Geometric Prompt-Trajectory Planning for Test-Time Scaling

Zhengqi Pei[1,2], **Anran Zhang**[2], **Qingming Huang**[2], **Shuhui Wang**[1*]
[1]State Key Lab. of AI Safety, Institute of Computing Technology,
Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences
{peizhengqi22, zhanganran22}@mails.ucas.ac.cn
qmhuang@ucas.ac.cn, wangshuhui@ict.ac.cn

Abstract

Test-time scaling (TTS) improves large language model (LLM) reasoning by allocating additional inference-time computation, often through repeated sampling and aggregation. This width-heavy strategy increases the generation cost and can reinforce common failure modes. We introduce *Maze*, a TTS framework that shifts part of this computation from repeated target-model calls to planning over ordered few-shot exemplars. The central idea is to treat an ordered exemplar sequence as a *prompt trajectory* embedded in a latent geometric space. Instead of asking the LLM for many independent attempts, *Maze* first ranks candidate prompt trajectories using cached encoder representations and a path scorer, *i.e.*, a lightweight learnable metric, then spends one call (or a few calls) only on the most promising trajectories. Directed transitions, node contributions, and query-conditioned cross-layer fusion model how the complete path guides the target model. The selected trajectory is executed with one call, or with a small call budget in MultiMaze. A separate frozen encoder preserves black-box access to the target LLM. Across five reasoning benchmarks, single-call Maze outperforms the evaluated single-call prompt-optimization baselines, while MultiMaze improves on 32-call majority vote using 16 target-model calls. A theoretical analysis relates selection quality to candidate coverage and ranking error, explaining when planner-side computation can substitute for repeated sampling and when it cannot. The code is available at https://github.com/Chipflow-pzq/Maze.

1 Introduction

Test-time scaling (TTS) can elicit stronger reasoning from large language models (LLMs). Beyond a single chain-of-thought (CoT) trajectory [Wei et al., 2022], additional computation can support repeated sampling, self-consistency, and structured search over intermediate thoughts [Wang et al., 2022, Yao et al., 2023, Besta et al., 2024]. These approaches spend a large part of their budget on target-model generation. Repeated calls increase total decoding work, while their accuracy gains depend on query difficulty and on whether the sampled answers reinforce useful evidence or a common failure mode [Chen et al., 2023, 2024]. This raises a complementary question: *can computation spent choosing a prompt reduce the number of reasoning attempts needed from the target model?*

The question is motivated by the sensitivity of in-context learning to both exemplar content and exemplar order [Lu et al., 2021, Guo et al., 2024]. Retrieval, prompt-flatness criteria and ordering-aware optimization already improve over random prompting [Rubin et al., 2021, Shen et al., 2023].

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

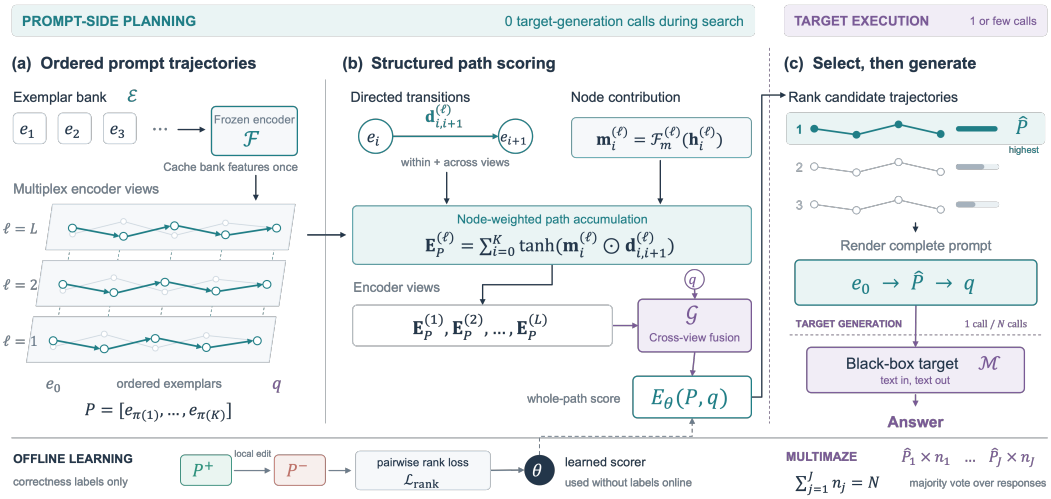


Figure 1: **Maze plans the prompt before target-model generation.** (a) Cached encoder views represent ordered exemplar trajectories from the system instruction e_0 to the query q. (b) Directed transitions, node contributions, and query-conditioned cross-layer fusion produce a whole-path score. (c) The highest-scoring trajectory becomes a complete prompt for the black-box target. Correctness feedback supervises the scorer offline; online ranking uses only input-side features. MultiMaze allocates N generation calls to selected trajectories and aggregates their answers by majority vote.

However, these approaches typically optimize pointwise relevance, a robustness proxy, or a task-level exemplar sequence. A query-conditioned score for the *complete ordered sequence* offers a different opportunity: to predict which combination of exemplars and transitions will be useful before paying for the corresponding generation. The relevant object is therefore not just an exemplar set, but a prompt trajectory.

We introduce *Maze* [2], a framework that treats few-shot prompting as *prompt-trajectory planning*. A trajectory connects the system instruction to the query through an ordered sequence of exemplars. Maze represents its nodes using cached features from a frozen encoder, scores directed transitions within and across encoder layers, and combines their contributions through query-conditioned fusion. The resulting lightweight scorer ranks candidate trajectories before the target LLM is invoked. The single-call Maze executes one selected prompt; MultiMaze applies the same principle under a small multi-call budget.

The distinction from thought-space search is operational. Tree of Thoughts and related approaches expand or evaluate generated textual states online [Yao et al., 2023, Besta et al., 2024]; Maze searches over *inputs* to the target model using cached representations. No target-generation call is required to score a candidate path. The encoder and the target have separate roles, so planning does not require access to the target's weights, gradients, or hidden states. This separation also preserves the structural distinction from pointwise retrieval: the score depends on the joint ordered path, not only on each exemplar's relevance to the query.

Our theoretical analysis identifies when this allocation of computation is useful. Coverage determines whether the explored pool contains a high-utility path, while rankability determines whether the scorer can select it. As a result, they bound the regret of one-call prompt selection and characterize a conditional conversion between the target-model sampling width and the planner-side search. The analysis distinguishes an oracle over prompt success probabilities from an oracle over realized answers under stochastic decoding.

We evaluate Maze on MMLU-Pro [Wang et al., 2024], GPQA [Rein et al., 2024], GSM8K [Cobbe et al., 2021], MATH500 [Hendrycks et al., 2021], and AIME [Veeraboina, 2023], using Qwen3 [Yang et al., 2025], Mistral [Jiang et al., 2023], and LLaMA3 [AI@Meta, 2024] models. Single-call Maze improves over the evaluated prompt-optimization baselines, and MultiMaze improves the accuracy–

[2] We use the maze metaphor as an intuition for structured search: a prompt is not just a set of examples, but an ordered path whose quality depends on the route, not only the destination.

cost trade-off of repeated sampling. Shared-pool controls separate prompt selection from answer aggregation, and proxy-encoder experiments demonstrate deployment on closed-source targets. Our contributions are as follows:

- **Prompt-trajectory planning.** We formulate a test-time decision over complete ordered exemplar sequences before target-model generation.
- **A compact geometric scorer.** Directed transitions, node contributions, and cross-layer fusion model query-conditioned path utility using reusable encoder features.
- **Conditional theoretical guarantees.** Coverage and ranking error bound selection regret and specify the scope of width–depth conversion.
- **Empirical evaluation.** Single-call and multi-call results, matched-pool controls, and black-box transfer characterize the benefits and limits of prompt-side planning.

Appendix D discusses related work; Appendix E details concrete instantiations of the method.

2 Methodology: Maze-guided reasoning

2.1 Problem setup and prompt paths

Let $\mathcal{M}$ denote a *target* LLM accessed via a standard completion API. We are given a candidate pool of exemplars $\mathcal{E}=\{e_i=(q_i,a_i)\}_{i=1}^{|\mathcal{E}|}$ and a test query q. A *prompt path* of length K is an ordered sequence $P=[e_{\pi(1)},\dots,e_{\pi(K)}]$, which induces a few-shot prompt $e_0 \Rightarrow e_{\pi(1)} \Rightarrow \cdots \Rightarrow e_{\pi(K)} \Rightarrow q$ with e_0 as the system prompt. A prompt renderer assembles the system instruction e_0, these exemplars, and q into $\mathrm{Prompt}(q,P)$. We seek paths that increase the probability of a correct answer under $\mathcal{M}$ within a fixed target-call budget. Exemplar content and order are decision variables; the target's weights and decoding procedure remain fixed. Different exemplar paths induce different reasoning behaviors, even for the same target query and the same base model. *Maze* treats this dependence as the core object to optimize.

2.2 Black-box contract: decoupling encoder from LLM

A separate frozen encoder $\mathcal{F}$ represents each input node across L views,

$$\mathcal{F}(x) = \{\mathbf{h}^{(l)}(x)\}_{l=1}^{L}, \qquad \mathbf{h}^{(l)}(x) \in \mathbb{R}^{d_l}. \tag{1}$$

Views may correspond to transformer layers or embedding heads; a fixed pooling rule summarizes the token states for each view. The node representation supplies features for scoring, while the renderer supplies the selected exemplars' complete text for generation. Bank features are cached and each new query is encoded once. The target LLM $\mathcal{M}$ supplies answers, whereas the encoder $\mathcal{F}$ supplies features; neither is fine-tuned. Using $\mathcal{F}=\mathcal{M}$ is possible for open-weight targets, but a separate encoder supports API-only deployment without target-model internals.

2.3 Reasoning as a *Maze* path

The *Mazr* is an external representation of the prompt space, rather than a reconstruction of the target's internal reasoning. Its nodes are the system instruction, exemplars, and query; its directed edges encode their order. For scoring, write the complete path as $(e_0,e_1,\dots,e_K,e_{K+1})$ with $e_{K+1}=q$. Planning evaluates alternative paths in this space and returns prompts to execute, without generating intermediate thoughts or consulting the test answer.

2.4 A multiplex metric geometry over prompt paths

Maze assigns each path a scalar *energy* $E_\theta(P,q)$, a query-conditioned ranking score computed from encoder features. The scorer preserves three forms of structure: directed transitions, source-node contributions, and interactions across encoder views.

Intra-layer and cross-layer relations. Let $\mu^{(l)}:\mathbb{R}^{d_l}\times\mathbb{R}^{d_l}\to\mathbb{R}^r$ and $\mu^{(j,l)}:\mathbb{R}^{d_j}\times\mathbb{R}^{d_l}\to\mathbb{R}^r$ be relation maps into a common low-dimensional space. For a neighborhood $\mathcal{N}(l)$ of other encoder

views, define

$$\mathbf{d}_{x,y}^{(l)} = \mu^{(l)}\big(\mathbf{h}_x^{(l)}, \mathbf{h}_y^{(l)}\big) + \sigma\left(\sum_{j\in\mathcal{N}(l)} \mu^{(j,l)}\big(\mathbf{h}_x^{(j)}, \mathbf{h}_y^{(l)}\big)\right), \tag{2}$$

where σ is a coordinate-wise nonlinearity with $\sigma(0) = 0$. For ordered layers, $\mathcal{N}(l)$ can be a local window restricted to available views. The relations are directed: reversing an edge changes its role in the prompt. They are learned transition features, not distances constrained by symmetry or the triangle inequality.

Node-weighted path energy. A node mapping $\mathscr{F}_m^{(l)} : \mathbb{R}^{d_l} \to \mathbb{R}^r$ produces $\mathbf{m}_i^{(l)} = \mathscr{F}_m^{(l)}(\mathbf{h}_i^{(l)})$, the source node's contribution to its outgoing transition. Using $\odot$ for elementwise multiplication, the layerwise path representation is

$$\mathbf{E}_P^{(l)} = \sum_{i=0}^{K} \sigma\Big(\mathbf{m}_i^{(l)} \odot \mathbf{d}_{i,i+1}^{(l)}\Big) \in \mathbb{R}^r. \tag{3}$$

Although transitions are summed, this representation is order-sensitive: changing exemplar order changes the directed edges and their source-node weights. Unlike an unweighted sum of linear displacements, this accumulation can distinguish paths with the same endpoints.

