HyPER: Bridging Exploration and Exploitation for Scalable LLM Reasoning with Hypothesis Path Expansion and Reduction

Shengxuan Qiu [*1,2] **Haochen Huang** [*1,2] **Shuzhang Zhong** [1,2] **Pengfei Zuo** [3] **Meng Li** [1,2]

Abstract

Scaling test-time compute with multi-path reasoning improves the accuracy of foundation models, but often incurs substantial redundant computation. We study how to make such reasoning more resource-efficient under a fixed inference budget. Our key observation is that the value of exploration and exploitation is phase-dependent: early decoding benefits from diverse hypothesis paths, whereas later stages require targeted refinement and reliable answer selection. We introduce *HyPER*, a training-free adaptive test-time compute policy that formulates multi-path reasoning as an online *expand–reduce* control problem. HyPER reallocates compute using lightweight confidence and diversity statistics, dynamically choosing among path branching, short-horizon expansion, token-level refinement, and standard decoding. To improve exploitation without full-path resampling, HyPER further leverages MoE routing diversity for single-token refinement with shared KV states, and uses a length- and confidence-aware voting rule to reduce answer-selection failures. Across four MoE models and diverse reasoning benchmarks, HyPER consistently improves the accuracy–compute trade-off, achieving the Pareto frontier while outperforming prior methods by $8{\sim}10\%$ and reducing token consumption by $25{\sim}40\%$. These results demonstrate that adaptive control is an effective mechanism for resource-efficient foundation-model inference.

1. Introduction

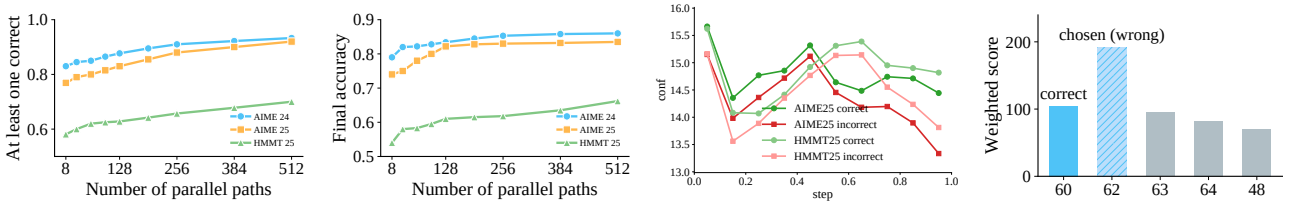
Large language models (LLMs) have demonstrated strong reasoning ability when prompted to generate chain-of-thought (CoT) solutions (Wei et al., 2022), yet a single reasoning path remains brittle on challenging problems (Wang et al., 2022; Zhao et al., 2026). Scaling *test-time compute* improves robustness by sampling or searching over multiple hypothesis paths (Zhang et al., 2025), but its practical value depends on how efficiently the extra budget is allocated between *exploration*—covering diverse candidate solutions—and *exploitation*—refining and selecting promising ones (Snell et al., 2024; Ding et al., 2025). This makes resource-efficient test-time scaling fundamentally an adaptive inference problem.

Existing methods only partially address this trade-off. Tree-based search methods expand intermediate reasoning steps using predefined branching rules and step-level scores (Yao et al., 2023; Xie et al., 2024; Snell et al., 2024; Hooper et al., 2025). Although they provide explicit exploration, their rigid step-level schedules can waste compute and disrupt the continuous reasoning behavior of post-trained models (Lian et al., 2025; Yang et al., 2025). In contrast, parallel reasoning methods such as Self-Consistency and Best-of-N sample complete CoT paths and aggregate final answers (Wang et al., 2022; Brown et al., 2024; Irvine et al., 2023). They preserve native generation, but heavily over-explore redundant paths and provide little generation-time exploitation. Recent adaptive methods improve efficiency through spawning, joining, or pruning (Lian et al., 2025; Wang et al., 2025a; Yu et al., 2025; Fu et al., 2025), but are often either training-dependent or primarily subtractive: they remove weak paths without reallocating the saved budget to refine promising hypotheses. Meanwhile, Mixture-of-Experts (MoE) reasoning models expose routing diversity that can support token-level refinement (Zibakhsh et al., 2025), but token-level gains alone do not decide *when* to branch, refine, or simply continue decoding.

These limitations motivate a closed-loop control view of test-time scaling. Figure 1 summarizes three empirical observations. First, increasing path width raises both the probability of containing at least one correct path and the final accuracy, but the marginal gain quickly diminishes, indicating that late-stage redundant exploration is inefficient. Second, correctness is often not distinguishable early: in Figure 1c, we normalize each generated path by length, av-

[1]Institute for Artificial Intelligence, Peking University, Beijing [2]School of Integrated Circuits, Peking University, Beijing [3]Huawei. Correspondence to: Meng Li <meng.li@pku.edu.cn>.

Proceedings of the 43^{rd} International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026.



(a) Correct-path existence. *(b)* Final accuracy. *(c)* Confidence divergence. *(d)* Voting failure.

Figure 1. Motivation for adaptive expand–reduce control. Wider path sampling improves coverage and accuracy but with diminishing returns; correct and incorrect paths show late-stage confidence divergence after length normalization; and answer aggregation can still fail even when a correct path exists.

erage token confidence at each relative position for correct and incorrect paths, and then aggregate across questions on AIME25 and HMMT25. The resulting dataset-level trends show that correct and incorrect paths have similar early confidence patterns but separate near the tail, suggesting that local late-stage refinement can be more efficient than full-path resampling. Third, even when a correct answer exists among sampled paths, majority or confidence-weighted voting can still select a frequent but wrong answer, exposing an *existence–selection gap*. Together, these observations suggest that an efficient method should adaptively decide when to explore, when to refine, and how to exploit reliable final-answer signals.

We introduce *HyPER* (**Hy**pothesis **P**ath **E**xpansion and **R**eduction), a training-free online policy that formulates multi-path reasoning as an adaptive *expand–reduce* control problem. HyPER monitors lightweight confidence and diversity statistics of the surviving hypothesis pool and dynamically reallocates compute among four actions: standard decoding, path branching, short-horizon expansion, and token-level refinement. This allows HyPER to preserve early exploration while shifting budget toward exploitation when the pool becomes redundant or uncertain.

HyPER makes three contributions. First, we design an online expand–reduce controller that coordinates exploration and exploitation under a fixed test-time budget using lightweight path statistics. Second, we introduce a MoE-based single-token refinement primitive that exploits routing diversity without full-path resampling and shares prefix KV states across token-level proposals. Third, we propose a length- and confidence-aware aggregation rule that improves answer-time exploitation and mitigates the existence–selection gap. HyPER requires no fine-tuning and can be directly applied to post-trained MoE reasoning models.

Across four MoE LLMs and diverse reasoning benchmarks, HyPER consistently improves the accuracy–compute trade-off, achieving the Pareto frontier while outperforming existing test-time scaling baselines by 8–10 percentage points and reducing token usage by approximately 25–40% under comparable compute budgets.

2. Background

Test-time scaling paradigms. LLM reasoning can be improved by allocating extra *test-time compute* along either *depth*, which extends deliberation within a single CoT (OpenAI et al., 2024; Guo et al., 2025; Kimi et al., 2025; xAI, 2025; Muennighoff et al., 2025; Ye et al., 2025), or *breadth*, which explores multiple hypothesis paths (Snell et al., 2024; Welleck et al., 2024). Key design choices include whether compute allocation is adaptive or fixed, what exploitation primitive is used, and whether guidance comes only from the policy model or from an additional verifier/reward model. Tree-based search methods, such as ToT, MCTS, and ETS (Yao et al., 2023; Xie et al., 2024; Snell et al., 2024; Hooper et al., 2025), expand intermediate reasoning states with step-level scores, but often rely on rigid schedules or auxiliary verifiers that can be costly for post-trained reasoning models. Parallel reasoning methods, such as Self-Consistency and Best-of-N (Wang et al., 2022; Brown et al., 2024; Irvine et al., 2023), preserve native generation by sampling complete paths, but spend much of the budget on redundant hypotheses and exploit mainly at final aggregation. Recent adaptive variants use spawn–join execution or confidence-guided pruning (Lian et al., 2025; Wang et al., 2025a; Yu et al., 2025; Fu et al., 2025), yet many remain training-dependent or primarily subtractive: they remove low-quality paths without reallocating the saved budget to refine promising ones. Appendix A provides a taxonomy.

Token-level scaling in MoE models. Mixture-of-Experts (MoE) models expose an additional token-level scaling dimension through expert-routing diversity. Recent methods such as RoE (Zibakhsh et al., 2025) exploit this by perturbing expert routing to generate multiple token-level proposals and aggregate them into a refined prediction. This provides efficient local exploitation, but existing token-level methods are typically always-on and agnostic to the global reasoning state. They therefore do not decide when compute should be spent on token refinement versus path-level exploration. This leaves a central challenge: *how to dynamically allocate a unified test-time budget across path expansion, pruning, token refinement, and answer selection.*

Table 1. Controller actions in HyPER.

Action	Role	Trigger	Operation
NONE	Standard decoding	High confidence, low entropy, enough diversity	Continue all active paths; no extra compute.
SINGLETOKEN	Token refinement	Low confidence, high entropy, or low token consensus	Refine the next token with MoE routing diversity; path count unchanged.
BRANCH	Path exploration	Low diversity and low consensus	Fork each survivor into $r_t = \lceil W/S_t \rceil$ continuations to restore coverage.
MULTITOKEN	Local expand–reduce	Low diversity with confidence dispersion	Spawn r_t children, decode $m = T$ tokens, and keep the best child per root.

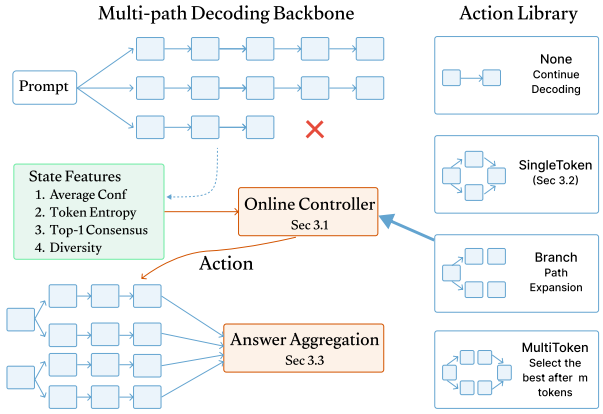


Figure 2. Overview of HyPER.

3. HyPER Design

Figure 2 overviews HyPER. HyPER formulates multi-path reasoning as *online expand–reduce control* over a surviving hypothesis pool $\mathcal{S}_t$. Under a fixed test-time budget, it couples three components: (i) a lightweight controller that periodically chooses among standard decoding, path branching, short-horizon expansion, and token-level refinement (§3.1); (ii) a MoE-based single-token refinement primitive that exploits routing diversity without full-path resampling (§3.2); and (iii) a length- and confidence-aware voting rule that improves final answer selection (§3.3). An always-on confidence-pruning filter removes low-quality paths, while the controller reallocates compute only over the remaining survivors.

Online control signals. HyPER uses lightweight, training-free statistics to summarize the quality and diversity of $\mathcal{S}_t$. For reliability, we track mean token confidence $\bar{C}_t$, top-k entropy H_t, and top-1 consensus β_t, where β_t is the vote share of the most common next token across active paths, following DeepConf (Fu et al., 2025). For diversity, we combine distribution-level divergence $D_{\text{dist},t}$ and suffix-level edit distance $D_{\text{seq},t}$ (Xia et al., 2025; Zheng et al., 2025) into $D_t = \eta D_{\text{dist},t} + (1-\eta) D_{\text{seq},t}$. Low D_t indicates trajectory collapse and favors exploration, whereas high D_t suggests sufficient coverage and allows the controller to prioritize refinement or continue decoding. All signals are inexpensive to compute online and are normalized using warm-up statistics. Full metric definitions are provided in Appendix C.

