REORGANIZING LATENT REASONING IN LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Complex reasoning requires coordinating processes such as retrieval, semantic understanding, and mathematical calculation in a task-dependent order. Chain-of-Thought (CoT) improves reasoning through explicit intermediate steps, forming an external reasoning chain. However, internal computation within each forward pass still follows a fixed layer-wise order that may not align with the task's functional dependencies. We investigate whether reorganizing this latent computation can improve reasoning. We introduce *Modular Chain-of-Cognition (MoCoC)*, a framework that derives a cognitive ordering from human-verified task decompositions, localizes internal regions associated with distinct cognitive functions, and dynamically routes intermediate representations across these regions accordingly. For example, a task may require semantic understanding before calculation, even when semantic heads lie deeper than mathematical heads; MoCoC routes deeper semantic representations back to mathematical regions to support the required functional order. Across multiple LLMs and diverse reasoning benchmarks, MoCoC improves reasoning accuracy by approximately 3 percentage points on average without updating model parameters. Experiments on vision-language models further demonstrate its applicability to multimodal reasoning. These findings suggest that reasoning can be improved by reorganizing latent computation to coordinate and compose functional modules within LLMs into task-appropriate cognitive chains.

1 INTRODUCTION

Complex reasoning often requires coordinating multiple cognitive processes in a task-dependent order. Solving a mathematical word problem, for example, may involve interpreting its semantics, recalling relevant knowledge, performing calculations, and integrating intermediate results. Chain-of-Thought (CoT) prompting and its extensions improve reasoning in large language models (LLMs) by eliciting intermediate steps in natural language, decomposing problems into sequential subproblems, and aggregating multiple reasoning paths (Wei et al., 2022; Kojima et al., 2022; Zhou et al., 2022; Wang et al., 2022). Their success highlights the value of organizing and exploring intermediate reasoning steps. However, these methods primarily structure reasoning at the generated-text level: although intermediate outputs guide subsequent generation, computation within each forward pass still follows the Transformer's (Vaswani et al., 2017) fixed layer-wise order.

Recent work explores reasoning beyond natural language tokens through continuous latent states, recurrent computation, and the internalization of explicit reasoning traces (Hao et al., 2024; Geiping et al., 2026; Deng et al., 2024; Zhu et al., 2025). These approaches expand the scope of latent reasoning, but generally do not explicitly organize computation around the functional roles of internal components or the dependencies required by a task. Meanwhile, evidence of attention-head specialization suggests a layered functional organization in LLMs, with functions such as semantic understanding, retrieval, mathematical calculation, and inference exhibiting distinct distributions across depth (Ma et al., 2026). This organization, however, need not match the functional order a task requires. For example, calculation may depend on semantic understanding even when the associated semantic heads lie deeper than mathematical heads. This potential mismatch raises a central question: *Can we improve reasoning by reorganizing latent computation to coordinate and compose cognitive functions in a task-appropriate order, beyond their native layer-wise arrangement?*

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.

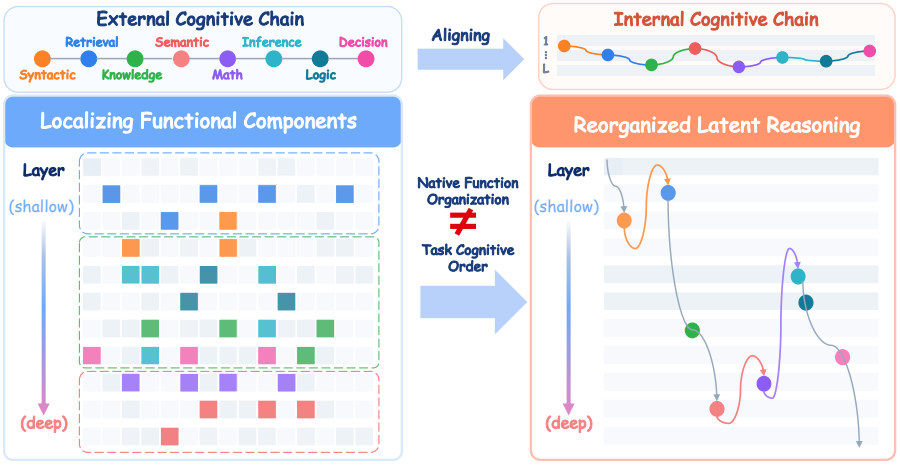


Figure 1: Overview of MoCoC. MoCoC localizes internal components associated with cognitive functions and aligns their computation with an external cognitive chain derived from human-verified task decompositions. By dynamically routing intermediate representations across layers, it reorganizes latent reasoning to follow task-dependent functional order without updating model parameters.

To investigate this question, we introduce **Mo**dular **C**hain-**o**f-**C**ognition (**MoCoC**) (Fig. 1), a framework that reorganizes a model's internal computation to better align with task-specific cognitive dependencies. Using CogQA (Ma et al., 2026), a human-verified dataset of reasoning problems decomposed into steps annotated with cognitive functions such as retrieval, semantic understanding, and inference, we derive a reference ordering of these functions. We then localize functionally specialized attention heads and compare their distribution across layers with this ordering to identify potential mismatches between the model's native computational order and the task's cognitive dependencies. Guided by these mismatches, MoCoC routes intermediate representations from deeper layers back to shallower regions associated with the relevant functions. This allows earlier components to process information developed in later layers, extending effective computational depth through function-guided revisitation. Without updating model parameters, MoCoC coordinates and composes existing capabilities into a task-appropriate chain of latent computation.

We evaluate MoCoC across multiple LLMs and diverse reasoning benchmarks, observing consistent improvements over standard inference without updating model parameters. Applying MoCoC only during first-token generation improves Llama-3.2-3B's average accuracy across seven benchmarks by 3 percentage points, with the largest gain on RTE, from 53.4% to 61.4% (+8.0 points). Extending MoCoC to vision-language models also improves multimodal reasoning performance. Case studies illustrate how function-guided reorganization helps models solve problems they fail under standard inference. These findings suggest that reasoning can be improved by coordinating and composing existing capabilities through task-appropriate latent computation.

Overall, this work highlights the importance of not only *how much computation* a model performs, but also *how that computation is organized*. Our findings suggest that reasoning depends on both the capabilities encoded within a model and their coordination and composition according to task-specific cognitive dependencies. Extending the organizing principle of CoT from explicit reasoning steps to internal computation, MoCoC offers a functionally grounded approach to improving reasoning without updating model parameters.

2 Related Work

Explicit and Latent Reasoning Chain-of-Thought (CoT) prompting and its extensions improve LLM reasoning through explicit intermediate steps, answer aggregation, problem decomposition, and structured search over reasoning paths (Wei et al., 2022; Wang et al., 2022; Zhou et al., 2022; Yao et al., 2023). These methods primarily organize reasoning as discrete language tokens. Latent reasoning instead explores intermediate computation without fully verbalizing each step (Zhu et al., 2025), extending computation along the sequence dimension through pause tokens or con-

tinuous hidden-state feedback (Goyal et al., 2024; Hao et al., 2024), or along the depth dimension through repeated layers and feedback connections (Saunshi et al., 2025; Tang & Lu, 2026; Zeng et al., 2026). Many such approaches require dedicated training to support modified computation. More closely related, layer-routing methods adapt the execution paths of frozen backbones by selectively skipping, executing, or repeating blocks (Heakl et al., 2026; Batorski et al., 2026; Li et al., 2026); freezing the backbone, however, does not eliminate the need to train routing modules. Recent training-free alternatives repeat selected layer segments at inference time, using representation interpolation or damped updates to stabilize computation (Lys et al., 2026; Chen et al., 2026). While these approaches enable flexible computational trajectories, they do not explicitly organize execution around cognitive functions and their dependencies. MoCoC differs in both motivation and routing criterion: it aligns latent computation with cognitive dependencies by linking internal routing to localized cognitive functions and organizing their execution according to an ordering derived from human-verified reasoning decompositions. This inference procedure is fully training-free, requiring neither backbone updates nor routing-policy training.