Query-conditioned fusion. The fusion map $\mathcal{G} : \mathbb{R}^r \times \mathbb{R}^r \to \mathbb{R}^r$ weights a path representation using the query node. Write $\mathbf{u}_{lj} = \mathcal{G}(\mathbf{m}_{K+1}^{(l)}, \mathbf{E}_P^{(j)}) \odot \mathbf{E}_P^{(j)}$. A channel reduction $\mathcal{C} : \mathbb{R}^r \to \mathbb{R}^s$ either retains the coordinates ($s = r$) or sums them ($s = 1$). For fusion neighbors $\mathcal{J}(l) \subseteq \mathcal{N}(l)$, let

$$\mathbf{F}_P^{(l)} = \mathcal{C}(\mathbf{u}_{ll}) + \sigma\left(\sum_{j\in\mathcal{J}(l)} \mathcal{C}(\mathbf{u}_{lj})\right). \tag{4}$$

The complete score is

$$E_\theta(P, q) = \mathcal{R}_\theta\left(\sum_{l\in\mathcal{A}} \alpha_l \mathbf{F}_P^{(l)}; q\right), \tag{5}$$

where $\mathcal{A}$ indexes contributing views, α_l are fixed aggregation weights, and $\mathcal{R}_\theta$ reads out a scalar from the fused representation and fixed input-side context for q. Scalar fusion uses coordinate sums and an identity readout; vector fusion retains coordinates for a learned readout. Boundary views can contribute within-view terms or supply context to interior views. Both constructions condition the whole-path score on the query and couple information across views.

Compact parameterization. The learned mappings are

$$\theta = \{\mu^{(l)}, \mu^{(j,l)}, \mathscr{F}_m^{(l)}, \mathcal{G}, \mathcal{R}_\theta\}. \tag{6}$$

Low-rank linear projections and a small fusion MLP give a compact realization of these operators. When views share a common width, the relation and node projections can share parameters across layers. Appendix E.2 gives scalar and vector implementations, including the projection, normalization, and boundary choices.

2.5 Learning the *Maze* offline

We learn path utility from correctness observations on training prompts. The target model $\mathcal{M}$ generates the responses used for supervision; the frozen encoder $\mathcal{F}$ supplies only representations. Training records associate a query q and path P with a correctness target $y \in [0, 1]$, obtained from one response or the empirical mean of repeated responses. Within-query comparisons train the scorer to prefer paths with higher observed utility. For a collection $\mathcal{D}_{\text{pair}}$ of such comparisons, the ranking objective has the form

$$\mathcal{L}_{\text{rank}} = \mathbb{E}_{(q,P_i,P_j,y_i,y_j)\sim\mathcal{D}_{\text{pair}}}\left[\ell_{\text{rank}}(E_\theta(P_i, q) - E_\theta(P_j, q), y_i, y_j)\right], \tag{7}$$

where ℓ_{rank} penalizes disagreement between score differences and correctness preferences. Direct success prediction supplies an additional pointwise objective when fitting correctness targets.

Local edits provide focused comparisons: replacing an exemplar or swapping two positions can produce a correct–incorrect pair while holding much of the prompt fixed. Comparisons against a

fixed reference prompt instead use the available outcome records for each query. Both supervise complete paths; local edits additionally identify specific content or order changes associated with different responses. Pair construction, logistic objectives, and optimization settings are detailed in Appendix E.3.

Only training queries and exemplars enter scorer fitting, and a query's own exemplar is excluded from its candidate bank. Validation supports model selection and, where specified, adaptation to a target API. Test answers never enter training or prompt selection. At inference, the fitted scorer uses only the unlabeled query and cached input representations.

2.6 Test-time deployment and *MultiMaze*

Single-call *Maze*. Given q, an input-side proposal constructs a pool $\mathcal{P}(q)$ of M_{pool} complete paths. *Maze* scores them using the cached bank and query features, selects $\widehat{P} = \arg\max_{P \in \mathcal{P}(q)} E_\theta(P, q)$, and executes only that prompt. Pool construction can combine sampled paths with a retrieved reference; it requires no target generation. The selected path maximizes the learned score within the explored pool, whose coverage and ranking reliability govern its utility.

Multi-call allocation. *MultiMaze* allocates N target calls across selected trajectories. Let $\widehat{P}_1, \ldots, \widehat{P}_J$ be the distinct selected paths and $n_j \geq 1$ the number of requests assigned to path j. With $a_{j,t}$ the parsed answer from its tth request, aggregation over all N responses gives

$$\widehat{a} = \text{Agg}((a_{j,t})_{j,t}), \qquad \sum_{j=1}^{J} n_j = N. \tag{8}$$

Selecting one winner per candidate pool spreads calls across independently planned trajectories; selecting the top-N paths uses a shared pool. Repeating one selected prompt is the case $J = 1$ and $n_1 = N$, with separate decoding requests. Prompt diversity and repeated decoding thus provide complementary ways to use a fixed generation budget after planning. The paths and repetition counts are chosen before generation, and the query representation is reused throughout. We use majority vote for answer aggregation; oracle pick is an evaluation diagnostic. Appendix E.4 gives the concrete selection, repetition, and voting procedures.

Planning cost. With shared linear projections, projected bank features can also be cached. For M scored paths and a fixed-size view neighborhood, scoring costs $O(MLr(K + h))$, in addition to one query-encoder forward and $O(Ldr)$ query projection work, where d is the common encoder width and h is the fusion width. Independent pools of size M_{pool} give $M = NM_{\text{pool}}$; repeating one selected path reuses a single search. The planning budget controls candidate coverage, whereas N controls target-generation width. Appendix B.1 gives the detailed accounting.

3 Theoretical analysis

For a fixed query q, let $r(P; q)$ be the probability that one target-model call under path P is correct, and let $r^\star(q) = \sup_P r(P; q)$. Two errors limit prompt planning: the explored pool may omit useful paths, and the scorer may rank them incorrectly. Suppose the pool contains a path with utility at least $r^\star(q) - \varepsilon$ with probability $1 - \delta_{\text{cov}}$. Suppose also that, with probability $1 - \eta$, all path pairs with utility gap greater than γ are strictly ordered correctly by the scorer. Then the selected path satisfies

$$\mathbb{E}[r(\widehat{P}; q)] \geq r^\star(q) - \varepsilon - \gamma - \eta - \delta_{\text{cov}}. \tag{9}$$

The proof bounds selection regret on the joint coverage-and-ranking event and charges at most one unit of regret otherwise; Appendix C gives the full results. Relative to random prompting, the available gain is the *promptability gap* $r^\star(q) - \mathbb{E}_{P \sim \pi}[r(P; q)]$, less these errors. For a reference prompt proposal π, let $\text{PromptBoN}_N(q) = \mathbb{E}[\max_{1 \leq i \leq N} r(P^{(i)}; q)]$ for independent $P^{(i)} \sim \pi(\cdot \mid q)$. If the explored pool contains M_{pool} independent draws from π, the same ranking condition gives

$$\mathbb{E}[r(\widehat{P}; q)] \geq \text{PromptBoN}_{\min\{N, M_{\text{pool}}\}}(q) - \gamma - \eta. \tag{10}$$

The expected pool optimum is at least this prompt-oracle utility, and the expected regret from ranking is at most $\gamma + \eta$ (Appendix C.4). This characterizes the selection benefit that planner-side search

can recover without executing every candidate. For independent sampling, if near-optimal paths have proposal probability $p_\varepsilon > 0$, then $\delta_{\text{cov}} \leq (1 - p_\varepsilon)^{M_{\text{pool}}}$. Increasing the pool reduces this coverage error, but cannot recover paths outside the proposal's support. Under stochastic decoding, this prompt-selection oracle differs from Pass@N, which checks whether any of N realized responses is correct. The theory does not equate one planned call with arbitrary multi-output aggregation, nor does a larger pool by itself guarantee a smaller ranking error.

4 Experiments

4.1 Benchmarks, metrics, and baselines

Benchmarks and data separation. We evaluate professional knowledge on MMLU-Pro [Wang et al., 2024], graduate-level scientific QA on GPQA [Rein et al., 2024], and mathematical reasoning on GSM8K [Cobbe et al., 2021], MATH500 [Hendrycks et al., 2021], and AIME [Veeraboina, 2023]. AIME 2021–2024 is reserved for testing. Scorer fitting and exemplar construction use training data; validation supports model selection and the labeled target-adaptation setting. Test questions and answers are excluded from those stages. Appendix A.7 gives the split accounting and evaluation protocol.

Metrics and models. We report exact-answer accuracy for multiple-choice tasks and exact match for numeric or symbolic answers. Unless otherwise stated, methods share the system instruction, temperature 0.3, and maximum generation length 4096. Targets are frozen Qwen3-4B/8B/14B [Yang et al., 2025], Mistral-7B [Jiang et al., 2023], and LLaMA3-8B [AI@Meta, 2024] models. Closed-source transfer is evaluated separately.

Baselines. *Original* uses one random K-shot prompt. pFLAT and EASE are prompt-flatness and ordering-aware exemplar-optimization baselines [Shen et al., 2023, Wu et al., 2024]. *Majority vote* aggregates multiple sampled answers [Wang et al., 2022]. *Random-proposal oracle pick* reports success when at least one of N unfiltered random-prompt responses is correct, corresponding to Pass@N for that proposal. It upper-bounds answer selection from its own realized candidates, not a different prompt-selection policy with the same call count.

Compute accounting. We report target-generation calls separately from query-encoder calls and end-to-end latency. Cached exemplar encoding and scorer training are offline costs. Table 7 measures online overhead on an RTX 4090; Appendix B.1 gives the accounting and complexity. Equal call counts do not imply identical token counts or latency.

4.2 Implementation details

Offline training. The full-layer benchmark configuration uses shared linear projections and a small MLP with metric dimension $r = 64$ and hidden width $h = 96$, approximately 0.35M parameters. We train it with Adam at learning rate approximately 10^{-3} using correctness-flipping local edits (Sec. 2.5). Prompt lengths are $K \in \{4, 8\}$; the larger-K curriculum first emphasizes content edits in the first $K/2$ positions. Training pairs are balanced across benchmarks, with 10K–500K pairs in the data-size experiments. Only the scorer receives gradient updates.

Inference. Qwen3-4B is the default frozen encoder; Table 12 compares encoder choices. The default candidate-pool size is $M_{\text{pool}} = 100$, independently of the number K of exemplars per path. Single-call Maze selects one maximum-energy path. For the main multi-call experiments, MultiMaze selects one winner from each of N independently sampled pools and votes over the N answers. Shared-pool top-N selection is evaluated separately to isolate the effect of ranking.

Runtime. Experiments use an RTX 4090 with 24 GB memory. Planner training takes several hours; contrastive-data preparation takes approximately 10–20 hours, depending on the labeling workload. These costs are amortized across subsequent queries rather than included in serving latency. Appendix E details the scorer parameterizations, training objectives, and fixed-budget inference procedures.

Table 1: **Overall accuracy (%) on LLM reasoning tasks with single-call methods.** Original uses one random few-shot call. The reported pFLAT [Shen et al., 2023] and EASE [Wu et al., 2024] are geometric and ordering-aware exemplar optimization methods. *Maze* selects the highest-scoring path from a random candidate pool and uses one target-model call.

		MMLU-Pro	GPQA	GSM8K	MATH500	AIME 21-24
Mistral-7B	Original	24.29	29.69	53.46	42.12	7.78
	pFLAT	26.43	31.70	58.08	45.28	12.35
	EASE	26.79	32.81	58.46	46.12	13.68
	Maze (Ours)	**28.94**	**34.35**	**63.56**	**48.20**	**18.23**
LLaMA3-8B	Original	41.37	31.65	80.90	48.40	4.12
	pFLAT	44.50	35.80	85.78	51.36	9.20
	EASE	45.10	35.85	85.91	52.80	9.35
	Maze (Ours)	**48.32**	**39.10**	**87.72**	**53.80**	**14.58**
Qwen3-8B	Original	59.74	38.35	89.98	86.82	47.57
	pFLAT	66.13	49.95	92.03	88.24	49.25
	EASE	66.95	48.30	91.77	87.98	50.32
	Maze (Ours)	**70.05**	**53.37**	**93.02**	**90.12**	**53.21**

4.3 Main results on reasoning tasks

Single-call inference-time scaling. Table 1 reports the single-call results. Maze improves over both pFLAT and EASE in every reported model–benchmark setting. For Qwen3-8B, gains over the stronger of these two baselines range from 0.99 to 3.42 percentage points. Relative to random prompting, the improvement on GPQA is 15.02 points. These results demonstrate useful prompt sensitivity without changing the target model or increasing its call budget.

