STOIC REASONER: Dual-Mode Transformers that Compress to Think and Decompress to Speak

Liu Yang*
University of Wisconsin-Madison

Angeliki Giannou*
University of Wisconsin-Madison

Kangwook Lee
University of Wisconsin-Madison

Robert Nowak
University of Wisconsin-Madison

Dimitris Papailiopoulos
University of Wisconsin-Madison

Abstract

Latent reasoning has emerged as an alternative to reasoning with natural language and involves feeding back the last layer's representation (soft token) to the input of the transformer. This idea is promising, since soft tokens have higher representation capacity compared to quantized vocabulary elements, *i.e.* hard tokens. Existing works on training transformers with soft tokens often suffer from performance loss, while in some cases sampling diverse outputs from the model can be challenging. We propose a training paradigm called STOIC REASONER (Soft TOken Implicit Context REASONER) for transformers that uses soft tokens, in which the model learns to operate in two modes; one that processes the soft tokens (latent thinking mode) and one that decompresses the soft tokens into few reasoning steps with hard tokens from the vocabulary (local decoding mode). We focus on logical and math reasoning tasks, and fine-tune pretrained models of different size. Our method achieves similar or better performance, compared to supervised fine-tuning with Chain-of-Thought data across all tasks; while it requires reduced KV cache and allows sampling different reasoning traces at inference.

1 Introduction

Reasoning models often rely on increased test-time compute through methods like Chain-of-Thought (CoT) prompting [Wei et al., 2022]; since they have been either trained on or guided by CoT data, which encourages them to generate intermediate reasoning steps in order to reach an answer. Even though CoT data have been shown to improve performance on reasoning tasks, one potential downside is that models are constrained to reason only using tokens of the vocabulary (hard tokens). Projecting the hidden states of a model to the vocabulary, can be thought as a discretization step, which potentially leads to information loss. Reasoning in the latent space has emerged as an alternative, in which the hidden states are not always realized as hard tokens, potentially leading to more expressive and thus shorter reasoning chains.

One particular approach involves the use of so-called soft tokens, which are not part of the discrete vocabulary of hard tokens. In this work, we focus primarily on soft tokens that are represented by the last-layer hidden states. Hao et al. [2024] took initial steps in this direction, proposing a training algorithm that gradually replaces CoT steps with soft tokens, without explicitly training them. Later,

*Denotes equal contribution

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: MATH-AI.

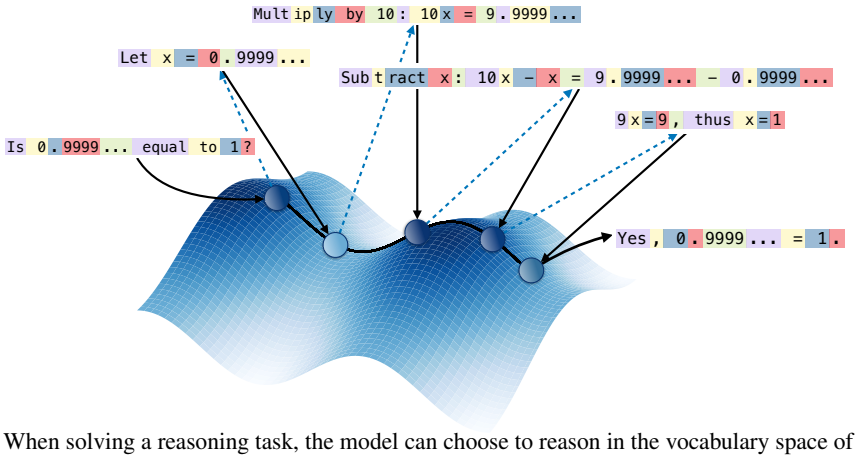

Figure 1: When solving a reasoning task, the model can choose to reason in the vocabulary space of hard (top), or with soft tokens in the latent space (bottom) and traverse between the two by processing the soft tokens, or decompressing the soft tokens into the CoT steps. This can be viewed as an interpolation between hard token and soft token methods.

Shen et al. [2025a] introduced an auxiliary loss to distill hidden representations from a teacher model trained on explicit CoTs. More recently, Hwang et al. [2025] proposed a three-stage training method: (i) fine-tuning a pretrained model on the target task, (ii) training an encoder–decoder to compress and decompress CoT steps into latent representations, and (iii) training a latent model guided by the learned encoder–decoder.