3.1. Online Path Expansion and Reduction

HyPER controls the surviving path pool $\mathcal{S}_t$ through periodic expand–reduce decisions. At every T decoding steps, the controller reads the normalized confidence and diversity statistics defined above and selects an action $a_t \in \{\text{NONE}, \text{SINGLETOKEN}, \text{MULTITOKEN}, \text{BRANCH}\}$. The action is applied to all surviving paths after always-on confidence pruning. Let $S_t = |\mathcal{S}_t|$ be the number of survivors and W the target path budget. To compensate for pruning-induced shrinkage, HyPER sets the per-path expansion factor to $r_t = \lceil W/S_t \rceil$, subject to the global width cap. Thus, when many paths have been pruned, the remaining survivors receive more exploration or refinement budget.

Table 1 summarizes the action semantics. NONE performs standard decoding when the pool is already reliable and diverse. SINGLETOKEN applies the token-level refinement primitive in §3.2 to each active path, improving the next-token decision without increasing the number of paths. BRANCH forks each survivor into r_t independent continuations, increasing hypothesis coverage when trajectories collapse. MULTITOKEN performs a short-horizon expand–reduce operation: each survivor spawns r_t children, the children decode for $m = T$ tokens, and only the highest-confidence child under each root is kept while the others are discarded and their KV caches released.

The controller uses fixed averaging scores over normalized statistics rather than learned policies or dataset-specific tuning. Intuitively, high confidence and high diversity favor NONE; local uncertainty favors SINGLETOKEN; diversity collapse favors BRANCH; and low diversity with confidence dispersion favors MULTITOKEN. The complete pipeline is shown in Algorithm 1, and the fixed scoring rules are provided in Appendix B.

3.2. Single-Token Expand-and-Aggregate

Addressing the finding that correct and incorrect paths often diverge late and are best separated by tail confidence (Observation 2), this module targets efficient *test-time exploitation* without the overhead of resampling full paths. We instantiate SINGLETOKEN as a token-level *expand–reduce* operator that leverages the latent routing diversity of MoE models. For the current token, we *expand* the computation into $K = r_t = \lceil W/S_t \rceil$ stochastic expert-routed proposals and then *reduce* them into a single refined logit vector via a reuse-aware two-pass sampler and confidence-weighted aggregation (Figure 3). Crucially, this mechanism stabilizes the tail confidence needed for correctness, while enforcing *intra-token diversity* to prevent refinement from collapsing path-level exploration.

Step 1: Gumbel-noise route sampling. To initiate the expansion, we first perturb the MoE router logits $\mathbf{r} \in \mathbb{R}^E$ for the current token. We draw K Gumbel noise vectors $\mathbf{g}^{(k)} \in \mathbb{R}^E$ with $g_e^{(k)} \sim \mathrm{Gumbel}(0,1)$, and construct a dense score matrix $S^{(0)} \in \mathbb{R}^{E\times K}$ where:

$$S^{(0)}_{:,k} = \mathbf{r} + \tau\, \mathbf{g}^{(k)}, \qquad k \in \{0,\dots,K-1\}, \tag{1}$$

with $\tau = 0.5$ controlling the exploration strength.

Step 2: Two-pass expert sampling. To enforce intra-token diversity, we feed $S^{(0)}$ into a fully vectorized two-pass sampling scheme rather than selecting experts independently. In the first pass, we apply standard top-m routing (where m is the backbone's fixed gate size) to each column of $S^{(0)}$, yielding a sparse matrix S. To discourage expert collisions across the K routes, we build a penalty matrix $\Delta \in \mathbb{R}^{E\times K}$ via a *column-wise* prefix accumulation. Setting the first column to zero ($\Delta_{:,0} = \mathbf{0}$), subsequent columns aggregate prior expert usage via $\Delta_{:,k} = \phi\big(\sum_{t=0}^{k-1} S_{:,t}\big)$ using a smooth squashing function $\phi(\cdot)$ (e.g., $\tanh$). In the second pass, we reuse the original dense scores and apply this deterministic penalty: $\widetilde{S}^{(2)} = S^{(0)} - \lambda\,\Delta$. A final top-$m$ routing on $\widetilde{S}^{(2)}$ yields the diversified expert selections. Pseudo-code is provided in Appendix E.

Step 3: Confidence-weighted logit aggregation. Once the K diverse expert routes are forwarded, we obtain K candidate next-token logit vectors. To reduce them into a single reliable prediction, we apply confidence-weighted aggregation. Each route inherits a *token confidence score* C_t derived from its predictive distribution (Equation 7), where higher C_t indicates lower uncertainty. We convert these scores into smoothed weights to softly combine the candidate logits, amplifying confident routes while reducing variance and preventing the diversity penalty from degrading generation quality. While involving multiple passes, these vectorized matrix operations impose negligible latency compared to generating a new token. By forcing local diversity without full-path resampling, SINGLETOKEN offers a high-efficiency exploitation primitive.

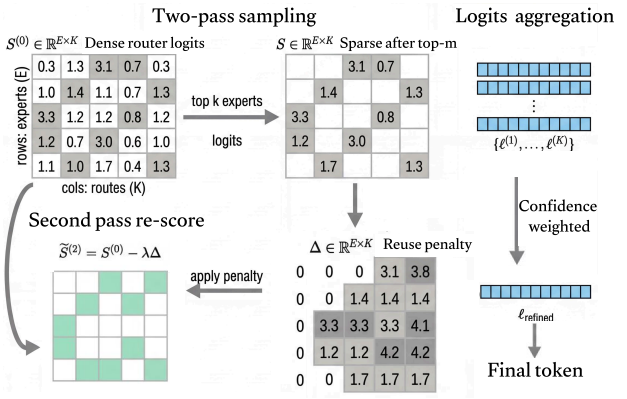


Figure 3. Two-pass expert sampling for single-token aggregation.

Memory optimization via KV cache sharing. Computationally, naive single-token expansion would multiply memory costs by K. To avoid this, we share the prefix KV states across the K routed proposals and localize stochastic routing only to the current token. Thus, KV memory scales strictly with the number of surviving paths (capped by $S_{\max}$) rather than with K (Zibakhsh et al., 2025). Details are provided in Appendix F.

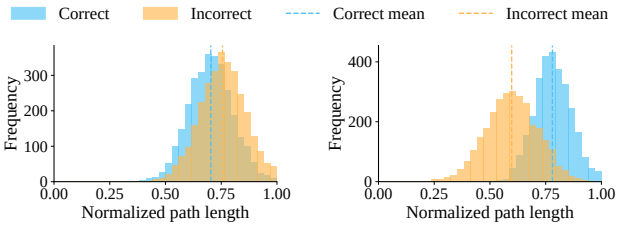
3.3. Answer Aggregation with Length- and Confidence-Aware Voting

To address the *existence–selection gap*, where having at least one correct path does not guarantee selecting the correct final answer, we use a structural signal for *answer-time exploitation*. As shown in Figure 4 and further analyzed in Appendix G.2, **confidence pruning inverts the typical length bias**: *surviving correct paths become systematically longer* than incorrect ones, as erroneous reasoning is frequently exposed by low-confidence steps and terminated early.

Leveraging this asymmetry, we collect all surviving and pruned paths that produced a scalar answer. Let $\mathcal{P}$ denote this set; for each path $p \in \mathcal{P}$, let a_p be its answer, L_p its length, and $\hat{c}_p$ its global average confidence. We emphasize that length serves as a robust signal *only* when coupled with confidence pruning, rather than as a universally valid metric (verified in Appendix G.3). We first restrict attention to the top-K_a answers by majority count, and then select the final answer by aggregating normalized length and confidence over supporting paths:

$$\hat{a} = \arg\max_{a\in\mathcal{A}_K} \sum_{p\in\mathcal{P}:\, a_p=a} \left(\lambda_{\text{len}} \frac{L_p}{\sum_{q\in\mathcal{P}} L_q} + \lambda_{\text{conf}} \frac{\hat{c}_p}{\max_{q\in\mathcal{P}} \hat{c}_q} \right),$$

$$\mathcal{A}_K = \mathrm{TopK}_a\left(\sum_{p\in\mathcal{P}} \mathbf{1}[a_p = a] \right). \tag{2}$$



(a) Before pruning. *(b)* After pruning.

Figure 4. Length asymmetry induced by confidence pruning.

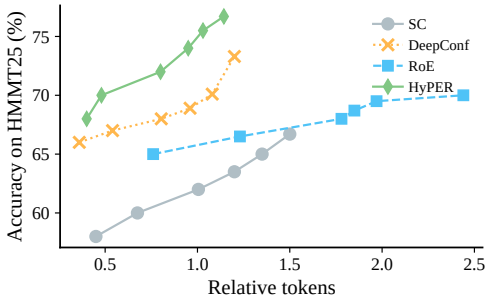


Figure 5. Accuracy–compute Pareto curves on HMMT25 under varying budgets $S_{\max} \in \{32, 64, 80, 128, 256, 512\}$. The relative token cost is normalized by the SC baseline at budget $S_{\max} = 80$.

Here λ_{len} and λ_{conf} control the relative contributions of rationale length and confidence. This rule is compatible with confidence pruning and expand–reduce control; Appendix G.4 provides empirical analyses of the length signal and robustness to long incorrect paths.

4. Experiments

4.1. Experimental Setup

Decoding, budget, and reproducibility. All methods use the same prompts, answer extraction, SC-style multi-path decoding backend, and sampling hyperparameters. Unless stated otherwise, we set the concurrent path cap to $S_{\max} = 80$ and apply a DeepConf-style warm-up pruning scheme: we draw $N_{\text{init}} = 16$ initial traces, estimate a sliding-window confidence threshold from the top-10 traces, and terminate paths whose confidence falls below the threshold (Fu et al., 2025). All reported results are averaged over five fixed random seeds $(42, 43, 44, 45, 46)$; the $\pm$ values denote standard deviations.

We use effective-budget alignment for fair comparison. A fixed initial width would undercharge HyPER because BRANCH and MULTITOKEN instantiate transient continuations. We therefore align methods by *Total Instantiated Path Count* (N_{inst}): since HyPER typically yields $N_{\text{inst}} \approx 1.5 S_{\max}$, SC and DeepConf are run with an equivalently enlarged initial width. Token cost is normalized by the standard SC-80 baseline ($1.0\times$). For RoE and SINGLETOKEN, each expert proposal is charged as K effective token expansions. In the Pareto analysis, we sweep $S_{\max} \in \{32, 64, 80, 128, 256, 512\}$ under the same effective-budget accounting.

Models, benchmarks, and baselines. We evaluate two MoE reasoning tiers. For hard reasoning, we test Qwen3-30B-A3B-Thinking-2507 and Qwen3-Next-80B-A3B-Thinking (Yang et al., 2025) on AIME24/25 (Mathematical Association of America, 2024; 2025), HMMT25 (HMMT Organization, 2025), and HLE (Phan et al., 2025). For light reasoning, we test OLMoE-1B-7B-0924-Instruct (Muennighoff et al., 2024) and DeepSeek-V2-Lite-Chat (DeepSeek-AI, 2024) on Math500 (Hendrycks et al., 2021), GSM8K (Cobbe et al., 2021), ARC-C, and ARC-E (Clark et al., 2018). Baselines include SC, Self-Certainty (Kang et al., 2025), DeepConf (Fu et al., 2025), and late-stage RoE (Zibakhsh et al., 2025). We use $\eta = 0.4$, $T = 64$, $\lambda = 0.1$, and $(\lambda_{\text{len}}, \lambda_{\text{conf}}) = (0.6, 0.4)$; sensitivity results are in Appendix G.1.