Multi-call scaling. In Table 2, MultiMaze with 16 target calls outperforms 32-call majority vote in every reported setting, halving the generation-call budget. On GPQA with Qwen3-14B, the improvement is 16.7 percentage points. MultiMaze can also exceed random-proposal oracle pick at 16 calls because the two methods execute different prompts: Maze screens candidates before generation, whereas the oracle selects among unfiltered realized responses. The oracle remains optimal conditional on the responses it receives.

Table 2: **Multi-call accuracy (%) and latency relative to one random-prompt call.** Majority vote uses 32 calls. Random oracle denotes random-proposal oracle pick over 16 unfiltered responses, successful when at least one is correct. MultiMaze uses 16 calls with non-oracle majority vote, selecting one prompt from each independent 100-path pool. The oracle and Maze therefore have equal target-call budgets but different prompt proposals.

		MMLU-Pro	GPQA	GSM8K	MATH500	AIME	Latency
Mistral-7B	Majority vote	28.93	37.28	73.46	48.25	19.25	38.2×
	Random oracle	35.00	39.96	80.38	52.67	23.46	18.1×
	MultiMaze	**37.25**	**43.52**	**84.39**	**55.58**	**27.75**	19.0×
LLaMA3-8B	Majority vote	53.53	41.03	93.05	54.28	15.65	37.5×
	Random oracle	60.59	46.15	94.59	56.76	17.82	17.2×
	MultiMaze	**62.39**	**51.48**	**95.52**	**60.52**	**21.68**	17.5×
Qwen3-14B	Majority vote	82.80	54.32	94.65	92.30	55.12	31.2×
	Random oracle	85.98	65.27	95.72	92.90	59.39	16.9×
	MultiMaze	**87.85**	**70.98**	**95.82**	**95.06**	**61.30**	16.7×

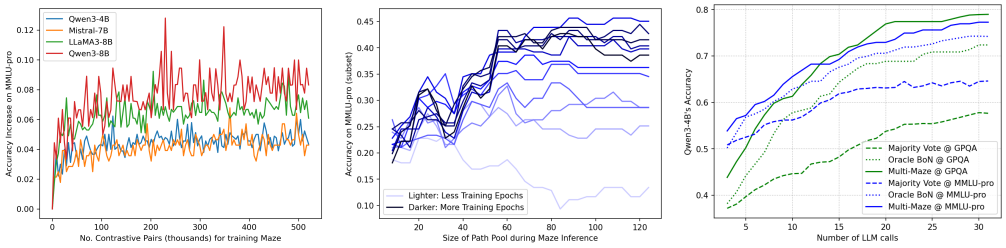
4.4 Controlled selection and scorer analysis

Selection and aggregation under a shared pool. To isolate prompt ranking, we compare random selection and Maze top-16 selection from an identical 1,600-path pool. Both execute exactly 16 prompts with the same target and decoding settings. Table 3 shows that Maze improves both majority-vote and oracle-pick accuracy, while oracle pick dominates voting within each selected set. Thus,

Table 3: **Selection and aggregation on a shared candidate pool.** Accuracy (%) with Qwen3-14B, 1,600 candidate paths, and 16 executed prompts per method. Oracle-Pick is diagnostic; majority vote is the deployable aggregation rule. This shared-pool comparison is separate from the independent-pool policy in Table 2.

Selection	Aggregation	MMLU-Pro	GPQA	GSM8K	MATH500	AIME
Random-16	Majority vote	81.2	51.0	94.2	91.8	53.5
Random-16	Oracle-Pick	86.0	65.3	95.7	92.9	59.4
Maze-top-16	Majority vote	88.2	71.6	95.9	95.3	61.8
Maze-top-16	Oracle-Pick	91.4	79.8	97.1	97.3	66.8

the improvement concerns which prompts are executed, rather than a change in aggregation. The shared-pool sweep in Appendix A.6 extends the comparison to $k \in \{1, 2, 4, 8, 16\}$, with $100k$ candidate paths and exactly k target calls for each selector. Maze improves over random selection under the same aggregation rule at every evaluated budget on both MMLU-Pro and GPQA. The gain is therefore not confined to the 16-call operating point.



(a) No. Contrastive Pairs. (b) Size of Path Pool. (c) No. target LLM calls.

Figure 2: **Scaling behavior of *Maze*.** (**a**) Accuracy @ MMLU-Pro vs. number of contrastive pairs used to train the metric mappings. (**b**) Qwen3-4B's Accuracy @ MMLU-Pro vs. path-pool size M_{pool} at inference. Darker hues indicate more training epochs (from 2 to 32) for the same configuration. (**c**) Multi-call Maze vs. Oracle-guided BoN and Majority Vote (Qwen3-4B): with the same or fewer calls, energy-ranked prompting improves the observed call–accuracy trade-off; the oracle curve uses random proposals.

What does the trajectory scorer contribute? Table 4 compares rankers built from the same cached representations. The learned controls share correctness supervision, candidate paths, and optimization budget, but differ in how they represent the ordered prompt. Content-only and query–exemplar similarity scores omit joint path structure; local-order and single-layer alternatives capture only part of it. Maze improves over each of these alternatives on all three evaluated tasks.

Table 4: **Cached-representation scorer comparison.** We compare *Maze* against progressively stronger prompt-ranking baselines built from the same cached representations. All learned controls use the same correctness-labeled prompts, representations, candidate paths, and optimization budget; their structural parameterizations differ.

Cached-representation scorer	What it captures	MMLU-Pro	GPQA-Main	MATH500
Mean-pool exemplars + query	content only, order-agnostic	63.2	42.2	87.0
Summing query–exemplar cosine similarity	relevance only	64.1	41.9	86.8
Adjacent-pair heuristic scorer	local order only	67.2	43.4	87.6
Single-layer pooled hidden states + MLP	content + weak order	68.2	46.7	88.2
Last-layer pairwise scorer + fusion	partial path structure	67.9	48.5	88.0
Full *Maze*	directed cross-layer structure	**69.9**	**53.5**	**90.4**

A parameter-matched two-layer sequence Transformer provides a stronger learned comparison (Table 5). Both models have approximately 0.35M parameters and receive the same representations and correctness supervision. Maze retains an advantage across all three tasks, including GPQA, where the repeated-run means are 53.5% and 50.6%. Together, these comparisons isolate a benefit of directed cross-layer structure beyond correctness-supervised prompt ranking alone.

Table 5: **Matched-capacity, matched-supervision sequence ranker.** Accuracy (%), mean ± standard deviation across repeated runs. Each scorer has approximately 0.35M parameters and uses the same input representations, candidate paths, and correctness supervision.

Scorer	MMLU-Pro	GPQA	MATH500
Two-layer sequence Transformer	68.3 ± 0.6	50.6 ± 1.2	88.8 ± 0.8
Full *Maze*	69.9 ± 0.7	53.5 ± 0.9	90.4 ± 0.6

Robustness across complete runs. We evaluate five runs that independently vary scorer initialization and training order, candidate-path pools, and target-model decoding seeds. Table 6 reports mean accuracy and standard deviation, not standard error. The Qwen3-8B MultiMaze results use independent candidate pools; the Qwen3-14B results use the shared-pool control of Table 3.

Table 6: **Variability over five complete runs.** Accuracy (%), reported as mean ± standard deviation, not standard error.

Target	Method	GPQA	AIME
Qwen3-8B	Random prompt, 1 call	38.35 ± 1.3	47.57 ± 2.6
	EASE, 1 call	48.30 ± 0.9	50.32 ± 2.1
	Maze, 1 call	53.37 ± 0.8	53.21 ± 1.9
	Random-path MV, 16 calls	48.8 ± 1.1	53.3 ± 2.2
	MultiMaze, 16 calls	69.1 ± 0.8	63.2 ± 1.7
Qwen3-14B	Random-16 + majority vote	51.0 ± 1.0	53.5 ± 2.1
	Random-16 + Oracle-Pick	65.3 ± 0.8	59.4 ± 1.9
	Maze-top-16 + majority vote	71.6 ± 0.7	61.8 ± 1.6
	Maze-top-16 + Oracle-Pick	79.8 ± 0.9	66.8 ± 1.5

For Qwen3-8B, paired bootstrap intervals resample test questions while preserving method pairing and matched seeds. The one-call Maze–EASE gain is $+5.1$ percentage points on GPQA (95% CI $[1.4, 8.8]$) and $+2.9$ on AIME ($[-0.8, 8.1]$). The 16-call MultiMaze–random-majority-vote gain is $+20.3$ on GPQA ($[15.6, 25.0]$) and $+9.9$ on AIME ($[3.3, 16.5]$). Thus, the GPQA improvements and both multi-call comparisons remain positive under this analysis, while the smaller single-call AIME difference is less certain.

4.5 Efficiency and black-box transfer

Online cost. Table 7 reports target calls, query-encoder calls, end-to-end latency, and accuracy together. On Qwen3-8B/MMLU-Pro, one-call latency is 15.5 s for random prompting and 15.8 s for Maze, including query encoding and scoring. MultiMaze uses 254.6 s at 16 calls, compared with 483.2 s for 32-call majority vote. At the same 16-call budget, random-proposal oracle pick takes 252.5 s; Maze's additional planning takes 2.1 s while changing which prompts are executed. The added planner cost is small in this measured setting, but actual latency depends on the serving configuration. Offline supervision and bank encoding are amortized over subsequent queries rather than included in these serving measurements.

Black-box transfer. Table 8 evaluates GPT-5.1 and Gemini 2.5 Pro using representations from the frozen Qwen3-4B encoder. Zero-shot target transfer uses the source-trained scorer unchanged, without collecting target-specific correctness labels. Validation-adapted transfer updates only the scorer using 32 non-test target queries and 512 target completions per benchmark; neither the encoder nor the target is fine-tuned. Both settings preserve black-box target access, but only the former is zero-shot with respect to target supervision. Appendix A.7 separates this adaptation budget from source-model training and validation.

Table 7: **Online compute accounting on Qwen3-8B/MMLU-Pro with an RTX 4090.** Latency includes query encoding and path scoring, but excludes offline bank encoding and training. Target generation and proxy encoding are counted separately.

Method	Target calls	Encoder calls	Latency (s/query)	Accuracy (%)
Original	1	0	15.5	59.74
Maze	1	1	15.8	70.05
Majority vote	32	0	483.2	82.35
Random oracle	16	0	252.5	85.71
MultiMaze	16	1	254.6	86.42

Table 8: **Black-box transfer via a frozen Qwen3-4B encoder.** Accuracy (%) is reported for one-call prompting and eight-call aggregation. Zero-shot transfer uses the source scorer unchanged. Adapted Maze uses 32 non-test target queries and 512 target completions per benchmark to update the scorer, not the encoder or target. The multi-call Maze rows use this adapted scorer. The baseline and adapted operating points are averaged over three attempts; zero-shot transfer is a separate control.

Target	Policy	MMLU-Pro	GPQA	AIME-24
GPT-5.1	Original, 1 call	86.2	83.8	82.8
	Maze, zero-shot, 1 call	87.9	85.8	83.6
	Maze, adapted, 1 call	88.9	87.2	84.2
	Majority vote, 8 calls	89.4	87.5	84.0
	Random oracle, 8 calls	91.0	88.4	86.2
	MultiMaze, 8 calls	**91.4**	**89.6**	**86.8**
Gemini 2.5 Pro	Original, 1 call	85.8	84.2	87.1
	Maze, zero-shot, 1 call	88.0	86.7	89.7
	Maze, adapted, 1 call	89.1	88.4	91.5
	Majority vote, 8 calls	89.0	88.3	92.2
	Random oracle, 8 calls	90.8	90.2	93.1
	MultiMaze, 8 calls	**91.3**	**91.0**	**94.5**

Zero-shot transfer improves on the one-call baseline for both targets in every reported task, showing that useful prompt-selection information transfers without target-specific correctness feedback. Adaptation provides a further improvement in each setting. With the adapted scorer, eight-call MultiMaze also improves on eight-call majority vote across both targets and all three tasks. These comparisons support the encoder–target separation while distinguishing transfer of a fixed scorer from its target-specific adaptation.