2.1 Functional Organization in Large Language Models

A growing body of work reveals functional specialization among Transformer components. Studies of attention heads (Zheng et al., 2024) identify distinct roles in pattern induction (Olsson et al., 2022), truthfulness (Li et al., 2023), information retrieval (Wu et al.), and safety alignment (Zhou et al., 2024). More systematic analyses link heads to cognitive functions, including retrieval, semantic understanding, mathematical calculation, and inference, revealing sparse specialization and distinct layer-wise distributions (Ma et al., 2026; Jiang et al., 2025). Complementary studies characterize MLPs as key–value memories and localize factual associations within them (Geva et al., 2021; Dai et al., 2022), while analyses across depth identify successive stages of feature construction, refinement, and prediction (Lad et al., 2026). These findings reveal a heterogeneous, layered functional organization, raising the question of whether its depth-wise arrangement aligns with the cognitive dependencies of a reasoning task. Building on component localization and intervention, MoCoC uses this organization to guide reasoning trajectories, routing representations across functionally relevant regions according to a cognitive ordering derived from human-verified reasoning decompositions.

3 Method

Setup and Notation. We use CogQA (Ma et al., 2026), which provides human-verified decompositions of reasoning problems into sequences of subquestions, each annotated with a cognitive function, such as retrieval, semantic understanding, and logical reasoning. These decompositions allow us to estimate a reference ordering of cognitive functions during reasoning. On the model side, we consider a decoder-only Transformer-based model $\mathcal{M}$ with L layers and H attention heads per layer. Let $\mathcal{F} = \{\phi_1, \ldots, \phi_K\}$ denote the set of cognitive functions, we identify attention heads associated with each function and characterize their distribution across layers. Comparing this functional organization with the CogQA-derived ordering reveals potential mismatches between the model's native layer-wise execution order and task-level cognitive dependencies. We use these mismatches to guide the routing of intermediate representations, reorganizing latent computation to better align with these dependencies.

3.1 Cognitive Ordering from Human-Verified Decompositions

We derive a reference cognitive ordering from CogQA (Ma et al., 2026), which is constructed from multiple NLP reasoning benchmarks. CogQA contains 570 main question–answer pairs and 3,402 validated subquestion–answer–cognitive function (subQAC) triplets spanning logical, mathematical, and commonsense reasoning. Each main question Q is decomposed into a human-verified sequence:

$$D(Q) = \big\langle (q_t, a_t, c_t) \big\rangle_{t=1}^{T_Q}, \tag{1}$$

where q_t, a_t, and $c_t \in \mathcal{F}$ denote the subquestion, its answer, and its cognitive-function label at step t, respectively, and T_Q is the number of steps. The labels cover eight main cognitive functions: **syntactic understanding**, **retrieval**, **knowledge recall**, **semantic understanding**, **mathematical calculation**, **inference**, **logical reasoning**, and **decision-making**.

To estimate when each cognitive function first appears without overweighting repeated occurrences, we retain only its first occurrence in each decomposition $D(Q)$. For each function ϕ appearing in $D(Q)$, let $\tau_Q(\phi) = \min\{t : c_t = \phi\}$ denote its first-occurrence index. Sorting the distinct functions by $\tau_Q(\phi)$ yields $\pi(Q) = \langle \pi_Q(1), \ldots, \pi_Q(n_Q) \rangle$, where $\pi_Q(i)$ is the function with the i-th smallest first-occurrence index and n_Q is the number of distinct functions in the decomposition. We exclude the 24 single-function chains, which provide no relative ordering information. For each remaining chain, we normalize each function's position to $[0, 1]$:

$$p_Q(\phi) = \frac{i-1}{n_Q - 1}, \qquad \pi_Q(i) = \phi. \tag{2}$$

The first and last functions thus receive positions of zero and one, respectively. We exclude single-function decompositions, as they provide no relative ordering information.

We define the *ordering score* of ϕ as its mean normalized position across the remaining chains containing it:

$$r(\phi) = \frac{1}{|\mathcal{Q}_\phi|} \sum_{Q \in \mathcal{Q}_\phi} p_Q(\phi), \qquad \mathcal{Q}_\phi = \{Q : \phi \in \pi(Q),\, n_Q \geq 2\}. \tag{3}$$

We sort functions by increasing $r(\phi)$ to obtain the global cognitive ordering in Table 1. The resulting order progresses from syntactic understanding and retrieval, through knowledge recall and semantic understanding, to mathematical calculation, inference, logical reasoning, and decision-making. This ordering summarizes the typical progression of first occurrences in the annotated reasoning chains and serves as an empirical reference for organizing latent computation.

Table 1: Ranks of each cognitive functions: CogQA-derived ordering scores and layer locations in LLM.

Source	Syntactic	Retrieval	Knowledge	Semantic	Math	Inference	Logical	Decision
CogQA	0.062	0.090	0.311	0.332	0.748	0.801	0.874	0.989
Llama-3.1-8B	14	13	14	25	24	14	14	14
Llama-3.2-3B	15	14	10	23	21	9	12	12
Gemma-3-4B	18	—	13	15	20	15	15	9
Gemma-3-12B	13	24	24	11	24	15	5	20

3.2 Localizing Functional Regions

Following Ma et al. (2026), we use nonlinear probing to identify attention heads associated with each cognitive function $\phi \in \mathcal{F}$. We select the top-scoring heads: $H_\phi = \mathrm{TopK}_{(\ell,h)} \left(s_\phi^{\ell,h}; k_\phi\right)$, where $s_\phi^{\ell,h}$ denotes the importance score of head h in layer ℓ for function ϕ. Since the selected heads may span multiple layers, we define a representative *functional layer* to summarize where the most strongly attributed heads are concentrated. We identify outlier heads within H_ϕ using Tukey's $1.5 \times$ IQR rule (Tukey et al., 1977):

$$\mathcal{O}_\phi = \left\{ (\ell, h) \in H_\phi : |s_\phi^{\ell,h}| > Q_{3,\phi} + 1.5\,(Q_{3,\phi} - Q_{1,\phi}) \right\}, \tag{4}$$

where $Q_{1,\phi}$ and $Q_{3,\phi}$ are the quartiles of absolute importance scores within H_ϕ. We count the outlier heads in each layer,

$$n_\phi^\ell = |\{h : (\ell, h) \in \mathcal{O}_\phi\}|, \tag{5}$$

and select the layer containing the most outlier heads:

$$\ell_\phi^\star = \arg\max_\ell n_\phi^\ell. \tag{6}$$

Ties are broken by the highest absolute head score among outliers in the tied layers. If $\mathcal{O}_\phi$ is empty, i.e. no head stands out from the score distribution, ϕ is assigned no functional layer and is excluded from routing in 3.4. The resulting functional layers provide model-specific reference locations for comparison with the cognitive ordering derived from CogQA.

