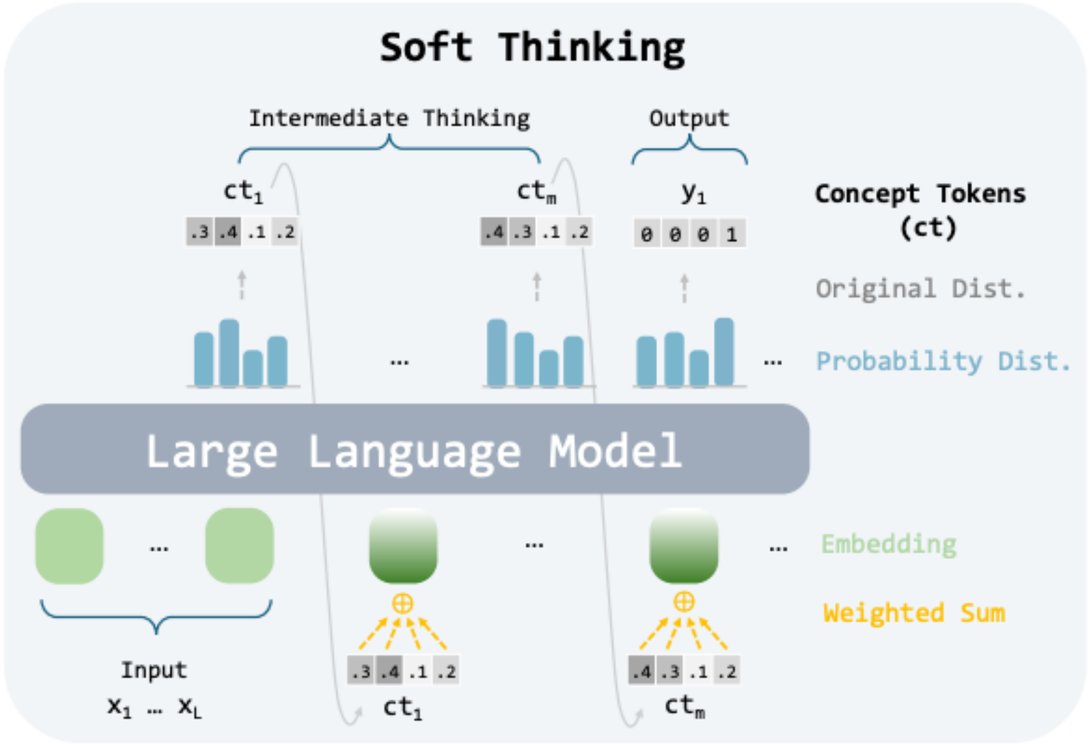
Seems like the picked scorer preforms similarly just pick k evenly distributed token.

Paper 3: Soft Thinking

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

By Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, Shuohang Wang, Yelong Shen, Xin Eric Wang