4.6 Ablations and limitations

For Qwen3-4B/MMLU-Pro, shuffling the order of a Maze-selected exemplar set reduces accuracy by 2.8 points on average, holding its content fixed. Figure 2a shows diminishing returns beyond approximately 200K contrastive pairs. Figure 2b likewise shows diminishing returns as the pool grows; in this configuration, $M_{\text{pool}} = 100$ performs comparably to 10,000 with approximately 1% of the path-scoring work. These are empirical operating points, not guarantees that enlarging a pool improves ranking. Cross-domain bank comparisons, component ablations, and geometric diagnostics appear in Appendix A. However, the benefits mainly depend on useful exemplars, reliable ranking, and sufficient reuse to amortize offline supervision. Proxy–target mismatch and residual decoding uncertainty can limit the gains (Appendix B.4).

5 Conclusion and limitations

Maze treats the prompt as a structured decision to optimize before generation. Its compact geometric scorer ranks ordered exemplar trajectories using cached representations, allowing one or a few target-model calls to benefit from a broader prompt-side search. The experiments distinguish gains from prompt selection, response aggregation, and target-specific adaptation. The theory identifies coverage and ranking error as the conditions governing this trade-off. In general, these results establish prompt-trajectory planning as a complementary allocation of test-time computation for both open-weight and black-box LLMs.

Acknowledgments and Disclosure of Funding

This work was supported in part by the National Key R&D Program of China under Grant 2023YFC2508704, in part by the National Natural Science Foundation of China under grant number 62236008 and 62521007, in part by the Natural Science Foundation of Beijing under grant number L251082, and in part by Shandong Provincial Natural Science Foundation under project ZR2025ZD01.

References

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.

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.

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.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 17682–17690, 2024.

Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. Universal self-consistency for large language model generation. *arXiv preprint arXiv:2311.17311*, 2023.

Lingjiao Chen, Jared Quincy Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei A Zaharia, and James Y Zou. Are more llm calls all you need? towards the scaling properties of compound ai systems. *Advances in Neural Information Processing Systems*, 37:45767–45790, 2024.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *arXiv preprint arXiv:2104.08786*, 2021.

Qi Guo, Leiyu Wang, Yidong Wang, Wei Ye, and Shikun Zhang. What makes a good order of examples in in-context learning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14892–14904, 2024.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. Learning to retrieve prompts for in-context learning. *arXiv preprint arXiv:2112.08633*, 2021.

Lingfeng Shen, Weiting Tan, Boyuan Zheng, and Daniel Khashabi. Flatness-aware prompt selection improves accuracy and sample efficiency. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 7795–7817, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.523. URL https://aclanthology.org/2023.findings-emnlp.523/.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

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.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Hemish Veeraboina. Aime problem set 1983-2024, 2023. URL https://www.kaggle.com/datasets/hemishveeraboina/aime-problem-set-1983-2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

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.

AI@Meta. Llama 3 model card, 2024. URL https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.

Zhaoxuan Wu, Xiaoqiang Lin, Zhongxiang Dai, Wenyang Hu, Yao Shu, See-Kiong Ng, Patrick Jaillet, and Bryan Kian Hsiang Low. Prompt optimization with ease? efficient ordering-aware automated selection of exemplars. *Advances in Neural Information Processing Systems*, 37:122706–122740, 2024.

Wei Zhang, Maosheng Zhang, Xiaoyan Sun, and Jingyi Mo. Reconstruction of three-dimensional sound field system based on machine learning method. *Electronics Letters*, 61(1):e70133, 2025.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Liang Wang, Nan Yang, and Furu Wei. Learning to retrieve in-context examples for large language models. *arXiv preprint arXiv:2307.07164*, 2023.

Kehan Xu, Kun Zhang, Jingyuan Li, Wei Huang, and Yuanzhuo Wang. Crp-rag: A retrieval-augmented generation framework for supporting complex logical reasoning and knowledge planning. *Electronics*, 14(1):47, 2024.

Kehan Xu, Kun Zhang, Wei Huang, Jingyuan Li, and Yuanzhuo Wang. Rlg-rag: Guiding the knowledge retrieval and evaluation in retrieval-augmented generation framework by reasoning logic. In *Companion Proceedings of the ACM on Web Conference 2025*, pages 1450–1454, 2025.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Ahmad Beirami, Alekh Agarwal, Jonathan Berant, Alexander D'Amour, Jacob Eisenstein, Chirag Nagpal, and Ananda Theertha Suresh. Theoretical guarantees on the best-of-n alignment policy. *arXiv preprint arXiv:2401.01879*, 2024.

Ryan D'Orazio, Danilo Vucetic, Zichu Liu, Junhyung Lyle Kim, Ioannis Mitliagkas, and Gauthier Gidel. Solving hidden monotone variational inequalities with surrogate losses. In *International Conference on Learning Representations*, volume 2025, pages 77721–77749, 2025.

Zichu Liu and Lacra Pavel. Recursive reasoning in minimax games: A level k gradient play method. *Advances in Neural Information Processing Systems*, 35:16903–16917, 2022.

A Appendix: more empirical results

This appendix supplements the main-text controlled comparisons, variability analysis, online-cost accounting, and black-box transfer results. It reports component and capacity ablations, encoder and exemplar-bank comparisons, diagnostic visualizations, matched-pool budget sweeps, and data-split details.

A.1 Component ablations: what makes *Maze* work

We ablate the components of *Maze*'s metric-geometry to isolate where gains arise (Table 9).

Layer usage. Using only the last or only a middle layer to define the manifold underperforms the multiplex variant of all-layers. The comparison supports complementary information across the evaluated encoder layers.

Cross-layer couplings. Removing cross-view terms from the transition relation (Eq. 2) or the path fusion (Eq. 4) degrades accuracy. These reductions support using both cross-layer relation features and query-conditioned fusion in the evaluated scorer.

Supervision signal. Replacing contrastive local edits (content replacement or order swaps that flip correctness) with random pairs consistently hurts performance. Contrastive supervision concentrates learning on decision boundaries of correctness, producing a sharper energy landscape for ranking candidates.

Overall. The complete *Maze*, with multiplex metrics, cross-layer fusion, and contrastive supervision, delivers the largest gains across the reported Qwen3-4B benchmarks, supporting the complementary roles of the components.

Table 9: **Ablation study for single-call *Maze* on LLM reasoning tasks.**

		MMLU-Pro	GPQA	GSM8K	MATH500
Qwen3-4B	Original	49.11	36.50	89.86	84.46
	only last layer	53.0	40.8	91.0	86.34
	only middle layer	53.2	41.1	91.6	86.68
	w/o Cross-layer metrics	53.5	42.4	92.1	86.46
	w/o Cross-layer fusions	52.5	42.7	92.0	86.10
	random pairs	52.3	40.7	91.5	85.14
	Maze (Ours)	**54.13**	**43.18**	**92.67**	**87.28**

A.2 Hyperparameter choices (hidden dimension, metric dimension, mappings)

Figure 3b illustrates how the accuracy of *Maze* on the MMLU-Pro subset varies with the number of parameters, adjusted by modifying the hidden dimension or metric dimension. We observe that increasing either the metric dimension or the hidden dimension leads to a larger parameter count and improved performance. When the model size reaches approximately 0.35 million parameters, performance begins to saturate, with gains becoming more marginal. Additional capacity yields smaller gains over the measured range. Table 10 presents example results using Qwen3-4B in MMLU-Pro and MATH500, where h denotes the hidden dimension and r the metric dimension.

Table 10: **Accuracy (%) for different projection and fusion dimensions.** Qwen3-4B is the target; h is the fusion hidden width and r the metric dimension.

Hidden width h	Metric dimension r	MMLU-Pro	MATH500
48	32	49.2	84.9
48	64	52.7	85.6
96	32	53.2	87.2
96	64	54.1	87.3
128	96	54.3	87.4

For practical considerations and limited computing resources, we did not scale further. The observed saturation does not establish how performance would scale beyond the tested configurations. Re-

Table 11: **Diagnostics on explored prompt pools.** Ranking AUC and correlation summarize agreement with path utility; effective rank summarizes feature concentration. Recall@16 concerns near-optimal paths within the explored pool.

Dataset	Ranking AUC	Spearman ρ	Effective rank ratio	Recall@16 (%)
MMLU-Pro	0.78	0.46	0.19	86.4
GPQA-Main	0.83	0.54	0.26	69.1
MATH500	0.82	0.51	0.23	93.4

garding Mass Mapping and Fusion Function, we also experimented with alternative mass mapping functions beyond the simple linear projection, as well as a deeper fusion network to replace the current shallow MLP. However, we found that these more complex variants did not yield significant improvements over our original design. For example, using an attention-based mass mapping or a 2-layer MLP as fusion mapping leads to a marginal increase less than 0.5%, while increasing implementation complexity. This suggests that our chosen combination "linear projection + simple FC" strikes a good balance between simplicity and effectiveness.

A.3 Theory-grounded diagnostics

Table 11 reports pairwise ranking AUC, Spearman correlation, normalized effective rank, and near-optimal-path Recall@16 on explored candidate pools. The ranking statistics indicate alignment between energy and path utility, while effective rank describes concentration of the measured feature matrix. Recall@16 describes retrieval within the explored pool; it does not establish coverage of the full prompt space. Likewise, average pairwise AUC does not imply the simultaneous margin-ordering event used in the proofs. These measurements are diagnostic evidence rather than verification of all theorem assumptions.

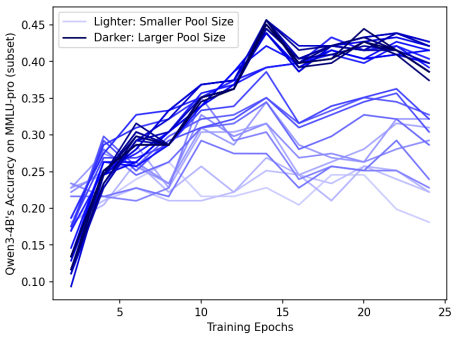
A.4 Other results

Training-set size and saturation. Accuracy improves as the number of contrastive pairs increases and then saturates beyond $200k$ pairs; stronger base models show a larger variance with further scaling (Fig. 2a).

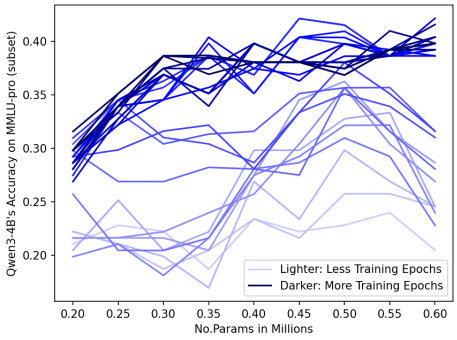
Inference pool size. Energy evaluation is matrix-vector dominated and sub-millisecond per path on our setup; accuracy plateaus after modest pool sizes (e.g., 60-80 paths) with diminishing returns toward (Fig. 2b).

Mapping capacity. Increasing the parameter count helps but saturates up to 0.35 millions (Appendix Fig. 3b), indicating that the benefit of Maze comes from structure, not just from large parametric capacity. A larger Maze model can lead to better performance, yet the compute-accuracy trade-off needs consideration.

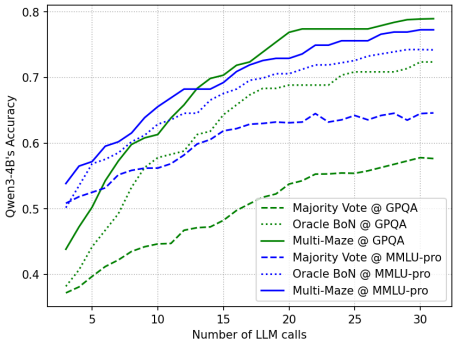
Cost–accuracy trade-off. *MultiMaze* curves (Fig. 2c - 3d) show that 3 to 5 calls recover a substantial fraction of the performance of the majority vote of 32 LLM calls, illustrating complementary gains from prompt selection and response aggregation.