3.3 IDENTIFYING MISORDERED COMPUTATION

The ordering score $r(\phi)$ summarizes the typical sequence of cognitive functions in human-verified reasoning decompositions, whereas the functional layer $\ell^\star_\phi$ represents their depth-wise concentration within the model. As shown in Table 1, these orderings do not always align. We define an *ordering violation* as a pair $(\phi_{\mathrm{pre}}, \phi_{\mathrm{dep}})$ in which a canonically earlier function is localized deeper than a canonically later one:

$$\underbrace{\ell^\star_{\phi_{\mathrm{pre}}} > \ell^\star_{\phi_{\mathrm{dep}}}}_{\text{architecturally deeper}} \quad \text{and} \quad \underbrace{r(\phi_{\mathrm{pre}}) < r(\phi_{\mathrm{dep}})}_{\text{canonically earlier}}. \tag{7}$$

This mismatch motivates routing information from the deeper functional region back to the shallower region for further processing. It identifies a potential conflict with the reference cognitive ordering, without establishing a strict dependency or assuming that a function is computed exclusively at its representative layer. On Llama-3.1-8B-Instruct, only 8 of the 18 candidate pairs with a deeper source layer satisfy the canonical-order criterion, substantially narrowing the set of potential routing connections.

3.4 REORGANIZING LATENT COMPUTATION

MoCoC treats functionally attributed head sets as computational modules and routes their outputs according to the cognitive ordering. Each ordering mismatch $(\phi_{\mathrm{pre}}, \phi_{\mathrm{dep}})$ defines a hop from the deeper source module to the input of the shallower target functional layer.We extract source signals from the selected heads, inject them at the target layer, and execute the resulting routing plan through cascading recomputation.

Routing Plan. Each ordering violation defines a routing hop. We order these hops according to the reference cognitive ordering of their target functions and merge those sharing a target layer into a single injection. The resulting plan is executed sequentially, allowing earlier interventions to inform subsequent source captures. For example, the Llama-3.1-8B configuration in Table 1 yields eight ordering violations: Syntactic Understanding→Retrieval; Semantic Understanding→{Mathematical Calculation, Inference, Logical Reasoning, Decision-making}; and Mathematical Calculation→{Inference, Logical Reasoning, Decision-making}. Since Inference, Logical Reasoning, and Decision-making share functional layer 14, these pairs yield three ordered routing steps:

$$14 \to \mathrm{Input}(13), \qquad 25 \to \mathrm{Input}(24), \qquad \{25, 24\} \to \mathrm{Input}(14). \tag{8}$$

Here, each source index denotes the functional layer from which the corresponding head-level signals are extracted, while $\mathrm{Input}(\ell)$ denotes the state entering target layer ℓ, i.e., the output of layer $\ell - 1$.

The first step routes syntactic information to the retrieval layer, and the second routes semantic information to the mathematical layer. The third combines semantic and updated mathematical signals at the shared functional layer for inference, logical reasoning, and decision-making. We recapture the mathematical source after recomputation with the semantic injection, allowing the final step to incorporate both the direct semantic signal and its influence on mathematical processing.

Source Capture. For each source function ϕ, we construct a function-associated signal by aggregating the selected head outputs located at and after its functional layer $\ell^\star_\phi$. Let $H^+_\phi = \{(\ell, h) \in H_\phi \mid \ell \geq \ell^\star_\phi\}$. The source signal is then

$$z_\phi = \sum_{(\ell,h) \in H^+_\phi} a^{\ell,h} W_O^{\ell,h}, \tag{9}$$

where $a^{\ell,h}$ denotes the output of head h in layer ℓ before the output projection, and $W_O^{\ell,h}$ is the corresponding block of the output projection matrix. These projections map head outputs into the shared hidden space, enabling direct aggregation without a learned adapter. By capturing function-associated heads after the functional layer, the signal incorporates downstream computation related to ϕ rather than being restricted to a single outlier layer. We compare this strategy with functional-layer-only and residual-stream capture in the ablation study. The resulting signal

is function-associated rather than function-exclusive and may also contain information relevant to other functions.

Target Injection. For each target, we sum its source captures $z_1, \ldots, z_m$ to obtain $x_{\text{pre}} = \sum_{j=1}^{m} z_j$. We then rescale this aggregate using a reference capture z_{ref} and add it to the injection-site hidden state x_{dep}:

$$x_{\text{dep}} \leftarrow x_{\text{dep}} + \alpha \frac{\|z_{\text{ref}}\|_2}{\max(\|x_{\text{pre}}\|_2, \epsilon)} x_{\text{pre}}, \tag{10}$$

where α controls injection strength and is shared across all hops, and $\epsilon > 0$ prevents division by zero. The reference capture is the first source in the hop's ordered source list. For merged hops involving a source updated by an earlier intervention, we place its recaptured signal first; in the third routing step above, this is the updated mathematical signal from layer 24. This normalization anchors the injection magnitude to a single source contribution, preventing it from growing solely with the number of aggregated sources.

We inject at $\text{Input}(\ell^{\star}_{\text{dep}})$, the input to the layer immediately preceding the target functional layer. This allows the preceding layer to process the injected signal before it reaches the target heads. Although the source signal is constructed from selected heads, the injection modifies the shared residual stream and can therefore affect all attention heads and MLP computations from the injection site onward.

4 Experiments

We evaluate whether organizing latent computation according to cognitive dependencies improves reasoning in frozen pretrained models. Our experiments address three questions: (1) Does MoCoC consistently improve reasoning across models and tasks? (2) Does this principle generalize to multimodal reasoning? (3) How do key design choices—including source capture, injection timing and strength, and cognitive ordering—affect its performance?

4.1 Experimental Setup

We evaluate MoCoC on multiple decoder-only LLMs, including Llama-3.1-8B-Instruct, Llama-3.2-3B-Instruct (Touvron et al., 2023), Gemma-3-4B, and Gemma-3-12B (Team et al., 2025). Our evaluation covers seven benchmarks: GSM8K (Cobbe et al., 2021) and AQuA (Ling et al., 2017) for mathematical reasoning, LogiQA (Liu et al., 2020) for logical reasoning, MMLU (Hendrycks et al., 2020) for broad knowledge and reasoning, WinoGrande (Sakaguchi et al., 2021) and PIQA (Bisk et al., 2020) for commonsense reasoning, and RTE (Wang et al., 2018) for textual entailment.

To assess generalization beyond language-only, we further evaluate Qwen2.5-VL-3B/7B (Bai et al., 2025) and Gemma-3-4B/12B (Team et al., 2025) on four multimodal benchmarks: MathVista (Lu et al., 2024) for visual mathematical reasoning, MMMU (Yue et al., 2024) for multidisciplinary multimodal understanding and reasoning, ScienceQA (Lu et al., 2022) for multimodal science question answering, and GQA (Hudson & Manning, 2019) for compositional visual reasoning.

