SELF-PROMPTED LATENT REASONERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Language Models (LLMs) reason effectively when intermediate steps are written out as chain-of-thought (CoT) text, and latent reasoning models further move the explicit, token-level processes into internal computations in latent spaces before explicit output. For frozen LLMs, latent input tuning methods train a lightweight module that inserts continuous vectors after the task question, which improves the model's task-specific understanding and stability while avoiding catastrophic forgetting. However, the vectors produced by existing methods carry mostly an adaptation to the task rather than question-specific information, as replacing them with their average over questions leaves accuracy essentially unchanged, and these methods consequently perform poorly on tasks that require composing several reasoning steps. To address this issue, we propose self-prompted latent reasoners, in which a minimal mapper transforms internal representations of the intermediate layer of a frozen LLM and feeds them back to its input embeddings. The resulting signal is input-dependent and helps the model to read its own unfinished, richer thoughts, while it requires no auxiliary model and no additional input positions. Experiments on multi-step reasoning benchmarks show that the proposed method substantially outperforms prompt tuning and existing latent reasoning methods under a matched protocol, while remaining on par with them on common arithmetic benchmarks.

1 INTRODUCTION

Large Language Models (LLMs) have made rapid progress in reasoning, and chain-of-thought (CoT) prompting, which elicits intermediate steps before the answer, has become a central mechanism behind it (Wei et al., 2022; Wang et al., 2023; Muennighoff et al., 2025). Explicit CoT, however, reasons through discrete tokens, so every step must be verbalized and an error made early propagates to the answer. Latent reasoning instead moves part of this computation into continuous representations, letting the model process intermediate states before it writes an answer (Hao et al., 2025; Shen et al., 2025; Zhu et al., 2025a). Methods such as Coconut (Hao et al., 2025) and CODI (Shen et al., 2025) achieve this by fine-tuning the whole model, which is costly and can erode, through catastrophic forgetting, the abilities an instruction-tuned model already has (Kalajdzievski, 2024; Lobo et al., 2025).

Latent input tuning avoids this by keeping the LLM frozen and training only a lightweight module that inserts continuous vectors after the question. Building on prompt tuning (Lester et al., 2021; Li & Liang, 2021), SoftCoT (Xu et al., 2025a) produces these vectors with a frozen assistant model and a trained projection, EBM-CoT (Chen et al., 2025) further calibrates them with an energy model, and Latent Recurrent Thoughts (LRT) (Chen & Fu, 2026) obtains them from a task-dedicated proposer and a recurrent refiner. Such vectors are meant to carry information about the particular question, yet their actual influence on the frozen LLM has rarely been tested directly. When the vectors of every question are replaced with their average over all questions, the accuracy of SoftCoT and EBM-CoT is essentially unchanged, and a learned prompt shared by all questions reaches the same or higher accuracy. Their vectors therefore encode an adaptation to the task rather than to the question, which answer-only supervision can reward when question dependence is not enforced. The vectors of LRT do depend on the question, but most of their energy is still shared across questions in a single direction. Consequently, these methods perform poorly on tasks that require composing several reasoning steps, since the intermediate, unfinished conclusions such tasks depend on differ from one question to the next and cannot be supplied by a signal shared by all questions.

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

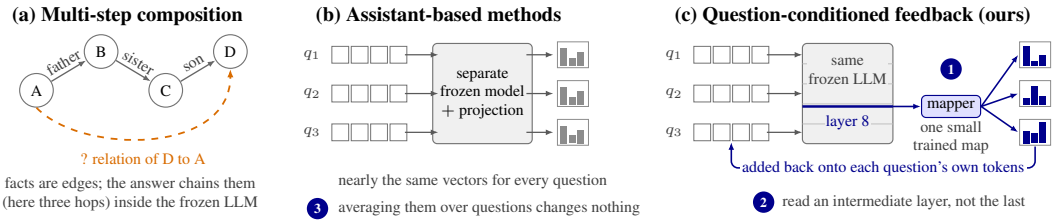

Figure 1: (a) Multi-step questions require relations to be composed inside the frozen LLM; (b) assistant-based latent methods such as SoftCoT and EBM-CoT supply nearly the same vectors for every question, (c) whereas the proposed method feeds the model's own intermediate reading of each question back onto its input.

To address this limitation, we propose self-prompted latent reasoners, in which the frozen LLM supplies the question-specific signal itself (Figure 1). Instead of relying on a separate proposer, the question is passed through the first layers of the frozen LLM, and the hidden state at each position, the model's own intermediate reading of the question, is transformed by a small trained mapper and added back onto the input embedding of the same token. The full model then reads the corrected question and generates the answer as usual. Because the correction is computed from the model's own state for the given question, it differs between questions by construction, while no auxiliary model is loaded and nothing is appended to the input. The read layer is a free choice, and an intermediate layer, whose representation of the question is contextual but not yet specialized for predicting the next token (Tenney et al., 2019; Liu et al., 2019; Skean et al., 2025), is more useful than either end of the model layers. The feedback can also be further repeated over several rounds with a different layer read in each round (Section 3).

Under a matched protocol with one frozen LLM, the proposed method obtains the highest accuracy on three multi-step reasoning datasets, clearly above prompt tuning, SoftCoT, EBM-CoT and LRT, and remains on par with these methods on two arithmetic datasets whose solutions are written out. The substitution test and a visualization on tuned input embeddings confirm that the gain depends on the question-specific part of the feedback, whose removal costs the proposed method most of its accuracy and leaves it below the frozen LLM alone.

The contributions, numbered as in Figure 1, are the following.

1. We propose a minimal method that improves a frozen LLM on multi-step reasoning with no added tokens, no auxiliary pretrained model, and only 12.9M trained parameters. The method obtains the highest accuracy among the evaluated methods on the three multi-step reasoning datasets without losing the frozen LLM's competitive edge on common arithmetic tasks.
2. We propose reading and feeding back the frozen LLM flexibly at an intermediate layer rather than at its final layer, where its representation of the question is richer.
3. We propose an average-substitution test that separates a signal specific to each question from a prompt shared by all questions. Under it the assistant-based methods are essentially unchanged, whereas the proposed method loses most of its gain, which identifies information that differs between questions as the source of the improvement.

2 Motivation

Question dependence in multi-step reasoning. When no reasoning is written out, the stated facts of a multi-step question must be composed inside the frozen LLM before the first answer token is produced (Figure 1a), and the intermediate conclusions of this composition, such as the relation between the first and the third person of a kinship chain, differ from one question to the next. An input that is identical for every question can therefore shift the output distribution of the task but cannot contribute to the composition that a particular question requires, so a latent signal intended to assist multi-step reasoning must vary with the question.