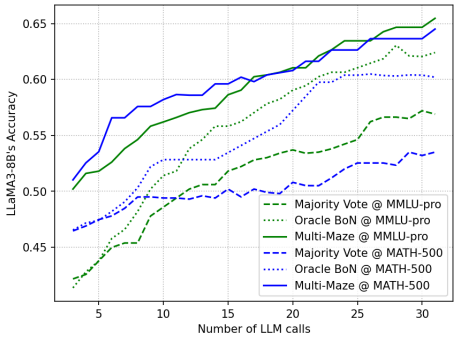
(a) Training epoch affects Accuracy.



(b) No.Params affects Accuracy.



(c) Multi-Maze vs. Majority Vote with Qwen3-4B.



(d) Multi-Maze vs. Majority Vote with LLaMA3-8B.

Figure 3: **Scaling behavior of Maze.** (**a**) Training epochs vs. accuracy on MMLU-Pro: more epochs yield steady gains that saturate after 15 epochs; darker hues indicate a larger pool size of paths (from 8 to 128). (**b**) Parameters count vs. Accuracy on MMLU-Pro (Qwen3-4B): darker hues indicate a larger training epoch (from 2 to 32). (**c**) Multi-call Maze vs. Oracle-guided BoN and Majority Vote (Qwen3-4B): with the same or fewer calls, energy-ranked prompting improves the observed call–accuracy trade-off; the oracle curve uses random proposals. (**d**) Multi-call Maze vs. Oracle-guided BoN and Majority Vote (LLaMA3-8B): a corresponding call–accuracy comparison for a different target model.

Table 12: **Accuracy gains with a swappable encoder**. We compare performance improvements (absolute accuracy gain over the raw LLM) when (i) using LLM itself to produce exemplar/query embeddings ("LLM itself") versus (ii) using a lightweight proxy encoder (Qwen3-4B). "/" denotes a setting not evaluated in this comparison. Proxy-encoder results are reported for Maze variants.

		MMLU-Pro		AIME 21-24	
Encoder Type		LLM itself	Qwen3-4B	LLM itself	Qwen3-4B
Mistral-7B	EASE	+2.5	/	+5.9	/
	Maze (Ours)	+4.3	+4.7	+10.8	+10.8
	Majority Vote	+4.7	/	+11.4	/
	MultiMaze (Ours)	+13.2	+12.0	+19.6	+19.5
LLaMA3-8B	EASE	+3.7	/	+5.2	/
	Maze (Ours)	+7.1	+6.8	+10.5	+10.0
	Majority Vote	+12.3	/	+11.5	/
	MultiMaze (Ours)	+20.9	+20.7	+17.3	+17.8
Qwen3-8B	EASE	+7.2	/	+2.8	/
	Maze (Ours)	+10.3	+10.1	+5.7	+5.5
	Majority Vote	+22.5	/	+8.1	/
	MultiMaze (Ours)	+24.7	+24.4	+15.6	+15.7
Qwen3-14B	EASE	+4.8	/	+1.5	/
	Maze (Ours)	+8.2	+8.7	+3.3	+3.8
	Majority Vote	+12.2	/	+3.1	/
	MultiMaze (Ours)	+17.0	+17.3	+9.3	+9.3

Table 13: **Accuracy gains on different domains**. We compare performance improvements (absolute accuracy gain over the raw LLM) when using the (training) exemplars pool from the other domains, *i.e.*, using MATH500's exemplars pool for the inference in GSM8K. The bolded values refer to the best gains, while the underlined values refer to the default same-domain gains.

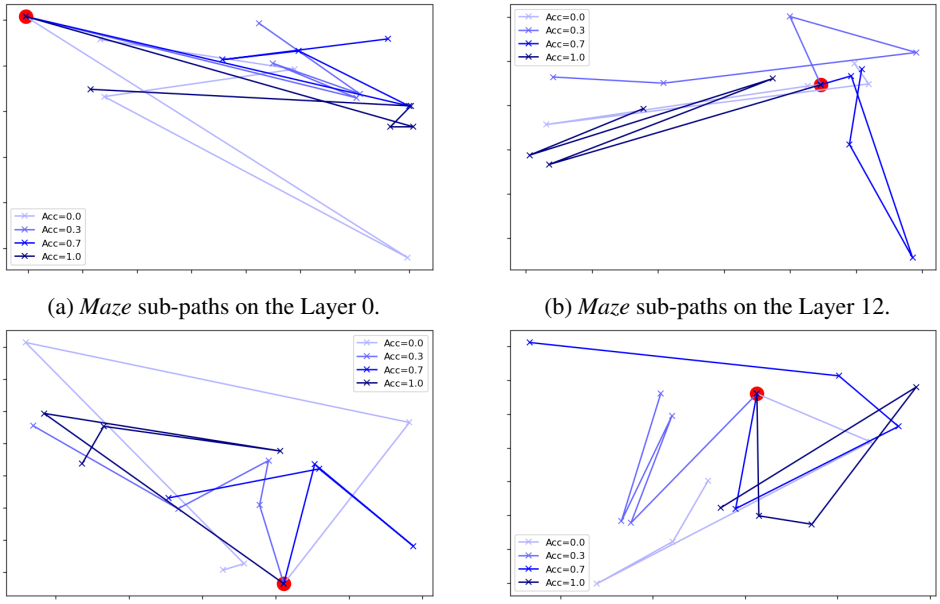
	Training domain	Test domain				
		MMLU-Pro	GPQA	GSM8K	MATH500	AIME
Qwen3-4B	MMLU-Pro	+5.1	+6.1	+2.3	+2.2	+4.3
	GPQA	**+5.3**	**+6.6**	+2.6	+2.4	+4.6
	GSM8K	+2.1	+1.5	+2.8	+2.4	+1.2
	MATH500	+4.0	+3.9	**+3.2**	+2.8	+5.1
	AIME	+2.0	+1.6	+2.9	**+3.0**	**+7.5**
Qwen3-8B	MMLU-Pro	**+10.3**	**+5.2**	+3.2	+3.1	+3.9
	GPQA	+9.6	+5.0	+3.0	+3.5	+4.3
	GSM8K	+5.4	+2.3	+3.1	+2.6	+2.4
	MATH500	+6.2	+3.8	+3.4	+3.3	+4.5
	AIME	+6.5	+4.2	**+3.5**	**+3.6**	**+5.6**

A.5 Visualization

Setup. We project hidden state layers for a fixed query and its candidate exemplar paths in 2D using principal component analysis (PCA). Concretely, for each inspected layer l, we collect the average hidden state along each candidate path, fit a PCA per layer on the union, and plot the resulting 2D embeddings. Within each panel, a polyline traces the exemplar order in a candidate path; the line darkness encodes the empirical correctness of that path when used as a few-shot prompt, and the query is marked separately.

Findings. The displayed two-dimensional arrangements change across encoder layers, and high- and low-utility paths overlap in these projections. This illustrates view-dependent geometry but does not rule out linear separation in the original high-dimensional features. The scorer ablations, rather than PCA overlap, provide the quantitative evidence for the value of cross-layer interactions.

Interpretation and caveats. These PCA plots are only qualitative diagnostics. Because PCA is unsupervised and two-dimensional, overlap in the plotted space does not rule out linear or nonlinear separability in the original high-dimensional representation. We therefore use the visualization only to illustrate that naive low-dimensional inspection is insufficient, and rely on held-out ranking AUC, Spearman correlation, controlled ablations, and downstream accuracy for quantitative evidence.



(a) *Maze* sub-paths on the Layer 0.

(b) *Maze* sub-paths on the Layer 12.

(c) *Maze* sub-paths on the Layer 24.

(d) *Maze* sub-paths on the Layer 36.

Figure 4: **Layer-wise PCA projections of exemplar sub-paths for a single MMLU-Pro query with Qwen3-4B.** Each polyline corresponds to the ordered exemplars in a candidate few-shot path. Darker lines denote paths that yield correct answers when executed as prompts. The red circle marks the test query's embedding. These projections illustrate how the same ordered paths appear in different encoder layers; they do not establish separability or its absence in the full representation space.

A.6 Controlled selection, aggregation, and variability

Shared candidate pools. Table 3 compares selection and aggregation separately. The candidate bank, sampling distribution, target model, number of exemplars, decoding configuration, and answer evaluator are identical across selection methods. For majority-vote ties, we use uniform random tie-breaking with shared seeds. An oracle over all 1,600 candidate paths is not a 16-call comparator: evaluating all their realized answers would require 1,600 target-model calls.

Table 14: **Call-budget sweep with shared upstream pools.** Qwen3-8B accuracy (%). For each query and budget k, both selection methods use the same pool of $100k$ paths and execute exactly k prompts. MV denotes majority vote; Oracle denotes Oracle-Pick on the corresponding executed response set.

Benchmark	k	Pool size	Random + MV	Random + Oracle	*Maze* + MV	*Maze* + Oracle
MMLU-Pro	1	100	59.74	59.74	70.05	70.05
	2	200	61.8	67.3	72.9	76.4
	4	400	66.5	74.2	77.8	82.0
	8	800	72.7	81.0	82.8	87.0
	16	1,600	78.5	85.8	86.7	90.2
GPQA	1	100	38.35	38.35	53.37	53.37
	2	200	39.6	43.7	56.1	59.8
	4	400	42.4	50.5	61.2	66.7
	8	800	46.0	57.1	65.8	73.2
	16	1,600	48.8	63.2	69.4	78.0

Variability and matched supervision. Section 4.4 and Table 6 report five-run means and standard deviations, with separate independent-pool and shared-pool protocols. The main-text paired intervals for Qwen3-8B resample test questions while preserving matched seeds and method pairing. For the shared-pool Qwen3-14B comparison, a hierarchical paired bootstrap resamples complete runs first and then questions within selected runs, retaining paired method outputs. Complete-run standard deviations and bootstrap confidence intervals summarize different sources of uncertainty and should not be interchanged. The matched-supervision cached-feature and sequence-ranker comparisons are presented in Tables 4 and 5.

A.7 Data separation and target-specific adaptation

The scorer-training queries, source-model validation queries, and final test queries are separated as summarized in Table 15. The exemplar bank is built from training data only, and a training query cannot appear in its own demonstration path. Final test questions, answers, and correctness labels are excluded from pair construction, scorer fitting, early stopping, hyperparameter selection, and target-specific adaptation. Their representations are computed only for final inference, not used to fit the planner.

Table 15: **Query splits for the benchmark evaluation.** Counts refer to queries, not the much larger number of sampled prompt pairs. MATH development denotes the 12,000-item pool excluding the held-out MATH500 questions.

Benchmark	Training	Validation	Test	Exemplar-bank source
MMLU-Pro	56/fold	14/fold	12,032	Training part of the official 70-item validation set
GPQA-Main	78	20	448	GPQA Extended excluding GPQA Main; training part only
GSM8K	6,726	747	1,319	Training part of the official training split
MATH500	11,000	1,000	500	Training part of the 12,000-item MATH development pool
AIME	825	60	120	1983–2018; validation 2019–2020; test 2021–2024

MMLU-Pro uses five-fold category-stratified cross-validation over its 70 non-test questions, with 56 training and 14 validation queries per fold. The GPQA non-test pool is formed by subtracting the 448-question Main set from the 546-question Extended set, then splitting the remaining 98 questions into 78 training and 20 validation queries. These small development sets motivate reporting repeated-run variability rather than treating any single prompt draw as deterministic performance.

Black-box access and supervision. Zero-shot transfer freezes the source-trained planner and applies it without collecting correctness labels from the target API. Validation-adapted transfer updates only the planner using 32 non-test target-API queries and 512 target completions per benchmark. This target-adaptation budget is separate from the source-scorer validation folds above; it never includes final test queries. Both settings access the target through text input and output only, without weights, gradients, or hidden states. Only the first is zero-shot with respect to target-specific correctness supervision. The single-call rows in Table 8 distinguish the two settings.

B Additional discussion

B.1 Compute and performance accounting

Online and offline costs. Maze scores cached input-side features before invoking the target model. Its online cost comprises one query-encoder forward, path scoring, and the selected target generations. For sequential execution, latency is approximately $T_{\text{enc}} + T_{\text{score}} + \sum_{i=1}^{N} T_{\text{gen},i}$. Batching and concurrency can change this latency without changing the number of target calls. Exemplar encoding, correctness-label collection, and scorer training are offline costs and must be amortized over deployment. An analogous use of learned mappings to reduce online computation appears in three-dimensional reconstruction, where a neural network predicts loudspeaker signal-allocation weights [Zhang et al., 2025]. Table 7 reports measured online latency, not total project cost.