Baselines. We compare MoCoC with standard inference using the same pretrained checkpoints. We also consider alternative approaches to latent computation: regularized inner looping (Lys et al., 2026) and learned layer-routing methods, including Dr.LLM (Heakl et al., 2026). We distinguish methods that require auxiliary routing-policy training from those that operate without additional training, while keeping the backbone frozen.

Evaluation Protocol. We use the same prompts, answer-extraction procedures, and exact-match scoring across methods for each benchmark. In our primary setting, MoCoC is enabled only during first-token generation; subsequent tokens follow the standard forward pass. We report per-benchmark accuracy and the unweighted average across the seven benchmarks, expressing absolute improvements in percentage points. To assess the cost of reorganizing computation, we also measure inference latency.

Table 2: Accuracy (%) on seven reasoning benchmarks across four LLM models. Bold indicates the best result within each model, and Avg. denotes the macro-average across the seven benchmarks.

Model	Method	GSM8K	LogiQA	MMLU	WinoGrande	PIQA	RTE	AQuA	Avg.
Llama-3.1-8B	Baseline	83.4	41.5	52.1	63.4	80.7	76.2	59.8	65.3
	Inner-loop	84.4	43.9	**53.3**	**66.8**	**82.4**	73.6	59.1	66.2
	Dr.LLM	78.6	40.9	52.3	66.1	79.1	75.5	59.8	64.6
	Ours	**85.0**	**44.7**	**53.3**	65.9	81.9	**79.1**	**63.0**	**67.5**
Llama-3.2-3B	Baseline	79.8	29.2	44.0	52.3	71.9	53.4	54.3	55.0
	Inner-loop	80.1	**30.1**	44.9	55.7	**75.4**	51.6	54.7	56.1
	Dr.LLM	74.3	26.9	41.2	49.9	68.1	54.9	50.4	52.2
	Ours	**81.0**	29.5	**45.0**	**56.7**	74.7	**61.4**	**57.9**	**58.0**
Gemma-3-12B	Baseline	**92.5**	57.9	65.5	70.1	88.2	84.5	78.3	76.7
	Inner-loop	91.9	59.8	**67.5**	**72.6**	87.3	**84.8**	80.7	**77.8**
	Dr.LLM	66.0	42.9	52.3	65.7	85.0	77.3	64.6	64.8
	Ours	92.4	**60.2**	66.5	71.3	**89.4**	82.7	**81.9**	**77.8**
Gemma-3-4B	Baseline	86.2	46.4	51.2	60.1	79.3	76.5	62.6	66.0
	Inner-loop	**87.0**	47.5	51.5	60.6	**80.6**	77.3	**71.7**	68.0
	Dr.LLM	45.9	41.8	41.3	51.2	69.5	41.2	53.1	49.1
	Ours	86.8	**47.6**	**53.9**	**62.2**	79.7	**78.0**	70.1	**68.3**

4.2 Main Results on LLMs.

Table 2 compares MoCoC with the zero-shot baseline, the training-free inner-loop method (Lys et al., 2026), and Dr.LLM. Overall, MoCoC achieves the best or tied-best macro-average across all four models, improving over the zero-shot baseline by 2.2, 3.0, 2.3, and 1.1 points on Llama-3.1-8B, Llama-3.2-3B, Gemma-3-4B, and Gemma-3-12B, respectively. Compared with the training-free inner-loop baseline, MoCoC achieves higher average accuracy on the first three models and ties on Gemma-3-12B.

The gains span diverse reasoning domains, including mathematical reasoning, logical reasoning and comprehension, general knowledge, and commonsense reasoning. The largest improvement occurs on RTE for Llama-3.2-3B, where accuracy increases from 53.4% to 61.4% (+8.0 points). MoCoC also yields substantial gains on AQuA for Gemma-3-4B (+7.5 points) and Gemma-3-12B (+3.6 points). Dr.LLM is less consistent across model families and often underperforms the zero-shot baseline, particularly on Gemma models, despite requiring an additionally trained module.

The improvements are consistent across both Llama and Gemma families and model scales from 3B to 12B. Notably, the smaller model in each family shows a larger average gain: +3.0 vs. +2.2 points for Llama-3.2-3B and Llama-3.1-8B, and +2.3 vs. +1.1 points for Gemma-3-4B and Gemma-3-12B. This pattern suggests that smaller models may have greater headroom to benefit from cognitive-function reshaping, while larger models still obtain consistent gains without additional model training. We further compare the inference cost of different methods in Appendix A.3, showing that MoCoC preserves baseline-level decoding throughput with only modest wall-clock overhead.

4.3 Generalization Across Modalities

We use CogVision (Jiang et al., 2025), which decomposes 1,409 image-grounded questions into 5,744 subQAC triplets, where each subquestion is annotated with a cognitive function. The taxonomy comprises eight functions spanning visual perception, knowledge recall, and language-based reasoning; details are provided in Appendix A.2.

We evaluate Qwen2.5-VL-3B/7B and Gemma-3-4B/12B on MathVista, MMMU, ScienceQA, and GQA, covering diverse multimodal reasoning settings. As shown in Table 3, MoCoC achieves the highest macro-average on all four VLMs, improving over the zero-shot baseline by 0.4–1.0 points. The gains are consistent across model families and scales, with notable improvements on GQA for Qwen2.5-VL-7B and on MathVista and MMMU for Gemma-3-12B. In contrast, the training-free inner-loop baseline provides only marginal gains on the Qwen models and even reduces average performance on both Gemma models, suggesting limited transfer of simple layer looping to

Table 3: Accuracy (%) on four reasoning benchmarks across four VLM models. Bold indicates the best result within each model, and Avg. denotes the macro-average across the seven benchmarks.

Model	Method	MathVista	MMMU	ScienceQA	GQA	Avg.
Qwen2.5-VL-7B	Baseline	68.7	52.9	**89.9**	48.6	65.0
	Inner-loop	68.6	51.8	89.3	51.4	65.3
	Ours	**68.9**	**53.0**	89.7	**52.2**	**66.0**
Qwen2.5-VL-3B	Baseline	54.3	43.6	**78.0**	48.7	56.2
	Inner-loop	55.8	**43.6**	76.9	49.0	56.3
	Ours	**55.9**	**43.6**	77.4	**49.5**	**56.6**
Gemma-3-4B	Baseline	**49.0**	42.1	80.7	**41.2**	53.3
	Inner-loop	47.4	43.4	80.9	35.0	51.7
	Ours	**49.0**	**43.7**	**81.5**	40.5	**53.7**
Gemma-3-12B	Baseline	59.7	53.0	85.9	44.6	60.8
	Inner-loop	60.0	52.3	**86.9**	42.9	60.5
	Ours	**61.3**	**54.2**	**86.9**	**44.8**	**61.8**

Table 4: Ablation study of MoCoC on Llama-3.1-8B.

Setting	Variant	GSM8K	LogiQA	WinoGrande	AQuA	Avg.
Baseline		83.4	41.5	63.4	59.8	62.0
Ours (default)		85.0	**44.7**	65.9	**63.0**	**64.6**
Where to capture	Functional layer only	84.7	41.2	63.2	55.1	61.0
	Residual stream	80.8	41.2	61.3	53.1	59.1
When to inject	All tokens	81.1	42.9	66.0	62.6	63.1
	Candidate-token protection	82.4	41.2	**66.3**	61.0	62.7
	Random 5% tokens	**85.2**	43.5	65.0	59.4	63.3

multimodal reasoning. MoCoC consistently yields higher average performance than the inner-loop baseline, indicating that function-guided cognitive reshaping generalizes more effectively beyond language-only reasoning without additional model training.