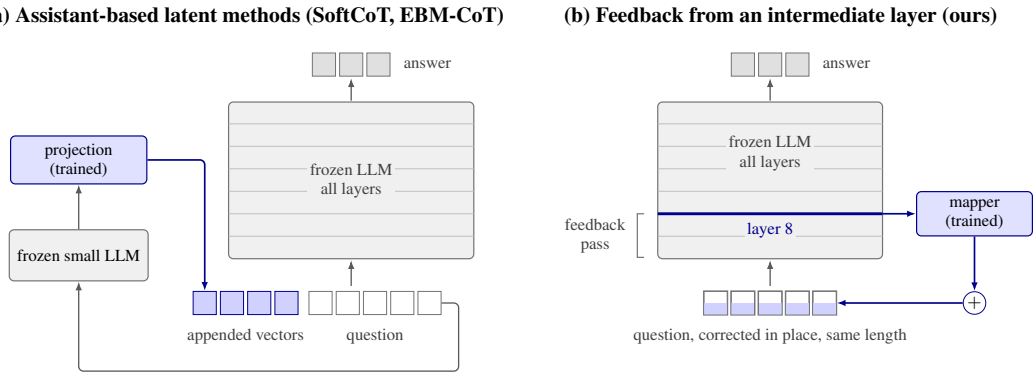

Figure 2: Assistant-based methods such as SoftCoT and EBM-CoT pass the question through a separate frozen model and a trained projection to produce vectors appended to the input (a), whereas the proposed method reads one intermediate layer of the same frozen LLM and maps it back onto the question's own input embeddings (b).

Intermediate representations as the source. A signal specific to the question can be drawn from the frozen LLM itself. Intermediate layers encode contextual and relational information that transfers across tasks, whereas the input layer represents each token in isolation and the last layers specialize in predicting the next token (Tenney et al., 2019; Liu et al., 2019; Skean et al., 2025). The intermediate state is available at the cost of a partial forward pass, lies in the representation of the model that produces the answer, and is aligned position by position with the prompt. A correction $\Delta(q)$ computed from it and added to the input embeddings $x_0 \in \mathbb{R}^{n \times d}$ of the n prompt tokens, giving $x_0 + \Delta(q)$, is therefore specific to the question by construction, preserves the sequence length, and requires no auxiliary model.

3 METHOD

Overview. The method is shown in Figure 2(b). The question is first passed through the first eight layers of a frozen language model, referred to as the host. One hidden vector is obtained per question token, which is the host's intermediate reading of the question. Each of these vectors is turned by a small trained module, the mapper, into a correction in the host's input embedding space, and the correction is added to the input embedding of the same token. The corrected question is then processed by all layers of the host to generate the answer, and the first, partial pass is referred to as the feedback pass.

Setting. A frozen decoder-only language model, the host, receives a prompt of n tokens and generates the answer. Its embedding table maps the prompt to input embeddings $x_0 \in \mathbb{R}^{n \times d}$, and its L transformer layers $f_1, \ldots, f_L$ produce hidden states $h_\ell(x) = f_\ell \circ \cdots \circ f_1(x)$ at every position, with $h_0(x) = x$. Whereas prompt tuning appends a matrix P shared by every question, giving $[x_0; P]$, and proposer-based methods append vectors $z(q)$ produced from the question q by a separate module, giving $[x_0; z(q)]$, the proposed method keeps the sequence length and adds a correction at the existing positions, giving $x_0 + \Delta(q)$, where $\Delta(q)$ is computed from the host's own reading of q.

The host. The host's weights are never updated. In all main experiments it is Qwen2.5-7B-Instruct, with $L = 28$ layers and $d = 3584$, and a 3B host is used in Appendix E. Because only x_0 is changed, one copy of the host serves every task.

Reading an intermediate layer. In the feedback pass, x_0 is run through the first λ layers of the host, and $h_\lambda(x_0)$ is kept at every prompt position, those of the chat template included. The read layer is a free choice. At $\lambda = 0$ the correction sees only each token's own embedding, and at $\lambda = L$ it sees the final states, which are specialized for predicting the next token. In between, each position carries the host's contextual reading of the question (Section 2). Layer $\lambda = 8$ of 28 is selected on a validation split (Section 5.3), and only the first λ layers are run.

The mapper. The hidden vector h at one position is converted by the mapper into a correction for the input embedding at the same position,

$$m_r(h) \;=\; g \cdot \rho\big(A\,h + b_r\big). \tag{1}$$

Here A is a $d \times d$ matrix initialized at the identity, b_r is a learned vector belonging to round r, ρ rescales each position so that its root mean square equals the average root mean square of the rows of the host's embedding table, and g is a learned scalar gate. After one round, $x_1 = x_0 + m_1\big(h_\lambda(x_0)\big)$ is read by the host. The rescaling is required as hidden states of the intermediate layers have norms two to three orders of magnitude above those of the input embeddings, so an unscaled hidden state would in effect replace the input instead of modifying it. The gate is initialized at 0.1 and the vectors b_r at small random values with standard deviation 0.02, so that training begins close to the unmodified host. In a low-rank special case, with $A = I + UV^\top$, where $U, V \in \mathbb{R}^{d\times k}$ and V is initialized at zero, the mapper keeps its identity start at a fraction of the parameters, which is used for the parameter-matched comparison in Section 5.4. Because A starts at the identity, training begins from the host's own reading of the question rather than noise.

Repeated feedback. The feedback pass can be repeated for R rounds. Round r reads layer λ_r of the state produced by the previous round and always writes its correction onto x_0,

$$x_r \;=\; x_0 + m_r\big(h_{\lambda_r}(x_{r-1})\big), \qquad r = 1, \ldots, R, \tag{2}$$

so each correction replaces, rather than accumulates on, the previous one. For training over several rounds, the number of rounds is drawn uniformly from $\{1, \ldots, 4\}$ for each example, and the gradient is passed through the last two rounds. The mapper holds one vector b_r for each of these rounds and one more, b_5, that is used by every later round and therefore keeps its random initialization. With a constant read layer $\lambda_r = \lambda$, every round from the fifth on thus applies the same map $T(x) = x_0 + m_5\big(h_\lambda(x)\big)$, and those rounds are its iterates $x_r = T(x_{r-1})$. If T is contractive, the iterates approach a fixed point $x^\star = T(x^\star)$ at which further rounds leave the input unchanged (Geiping et al., 2025; Blayney et al., 2026). The read layer can instead be cycled through a schedule, here 8, 16, 20 and 24, so that consecutive rounds apply different maps. One round is used for all results in the main tables, both in training and at evaluation, and the repeated form is analyzed in Section 5.6. Repeated feedback therefore needs no parameters beyond one vector per trained round.