These methods often result in performance loss, compared to explicit CoT training or require training more than one models. More importantly, models that reason only in the latent space produce deterministic trajectories, and it is unclear how sampling should be implemented at inference. This conflicts with post-training algorithms like GRPO [Liu et al., 2024] that require exploring multiple candidate reasoning paths. Our approach, STOIC REASONER (Soft TOken Implicit Context REASONER), mitigates these concerns by training *one* transformer model that performs in a dual nature; *the latent thinking mode*, in which the model learns to generate the next soft token and *the local decoding mode*, in which the model learns to decode the soft tokens into CoT step(s). The model can also choose to update the soft tokens, using the representation generated in the local decoding mode.

2 Related Work

Latent Reasoning via Tokens. The idea of augmenting LLM's capabilities, by providing special tokens of the vocabulary in the input, has been shown to improve reasoning with minimal training or architectural changes [Goyal et al., 2024, Pfau et al., 2024, Sun et al., 2025]. Another line of work studies soft tokens–continuous embeddings not tied to the discrete vocabulary. Soft tokens can be constructed in multiple ways; one is by superposition over input embeddings [Xiong et al., 2024, Zhang et al., 2025, Wu et al., 2025, Gozeten et al., 2025], or (ii) by using last-layer hidden states [Hao et al., 2024, Tack et al., 2025, Zhu et al., 2025, Shen et al., 2025b, Hwang et al., 2025]. Our work is closer to the second approach, which feeds the model's final-layer hidden states back into the input. Beyond these, several methods leverage learned soft embeddings from other sources like intermediate hidden representations of a frozen LM [Cheng and Durme, 2024], or VQ-VAE to discretize and compress CoT embeddings into soft tokens [Su et al., 2025].

Transformers with Memory. Relevant to our work is the literature on augmenting transformers with some form of memory of the past context. The work of Dai et al. [2019] combines past hidden representations through the attention mechanism. Bulatov et al. [2022, 2023] introduce recurrent memory transformer (RMT) which use the last-layer hidden representations, append them in the beginning, and end of each segment and iteratively update them. Chevalier et al. [2023] fine-tune pretrained models by using adaptive instead of a constant number of memory tokens and show benefits for long context length.

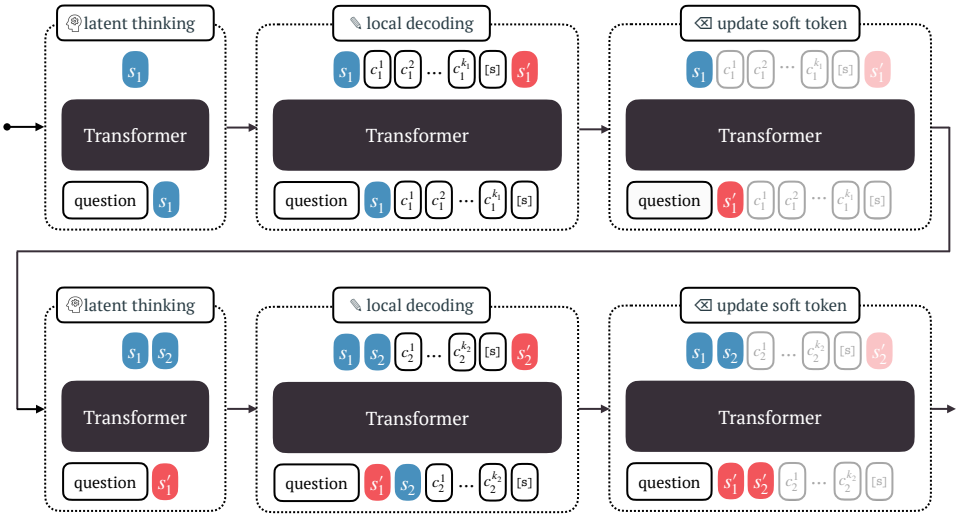

Figure 2: *Training framework of our method.* We train a *single* transformer to operate in two modes. In the *latent thinking* mode, given the question and compressed context, the model predicts a new soft token. In the *local decoding* mode, it decompresses the soft token into a few Chain-of-Thought steps until a `[switch]` token is produced, after which the model choose to update the soft token using the representation at `[switch]`.