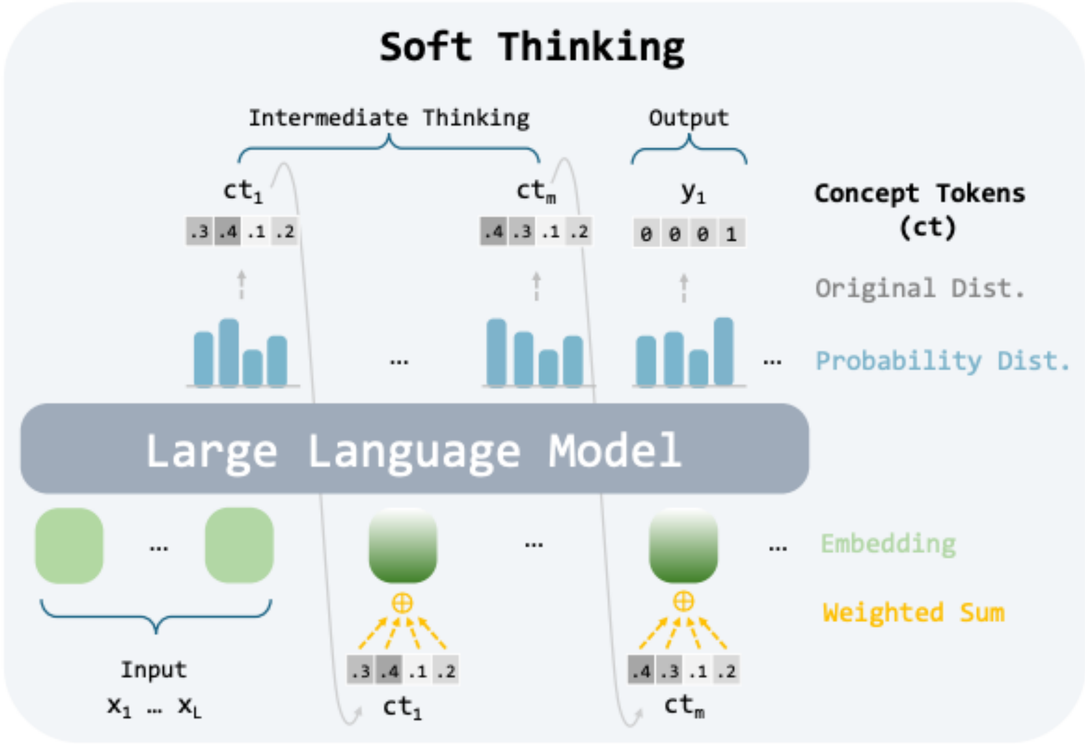
Overview



- **Similar:** Emulate human-like “soft” reasoning by generating soft, abstract concept tokens in a continuous concept space
- **Different:** A Training-free method
- Concept tokens are created by the probability-weighted mixture of token embeddings