Training. The mapper, consisting of the matrix A, the round vectors b_r and the gate g, is the only trained component, with 12.9M parameters against 7.6B in the host, and one mapper is trained for each task. For a prompt with embeddings x_0 and target tokens $y_1, \ldots, y_T$, the loss is the cross-entropy of the target given the corrected prompt,

$$\mathcal{L} \;=\; -\sum_{t=1}^{T} \log p_{\text{host}}\big(y_t \mid x_1,\, y_{<t}\big). \tag{3}$$

In the main protocol the target is the answer alone (Section 4), so no intermediate reasoning text is supervised or generated. The gradient reaches the mapper through all layers of the frozen host and through the first λ layers again in the feedback pass. At inference, one feedback pass through 8 of the 28 layers is run on the prompt and the answer is generated from x_1 with the ordinary key-value cache, with the cost of 1.29 forward passes in total (Table 6).

4 Experimental Setup

One protocol for every method. The same frozen host, data, supervision, greedy-decoding budget, and learning-rate grid are used for every method on a given dataset, with a rate selected per method on a held-out validation split of the training data. Because accuracy can be shifted by a small amount when published rates are suboptimal or tuning is unequal, the grid and selections are reported in Appendix A for fair comparison among methods.

Host, seeds and decoding. Qwen2.5-7B-Instruct (Qwen Team, 2024), with 7.6B parameters, is used identically. Results are means with across-seed standard deviations over seven training seeds on CLUTRR and ProsQA and five on the other datasets; prompt tuning uses five seeds, and LRT on

Table 1: Accuracy on the multi-step reasoning datasets under answer-only training.

method	CLUTRR	ProsQA	ProntoQA
Host alone	21.0	46.3	51.7
Zero-shot CoT	15.1	60.4	81.7
Prompt tuning	44.3 ± 0.9	73.9 ± 1.7	91.2 ± 6.7
SoftCoT	40.4 ± 1.2	72.9 ± 1.5	97.9 ± 0.3
EBM-CoT	38.3 ± 2.4	71.6 ± 2.4	96.2 ± 2.3
LRT	41.4 ± 2.5	63.2 ± 2.9	92.6 ± 2.8
Ours	**50.5** ± 1.9	**88.0** ± 6.7	**99.6** ± 0.2

Table 2: Accuracy on the arithmetic datasets with the bare question.

method	GSM8K	AQuA
Zero-shot CoT	84.5	80.3
Prompt tuning	79.6	61.4
SoftCoT	**79.8**	**62.4**
EBM-CoT	79.7	**62.4**
Ours	78.7	61.3

ProntoQA and the parameter-matched and smaller-host runs three. Greedy decoding is limited to 32 new tokens on CLUTRR, ProsQA and ProntoQA, enough for answers but not for written reasoning, and to 512 on GSM8K and 1024 on AQuA, where a generation stops once its first answer line is complete. Zero-shot CoT, which writes longer free-form solutions, is given 2048 tokens, or 4096 on ProsQA and ProntoQA.

Datasets. Three multi-hop reasoning datasets, in which a chain of stated facts must be composed, and two common arithmetic datasets are used for evaluation. In CLUTRR (Sinha et al., 2019), the relation between two family members is inferred by composing two to ten stated kinship relations, with training on only two to four and 1028 test items. In ProsQA (Hao et al., 2025), an entity is assigned to one of two concepts by following three to six inclusion rules, with training on three and four and 800 test items. In ProntoQA (Saparov & He, 2023), a statement is judged true or false by deduction over two to nine steps from fictional rules, with 785 test items. In GSM8K (Cobbe et al., 2021), 1319 grade-school word problems with numerical answers are solved. In AQuA (Ling et al., 2017), 254 multiple-choice algebra word problems are solved.

Baselines. The baselines are prompt tuning with 128 vectors placed after the question (Lester et al., 2021), SoftCoT with its 1.5B assistant (Xu et al., 2025a), EBM-CoT, which adds energy-based calibration to SoftCoT (Chen et al., 2025), and Latent Recurrent Thoughts (LRT) with its task-dedicated proposer and recurrent refiner (Chen & Fu, 2026), run from the official implementation with the same prompt as the other methods. The host alone and zero-shot CoT are also used for comparison. Retraining is performed because the published settings use different hosts, data, and answer formats. Configuration details are further listed in Appendix A.

5 Results

5.1 Comparison with prior work

To test whether question-conditioned feedback improves the host, we compare it with every baseline under the shared protocol of Section 4. The proposed method obtains the highest accuracy on all three multi-step reasoning datasets (Table 1), with margins over every baseline, prompt tuning included, that exceed the spread across seeds, and it shows no significant disadvantage on the two arithmetic datasets (Table 2).

The two groups of datasets differ in where the computation is carried out. In the multi-step datasets only the answer is generated, so the relations must be composed inside the host before the first answer token. Prompt tuning and the near-constant vectors of the assistant-based methods (Section 5.2) supply the same input for every question, which can adjust the answer format and a task-level prior but not the composition of a particular question, whereas the proposed correction is computed from the host's reading of that question. By depth (Appendix B), the lead on ProsQA holds beyond the trained depths, at 84.7 against 74.7 for prompt tuning on depths 5 and 6. On CLUTRR it is confined to the trained depths 2 to 4, at 88.8 against 75.4 for LRT, and all methods are comparable on depths 5 to 10, where the number of possible relation chains grows as 14^k and every test chain is new. In the arithmetic datasets the intermediate steps are carried by the written solution, and every trained method lands close to the others and below the untrained host, because training makes them

Table 3: CLUTRR accuracy of each method with its own injected vectors compared with the same vectors averaged over questions or swapped between questions.

method	own vectors	averaged	swapped	own − swapped	shared component
SoftCoT	40.4	40.4	40.1	0.3	0.999
EBM-CoT	38.5	38.5	38.1	0.4	0.999
LRT	41.4	20.9	22.1	19.4	0.94
Ours	50.5	8.4	10.1	40.5	0.53

