GLIDE: Generalized Layer-wise Intrinsic Distributional Evaluation for Heterogeneous LLM Agents

Anonymous ACL submission

Abstract

LLM agents require reliable step-level evaluation to compare candidate branches and allocate computation effectively. However, lightweight evaluation remains challenging. External verifiers introduce additional inference cost, while agent-produced confidence or self-evaluation scores can be miscalibrated, especially when candidates are generated by heterogeneous agents. We propose **G**eneralized **L**ayer-wise **I**ntrinsic **D**istributional **E**valuation (**GLIDE**) for LLM agents. GLIDE derives intrinsic step evidence from layer-wise residual coherence, which measures whether local residual updates consistently support the global residual change induced by a candidate step. It calibrates this evidence against the recent score distribution of the generating agent and converts it into a pessimistic reward that jointly accounts for absolute residual evidence and agent-relative standing. The reward provides a cross-agent value signal for MCTS branch selection, while normalized predictive uncertainty guides adaptive branching. Experiments on multi-hop reasoning, sequential decision making, and symbolic logic show that GLIDE improves task performance, step-level ranking quality, and computational efficiency without external verifiers or task-specific supervision.

1 Introduction

Large language models are increasingly used as agents for complex reasoning and interactive decision-making, where success depends on a sequence of intermediate decisions rather than a single generation. A locally plausible yet unhelpful step can redirect the agent toward an erroneous trajectory and cause errors to accumulate. Recent frameworks mitigate this brittleness by exploring multiple candidate continuations through tree-structured reasoning, MCTS-style planning, and multi-agent trajectory exploration (Yao et al., 2023a; Hao et al., 2023; Zhou et al., 2024; Zhu et al., 2026). Yet exploration is only useful when candidate branches can be reliably compared and computation can be allocated to promising directions. Inaccurate, sparse, or poorly calibrated comparison signals may repeatedly favor misleading trajectories, making intermediate-step evaluation a central bottleneck in inference-time agent control.

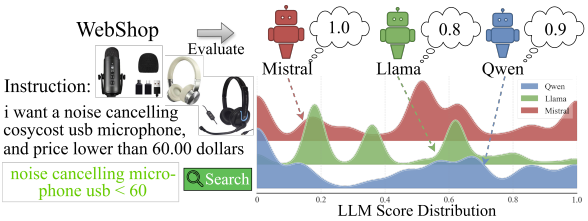

Figure 1: **Score calibration mismatch across heterogeneous agents.** Self-generated score distributions show that different LLM agents produce incompatible confidence scales on comparable inputs, making raw scores unreliable for cross-agent comparison.

A natural solution is to train Process Reward Models, which provide dense step-level supervision through dedicated discriminators (He et al., 2024; Zhang et al., 2024; Zhao et al., 2025). While effective in controlled settings, PRMs require task-specific supervision data, additional training procedures, and extra verifier inference. These requirements make them difficult to deploy as lightweight, general-purpose evaluators for online branch comparison. As a result, many inference-time systems turn to lighter alternatives based on prompt-based self-evaluation or LLM-as-a-Judge (Gu et al., 2024; Zhuge et al., 2024; You et al., 2026), where an LLM is asked to score or compare candidate steps without training a separate reward model.

However, judge-based evaluation introduces its own limitations. Its judgments are bounded by the reasoning capability and calibration behavior of the evaluator model, and it may assign high scores to fluent yet incorrect intermediate steps. The issue becomes more pronounced in heterogeneous agent

pools (Park et al., 2025; Ye et al., 2025; Zhu et al., 2026), where candidate branches may be generated by agents built on different base models, prompting policies, or decoding behaviors. Different agents may contribute complementary reasoning patterns, action preferences, and failure modes, but they also produce confidence estimates, self-evaluation scores, or other generator-dependent signals on incompatible scales. As illustrated in Figure 1, a numerically high score from one agent may reflect calibration bias rather than a genuinely better candidate step. Consequently, lightweight evaluation in heterogeneous agent inference faces a practical dilemma. External evaluators can provide a more shared scoring basis but are costly to invoke at every candidate step, whereas generator-dependent signals are inexpensive but cannot be directly compared across agents.