4.4 Ablation Study

We conduct ablation studies of MoCoC on Llama-3.1-8B using four benchmarks spanning mathematical, logical, and commonsense reasoning. We examine three key design choices: where to capture function-associated signals, when to inject them, and how strongly they should be injected. Results are summarized in Table 4.

Where to capture. The choice of source representation is critical. Head-level capture over the full top-k set achieves the highest average accuracy of 64.6%, compared with 61.0% when restricting capture to heads in the functional (outlier) layer and 59.1% when routing the full residual stream under the same first-token-only schedule. This suggests that the useful signal is distributed across capability-associated heads rather than confined to a single layer, while the residual stream introduces unrelated information from other heads and MLPs. The degradation becomes even larger when residual-stream routing is repeated during generation, reducing the average to 55.1% with candidate protection and 49.1% when applied at every token.

Table 5: Effect of injection strength on Llama-3.1-8B (accuracy %). Ours uses $s = 0.3$. Bold marks the best result in each column.

Strength	GSM8K	LogiQA	WinoGrande	AQuA	Avg.
$s = 0.1$	84.6	43.0	62.7	59.8	62.5
$s = 0.3$	**85.0**	**44.7**	**65.9**	**63.0**	**64.6**
$s = 0.5$	83.4	41.2	63.2	57.1	61.2

When to inject. Injecting only during the prefill pass that produces the first token performs best overall, reaching 64.6% average accuracy. Applying routing at every generated token lowers the average to 63.1%, while candidate-token protection reaches 62.7%. A random 5% token injection

Table 6: Effect of cognitive-function ordering on Llama-3.1-8B (accuracy %). We compare the canonical MoCoC routing order with two random permutations of the eight functions, while keeping the head sets and functional layers fixed. Thus, only the routing directions and target functions differ.

Plan	GSM8K	LogiQA	MMLU	WinoGrande	PIQA	RTE	AQuA	Avg.
Baseline	83.4	41.5	52.1	63.4	80.7	76.2	59.8	65.3
Ours	**85.0**	**44.7**	53.3	**65.9**	**81.9**	79.1	**63.0**	**67.5**
Random order (2 hops)	84.3	39.5	53.5	64.0	**81.9**	**80.1**	59.1	66.1
Random order (3 hops)	84.5	42.7	**54.1**	65.6	81.6	79.4	58.3	66.6

serves as a sparse control and achieves 63.0%, remaining below first-token-only routing. These results indicate that the benefit is not simply due to sparse intervention: reshaping before generation is more effective than repeatedly perturbing intermediate reasoning states, while also avoiding additional decoding-time intervention.

How strongly to inject. Performance also depends on the injection strength. The default $s = 0.3$ achieves the best average accuracy of 64.6%, whereas a weaker injection ($s = 0.1$) reaches 62.5% and a stronger one ($s = 0.5$) falls to 61.2%. This suggests that moderate injection provides sufficient routed information to influence subsequent computation without overwhelming the existing representation.

Does the cognitive-function order matter? Table 6 shows that the canonical cognitive-function order is important for effective reshaping. Although random permutations occasionally improve individual tasks, their average performance (66.1–66.6%) remains below MoCoC (67.5%). Since the head sets and capture layers are unchanged, this difference isolates the effect of routing order rather than the selected components themselves. Random orders can create mismatched transitions between source and target functions, weakening or even reversing the benefit of reshaping, as seen on LogiQA and AQuA.

We further examine case studies in Appendix A.4 further illustrate this effect: MoCoC restores a more coherent progression from problem understanding to downstream reasoning, whereas mismatched routing can perturb this functional sequence.

5 Conclusion

We introduced MoCoC, a framework that improves latent reasoning by reorganizing computation across functionally specialized regions of a model. Guided by a reference cognitive ordering derived from human-verified reasoning decompositions, MoCoC routes intermediate representations to better align internal computation with task-level cognitive dependencies. Experiments across multiple LLMs show consistent gains in average reasoning accuracy without updating model parameters, while results on vision-language models demonstrate its applicability to multimodal reasoning. These findings highlight the importance of coordinating and composing existing capabilities during inference. MoCoC derives a single reference ordering from CogQA for LLMs and CogVision for VLMs and reuses it across tasks. While efficient, this may miss dataset- or input-specific dependencies, and both first-occurrence ordering and single-layer function localization simplify potentially distributed mechanisms. Future work could explore context-dependent orderings and distributed functional representations for more adaptive routing and training.

AI Use Statement

We used generative AI tools to improve the manuscript's clarity and readability, assist with finding related work, and support code development and debugging. The authors reviewed and revised the AI-assisted text, checked the cited references for accuracy and relevance, and reviewed and validated the AI-assisted code. We take full responsibility for the final content of this work, including all text, code, citations, and claims.

Reproducibility Statement

The main text describes our method and experimental setup, with additional implementation details, hyperparameter settings, and evaluation procedures provided in the appendix. We identify the models and datasets used in our experiments to facilitate replication. Upon acceptance, we will release the source code.

References

John R Anderson. *Rules of the mind*. Psychology Press, 2014.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025.

Paweł Batorski, Abtin Pourhadi, Akylgali Aitaza, Przemysław Spurek, and Paul Swoboda. Macro: Markov chain routing of transformer layers. *arXiv preprint arXiv:2608.05872*, 2026.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Lizhang Chen, Jonathan Li, Chen Liang, Ni Lao, and Qiang Liu. Training-free looped transformers. *arXiv preprint arXiv:2605.23872*, 2026.

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.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8493–8502, 2022.

Yuntian Deng, Yejin Choi, and Stuart Shieber. From explicit cot to implicit cot: Learning to internalize cot step by step. *arXiv preprint arXiv:2405.14838*, 2024.

Adele Diamond. Executive functions. *Annual review of psychology*, 64(1):135–168, 2013.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *Advances in Neural Information Processing Systems*, 38: 41340–41391, 2026.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 5484–5495, 2021.

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. In *International Conference on Learning Representations*, volume 2024, pp. 27896–27923, 2024.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

Ahmed Heakl, Martin Gubri, Salman Khan, Sangdoo Yun, and Seong Joon Oh. Dr. llm: Dynamic layer routing in llms. In *International Conference on Learning Representations*, volume 2026, pp. 73320–73337, 2026.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6693–6702. IEEE, 2019.

Yanbei Jiang, Xueqi Ma, Shu Liu, Sarah Monazam Erfani, Tongliang Liu, James Bailey, Jey Han Lau, and Krista A Ehinger. Investigating the functional roles of attention heads in vision language models: Evidence for reasoning modules. *arXiv preprint arXiv:2512.10300*, 2025.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

Vedang Lad, Jin Hwa Lee, Wes Gurnee, and Max Tegmark. Remarkable robustness of llms: Stages of inference? *Advances in Neural Information Processing Systems*, 38:130050–130083, 2026.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in Neural Information Processing Systems*, 36:41451–41530, 2023.

Ziyue Li, Yang Li, and Tianyi Zhou. Skip a layer or loop it? learning program-of-layers in llms. *arXiv preprint arXiv:2606.06574*, 2026.