Token usage. Maze changes the prompt but not the target's autoregressive decoder or generation cap. In the reported MATH500 measurement, mean output lengths are 642 tokens for random single-call prompting and 647 for Maze. This observation does not imply equal token costs across tasks or selected exemplars. With a multi-call policy, every generated response is charged, including invalid or truncated responses.

Scoring complexity. Let d be the encoder width, L the number of views, r the projected dimension, h the fusion width, and K the exemplar count. After fitting the scorer, linear node projections can be cached at cost $O(|\mathcal{E}|Ldr)$ and storage $O(|\mathcal{E}|Lr)$. The query projections cost $O(Ldr)$ in addition to the encoder forward. Scoring M paths then costs

$$O(ML\left[K(W+1)r + (W+1)rh\right]), \tag{11}$$

which is $O(MLr(K+h))$ for fixed neighborhood radius $W = 2$. Selecting the maximum costs $O(M)$; selecting the top N can use $O(M \log N)$ comparisons. For independent-pool MultiMaze, $M = NM_{\text{pool}}$ candidate instances are scored; repeated-prompt allocation uses $M = M_{\text{pool}}$ and reuses the selected path. Both make N target calls while sharing one query encoding. The relative importance of these terms depends on the target, context length, hardware, and runtime.

B.2 Design motivation

Compact geometry. If reward-relevant path features lie near a low-dimensional subspace and utility varies smoothly, a compact representation can preserve utility up to projection and approximation error (Theorem C.7). This motivates a low-rank scorer, but does not guarantee that any fixed projection or small MLP will learn the required ordering.

Directed path structure. A relevance sum does not encode exemplar order. Maze instead represents directed transitions, modulates them by source-node contributions, and combines layerwise path summaries in the context of the query. The learned operators are shared across nodes, allowing the same compact scorer to evaluate many candidate sequences.

Local supervision. Correctness-flipping edits provide supervision on prompt changes whose downstream behavior differs, rather than only on semantic similarity. Many pairs can share a query or exemplar, so the raw pair count is not the number of independent observations. Query-disjoint evaluation and repeated complete runs are important complements to pair-level training accuracy.

Selection and aggregation. Prompt selection changes the distribution from which responses are generated; aggregation operates on responses already obtained. Their effects are separated by using the same upstream pool and the same aggregator in the controlled comparisons. A stronger prompt proposal can improve voting or oracle-pick accuracy without contradicting the oracle's optimality on a fixed realized output set.

B.3 Why can single-call *Maze* outperform majority vote?

This subsection provides a theory-level explanation (aligned with Section 3) for why a *single-call Maze* decision can outperform a *width-W* majority-vote / self-consistency baseline, despite using

fewer LLM calls. The key point is that *majority vote is a consensus aggregator*, while *Maze is a selective optimizer* (it approximates a best-of selection over a structured candidate pool via an LLM-free energy). These two aggregation principles behave very differently when (i) errors are correlated, (ii) candidate "competence" is heterogeneous, or (iii) the model exhibits spurious-consensus modes.

Setup (consistent with Appendix C). Fix an instance q. A prompt-path P induces a one-call success probability $r(P;q) \in [0,1]$. *Maze* constructs a pool $\mathcal{P}_D(q)$ and selects $\widehat{P}_D(q) \in \arg\max_{P\in\mathcal{P}_D(q)} E_\theta(P,q)$ before making a *single* model call. We denote the pool-optimal reward by $r^\star_D(q) = \max_{P\in\mathcal{P}_D(q)} r(P;q)$.

B.3.1 Majority vote (self-consistency) and when it is provably beneficial

A common width-W baseline is *self-consistency*: sample W outputs (e.g., W reasoning traces) under a fixed prompt family and return the most frequent final answer. Let $A^\star(q)$ be the (unique) correct final answer, and let $A_1,\dots,A_W$ be the sampled answers. The majority-vote predictor is

$$\widehat{A}_{\mathrm{MV}}(q) \in \arg\max_a \sum_{i=1}^{W} \mathbf{1}\{A_i = a\}, \qquad \mathrm{Succ}_{\mathrm{MV}}(W;q) := \mathbb{P}\Big[\widehat{A}_{\mathrm{MV}}(q) = A^\star(q)\Big]. \tag{12}$$

In the *idealized* binary setting with i.i.d. voters of identical competence $p > 1/2$ (each voter is more likely correct than not, and votes are independent), majority-vote success tends to 1 as $W \to \infty$; for odd W, it increases with the number of voters. This is the classic "wisdom of crowds" regime: majority vote is reliable because *independent* weak signals average out.

B.3.2 When those assumptions fail for LLM sampling

The assumptions underlying the binary Condorcet setting need not describe LLM answer distributions:

Dependence and shared error modes. A shared high-probability wrong answer does not itself imply statistical dependence between separately seeded responses. Even independent samples can repeatedly return the same incorrect mode. Actual dependence, when present, additionally limits the benefit of aggregation.

Heterogeneous competence and spurious consensus. Different traces/prompt-realizations can have sharply different success probabilities. Moreover, the model may have a *high-probability wrong mode* that produces the *same* wrong answer consistently. In such cases, majority vote amplifies the spurious-consensus mode rather than averaging it out.

The distinction between the most frequent answer and the most useful prompt is sufficient to separate the two procedures in the example below.

B.3.3 Majority vote is a "mode seeker", while *Maze* is a "best-of selector"

Majority vote returns the most frequent answer, i.e., it is a *mode-seeking* operator on the answer distribution. In contrast, Section 3 models *Maze* as approximating

$$P^\star_D(q) \in \arg\max_{P\in\mathcal{P}_D(q)} r(P;q), \tag{13}$$

up to an energy-rankability error (Appendix C). Thus, *Maze* behaves like a *best-of* selector on a structured pool: it tries to choose a high-reward candidate even if that candidate is *rare* under naive sampling.

This difference matters because consensus and optimality are not equivalent: a correct solution can be *rare* (low-probability) yet exist as a high-quality candidate, and majority vote can systematically reject it if the wrong mode is more frequent.

B.3.4 A separation between prompt selection and majority vote

The following example isolates the effect of a dominant incorrect response mode.

Theorem B.1 (Spurious-consensus regime where majority vote fails but single-call Maze succeeds)**.** *Fix an instance q with a unique correct answer $A^\star(q)$. Assume there exist two prompt-paths $P_{\mathrm{good}}, P_{\mathrm{bad}} \in \mathcal{P}_D(q)$ such that:*

1. *(**Conditional answers**) Executing P_{good} returns $A^\star(q)$ with probability p_{g}; executing P_{bad} returns a fixed wrong answer $A^\times(q) \neq A^\star(q)$ with probability 1.*
2. *(**Spurious consensus under width sampling**) The majority-vote baseline samples W i.i.d. trials that each follow P_{good} with probability π and P_{bad} with probability $1-\pi$, independently across trials, with $\pi < \frac{1}{2}$.*
3. *(**Maze rankability**) Maze selects P_{good} with probability at least $1-\eta$ (energy ranks the better candidate), and conditional on selecting P_{good} its one-call success is p_{g}.*

Then the width-W majority vote success probability satisfies

$$\mathrm{Succ}_{\mathrm{MV}}(W; q) \;\le\; \exp\Big(-2W\big(\tfrac{1}{2}-\pi\big)^2\Big), \tag{14}$$

while the single-call Maze success probability satisfies

$$\mathrm{Succ}_{\mathrm{Maze}}(D; q) \;:=\; \mathbb{P}\Big[Y(\widehat{P}_D(q), q) = A^\star(q)\Big] \;\ge\; (1-\eta)\, p_{\mathrm{g}}. \tag{15}$$

In particular, as $W \to \infty$, majority vote fails exponentially fast (tends to 0), whereas Maze remains bounded below by $(1-\eta)p_{\mathrm{g}}$.

Proof. In the stated mixture, each trial produces the correct answer if and only if P_{good} *and* P_{good} succeeded. But in this construction P_{bad} is deterministically wrong and always produces the *same* wrong answer $A^\times(q)$, so the majority vote is correct only if at least $W/2$ trials followed P_{good} *and* returned $A^\star(q)$. Since $p_{\mathrm{g}} \le 1$, we upper bound majority correctness by the probability that at least $W/2$ trials followed P_{good}: let $S \sim \mathrm{Binomial}(W, \pi)$ be the count of trials using P_{good}. Then

$$\mathrm{Succ}_{\mathrm{MV}}(W; q) \;\le\; \mathbb{P}\big[S \ge \tfrac{W}{2}\big] = \mathbb{P}\big[S - \mathbb{E}S \ge \big(\tfrac{1}{2}-\pi\big)W\big] \;\le\; \exp\Big(-2W(\tfrac{1}{2}-\pi)^2\Big), \tag{16}$$

where the last step is Hoeffding's inequality for Bernoulli sums.

For *Maze*, according to the rankability assumption, it selects P_{good} with probability at least $1-\eta$ and then succeeds with probability p_{g}, so $\mathrm{Succ}_{\mathrm{Maze}}(D; q) \ge (1-\eta)p_{\mathrm{g}}$. □

Interpretation. Theorem B.1 gives a concrete regime in which selection and voting differ: a wrong-but-high-probability reasoning mode produces a stable wrong answer (strong internal "consensus"), and the majority vote reinforces it. Maze can win by *selecting* a better prompt-path (or reasoning scaffold) that is rare under naive sampling.

B.3.5 Connection to Section 3: depth can dominate width

Section 3 formalizes that *Maze*'s depth-D search enlarges $\mathcal{P}_D(q)$ and, with a rankable energy, approximates the pool-optimal $r^\star_D(q)$. The majority vote, in contrast, spends the budget on repeated stochastic sampling *without* changing the underlying prompt structure. Therefore: (i) when improvements mainly come from *finding a better prompt-path* (large promptability gap), depth-based selection is the right lever. (ii) when samples are highly correlated or dominated by a spurious-consensus wrong mode, increasing width can be ineffective or even harmful, while depth-based selection can still succeed.

In short, single-call *Maze* can outperform majority vote because it optimizes *which computation to run* (selecting a high-reward prompt-path) instead of merely repeating the same computation many times and hoping the majority corrects the bias.

B.4 Scope and limitations

The exemplar bank and candidate proposal determine which trajectories are available: ranking cannot recover a useful path that is absent from the explored pool. The scorer can also misrank paths

under domain shift or proxy–target mismatch. Larger pools increase coverage opportunities but do not force ranking error to decrease, and compact geometric features need not expose all reward-relevant information. Offline correctness supervision and training are substantial costs, particularly for low-volume deployments.

The theoretical results concern expected one-call prompt utility under explicit coverage and rankability assumptions. They do not generally equate one stochastic generation with the probability that at least one of many generations succeeds. The experiments cover a bounded set of tasks, models, and operating points. Repeated-run analysis is concentrated on GPQA and AIME rather than every cell, and the smaller AIME evaluations carry substantial uncertainty. Performance depends on the feature configuration and call allocation as well as the encoder and target model.

Broader impacts. Reducing repeated target-model generation can make reasoning systems more accessible, but lower per-query cost does not establish a reduction in total energy use or deployment volume. A planner trained on correctness alone can retain biases or unsafe behavior from its exemplar bank and target model, and confident prompt selection is not a correctness or safety certificate. Deployment should preserve the target's safeguards, use appropriately licensed and privacy-reviewed exemplars, and retain validation appropriate to the application.

C Formal statements and proofs

C.1 Setup and notation

Fix a query q, a target model $\mathcal{M}$, and a deterministic answer evaluator. For a feasible ordered exemplar path $P \in \mathcal{P}$, let $Y(P, q)$ denote one response under the specified decoding distribution and define

$$r(P; q) = \Pr\{Y(P, q) \text{ is correct}\} \in [0, 1]. \tag{17}$$

The probability concerns response randomness, not the planner's knowledge of the answer. The scorer receives neither the reference answer nor test correctness. A planner budget D produces a candidate pool

$$\mathcal{P}_D(q) = \{P_1, \ldots, P_{M(D,q)}\} \subset \mathcal{P}. \tag{18}$$

Here D denotes planner-side search work; it is not target-model decoding depth or the number of exemplars in a path. For brevity, write $M = M(D, q)$. The selected path and pool-optimal utility are

$$\widehat{P}_D(q) \in \arg\max_{P \in \mathcal{P}_D(q)} E_\theta(P, q), \tag{19}$$

$$r^\star_D(q) = \max_{P \in \mathcal{P}_D(q)} r(P; q), \qquad r^\star(q) = \sup_{P \in \mathcal{P}} r(P; q). \tag{20}$$

Any score ties are resolved by a fixed rule independent of test correctness.