This dilemma motivates an evaluation signal that is intrinsic to generation, inexpensive to compute, and calibrated before being used for cross-agent comparison. Rather than invoking an external verifier, we examine whether useful candidate steps leave consistent signatures in the generator's internal computation. Under the residual-stream view of Transformers (Elhage et al., 2021), layer-wise updates write additive changes into a shared representation stream. We find that useful candidate steps tend to exhibit stable local-to-global residual coherence, meaning that layer-wise updates consistently align with the global update induced by the candidate step. Failure-specific transitions, in contrast, show weaker or less stable agreement between local layer updates and the overall step update. This regularity suggests an intrinsic indicator of step-level usefulness that can be computed from the generator's forward pass without task-specific supervision or additional judge calls.

Building on this observation, we propose GLIDE, **G**eneralized **L**ayer-wise **I**ntrinsic **D**istributional **E**valuation for heterogeneous LLM-agent MCTS. GLIDE first derives a step-level intrinsic preference signal from layer-wise residual coherence, capturing whether local hidden-state updates consistently support the global update induced by a candidate step. It then calibrates this signal against each agent's recent score distribution and converts it into a pessimistic reward, making candidate steps produced by different agents more reliably comparable. The calibrated reward is used for MCTS value estimation and further combined with predictive uncertainty to support adaptive branching. As a result, GLIDE enables intrinsic node evaluation and resource allocation without external verifiers, task-specific supervision, or additional judge passes.

Our contributions are summarized as follows:

- We identify layer-wise residual coherence as an intrinsic regularity for step-level preference estimation.
- We propose GLIDE, which combines intrinsic layer-wise evidence with agent-wise distributional calibration for heterogeneous LLM-agent MCTS.
- Experiments show that GLIDE improves performance-cost trade-offs and step-level preference quality.

2 Related Work

Agentic Reasoning and Planning LLM reasoning has shifted from linear generation to structured search and planning, including CoT-style rationale elicitation (Wei et al., 2022), interactive refinement with ReAct (Yao et al., 2023b) and Reflexion (Shinn et al., 2023), and multi-path aggregation (Wang et al., 2023; Zhang et al., 2023). Recent work further formulates reasoning as state-space search, covering tree/graph exploration such as ToT (Yao et al., 2023a), GoT (Besta et al., 2024), and MDToC (Ta and Oates, 2025), MCTS-style planners such as RAP (Hao et al., 2023) and LATS (Zhou et al., 2024), and heterogeneous multi-agent systems such as SYMPHONY (Zhu et al., 2026), MoA (Wang et al., 2024a), and MASTER (Gan et al., 2025). However, these systems still rely on reliable value estimation, where judge-based scoring adds inference cost and heterogeneous agents introduce non-comparable score scales. This motivates intrinsic, calibrated search-control signals without external verifiers.

Verification Mechanisms and Process Supervision Intermediate-step evaluation mainly follows two lines: extrinsic process supervision and intrinsic model signals. PRM-based methods provide dense step-level feedback through trained discriminators (Wang et al., 2024b; He et al., 2024), but even with automated synthesis or self-training (Zhang et al., 2025, 2024), they require task-specific data, extra training, and additional inference. Intrinsic alternatives based on token-level statistics, including perplexity, entropy, and

energy scores (Si et al., 2022; Huang et al., 2023; Malinin and Gales, 2020), avoid these costs but largely reflect local confidence and can fail under fluent hallucinations. Recent hidden-state methods support trace diagnosis or trajectory pruning (Vilas et al., 2025; Liang et al., 2025, 2026; Bi et al., 2025; Jiang et al., 2026), and CoE (Wang et al., 2025b) studies hidden-state trajectories, but they mainly target full-chain analysis or local smoothness rather than online step-wise ranking for calibrated heterogeneous control. This motivates a supervision-free intrinsic signal tailored to step-wise heterogeneous search.