4.2. Main Results

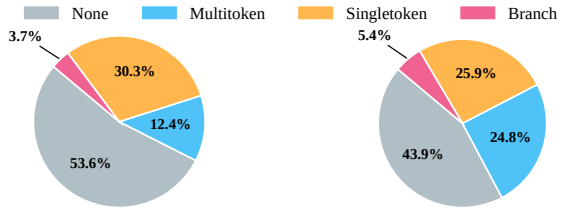
Accuracy–Compute Trade-off. Table 2 summarizes the performance under the effective-budget-aligned setup. Across four MoE backbones, HyPER consistently improves the accuracy–compute trade-off over SC, Self-Certainty, DeepConf, and RoE, with gains most pronounced on hard reasoning benchmarks. Figure 5 further confirms that HyPER achieves the Pareto frontier across varying budget scales ($S_{\max} \in \{32, \ldots, 512\}$), showing that its adaptive control systematically converts test-time compute into useful exploration or exploitation.

Physical Efficiency (Latency & Memory). While Table 2 aligns the mathematical compute budget via effective tokens, resource-adaptive inference fundamentally targets physical execution efficiency. As detailed in Appendix K, despite the dynamic control overhead, HyPER drastically reduces the raw generated tokens via early pruning. For instance, on AIME25 with Qwen3-30B, HyPER cuts raw tokens by $\sim 47\%$ compared to SC. This reduction directly translates to a proportionate drop in end-to-end wall-clock latency (9.98×10^6 ms $\rightarrow 5.88 \times 10^6$ ms). Furthermore, thanks to shared prefix states during local refinement, HyPER successfully lowers the estimated peak KV Cache memory (24.0 GB $\rightarrow$ 18.48 GB), strongly aligning with resource-efficient deployment goals.

Architecture Generality. Although our main evaluation utilizes MoE models to exploit SINGLETOKEN routing diversity, the core expand–reduce controller is architecture-agnostic. We provide a controller-only demonstration on dense models in Appendix I, confirming its broad applicability.

Table 2. Token cost and accuracy across models and datasets. Token is normalized by SC within each model–dataset setting. Accuracy is reported as mean ± standard deviation over five fixed seeds $(42, 43, 44, 45, 46)$.

Model	Dataset	SC		Self-Certainty		DeepConf		RoE		HyPER	
		Token	Acc	Token	Acc	Token	Acc	Token	Acc	Token	Acc
Qwen3-30B	AIME24	1.00	88.0 (± 2.0)	0.94	83.3 (± 3.3)	0.46	92.7 (± 2.7)	1.38	88.7 (± 2.1)	0.54	**94.0** (± 4.0)
	AIME25	1.00	86.0 (± 2.7)	1.13	80.0 (± 6.7)	0.67	90.7 (± 2.6)	1.64	84.7 (± 4.7)	0.71	**95.3** (± 2.0)
	HMMT25	1.00	69.4 (± 2.7)	1.03	68.7 (± 2.1)	0.84	74.0 (± 2.7)	1.78	71.3 (± 2.0)	0.77	**78.7** (± 6.4)
	HLE	1.00	6.5 (± 0.5)	1.15	2.5 (± 2.5)	0.91	11.5 (± 0)	3.55	7.0 (± 0.5)	0.81	**13.5** (± 1.5)
Qwen3-Next	AIME24	1.00	90.7 (± 1.6)	1.06	87.7 (± 2.3)	0.47	94.7 (± 2.0)	1.59	92.0 (± 2.0)	0.61	**97.3** (± 0.6)
	AIME25	1.00	88.0 (± 2.0)	0.95	83.3 (± 3.3)	0.53	94.0 (± 2.7)	1.46	86.7 (± 3.4)	0.59	**96.0** (± 2.7)
	HMMT25	1.00	76.7 (± 3.4)	1.00	70.0 (± 0)	0.76	80.7 (± 6.0)	1.76	78.0 (± 5.3)	0.74	**85.3** (± 5.3)
	HLE	1.00	7.0 (± 0)	1.25	2.5 (± 0)	0.84	11.0 (± 0)	2.98	6.5 (± 0.5)	0.76	**15.5** (± 2.0)
OLMoE	GSM8K	1.00	75.4 (± 0.6)	1.20	76.1 (± 1.3)	0.41	77.6 (± 0.3)	1.64	76.3 (± 0.7)	0.48	**80.3** (± 0.8)
	MATH500	1.00	37.3 (± 1.1)	1.09	37.9 (± 0.7)	0.55	43.7 (± 1.3)	2.35	38.2 (± 2.6)	0.73	**46.7** (± 1.5)
	ARC-C	1.00	63.3 (± 0.3)	0.76	66.4 (± 0.6)	0.32	68.7 (± 0.5)	1.67	65.2 (± 1.3)	0.68	**70.1** (± 0.9)
	ARC-E	1.00	80.1 (± 1.2)	0.88	82.2 (± 0.8)	0.25	84.7 (± 0.7)	2.04	83.8 (± 1.5)	0.36	**86.5** (± 0.9)
DeepSeek-V2	GSM8K	1.00	80.3 (± 1.3)	1.00	81.7 (± 2.4)	0.32	83.3 (± 0.9)	2.31	81.2 (± 0.2)	0.64	**85.5** (± 0.3)
	MATH500	1.00	37.8 (± 0.5)	0.75	38.3 (± 0.4)	0.65	40.2 (± 0.8)	1.93	37.5 (± 2.3)	0.94	**40.8** (± 3.3)
	ARC-C	1.00	65.2 (± 0.7)	1.43	64.7 (± 3.2)	0.53	66.2 (± 0.8)	1.65	63.7 (± 2.2)	0.74	**66.2** (± 2.4)
	ARC-E	1.00	84.3 (± 3.1)	1.04	84.5 (± 1.3)	0.31	86.2 (± 0.9)	1.86	84.3 (± 2.5)	0.56	**86.9** (± 1.8)



(a) AIME24. *(b)* HLE.

Figure 6. Dataset-level controller action frequencies.

4.3. Controller Behavior

We analyze whether HyPER adapts its compute allocation to task difficulty. Figure 6 reports dataset-level action frequencies for Qwen3-30B on AIME24 and HLE. On the easier AIME24 setting, the controller more often selects NONE, avoiding unnecessary expansion when the surviving pool is already confident and diverse. On the harder HLE setting, the fraction of NONE decreases, while BRANCH, MULTITOKEN, and SINGLETOKEN are triggered more frequently. This confirms that HyPER does not merely add fixed extra compute; it reallocates budget according to the evolving confidence and diversity state of the path pool.

4.4. Ablation Studies

Table 3 isolates the effects of adaptive expansion and final aggregation. For expansion, SINGLETOKEN-only improves local token quality but spends more tokens and can reduce path diversity, while MULTITOKEN-only and the manual schedule lack instance-adaptive timing. HyPER matches or improves their accuracy with lower token cost by choosing exploration and exploitation actions according to the current pool state. For voting, majority voting ignores path quality, and confidence weighting can over-favor short but misleading paths. Length-aware confidence voting performs best, confirming that path length and confidence provide complementary selection signals under confidence pruning.

Table 3. Ablations on expansion policies and voting rules. Token cost is normalized to SC.

Variant	AIME25		HMMT25	
	Acc.	Tok.	Acc.	Tok.
Expansion policy				
Self-consistency	86.7	1.00×	66.7	1.00×
SingleToken-only	93.3	1.81×	70.0	1.77×
MultiToken-only	90.0	1.87×	66.7	1.82×
Manual schedule	93.3	1.78×	**76.7**	1.87×
HyPER	**96.7**	1.62×	**76.7**	1.54×
Voting rule with the same HyPER trajectories				
Majority voting	86.7	–	73.3	–
Confidence-weighted voting	90.0	–	73.3	–
Length-aware conf. voting	**96.7**	–	**76.7**	–

5. Conclusion

We introduce HyPER, a training-free online control policy that unifies adaptive exploration and exploitation for scalable reasoning. By leveraging phase-dependent dynamics and MoE routing diversity, HyPER consistently achieves better accuracy–compute trade-offs than static baselines. We further bridge the existence–selection gap by identifying path length as a robust correctness signal under confidence pruning. While our token-level primitive leverages MoE routing, our control policy generalizes to dense models, as shown in preliminary experiments. Crucially, HyPER improves hypothesis generation and is orthogonal to learned verifiers; combining its dynamic policy with Process Reward Models (PRMs) remains a promising avenue for future test-time scaling.

References

An, C., Xie, Z., Li, X., Li, L., Zhang, J., Gong, S., Zhong, M., Xu, J., Qiu, X., Wang, M., and Kong, L. Polaris: A post-training recipe for scaling reinforcement learning on advanced reasoning models, 2025. URL https://hkunlp.github.io/blog/2025/Polaris. https://hkunlp.github.io/blog/2025/Polaris.

Anthropic. Introducing claude sonnet 4.5, September 2025. URL https://www.anthropic.com/news/claude-sonnet-4-5. https://www.anthropic.com/news/claude-sonnet-4-5.

Besta, M., Blach, N., Kubicek, A., Gerstenberger, R., Podstawski, M., Gianinazzi, L., Gajda, J., Lehmann, T., Niewiadomski, H., Nyczyk, P., and Hoefler, T. Graph of thoughts: Solving elaborate problems with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690, March 2024. ISSN 2159-5399. doi: 10.1609/aaai.v38i16.29720. URL http://dx.doi.org/10.1609/aaai.v38i16.29720.

Bi, Z., Han, K., Liu, C., Tang, Y., and Wang, Y. Forest-of-thought: Scaling test-time compute for enhancing llm reasoning, 2024. URL https://arxiv.org/abs/2412.09078.

Brown, B., Juravsky, J., Ehrlich, R., Clark, R., Le, Q. V., Ré, C., and Mirhoseini, A. Large language monkeys: Scaling inference compute with repeated sampling, 2024. URL https://arxiv.org/abs/2407.21787.

Chen, L., Davis, J. Q., Hanin, B., Bailis, P., Stoica, I., Zaharia, M., and Zou, J. Are more llm calls all you need? towards scaling laws of compound inference systems, 2024. URL https://arxiv.org/abs/2403.02419.

Chen, X., Aksitov, R., Alon, U., Ren, J., Xiao, K., Yin, P., Prakash, S., Sutton, C., Wang, X., and Zhou, D. Universal self-consistency for large language model generation, 2023. URL https://arxiv.org/abs/2311.17311.

Chen, Y., Chen, J., Meng, R., Yin, J., Li, N., Fan, C., Wang, C., Pfister, T., and Yoon, J. Tumix: Multi-agent test-time scaling with tool-use mixture, 2025. URL https://arxiv.org/abs/2510.01279.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Cordero-Encinar, P. and Duncan, A. B. Certified self-consistency: Statistical guarantees and test-time training for reliable reasoning in llms, 2025. URL https://arxiv.org/abs/2510.17472.

DeepSeek-AI. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 2024. URL https://arxiv.org/abs/2405.04434.

Ding, Y., Jiang, W., Liu, S., Jing, Y., Guo, J., Wang, Y., Zhang, J., Wang, Z., Liu, Z., Du, B., Liu, X., and Tao, D. Dynamic parallel tree search for efficient llm reasoning, 2025. URL https://arxiv.org/abs/2502.16235.

Fu, Y., Chen, J., Zhu, S., Fu, Z., Dai, Z., Zhuang, Y., Ma, Y., Qiao, A., Rosing, T., Stoica, I., and Zhang, H. Efficiently scaling llm reasoning with certaindex, 2024. URL https://arxiv.org/abs/2412.20993.

Fu, Y., Wang, X., Tian, Y., and Zhao, J. Deep think with confidence, 2025. URL https://arxiv.org/abs/2508.15260.

Guo, D., Yang, D., Zhang, H., Song, J., Wang, P., Zhu, Q., Xu, R., Zhang, R., Ma, S., Bi, X., et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL https://doi.org/10.1038/s41586-025-09422-z.

Hassid, M., Synnaeve, G., Adi, Y., and Schwartz, R. Don't overthink it. preferring shorter thinking chains for improved llm reasoning, 2025. URL https://arxiv.org/abs/2505.17813.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset, 2021. URL https://arxiv.org/abs/2103.03874.