Method

- Replace the discrete token in CoT with **soft token**



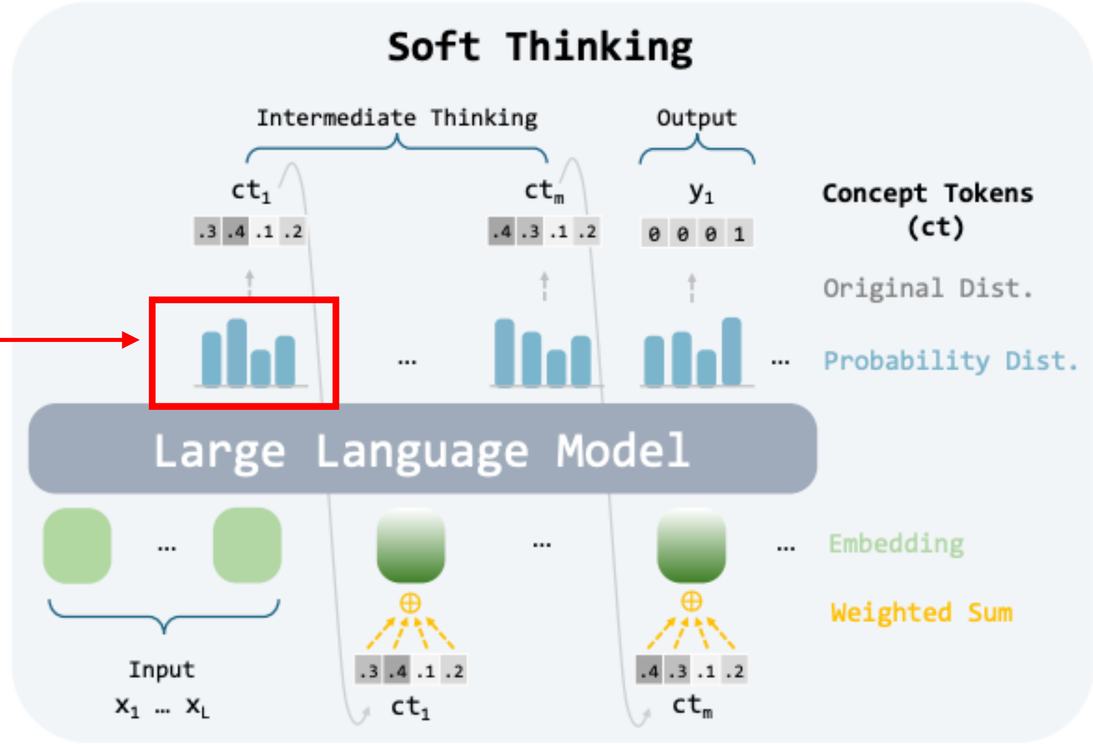
Method

- Replace the discrete token in CoT with **soft token**

In CoT, we sample a token from this distribution

$$t_i \sim p_i = \mathbf{LLM}\big(e(x_{1:l}), e(t_{1:i-1})\big)$$

The embedding of the selected token will be the next embedding given to the transformer layer



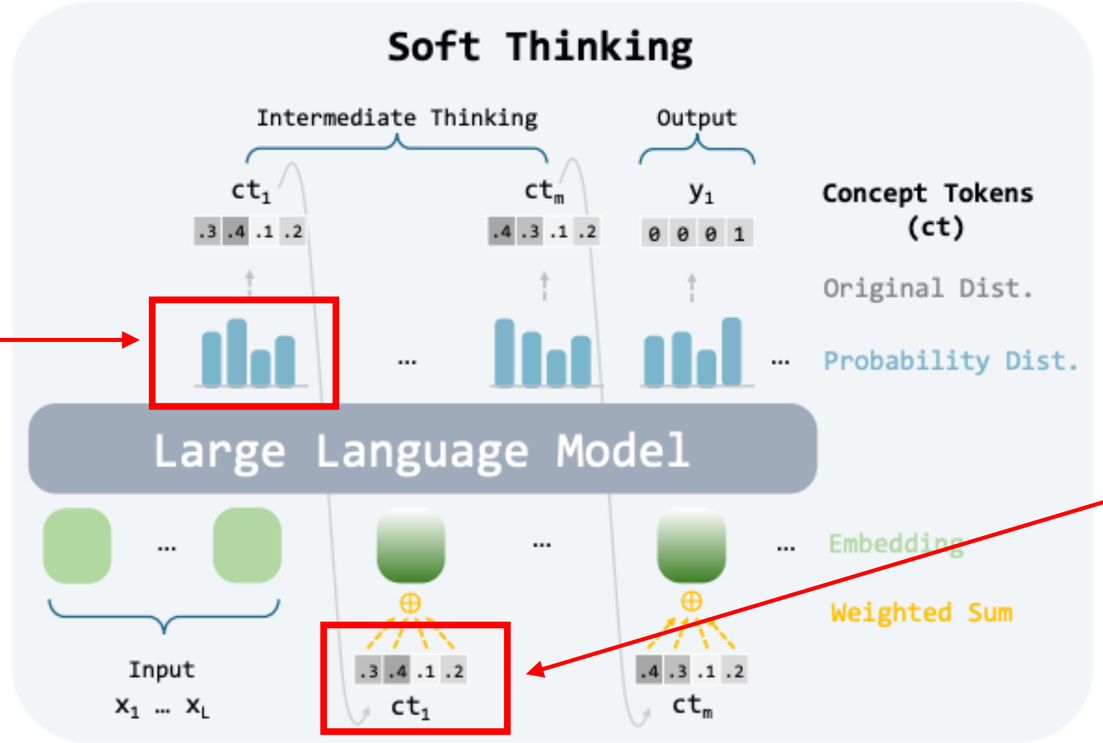
Method

- Replace the discrete token in CoT with **soft token**

In CoT, we sample a token from this distribution

$$t_i \sim p_i = \mathbf{LLM}\big(e(x_{1:l}), e(t_{1:i-1})\big)$$

The embedding of the selected token will be the next embedding given to the transformer layer



In Soft Thinking, we utilize this distribution to do a soft aggregation over the entire vocabulary

$$\tilde{e}_{\text{next}} = \sum_{k=1}^{|V|} ct[k]\, e(k) = \sum_{k=1}^{|V|} p[k]\, e(k)$$

Method

- Introduce **Cold Stop**

- Feeding in continuous concept tokens during inference places the model in an out-of-distribution (OOD) regime, leading to generation collapse (e.g. repetition)

Question:
1+1=?

