ORCHESTRATION WITHIN LLMS VIA DISCOVERING AND ORCHESTRATING INTERNAL OPERATORS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Language Models (LLMs) are increasingly composed into agentic systems, where orchestration has become a key factor in system performance. Rather than relying on ever-larger models, agentic systems can solve complex tasks with models of bounded size, which makes orchestration a practical alternative to scaling. This motivates internalizing such orchestration within a single model, which holds the promise to improve both performance and parameter efficiency. To achieve it, we propose a framework for internally orchestrating LLMs, which performs orchestration inside a single LLM through two modules. The first module identifies patterns of coordinated activity among attention heads, which we call modes, and measures their causal effects on subsequent reasoning. We find that modes fall into three functionally distinct classes, corresponding to exploration, effort, and monitoring. Building on these findings, we introduce a Mode-Guided Orchestrator that uses mode-derived signals to determine when to intervene, adaptively invokes mode-specific interventions, and aggregates results through sequential consensus. Together, these modules dynamically construct query-specific computation graphs at inference time. Across five backbones and multiple benchmarks spanning mathematical, scientific, medical, and multi-hop reasoning, our method lies on the accuracy-compute Pareto frontier compared to the selected baselines.

1 INTRODUCTION

Large Language Models (LLMs) are increasingly composed into agentic systems, in which an orchestrator decomposes a task, dispatches sub-tasks, and aggregates the results (Wu et al., 2024; Hong et al., 2024; Shen et al., 2023). Orchestration has become a primary determinant of system performance, and it offers an alternative to scale: a well-orchestrated system built from bounded-size models can complete tasks that a single pass of a much larger model cannot (Snell et al., 2025; Wang et al., 2025; Zhang et al., 2025b).

This motivates developing effective orchestration within a single model, with the potential to not only improve model performance but also achieve greater parameter efficiency. Adaptive computation and inference-time reasoning dynamically allocate computation or increase reasoning effort at inference time, and can in fact be viewed as forms of internal orchestration (Raposo et al., 2024; Schuster et al., 2022; Muennighoff et al., 2025; Wang et al., 2023). This motivates a systematic study of internal orchestration: how can it be characterized and developed within a unified framework?

Orchestration requires operators: components that can be invoked to perform specific functions. In multi-agent systems (MAS), these operators take the form of explicitly defined planners, critics, or verifiers (Zou et al., 2026; Zhang et al., 2025a; Wu et al., 2024). Agent harnesses similarly provide tools, routines, or execution primitives that the model can invoke and combine (Yao et al., 2023b; Yang et al., 2024; Khattab et al., 2024). In both cases, the main problem is to determine how the available operators should be selected, ordered, and coordinated to solve a task effectively. However, internally orchestrating a model is fundamentally different, as there is no predefined set of operations to schedule. *This brings the first challenge: whether a set of specialized operators exists within an LLM, and, if so, what these operators are.*

To address this challenge, we seek reproducible patterns of internal computation that can be both monitored and causally manipulated within a frozen model. Specifically, as shown in the left panel of Figure 1, by analyzing how individual attention heads contribute to the residual stream, we find

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.

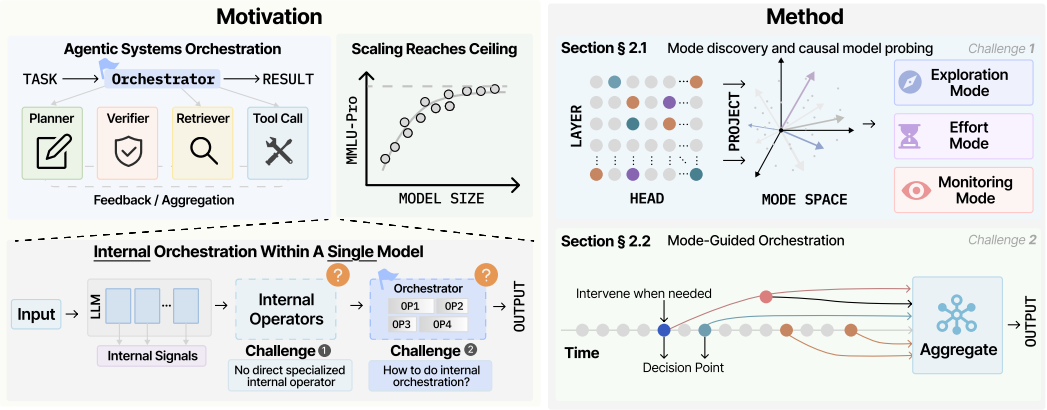


Figure 1: Overview of this work. Left: orchestration offers an alternative to scaling model size for improving model performance, motivating orchestration within a single LLM, which raises two challenges. Right: our approach addresses these challenges through mode discovery and causal mode probing and mode-guided orchestration at inference time.

that their outputs are organized along a small number of shared directions. We refer to each such direction as a *mode*. To quantify the causal effect of each mode, we introduce *Causal Mode Probing*, a permutation-controlled intervention that applies bidirectional gain perturbations to mode-associated attention heads. We compare the observed behavioral changes with those produced by a matched control, in which the same gain values are assigned to randomly selected attention heads. This helps isolate effects attributable to the mode structure itself. The intervention responses reveal three distinct response profiles: (1) *Exploration modes*, whose perturbation can increase trajectory diversity and promote alternative solution paths; (2) *Effort modes*, which can regulate the amount of reasoning by expanding or contracting trajectory length; and (3) *Monitoring modes*, which carry signals that reflect the quality of the model's reasoning trajectories.

With these internal operators characterized, *the second challenge is how to orchestrate these operators adaptively at inference time*. To address it, we further design a *Mode-Guided Orchestrator* that jointly determines when to intervene, which internal operators to invoke, and how to aggregate results. Specifically, as shown in the right panel of Figure 1, the orchestrator monitors the main trajectory using a risk score derived from mode signals and decides at the orchestration boundary whether intervention is necessary. Once triggered, it adaptively samples interventions from an operator pool, where each operator applies a different mode-specific intervention to the attention heads and then continues generation independently. The resulting trajectories are then aggregated through sequential consensus: intervened branches vote alongside the original trajectory until one answer establishes a lead. In this way, we dynamically construct query-specific computation graphs at inference time. Experiments show that the proposed mode discovery method generalizes across tasks and model backbones: the discovered modes span a nearly invariant subspace across tasks and exhibit shared functional structure across model backbones. Building on this, the proposed Mode-Guided Orchestrator consistently improves averaged performance over selected baselines. Our contributions are summarized as below.

- We formulate *internal orchestration*, asking whether a single LLM contains specialized operators that can be discovered and coordinated during inference.
- We identify a compact set of reproducible internal *modes* and introduce *Causal Mode Probing* to characterize their functions, revealing *Exploration*, *Effort*, and *Monitoring* modes.
- We develop a *Mode-Guided Orchestrator* that uses mode signals to determine when to intervene, invokes mode-specific operators to generate alternative trajectories, and aggregates their outputs through sequential consensus to produce a final answer.
- Experiments show that the discovered modes span a nearly task-invariant subspace and exhibit shared functional structure across model backbones. Our proposed internal orchestration lies on the accuracy-compute Pareto frontier compared to the selected baselines.

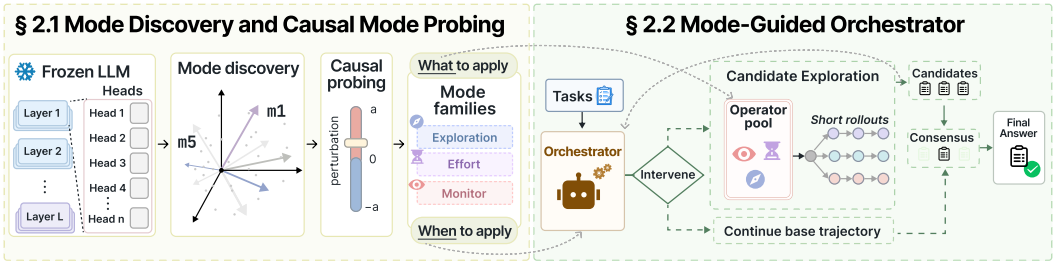


Figure 2: Method overview. Left: mode discovery decomposes the attention heads into a small set of shared modes, and Causal Mode Probing assigns them to Exploration, Effort, and Monitoring. Right: the Mode-Guided Orchestrator monitors the reasoning trajectory, intervenes when needed with short rollouts from the operator pool, and aggregates the resulting candidates by consensus.