Prompt selection and response selection. For a reference proposal $\pi(\cdot \mid q)$, define the *prompt-selection oracle*

$$\mathrm{PromptBoN}_N(q) = \mathbb{E}\left[\max_{1 \le i \le N} r(P^{(i)}; q)\right], \qquad P^{(i)} \overset{\text{i.i.d.}}{\sim} \pi(\cdot \mid q). \tag{21}$$

This oracle selects a prompt by its one-call success probability. In contrast, Oracle-Pick@N in the experiments selects among N already generated answers, succeeding whenever at least one is correct. If responses are independent conditional on the sampled prompts, its success probability is

$$\mathrm{Pass@}N(q) = 1 - \mathbb{E}\left[\prod_{i=1}^{N}(1 - r(P^{(i)}; q))\right]. \tag{22}$$

Without conditional independence, Pass@N is still the probability of at least one correct response, but need not have this product form. The two oracles coincide for deterministic decoding, where $r(P; q) \in \{0, 1\}$, but not in general. An answer-selection rule, including majority vote, cannot outperform Oracle-Pick on the *same realized response set*. This statement does not compare different proposal mechanisms or algorithms that synthesize a new answer.

C.2 Coverage and rankability

We state both assumptions for a fixed q and deterministic budget D. Their probabilities include candidate-pool randomness and, where relevant, scorer-training randomness.

Coverage. For $\varepsilon > 0$, let $\delta_{\mathrm{cov}}(\varepsilon; D, q) \in [0, 1]$ satisfy

$$\Pr\{r^\star_D(q) \ge r^\star(q) - \varepsilon\} \ge 1 - \delta_{\mathrm{cov}}(\varepsilon; D, q). \tag{23}$$

Any asymptotic conclusion additionally assumes that this failure probability tends to zero as the search budget grows. For example, if paths are sampled independently from π and π assigns probability $p_\varepsilon > 0$ to paths with utility at least $r^\star(q) - \varepsilon$, then $\delta_{\mathrm{cov}} \le (1 - p_\varepsilon)^M$. This example makes explicit why more search cannot recover useful paths outside the proposal's support.

Strict margin-rankability. For $\gamma(D, q) \ge 0$ and $\eta(D, q) \in [0, 1]$, suppose that, with probability at least $1 - \eta(D, q)$, the following implication holds simultaneously for all $P, P' \in \mathcal{P}_D(q)$:

$$r(P; q) > r(P'; q) + \gamma(D, q) \quad \Longrightarrow \quad E_\theta(P, q) > E_\theta(P', q). \tag{24}$$

Strict score separation is needed only for utility gaps exceeding γ. Randomly breaking ties does not itself establish this condition. A larger candidate pool improves coverage under suitable sampling assumptions, but need not improve simultaneous ranking accuracy.

C.3 Selection lemmas

Lemma C.1 (Exact order-consistency)**.** *If $r(P;q) > r(P';q)$ implies $E_\theta(P,q) > E_\theta(P',q)$ for every pair in the pool, every score maximizer attains $r^\star_D(q)$.*

Proof. If a score maximizer had smaller utility than a utility maximizer, strict order-consistency would assign it a smaller score, a contradiction. □

Lemma C.2 (Margin-rankability and selection regret)**.** *Under Eq. (24), with probability at least $1-\eta(D,q)$,*

$$r(\widehat{P}_D(q);q) \geq r^\star_D(q) - \gamma(D,q). \tag{25}$$

Moreover,

$$\mathbb{E}[r(\widehat{P}_D(q);q)] \geq \mathbb{E}[r^\star_D(q)] - \gamma(D,q) - \eta(D,q). \tag{26}$$

Proof. On the rankability event, a selected path with utility below $r^\star_D - \gamma$ would have a strictly smaller score than a pool utility maximizer. This contradicts its selection. The nonnegative regret $r^\star_D - r(\widehat{P}_D;q)$ is at most γ on this event and at most one otherwise. Taking expectations proves the second inequality without assuming independence between the pool optimum and the ranking event. □

C.4 Pool-optimality and prompt-level best-of-N

Theorem C.3 (Coverage and ranking bound)**.** *Under Eqs. (23) and (24),*

$$\Pr\{r(\widehat{P}_D(q);q) \geq r^\star(q) - \varepsilon - \gamma(D,q)\} \geq 1 - \eta(D,q) - \delta_{\mathrm{cov}}(\varepsilon;D,q), \tag{27}$$

and

$$\mathbb{E}[r(\widehat{P}_D(q);q)] \geq r^\star(q) - \varepsilon - \gamma(D,q) - \eta(D,q) - \delta_{\mathrm{cov}}(\varepsilon;D,q). \tag{28}$$

If, in addition, the pool contains M i.i.d. draws from π, or its maximum utility stochastically dominates that of such draws, then for every integer $N \geq 1$,

$$\mathbb{E}[r(\widehat{P}_D(q);q)] \geq \mathrm{PromptBoN}_{\min\{N,M\}}(q) - \gamma(D,q) - \eta(D,q). \tag{29}$$

Proof. On the intersection of the coverage and rankability events, Lemma C.2 gives $r(\widehat{P}_D;q) \geq r^\star_D - \gamma \geq r^\star - \varepsilon - \gamma$. A union bound proves Eq. (27). For the expected bound, global regret is at most $\varepsilon + \gamma$ on this intersection and at most one elsewhere. This gives Eq. (28), including cases in which the stated lower bound is negative.

Under the pool-dominance condition, $\mathbb{E}[r^\star_D(q)] \geq \mathrm{PromptBoN}_M(q)$. Apply Eq. (26) and monotonicity of $\mathrm{PromptBoN}_M$ in M to obtain Eq. (29). No factorization of a pool-dependent reward and a ranking-event indicator is used. □

The selected one-call utility approaches $r^\star(q)$ only when coverage, ranking error, and the utility margin all vanish along the chosen sequence of budgets and scorers. These conditions do not follow from increasing M alone.

C.5 Promptability gap

Definition C.4 (Promptability gap)**.** For the random-prompt baseline, define

$$r_{\mathrm{base}}(q) = \mathbb{E}_{P\sim\pi}[r(P;q)], \qquad \Delta(q) = r^\star(q) - r_{\mathrm{base}}(q). \tag{30}$$

Corollary C.5 (Available gain from prompt selection)**.** *Under the assumptions of Theorem C.3,*

$$\mathbb{E}[r(\widehat{P}_D(q);q)] - r_{\mathrm{base}}(q) \geq \Delta(q) - \varepsilon - \gamma(D,q) - \eta(D,q) - \delta_{\mathrm{cov}}(\varepsilon;D,q). \tag{31}$$

Proof. Subtract $r_{\mathrm{base}}(q)$ from Eq. (28) and substitute Eq. (30). □

The gap measures how much a task can benefit from better prompts under the fixed target model and prompt family. Task difficulty and model size alone do not determine this gap or the ranking error. They therefore do not imply a universal monotonic trend in the gains from *Maze*.

C.6 Conditional width–depth conversion

Corollary C.6 (Conversion for prompt-level best-of-W selection)**.** *Fix an integer $W \geq 1$. Under the pool-dominance and rankability conditions of Theorem C.3, if $M(D, q) \geq W$, then*

$$\mathbb{E}[r(\widehat{P}_D(q); q)] \geq \mathrm{PromptBoN}_W(q) - \gamma(D, q) - \eta(D, q). \tag{32}$$

Consequently, if a budget and scorer satisfy $M(D, q) \geq W$ and $\gamma(D, q) + \eta(D, q) \leq \delta$, one planned call is within δ of the prompt-selection oracle.

Proof. Set $N = W$ in Eq. (29). □

The result concerns selection by expected one-call utility. For example, if every prompt succeeds with probability 0.6 under independent binary-answer decoding, then $\mathrm{PromptBoN}_3 = 0.6$, while three-answer majority vote succeeds with probability $3(0.6)^2(0.4) + (0.6)^3 = 0.648$ and Pass@3 equals $1 - 0.4^3 = 0.936$. No prompt ranker can remove this residual response randomness. This distinction explains why planner-side selection and multi-call aggregation remain complementary.

C.7 Compact reward-relevant representations

A low-dimensional representation can support ranking when it retains the features on which utility depends. Let $\phi(P, q) \in \mathbb{R}^p$ be a path feature map, let $B \in \mathbb{R}^{d \times p}$ have orthonormal rows, and set $\Pi = B^\top B$. Assume, uniformly on the pool,

$$r(P; q) = g(\phi(P, q)), \qquad \|\phi(P, q) - \Pi\phi(P, q)\|_2 \leq \rho, \tag{33}$$

where $g : \mathbb{R}^p \to [0, 1]$ is L_g-Lipschitz on a domain containing both the original features and their projections.

Theorem C.7 (Compact energy on an approximately low-dimensional pool)**.** *With $\widetilde{g}(z) = g(B^\top z)$,*

$$\max_{P \in \mathcal{P}_D(q)} |\widetilde{g}(B\phi(P, q)) - r(P; q)| \leq L_g\rho. \tag{34}$$

If a learned scalar scorer approximates $\widetilde{g}$ on the projected pool with uniform error at most α, it strictly preserves utility order whenever the utility gap exceeds $2(L_g\rho + \alpha)$.

Proof. For each path, $\widetilde{g}(B\phi) = g(B^\top B\phi) = g(\Pi\phi)$. The Lipschitz condition gives $|g(\Pi\phi) - g(\phi)| \leq L_g\|\Pi\phi - \phi\|_2 \leq L_g\rho$. Adding the scorer approximation error yields a uniform error $\beta = L_g\rho + \alpha$. For $r(P; q) > r(P'; q) + 2\beta$, the estimated score of P is at least $r(P; q) - \beta$, strictly exceeding $r(P'; q) + \beta$, which upper-bounds the score of P'. □

This is a sufficient representation condition, not a guarantee that the particular *Maze* parameterization will learn the required scorer from a finite training set. Empirical effective rank and ranking statistics are useful diagnostics, but do not establish a uniform approximation bound or certify all pairs in an arbitrarily large pool.

D Related work

Few-shot reasoning and exemplar selection. Few-shot learning exposes useful task behavior through demonstrations without updating model parameters [Brown et al., 2020]. Chain-of-thought prompting extends demonstrations with intermediate reasoning and improves arithmetic, commonsense, and symbolic reasoning [Wei et al., 2022]. Its performance depends on exemplar content and order [Lu et al., 2021, Guo et al., 2024]. Learned retrievers use language-model feedback to train efficient example-selection models [Rubin et al., 2021, Wang et al., 2023]. Like these methods, Maze can cache exemplar representations. Its distinguishing decision variable is the joint ordered trajectory, rather than a sum of independently evaluated exemplar relevance scores. Complementary retrieval-augmented approaches organize external knowledge through reasoning graphs [Xu et al., 2024] and guide knowledge retrieval and evaluation using reasoning logic [Xu et al., 2025].

Prompt-flatness and ordering-aware optimization. pFLAT selects prompts using a flatness criterion based on sensitivity to perturbations of model parameters [Shen et al., 2023]. EASE uses an ordering-aware neural predictor and neural-bandit optimization to identify an exemplar sequence that performs well across task queries [Wu et al., 2024]. These are learning- or optimization-based prompt-selection methods, not merely similarity heuristics, and both can be evaluated with one target call. Maze instead learns a query-conditioned score over complete directed paths and uses it to screen a candidate pool before generation. The learned-ranker controls separate the benefit of correctness supervision from that of its structural parameterization.

Test-time scaling and search. Self-consistency aggregates sampled reasoning outputs [Wang et al., 2022]; Tree of Thoughts and Graph of Thoughts organize search over generated intermediate states [Yao et al., 2023, Besta et al., 2024]. Analyses of compound systems show that adding calls can have non-monotonic effects when question difficulty and answer distributions are heterogeneous [Chen et al., 2024]. Compute-optimal test-time scaling adapts inference strategies to the problem and available budget [Snell et al., 2024]. Maze is complementary to these output-side procedures: it allocates computation to choosing the prompt before sampling, and MultiMaze can combine that choice with answer aggregation.