3 Our Method

In this section, we describe our training algorithm applied to reasoning tasks. In the Appendix A.2 we provide more details.

Details of STOIC REASONER. In Figure 2 we provide a depiction of how STOIC REASONER works. In detail, given CoT training data, in which the steps are explicitly defined, we split them (together with the label) into k chunks, where k is the maximum number of soft tokens used. Each chunk is either empty, contains one, or more steps of CoTs. At the end of each chunk we append a special `[switch]` token. We start with the latent thinking mode, given a question q, the first soft token is produced by passing in the model the question and keeping the last layer's hidden representation (s_1) corresponding to the last token. Afterwards, we enter the local decoding mode in which the first chunk of data is fed to the model together with the question and the first soft token. The model predicts some labels which we keep to apply loss. The last layer's hidden representation of the `[switch]` (noted as s_1' in Figure 2) is used to update the soft token. The steps are repeated until all the chunks have been used.

Enhancing soft tokens. In order to enhance reasoning through soft tokens and learn more robust representations by allowing the model to sometimes skip the local decoding step, we introduce a stochastic soft token updating rule to our method. Specifically, during training with probability p_{update}, we update the soft token with the last layer's hidden representation of the `[switch]` (s_1') and with probability $1 - p_{\text{update}}$, we do not update it (keep s_1). At inference the user can pick a new probability p_{update}, and enable or disable the generation of hard tokens to speed up inference.

Scheduling on soft tokens. To improve the training stability and learning curve, we apply a curriculum on the number of soft tokens. We start the training with fewer number of soft tokens to keep training closer to supervised fine-tuning with explicit CoTs, in which case one soft token needs to encode multiple CoT steps. As the training progresses we gradually increase the soft tokens used, until we reach the maximum number of soft tokens that we want the model to adapt to. The training steps for each stage of the scheduling are decided as a percentage of the total number of steps. We discuss the effects of scheduling in Appendix A.4.

Remark 1. STOIC REASONER is closely related to Autocompressor [Chevalier et al., 2023] which primarily extends effective context length by updating learned memory tokens and prepending them to subsequent segments. Our approach came from a different perspective: rather than increasing the retrievable context length, we study how such latent states can compress CoT traces while *preserving*

accuracy and reducing KV-cache. Furthermore, the update probability p_{update} interpolates between memory-augmented processing and purely latent soft-token computation.

4 Experiments

We evaluate STOIC REASONER by fine-tuning pretrained language models on mathematical and logical reasoning benchmarks. The detailed setup is in Appendix A.3.

Performance on Reasoning Tasks by Different Models. The results on different reasoning tasks and different models are reported in Table 1. On **GSM8k**, for GPT2-small, STOIC REASONER outperforms the CoT baseline and the other soft token approaches. On **ProntoQA**, our approach matches CoT at 100%. On **ProsQA**, latent/soft token methods generally beat CoT, likely because the task benefits from exploring alternatives without committing to a single path. Mirroring this behavior by using $p_{\text{update}} < 1$, our method achieves significantly better performance compared to CoT and competitive with prior soft token approaches.

Model	**GSM8k**	**ProsQA**	**ProntoQA**
CoT	44.73	84.6	**100**
iCoT	30.0	**98.2**	-
Coconut	34.1	97.0	99.8
CODI	42.9	-	-
SbS	40.3	92.6	-
STOIC	**47.84**	94.6	**100**

Model	CoT	Ours
GPT2-small	44.73	**47.84**
Gemma3-270M	**48.75**	47.92
Qwen2.5-0.5B	58.22	**58.45**

Table 1: *(left)* Performance of GPT2-small on GSM8k, ProntoQA and ProsQA with different soft token/latent methods, and *(right)* Test accuracy on GSM8k for GPT2-small, Gemma3-270M and Qwen2.5-0.5B. Note that we did not re-implement these methods except for SFT on CoTs; we report the results that each paper reported (iCOT: Deng et al. [2024], Coconut: Hao et al. [2024], CODI: Shen et al. [2025a] and SbS Hwang et al. [2025]).