3 GLIDE: Generalized Layer-wise Intrinsic Distributional Evaluation

GLIDE constructs a calibrated step-level reward for heterogeneous LLM-agent search without reward-model training or external judge calls. Specifically, it measures layer-wise residual coherence as the intrinsic stability score S_{ISLE}, normalizes this score within the generating agent's recent behavior, and forms a pessimistic reward R_{GLIDE} for cross-agent value comparison. The calibrated reward drives MCTS selection, and agent-relative predictive uncertainty further determines how many successors should be expanded. The full algorithm is given in Appendix A.

3.1 Layer-wise Intrinsic Stability Evaluation

We construct an intrinsic step-level signal from the generator's forward computation by measuring whether a candidate step is supported by stable residual evidence. Under the residual-stream view of Transformers (Elhage et al., 2021), each layer writes an additive update into a shared representation stream, and prior work suggests that hidden-state dynamics contain structured signals related to reasoning behavior (Wang et al., 2025b,a; Zhu et al., 2025; Geva et al., 2022; Dar et al., 2023). Thus, a useful step should not induce layer-wise updates that behave as unrelated perturbations; instead, these updates should coherently accumulate toward the global residual change associated with the step.

Consider an input state x and a candidate step a realized as a continuation $y_{1:\ell}$. All quantities are conditioned on the transition (x, a). Let $\mathbf{H}^{(l)} \in \mathbb{R}^{\ell \times d}$ denote the hidden states of the generated continuation at layer l. We summarize the layer representation of this step by $\mathbf{h}^{(l)} = \frac{1}{\ell}\sum_{j=1}^{\ell} \mathbf{H}_j^{(l)}$. The local residual update at layer l and the global residual update are defined as

$$\mathbf{u}_l = \mathbf{h}^{(l+1)} - \mathbf{h}^{(l)}, \quad \mathbf{u}_{\text{global}} = \mathbf{h}^{(L)} - \mathbf{h}^{(0)}. \quad (1)$$

We quantify layer-wise support through the directional agreement between each local update $\mathbf{u}_l$ and the global residual update $\mathbf{u}_{\text{global}}$. This choice is meaningful when the residual updates of a candidate step are not arbitrary perturbations, but share a coherent low-dimensional structure and accumulate toward the same step-level change. We state this condition as follows.

Assumption 1 (Coherent residual accumulation)**.** *For a fixed candidate step, the layer-wise residual updates $\{\mathbf{u}_l\}_{l=0}^{L-1}$ concentrate near a shared low-dimensional residual subspace, within which a dominant component accumulates consistently across layers with non-vanishing magnitude.*

Under this assumption, a candidate step is supported by residual evidence that is both locally structured and progressively accumulated. The concentration condition prevents the layer-wise updates from behaving as independent high-dimensional fluctuations, while the dominant-component condition ensures that their contributions do not vanish or cancel across layers. Consequently, the average local-to-global agreement provides a natural proxy for the stability of intrinsic step evidence. Appendix C.2.1 and Appendix C.2.2 provide the corresponding low-rank derivation and margin analysis. We therefore define the residual-coherence term as

$$\Phi_{\text{coh}}(x, a) = \text{sigmoid}\left(\frac{\sqrt{d}}{L}\sum_{l=0}^{L-1}\cos(\mathbf{u}_l, \mathbf{u}_{\text{global}})\right), \quad (2)$$

where the cosine terms measure local-to-global directional support, and the layer average captures whether such support is persistent across the computation. The factor $\sqrt{d}$ compensates for the concentration of high-dimensional cosine values (Vershynin, 2018), while the sigmoid maps the aggregated evidence to a bounded scale.