Theoretical perspective. Best-of-N analyses characterize selection among model outputs and the effects of verifier or reward-model policies [Beirami et al., 2024]. Surrogate-based solvers for hidden monotone variational inequalities [D'Orazio et al., 2025] and recursive gradient play in minimax games [Liu and Pavel, 2022] address continuous optimization, whereas our analysis isolates coverage and ranking of discrete prompt trajectories before generation. The distinction between a prompt's expected success and a realized response's correctness is essential under stochastic decoding. The resulting guarantees concern selection regret and conditional width–depth conversion, not a universal replacement of repeated generation by one call.

E Implementation and reproducibility

E.1 Representations and prompt construction

Maze separates the representation of a prompt trajectory from its execution by the target model. The encoder, geometric scorer, prompt renderer, and target adapter communicate through the query and ordered exemplar identities. A configuration fixes the node text, encoder checkpoint, selected views, token pooling, and numerical precision. The scorer consumes input-side features and returns a selected path; the renderer assembles the corresponding complete prompt.

Node encodings. Alongside the full-layer benchmark configuration in Section 4.2, two compact configurations use four exemplars and therefore six path nodes. The *question-node* configuration represents exemplar questions in four encoder views and uses a zero start vector. The *full-QA* configuration represents the system instruction and complete question–answer/rationale exemplars in 32 views. Both send four complete exemplars to the target model. The question-node representation compresses the information available to the planner, not the text supplied to the target.

Pooling and cached states. For fresh text encoding, one implementation averages the token states in each selected layer, accumulating in float32 and storing the result in float16. Numerical reproduction of a trained scorer uses its corresponding feature tensors, whose pooling, node text, and precision are part of the configuration. Changing these choices changes the scorer's inputs, even when the renderer produces the same target prompt. The system and bank states are cached; the query is encoded once before any path is selected.

E.2 Geometric scorer instantiations

Shared projections. For views with common width d, a shared relation map and node map are

$$\mu\big(\mathbf{h}_x^{(l)}, \mathbf{h}_y^{(j)}\big) = \big(\mathbf{h}_x^{(l)} - \mathbf{h}_y^{(j)}\big)\mathbf{W}_\mu, \qquad \mathbf{W}_\mu \in \mathbb{R}^{d\times r}, \tag{35}$$

$$\mathscr{F}_m\big(\mathbf{h}_i^{(l)}\big) = \mathbf{h}_i^{(l)}\mathbf{W}_m, \qquad \mathbf{W}_m \in \mathbb{R}^{d\times r}, \tag{36}$$

using row-vector notation. The fusion map is a bias-free MLP,

$$\mathcal{G}(\mathbf{m}, \mathbf{E}) = \tanh\big([\mathbf{m}; \mathbf{E}]\mathbf{W}_g^{\mathrm{hid}}\big)\,\mathbf{W}_g^{\mathrm{out}}, \tag{37}$$

where $\mathbf{W}_g^{\mathrm{hid}} \in \mathbb{R}^{2r\times h}$ and $\mathbf{W}_g^{\mathrm{out}} \in \mathbb{R}^{h\times r}$. These shared projections and the fusion MLP have $2dr + 3rh$ parameters, before scalar readout or auxiliary heads. With $\sigma = \tanh$ and $\mathcal{N}(l) = \mathcal{N}_2(l) = \{j : 1 \le j \le L,\ 0 < |j - l| \le 2\}$, Eq. (2) uses a two-layer neighborhood on each side where available.

Scalar fusion. The full-layer scalar configuration takes $\mathcal{C}(\mathbf{u}) = \sum_{k=1}^{r} u_k$, so $\mathcal{C}(\mathcal{G}(\mathbf{m}, \mathbf{E}) \odot \mathbf{E}) = \langle \mathcal{G}(\mathbf{m}, \mathbf{E}), \mathbf{E}\rangle$. It uses an identity readout, $\alpha_l = 1$, and $\mathcal{A} = \{1, \dots, L\}$ in Eq. (5). For interior layers $\mathcal{I} = \{3, \dots, L-2\}$, $\mathcal{J}(l) = \mathcal{N}_2(l)$; for boundary layers, $\mathcal{J}(l) = \varnothing$. Thus all layers contribute, with cross-view fusion in the interior and within-view fusion at the boundary. The benchmark configuration uses $r = 64$ and $h = 96$.

Vector fusion. The compact heads normalize each node/view independently before projection,

$$\overline{\mathbf{h}}_i^{(l)} = \sqrt{d}\,\frac{\mathbf{h}_i^{(l)}}{\max(\|\mathbf{h}_i^{(l)}\|_2, 10^{-8})}, \qquad \mathbf{z}_i^{(l)} = \overline{\mathbf{h}}_i^{(l)}\mathbf{W}_\mu, \quad \mathbf{m}_i^{(l)} = \overline{\mathbf{h}}_i^{(l)}\mathbf{W}_m. \tag{38}$$

Normalization uses only that node's coordinates, without test-set statistics or labels. The transition feature is

$$\mathbf{d}_{i,i+1}^{(l)} = \mathbf{z}_i^{(l)} - \mathbf{z}_{i+1}^{(l)} + \tanh\left(\sum_{j\in\mathcal{N}_2(l)} (\mathbf{z}_i^{(j)} - \mathbf{z}_{i+1}^{(l)})\right), \tag{39}$$

followed by the path accumulation in Eq. (3). Setting $\mathcal{C}$ to the identity retains vector coordinates through fusion:

$$\mathbf{F}_P^{(l)} = \mathcal{G}(\mathbf{m}_q^{(l)}, \mathbf{E}_P^{(l)}) \odot \mathbf{E}_P^{(l)} + \tanh\left(\sum_{j \in \mathcal{N}_2(l)} \mathcal{G}(\mathbf{m}_q^{(l)}, \mathbf{E}_P^{(j)}) \odot \mathbf{E}_P^{(j)}\right), \tag{40}$$

$$\mathbf{v}_P = |\mathcal{A}|^{-1/2} \sum_{l \in \mathcal{A}} \mathbf{F}_P^{(l)}, \qquad s(P,q) = \mathbf{v}_P \mathbf{w} + b. \tag{41}$$

Here $\mathbf{w} \in \mathbb{R}^{r \times 1}$ and $b \in \mathbb{R}$ are learned. The success logit s supplies the scalar path score; applying a sigmoid preserves its ranking.

For the four-view question-node head, $\mathcal{A} = \{1, 2, 3, 4\}$ and each neighborhood contains only available views, giving the normalization factor $1/2$. Its default dimensions are $d = 2560$, $r = 64$, and $h = 96$. For the full-QA head, all 32 views supply neighbors, while $\mathcal{A} = \{3, \ldots, 30\}$ contributes to the final score, giving the factor $1/\sqrt{28}$. This head uses $d = 2560$, $r = 16$, and $h = 32$. Scalar fusion reduces the coordinates before cross-view nonlinearity; vector fusion retains them until readout. They are two parameterizations of the same directed, node-weighted, query-conditioned construction.

E.3 Correctness supervision and optimization

Local-edit comparisons. The local-edit experiments in Section 4 sample ordered K-shot paths from the training bank, excluding the current query's exemplar. A content edit replaces one exemplar with an eligible exemplar outside the path; an order edit swaps two positions. The target model executes both prompts, and the pair is retained when evaluated correctness differs. For unchanged outcomes, editing continues for at most $A_{\max} = 16$ attempts. A retained pair consists of consecutive paths at an observed correct–incorrect transition; sequences without such a transition are discarded. The resulting comparisons preserve much of the input while concentrating supervision on behaviorally consequential changes.

Success prediction and reference comparisons. The compact heads use a success logit s_i, correctness target $y_i \in [0, 1]$, and reference index $a(i)$ for the same query. Let $\ell_{\mathrm{BCE}}(s, y) = \log(1+e^s) - ys$, evaluated numerically in a stable form. Their success and within-query ranking losses are

$$\mathcal{L}_{\mathrm{succ}} = \langle \ell_{\mathrm{BCE}}(s_i, y_i) \rangle_w, \tag{42}$$

$$\mathcal{L}_{\mathrm{rank}} = \left\langle \ell_{\mathrm{BCE}}\left(s_i - s_{a(i)}, \frac{1 + y_i - y_{a(i)}}{2}\right) \right\rangle_w, \tag{43}$$

where $\langle \cdot \rangle_w$ is the normalized training-weighted mean. Equation (43) instantiates the pairwise objective in Eq. (7). For a binary correct–incorrect comparison, its positive preference term is $\log(1 + \exp(s^- - s^+))$. The reference comparisons use observed candidate outcomes rather than a requirement to obtain a label flip by editing.

Auxiliary objectives. The full-QA configuration additionally predicts cost and changes relative to the reference path:

$$\mathcal{L} = \mathcal{L}_{\mathrm{succ}} + 0.5\mathcal{L}_{\mathrm{rank}} + 0.1\mathcal{L}_{\mathrm{cost}} + 0.25\mathcal{L}_{\mathrm{uplift}} + 0.25\mathcal{L}_{\mathrm{harm}}. \tag{44}$$

The cost head predicts a scaled log-token target with squared error. The uplift head uses squared error on paired improvement targets, and the harm head uses binary cross-entropy on paired harm targets. Paired losses use same-target, same-seed outcomes; unmatched entries are masked. Without paired supervision, the uplift and harm terms are omitted. Observed correctness and response lengths are training targets, never inference-time inputs. This objective includes no separate local order-edit loss.

The full-QA head is trained with AdamW, learning rate 10^{-3}, weight decay 10^{-3}, at most 40 epochs, and early-stopping patience of five epochs. Queries whose candidate outcomes are all correct within the training fold receive weight 0.1. The local-edit benchmark settings use Adam and the dimensions specified in Section 4.2. Model selection uses non-test queries, and target-specific adaptation uses the disjoint queries described in Appendix A.7.

E.4 Selection and target-call policies

Path scoring. The four-head success selector averages logits, $\overline{s}(P,q) = \frac{1}{4}\sum_{j=1}^{4} s_j(P,q)$, and selects the highest-scoring candidate. Exact ties favor the fixed retrieval reference, followed by the candidate's hash. These scorer evaluations make no target-generation calls. A reference-relative readout instead uses the path representation, the fixed reference representation, and the query to predict uplift $\widehat{u}(P,q)$ and harm $\widehat{h}(P,q)$. Its selection score is $\widehat{u}(P,q) - 0.25\widehat{h}(P,q)$, with the reference assigned score zero. It selects the best positive-scoring alternative, or retains the reference when no alternative has positive score. Both decisions depend only on input-side features and fitted parameters.

Independent-pool allocation. The main MultiMaze experiments construct N independent pools of $M_{\text{pool}} = 100$ paths and select one maximum-energy path from each. Each selection receives one target call, followed by majority vote over the N responses. This scores $100N$ candidate instances while reusing one query encoding. If pools select the same path, its repetition count in Eq. (8) equals the number of requests assigned to it.

Shared-pool allocation. The controlled comparisons select the top-N distinct paths from one shared pool and assign one request to each, giving $J = N$ and $n_j = 1$. Random selection and Maze ranking receive the same upstream candidates and call budget. The comparisons apply voting and oracle pick to the selected responses to isolate proposal quality from aggregation. Voting ties are resolved uniformly using a shared random seed (Appendix A.6).

Repeated-prompt allocation. SC3 selects one complete prompt and executes it in three distinct requests with distinct fixed seeds, giving $J = 1$ and $n_1 = 3$. The three observed responses produce one vote per query. Its Raw MV3 comparator uses three separately sampled ordered prompts, each containing four distinct exemplars from the eligible bank. For these configurations, voting ignores invalid parsed answers and resolves ties by the earliest response in the fixed request order. Invalid and truncated requests remain included in cost accounting. A repeated-prompt vote is evaluated from its three actual requests, whereas single-call accuracy averages individual response correctness; the two estimators are kept separate.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: See the last itemized section of introduction

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See the Section B.4

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: See the Section C

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: See the Section 4.2

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Codes are uploaded in supplemental material.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: See the Section 4.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: See the main results in Section 4.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See the Section 4.1 and paragraph "Fair compute accounting".

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: the research conform with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See the Section B.4

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: See the Section 4.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: See the Section 4.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: the paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: the paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: the paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: the core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.