2 METHOD

We formulate internal orchestration as two challenges. The first is to determine whether an LLM contains a set of specialized operators and how to identify them; the second is how to orchestrate these operators adaptively at inference time. Sections 2.1.1 and 2.1.2 address the first by discovering operators in the attention heads and establishing their causal roles. As shown in the left panel of Figure 2, starting from the attention heads of every layer, *mode discovery* (Section 2.1.1) finds that what the heads write into the residual stream is organized along a small number of shared directions, which we term the *modes*. Each mode pairs a weighted set of heads with a direction in the residual stream. *Causal Mode Probing* (Section 2.1.2) then perturbs one mode at a time against a matched control and records how the reasoning trajectory responds. The responses sort the modes into three families: Exploration, Effort and Monitoring modes. Section 2.2 addresses the second challenge with a mode-guided orchestrator that decides when to intervene, how to invoke the operators, and how to aggregate the resulting trajectories (the right panel of Figure 2). The orchestrator reads the Monitoring modes and decides whether to intervene. When it does, the model enters *candidate exploration*: it applies operators from the pool and runs short rollouts, each yielding a candidate answer. Candidates are collected sequentially and aggregated by *consensus*, which stops when one answer leads the rest and returns it as the final answer.

2.1 MODE DISCOVERY AND CAUSAL MODE PROBING

We want to find a set of internal operators that we can both observe and manipulate. Section 2.1.1 first identifies *modes*, which are recurring patterns of coordinated attention-head activity that emerge during reasoning, each pairing a weighted set of attention heads. Section 2.1.2 then measures the causal effect of each mode through Causal Mode Probing, a controlled perturbation that reveals the effect of adjusting a mode.

2.1.1 MODE DISCOVERY

To keep the entire reasoning process inside the model, where it can be observed and adjusted continuously, we consider a frozen decoder-only model f_θ in a soft-feedback latent loop (Zhang et al., 2025c). Instead of committing to a token at every step, the loop feeds back a continuous function of the hidden state, so a small intervention produces a small change in the trajectory rather than a token flip. Assume the model has L layers, H heads per layer ($N = LH$ heads), width d and an input-embedding table $\mathbf{E} \in \mathbb{R}^{|\mathcal{V}|\times d}$. At latent step t, the last-layer hidden state $\mathbf{h}_t$ gives logits $\mathbf{z}_t = \mathrm{lm_head}(\mathbf{h}_t)$. We keep the k_{top} largest logits, form a distribution over them at temperature T, and feed back the expected embedding:

$$p_{t,i} = \frac{\exp(z_{t,i}/T)}{\sum_{j\in\mathcal{K}_t}\exp(z_{t,j}/T)} \ (i \in \mathcal{K}_t), \qquad \boldsymbol{\mu}_t = \sum_{i\in\mathcal{K}_t} p_{t,i}\mathbf{E}_i, \qquad \mathbf{e}_{t+1} = \bar{\rho}\,\frac{\boldsymbol{\mu}_t}{\|\boldsymbol{\mu}_t\|_2}, \tag{1}$$

where $\mathcal{K}_t$ is the top-k_{top} set and $\bar{\rho}$ is the mean row norm of $\mathbf{E}$, which rescales the mixture $\boldsymbol{\mu}_t$ to the input scale the model was trained on. Running the loop until the model stops yields the sequence of hidden states $\mathbf{h}_1, \ldots, \mathbf{h}_\tau$, which we refer to as the *reasoning trajectory*.

In each layer, the attention heads read from the prompt and from earlier latent steps and write to the residual stream through:

$$\mathbf{h}^{(l+1)} = \mathbf{h}^{(l)} + \sum_{h=1}^{H} \mathbf{c}_{l,h} + \mathrm{MLP}_l(\cdot), \qquad \mathbf{c}_{l,h} = \mathbf{W}_O^{(l,h)}\, \mathbf{a}_{l,h} \in \mathbb{R}^d, \tag{2}$$

where $\mathbf{a}_{l,h}$ is the head's attention output and $\mathbf{c}_{l,h}$ is its *write* into the residual stream. We build our analysis on $\mathbf{c}_{l,h}$ because it is what a head actually contributes to the computation, expressed in a space shared by all heads, so that writes across the population can be compared and adjusted without altering what each head attended to. However, a model usually has hundreds to thousands of heads, individual heads are redundant and polysemantic, and an operation is typically spread over many heads while one head takes part in several. We therefore treat the heads as a population. At a latent step they jointly write a matrix $\mathbf{X}_t \in \mathbb{R}^{N\times d}$, one row per head, and what we look for is low-rank structure in this matrix that recurs across steps, items and tasks. We obtain it by decomposing

$$\bar{\mathbf{M}} = \frac{1}{|\mathcal{T}|} \sum_{\text{task}\in\mathcal{T}} \mathcal{N}\Big(\mathbb{E}_{i,t}\big[\mathbf{X}_t^{(i)}\big]\Big) = \sum_m \sigma_m\, \mathbf{u}_m \mathbf{v}_m^\top, \qquad \mathcal{N}(\mathbf{X})_r = \mathbf{X}_r/\|\mathbf{X}_r\|, \tag{3}$$

where $\mathcal{T}$ is the set of tasks used for discovery, $\mathbf{X}_t^{(i)}$ is the write matrix of training item i at sampled step t, $\mathcal{N}$ normalises each row to unit length, and σ_m, $\mathbf{u}_m$, $\mathbf{v}_m$ are the singular values and vectors. Averaging over items keeps only what a head writes consistently. Row normalisation removes the large differences in write magnitude across heads, which would otherwise let a few loud heads dominate the decomposition, and focuses it on the direction a head writes in. We use a singular value decomposition rather than a clustering because it gives each pattern two sides at once: mode m is the pair $(\mathbf{u}_m, \mathbf{v}_m)$, where the *head loadings* $\mathbf{u}_m \in \mathbb{R}^N$ say how much each head takes part and will later serve as the weights for intervening, and the *write direction* $\mathbf{v}_m \in \mathbb{R}^d$ says where that set of heads writes and serves as the direction for reading. A head may load on several modes with different signs, which a hard partition does not allow. We keep the M leading modes. Section 3.1 shows that these modes are largely shared across tasks and backbones.

2.1.2 Causal Mode Probing

The decomposition establishes that a mode exists but not what it does; for a mode to qualify as an operator, intervening on it must produce a specific, reproducible change in behavior. To measure the causal effect of each mode we propose Causal Mode Probing, a permutation-controlled intervention. For mode m, let $\mathbf{u}_m \in \mathbb{R}^N$ denote its loadings over the N attention heads and $\mathbf{v}_m$ its write direction. At a selected boundary, we perturb the mode by rescaling each head's residual write,

$$\mathbf{c}_{l,h} \;\to\; \big(1 + s\,\alpha\,\tilde{u}_{m,(l,h)}\big)\,\mathbf{c}_{l,h}, \qquad \tilde{\mathbf{u}}_m = \mathbf{u}_m / \max_{l,h} |u_{m,(l,h)}|, \qquad s \in \{-1, +1\}, \tag{4}$$

for the next k latent steps. The strength α sets how far each head is rescaled and the sign s chooses between amplifying and attenuating the mode. The vector this adds to the residual stream at step t can be represented as

$$\boldsymbol{\delta}_t^{(m,s)} = \sum_{l,h} s\,\alpha\,\tilde{u}_{m,(l,h)}\,\mathbf{c}_{l,h} = s\,\alpha\,\mathbf{X}_t^\top \tilde{\mathbf{u}}_m, \qquad \mathbf{v}_m^\top \boldsymbol{\delta}_t^{(m,s)} = s\,\alpha\,\tilde{\mathbf{u}}_m^\top \mathbf{X}_t \mathbf{v}_m, \tag{5}$$

that is, the loading-weighted sum of what the mode's heads are writing at that moment, which moves the stream along the mode's own direction in proportion to the mode's current activity. In this way, amplifying asks the model for more of an operation it is already performing, attenuating asks for less, and heads that do not take part are left alone. The perturbation is kept for k steps so that we observe the trajectory's reaction to it rather than the behavior of a different model, and it is applied in both signs because the two directions of a mode need not have symmetric effects, and the difference between them turns out to be informative.