Task Difficulty Generalization. We further evaluate on iGSM [Ye et al., 2024], a GSM8K-style synthetic benchmark that controls the number of arithmetic operations (*op*) required to solve each problem. We fine-tune a pretrained GPT2-small model on iGSM-Easy ($op \leq 9$) and iGSM-Med ($op \leq 15$) datasets, and then test on out-of-distribution (OOD) data. As shown in Figure 3, our method matches the CoT baseline on in-distribution problems and consistently retains higher accuracy on OOD problems; moreover, the accuracy decays more slowly as *op* increases.

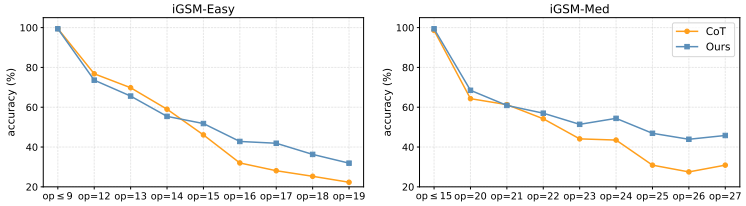

Figure 3: *In-Distribution and OOD Performance of iGSM Easy and Medium Dataset.* Across both settings, STOIC REASONER retains higher accuracy than CoT under OOD shifts with longer reasoning chains and degrades in a slower rate as the number of operations increases.

Test-time Performance. As mentioned earlier, our method naturally supports sampling via local decoding and the update of the soft token accordingly; different traces in local decoding imply different updated soft token.

More results on different temperature are shown in Appendix A.4.

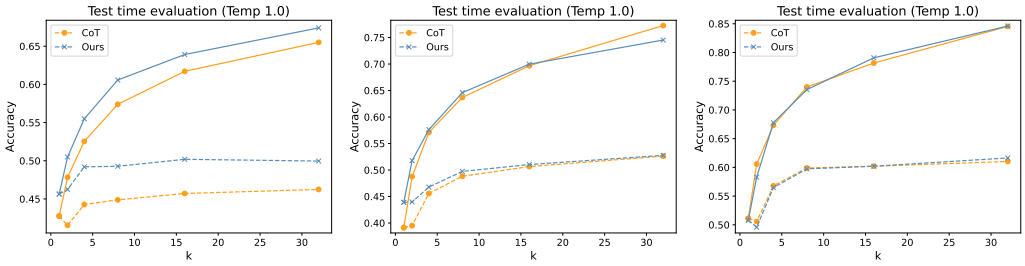

Figure 4: *Pass@K (solid line) and majority-vote (dashed line) results on GSM8K for GPT2-small (left) Gemma3-270m (middle) and Qwen2.5-0.5B (right).* Our method shows monotonic gains as k increases, following the same trend as the CoT baseline. Temperature is set at 1 for the top and 1.5 for the bottom.

Effect of the soft token update probability. We study the effect of p_{update} on ProsQA, where soft token methods outperform the explicit CoT. We sweep $p_{\text{update}} \in \{0, 0.2, 0.5, 0.7, 1\}$ and the results can be seen in Table 2.

Probability of updating	**0.0%**	**20%**	**50%**	**70%**	**100%**
Always decoding	0.0	91.0	91.6	87.4	82.0
Probabilistic decoding	83.0	91.6	**94.6**	89.6	72.0
Latent only	90.6	89.0	94.0	93.4	0.0

Table 2: *Performance on ProsQA of GPT2-small with variable soft token update probability p_{update} (columns) and across test-time strategies (rows).* "Always decoding" always performs local decoding; "Probabilistic decoding" performs local decoding with probability p_{update} matching the column; and "Latent only" uses only soft token updates.

References

Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. Recurrent memory transformer. Advances in Neural Information Processing Systems, 35:11079–11091, 2022.

Aydar Bulatov, Yuri Kuratov, Yermek Kapushev, and Mikhail S Burtsev. Scaling transformer to 1m tokens and beyond with rmt. arXiv preprint arXiv:2304.11062, 2023.

Jeffrey Cheng and Benjamin Van Durme. Compressed Chain of Thought: Efficient Reasoning Through Dense Representations, December 2024. URL http://arxiv.org/abs/2412.13171. arXiv:2412.13171 [cs].

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. arXiv preprint arXiv:2305.14788, 2023.

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.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. arXiv preprint arXiv:1901.02860, 2019.