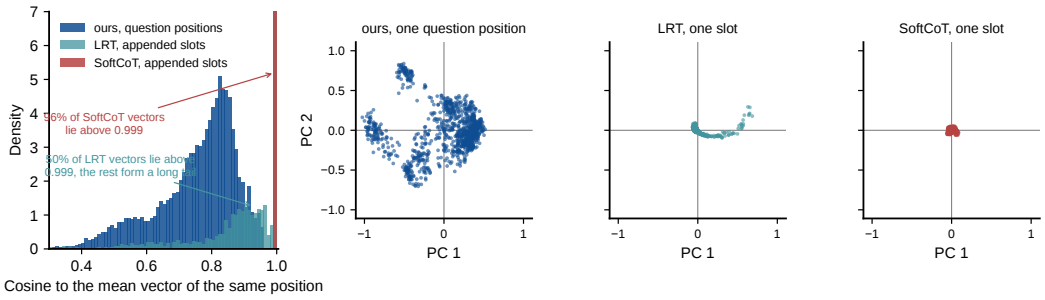

Figure 3: Variation of the injected vectors across CLUTRR questions for the proposed method compared with LRT and SoftCoT.

imitate reference solutions far shorter than the reasoning the host learned in instruction tuning. On GSM8K, the median solution has 150 tokens for the proposed method and prompt tuning and 193 for SoftCoT and EBM-CoT, against 146 for the references and 266 for the host. Depth results for the other datasets are given in Appendix C, and the CLUTRR ordering is reproduced on a smaller host in Appendix E.

5.2 Question dependence of the injected signal

To test whether each method's injected signal depends on the question, we replace each question's vectors with their average over the test set, or with the vectors of another question, and measure the share of vector energy carried by the average (Table 3, with EBM-CoT on one seed and LRT on seven). SoftCoT and EBM-CoT lose no points under averaging and at most 0.4 under swapping, and at least 99.9 percent of their energy is carried by the average, so they supply the same input for every question. LRT, whose proposer is trained on the task, loses 20.5 and 19.4 points, so its vectors do carry question-specific information, although 94 percent of their energy is still shared. For the proposed method, 53 percent of the energy is shared, between 0.50 and 0.55 per seed, and 42 and 40 points are lost under averaging and swapping, from 39 to 46 across seven seeds, which leaves accuracy below that of the host alone. Swapping keeps every correction a genuine one, taken from another question, yet it costs nearly as much as averaging, so the loss comes from the part of the correction that is specific to the question rather than from an unnatural input. Figure 3 visualizes this difference. The vectors of SoftCoT collapse into one tight cluster, those of LRT vary along a single direction, and the corrections that the proposed method maps from layer 8 spread widely across questions and in many directions.

5.3 Choice of the read layer

To test the influence of the read layer, we train the method with each read layer under the same fixed setup, select the layer on the validation split, and report test accuracy in Figure 4. CLUTRR accuracy rises toward the lower intermediate layers and falls again toward the last layers, with 47.9, 50.7, and 49.2 at layers 4, 8, and 12 over three seeds, and 48.9, 49.0, 47.4, and 47.6 at layers 16, 20, 24, and 28 on one seed, in line with reports that intermediate layers of language models carry the most transferable contextual representations, whereas the last layers specialize in next-token

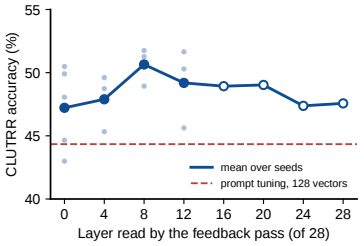

Figure 4: CLUTRR accuracy against the read layer, compared with prompt tuning.

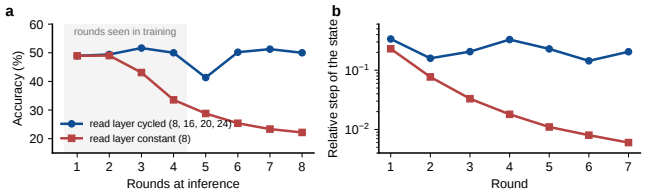

Figure 5: Accuracy (a) and change of the input between rounds (b) for a cycled and a constant read layer in one checkpoint.

Table 4: CLUTRR accuracy of prompt tuning and ours against trained parameters.

method	params	all	depths 2–4
Prompt, 32 vectors	0.11M	40.6	48.6
Prompt, 128 vectors	0.46M	44.3	53.6
Prompt, 512 vectors	1.84M	44.5	55.1
Ours, rank 64	0.47M	43.1	76.3
Ours, rank 256	1.85M	43.8	80.8
Ours, full rank	12.9M	**50.5**	**88.8**

Table 5: Accuracy of answer probes on the host's state compared with ours.

	CLUTRR depths			ProsQA
input	all	2–4	5–10	all
Majority class	7.6			49.5
Layer 0, token mean	12.9	34.2	7.1	49.9
Layer 4, last token	20.7	56.3	10.9	52.1
Layer 8, last token	21.5	53.6	12.7	53.9
Layer 8, token mean	15.0	26.1	11.9	50.4
Ours, reads layer 8	**50.5**	**88.8**	**40.0**	**88.0**

prediction (Liu et al., 2019; Skean et al., 2025). A learned convex combination of all layers stays near uniform and reaches 48.8 against 51.8 for the best single layer, so in these tasks, one selected layer is sufficient for comparison and is used elsewhere.

5.4 Accuracy against trained parameters

Among the baselines, SoftCoT, EBM-CoT and LRT train 10.5M to 16.5M parameters, the same order as the 12.9M of the full map (Table 6), whereas the 128-vector prompt trains 28 times fewer. To test whether the gain over prompt tuning comes from this difference, we restrict the mapper's matrix to $A = I + UV^\top$ of rank 64 or 256 and give the prompt 32 to 512 vectors (Table 4, three seeds for each new setting). On the trained depths, where the relations must be composed, the rank-64 map reaches 76.3 against 53.6 for a prompt of the same size. As parameters are added, prompt tuning gains 1.5 points on the trained depths from 128 to 512 vectors, whereas the proposed method gains 12.5 from rank 64 to full rank. Prompt tuning learns one input shared by all questions, whereas the proposed method transforms the host's reading of each question, which added capacity can continue to exploit.

5.5 Comparison with a read-out probe

Because the correction is computed from the host's layer-8 state, the method could in principle read an answer that the host has already formed there and steer the output toward it. To test this, we train linear and small MLP classifiers to predict the answer from the same state of the same prompt, taken at the last token or averaged over tokens, and select their regularization on the validation split (Table 5, details in Appendix G). On the trained CLUTRR depths the probes reach at most 56.3, close to the 53.6 of prompt tuning and far below the 88.8 of the method, on the unseen depths they fall to the level of the answer prior, and on ProsQA every probe stays within five points of the majority class while the method reaches 88.0. The layer-0 control, which sees each token's embedding alone, already reaches 34.2 on the trained CLUTRR depths, so much of what the probes extract is a lexical prior rather than a composed relation. The answer is therefore not present in the fed-back state in a form that a simple reader can extract, and the gain requires the computation that the full host performs on the corrected input.