Without cold stop:
<think>let me solve this question. .. (some thinking) .. Therefore the correct answer is 2, Erm, let me verify my answer in another way. let's Let's Let's Let's Let's . (repetition occurs until the maximum number of tokens is reached)

With cold step:
<thine>let me solve this question. .. (some thinking) .. Therefore the correct answer is 2, Emm, let me verify my answer in another way. let's Let's Let's(cold stop here)</think> The final answer is 2.

Method

- Introduce **Cold Stop**
- Feeding in continuous concept tokens during inference places the model in an out-of-distribution (OOD) regime, leading to generation collapse (e.g. repetition)
- **Stop intermediate reasoning when the model becomes overconfident**
  - Use entropy as a measurement of confidence
  - Stop thinking when encountering low-entropy distributions for a certain number of times

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

Experiments

- Soft Thinking achieves higher Pass@1, while requiring fewer tokens
- Mathematical datasets:

	Accuracy ↑					Generation Length ↓				
	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 (↑ 0.84)	3827	11086	1536	7417	5967 (↓ 7.8%)
Soft Thinking	**98.00**	**83.33**	**96.81**	**67.17**	**86.32** (↑ 2.48)	**3644**	**10627**	**1391**	**7213**	**5719** (↓ 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 (↓ 3.64)	3651	8050	1048	8395	5286 (↑ 5.8%)
Soft Thinking	**95.00**	**76.66**	**95.83**	**64.64**	**83.03** (↑ 1.71)	**3373**	**6620**	**785**	**4722**	**3875** (↓ 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 (↑ 0.61)	**2877**	9457	606	**4443**	4345 (↓ 3.1%)
Soft Thinking	**94.80**	**73.33**	**94.90**	**66.66**	**82.42** (↑ 1.11)	3021	**6644**	**597**	4470	3683 (↓ 17.9%)

Experiments

- Soft Thinking achieves higher Pass@1, while requiring fewer tokens
- Coding datasets:

	Accuracy ↑				Generation Length ↓			
	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 (↓ 2.51)	**2396**	**2069**	**7034**	**3833** (↓ 21.8%)
Soft Thinking	**98.17**	**97.66**	**62.72**	**86.18** (↑ 0.48)	2638	2157	7535	4110 (↓ 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 (↓ 10.53)	**2294**	**1703**	**4702**	**2900** (↓ 38.9%)
Soft Thinking	**97.56**	**95.33**	**59.50**	**84.13** (↑ 0.90)	2713	2534	6255	3834 (↓ 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 (↓ 5.84)	2192	1979	5438	3203 (↓ 28.3%)
Soft Thinking	**98.17**	**94.94**	**58.42**	**83.84** (↑ 0.70)	2498	2214	6512	3741 (↓ 16.3%)

Why Soft Thinking helps?

- Using concept tokens allows the model to avoid making hard decisions too early.
- Keeping the full probability distribution over vocabulary gives it the flexibility to explore different reasoning paths, especially when it’s unsure.
- **Enable the simultaneous exploration of diverse reasoning paths**

Why Soft Thinking helps?

- Using concept tokens allows the model to avoid making hard decisions too early.
- Keeping the full probability distribution over vocabulary gives it the flexibility to explore different reasoning paths, especially when it’s unsure.
- **Enable the simultaneous exploration of diverse reasoning paths**

Is that actually the case?

Paper 4: LLMs are Single-threaded Reasoners

LLMs are Single-threaded Reasoners: Demystifying the Working Mechanism of Soft Thinking

By Chunhung Wu, Jinliang Lu, Zixuan Ren, Gangqiang Hu, Zhi Wu, Dai Dai, Hua Wu

Is Soft Thinking Effective?