HMMT Organization. Harvard–mit mathematics tournament (hmmt), february 2025. https://www.hmmt.org, 2025. Accessed: 2025.

Hooper, C., Kim, S., Moon, S., Dilmen, K., Maheswaran, M., Lee, N., Mahoney, M. W., Shao, S., Keutzer, K., and Gholami, A. Ets: Efficient tree search for inference-time scaling, 2025. URL https://arxiv.org/abs/2502.13575.

Irvine, R., Boubert, D., Raina, V., Liusie, A., Zhu, Z., Mudupalli, V., Korshuk, A., Liu, Z., Cremer, F., Assassi, V., Beauchamp, C.-C., Lu, X., Rialan, T., and Beauchamp, W. Rewarding chatbots for real-world engagement with millions of users, 2023. URL https://arxiv.org/abs/2303.06135.

Ji, S., Wang, Y., Liu, Y., Zhu, Q., and Che, W. Seer self-consistency: Advance budget estimation for adaptive test-time scaling, 2025. URL https://arxiv.org/abs/2511.09345.

Kang, Z., Zhao, X., and Song, D. Scalable best-of-n selection for large language models via self-certainty, 2025. URL https://arxiv.org/abs/2502.18581.

Kimi, T., Du, A., Gao, B., Xing, B., Jiang, C., Chen, C., Li, C., Xiao, C., Du, C., Liao, C., Tang, C., Wang, C., Zhang, D., Yuan, E., Lu, E., Tang, F., Sung, F., Wei, G., Lai, G., Guo, H., Zhu, H., Ding, H., Hu, H., Yang, H., Zhang, H., Yao, H., Zhao, H., Lu, H., Li, H., Yu, H., Gao, H., Zheng, H., Yuan, H., Chen, J., Guo, J., Su, J., Wang, J., Zhao, J., Zhang, J., Liu, J., Yan, J., Wu, J., Shi, L., Ye, L., Yu, L., Dong, M., Zhang, N., Ma, N., Pan, Q., Gong, Q., Liu, S., Ma, S., Wei, S., Cao, S., Huang, S., Jiang, T., Gao, W., Xiong, W., He, W., Huang, W., Xu, W., Wu, W., He, W., Wei, X., Jia, X., Wu, X., Xu, X., Zu, X., Zhou, X., Pan, X., Charles, Y., Li, Y., Hu, Y., Liu, Y., Chen, Y., Wang, Y., Liu, Y., Qin, Y., Liu, Y., Yang, Y., Bao, Y., Du, Y., Wu, Y., Wang, Y., Zhou, Z., Wang, Z., Li, Z., Zhu, Z., Zhang, Z., Wang, Z., Yang, Z., Huang, Z., Huang, Z., Xu, Z., Yang, Z., and Lin, Z. Kimi k1.5: Scaling reinforcement learning with llms, 2025. URL https://arxiv.org/abs/2501.12599.

Kuhn, L., Gal, Y., and Farquhar, S. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation, 2023. URL https://arxiv.org/abs/2302.09664.

Li, Y., Yuan, P., Feng, S., Pan, B., Wang, X., Sun, B., Wang, H., and Li, K. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning, 2024. URL https://arxiv.org/abs/2401.10480.

Lian, L., Wang, S., Juefei-Xu, F., Fu, T.-J., Li, X., Yala, A., Darrell, T., Suhr, A., Tian, Y., and Lin, X. V. Threadweaver: Adaptive threading for efficient parallel reasoning in language models, 2025. URL https://arxiv.org/abs/2512.07843.

Liao, B., Xu, Y., Dong, H., Li, J., Monz, C., Savarese, S., Sahoo, D., and Xiong, C. Reward-guided speculative decoding for efficient llm reasoning, 2025. URL https://arxiv.org/abs/2501.19324.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Lin, W., Li, X., Yang, Z., Fu, X., Zhen, H.-L., Wang, Y., Yu, X., Liu, W., Li, X., and Yuan, M. Trimr: Verifier-based training-free thinking compression for efficient test-time scaling, 2025. URL https://arxiv.org/abs/2505.17155.

Liu, R., Gao, J., Zhao, J., Zhang, K., Li, X., Qi, B., Ouyang, W., and Zhou, B. Can 1b llm surpass 405b llm? rethinking compute-optimal test-time scaling, 2025. URL https://arxiv.org/abs/2502.06703.

Lu, J., Yu, H., Xu, S., Ran, S., Tang, G., Wang, S., Shan, B., Fu, T., Feng, H., Tang, J., Wang, H., and Huang, C. Prolonged reasoning is not all you need: Certainty-based adaptive routing for efficient llm/mllm reasoning, 2025. URL https://arxiv.org/abs/2505.15154.

Luo, F., Chuang, Y.-N., Wang, G., Le, H. A. D., Zhong, S., Liu, H., Yuan, J., Sui, Y., Braverman, V., Chaudhary, V., and Hu, X. Autol2s: Auto long-short reasoning for efficient large language models, 2026. URL https://arxiv.org/abs/2505.22662.

Mathematical Association of America. American invitational mathematics examination (aime) 2024. https://www.maa.org/math-competitions/aime, 2024. Accessed: 2024.

Mathematical Association of America. American invitational mathematics examination (aime) 2025. https://www.maa.org/math-competitions/aime, 2025. Accessed: 2025.

Muennighoff, N., Soldaini, L., Groeneveld, D., Lo, K., Morrison, J., Min, S., Shi, W., Walsh, P., Tafjord, O., Lambert, N., Gu, Y., Arora, S., Bhagia, A., Schwenk, D., Wadden, D., Wettig, A., Hui, B., Dettmers, T., Kiela, D., Farhadi, A., Smith, N. A., Koh, P. W., Singh, A., and Hajishirzi, H. Olmoe: Open mixture-of-experts language models, 2024. URL https://arxiv.org/abs/2409.02060.

Muennighoff, N., Yang, Z., Shi, W., Li, X. L., Fei-Fei, L., Hajishirzi, H., Zettlemoyer, L., Liang, P., Candès, E., and Hashimoto, T. s1: Simple test-time scaling, 2025. URL https://arxiv.org/abs/2501.19393.

Ning, X., Lin, Z., Zhou, Z., Wang, Z., Yang, H., and Wang, Y. Skeleton-of-thought: Prompting llms for efficient parallel generation, 2023. URL https://arxiv.org/abs/2307.15337.

OpenAI, Jaech, A., Kalai, A., Lerer, A., Richardson, A., El-Kishky, A., Low, A., Helyar, A., Madry, A., Beutel, A., et al. Openai o1 system card, 2024. URL https://arxiv.org/abs/2412.16720.

Phan, L., Gatti, A., Han, Z., Li, N., Hu, J., Zhang, H., Zhang, C. B. C., Shaaban, M., Ling, J., Shi, S., and et al. Humanity's last exam. 2025. URL https://arxiv.org/abs/2501.14249.

Qiu, J., Sun, X., Sabet, A. H. N., and Zhao, Z. Scalable fsm parallelization via path fusion and higher-order speculation. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 887–901, 2021.

Qiu, J., Lu, Y., Zeng, Y., Guo, J., Geng, J., Zhu, C., Juan, X., Yang, L., Wang, H., Huang, K., Wu, Y., and Wang, M. Treebon: Enhancing inference-time alignment with speculative tree-search and best-of-n sampling, 2024. URL https://arxiv.org/abs/2410.16033.

Qwen, :, Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Tang, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., and Qiu, Z. Qwen2.5 technical report, 2024. URL https://arxiv.org/abs/2412.15115.

Shi, J., Zhu, Y., Shi, Z., Zhao, D., Li, Q., and Jiang, Y. Speccot: Accelerating chain-of-thought reasoning through speculative exploration. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 24405–24415, 2025.

Snell, C., Lee, J., Xu, K., and Kumar, A. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

Taubenfeld, A., Sheffer, T., Ofek, E., Feder, A., Goldstein, A., Gekhman, Z., and Yona, G. Confidence improves self-consistency in llms. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20090–20111. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.1030. URL http://dx.doi.org/10.18653/v1/2025.findings-acl.1030.

Wang, J., Fang, M., Wan, Z., Wen, M., Zhu, J., Liu, A., Gong, Z., Song, Y., Chen, L., Ni, L. M., Yang, L., Wen, Y., and Zhang, W. Openr: An open source framework for advanced reasoning with large language models, 2024. URL https://arxiv.org/abs/2410.09671.

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models, 2022. URL https://arxiv.org/abs/2203.11171.

Wang, X., Huang, Y., Wang, Y., Luo, X., Guo, K., Zhou, Y., and Zhang, X. Adareasoner: Adaptive reasoning enables more flexible thinking, 2025a. URL https://arxiv.org/abs/2505.17312.

Wang, X., Li, Y., Feng, S., Yuan, P., Zhang, Y., Shi, J., Tan, C., Pan, B., Hu, Y., and Li, K. Every rollout counts: Optimal resource allocation for efficient test-time scaling. *arXiv preprint arXiv:2506.15707*, 2025b.

Wang, Y., Luo, H., Yao, H., Huang, T., He, H., Liu, R., Tan, N., Huang, J., Cao, X., Tao, D., and Shen, L. R1-compress: Long chain-of-thought compression via chunk compression and search, 2025c. URL https://arxiv.org/abs/2505.16838.

Wang, Y., Wang, J., Chen, R., and Wei, X.-Y. Optscale: Probabilistic optimality for inference-time scaling, 2025d. URL https://arxiv.org/abs/2506.22376.

Wang, Y., Zhang, Y., Yu, T., Xu, C., Zhang, F., and Lian, F. Adaptive deep reasoning: Triggering deep thinking when needed, 2025e. URL https://arxiv.org/abs/2505.20101.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Welleck, S., Bertsch, A., Finlayson, M., Schoelkopf, H., Xie, A., Neubig, G., Kulikov, I., and Harchaoui, Z. From decoding to meta-generation: Inference-time algorithms for large language models, 2024. URL https://arxiv.org/abs/2406.16838.

Wu, Y., Sun, Z., Li, S., Welleck, S., and Yang, Y. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models, 2024. URL https://arxiv.org/abs/2408.00724.

xAI. Grok 4. https://x.ai/news/grok-4, 2025.

Xia, Z., Sun, H., Wang, J., Qi, Q., Wang, H., Fu, X., and Liao, J. The threat of PROMPTS in large language models: A system and user prompt perspective. In Che, W., Nabende, J., Shutova, E., and Pilehvar, M. T. (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 12994–13035, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN

979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.675. URL https://aclanthology.org/2025.findings-acl.675/.

Xie, Y., Goyal, A., Zheng, W., Kan, M.-Y., Lillicrap, T. P., Kawaguchi, K., and Shieh, M. Monte carlo tree search boosts reasoning via iterative preference learning, 2024. URL https://arxiv.org/abs/2405.00451.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Zhou, J., Lin, J., Dang, K., Bao, K., Yang, K., Yu, L., Deng, L., Li, M., Xue, M., Li, M., Zhang, P., Wang, P., Zhu, Q., Men, R., Gao, R., Liu, S., Luo, S., Li, T., Tang, T., Yin, W., Ren, X., Wang, X., Zhang, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Zhang, Y., Wan, Y., Liu, Y., Wang, Z., Cui, Z., Zhang, Z., Zhou, Z., and Qiu, Z. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., and Narasimhan, K. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Ye, Y., Huang, Z., Xiao, Y., Chern, E., Xia, S., and Liu, P. Limo: Less is more for reasoning, 2025. URL https://arxiv.org/abs/2502.03387.

Yu, Z., Xu, T., Jin, D., Sankararaman, K. A., He, Y., Zhou, W., Zeng, Z., Helenowski, E., Zhu, C., Wang, S., Ma, H., and Fang, H. Think smarter not harder: Adaptive reasoning with inference aware optimization, 2025. URL https://arxiv.org/abs/2501.17974.