A behavioral change under the perturbation is not sufficient evidence of a causal effect of the mode: rescaling an arbitrary set of heads may also change the trajectory. We therefore construct a control that keeps the same scaling factors but randomizes their assignment to heads. Let $\boldsymbol{\gamma}^{(m,s)} \in \mathbb{R}^N$ collect the factors $1 + s\,\alpha\,\tilde{u}_{m,(l,h)}$ of equation 4, and let $\mathbf{P}$ be a random permutation of the N heads. The control and the resulting read-out are

$$\boldsymbol{\gamma}^{\mathrm{ctrl}} = \mathbf{P}\,\boldsymbol{\gamma}^{(m,s)}, \qquad \Delta^{(m,s)} = R\big(\boldsymbol{\gamma}^{(m,s)}\big) - R\big(\boldsymbol{\gamma}^{\mathrm{ctrl}}\big), \tag{6}$$

where $R(\cdot)$ is any measurement of the trajectory continued from the boundary. The control keeps the same multiset of scaling factors and touches the same number of heads; the only thing it removes is that the factors follow the loadings of mode m. In this way, we make the difference between the perturbation and its control come from the mode structure.

A perturbation can change a trajectory in several ways, and we characterize it by two measurements: (1) how committed the model is to its current path, read from the entropy H of its next-token distribution averaged over the steps after the boundary; (2) how much computation it spends, the number of latent steps τ from the boundary to termination. The paired differences $\Delta^{(m,s)}$ of these measurements form the mode's *response profile*. We turn profiles into roles by a local sensitivity analysis: each mode is perturbed alone and every measurement is tested against its permuted control. The responses fall into three kinds.

- **Exploration mode**: perturbing the mode raises the entropy of the fed-back distribution well above its baseline. Such a mode helps hold the current solution path in place; disturbing it increases trajectory diversity and opens alternative solution paths.
- **Effort mode**: the entropy of the fed-back distribution stays close to its baseline, but the trajectory becomes longer in one direction and shorter in the other, controlling the amount of reasoning the model performs.
- **Monitoring mode**: the mode's activity carries a signal of trajectory quality.

More details are provided in Section 3.1.

2.2 Mode-Guided Orchestration

Causal Mode Probing leaves us with an operator set $\mathcal{O} = \{(m, s)\}$, where m indexes a mode and $s \in \{-1, +1\}$ the direction in which it is rescaled; each operator inherits the role $\rho(m) \in \{\text{Exploration}, \text{Effort}, \text{Monitoring}\}$ of its mode. For a problem x, the orchestrator dynamically builds a computational graph: a main trajectory that yields an answer a_0, branches that fork from it under different operators and yield answers $a_1, a_2, \ldots$, and an aggregation of these answers into a final $\hat{a}$. Its objective is accuracy at bounded compute,

$$\max_{\pi} \ \mathbb{E}_x\big[\mathbb{I}\{\hat{a} = a^\star(x)\}\big] \quad \text{s.t.} \quad \mathbb{E}_x[C(x)] \le B, \tag{7}$$

where C counts latent steps relative to the main trajectory. Interventions should land where the reasoning can be redirected without cutting a thought in half, so the policy π acts at sentence boundaries rather than at arbitrary steps. In the latent loop these boundaries are detected from the model's own output distribution, whenever it places most of its mass on a sentence terminator. At a boundary, π decides whether to branch, which operators to issue, and how to aggregate the resulting trajectories into the final answer $\hat{a}$.

Intervention effectiveness and diagnostic reliability peak at different stages of a trajectory: perturbations are most effective early in generation, whereas reliable diagnosis requires sufficient trajectory context. We therefore temporally decouple diagnosis and intervention. Specifically, we design a risk score to determine whether intervention at t_{f} is needed:

$$r = g\big(\mathbf{z}_{t_{\mathrm{f}}}^{\mathrm{Mon}},\ c_0,\ \tau_0,\ \kappa_0\big), \tag{8}$$

where $\mathbf{z}_{t_{\mathrm{f}}}^{\mathrm{Mon}}$ collects the activities of the Monitoring modes at t_{f}, c_0 is the confidence of the trajectory, τ_0 its length and κ_0 an indicator of whether it reached the step ceiling, and g is a logistic model fitted once on training sets.

If intervention is triggered, we create a branch under operator (m, s),

$$a_{(m,s)} = \mathrm{Readout}\Big(\mathcal{T}_{(m,s)}^{k}\big(\mathcal{S}_{t_{\mathrm{f}}}\big)\Big), \tag{9}$$

where $\mathcal{T}_{(m,s)}^{k}$ restores the saved snapshot $\mathcal{S}_{t_{\mathrm{f}}}$, applies the rescaling of equation 4 for k latent steps, and runs the unmodified model to termination. Operators are applied one at a time, following a schedule σ over the operator set, and every branch starts from the same snapshot, so the branches share the prefix up to t_{f} and each costs only the steps it adds.

To determine both how many branches to create and how to combine their answers, we further propose *sequential consensus*. Let $n_j(a) = \mathbb{I}\{a_0 = a\} + \sum_{i \le j} \mathbb{I}\{a_i = a\}$ be the votes for answer a after j branches. The orchestrator stops at the first

$$J = \min\left(\left\{1 \le j \le K_{\max} : \max_a n_j(a) - \max_{a' \neq a_j^\star} n_j(a') \ge \ell\right\} \cup \{K_{\max}\}\right), \tag{10}$$

where $a_j^\star$ is the current leader, ℓ the required lead and $K_{\max}$ the budget, and returns the leading answer. The rule reads the branches as evidence and stops when the required lead is reached or the branch budget is exhausted, whichever occurs first. This concentrates compute on the problems the model is uncertain about, without any explicit estimate of difficulty, and leaves a single parameter, the lead ℓ, to trade accuracy against compute.

3 Evaluation

Setup. Our experiments span five models, Qwen3-1.7B, Qwen3-4B, Qwen3-8B (Yang et al., 2025), Mistral-7B-Instruct-v0.3 (Jiang et al., 2023) and Llama-3.1-8B-Instruct (Grattafiori et al., 2024), and five benchmarks, GSM8K (Cobbe et al., 2021), ARC-Challenge and ARC-Easy (Clark et al., 2018), MedQA-USMLE (Jin et al., 2021) and HotpotQA-Distractor (Yang et al., 2018). We compare against methods that improve model performance at inference time. (1) *Single-model methods*, which improve one model's own reasoning, in text space by sampling and aggregating several chains of thought and in latent space by altering how the model reasons internally. Specifically, in text space, we use zero-shot chain-of-thought prompting (**CoT**) (Kojima et al., 2022), self-consistency with 4 and 8 samples (**SC@4**, **SC@8**) (Wang et al., 2023), early-stopping self-consistency (**ESC**) (Li et al., 2024), **Adaptive-Consistency** (Aggarwal et al., 2023) and **DeepConf** (Fu et al., 2026). In latent space, we use **Soft Thinking** (Zhang et al., 2025c), and **LTPO** (Ye et al., 2026). (2) *Multi-agent systems*, which compose multiple models into a system that communicates either in text or in latent space. For text-space communication, we consider **Multi-Agent Debate** (Du et al., 2023) and **TextMAS** (Zou et al., 2026), with both sequential and hierarchical designs. For latent-space communication, we use **LatentMAS** (Zou et al., 2026), again considering both sequential and hierarchical designs. More implementation details are provided in Appendix.