Table 6: Extra stored parameters, vectors appended to the input and measured inference latency of each method on one A100 for a 227-token (short) and a 2859-token prompt (long).

method	extra parameters	appended vectors	first token (ms) short / long	32 tokens (ms) short / long
Host alone	0	0	29 / 206	738 / 912
Prompt tuning	0.5M	128	38 / 224	748 / 931
SoftCoT	1.5B + 16.5M	32	58 / 273	767 / 983
EBM-CoT	1.5B + 13.9M	32	59 / 273	773 / 981
LRT	10.5M	5	87 / 431	799 / 1137
Ours	12.9M	0	37 / 265	747 / 972

5.6 Dynamics of repeated feedback

To test whether repeated feedback keeps refining the input, we run one checkpoint trained over one to four rounds for up to eight rounds, reading either the same layer in every round or a cycled schedule of layers, and measure both accuracy and how much the corrected input changes from one round to the next (Figure 5). When the same layer is read, this change halves every round and vanishes after about five rounds, so further rounds no longer alter the input, and accuracy falls from 48.9 after one round to 22.2 after eight. When the read layer is cycled, the input keeps changing through all eight rounds, returning to nearly the same state every four rounds as the schedule repeats, and accuracy increases slightly on the rounds seen in training. The same comparison on ProsQA is given in Appendix D.

6 Discussion

The results indicate that a frozen LLM's intermediate representation can be recycled into its own input through a small learned transformation, leading to an efficient, question-conditioned improvement over proposer-based soft prompts. The substitution test places the source of this improvement in the part of the correction that differs between questions, which a shared signal such as prompt tuning cannot supply.

Inference cost. The cost of each method at inference is compared in Table 6, with the median of twenty interleaved runs and greedy decoding. The proposed method adds a partial pass through 8 of the 28 layers, so its first-token latency on the long prompt is 1.29 times the host's, and appends nothing to the input. SoftCoT and EBM-CoT must hold and run a separate 1.5B assistant, which costs more on short prompts and about the same on long ones, and LRT runs the host once to extract the features its solver reads and again to answer, which doubles the prompt cost. Prompt tuning is the cheapest, but it appends 128 vectors to every input, and its accuracy is 6.2 points lower on CLUTRR and 14.1 on ProsQA.

Limitations. Three questions raised by the method are left open for future work here. The first is reasoning depth. Repeated feedback with a cycled read layer keeps changing the input without collapsing to a fixed point, but it does not yet raise accuracy substantially. We attribute this mainly to the simplicity of the current mapper. With the intermediate layer information and input-dependent feedback shown, an information extractor with more complexity, a more flexible schedule determining the optimal feedback rounds at test-time, and additional supervision between feedback repetitions could lead this framework in future work into deeper reasoning with more feedback rounds.

The second is the location and potential mixture of the feedback. The read layer is chosen on the validation split, and a learned weighting over all layers stayed near uniform without reaching the selected layer, so a per-question choice given the variety of intermediate layer information is open.

The third is the computation performed in the feedback mapper. One linear map of a hidden state can reorganize what the host has computed but not perform a computation the host does not perform, as seen on GSM8K and AQuA, where a more capable extraction of information and potential offload of computation could lead to a more flexible design on complex or out-of-distribution tasks.

7 Related Work

Continuous prompts for a frozen LLM. A few trained continuous vectors are prepended to a frozen LLM by prompt and prefix tuning, through which much of fine-tuning's benefit can be recovered with fewer trained parameters (Lester et al., 2021; Li & Liang, 2021). In reasoning extensions, vectors are produced by trained modules. A frozen small assistant and trained projection are used by SoftCoT (Xu et al., 2025a), sampling and voting are used by SoftCoT++ (Xu et al., 2025b), learned-energy refinement is applied by EBM-CoT (Chen et al., 2025), and a task proposer is combined with recurrent low-dimensional refinement by Latent Recurrent Thoughts (Chen & Fu, 2026). Final states or anchors of the latent sequence are moved by learned controllers in ReLAR and AdaAnchor (Hsu & Yao, 2026; Sheshanarayana et al., 2026). Vectors are appended to the input by these methods. For SoftCoT and EBM-CoT, little question-specific variation is shown in Section 5.2. Here, the frozen LLM's own state is instead folded into existing positions without a proposer.

Latent reasoning that trains or loops the decoder. The last hidden state is fed back by Coconut (Hao et al., 2025), written chains are distilled by CODI (Shen et al., 2025), probability-weighted embeddings are fed back by Soft Thinking (Zhang et al., 2025), and latent positions are supervised by LOTUS in a looped transformer (Fan et al., 2026; Farhan & Chaudhary, 2026). The answering model's weights are changed by all of them, unlike in the frozen-decoder setting. Weight reuse for depth is represented by looped transformers (Saunshi et al., 2025), recurrent-depth pretraining (Geiping et al., 2025; Zhu et al., 2025b), and test-time depth extrapolation (Kohli et al., 2026). Recurrent blocks may converge to fixed points (Blayney et al., 2026), which are the predictions in attractor and equilibrium formulations (Fein-Ashley & Rashidinejad, 2026; Bai et al., 2019; Huang et al., 2026). Later changes in a frozen LLM can be limited by such convergence. The number of rounds is determined by learned stopping (Geiping et al., 2025; Zhu et al., 2025b; Song et al., 2026), whereas the within-round map is varied here, making the approaches complementary.

Steering a frozen LLM from outside. Frozen logits are shifted by tuned and untuned small models in proxy-tuning (Liu et al., 2024), a small model is used for drafting in speculative decoding (Leviathan et al., 2023), and inputs are edited by policy gradient or reward-based prompt search in latent test-time optimization (Li et al., 2025; Ye et al., 2026). The continuous-input interface is shared, but the signal comes from the frozen LLM rather than a second network or test-time search.

Parameter-efficient adaptation and activation steering. Low-rank adaptation trains updates to the model's weights (Hu et al., 2022), which change the host's computation in every adapted layer and for every generated token and can therefore cause catastrophic forgetting of the instruction-tuned model's abilities (Kalajdzievski, 2024; Lobo et al., 2025). Prefix and prompt tuning keep the weights but prepend trained vectors that are identical for every input (Li & Liang, 2021; Lester et al., 2021), so they lengthen each sequence without adapting to the particular question. Activation steering adds a fixed direction to intermediate activations (Turner et al., 2023; Zou et al., 2023; Li et al., 2023). The design proposed here leaves the weights unchanged and appends nothing to the input, and it computes a correction for each question from the frozen LLM's own intermediate state, so the input it supplies differs between questions.