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, 2017.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. *arXiv preprint arXiv:2007.08124*, 2020.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in neural information processing systems*, 35:2507–2521, 2022.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations*, volume 2024, pp. 23439–23554, 2024.

Jonathan Lys, Vincent Gripon, Bastien Pasdeloup, Axel Marmoret, Lukas Mauch, Fabien Cardinaux, and Ghouthi Boukli Hacene. Inner loop inference for pretrained transformers: Unlocking latent capabilities without training. *arXiv preprint arXiv:2602.14759*, 2026.

Xueqi Ma, Jun Wang, Yanbei Jiang, Sarah Erfani, Tongliang Liu, and James Bailey. Cognitive mirrors: Exploring the diverse functional roles of attention heads in llm reasoning. *Advances in Neural Information Processing Systems*, 38:23978–24004, 2026.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

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*, volume 2025, pp. 14855–14881, 2025.

Mohan Tang and Sidi Lu. Turbo connection: Reasoning as information flow from higher to lower layers. *arXiv preprint arXiv:2602.17993*, 2026.

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.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

John Wilder Tukey et al. *Exploratory data analysis*, volume 2. Addison-wesley Reading, MA, 1977.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP*, pp. 353–355, 2018.

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

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.

Wenhao Wu, Yizhong Wang, Guangxuan Xiao, Hao Peng, and Yao Fu. Retrieval head mechanistically explains long-context factuality, 2024. *URL https://arxiv. org/abs/2404*, 15574.

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

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9556–9567, 2024.

Boyi Zeng, Yiqin Hao, Zitong Wang, Shixiang Song, He Li, Feichen Song, Yifan Liu, Ziwei He, Xinbing Wang, and Zhouhan Lin. Depth-attention: Cross-layer value mixing for language models. *arXiv preprint arXiv:2606.05014*, 2026.

Zifan Zheng, Yezhaohui Wang, Yuxin Huang, Shichao Song, Mingchuan Yang, Bo Tang, Feiyu Xiong, and Zhiyu Li. Attention heads of large language models: A survey. *arXiv preprint arXiv:2409.03752*, 2024.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.

Zhenhong Zhou, Haiyang Yu, Xinghua Zhang, Rongwu Xu, Fei Huang, Kun Wang, Yang Liu, Junfeng Fang, and Yongbin Li. On the role of attention heads in large language model safety. *arXiv preprint arXiv:2410.13708*, 2024.

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*, 2025.

A Appendix

A.1 Experimental Details

Ablation configurations. All variants run on Llama-3.1-8B with the generation protocol of Table 2 (zero-shot prompt, greedy decoding, 2048 new tokens, rule-based scoring, the same baseline for every row) and change exactly one factor of our configuration.

Ours. The cascade plan of Table 1 (three hops: layer 25 $\rightarrow$ 12 for Retrieval, layer 25 $\rightarrow$ 23 for Math, layers 24 and 25 $\rightarrow$ 13 for the three reasoning functions); *head-level capture* with the *full_topk* scope, i.e. at a capture layer only the output of the attention heads that belong to the captured function (its complete top-k head set, projected through their own `o_proj` columns) is taken, not the whole hidden state; the captured signal is blended into the target layer with strength s=0.3; *first-token-only* injection, i.e. the hops fire once on the prompt (the prefill pass that produces the first generated token) and every later token is generated without intervention; no stage gate.

Injection: all tokens. Identical to ours except that the hops fire on every generated token (`protect_mode single` without a protected function), so the captured signal keeps being blended into the target layers throughout decoding.

Injection: candidate-token protection. The token-selection rule that preceded first-token-only. For every generated token the eight functions' head activations are turned into z-scores (mean/std calibration); the hops are paused on a token whose dominant function among the candidates Semantic Understanding and Knowledge Recall has a z-score above $+1.0$, i.e. injection is suppressed on tokens that are themselves doing semantic understanding or knowledge recall, and fires on all other tokens. Tracing this rule showed that it protects 98–99.6% of the generated tokens, which is what motivated the explicit first-token-only rule.

Injection: random 5% of tokens. A sparsity control: the hops fire on a fixed-seed random 5% of the generated tokens (plus the prompt), with no z-score or token-type criterion, to test whether the benefit of sparse injection comes from sparsity alone or from injecting at the right place.

Target: residual stream. The three residual-stream rows keep the plan, strength and token rule of the corresponding head-level row but disable head-level capture: the whole hidden state of the capture layer is blended into the target layer, so the injected vector also carries the contribution of every head that has nothing to do with the captured function. The three rows use first-token-only, all-token and candidate-token injection respectively.

Heads: outlier_only. Head-level capture restricted to the heads of the captured function that sit at its own score-outlier layer (the narrower original scope), instead of the function's full top-k head set spread over every layer it is reachable in.

A.2 CogVision

CogVision (Jiang et al., 2025) is a benchmark for evaluating fine-grained multimodal reasoning. It contains 1,409 main questions and 5,744 associated subquestions. Each example consists of a main question and its answer, a decomposition into subquestions and subanswers, and an annotation specifying the perceptual or cognitive function required to answer each subquestion.

CogVision defines eight functions spanning the reasoning process required for complex multimodal question answering. These functions are grounded in established frameworks from cognitive science (Anderson, 2014; Diamond, 2013), which emphasize the roles of perception, working memory, and reasoning in human cognition. Three functions concern visual processing and five concern language-based processing, together covering the perceptual, mnemonic, and inferential operations involved in answering a multimodal question.

The functions related to vision include:

- **Low-level Visual Reception**: Recognizing basic visual features such as color, shape, size, position, motion.
- **High-level Visual Reception**: Integrating visual information to recognize objects, patterns, and scene structure with semantic meaning.

Table 7: Inference cost on Llama-3.1-8B (fp16, one A100 80GB, batch size 1, greedy, max 2048 new tokens; first 10 questions of GSM8K and WinoGrande, wall-clock per question including the prompt).

Dataset	Method	Time / question (s)	Generated tokens	Throughput (tok/s)	Rel. time
GSM8K	Baseline	31.3	1440	46.0	×1.00
	Inner-loop	35.6	1366	38.4	×1.14
	Dr.LLM	37.9	1754	46.3	×1.21
	Ours	37.5	1738	46.4	×1.20
WinoGrande	Baseline	3.1	141	46.1	×1.00
	Inner-loop	4.7	179	38.3	×1.53
	Dr.LLM	4.0	185	46.1	×1.31
	Ours	4.3	200	46.1	×1.41

- **Visual Knowledge Recall**: Applying long-term visual knowledge related to visual concepts and their properties.

The functions related to language include:

- **Language Information Extraction and Understanding**: locating and understanding relevant information from an external source or prior context.
- **Language Knowledge Recall**: Accessing domain-specific knowledge without visual input.
- **Math Reasoning**: Performing counting, arithmetic, comparison, and logic-based operations.
- **Inference**: deriving implicit information that is not directly stated.
- **Decision-Making**: Selecting the best outcome among alternatives based on reasoning.

Together, these eight functions provide a fine-grained decomposition of the cognitive operations underlying multimodal reasoning, which we use to derive the reference ordering for MoCoC in Section 4.3.

A.3 Inference Cost.