3.1 Generalization of Mode Structure and Function

We first show that the modes are a property of the backbone rather than of the particular data used to extract them. We conduct two experiments: the first experiment tests the stability of the modes across different samples from the same benchmark, while the second experiment tests their consistency across different benchmarks. First, for each backbone and benchmark, we draw a set of samples, split it into two disjoint subsets, and independently apply the extraction procedure from Section 2.1.1 to each subset. We then compare the resulting mode sets. Table 1 reports the mean principal cosine between the two sets of modes, which is at least 0.987 for every backbone and benchmark. For comparison, random matrices of the same shape give 0.07-0.11. These results show that the extracted modes are insensitive to the particular samples from the same benchmark used for extraction.

Table 1: Mode consistency within the same benchmark. "Q" denotes Qwen3.

Benchmark	Q-1.7B	Q-4B	Q-8B	Mistral	Llama
GSM8K	0.996	0.997	0.997	0.995	0.996
ARC-C	0.995	0.995	0.996	0.989	0.995
ARC-E	0.993	0.994	0.996	0.994	0.991
MedQA	0.996	0.987	0.995	0.991	0.994
HotpotQA	0.996	0.997	0.997	0.996	0.992

We next test whether the same modes are shared across tasks. For each backbone, we independently extract modes from randomly sampled examples from each benchmark using the method in Section 2.1.1, and compare the resulting mode sets for every pair of benchmarks. Figure 3 shows the mean principal cosine between mode sets extracted from each pair of benchmarks, with colors and marker shapes distinguishing the five backbones. Across all benchmark pairs and backbones, the mean principal cosine ranges from 0.861 to 0.996, with an overall mean of 0.922. Thus, the mode set extracted from one benchmark recovers most of the structure obtained from another. Taken together, these results indicate that the modes are insensitive to the choice of samples and largely

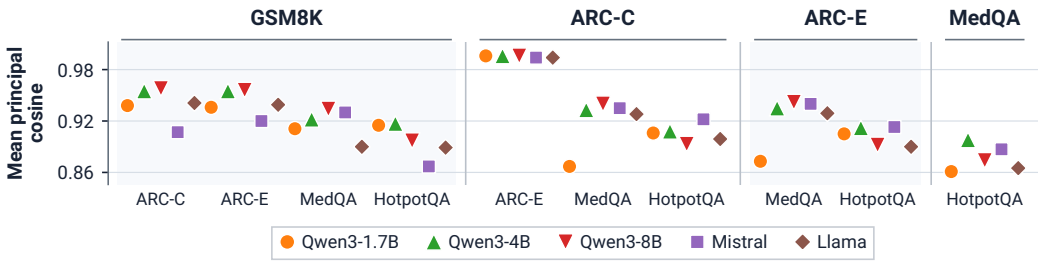


Figure 3: Mode consistency across benchmarks. Each marker is the mean principal cosine between the mode sets extracted independently on two benchmarks for one backbone; the group label names one benchmark of the pair and the x-axis label the other. Color and marker shape give the backbone.

Table 2: Accuracy (%) comparison across methods. Results are averaged over five different backbones to account for variation in model architecture. The best results are shown in bold.

Method	GSM8K	ARC-C	ARC-E	MedQA	HotpotQA
CoT	78.2	85.8	93.4	61.0	66.9
Self-Consistency@4	82.2	87.4	94.4	62.3	66.5
Self-Consistency@8	84.2	87.7	94.7	63.4	67.1
Early-Stopping SC	84.2	87.7	94.7	63.4	67.1
Adaptive-Consistency	84.2	87.7	94.7	63.4	67.1
DeepConf	**84.8**	87.9	94.8	63.6	67.2
Soft Thinking	79.0	85.9	93.6	61.2	71.0
LTPO	77.7	84.3	92.3	60.3	68.3
Multi-Agent Debate	81.5	87.3	94.3	62.8	69.4
TextMAS, sequential	77.1	84.1	93.7	59.7	64.2
TextMAS, hierarchical	76.1	79.5	87.0	59.9	70.8
LatentMAS, sequential	75.2	80.8	89.8	57.4	61.0
LatentMAS, hierarchical	79.1	82.4	90.8	57.2	60.2
Ours	84.2	**88.2**	**95.2**	**64.0**	**71.2**

preserved across tasks, supporting that they reflect an intrinsic property of the backbone rather than dataset-specific variation. More details are provided in Appendix A.3.

We next report the results of Causal Mode Probing, which we use to characterize the functional role of each mode. For each mode, we report the profile on two key measurements: the entropy of the fed-back distribution and the length of the continuation after intervention. Both quantities are measured against the permutation control. Figure 4 summarizes these effects, with marker shape indicating the backbone. Across all backbones, the modes exhibit the same three broad functional classes, although the specific modes assigned to each class may differ. Exploration modes, shown in orange, primarily increase the entropy of the fed-back distribution. Effort modes, shown in blue, primarily affect continuation length: scaling in one direction lengthens the continuation, while scaling in the opposite direction shortens it. More details are provided in Appendix A.4.

3.2 Accuracy and Efficiency Evaluation

Accuracy evaluation. Table 2 compares our method with selected baselines across five benchmarks. Every entry is averaged over the five backbones, Qwen3-1.7B, Qwen3-4B, Qwen3-8B, Mistral-7B and Llama-3.1-8B. Our method achieves the best average accuracy on four of the five benchmarks, ARC-C, ARC-E, MedQA and HotpotQA. On GSM8K, our method achieves 84.2%, matching Self-Consistency@8, Early-Stopping SC, and Adaptive-Consistency. DeepConf performs 0.6 points better; however, this gain comes from aggregating eight complete sampled reasoning chains, requiring roughly twice the compute used by our method on this benchmark. While sampling-based methods such as Self-Consistency and DeepConf provide strong gains on GSM8K, their improvements are less pronounced on knowledge-intensive and multi-hop tasks. Similarly,

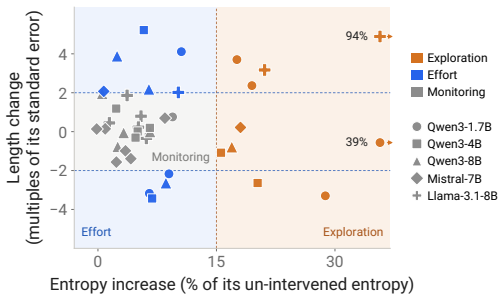


Figure 4: Functional profile of each mode: entropy response (x) versus length response (y), relative to the permutation control. Color gives the role, marker shape the backbone.

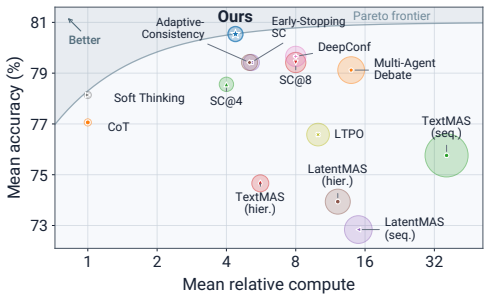


Figure 5: Accuracy-compute trade-off averaged over five benchmarks. Bubble area is proportional to mean compute. Our method lies on the Pareto frontier (dashed).

latent reasoning methods such as Soft Thinking and LTPO show competitive performance on HotpotQA, but tend to underperform on mathematical, and science and medical question answering. Multi-Agent Debate stays less than one point behind Self-Consistency@8 on ARC-C, ARC-E and MedQA. TextMAS and LatentMAS generally underperform CoT on GSM8K, ARC-C, ARC-E, and MedQA. Notably, these gains are averaged over five different backbone models, indicating that the improvement is robust to variations in the underlying model architecture rather than being driven by a particular backbone. Overall, the results demonstrate that our method provides a more general and reliable internal orchestration strategy, achieving strong performance across diverse benchmarks.