8 Conclusion

We examined the signal that latent input tuning supplies to a frozen LLM and found that the vectors of assistant-based methods encode an adaptation to the task rather than question-specific information. Self-prompted latent reasoners instead feed the frozen LLM's own intermediate reading of each question back onto its input, without an auxiliary model or added input positions, and obtain the highest accuracy among the evaluated methods on three multi-step reasoning benchmarks while remaining on par with them on written arithmetic. Reading a frozen LLM's intermediate state and writing a correction back to its input provides a simple framework on which richer mappers, more selective extraction, and the offloading of deeper reasoning to repeated rounds can be built.

AI USE STATEMENT

Generative AI was used for manuscript language and diagram polishing, citation-metadata verification, and submission-format checks. Generative AI also assisted in experimental scripts and result extraction. All AI-assisted edits and scripts are fully reviewed and verified by the authors before submission.

REPRODUCIBILITY STATEMENT

The frozen host, the training data and the evaluation splits are public. The full update rule is given in Section 3, and the protocol shared by every method is given in Section 4. The searched learning-rate grid, the value selected for each method and the seeds are listed in Appendix A. The substitution procedure behind Table 3 is given in Appendix F in enough detail for repetition on any method in this family.

REFERENCES

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Hugh Blayney, Álvaro Arroyo, Johan Obando-Ceron, Pablo Samuel Castro, Aaron Courville, Michael M. Bronstein, and Xiaowen Dong. A mechanistic analysis of looped reasoning language models. *arXiv preprint arXiv:2604.11791*, 2026.

Zhaoliang Chen and Jie Fu. Latent recurrent thoughts: Recurrent refinement of proposed latents for reasoning with frozen LLMs. *arXiv preprint arXiv:2609.01117*, 2026.

Zhikang Chen, Sen Cui, Deheng Ye, Yu Zhang, Yatao Bian, and Tingting Zhu. Think consistently, reason efficiently: Energy-based calibration for implicit chain-of-thought. *arXiv preprint arXiv:2511.07124*, 2025.

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

Ying Fan, Anej Svete, and Kangwook Lee. Bridging the gap between latent and explicit reasoning with looped transformers. *arXiv preprint arXiv:2606.31779*, 2026.

Mujtaba Farhan and Maheep Chaudhary. Why limit the residual stream to layers and not tokens? Persistent memory for continuous latent reasoning. *arXiv preprint arXiv:2606.07720*, 2026.

Jacob Fein-Ashley and Paria Rashidinejad. Solve the loop: Attractor models for language and reasoning. *arXiv preprint arXiv:2605.12466*, 2026.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In *Advances in Neural Information Processing Systems*, volume 38, 2025. arXiv:2502.05171.

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. In *Conference on Language Modeling*, 2025. arXiv:2412.06769.

Chia-Hsuan Hsu and Jui-Ming Yao. Learning to refine hidden states for reliable LLM reasoning. *arXiv preprint arXiv:2606.17524*, 2026.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Benhao Huang, Zhengyang Geng, and J. Zico Kolter. Equilibrium reasoners: Learning attractors enables scalable reasoning. In *International Conference on Machine Learning*, 2026. arXiv:2605.21488.

Damjan Kalajdzievski. Scaling laws for forgetting when fine-tuning large language models. *arXiv preprint arXiv:2401.05605*, 2024.

Harsh Kohli, Srinivasan Parthasarathy, Huan Sun, and Yuekun Yao. Loop, think, & generalize: Implicit reasoning in recurrent-depth transformers. In *Conference on Language Modeling*, 2026. arXiv:2604.07822.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3045–3059, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.243. URL https://aclanthology.org/2021.emnlp-main.243/.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19274–19286. PMLR, 2023. URL https://proceedings.mlr.press/v202/leviathan23a.html.

Hengli Li, Chenxi Li, Tong Wu, Xuekai Zhu, Yuxuan Wang, Zhaoxin Yu, Eric Hanchen Jiang, Song-Chun Zhu, Zixia Jia, Ying Nian Wu, and Zilong Zheng. Seek in the dark: Reasoning via test-time instance-level policy gradient in latent space. *arXiv preprint arXiv:2505.13308*, 2025.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, Online, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.353. URL https://aclanthology.org/2021.acl-long.353/.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 158–167. Association for Computational Linguistics, 2017. doi: 10.18653/v1/P17-1015.

Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A. Smith. Tuning language models by proxy. In *Conference on Language Modeling*, 2024. arXiv:2401.08565.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 1073–1094. Association for Computational Linguistics, 2019. doi: 10.18653/v1/N19-1112.

Elita Lobo, Chirag Agarwal, and Himabindu Lakkaraju. On the impact of fine-tuning on chain-of-thought reasoning. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 11679–11698. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.naacl-long.584.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20275–20321, Suzhou, China, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.1025. URL https://aclanthology.org/2025.emnlp-main.1025/.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Abulhair Saparov and He He. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=qFVVBzXxR2V.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *International Conference on Learning Representations*, 2025. arXiv:2502.17416.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. CODI: Compressing chain-of-thought into continuous space via self-distillation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 677–693. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.emnlp-main.36/.

Disha Sheshanarayana, Rajat Subhra Pal, Manjira Sinha, and Tirthankar Dasgupta. Thinking in latents: Adaptive anchor refinement for implicit reasoning in LLMs. In *Latent & Implicit Thinking Workshop at the International Conference on Learning Representations (ICLR 2026)*, 2026. arXiv:2603.15051.

Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L. Hamilton. CLUTRR: A diagnostic benchmark for inductive reasoning from text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4506–4515, Hong Kong, China, 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1458. URL https://aclanthology.org/D19-1458/.

Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Nikul Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. Layer by layer: Uncovering hidden representations in language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 55854–55875. PMLR, 2025.

Shixiang Song, He Li, Zitong Wang, Boyi Zeng, Feichen Song, Yixuan Wang, Zhiqin John Xu, Ziwei He, and Zhouhan Lin. AdaPonderLM: Gated pondering language models with token-wise adaptive depth. *arXiv preprint arXiv:2603.01914*, 2026.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4593–4601, 2019.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J. Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering. *arXiv preprint arXiv:2308.10248*, 2023.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. SoftCoT: Soft chain-of-thought for efficient reasoning with LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23336–23351, Vienna, Austria, 2025a. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.1137. URL https://aclanthology.org/2025.acl-long.1137/.

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

Wengao Ye, Yan Liang, and Lianlei Shan. Thinking on the fly: Test-time reasoning enhancement via latent thought policy optimization. In *International Conference on Learning Representations*, 2026. arXiv:2510.04182.