Table 7 compares inference efficiency on Llama-3.1-8B. MoCoC preserves essentially the same decoding throughput as the baseline (46.4 vs. 46.0 tok/s on GSM8K and 46.1 vs. 46.1 tok/s on WinoGrande), since its additional capture/injection operations occur only before generation. In contrast, the inner-loop baseline repeatedly executes additional layers at every decoding step, reducing throughput to about 38.4 tok/s. The wall-clock overhead of MoCoC is $\times 1.20$ on GSM8K and $\times 1.41$ on WinoGrande, partly reflecting its longer generated responses and the greater relative impact of the one-time preprocessing cost on short outputs. Dr.LLM shows comparable wall-clock overhead while also requiring an additional routing module. Overall, MoCoC adds modest inference latency without increasing the per-token decoding cost.

A.4 Case Study

Success analysis. Both successful cases show the same change in the shape of the reasoning chain rather than a repaired arithmetic step. The zero-shot baseline starts computing immediately: in Case 1 it introduces the unknown and writes an equation in its first sentence, and the equation encodes a misreading of the question ("bypass 8 flowers" becomes $0.9995N - N = 8$); in Case 2 it declares in its second step that two successive discounts add, and once the sum is not among the options it has no way back and loops until the length limit. The reshaped chain opens with one or two steps that do no arithmetic at all: it restates the relation the question is about ("the relationship between the percentage … and the total number", "an additional 20% discount on the already discounted price") and only then sets up the computation, which is now routine. This is exactly the reordering the plan is designed to: in Llama-3.1-8B the heads of Semantic Understanding sit at layer 25, deeper than the layer 13–14 heads of Inference, Logical Reasoning and Decision-making, so in the unmodified forward pass the reasoning functions act on a representation that does not yet contain the model's own semantic reading of the problem. Injecting the captured layer-25

Case 1: the baseline writes an equation before understanding what is asked; reshaping restores the order (AQuA, Llama-3.1-8B)

Query: A bee bypasses 0.05% of flowers it flies by because it can sense they don't have any nectar in them. How many flowers will the bee fly by to bypass 8 flowers? Options: A) 2000 B) 4000 C) 8000 D) 16000 E) 32000
Original Output: Let's denote the total number of flowers the bee will fly by as N. The bee bypasses 0.05% of flowers, which means it will visit 99.95% of flowers. The number of flowers the bee will visit is 0.9995N. The bee will bypass 8 flowers, which means 0.9995N - N = 8. Solving for N, we get N = 8000. The answer is C.
Reshaped Output (Ours): Step 1: To solve this problem, we first need to understand the relationship between the percentage of flowers the bee bypasses and the total number of flowers it flies by. Step 2: The bee bypasses 0.05% of flowers it flies by, which means it visits 100% - 0.05% = 99.95% of the flowers. Step 3: We are asked to find out how many flowers the bee will fly by to bypass 8 flowers. This means we need to find the total number of flowers such that 0.05% of them equals 8. Step 4: To do this, we can set up a proportion: 0.05% of x = 8, where x is the total number of flowers the bee flies by. Step 5: We can then solve for x by converting the percentage to a decimal and dividing both sides of the equation by 0.0005 (which is 0.05% expressed as a decimal). Step 6: This gives us x = 8 / 0.0005 = 16000. Step 7: Therefore, the bee will fly by 16000 flowers to bypass 8 flowers. The answer is D.

signal before layer 13 on the prompt makes that reading available when the chain is planned, and the first generated steps are spent on understanding instead of on a premature equation. The effect is confined to the prompt: both reshaped outputs are ordinary greedy continuations after the first token, which is why the later steps stay fluent and the arithmetic itself is unchanged.

Failure analysis. Reshaping is not monotone: on Llama-3.1-8B, 578 questions flip from wrong to right and 471 from right to wrong across the seven datasets of Table 2 (net $+107$; the sign test in the per-dataset tables is computed on these counts). Only 64 of the 471 regressions (14%) are degenerate outputs (repetition loops or no final answer); the rest are fluent chains that commit to a wrong intermediate claim. Case 3 is typical. The reshaped chain performs the semantic step correctly ("the remaining 20% are all yellow"), but then treats this restated sub-goal as the quantity asked for and never adds back the 240 yellow balls that were sold, whereas the baseline reaches the answer through the white balls without ever forming that intermediate statement. Injecting the deeper semantic state before the reasoning layers strengthens the model's commitment to its reading of the problem: when the reading is complete, the downstream chain is corrected (Cases 1–2); when it is partial, the later arithmetic inherits the omission. Two further patterns recur among the regressions: over-constrained readings of a condition (e.g. "Susan is on the left of Tim" interpreted as "Susan and Tim occupy the first two seats", giving 1 instead of 3 arrangements), and arithmetic slips in chains that are otherwise identical to the baseline's (e.g. $3 + 3 + 1.5 + 1.5$ summed to 8 on GSM8K). The method changes *where* understanding happens in the forward pass, not the model's arithmetic or its calibration, so it cannot repair errors that arise after the injected stage.

Case 2: the baseline adds two discounts and loops; reshaping first interprets "additional" as compounding (AQuA, Llama-3.1-8B)

Query: The manufacturer of tyres is offering a 20% discount on the price of its tubeless tyres. Some retailers are offering additional discounts. If a retailer offers an additional 20% discount, then what is the total discount available at that retailer? Options: A) 10% B) 25% C) 28% D) 30% E) 36%

Original Output: Step 1: The manufacturer is offering a 20% discount on the price of its tubeless tyres. Step 2: If a retailer offers an additional 20% discount, then the total discount available at that retailer is the sum of the discount offered by the manufacturer and the additional discount offered by the retailer. Step 3: To find the total discount, we need to add the two discounts together. Since both discounts are 20%, we can add them together to get 40%. However, this is not an option. We need to find the percentage equivalent of 80% (100% - 20% - 20%). Step 4: To find the percentage equivalent of 80%, we can subtract 80 from 100 to get 20, and then divide by 100 to get 0.20. Then, we can multiply 0.20 by 100 to get 20%. However, this is not an option either. [... the model repeats this step until the generation limit and never states an answer]

Reshaped Output (Ours): Step 1: First, let's understand the initial discount offered by the manufacturer. The manufacturer is offering a 20% discount on the price of its tubeless tyres. Step 2: Now, let's consider the additional discount offered by the retailer. The retailer is offering an additional 20% discount on the already discounted price. Step 3: To find the total discount available at the retailer, we need to calculate the final price after both discounts have been applied. Step 4: Let's assume the original price of the tyre is $100. After the manufacturer's 20% discount, the price becomes $80. Step 5: Now, the retailer offers an additional 20% discount on the $80 price. This means the retailer's discount is applied to the already discounted price of $80. Step 6: To find the amount of the retailer's discount, we calculate 20% of $80, which is $16. Step 7: Therefore, the final price after both discounts is $80 - $16 = $64. Step 8: To find the total discount, we subtract the final price from the original price: $100 - $64 = $36. Step 9: The total discount available at the retailer is 36% of the original price. Step 10: Therefore, the correct option is E) 36%. The answer is E.

Table 8: Table 2 with uncertainty: accuracy (%) and its standard deviation over 1000 bootstrap resamples of the questions (each dataset resampled with replacement; Avg. is the macro-average inside each resample). Bold marks the best mean per column within a model.