Yuntian Deng, Yejin Choi, and Stuart Shieber. From Explicit CoT to Implicit CoT: Learning to Internalize CoT Step by Step, May 2024. URL http://arxiv.org/abs/2405.14838. arXiv:2405.14838 [cs].

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens, 2024. URL http://arxiv.org/abs/2310.02226.

Halil Alperen Gozeten, M. Emrullah Ildiz, Xuechen Zhang, Hrayr Harutyunyan, Ankit Singh Rawat, and Samet Oymak. Continuous chain of thought enables parallel exploration and reasoning, 2025. URL http://arxiv.org/abs/2505.23648.

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, December 2024. URL http://arxiv.org/abs/2412.06769. arXiv:2412.06769 [cs].

Hyeonbin Hwang, Byeongguk Jeon, Seungone Kim, Jiyeon Kim, Hoyeon Chang, Sohee Yang, Seungpil Won, Dohaeng Lee, Youbin Ahn, and Minjoon Seo. Let's predict sentence by sentence, 2025. URL http://arxiv.org/abs/2505.22202. version: 1.

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

Jacob Pfau, William Merrill, and Samuel R. Bowman. Let's think dot by dot: Hidden computation in transformer language models, 2024. URL http://arxiv.org/abs/2404.15758.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. OpenAI blog, 1(8):9, 2019.

Abulhair Saparov and He He. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. arXiv preprint arXiv:2210.01240, 2022.

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

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. CODI: Compressing chain-of-thought into continuous space via self-distillation, 2025b. URL http://arxiv.org/abs/2502.21074. version: 1.

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

Yuchang Sun, Yanxi Chen, Yaliang Li, and Bolin Ding. Enhancing latent computation in transformers with latent tokens, 2025. URL http://arxiv.org/abs/2505.12629.

Jihoon Tack, Jack Lanchantin, Jane Yu, Andrew Cohen, Ilia Kulikov, Janice Lan, Shibo Hao, Yuandong Tian, Jason Weston, and Xian Li. LLM Pretraining with Continuous Concepts, February 2025. URL http://arxiv.org/abs/2502.08524. arXiv:2502.08524 [cs].

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. arXiv preprint arXiv:2503.19786, 2025.

Qwen Team. Qwen2 technical report. arXiv preprint arXiv:2407.10671, 2, 2024.

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.

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

Zheyang Xiong, Ziyang Cai, John Cooper, Albert Ge, Vasilis Papageorgiou, Zack Sifakis, Angeliki Giannou, Ziqian Lin, Liu Yang, Saurabh Agarwal, et al. Everything everywhere all at once: Llms can in-context learn multiple tasks in superposition. arXiv preprint arXiv:2410.05603, 2024.

Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. Physics of language models: Part 2.1, grade-school math and the hidden reasoning process. arXiv preprint arXiv:2407.20311, 2024.

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, Shuohang Wang, Yelong Shen, and Xin Eric Wang. Soft thinking: Unlocking the reasoning potential of LLMs in continuous concept space, 2025. URL http://arxiv.org/abs/2505.15778.

Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart Russell, and Yuandong Tian. Reasoning by Superposition: A Theoretical Perspective on Chain of Continuous Thought, May 2025. URL http://arxiv.org/abs/2505.12514. arXiv:2505.12514 [cs] version: 1.

A Appendix

A.1 Dataset Details

For GSM8K, ProntoQA, and ProsQA, we adopt the preprocessing and train/validation/test splits of Hao et al. [2024]. For iGSM, we generate the dataset using the official open-source implementation [2]. We record the data split in Table 3, and the max number of CoT steps in each dataset in Table 4. For the OOD dataset in iGSM, the number of CoT step is the number of the operation op, so the iGSM with $op = 20$ contains 20 CoT steps.

Table 3: Dataset splits.

Dataset	Training	Validation	Test	OOD
GSM8k	385,620	500	1,319	-
ProntoQA	9,000	200	800	-
ProsQA	17,886	300	500	-
iGSM-med/easy	1,498,500	500	1,000	1,000

Table 4: Statistics of Dataset.