- Vanilla Soft Thinking consistently underperforms compared to discrete Token Thinking

Thinking Mode	AIME24	AIME25	MATH500	AMC23	GPQA-Diamond	HumanEval	MBPP	LiveCodeBench	Avg
	Deepseek-R1-Distill-Qwen-32B								
Token (Greedy)	66.66	50.00	92.20	85.00	60.10	87.20	88.71	42.65	71.57
Token (Sampling)	72.08	55.63	94.50	95.46	60.60	97.25	95.13	57.35	78.50
Soft (Vanilla)	62.00	49.17	91.60	90.00	60.10	86.41	87.93	44.80	72.13
	QwQ-32B								
Token (Greedy)	80.00	70.00	97.00	100.00	64.14	95.12	96.10	58.78	82.64
Token (Sampling)	77.92	67.50	96.20	97.50	62.63	98.17	96.89	62.00	82.35
Soft (Vanilla)	76.67	62.29	96.20	98.75	59.60	93.90	95.33	57.71	80.06
	Skywork-OR1-32B								
Token (Greedy)	76.67	73.33	95.80	90.00	56.06	81.71	86.38	54.84	76.85
Token (Sampling)	78.75	71.25	96.40	98.28	62.62	96.95	97.28	62.37	82.99
Soft (Vanilla)	79.16	69.38	96.00	97.97	59.60	85.37	90.66	55.56	79.21

Is Soft Thinking Effective?

- Vanilla Soft Thinking consistently underperforms compared to discrete Token Thinking

Thinking Mode	AIME24	AIME25	MATH500	AMC23	GPQA-Diamond	HumanEval	MBPP	LiveCodeBench	Avg
	Deepseek-R1-Distill-Qwen-32B								
Token (Greedy)	66.66	50.00	92.20	85.00	60.10	87.20	88.71	42.65	71.57
Token (Sampling)	72.08	55.63	94.50	95.46	60.60	97.25	95.13	57.35	78.50
Soft (Vanilla)	62.00	49.17	91.60	90.00	60.10	86.41	87.93	44.80	72.13
	QwQ-32B								
Token (Greedy)	80.00	70.00	97.00	100.00	64.14	95.12	96.10	58.78	82.64
Token (Sampling)	77.92	67.50	96.20	97.50	62.63	98.17	96.89	62.00	82.35
Soft (Vanilla)	76.67	62.29	96.20	98.75	59.60	93.90	95.33	57.71	80.06
	Skywork-OR1-32B								
Token (Greedy)	76.67	73.33	95.80	90.00	56.06	81.71	86.38	54.84	76.85
Token (Sampling)	78.75	71.25	96.40	98.28	62.62	96.95	97.28	62.37	82.99
Soft (Vanilla)	79.16	69.38	96.00	97.97	59.60	85.37	90.66	55.56	79.21

What is Vanilla Soft Thinking?

Is Soft Thinking Effective?

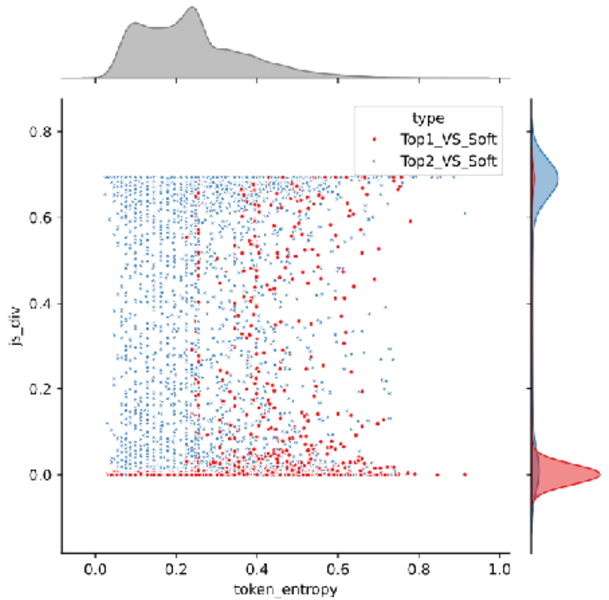
- Vanilla Soft Thinking consistently underperforms compared to discrete Token Thinking