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. In *Advances in Neural Information Processing Systems*, 2025. arXiv:2505.15778.

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

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, et al. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025b.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. Representation engineering: A top-down approach to AI transparency. *arXiv preprint arXiv:2310.01405*, 2023.

A PER-METHOD CONFIGURATIONS

In every trained method, Qwen2.5-7B-Instruct is used in bfloat16 for two epochs over the training set with a batch size of two, and greedy decoding with 32 new tokens under answer-only targets, or 512 on GSM8K and 1024 on AQuA under written solutions. For each method, a learning rate was selected on the held-out validation split over the grid $\{10^{-4}, 3\times10^{-4}, 10^{-3}, 3\times10^{-3}, 10^{-2}\}$ on CLUTRR and without 10^{-2} on ProsQA. The selected value was then used for seven seeds (42 to 48), except that prompt tuning was evaluated with five seeds (42 to 46).

Ours. Layer 8 is read, and one round is used both in training and at evaluation. A learning rate of 10^{-3} is used on both datasets. The trainable parameters are the 3584×3584 matrix A, the round vector b_1 and the gate, 12.9M in total. Across the seven CLUTRR seeds, a gate value of 0.67 is obtained with a standard deviation of 0.02.

Prompt tuning. A total of 128 free vectors of dimension 3584 are inserted after the question and trained with the same trainer. A learning rate of 10^{-3} is used on both datasets with five seeds (42 to 46). With 32 and 512 vectors, 40.6 and 44.5 are reached on CLUTRR (Table 4).

SoftCoT. Qwen2.5-1.5B-Instruct is used as the frozen assistant by which 32 soft thoughts are produced, and a 16.5M-parameter projection into the host is trained. A learning rate of 10^{-3} is used on CLUTRR, and 3×10^{-4} is used on ProsQA.

EBM-CoT. EBM-CoT is implemented on SoftCoT's official code with the same assistant and 32 soft thoughts in training and at evaluation and three Langevin steps, following Algorithms 1 and 2 of Chen et al. (2025). Its learning rate is selected on the validation split over $\{2\times10^{-5}, 10^{-4}, 3\times10^{-4}, 10^{-3}, 3\times10^{-3}\}$. On ProsQA one seed diverges during training and answers every question with the same concept, and it is excluded from the reported mean.

LRT. The official implementation is run with its two stages, the task-dedicated proposer and the recurrent refiner, and its published hyperparameters, with the user message set to the prompt used by every other method. The learning rate of the refiner is selected on the validation split.

Anchors. For the host-only anchor, the question is followed by a one-line instruction for the untrained host to answer in the required form (for CLUTRR, *Answer with only the relation, in the form #### ⟨relation⟩*; the number, the option letter, true or false, or the concept for the others). The same 32-token budget and extractor as in every trained method are used. Zero-shot chain of thought is the bare question in the host's own template with a budget of 2048 tokens, or 4096 on ProsQA and ProntoQA, and the same answer extraction.

Evaluation conventions. In the written-reasoning columns the user message is the raw question for every row. The chat form around it differs by family and was not equalized. The SoftCoT family, which includes EBM-CoT, is evaluated in its upstream test form, where the user turn is closed and no assistant generation prompt is appended, because its slot indices count back from the end of that sequence. Zero-shot CoT, prompt tuning and the proposed method use the host's chat template with the generation prompt. The effect of this difference has not been measured.

In the answer-only columns the user text itself differs by family, for a reason inherent to the methods rather than by choice. The SoftCoT family, which includes EBM-CoT, requires placeholder positions inside the prompt for its soft thoughts, so each of these methods uses the template it defines, while the proposed method and prompt tuning add nothing to the sequence and use the bare question. Each method is therefore evaluated in the form it requires, and the host anchors are reported under both conditions in Table 1, the bare question for zero-shot chain of thought and the instructed form for the direct-answer row.

B CLUTRR ACCURACY BY PROBLEM DEPTH

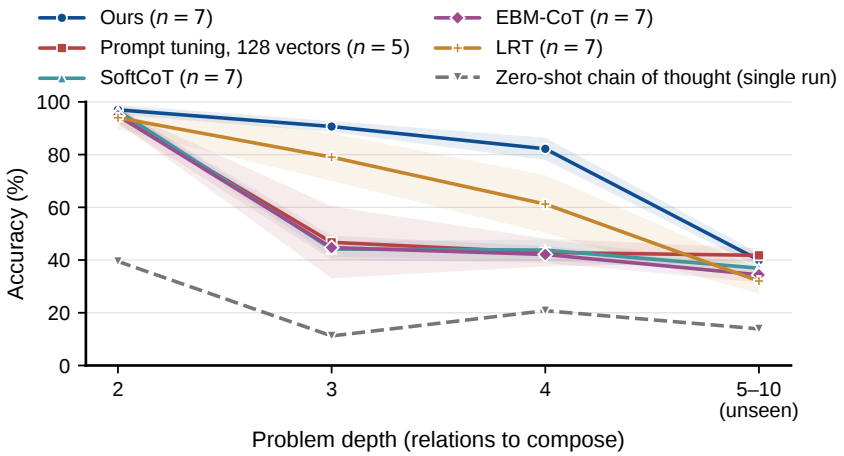

Figure 6: CLUTRR accuracy of every method at the trained depths 2 to 4 and pooled over the unseen depths 5 to 10.

Training covers depths 2 to 4 only, and the 806 test items at depths 5 to 10 are pooled in Figure 6. On the trained depths, 88.8 is obtained by the proposed method against 53.6 for prompt tuning and 75.4 for LRT, the strongest baseline there, with the largest leads at depths 3 and 4, where 91 and 82 are obtained against 79 and 61 for LRT. Two-relation items are solved by every method.

On the unseen depths, 40.0 is obtained against 41.8 for prompt tuning, 36.9 for SoftCoT, 34.4 for EBM-CoT and 32.0 for LRT. Accuracy is not reduced further from depth 5 to depth 10 for any method, which would not be expected if the relations were composed, and the evidence indicates that these items are answered from the answer distribution and surface cues instead. The gender of the answer is correct for 97 to 99 percent of predictions, so errors are made on the kind of relation. The answers at unseen depths are also distributed differently from those seen in training, with niece, nephew, aunt or uncle as the answer for 62 percent of unseen-depth items against 20 percent of training items, and a relation word copied from the story is emitted for 35 to 50 percent of predictions, against 15 to 22 percent at depth 3. Accuracy there is therefore governed by how often each method emits the frequent relations, and the per-depth differences between the proposed

method and prompt tuning alternate in sign, at $+1.6$, -7.0, $+0.5$, $+0.5$, -4.6 and -5.2 from depth 5 to 10 with 105 to 185 items per depth. Composition within the trained depths is improved by the proposed method, but extrapolation to longer chains is not obtained by it or by any evaluated baseline.