Dataset	Training	Validation	Test	`Example CoT step`
GSM8k	13	8	8	`«12+3=15»`
ProntoQA	11	11	11	`Each yumpus is a wumpus.`
ProsQA	6	6	6	`Every hilpus is a numpus.`
iGSM-med	15	13	13	`Define Niagara Falls Aviary's Enclosure as y; so y = b = 20.`
iGSM-easy	9	9	9	`Define Goat Cheese's Rye as S; so S = 3.`

A.2 Details on Our Method

Reduced KV-Cache at Inference. If the probability of updating the soft token is set to be $p_{\text{update}} \in [0, 1]$, meaning we always generate the hard tokens, then our method for inference uses KV-cache which scales as $q + k + L$, where q is the size of the length of question, k is the number of soft tokens used, L is the maximum length of any generation. The inference FLOPs are scaling as

$$\sum_{l=1}^{q} l + \sum_{i=1}^{k}[(q+i) + p_{\text{update}} \sum_{j=1}^{L}(q+i+j)] = \mathcal{O}(q^2 + k^2 + p_{\text{update}} kL(k+L+q)$$

Similarly, for the standard model that is fine-tuned in the CoTs the KV-cache scales as $q + KL$, while the FLOPs scale as

$$\sum_{l=1}^{q} l + \sum_{i=1}^{kL}(q+i) = \mathcal{O}(q^2 + qkL + k^2L^2)$$

Algorithm for training and inference. Here we present the detailed training and inference algorithm in Algorithm 1 and 2.

[2] `https://github.com/facebookresearch/iGSM`

Algorithm 1 Training

Input: data point (q, y), Transformer TF_θ, probability p_{update}.
Choose k {number of soft tokens}
$\{x_j\}_{j=1}^{k} \leftarrow \text{RANDOMPARTITION}(y, k)$ {split y (CoTs + labels) into k parts}
$h_0 \leftarrow TF_\theta(q)[-1]$
for $j = 1, \ldots, k$ **do**
  $z_j \leftarrow TF_\theta\big([q, h_{0:j-1}, x_j, \texttt{[switch]}]\big)$
  if $\mathcal{U}(0,1) < p_{\text{update}}$ **then**
    $h_{j-1} \leftarrow z_j[-1]$ {update soft token}
  end if
  $t_j \leftarrow \text{PROJ}_{\text{vocab}}(z_j[:-1])$
  $h_j \leftarrow TF_\theta\big([q, h_{0:j-1}]\big)[-1]$
end for
$\mathcal{L} \leftarrow \text{LOSS}\big(\{t_j\}_{j=1}^{k}, y\big)$

Algorithm 2 Inference

Input: q, TF_θ, p_{update}, max_k.
$i \leftarrow 0$, $h_0 \leftarrow TF_\theta(q)[-1]$
while $i < max_k$ and $t \neq \texttt{[eos]}$ **do**
  $t \leftarrow \text{PROJ}_{\text{vocab}}(TF_\theta([q, h_{0:i-1}])[-1])$
  if $\mathcal{U}(0,1) < p_{\text{update}}$ and $t \neq \texttt{[ans]}$ **then**
    T $\leftarrow []$
    while $t \neq \texttt{[switch]}$ **do**
      T.append(t)
      $z \leftarrow TF_\theta([q, h_{0:i}, \text{T}])[-1]$
      $t = \text{PROJ}_{\text{vocab}}(z)$
    end while
    $h_{i-1} \leftarrow z[-1]$ {update soft token}
  end if
  $h_i \leftarrow TF_\theta\big([q, h_{0:i-1}]\big)[-1]$
  $i \leftarrow i + 1$
end while

A.3 Experimental Setup

We fine-tune GPT2-small (124M) [Radford et al., 2019], Gemma3 (270M) [Team et al., 2025], and Qwen2.5 (0.5B) [Team, 2024] on four datasets: GSM8k [Cobbe et al., 2021], iGSM [Ye et al., 2024], ProsQA [Hao et al., 2024], and ProntoQA [Saparov and He, 2022]. For GSM8k we adopt the augmented split of Deng et al. [2024]; for ProsQA we follow Hao et al. [2024]. For iGSM we use two settings: *medium* with maximum number of operations $op \leq 15$ and out-of-distribution (OOD) evaluation at $op = 20, \cdots, 27$, and *easy* with $op \leq 9$ and OOD at $op = 12, \cdots, 19$.