Thinking Mode	AIME24	AIME25	MATH500	AMC23	GPQA-Diamond	HumanEval	MBPP	LiveCodeBench	Avg
	Deepseek-R1-Distill-Qwen-32B								
Token (Greedy)	66.66	50.00	92.20	85.00	60.10	87.20	88.71	42.65	71.57
Token (Sampling)	72.08	55.63	94.50	95.46	60.60	97.25	95.13	57.35	78.50
Soft (Vanilla)	62.00	49.17	91.60	90.00	60.10	86.41	87.93	44.80	72.13
	QwQ-32B								
Token (Greedy)	80.00	70.00	97.00	100.00	64.14	95.12	96.10	58.78	82.64
Token (Sampling)	77.92	67.50	96.20	97.50	62.63	98.17	96.89	62.00	82.35
Soft (Vanilla)	76.67	62.29	96.20	98.75	59.60	93.90	95.33	57.71	80.06
	Skywork-OR1-32B								
Token (Greedy)	76.67	73.33	95.80	90.00	56.06	81.71	86.38	54.84	76.85
Token (Sampling)	78.75	71.25	96.40	98.28	62.62	96.95	97.28	62.37	82.99
Soft (Vanilla)	79.16	69.38	96.00	97.97	59.60	85.37	90.66	55.56	79.21

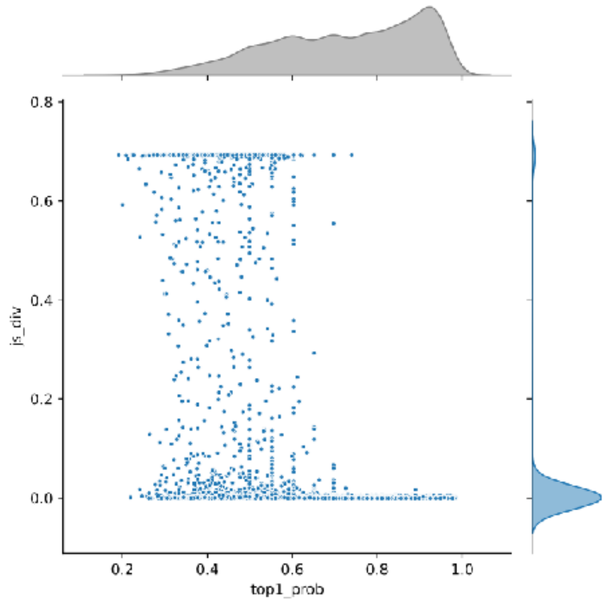
Hypothesis: LLMs are Single-Threaded Reasoners

Analysis

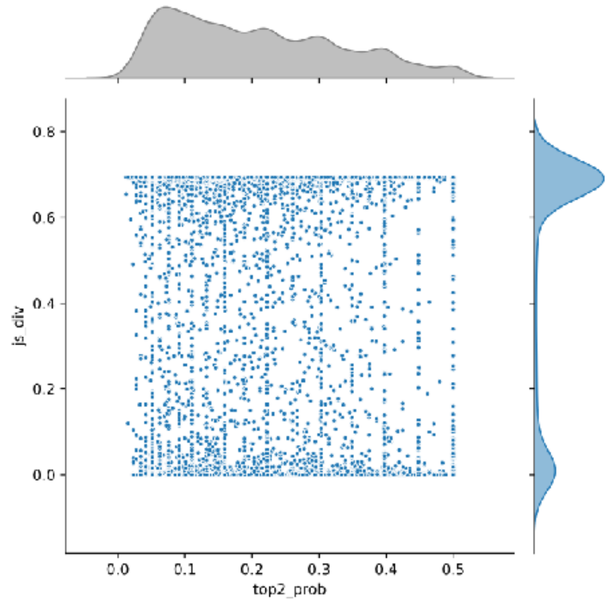
- Model's output probabilities are quite similar between using Soft Token and using the token with highest probability