Accuracy-compute trade-off. Figure 5 visualizes the accuracy-compute trade-off, with accuracy and task-normalized compute averaged across backbones and benchmarks. Compute is reported for every method relative to one greedy trajectory of its own kind: for text-space methods, the number of decoded tokens divided by the tokens of one greedy chain of thought on the same item; for latent-space methods, including ours, the number of latent steps divided by the steps of one greedy latent trajectory. Both references therefore sit at $1\times$. The per-backbone compute of every method on every benchmark is provided in Appendix A.6. Our method achieves an average accuracy of 80.56% at a mean normalized compute cost of $4.38\times$ the CoT baseline, positioning it on the Pareto frontier in the high-accuracy regime. Compared with Self-Consistency@8 and DeepConf, our method improves average accuracy by 1.14 and 0.90 percentage points, respectively, while reducing mean normalized compute by 45.7%. This advantage also extends to baselines specifically designed to reduce the computational cost of consistency-based inference. Early-Stopping SC and Adaptive-Consistency both achieve 79.42% average accuracy at compute costs of $5.13\times$ and $5.04\times$, respectively. Our method surpasses their accuracy by 1.14 percentage points while requiring 14.6% and 13.1% less compute, respectively. Our method therefore strictly dominates these four baselines in the comparison, achieving higher accuracy with lower computational cost rather than relying on a larger inference budget. Overall, these results demonstrate that our method advances the accuracy-compute Pareto frontier in the five benchmarks, providing a more compute-efficient route to high-accuracy reasoning than the compared high-budget and adaptive consistency baselines.

4 Related Work

External orchestration and multi-agent systems. LLM-based multi-agent systems solve problems by coordinating model invocations through explicit roles and interaction protocols. AutoGen supports configurable conversational agents (Wu et al., 2024). MetaGPT organizes specialized agents around standardized workflows (Hong et al., 2024). Multi-agent debate allows agents to revise their responses after examining one another's reasoning over multiple rounds (Du et al., 2023). Mixture-of-Agents organizes LLM agents into successive layers, with each agent using outputs from the previous layer as context for its own response (Wang et al., 2025). Another line of work optimizes the orchestration structure itself. GPTSwarm represents language agents as computational graphs and optimizes both node-level prompts and graph connectivity (Zhuge et al., 2024). Automated Design of Agentic Systems (ADAS) introduces Meta Agent Search, in which a meta-

agent iteratively writes new agent programs in code, drawing on an archive of previous designs and their evaluation results (Hu et al., 2025). AFlow formulates workflow generation as a search over code-based workflows and uses Monte Carlo Tree Search with execution feedback to improve them (Zhang et al., 2025b). These methods make the arrangement of model calls a target of optimization. Recent work also explores collaboration in latent space. LatentMAS retains agent-level coordination but combines latent reasoning through layer-wise KV caches (Zou et al., 2026). Agent Primitives introduces reusable modules for Review, Voting and Selection, and Planning and Execution, using KV-cache communication among agents within each module (Jin et al., 2026). Across these approaches, the units of orchestration remain model calls or system-level modules, whether their organization is specified manually or optimized automatically. In contrast, we begin from a more fundamental question: rather than prescribing an external inventory of operators, we seek to discover and characterize reusable internal operators that exist within a frozen backbone.

Adaptive computation and inference-time reasoning. Adaptive-depth methods regulate inference cost by controlling which computations are executed for each token. Specifically, Mixture-of-Depths learns to route selected tokens through transformer blocks under a bounded per-layer computation budget, allowing other tokens to bypass these computations (Raposo et al., 2024). Confident Adaptive Language Modeling uses confidence-based early exits to adapt computational depth at individual generation steps (Schuster et al., 2022). These methods allocate computation within a forward pass by controlling which tokens receive additional processing and how many layers are evaluated. Rather than taking architectural blocks as the units of adaptive computation, our framework discovers and orchestrates reusable reasoning operators within a frozen model. This exposes a functional interface to internal computation, allowing inference to be organized around the roles of internal operations rather than the boundaries of network layers.

For inference-time reasoning, chain-of-thought prompting elicits explicit intermediate steps, including in zero-shot settings (Wei et al., 2022; Kojima et al., 2022). Self-consistency samples multiple reasoning paths and aggregates their answers (Wang et al., 2023). Large-scale studies of repeated sampling show that additional samples can substantially improve coverage of correct solutions, while identifying those solutions remains a separate challenge (Brown et al., 2024). Verifier-based methods address this selection problem by learning to rank complete solutions (Cobbe et al., 2021), while process-supervised reward models use intermediate-step feedback to improve the evaluation and selection of reasoning traces (Lightman et al., 2024). Tree of Thoughts extends inference beyond complete-solution selection by exploring and evaluating intermediate textual reasoning states (Yao et al., 2023a). TS-LLM further uses learned value functions with tree-search algorithms, including AlphaZero-style Monte Carlo tree search, to guide decoding (Feng et al., 2024). These methods improve inference by designing mechanisms for generating, evaluating, and selecting reasoning alternatives. Adaptive sampling further determines when additional trajectories are worth generating. Early-stopping Self-Consistency and Adaptive-Consistency use agreement among collected answers to determine when further sampling is unnecessary (Li et al., 2024; Aggarwal et al., 2023). DeepConf uses model-internal confidence signals to filter completed traces or terminate low-confidence traces during generation (Fu et al., 2026). Rather than specifying alternatives only through token- or thought-level expansions, our method optimizes the internal computation that produces each continuation through a reusable, experimentally characterized operator set.

5 Conclusion

We presented a framework for discovering and orchestrating internal operators within a single frozen LLM. By analyzing attention-head writes to the residual stream and applying Causal Mode Probing, we identified a compact set of modes with Exploration, Effort, and Monitoring roles. Then, we propose the Mode-Guided Orchestrator, which uses mode-derived signals to determine when to intervene, invokes mode-specific operators to generate alternative trajectories, and aggregates their answers through sequential consensus. This constructs query-specific computation graphs at inference time without updating the backbone parameters. Our experiments show that the discovered modes span a largely task-invariant subspace within each backbone and exhibit shared functional structure across backbones. Across five backbones and multiple benchmarks covering mathematical, scientific, medical, and multi-hop reasoning, our method lies on the accuracy-compute Pareto frontier among the selected baselines.

AI USE STATEMENT

In this work, we used generative AI tools to polish the writing, assist with code debugging, and search for relevant literature. The authors reviewed all AI-assisted work and take full responsibility for the final content of this paper.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we provide detailed implementation information in the appendix. We will also make our code publicly available.

REFERENCES

Pranjal Aggarwal, Aman Madaan, Yiming Yang, and Mausam. Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with LLMs. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12375–12396, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.761. URL https://aclanthology.org/2023.emnlp-main.761/.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling, 2024. URL https://arxiv.org/abs/2407.21787.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

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, 2021. URL https://arxiv.org/abs/2110.14168.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate, 2023. URL https://arxiv.org/abs/2305.14325.

Xidong Feng, Ziyu Wan, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. Alphazero-like tree-search can guide large language model decoding and training, 2024. URL https://arxiv.org/abs/2309.17179.

Yichao Fu, Xuewei Wang, Hao Zhang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 94355–94377, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/98257285340854262185500e59bc0f28-Paper-Conference.pdf.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, zili wang, Steven Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. Metagpt: Meta programming for a multi-agent collaborative framework. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 23247–23275, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/6507b115562bb0a305f1958ccc87355a-Paper-Conference.pdf.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 21344–21377, 2025. URL

https://proceedings.iclr.cc/paper_files/paper/2025/file/36b7acf6f6010652b3f2a433774a66fe-Paper-Conference.pdf.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14), 2021. ISSN 2076-3417. doi: 10.3390/app11146421. URL https://www.mdpi.com/2076-3417/11/14/6421.

Haibo Jin, Peng Kuang, Ye Yu, Xiaopeng Yuan, and Haohan Wang. Agent primitives: Reusable latent building blocks for multi-agent systems, 2026. URL https://arxiv.org/abs/2602.03695.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan A, Saiful Haq, Ashutosh Sharma, Thomas Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. Dspy: Compiling declarative language model calls into state-of-the-art pipelines. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 54928–54958, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/f1cf02ce09757f57c3b93c0db83181e0-Paper-Conference.pdf.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 22199–22213. Curran Associates, Inc., 2022. doi: 10.52202/068431-1613. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/8bb0d291acd4acf06ef112099c16f326-Paper-Conference.pdf.

Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 14751–14768, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/3fe2a777282299ecb4f9e7ebb531f0ab-Paper-Conference.pdf.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/aca97732e30bcf1303bc22ac3924fd16-Paper-Conference.pdf.

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 Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20275–20321, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1025. URL https://aclanthology.org/2025.emnlp-main.1025/.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models, 2024. URL https://arxiv.org/abs/2404.02258.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. In S. Koyejo, S. Mohamed,

A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 17456–17472. Curran Associates, Inc., 2022. doi: 10.52202/068431-1269. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/6fac9e316a4ae75ea244ddcef1982c71-Paper-Conference.pdf.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 38154–38180. Curran Associates, Inc., 2023. doi: 10.52202/075280-1657. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/77c33e6a367922d003ff102ffb92b658-Paper-Conference.pdf.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling parameters for reasoning. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 10131–10165, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/1b623663fd9b874366f3ce019fdfdd44-Paper-Conference.pdf.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Y Zou. Mixture-of-agents enhances large language model capabilities. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 33944–33963, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/5434be94e82c54327bb9dcaf7fca52b6-Paper-Conference.pdf.

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, 2023. URL https://arxiv.org/abs/2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837. Curran Associates, Inc., 2022. doi: 10.52202/068431-1800. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. Autogen: Enabling next-gen LLM applications via multi-agent conversation. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024. URL https://openreview.net/forum?id=uAjxFFing2.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 50528–50652. Curran Associates, Inc., 2024. doi: 10.52202/079017-1601. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/5a7c947568c1b1328ccc5230172e1e7c-Paper-Conference.pdf.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question

answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

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. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 11809–11822. Curran Associates, Inc., 2023a. doi: 10.52202/075280-0517. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/271db9922b8d1f4dd7aaef84ed5ac703-Paper-Conference.pdf.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023b. URL https://arxiv.org/abs/2210.03629.

Wengao Ye, Yan Liang, and Lianlei Shan. Thinking on the fly: Test-time reasoning enhancement via latent thought policy optimization. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 156213–156239, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/fce2d8a485746f76aac7b5650db2679d-Paper-Conference.pdf.

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. Multi-agent architecture search via agentic supernet. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 75834–75852. PMLR, 13–19 Jul 2025a. URL https://proceedings.mlr.press/v267/zhang25bi.html.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, XiongHui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. Aflow: Automating agentic workflow generation. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 34040–34077, 2025b. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/5492ecbce4439401798dcd2c90be94cd-Paper-Conference.pdf.

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, and Xin Wang. Soft thinking: Unlocking the reasoning potential of llms in continuous concept space. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 168990–169012. Curran Associates, Inc., 2025c. doi: 10.52202/085713-5629. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/f7396d1c54d51416958d63e285377103-Paper-Conference.pdf.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. Gptswarm: language agents as optimizable graphs. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Jiaru Zou, Ruizhong Qiu, Gaotang Li, Xiyuan Yang, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, James Zou, Mengdi Wang, and Ling Yang. Latent collaboration in multi-agent systems. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=syG9I9ofd8.

A APPENDIX

This appendix reports the experimental settings, baseline protocols, the details of the design, detailed experimental results, and limitation discussion.

A.1 EXPERIMENTAL SETTINGS

Models and benchmarks. The experiments use Qwen3-1.7B, Qwen3-4B, Qwen3-8B, Mistral-7B-Instruct-v0.3 and Llama-3.1-8B-Instruct. Tables abbreviate the last two as Mistral and Llama. Table 3 lists the counts and the per-benchmark limits.

Table 3: Evaluation items and per-benchmark limits. The text-token limit applies to the text-space baselines, the latent-step ceiling to the latent-space baselines and to ours. The HotpotQA subset is the first 500 items of the evaluation ordering.

Benchmark	Evaluated items	Max. text tokens	Latent-step ceiling
GSM8K	1,300	1,024	1,024
ARC-Challenge	1,172	1,024	1,024
ARC-Easy	2,376	1,024	1,024
MedQA-USMLE	1,273	1,024	1,024
HotpotQA-Distractor	500	1,024	1,024

Latent loop. The loop of equation 1 feeds back the top-k_{top} mixture at temperature T; both k_{top} and T are hyperparameters of the loop and can be adjusted. We use $T{=}0.5$ and $k_{\text{top}}{=}5$ on all benchmarks except HotpotQA, where the fed-back distribution is taken over the full vocabulary at $T{=}2$ on Qwen3-1.7B, Qwen3-4B and Mistral; on Qwen3-8B and Llama, HotpotQA uses the same $T{=}0.5$, $k_{\text{top}}{=}5$ setting as the other benchmarks. A trajectory ends when the fed-back distribution places its top-1 mass on the answer terminator, or at the latent-step ceiling of Table 3. The answer is read out from the final state by one short greedy decode.

Hyperparameters. The adjustable hyperparameters of our design include: the number of modes kept per backbone, M; the strength α and the duration k of the rescaling in equation 4; the earliest latent step at which the orchestrator may take its snapshot; the required vote lead ℓ; and the branch budget $K_{\max}$. We use $M{=}8$, $\alpha{=}1$, $k{=}8$, the first boundary at or after latent step 24, $\ell{=}3$ and $K_{\max}{=}16$. The same values are used on every backbone and benchmark.

A.2 BASELINES

Text-space methods. CoT is one greedy text continuation. The sampling-based baselines share an ordered bank of eight continuations per item, sampled at temperature 0.6 and top-$p{=}0.95$ within the text-token limits of Table 3. SC@4 and SC@8 take the majority over the first four or eight samples. ESC draws the bank in windows of four and stops when a window is unanimous, up to eight. Adaptive-Consistency stops when the Beta-posterior probability that the current majority is final exceeds 0.95, up to eight. DeepConf (offline) is run with the same bank of eight traces as SC@8 (the original uses hundreds). Each trace is scored by its group confidence, the mean over sliding windows of the mean negative log-probability of the top-ten tokens; with our 1,024-token traces the 2,048-token window covers the whole trace, so the score reduces to the trace's mean confidence. The bottom 10% of traces (one of eight) is discarded and a confidence-weighted majority is taken, i.e. the paper's DeepConf-high filter at $N{=}8$.

Latent-space methods. Soft Thinking is re-implemented inside our loop with its published settings, a top-10 mixture at $T{=}0.6$ and a cold stop when the feedback entropy stays below 0.01 for 256 steps, with no branching. Its fed-back mixture is norm-matched to the mean input-embedding norm as in the rest of ours. LTPO uses the official code with the configuration of its official example script (8 thought tokens, 10 optimization steps, $\sigma{=}4$, learning rate 0.04). Each of the ten optimization steps calls the model to generate a scoring response of up to 2,048 tokens for the reward, followed by a backward pass; the final answer is one further generation. The official code does not instrument compute, so we report the lower bound of $10\times$ (ten full decodes per item).

Multi-agent systems. Multi-Agent Debate runs three agents for two rounds at T=0.6, top-p=0.95, and takes the majority over the final round. For TextMAS and LatentMAS with Qwen3-4B and Qwen3-8B on GSM8K, ARC-Easy, and MedQA, we report the results from Zou et al. (2026). All remaining TextMAS and LatentMAS results were obtained by us using the official code. The compute of Multi-Agent Debate and TextMAS counts the tokens decoded by all agents; that of LatentMAS counts the judger's decoded tokens plus the latent steps of the other three agents, both relative to CoT.