Zhang, Q., Lyu, F., Sun, Z., Wang, L., Zhang, W., Hua, W., Wu, H., Guo, Z., Wang, Y., Muennighoff, N., King, I., Liu, X., and Ma, C. A survey on test-time scaling in large language models: What, how, where, and how well?, 2025. URL https://arxiv.org/abs/2503.24235.

Zhao, C., Tan, Z., Ma, P., Li, D., Jiang, B., Wang, Y., Yang, Y., and Liu, H. Is chain-of-thought reasoning of llms a mirage? a data distribution lens, 2026. URL https://arxiv.org/abs/2508.01191.

Zheng, X., Huang, Z., Chiang, M.-F., Witbrock, M. J., and Zhao, K. Kcr: Resolving long-context knowledge conflicts via reasoning in llms, 2025. URL https://arxiv.org/abs/2508.01273.

Zhu, J., Huang, Y., Shen, Y., Zhao, J., and Zou, A. Path-consistency with prefix enhancement for efficient inference in llms, 2024. URL https://arxiv.org/abs/2409.01281.

Zibakhsh, S., Samragh, M., Nishu, K., Hannah, L., Kundu, A., and Cho, M. Moes are stronger than you think: Hyper-parallel inference scaling with roe, 2025. URL https://arxiv.org/abs/2509.17238.

Table 4. Taxonomy of representative test-time scaling methods. Adap ctrl indicates online expansion/refinement decisions; answer eval indicates final answer aggregation or selection over multiple paths.

Method	Gran.	Exploration		Exploitation			Train-free	
		Adap ctrl	No rigid search	Answer eval	Prune	Token refine	Policy model	Reward model
Tree-based search schemes								
ToT (Yao et al., 2023)	step	✗	✗	✓	✗	✗	✓	✗
MCTS (Xie et al., 2024)	step	✓	✗	✓	✗	✗	✓	✗
ETS (Hooper et al., 2025)	step	✓	✗	✓	✓	✗	✗	✗
Parallel reasoning								
SC (Wang et al., 2022)	path	✗	✓	✓	✗	✗	✓	✗
BoN (Brown et al., 2024)	path	✗	✓	✓	✗	✗	✓	✗
Adaptive reasoning								
DeepConf (Fu et al., 2025)	path	✓	✓	✓	✓	✗	✓	✓
Thread (Lian et al., 2025)	step	✓	✓	✓	✗	✗	✗	✓
Token-level refinement								
RoE (Zibakhsh et al., 2025)	token	✗	✓	✗	✗	✓	✓	✓
Expand–reduce control								
HyPER (ours)	token/path	✓	✓	✓	✓	✓	✓	✓

A. Taxonomy of Test-Time Scaling Methods

Table 4 summarizes representative test-time scaling methods by their decision granularity and how they instantiate exploration, exploitation, and supervision. HyPER differs from prior methods by jointly supporting online adaptive control, non-rigid multi-path generation, confidence-based pruning, token-level refinement, and answer-time aggregation without additional training.

B. Controller Scoring Rules

At each decision point, HyPER maps each statistic to $[0, 1]$ using $\widehat{X}_t = \min(X_t / X_{\max}, 1)$, where $X_{\max}$ is estimated during warm-up. The controller then computes one score per action and selects the action with the highest score:

$$\text{Score}_{\text{NONE}}(t) = \frac{1}{3}\left(\widehat{\bar{C}}_t + (1 - \widehat{H}_t) + \widehat{D}_t\right), \tag{3}$$

$$\text{Score}_{\text{SINGLETOKEN}}(t) = \frac{1}{3}\left((1 - \widehat{\bar{C}}_t) + \widehat{H}_t + (1 - \beta_t)\right), \tag{4}$$

$$\text{Score}_{\text{BRANCH}}(t) = \frac{1}{2}\left((1 - \widehat{D}_t) + (1 - \beta_t)\right), \tag{5}$$

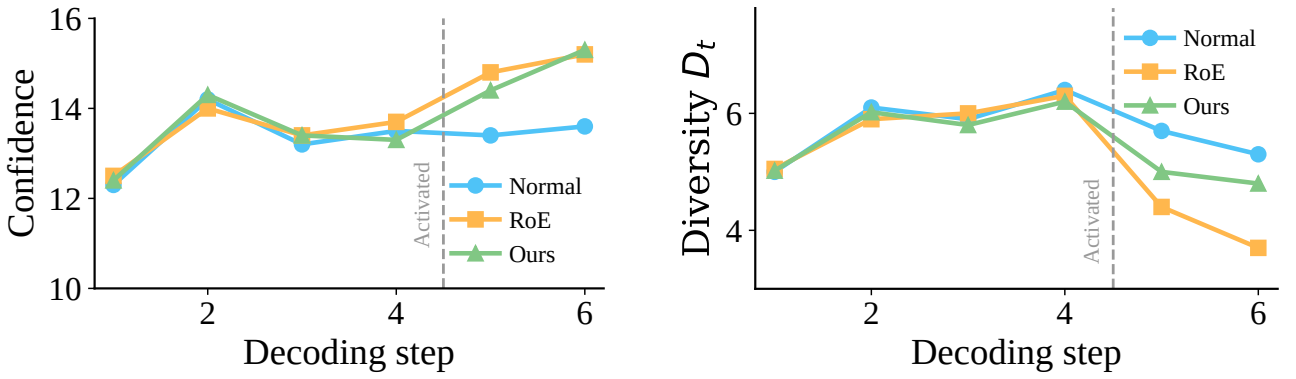
$$\text{Score}_{\text{MULTITOKEN}}(t) = \frac{1}{3}\left((1 - \widehat{D}_t) + (1 - \widehat{\bar{C}}_t) + \widehat{\text{Var}(C)}_t\right). \tag{6}$$

The coefficients are fixed normalization factors rather than tuned hyperparameters. High confidence, low entropy, and high diversity favor NONE; low confidence or high entropy favors SINGLETOKEN; diversity collapse and low consensus favor BRANCH; and low diversity with confidence dispersion favors MULTITOKEN. Although diversity computation involves pairwise comparisons, the path pool is capped by $S_{\max}$ and the controller is invoked only every T steps, so the amortized overhead is small in practice.

B.1. Two-Pass Sampling versus RoE

Figure 7 compares our SingleToken two-pass refinement against a RoE-style uniform token-level refinement baseline. Panel (a) shows that both methods substantially improve tail-token confidence over no refinement, indicating that token-level logit aggregation is an effective exploitation primitive and that SingleToken retains the token-quality gains of hyper-parallel routing. Panel (b) further shows that SingleToken consistently yields higher path diversity D_t, mitigating the rapid diversity collapse observed under the RoE baseline.

While our diversity-aware penalty is applied *within* each path (encouraging different expert sets across the K intra-token routes), it can still increase *inter-path* diversity measured by D_t: by broadening the set of plausible next-token logits considered per token, the refinement step becomes less likely to deterministically steer different sampled prefixes into the same continuation, thereby preserving (and sometimes amplifying) trajectory-level disagreements across paths without



(a) Both RoE and our aggregation method raise tail confidence.

(b) We maintain relatively higher diversity than RoE.

Figure 7. Comparison between our method and RoE.

directly coupling routes across paths.

C. Definitions of online metrics

Per-path confidence and top-k entropy. Let $P_{p,t}(\cdot)$ denote the next-token distribution at step t on path p, and let $y_t^{(p)}$ be the token actually selected. We define token confidence following the DeepConf approach. Crucially, this metric focuses on the **alternative candidates** (competitors) to the selected token. Let $\mathcal{C}_k(p,t)$ be the set of the top-k tokens excluding the selected one (or simply the top-k candidates if $y_t^{(p)}$ is dominant). The confidence is calculated as the negative average log-probability of these candidates:

$$c_{p,t} = -\frac{1}{k} \sum_{v \in \mathcal{C}_k(p,t)} \log P_{p,t}(v). \tag{7}$$

This formulation captures the "impossibility" of alternative paths. When the model is highly confident in $y_t^{(p)}$, the probabilities of competing tokens $P_{p,t}(v)$ approach zero, causing their negative log-probabilities to become large positive values. Thus, a **higher** $c_{p,t}$ indicates that the model considers all alternatives to be highly unlikely.

For entropy, we use a top-k approximation over the full candidate set $\mathcal{V}_k(p,t)$ (including the selected token). Let $\tilde{P}_{p,t}(v) = \frac{P_{p,t}(v)}{\sum_{u \in \mathcal{V}_k(p,t)} P_{p,t}(u)}$ be the renormalized distribution. Then the token-level entropy is:

$$h_{p,t} = -\sum_{v \in \mathcal{V}_k(p,t)} \tilde{P}_{p,t}(v) \log \tilde{P}_{p,t}(v). \tag{8}$$

Unlike confidence, a *lower* entropy $h_{p,t}$ signifies higher certainty.

Global confidence statistics. From these path-wise quantities we define the global metrics:

$$\bar{C}_t = \frac{1}{S_t} \sum_{p \in \mathcal{S}_t} c_{p,t}, \tag{9}$$

$$H_t = \frac{1}{S_t} \sum_{p \in \mathcal{S}_t} h_{p,t}, \tag{10}$$

$$\beta_t = \frac{1}{S_t} \max_v \left| \left\{ p \in \mathcal{S}_t : \; y_t^{(p)} = v \right\} \right|. \tag{11}$$

Distribution-level diversity via Jensen–Shannon divergence. We measure local disagreement between predictive distributions using the Jensen–Shannon divergence (JSD). For two distributions P and Q over the same support, define

$$\mathrm{JSD}(P\|Q) = \frac{1}{2}\mathrm{KL}(P\|M) + \frac{1}{2}\mathrm{KL}(Q\|M)\,, \tag{12}$$

$$M = \frac{1}{2}(P+Q), \qquad \mathrm{KL}(P\|Q) = \sum_v P(v)\log\frac{P(v)}{Q(v)}. \tag{13}$$

Using the (optionally top-k renormalized) next-token distributions $\tilde{P}_{p,t}$, we define

$$D_{\mathrm{dist},t} = \frac{2}{S_t(S_t-1)} \sum_{p<q,\ p,q\in\mathcal{S}_t} \mathrm{JSD}\left(\tilde{P}_{p,t} \,\middle\|\, \tilde{P}_{q,t}\right). \tag{14}$$

Suffix-level diversity via normalized edit distance. Let $\mathrm{suf}_L(p,t)$ be the length-L suffix token sequence of path p up to step t (or the entire suffix since the last decision step if shorter). Let $\mathrm{Lev}(\cdot,\cdot)$ denote Levenshtein edit distance. We use a length-normalized edit distance:

$$D_{\mathrm{seq},t} = \frac{2}{S_t(S_t-1)} \sum_{p<q,\ p,q\in\mathcal{S}_t} \frac{\mathrm{Lev}(\mathrm{suf}_L(p,t), \mathrm{suf}_L(q,t))}{\max\left(|\mathrm{suf}_L(p,t)|, |\mathrm{suf}_L(q,t)|\right)}. \tag{15}$$

Composite diversity and confidence dispersion. We combine both notions of diversity as

$$D_t = \eta\, D_{\mathrm{dist},t} + (1-\eta)\, D_{\mathrm{seq},t}, \tag{16}$$

and we measure confidence dispersion across paths by

$$\mathrm{Var}(C)_t = \frac{1}{S_t} \sum_{p\in\mathcal{S}_t} \left(c_{p,t} - \bar{C}_t\right)^2. \tag{17}$$

Global-max normalization (used by the controller). To map each metric onto a comparable scale, we normalize by a global maximum estimated during warm-up:

$$X_t^{\mathrm{norm}} = \min\left(\frac{X_t}{X_{\max}}, 1\right), \tag{18}$$

where $X_{\max}$ is the maximum observed value of X_t in warm-up. We apply Eq. (18) to $\bar{C}_t$, H_t, D_t, and $\mathrm{Var}(C)_t$.

D. Full HyPER Decoding Pipeline

The detailed decoding pipeline of HyPER is described in Algorithm 1.