(a) Soft Token entropy



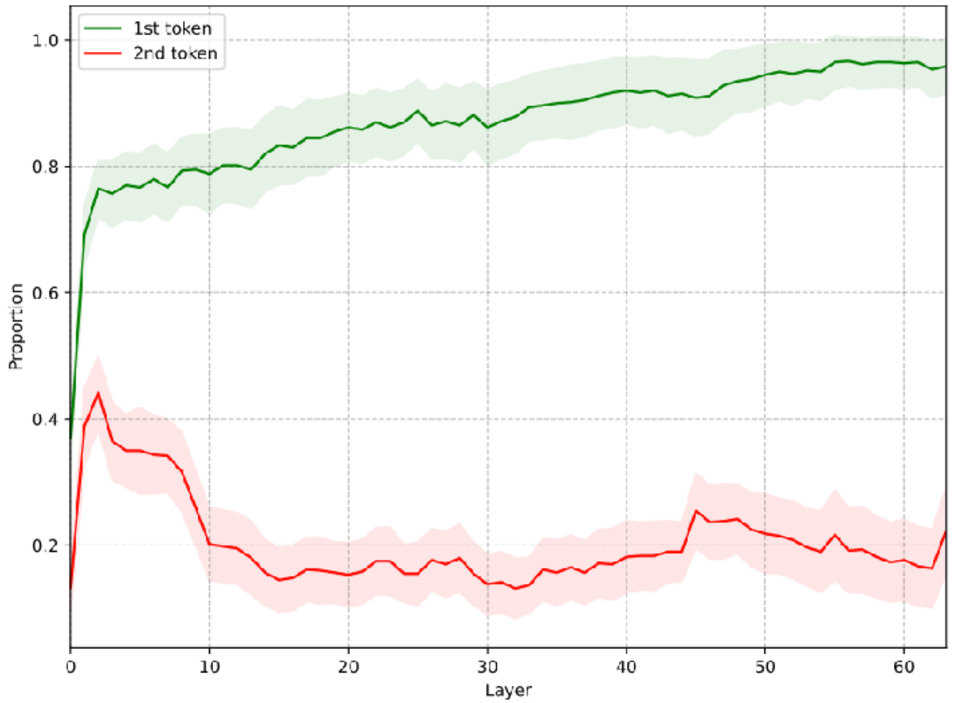
(b) 1st Token



(c) 2nd Token

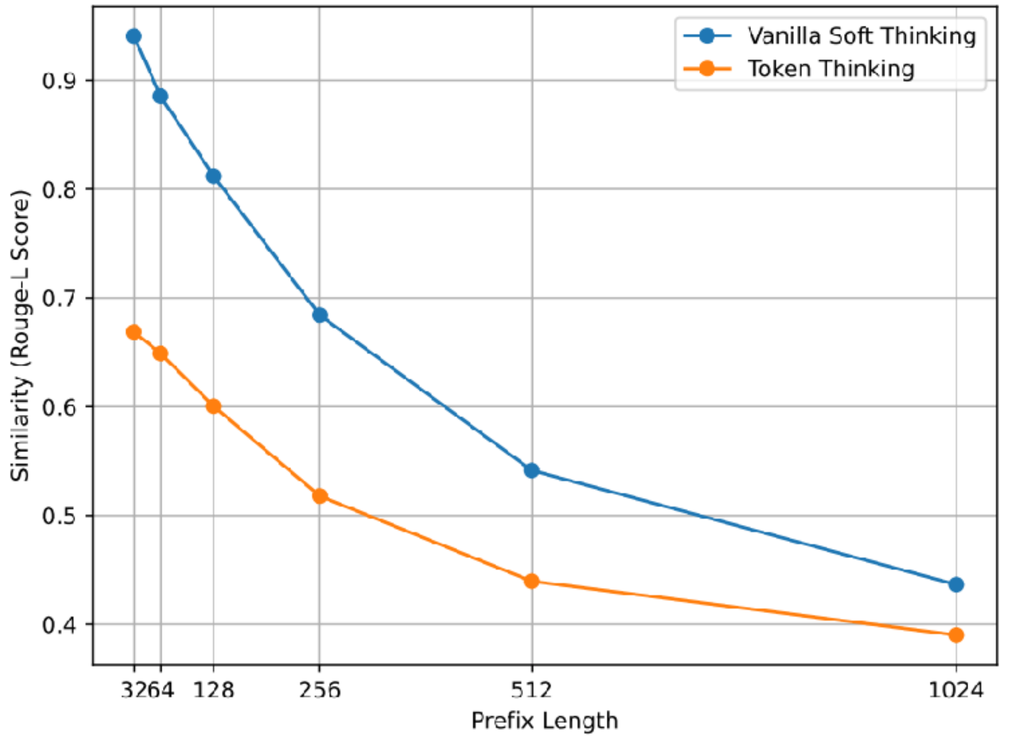
Analysis

- As layer deepening, the overlap top-probability tokens between using Soft Token and the token with highest probability is increasing



Analysis

- Sequence similarity between Soft Thinking and Greedy Token Thinking is higher than Token Thinking and Greedy Token Thinking



Conclusion

- The model is more likely a greedy reasoner when using Soft Thinking
- Although softly aggregating over vocabulary, the model mostly relies on the token with highest potability during inference

Solution

- We also need some randomness in Soft Thinking
- Dirichlet Sampling:

$$f(x_1, ..., x_n; \alpha_1, ...\alpha_n) = \frac{1}{\mathrm{B}(\alpha)} \prod_{i=1}^{n} x_i^{\alpha_i - 1}$$

- Gumbel-Softmax Trick:

$$y_i = \frac{\exp\left((g_i + \log(\pi_i))/\tau\right)}{\sum_{k=1}^{n} \exp\left((g_k + \log(\pi_k))/\tau\right)}$$

Experiments

- Soft Thinking is improved when introducing randomness