Model	Method	GSM8K	LogiQA	MMLU	WinoGrande	PIQA	RTE	AQuA	Avg.
Llama-3.1-8B	Baseline	83.4±1.0	41.5±1.9	52.1±1.5	63.4±1.5	80.7±1.3	76.2±2.6	59.8±3.1	65.3±0.7
	Inner-loop	84.4±1.0	43.9±2.0	**53.3±1.6**	**66.8±1.4**	**82.4±1.2**	73.6±2.7	59.1±3.3	66.2±0.8
	Dr.LLM	78.6±1.1	40.9±1.9	52.3±1.6	66.1±1.5	79.1±1.2	75.5±2.7	59.8±3.0	64.6±0.8
	Ours	**85.0±1.0**	**44.7±1.9**	**53.3±1.6**	65.9±1.5	81.9±1.2	**79.1±2.4**	**63.0±3.1**	**67.5±0.7**
Llama-3.2-3B	Baseline	79.8±1.1	29.2±1.7	44.0±1.6	52.3±1.6	71.9±1.4	53.4±3.1	54.3±3.2	55.0±0.8
	Inner-loop	80.1±1.1	**30.1±1.8**	44.9±1.5	55.7±1.6	**75.4±1.4**	51.6±3.1	54.7±3.1	56.1±0.8
	Dr.LLM	74.3±1.2	26.9±1.7	41.2±1.6	49.9±1.6	68.1±1.4	54.9±3.0	50.4±3.3	52.2±0.8
	Ours	**81.0±1.0**	29.5±1.8	**45.0±1.5**	**56.7±1.6**	74.7±1.3	**61.4±2.9**	**57.9±3.2**	**58.0±0.8**
Gemma-3-4B	Baseline	86.2±0.9	46.4±2.0	51.2±1.5	60.1±1.5	79.3±1.3	76.5±2.7	62.6±3.0	66.0±0.7
	Inner-loop	**87.0±0.9**	47.5±1.9	51.5±1.6	60.6±1.6	**80.6±1.2**	77.3±2.5	**71.7±2.8**	68.0±0.7
	Dr.LLM	45.9±1.3	41.8±1.9	41.3±1.6	51.2±1.6	69.5±1.4	41.2±3.1	53.1±3.2	49.1±0.8
	Ours	86.8±0.9	**47.6±2.0**	**53.9±1.6**	**62.2±1.5**	79.7±1.2	**78.0±2.5**	70.1±2.9	**68.3±0.7**
Gemma-3-12B	Baseline	92.5±0.7	57.9±2.0	65.5±1.5	70.1±1.5	88.2±1.1	84.5±2.2	78.3±2.6	76.7±0.7
	Inner-loop	**92.9±0.7**	59.8±1.9	**67.5±1.4**	**72.6±1.4**	87.3±1.0	**84.8±2.1**	80.7±2.6	**77.9±0.6**
	Dr.LLM	66.0±1.3	42.9±2.0	52.3±1.6	65.7±1.5	85.0±1.1	77.3±2.6	64.6±3.0	64.8±0.7
	Ours	92.4±0.7	**60.2±1.9**	66.5±1.5	71.3±1.4	**89.4±1.0**	82.7±2.3	**81.9±2.4**	77.8±0.6

Case 3 (failure): a correct semantic step is mistaken for the quantity asked; the sold yellow balls are never added back (AQuA, Llama-3.1-8B)

Query: A sporting goods store carries only yellow and white golf balls. At the beginning of the day it had 600 golf balls in stock, and by the end of the day it had sold 80% of its inventory of golf balls. If the store sold an equal number of yellow and white golf balls, and in doing so sold all of its white golf balls, how many yellow golf balls did the store have to begin the day? Options: A) 80 B) 120 C) 240 D) 320 E) 360

Original Output (correct): Step 1: The problem states that the store had 600 golf balls in stock at the beginning of the day and sold 80% of its inventory by the end of the day. Step 2: This means that the store sold 80% of 600, which is 0.8 * 600 = 480 golf balls in total. Step 3: Since the store sold an equal number of yellow and white golf balls, and in doing so sold all of its white golf balls, we can infer that the number of white golf balls sold is half of the total number of golf balls sold, which is 480 / 2 = 240. Step 4: This implies that the store had 240 white golf balls at the beginning of the day. Step 5: Since the store had 600 golf balls in total at the beginning of the day, the number of yellow golf balls can be found by subtracting the number of white golf balls from the total number of golf balls, which is 600 - 240 = 360. Step 6: Therefore, the store had 360 yellow golf balls at the beginning of the day. The answer is E.

Reshaped Output (Ours, wrong): Step 1: To solve this problem, we need to understand the given information and use it to find the number of yellow golf balls the store had at the beginning of the day. Step 2: We know that the store had 600 golf balls in stock initially and sold 80% of its inventory by the end of the day. This means that 20% of the golf balls were left in stock. Step 3: Since the store sold an equal number of yellow and white golf balls, and in doing so sold all of its white golf balls, we can infer that the remaining 20% of the golf balls are all yellow. Step 4: To find the number of yellow golf balls, we need to calculate 20% of the total number of golf balls, which is 600. Step 5: We can calculate 20% of 600 by multiplying 600 by 0.2, which gives us 120. Step 6: Therefore, the store had 120 yellow golf balls at the beginning of the day. The answer is B.

Table 9: Table 3 with uncertainty: accuracy (%) and its standard deviation over 1000 bootstrap resamples of the questions (Avg. is the macro-average inside each resample). Bold marks the best mean per column within a model.

Model	Method	MathVista	MMMU	ScienceQA	GQA	Avg.
Qwen2.5-VL-7B	Baseline	68.7±1.5	52.9±1.8	**89.9±0.9**	48.6±1.5	65.0±0.7
	Inner-loop	68.6±1.4	51.8±1.8	89.3±1.0	51.4±1.6	65.3±0.8
	Ours	**68.9±1.5**	**53.0±1.8**	89.7±1.0	**52.2±1.6**	**66.0±0.7**
Qwen2.5-VL-3B	Baseline	54.3±1.5	43.6±1.8	**78.0±1.3**	48.7±1.6	56.2±0.8
	Inner-loop	55.8±1.6	**43.6±1.7**	76.9±1.4	49.0±1.6	56.3±0.8
	Ours	**55.9±1.6**	**43.6±1.8**	**77.4±1.4**	**49.5±1.5**	**56.6±0.8**
Gemma-3-4B	Baseline	**49.0±1.6**	42.1±1.8	80.7±1.2	**41.2±1.5**	53.3±0.8
	Inner-loop	47.4±1.6	43.4±1.8	80.9±1.2	35.0±1.4	51.7±0.8
	Ours	**49.0±1.6**	**43.7±1.7**	**81.5±1.2**	40.5±1.6	**53.7±0.8**
Gemma-3-12B	Baseline	59.7±1.6	53.0±1.7	85.9±1.1	44.6±1.6	60.8±0.8
	Inner-loop	60.0±1.5	52.3±1.8	**86.9±1.0**	42.9±1.6	60.5±0.8
	Ours	**61.3±1.5**	**54.2±1.7**	**86.9±1.1**	**44.8±1.6**	**61.8±0.8**