A.3 Mode extraction and generalization

The principal cosines between two mode sets. Let $\mathbf{U}_A, \mathbf{U}_B \in \mathbb{R}^{N\times 8}$ hold the head loadings of two mode sets as columns, and let $\xi_1, \ldots, \xi_8$ be the singular values of $\mathbf{U}_A^\top \mathbf{U}_B$, the principal cosines between the two sets. The mean principal cosine reported throughout Section 3.1 is

$$\bar{\xi}(A, B) = \frac{1}{8}\sum_{j=1}^{8} \xi_j. \tag{11}$$

Two mode sets are compared as sets rather than mode by mode because individual singular vectors are not stable when neighboring singular values are close: a small perturbation of the data can rotate the vectors within the plane the two modes span, or exchange their rank, while the plane itself is stable. A mode-by-mode cosine would charge such a rotation to the model; the mean principal cosine of equation 11 is invariant to it. The correspondence nevertheless holds mode by mode for all but a few modes. On the fifteen backbone-by-benchmark cells of GSM8K, ARC-C and HotpotQA, 116 of the 120 modes (eight per cell) match their counterpart from the other half at $|\cos| \geq 0.93$, with a median of 0.993. Two of the four exceptions are modes 3 and 4 (zero-based, as in Table 6) of Qwen3-8B on GSM8K, whose singular values differ by 0.17%: they exchange rank between the two halves while the plane they span is preserved (cosine 0.99 with the exchanged partner). The other two are modes 3 and 4 of Qwen3-1.7B on ARC-C, which match their own counterparts at 0.928, just below the threshold. In every other cell the singular-value gap at that position is at least 3%. Referring to modes by their index, as Table 6 does, is therefore well defined.

The mean principal cosine is bounded in $[0, 1]$, but the value reached by two unrelated mode sets is not zero: it depends on the head count N, which differs across backbones. Table 4 gives the value for two random sets of eight orthonormal loading vectors; this is the 0.07-0.11 range quoted in Section 3.1.

Table 4: Chance level of the mean principal cosine for two random mode sets.

Backbone	Layers × heads	N	Chance level
Qwen3-1.7B	28 × 16	448	0.110
Qwen3-4B	36 × 32	1,152	0.069
Qwen3-8B	36 × 32	1,152	0.070
Mistral	32 × 32	1,024	0.075
Llama	32 × 32	1,024	0.073

Table 5 shows the numbers used to plot Figure 3: the mean principal cosine between the mode sets extracted on two benchmarks, for all ten benchmark pairs and each of the five backbones.

A.4 Causal Mode Probing

Intervention. For mode m the loadings are normalized as $\tilde{\mathbf{u}}_m = \mathbf{u}_m / \|\mathbf{u}_m\|_\infty$ and every head r is rescaled by

$$\gamma_{m,s,r} = \max\big(0,\ 1 + s\,\alpha\,\tilde{u}_{m,r}\big), \qquad s \in \{-1, +1\}, \quad \alpha = 1, \quad k = 8, \tag{12}$$

for the first eight latent steps after a saved boundary, or until the trajectory terminates if that comes first; afterwards the unmodified model runs to termination. With α=1 the gains lie in $[0, 2]$. Because a loading vector has entries of both signs, $s = +1$ does not amplify every participating head: the two signs are the two loading-aligned directions of the intervention, not uniform increases or decreases of attention output.

Table 5: The numbers plotted in Figure 3: the mean principal cosine between the mode sets extracted on two benchmarks, for all ten pairs, one column per backbone. "Q" denotes Qwen3.

Benchmark pair		Q-1.7B	Q-4B	Q-8B	Mistral	Llama
GSM8K	ARC-C	0.938	0.955	0.958	0.907	0.941
	ARC-E	0.936	0.955	0.956	0.920	0.939
	MedQA	0.911	0.922	0.934	0.930	0.890
	HotpotQA	0.915	0.917	0.897	0.867	0.889
ARC-C	ARC-E	0.996	0.996	0.996	0.994	0.994
	MedQA	0.867	0.933	0.940	0.935	0.928
	HotpotQA	0.906	0.908	0.893	0.922	0.899
ARC-E	MedQA	0.873	0.935	0.942	0.940	0.929
	HotpotQA	0.905	0.912	0.892	0.913	0.890
MedQA	HotpotQA	0.861	0.898	0.874	0.887	0.865

Control. The control starts from the same saved boundary and applies $\boldsymbol{\gamma}^{\mathrm{ctrl}}_{m,s} = \mathbf{P}\,\boldsymbol{\gamma}_{m,s}$, where $\mathbf{P}$ permutes the gains across all N heads. One permutation is drawn per boundary and mode, seeded by the item and boundary, and shared by the two signs, so probe and control are paired at every level. The control preserves the multiset of gains, the number of non-unit gains, and every permutation-invariant norm of $\boldsymbol{\gamma} - \mathbf{1}$. It does not preserve the norm of the residual perturbation itself, since write norms differ across heads: at a layer with head writes $\mathbf{X}_l \in \mathbb{R}^{H\times d}$ the perturbation is $\boldsymbol{\delta}_l = \mathbf{X}_l^\top(\boldsymbol{\gamma}_l - \mathbf{1})$, whose energy depends on which heads receive which gain. Therefore, the contrast evaluates the assignment of a fixed gain multiset to the mode's heads against a random assignment of the same multiset.

Measurements. Let b index a boundary. For a continued trajectory, H_b is the mean entropy of the fed-back next-token distribution over all latent steps after the boundary, and τ_b is the number of latent steps after the boundary. The paired differences are

$$D^H_{b,m,s} = H^{\mathrm{mode}}_{b,m,s} - H^{\mathrm{ctrl}}_{b,m,s}, \qquad D^\tau_{b,m,s} = \tau^{\mathrm{mode}}_{b,m,s} - \tau^{\mathrm{ctrl}}_{b,m,s}. \tag{13}$$

The directional length response is the antisymmetric part of D^τ across the two signs,

$$A_{b,m} = \tfrac{1}{2}\big(D^\tau_{b,m,+} - D^\tau_{b,m,-}\big), \qquad \overline{A}_m = \frac{1}{B}\sum_b A_{b,m}, \qquad t^A_m = \frac{\overline{A}_m}{s(A_{\cdot,m})/\sqrt{B}}, \tag{14}$$

with $s(\cdot)$ the sample standard deviation over boundaries; the entropy statistics $t^H_{m,s}$ are formed in the same way from $D^H_{b,m,s}$. Entropy effects are reported relative to the backbone, $E_m = 100\max_s \overline{D}^H_{m,s}/H_0$, where H_0 is the backbone's mean fed-back entropy at an unmodified boundary: 0.134, 0.128, 0.157, 0.401 and 0.437 for Qwen3-1.7B, Qwen3-4B, Qwen3-8B, Mistral and Llama. Baseline entropies span $3.4\times$, so an absolute threshold would label every Mistral and Llama mode Exploration. E_m and t^A_m are the two axes of Figure 4.

Table 6: Role of each mode on GSM8K fork boundaries. Mode indices are zero-based in order of singular value. E: Exploration; F: Effort; M: Monitoring. B is the number of boundaries.

		Mode index							
Backbone	B	0	1	2	3	4	5	6	7
Qwen3-1.7B	275	E	E	E	E	M	F	F	F
Qwen3-4B	618	M	M	E	M	F	E	M	F
Qwen3-8B	192	M	M	E	M	F	F	F	M
Mistral	293	M	M	E	M	F	M	M	M
Llama	409	M	M	E	F	M	M	E	M

Evidence for the Monitoring role. Monitoring is defined by the absence of a response on either measurement, so those modes sit near the origin of Figure 4 by construction. What justifies the name is what the modes carry when they are read rather than driven. A logistic regression on the activity of a backbone's Monitoring modes at the early fork, fitted on that backbone's GSM8K training boundaries with wrong and right trajectories balanced and then frozen, predicts whether the single trajectory will end wrong. On four of the five backbones, reading the Monitoring modes alone recovers the AUROC of reading all eight.

A.5 Orchestrator: decision rules

Operator pool and scheduling. The operator pool is $\mathcal{O} = \{(m, s) : m = 0, \ldots, 7;\ s \in \{-1, +1\}\}$, the eight modes in both signs. The schedule that decides which operator to try next could be made elaborate: it could rank operators by their role, condition on the state of the main trajectory, or be trained separately for each benchmark and backbone. We choose a simple yet effective schedule that follows the roles: Exploration operators are drawn first, then Effort, then Monitoring, in a seeded random order within each role. Every branch restores the snapshot, applies equation 12 for k latent steps, and runs the unmodified model to termination; if the loop emits the answer terminator during those k steps, the branch ends there.