	AIME24	AIME25	MATH500	AMC23	GPQA-Diamond	HumanEval	MBPP	LiveCodeBench	Avg
	Deepseek-R1-Distill-Qwen-32B								
Token (Sampling)	72.08	55.63	94.50	95.46	60.60	97.25	95.13	57.35	78.50
Soft (Vanilla)	62.00	49.17	91.60	90.00	60.10	86.41	87.93	44.80	72.13
Soft (Dirichlet)	69.79	54.58	94.60	94.53	62.12	98.17	95.72	57.35	78.36
Soft (Gumbel)	72.92	55.42	96.00	95.62	63.13	98.17	95.64	59.50	79.55
	QwQ-32B								
Token (Sampling)	77.92	67.5	96.20	97.5	62.63	98.17	96.89	62.00	82.35
Soft (Vanilla)	76.67	62.29	96.20	98.75	59.60	93.90	95.33	57.71	80.06
Soft (Dirichlet)	76.67	68.13	96.60	96.56	61.62	96.34	95.72	59.50	81.39
Soft (Gumbel)	78.96	68.95	97.20	98.28	67.67	97.56	97.66	62.72	83.04
	Skywork-OR1-32B								
Token (Sampling)	78.75	71.25	96.40	98.28	62.62	96.95	97.28	62.37	82.99
Soft (Vanilla)	79.16	69.38	96.00	97.97	59.60	85.37	90.66	55.56	79.21
Soft (Dirichlet)	78.96	71.25	96.20	97.50	66.16	96.34	97.28	61.29	83.12
Soft (Gumbel)	79.79	73.75	97.40	98.59	67.67	97.56	98.05	64.16	83.41

Thanks for Listening!