C ACCURACY BY PROBLEM DEPTH ON PRONTOQA, GSM8K AND AQUA

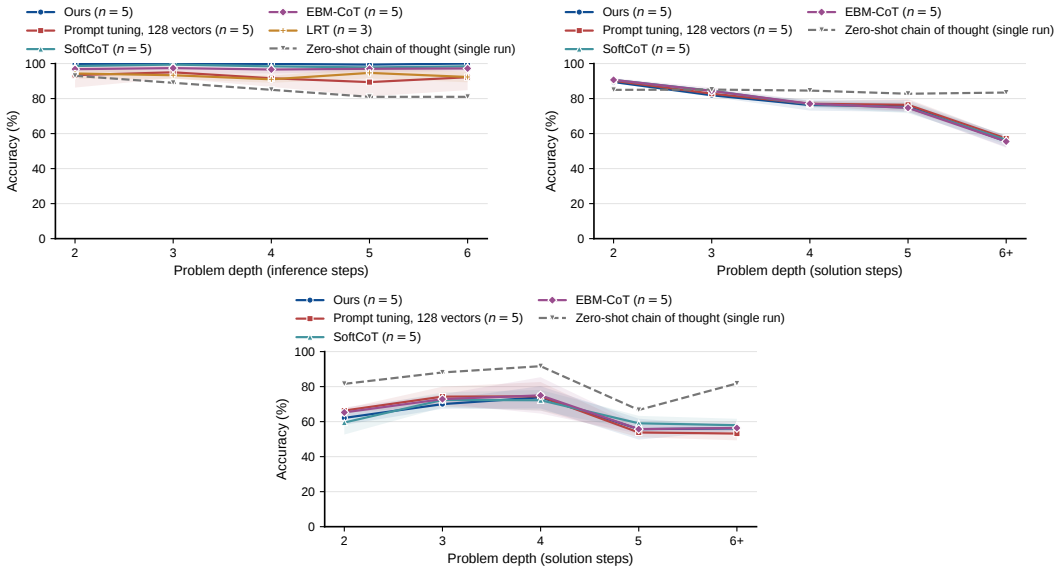

Figure 7: Accuracy of every method by problem depth on ProntoQA under answer-only supervision and on GSM8K and AQuA with written solutions.

Experiments on all three datasets were run under the protocol of Section 4 with five seeds per method. Learning rates were selected on the held-out validation split, over $\{3 \times 10^{-4}, 10^{-3}\}$ for SoftCoT and, for prompt tuning on GSM8K and AQuA, over $\{10^{-4}, 3 \times 10^{-4}, 10^{-3}, 3 \times 10^{-3}\}$, giving 10^{-3} and 3×10^{-3}. The proposed method, and prompt tuning on ProntoQA, use the rate selected on CLUTRR.

D ROUNDS ON A SECOND DATASET

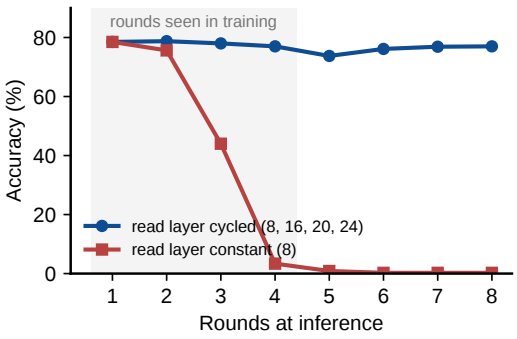

Figure 8: Accuracy on ProsQA against the number of rounds at inference for one checkpoint read two ways, cycled and constant, with the shaded range the rounds drawn during training.

The contraction measured on CLUTRR in Section 5.6 is sharper on ProsQA. Reading one checkpoint two ways, the cycled schedule holds between 73.8 and 78.8 across eight rounds, twice the

number drawn in training, while the constant read falls from 78.5 at one round to 44.0 at three, 3.4 at four and 0.25 from six onward, where the model no longer emits a parsable answer. The collapse is therefore total rather than partial, which is what an iterated contraction produces once the state has left the region the host can read. Both schedules are applied to the same weights, so the difference is a property of the update rule. Accuracy on the 124 items at depths 5 and 6 tracks the overall curve, with one exception at two rounds, 82.3 against 79.0 at one round, which is within one standard error of about 3.7 points on a subset that size.

E A SECOND HOST

Table 7: CLUTRR accuracy of every method with the smaller Qwen2.5-3B-Instruct as the frozen host.

method	CLUTRR, 3B host
Prompt tuning	39.4 ± 1.1
SoftCoT	37.0 ± 1.3
Ours	$\mathbf{44.6} \pm 0.3$

With a 3B host of the same family, the proposed method remains above prompt tuning and SoftCoT, and both gaps are smaller. Prompt tuning is exceeded by 5.2 points and SoftCoT by 7.6, against 6.2 and 10.1 on the 7B host. The smaller gap is consistent with, but does not establish, differences in the intermediate representations available to the correction. With three seeds, significance is not reached by any difference on the 3B host.

F THE AVERAGING SUBSTITUTION

Injected vectors are averaged by position over the test set and substituted before decoding, with their number and placement preserved. For the proposed method, corrections in Equation 1 are averaged across 1028 CLUTRR items at the 284 token-bearing positions counted backward from the sequence end, since questions differ in length, whereas SoftCoT's 32 appended slots are aligned directly. The shared component in Table 3 is $\sum_j \|\bar{v}_j\|^2 \big/ \frac{1}{N}\sum_{i=1}^{N}\sum_j \|v_{ij}\|^2$, where v_{ij} is the vector injected at aligned position or slot j for question i and $\bar{v}_j$ its mean over the N test questions; it equals 1 when every question receives the same vectors.

G READ-OUT PROBE ON THE LAYER-8 STATE

The frozen host is truncated to its first eight layers and given the exact prompt the method receives, and the state at the last prompt token and the mean over prompt tokens are taken from layers 0, 4 and 8. A logistic regression and a one-hidden-layer MLP are trained on 6000 of the method's training questions, the regularization strength is selected on 1000 questions from the held-out portion of the training data, and each probe is scored once on the test set. The CLUTRR label is the relation, and the ProsQA label is which of the two concepts named in the question is the answer. For each input, Table 5 reports the classifier with the higher validation accuracy. Layer 0, which sees each token's embedding only, serves as the control.