Sequential consensus. After j branches the count of answer a is

$$n_j(a) = \mathbb{I}[a_0 = a] + \sum_{i=1}^{j} \mathbb{I}[a_i = a], \tag{15}$$

where answers are canonicalized before comparison, and a branch that produces no answer casts no vote. Let $n_j^{(1)}$ and $n_j^{(2)}$ be the largest and second-largest counts, with $n_j^{(2)}{=}0$ when only one answer has appeared. The orchestrator stops at

$$J = \min\Big(\{\, j \le K_{\max} :\ n_j^{(1)} - n_j^{(2)} \ge \ell \,\} \cup \{K_{\max}\}\Big), \tag{16}$$

and returns the leading answer, with ties resolved in favor of a_0. $K_{\max}{=}16$; when the main trajectory itself hit the latent-step ceiling, the budget is reduced to 4. We use $\ell{=}3$. The compute of an item is $1 + \sum_{i \le J}(\tau_i - t_{\mathrm{f}})/\tau_0$, the branches' latent steps after the fork relative to the main trajectory, with the snapshot free.

A.6 Per-backbone results

Tables 7-11 report the compute of every method on every benchmark, one table per backbone. Compute is measured relative to one greedy trajectory of the method's own kind: for text-space methods, decoded tokens divided by the tokens of one greedy chain of thought (for Multi-Agent Debate and TextMAS, the tokens generated by all agents; for LatentMAS, the judger's tokens plus the 30 latent steps of the other agents); for latent-space methods and ours, latent steps divided by the steps of one greedy latent trajectory. LTPO is not instrumented by its official code and is listed at its lower bound of ten full decodes per item. On HotpotQA the text-space baselines answer with a short extractive prompt, so their greedy reference is only 4-7 tokens long; methods that generate a reasoning chain before answering (Multi-Agent Debate, TextMAS, LatentMAS) therefore show ratios of 10-400$\times$ on this benchmark.

A.7 Discussion of Potential Limitations and Future Directions

There are several directions that could further improve the proposed framework for internal orchestration. A key direction is operator scheduling. More elaborate schedules could condition operator selection on the state of the main trajectory or be trained separately for each benchmark and backbone. Such schedules could prioritize operators according to the needs of each query, potentially improving the accuracy-compute trade-off. In addition, the scope of operator discovery could be extended beyond attention heads to include other model components.

Table 7: Compute on Qwen3-1.7B.

Method	GSM8K	ARC-C	ARC-E	MedQA	HotpotQA
CoT	1.00	1.00	1.00	1.00	1.00
Self-Consistency@4	4.03	4.03	4.00	4.08	4.03
Self-Consistency@8	8.06	8.07	8.00	8.19	8.05
Early-Stopping SC	5.40	5.00	4.50	6.20	4.40
Adaptive-Consistency	5.30	4.90	4.50	6.10	4.40
DeepConf	8.06	8.07	8.00	8.19	8.05
Soft Thinking	1.00	1.00	1.00	1.00	1.00
LTPO	$\geq$10	$\geq$10	$\geq$10	$\geq$10	$\geq$10
Multi-Agent Debate	5.88	6.14	6.04	6.01	8.62
TextMAS, sequential	2.87	2.78	7.53	2.24	53.33
TextMAS, hierarchical	2.46	0.92	7.82	1.17	6.46
LatentMAS, sequential	2.11	1.62	1.50	1.46	97.92
LatentMAS, hierarchical	2.56	2.12	1.87	2.07	88.96
Ours	4.12	3.50	3.13	4.95	14.81

Table 8: Compute on Qwen3-4B.

Method	GSM8K	ARC-C	ARC-E	MedQA	HotpotQA
CoT	1.00	1.00	1.00	1.00	1.00
Self-Consistency@4	3.99	3.98	4.00	4.03	4.06
Self-Consistency@8	8.00	7.96	8.00	8.06	8.15
Early-Stopping SC	4.60	4.40	4.20	5.80	4.90
Adaptive-Consistency	4.50	4.40	4.20	5.70	4.80
DeepConf	8.00	7.96	8.00	8.06	8.15
Soft Thinking	1.00	1.00	1.00	1.00	1.00
LTPO	$\geq$10	$\geq$10	$\geq$10	$\geq$10	$\geq$10
Multi-Agent Debate	5.45	5.97	6.08	5.48	33.43
TextMAS, sequential	10.80	2.91	9.67	6.44	63.23
TextMAS, hierarchical	10.54	1.28	8.21	10.90	10.90
LatentMAS, sequential	2.17	1.90	2.44	2.79	53.10
LatentMAS, hierarchical	1.99	1.79	1.57	1.70	58.62
Ours	3.28	3.10	2.93	4.36	3.40

Table 9: Compute on Qwen3-8B.

Method	GSM8K	ARC-C	ARC-E	MedQA	HotpotQA
CoT	1.00	1.00	1.00	1.00	1.00
Self-Consistency@4	3.96	4.02	4.01	4.06	4.82
Self-Consistency@8	7.91	8.03	8.00	8.09	8.87
Early-Stopping SC	4.40	4.30	4.20	5.70	5.70
Adaptive-Consistency	4.40	4.30	4.10	5.60	5.60
DeepConf	7.91	8.03	8.00	8.09	8.87
Soft Thinking	1.00	1.00	1.00	1.00	1.00
LTPO	$\geq$10	$\geq$10	$\geq$10	$\geq$10	$\geq$10
Multi-Agent Debate	6.07	6.52	6.60	5.74	52.08
TextMAS, sequential	8.37	2.88	7.53	6.76	81.76
TextMAS, hierarchical	8.54	2.24	8.08	10.93	19.56
LatentMAS, sequential	3.21	1.84	1.88	2.51	75.49
LatentMAS, hierarchical	1.38	1.94	1.22	1.64	81.40
Ours	3.11	3.01	2.84	4.22	3.34

Table 10: Compute on Mistral.

Method	GSM8K	ARC-C	ARC-E	MedQA	HotpotQA
CoT	1.00	1.00	1.00	1.00	1.00
Self-Consistency@4	3.79	4.12	4.05	3.94	4.05
Self-Consistency@8	7.57	8.23	8.13	7.88	8.12
Early-Stopping SC	7.10	4.80	4.50	5.60	6.00
Adaptive-Consistency	7.00	4.70	4.40	5.40	6.00
DeepConf	7.57	8.23	8.13	7.88	8.12
Soft Thinking	1.00	1.00	1.00	1.00	1.00
LTPO	$\geq$10	$\geq$10	$\geq$10	$\geq$10	$\geq$10
Multi-Agent Debate	6.61	8.11	8.34	6.73	11.52
TextMAS, sequential	5.82	14.56	16.73	7.75	115.34
TextMAS, hierarchical	3.15	2.10	2.03	1.55	4.63
LatentMAS, sequential	1.13	2.19	2.41	1.32	30.99
LatentMAS, hierarchical	0.89	1.18	1.24	0.59	10.54
Ours	7.98	4.01	3.26	5.42	5.97

Table 11: Compute on Llama.

Method	GSM8K	ARC-C	ARC-E	MedQA	HotpotQA
CoT	1.00	1.00	1.00	1.00	1.00
Self-Consistency@4	4.06	3.91	3.96	3.97	4.10
Self-Consistency@8	8.13	7.84	7.92	7.95	8.18
Early-Stopping SC	5.40	5.30	4.90	5.80	5.20
Adaptive-Consistency	5.30	4.90	4.60	5.70	5.20
DeepConf	8.13	7.84	7.92	7.95	8.18
Soft Thinking	1.00	1.00	1.00	1.00	1.00
LTPO	$\geq$10	$\geq$10	$\geq$10	$\geq$10	$\geq$10
Multi-Agent Debate	6.83	5.67	5.73	5.14	117.31
TextMAS, sequential	7.27	8.08	8.67	6.11	444.53
TextMAS, hierarchical	3.25	2.13	0.81	2.60	8.12
LatentMAS, sequential	1.14	1.07	1.10	0.89	79.82
LatentMAS, hierarchical	1.04	0.76	0.74	0.46	36.26
Ours	4.12	3.76	3.11	5.04	2.70