E. The pseudo-code of Two-Pass Expert Sampling with Diversity Penalty

The detailed sampling strategy is described in Algorithm 2.

F. KV caching for Single-Token Aggregation

Single-token aggregation expands each surviving hypothesis path by sampling K parallel expert-routed proposals for the *current* token. A naive implementation would maintain K independent KV caches (one per routed proposal), leading to $O(K)$ growth in both memory and compute. Instead, we adopt RoE's *Clean Cache* strategy (Zibakhsh et al., 2025), which localizes stochasticity to the current token and shares history states across proposals.

Clean Cache. For each surviving path, we construct a batched forward pass over the K routed proposals for the next-token computation, but we apply stochastic routing *only* at the current token. Concretely, we include a deterministic "clean" proposal (batch index 0) by setting the routing temperature for that proposal to zero, and we reuse its KV cache as the shared history cache for all other proposals in the batch. This yields sufficient diversity from per-token routing perturbations while ensuring that the KV-cache *memory footprint does not scale with* K; it matches the memory usage of standard decoding for the same hypothesis path history, with additional overhead confined to the batched forward computation of the current token.

Algorithm 1 HyPER decoding pipeline

Require: Prompt x, LLM f_θ, interval T, budgets $T_{\max}, S_{\max}$
Ensure: Final answer $\hat{a}$
Warm-up: run SC-style decoding to obtain an initial path pool $\mathcal{S}_0$ and a pruning threshold τ
for $t = 1, 2, \ldots, T_{\max}$ **do**
  if $\mathcal{S}_t = \emptyset$ **then break**
  end if
  if $t \bmod T = 1$ **then**
    Compute online pool statistics from $\mathcal{S}_t$:
    confidence/uncertainty $(\bar{C}_t, H_t, \beta_t)$ and diversity D_t
    (optionally) dispersion $\mathrm{Var}(C)_t$ and pool size $|\mathcal{S}_t|$
    Select an action $a_t \in \{\textsc{None}, \textsc{SingleToken}, \textsc{MultiToken}, \textsc{Branch}\}$:
    $a_t \leftarrow \arg\max_a \mathrm{Score}(a; x, \mathcal{S}_t)$ ▷ cf. Sec. 3 / App. C
  end if
  Apply action a_t **to update the pool** $\mathcal{S}_t$**:**
  if $a_t = \textsc{None}$ **then**
    Decode one token for each $p \in \mathcal{S}_t$ with standard routing
  else if $a_t = \textsc{SingleToken}$ **then**
    Decode one token for each $p \in \mathcal{S}_t$ using single-token aggregation
  else if $a_t = \textsc{Branch}$ **then**
    Expand the pool by duplicating selected paths and decoding them independently
  else if $a_t = \textsc{MultiToken}$ **then**
    Perform short-horizon local expansion for m steps and merge back into the pool
  end if
  Update sliding-window confidences; prune paths with $c_{p,t} < \tau$ or EOS/max-length
end for
Collect completed paths and aggregate answers via length- and confidence-aware voting to obtain $\hat{a}$
return $\hat{a}$

Overall memory scaling. Because different hypothesis paths have different prefix histories, KV caches are still maintained *per path*. Therefore, the KV-cache memory scales primarily with the number of *surviving paths* $|\mathcal{S}_t|$ (capped by $S_{\max}$), rather than with $K \cdot |\mathcal{S}_t|$. In practice, the extra cost of single-token aggregation is dominated by the batched MoE forward for the current token, while KV-cache storage remains comparable to standard multi-path decoding under the same $|\mathcal{S}_t|$ cap.

G. Additional Experimental Analysis

G.1. Hyperparameter sensitivity

We study the sensitivity of HyPER to several key hyperparameters on two challenging benchmarks, AIME25 and HMMT25, under the same main-result evaluation protocol and fixed budget. We vary one hyperparameter at a time while keeping all other settings identical to the main configuration (Section 4.1), where the top-K_a answer voting cutoff is fixed at $K_a = 3$. Figure 8 reports the accuracy trends across three representative values for each hyperparameter.

Controller decision interval T. We sweep the decision interval $T \in \{32, 64, 128\}$. Overall, accuracy exhibits mild fluctuations but remains within an acceptable range on both datasets. Smaller T reacts more frequently to transient pool statistics, while larger T updates less often and can be slower to adapt when the pool transitions between exploration- and exploitation-dominant phases. We use $T = 64$ as a stable default that balances responsiveness and signal stability.

Diversity composition weight η. We sweep the diversity composition weight $\eta \in \{0.2, 0.4, 0.8\}$ in the pool-level diversity signal. Across both datasets, performance varies moderately but remains stable around the default $\eta = 0.4$. Extremely small or large η can over-emphasize one component of the diversity signal and lead to slightly suboptimal exploration/exploitation allocation, consistent with the role of η as a trade-off knob.

Algorithm 2 Two-Pass Expert Sampling with Diversity Penalty

Require: Router logits $L \in \mathbb{R}^{B \times E}$, top-$k$ value k, penalty strength λ
Ensure: Final expert indices I_2 and weights V_2

First pass: standard noisy top-k routing
$G \leftarrow -\log\big(-\log(\text{rand\_like}(L))\big)$
$\tilde{L} \leftarrow L + G$
$P_1 \leftarrow \text{softmax}(\tilde{L})$
$(I_1, V_1) \leftarrow \text{topk}(P_1, k)$
Track expert usage as a sparse score matrix
$S \leftarrow \mathbf{0}_{E \times B}$ ▷ rows: experts, cols: routes (tokens)
for $b = 1$ to B **do**
  for $j = 1$ to k **do**
    $e \leftarrow I_1[b, j]$ ▷ the j-th selected expert for route b
    $S[e, b] \leftarrow \tilde{L}[b, e]$
  end for
end for
$U_{\text{cum}} \leftarrow \text{cumsum}(S, \text{dim} = 2)$
$U_{\text{prev}} \leftarrow \text{roll}(U_{\text{cum}}, 1, \text{dim} = 2)$
$U_{\text{prev}}[:, 0] \leftarrow 0$ ▷ no history for the first route
Second pass: apply deterministic diversity penalty and re-route
$\Delta \leftarrow \tanh\big(U_{\text{prev}}^{\top}\big)$ ▷ broadcast to shape $B \times E$
$L_2 \leftarrow \tilde{L} - \lambda \cdot \Delta$
$P_2 \leftarrow \text{softmax}(L_2)$
$(I_2, V_2) \leftarrow \text{topk}(P_2, k)$
return (I_2, V_2)

Two-pass reuse penalty strength λ. We sweep the reuse-penalty strength $\lambda \in \{0.05, 0.1, 0.2\}$ in the two-pass sampler used by SINGLETOKEN. Among the tested hyperparameters, λ can have a more visible effect: a *too-light* penalty (e.g., $\lambda = 0.05$) weakly discourages expert reuse and thus behaves closer to independent route sampling (i.e., closer in spirit to prior token-level routing perturbations), which may reduce intra-token route diversity; conversely, a *too-strong* penalty (e.g., $\lambda = 0.2$) can over-penalize frequently selected experts and distort expert choice, degrading refinement quality. We therefore use $\lambda = 0.1$ as the default, which provides consistent gains without overly constraining expert selection.

G.2. Length Bias Robustness Analysis

In this section, we analyze the robustness of the length bias phenomenon across different datasets and temperature settings. We examine how path lengths for correct and incorrect answers are distributed under various conditions. The results show that, even with different temperature settings, the length of correct paths consistently tends to be longer than incorrect ones, indicating that path length remains a strong feature for correctness, even in the presence of noise and varying conditions.

G.3. Ablation Study on Voting Strategies

To demonstrate that length is an auxiliary signal and not the sole factor in path selection, we perform an ablation study comparing three different voting strategies: length-based voting, confidence-based voting, and combined voting. The table below summarizes the accuracy of these strategies across three datasets. As shown, while length-based voting performs better than random, the combined voting strategy, which leverages both confidence and length, consistently achieves the highest accuracy, confirming that confidence-based signals play a central role.

Voting Strategy	AIME24	AIME25	HMMT25
Length-Based Voting	90%	93.3%	73.3%
Confidence-Based Voting	93.3%	90.0%	73.3%
Combined Voting (Length + Confidence)	96.7%	96.7%	76.7%

Table 5. Accuracy comparison between length-based voting, confidence-based voting, and combined voting strategies. The combined voting strategy outperforms both length-based and confidence-based voting, demonstrating that length is an auxiliary signal.

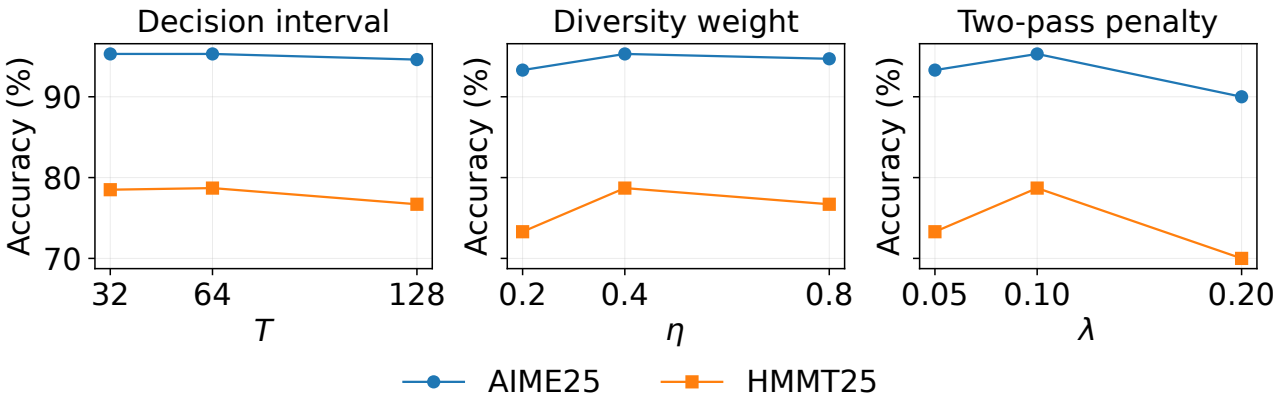


Figure 8. **Hyperparameter sensitivity** on AIME25 and HMMT25. We sweep the controller decision interval T, diversity weight η, and two-pass penalty strength λ, while fixing all other settings and the evaluation budget to the main configuration.

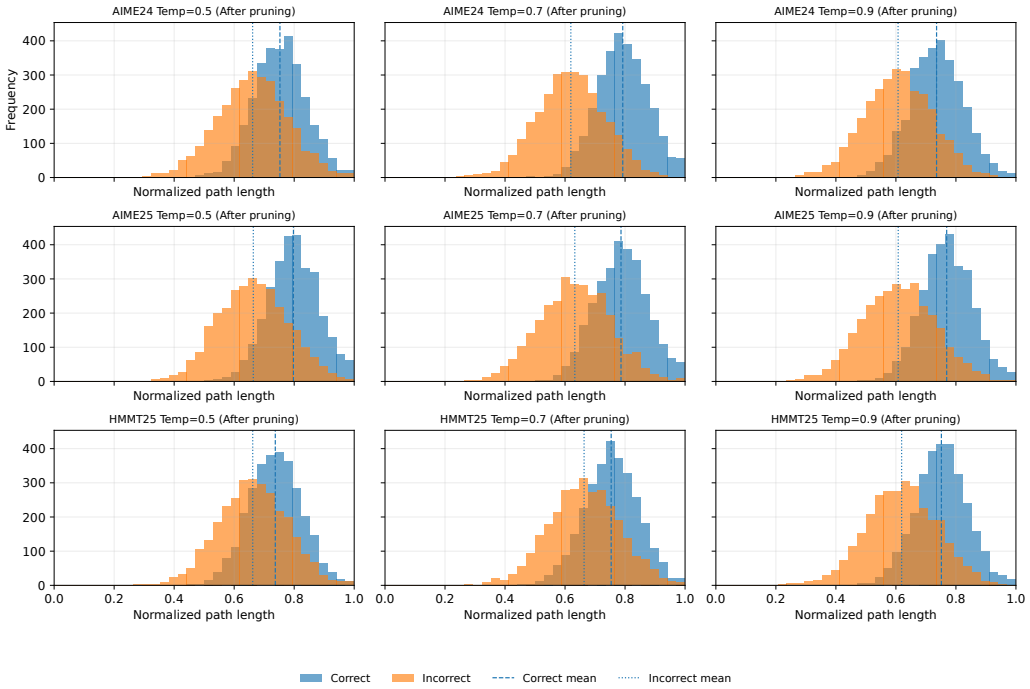


Figure 9. Robustness of length bias across three datasets (AIME24, AIME25, HMMT25) and three temperature settings (0.5, 0.7, 0.9). In all configurations, correct paths consistently have longer lengths than incorrect paths, demonstrating the robustness of the length signal in guiding correct answer selection.