Our main baseline is supervised fine-tuning (SFT) on explicit Chain-of-Thought traces. To ensure a fair comparison, we use identical hyperparameters for our method and the SFT baseline: we apply learning rate 2e-4 when finetuning GPT2 and 5e-5 when finetuning Gemma3 or Qwen2.5 models. On GSM8k we finetune for 15 epochs for GPT2, and 5 epochs for Gemma3 and Qwen2.5, while for ProsQA and ProntoQA we finetune for 60 epochs. We apply weight decay 0.01, cosine annealing learning rate scheduler, and effective batch size 32.

Regarding the hyperparameter that is specific to our method: Let p_{update} (Section 3) denote the probability of updating the soft tokens during training; we set $p_{\text{update}} = 0.5$ for ProsQA and $p_{\text{update}} = 1$ elsewhere. Soft token scheduling is enabled by default, where the max number of soft token is set such that for the given task, each soft token encode at most 1 CoT step.

A.4 Additional Experimental Results

Effects of number of soft tokens. In GSM8k, we set the max number of soft tokens to be $k_{\max} = 12$, since most GSM8k CoT traces contain ≤ 10 steps, this allows (when possible) for at most one soft token per step. To study the effects of a smaller soft token budget, we also evaluate $k_{\max} \in \{4, 6\}$, where some soft tokens must summarize more than one step on average. We report the performance of different numbers of soft tokens in Table 5. As shown in the table, encoding fewer steps per token improves reasoning performance, and one-per-step suffices in matching and surpassing the performance with CoT.

	CoT	**4 soft tokens**	**6 soft tokens**	**12 soft tokens**
GPT2-GSM8k	44.73	42.68	44.2	**47.84**

Table 5: *Performance of GPT2-small on GSM8k for different number of maximum soft tokens with scheduling and probability* $p_{update} = 1.0$.

Effects of scheduling. We train end-to-end to jointly learning the target task and generation of soft tokens. Using fewer soft tokens keeps training closer to the explicit-CoT baseline, but restricts

the amount of time in which the model is using the maximum number of soft tokens. We therefore study how different schedules during training, affect the performance of the model. Specifically, we increase k by one every $N\%$ of total steps until reaching $k_{\max}$; for the remaining steps of training we use $k_{\max}$. As shown in Table 6 , an interval of about 6% gives the best GSM8k accuracy, balancing task learning and soft token learning; smaller or larger intervals still lead to performance close (or even better) to the explicit CoT baseline. Following this rule, we use a scheduling interval of $\approx 6\%$ for both Qwen2.5 and Gemma3.

Percentage of total steps	**0.0%**	**4.4%**	**5.6%**	**6.7%**	**8.3%**
GPT2-GSM8k	43.44	43.9	**47.84**	46.32	47.38

Table 6: *Performance of GPT2-small on GSM8k for various scheduling schemes with max soft token budget* $k_{\max} = 12$. The percentage of the steps corresponds to the amount of steps performed for increasing the number of soft tokens by one. The remaining steps are performed with keeping the number of soft tokens to be the maximum. The first column (0.0%) corresponds to using the maximum number of soft tokens from the beginning.

Test-time performance. We evaluate `pass@K` and majority voting against the baseline. As shown in Figure 5, with GPT2-small trained on the GSM8k dataset, our method exhibits the same monotonic increase of accuracy, with increasing k in both `pass@K` and majority voting, indicating it can be applied to post-training on reasoning tasks. Interestingly, even though our method is slightly worse on Gemma3 under greedy decoding (Table 1, right), with temperature sampling it performs well and is more robust to higher temperatures: when T increases from 1.0 to 1.5, our method maintains similar performance, whereas the CoT baseline drops.

Soft token update probability for GSM8k. In the main text we examined p_{update} on ProsQA, where skipping local decoding (*i.e.,* performing a latent update instead) helps the model learn useful soft token representations. Here we extend the analysis to GSM8k. We train with $p_{\text{update}} = 0.5$ and vary p_{update} at inference. As shown in Table 7, GSM8k accuracy drops whenever $p_{\text{update}} > 0$, indicating that inserting latent-only steps at test time hurts performance relative to always decoding. We hypothesize this effect is task-dependent: tasks like ProsQA benefit from encoding parallel reasoning traces in the latent space, so using p_{update} during training and inference can help; in contrast, GSM8k appears to benefit from grounding intermediate steps in hard tokens, so nonzero p_{update} reduces accuracy, however with the trade-off of generating fewer hard tokens.