G.4. Analysis of "Long but Incorrect" Paths

A common concern with length-based signals is the possibility of selecting long but incorrect paths. To address this, we provide a simple analysis of how our method handles such cases. When faced with a long but incorrect path, our confidence-based aggregation mechanism mitigates the potential risk by assigning lower confidence to incorrect paths, even if they are long. Additionally, the pruning mechanism ensures that only paths with high confidence are retained, further reducing the likelihood of selecting long but incorrect paths.

In our experiments, we observed that the number of long-but-incorrect paths was minimal. The pruning threshold, guided by confidence, effectively eliminated most of these paths, ensuring that the final answer was selected from a set of high-confidence paths. Moreover, in cases where long paths were selected but incorrect, confidence-based voting was able to correctly identify and discard them in favor of shorter, correct alternatives.

This behavior demonstrates that our method does not solely rely on path length, but combines length and confidence to make more robust predictions, as evidenced by the consistent performance of the combined voting strategy.

H. Importance-Sampling View of Length–Confidence Voting

Goal: answer marginalization under a biased decoding proposal. Let x be the input problem and let r denote a full reasoning trajectory (a completed CoT path). Each trajectory deterministically induces a scalar answer via an extraction map $a(r) \in \mathcal{A}$. Our evaluation target is the answer marginal under an (implicit) target path distribution $P^\star(r \mid x)$:

$$P^\star(a \mid x) = \sum_{r:\, a(r)=a} P^\star(r \mid x). \tag{19}$$

Hence selecting the most probable answer under the marginal objective is

$$a^\star(x) = \arg\max_{a \in \mathcal{A}} P^\star(a \mid x) = \arg\max_{a \in \mathcal{A}} \sum_{r:\, a(r)=a} P^\star(r \mid x). \tag{20}$$

In practice, we do not sample trajectories from $P^\star(r \mid x)$. Instead, the multi-path decoding pipeline (sampling temperature/top-p, pruning, branching, etc.) induces a *proposal* distribution over completed trajectories, denoted by $q(r \mid x)$. We observe N sampled trajectories $r_1, \ldots, r_N \sim q(\cdot \mid x)$, with extracted answers $a_i \triangleq a(r_i)$. For each sampled path we also compute observable path features: length L_i and a global confidence statistic $\hat{c}_i$ from the pruning backbone.

Importance sampling identity. Define the unnormalized target *answer mass*

$$Z(a) \triangleq \sum_{r:\, a(r)=a} P^\star(r \mid x), \tag{21}$$

so that $P^\star(a \mid x) = Z(a)/\sum_{a'} Z(a')$ and maximizing $P^\star(a \mid x)$ is equivalent to maximizing $Z(a)$. For any answer a, by the standard change-of-measure,

$$\begin{aligned} Z(a) &= \sum_r \mathbf{1}[a(r) = a]\; P^\star(r \mid x) = \sum_r q(r \mid x)\; \mathbf{1}[a(r) = a]\; \underbrace{\frac{P^\star(r \mid x)}{q(r \mid x)}}_{w(r)} \\ &= \mathbb{E}_{r \sim q(\cdot \mid x)}\big[\mathbf{1}[a(r) = a] \cdot w(r)\big], \end{aligned} \tag{22}$$

where $w(r) = P^\star(r \mid x)/q(r \mid x)$ is the (generally intractable) density ratio. Thus, if $w(r)$ were available, a Monte Carlo estimator of $Z(a)$ would be

$$\widehat{Z}_{\mathrm{IS}}(a) = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}[a_i = a]\; w(r_i), \qquad \widehat{a}_{\mathrm{IS}} = \arg\max_{a \in \mathcal{A}} \widehat{Z}_{\mathrm{IS}}(a). \tag{23}$$

Equivalently, one may estimate the normalized marginal via *self-normalized importance sampling (SNIS)*:

$$\widehat{P}_{\mathrm{SNIS}}(a \mid x) = \sum_{i=1}^{N} \bar{w}_i\; \mathbf{1}[a_i = a], \qquad \bar{w}_i = \frac{w(r_i)}{\sum_{j=1}^{N} w(r_j)}. \tag{24}$$

Both (23) and (24) implement the marginalization objective (19) under a biased proposal q.

Proxy density-ratio modeling with low-variance weighting. The exact ratio $w(r)$ is unavailable for our decoding pipeline. We therefore approximate the ratio by a *proxy* weight that depends on stable, path-level observables. Let $\phi(r) \in \mathbb{R}^d$ be a feature map; in HyPER we use two normalized features derived from $(L_i, \hat{c}_i)$:

$$\tilde{L}_i \triangleq \frac{L_i}{\sum_{j=1}^{N} L_j}, \qquad \tilde{C}_i \triangleq \frac{\hat{c}_i}{\max_{j \in [N]} \hat{c}_j}, \qquad \phi(r_i) \triangleq (\tilde{L}_i, \tilde{C}_i). \tag{25}$$

We adopt a log-linear proxy for the ratio,

$$w(r) \;\approx\; \tilde{w}_\theta(r) \triangleq \exp\big(\theta^\top \phi(r)\big), \qquad \theta = (\theta_L, \theta_C), \tag{26}$$

which can be interpreted as a smooth, controlled reweighting. Plugging (26) into (23) yields the proxy-IS answer mass estimator:

$$\widehat{Z}_{\text{proxy}}(a) \propto \sum_{i:\, a_i = a} \exp\big(\theta^\top \phi(r_i)\big). \tag{27}$$

Proposition H.1 (HyPER voting as linearized proxy-IS ranking)**.** *Fix a candidate answer set $\mathcal{A}' \subseteq \mathcal{A}$. Assume the proxy density ratio takes the log-linear form (26) and that the weights are* mild *in the sense that $|\theta^\top \phi(r_i)| \le \epsilon$ for all $i \in [N]$ and some small ϵ. Then, up to an additive answer-independent constant and an $O(\epsilon^2)$ remainder, maximizing the proxy-IS mass estimator (27) over $\mathcal{A}'$ is equivalent to maximizing the linear score*

$$\text{score}(a) = \sum_{i:\, a_i = a} \theta^\top \phi(r_i) = \sum_{i:\, a_i = a} \Big(\lambda_{\text{len}} \tilde{L}_i + \lambda_{\text{conf}} \tilde{C}_i\Big), \qquad a \in \mathcal{A}', \tag{28}$$

where $(\lambda_{\text{len}}, \lambda_{\text{conf}})$ reparameterize θ after feature rescaling.

Proof. For each sample i, apply a first-order Taylor expansion:

$$\exp(\theta^\top \phi(r_i)) = 1 + \theta^\top \phi(r_i) + O(\epsilon^2). \tag{29}$$

Substituting into (27) yields, for any a,

$$\widehat{Z}_{\text{proxy}}(a) \propto \sum_{i:\, a_i = a} \big(1 + \theta^\top \phi(r_i) + O(\epsilon^2)\big) = \underbrace{\sum_{i:\, a_i = a} 1}_{\text{frequency}} + \sum_{i:\, a_i = a} \theta^\top \phi(r_i) + O\big(n_a \epsilon^2\big)\,, \tag{30}$$

where $n_a = \sum_{i=1}^N \mathbf{1}[a_i = a]$ is the number of supporting paths for answer a. When comparing answers within a fixed candidate set $\mathcal{A}'$, the constant offset induced by the 1 term can be (i) used only for candidate formation or (ii) absorbed into the comparison as an additive term, while the $O(n_a \epsilon^2)$ remainder is uniformly small under mild weights. Dropping answer-independent constants yields the linear ranking rule (28). □

Top-K truncation as variance control (rare-answer instability). A practical issue in (S)NIS for discrete selection is that answers with extremely small support under the proposal yield high-variance Monte Carlo estimates: for rare events, $\mathbf{1}[a(r) = a]$ is almost always zero, so $\widehat{Z}(a)$ is unstable under finite N. The following elementary calculation formalizes this intuition in the simplest (uniform-weight) case.

Lemma H.2 (Relative error of frequency (uniform-weight marginalization))**.** *Let $r_1, \ldots, r_N \sim q(\cdot \mid x)$ be i.i.d. samples and define $A_i \triangleq a(r_i)$. For any fixed answer a, let $p_a \triangleq \Pr_{r \sim q}[a(r) = a]$. The frequency estimator $\hat{p}_a \triangleq \frac{1}{N} \sum_{i=1}^N \mathbf{1}[A_i = a]$ satisfies*

$$\text{Var}[\hat{p}_a] = \frac{p_a(1 - p_a)}{N}, \qquad \frac{\sqrt{\text{Var}[\hat{p}_a]}}{p_a} = \sqrt{\frac{1 - p_a}{N p_a}}. \tag{31}$$

In particular, the relative standard deviation scales as $1/\sqrt{N p_a}$ and blows up when p_a is small.

Proof. $\sum_{i=1}^N \mathbf{1}[A_i = a] \sim \text{Binomial}(N, p_a)$, so $\text{Var}[\hat{p}_a] = \frac{1}{N^2} N p_a (1 - p_a) = \frac{p_a(1-p_a)}{N}$, and the relative standard deviation follows immediately. □

Two-stage truncate-then-reweight. Motivated by Lemma H.2, we use a two-stage procedure: (i) truncate to a candidate set of sufficiently supported answers under the proposal (low-variance coarse screening), then (ii) apply proxy-IS reweighting to distinguish candidates (information-rich fine selection).

Stage 1 (candidate truncation). Form a candidate answer set by majority support under the proposal samples:

$$\mathcal{A}_K = \text{TopK}_a \left(\sum_{i=1}^N \mathbf{1}[a_i = a] \right). \tag{32}$$

Stage 2 (linearized proxy-IS within candidates). Choose the final answer by the stabilized linear proxy-IS score from Proposition H.1:

$$\widehat{a} = \arg\max_{a \in \mathcal{A}_K} \sum_{i:\, a_i = a} \Big(\lambda_{\text{len}} \tilde{L}_i + \lambda_{\text{conf}} \tilde{C}_i \Big). \tag{33}$$

Corollary H.3 (Support lower bound implied by Top-K truncation)**.** *Let $\hat{p}_a \triangleq \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}[a_i = a]$ denote the empirical support of answer a and let $\hat{p}_{(K)}$ be the K-th largest value among $\{\hat{p}_a\}_{a \in \mathcal{A}}$. Then for every $a \in \mathcal{A}_K$ we have $\hat{p}_a \geq \hat{p}_{(K)}$. Consequently, the uniform-weight relative error bound in Lemma H.2 (when expressed in terms of empirical support) is uniformly controlled over $\mathcal{A}_K$ by replacing p_a with $\hat{p}_{(K)}$.*

Takeaway. Under a biased decoding proposal $q(r \mid x)$, our voting rule can be viewed as a stabilized approximation to marginal answer selection (20) via (self-normalized) importance sampling: Top-K truncation reduces rare-answer variance, and length/confidence provide a low-variance proxy correction for the intractable density ratio $P^\star(r \mid x)/q(r \mid x)$.

We do not claim this model is realistic; rather, it shows that our aggregation rule is not ad hoc, but arises naturally under minimal assumptions consistent with confidence pruning.

I. Non-MoE experiment: testing the controller without single-token aggregation

HyPER is instantiated on mixture-of-experts models through the single-token aggregation module, which refines token predictions by combining multiple routed expert proposals at each step. To show that the *controller and selection logic of HyPER are not tied to MoE routing*, we conduct an experiment on a *dense* model where no refinement is available. In this setting, we simply *remove* the single-token aggregation action from the action library and retain only (i) confidence- and diversity-aware BRANCH/MULTI-TOKEN AGGREGATION decisions and (ii) our length- and confidence-aware answer voting. Confidence pruning remains always on (as in the MoE setting) and is applied after each step. This setup isolates whether HyPER's online statistics and expand–reduce controller still provide value when token-level refinement is absent.

Setup. We use `deepseek-ai/DeepSeek-R1-0528-Qwen3-8B` on HMMT25 with a maximum of $S_{\max} = 32$ parallel paths. All decoding hyperparameters follow the MoE experiments. The SC baseline corresponds to single-path chain-of-thought decoding, and the DeepConf baseline applies confidence pruning over multiple independent paths. HyPER uses the same controller and answer voting as in the MoE case, but its action set excludes SINGLE-TOKEN AGGREGATION; the controller may only choose to widen or locally branch–merge the path pool based on online statistics, while confidence pruning remains always on and is applied after each step.

Why this test? Dense models lack intrinsic expert routing, so token-level refinement is unavailable. This experiment therefore probes the following question: *Are HyPER's statistics and controller intrinsically useful for deciding when to branch, even without any token-level refinement?* If so, this indicates that the expand–reduce control logic is architecture-agnostic and not dependent on MoE-specific structure.

Results. Table 6 shows that the HyPER controller (without single-token aggregation) still improves over both SC and DeepConf on HMMT25. Although the gains are naturally smaller than in the MoE setting, the result confirms that the controller's online signals and decision rules continue to help allocate compute effectively in a dense model, supporting the claim that HyPER's expand–reduce framework extends beyond MoE architectures.

Method	Accuracy (%) on HMMT25
SC	71.3
DeepConf (confidence pruning)	74.0
HyPER (controller only, no single-token aggregation)	78.7

Table 6. Non-MoE demonstration on HMMT25 with `deepseek-ai/DeepSeek-R1-0528-Qwen3-8B` at $S_{\max} = 32$.Even when single-token aggregation is removed entirely, HyPER's statistic-driven expand–reduce controller and length-/confidence-aware voting still improve over SC and DeepConf, indicating that the framework is not specific to MoE architectures.

Table 7. **Accuracy of SC vs. step-level tree search on AIME24** (Qwen3-30B-A3B-Thinking-2507). Forcing ToT-style step expansion on a post-training thinking model does not yield gains over standard SC.

Method	Acc. (%)
SC (path-based)	83.3
REBASE (8 branches)	80.0
REBASE (16 branches)	83.3
ETS	73.3
MCTS (REST-style expansion)	73.3

J. Effect of Step-Level Tree Search on Thinking Models

Our main experiments focus on SC-style multi-path decoding rather than explicit tree search over intermediate steps (Section 4.1). To support this design choice, we ran a small controlled study on AIME24 using Qwen3-30B-A3B-Thinking-2507, comparing standard self-consistency with several representative step-based methods adapted from the test-time scaling literature. All methods use the same prompt format and comparable test-time compute.

Table 7 reports the accuracies. The SC baseline attains 83.3% accuracy. When we force the model into REBASE-style tree search with 8 branches, accuracy drops to 80.0%, and even with 16 branches it only matches the SC baseline at 83.3%. PRM-guided ETS performs worse at 73.3%, and an MCTS-style CoT variant with REST-like expansion also reaches only 73.3%. In other words, naïvely imposing step-level tree search on this post-training "thinking" model does not improve—and often degrades—performance relative to simple path-based SC.

These results are consistent with our hypothesis that such models are tuned to produce internally coherent chains-of-thought, and that external tree structures can disrupt their native reasoning trajectories. This motivates our focus on SC-style multi-path decoding and the design of HyPER as a path-based expand–reduce method, rather than a ToT-style step-expansion algorithm.

K. Detailed Absolute Compute Accounting

To complement the normalized effective token cost reported in the main text, we provide the detailed, absolute physical execution metrics for our experiments in Table 8. All profiling was conducted on identical hardware setups (NVIDIA A100-80GB GPUs).

We report the following key metrics to evaluate the system-level overhead and efficiency:

- **Abs Tokens**: The absolute number of raw tokens physically generated by the model.
- **Eff Tokens**: The effective token cost used for mathematical budget alignment (charging K effective expansions for each K-routed expert proposal).
- **Latency (ms)**: The absolute end-to-end wall-clock latency per problem.
- **Peak/Avg Active Paths**: The maximum and average number of concurrent paths kept active in the decoding pool.
- **Est. Peak KV Cache (GB)**: The estimated peak memory footprint of the KV cache during the search process.

Notably, the physical latency tracks the **Abs Tokens** rather than the Eff Tokens. Because HyPER's dynamic controller and confidence-pruning mechanism heavily filter out unpromising paths, it consistently generates fewer raw tokens than the statically budgeted baselines (SC, Self-Certainty, and RoE). Consequently, the actual wall-clock latency and Peak KV Cache memory footprint are substantially reduced, offsetting the localized overhead of the controller and the batched MoE routing passes.

Table 8. Detailed absolute compute accounting and physical overhead profiling across models and datasets.

Source	Model	Dataset	Method	Acc. (%)	SC-norm	Abs Tokens	Eff Tokens	Latency (ms)	Peak Paths	Avg Paths	Peak KV (GB)
Table 2	Qwen3-30B	AIME24	SC	88.0	1.00	3.87×10^6	3.87×10^6	9.12×10^6	120	30.0	22.0
Table 2	Qwen3-30B	AIME24	Self-Certainty	83.3	0.94	3.64×10^6	3.64×10^6	8.57×10^6	120	28.5	20.7
Table 2	Qwen3-30B	AIME24	DeepConf	92.7	0.46	1.78×10^6	1.78×10^6	4.32×10^6	120	13.4	10.1
Table 2	Qwen3-30B	AIME24	RoE	88.7	1.38	2.54×10^6	5.34×10^6	7.19×10^6	80	15.2	14.4
Table 2	Qwen3-30B	AIME24	HyPER	94.0	0.54	1.55×10^6	2.09×10^6	4.09×10^6	80	12.6	12.45
Table 2	Qwen3-30B	AIME25	SC	86.0	1.00	4.22×10^6	4.22×10^6	9.98×10^6	120	33.0	24.0
Table 2	Qwen3-30B	AIME25	DeepConf	90.7	0.67	2.83×10^6	2.83×10^6	6.89×10^6	120	21.1	16.1
Table 2	Qwen3-30B	AIME25	RoE	84.7	1.64	3.30×10^6	6.92×10^6	9.35×10^6	80	23.4	18.8
Table 2	Qwen3-30B	AIME25	HyPER	95.3	0.71	2.22×10^6	3.00×10^6	5.88×10^6	88	20.4	18.48
Table 2	Qwen3-30B	HMMT25	SC	69.4	1.00	5.75×10^6	5.75×10^6	13.58×10^6	120	36.0	32.7
Table 2	Qwen3-30B	HMMT25	Self-Certainty	68.7	1.03	5.92×10^6	5.92×10^6	13.97×10^6	120	37.2	33.7
Table 2	Qwen3-30B	HMMT25	DeepConf	74.0	0.84	4.83×10^6	4.83×10^6	11.44×10^6	120	32.8	27.5
Table 2	Qwen3-30B	HMMT25	RoE	71.3	1.78	4.88×10^6	10.24×10^6	13.48×10^6	80	37.5	35.6
Table 2	Qwen3-30B	HMMT25	HyPER	78.7	0.77	3.14×10^6	4.43×10^6	8.38×10^6	104	36.8	28.4
Table 2	Qwen3-30B	HLE	SC	6.5	1.00	6.75×10^6	6.75×10^6	15.92×10^6	120	41.0	38.4
Table 2	Qwen3-30B	HLE	Self-Certainty	2.5	1.15	7.76×10^6	7.76×10^6	18.31×10^6	120	43.0	44.2
Table 2	Qwen3-30B	HLE	DeepConf	11.5	0.91	6.14×10^6	6.14×10^6	14.55×10^6	120	39.2	35.0
Table 2	Qwen3-30B	HLE	RoE	7.0	3.55	10.65×10^6	23.96×10^6	24.86×10^6	80	47.0	54.6
Table 2	Qwen3-30B	HLE	HyPER	13.5	0.81	3.90×10^6	5.47×10^6	10.01×10^6	92	45.4	34.9
Table 2	Qwen3-Next	AIME24	SC	90.7	1.00	3.95×10^6	3.95×10^6	9.30×10^6	120	31.0	27.0
Table 2	Qwen3-Next	AIME24	Self-Certainty	87.7	1.06	4.19×10^6	4.19×10^6	9.86×10^6	120	32.0	28.6
Table 2	Qwen3-Next	AIME24	DeepConf	94.7	0.47	1.86×10^6	1.86×10^6	4.50×10^6	120	12.0	12.7
Table 2	Qwen3-Next	AIME24	RoE	92.0	1.59	2.99×10^6	6.28×10^6	8.45×10^6	80	14.5	20.4
Table 2	Qwen3-Next	AIME24	HyPER	97.3	0.61	1.78×10^6	2.41×10^6	4.71×10^6	80	11.3	18.86
Table 2	Qwen3-Next	AIME25	SC	88.0	1.00	4.29×10^6	4.29×10^6	10.13×10^6	120	34.0	29.0
Table 2	Qwen3-Next	AIME25	DeepConf	94.0	0.53	2.27×10^6	2.27×10^6	5.53×10^6	120	19.6	15.3
Table 2	Qwen3-Next	AIME25	RoE	86.7	1.46	2.98×10^6	6.26×10^6	8.45×10^6	80	22.4	20.1
Table 2	Qwen3-Next	AIME25	HyPER	96.0	0.59	1.87×10^6	2.53×10^6	4.96×10^6	88	18.4	14.43
Table 2	Qwen3-Next	HMMT25	SC	76.7	1.00	5.95×10^6	5.95×10^6	14.05×10^6	120	37.0	40.1
Table 2	Qwen3-Next	HMMT25	Self-Certainty	70.0	1.00	5.95×10^6	5.95×10^6	14.05×10^6	120	38.0	40.7
Table 2	Qwen3-Next	HMMT25	DeepConf	80.7	0.76	4.52×10^6	4.52×10^6	10.72×10^6	120	31.8	30.5
Table 2	Qwen3-Next	HMMT25	RoE	78.0	1.76	4.99×10^6	10.47×10^6	13.74×10^6	80	39.5	41.2
Table 2	Qwen3-Next	HMMT25	HyPER	85.3	0.74	3.10×10^6	4.40×10^6	8.07×10^6	104	33.1	29.8
Table 2	Qwen3-Next	HLE	SC	7.0	1.00	6.95×10^6	6.95×10^6	16.44×10^6	120	42.0	46.8
Table 2	Qwen3-Next	HLE	Self-Certainty	2.5	1.25	8.69×10^6	8.69×10^6	20.57×10^6	120	44.0	58.6
Table 2	Qwen3-Next	HLE	DeepConf	11.0	0.84	5.84×10^6	5.84×10^6	13.91×10^6	120	39.4	39.4
Table 2	Qwen3-Next	HLE	RoE	6.5	2.98	9.22×10^6	20.71×10^6	22.35×10^6	80	48.0	62.1
Table 2	Qwen3-Next	HLE	HyPER	15.5	0.76	3.77×10^6	5.28×10^6	9.58×10^6	132	40.9	43.6