Probability of updating	**0.0%**	**20%**	**50%**	**70%**	**100%**
GPT2-GSM8k	25.24	31.31	37.91	40.79	44.807

Table 7: *Performance of GPT2-small on GSM8k for variable soft token update probability.*

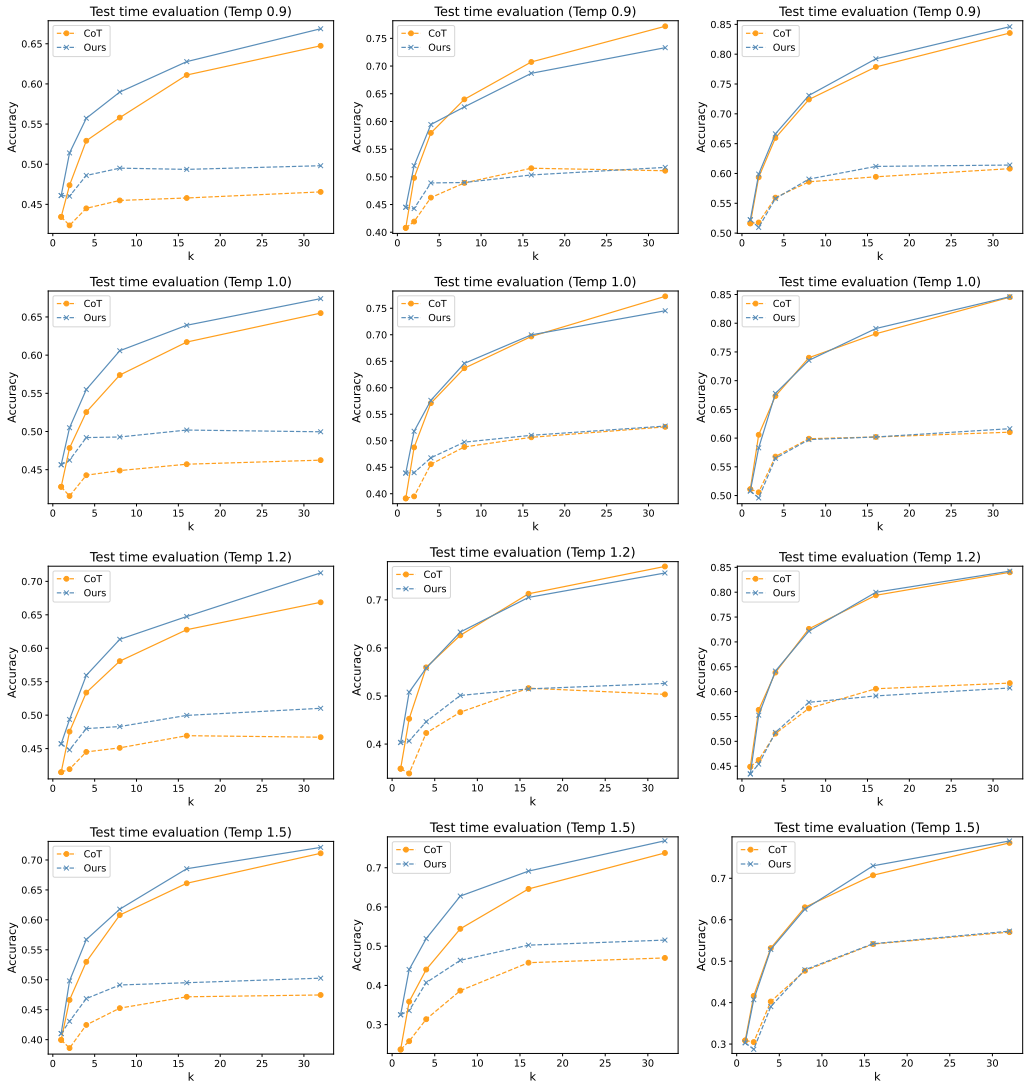

Figure 5: Additional results for GPT2 small, Gemma3-270m and Qwen2.5-0.5B on majority voting and `pass@K`, using different temperatures. Both methods follow a similar curve.