Residual coherence alone may still assign a high value to an internally consistent but weakly determined continuation. We therefore introduce token-level decisiveness as a conservative gate rather than as an independent evaluator:

$$\Phi_{\text{dec}}(x, a) = \left(\frac{1}{\ell}\sum_{j=1}^{\ell}\max_{w\in\mathcal{V}} p_\theta(w \mid x, y_{<j})\right)^2. \quad (3)$$

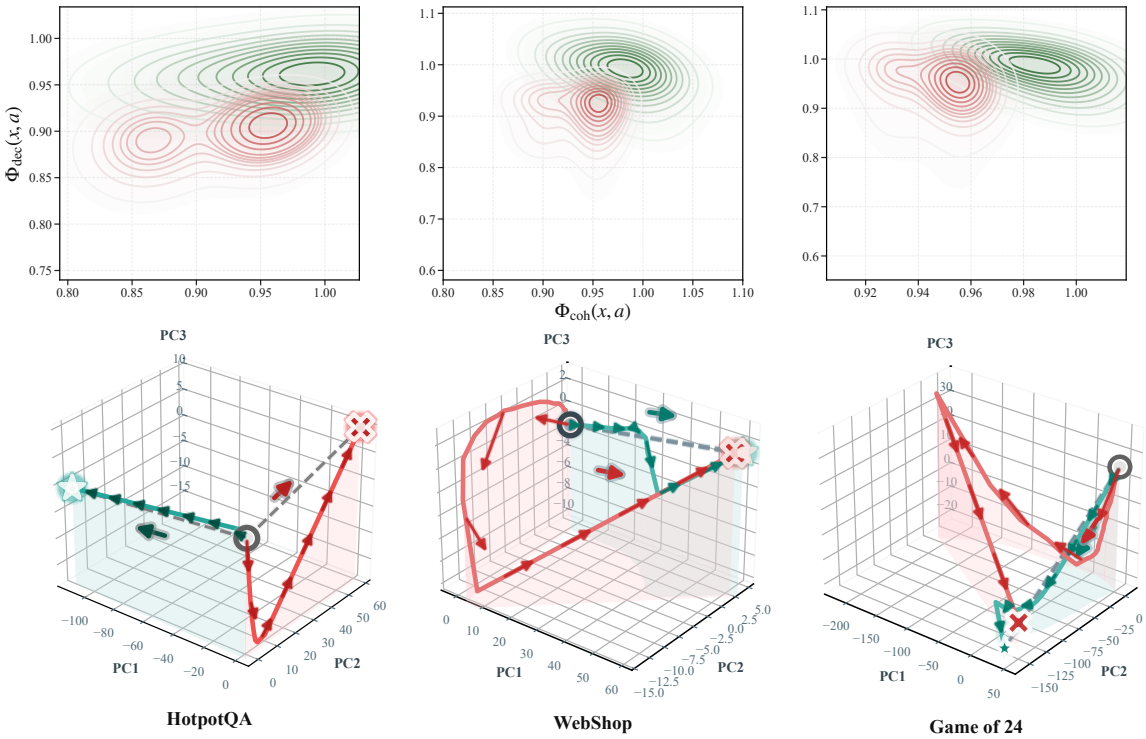

Figure 2: **Diagnostics of intrinsic stability in layer-wise evidence. Top**: Transitions from successful trajectories (Green) concentrate in high-coherence/high-decisiveness regions, while transitions unique to failed trajectories (Red) are dispersed in the $(\Phi_{\text{coh}}, \Phi_{\text{dec}})$ space. **Bottom**: Successful trajectories (Green) exhibit stable layer-wise residual evidence, whereas failed trajectories (Red) fluctuate and do not consistently progress toward a shared direction.

This term downweights steps generated under diffuse predictive distributions. The square makes the gate stricter, so that a candidate obtains a high score only when the model is both internally coherent and locally decisive.

We then define the **I**ntrinsic **S**tability of **L**ayer-wise **E**vidence (ISLE) score, denoted as S_{ISLE}, by

$$S_{\text{ISLE}}(x, a) = \Phi_{\text{coh}}(x, a) \cdot \Phi_{\text{dec}}(x, a). \quad (4)$$

A high S_{ISLE} therefore requires two forms of evidence to agree. Layer-wise residual updates should consistently support the global residual update, and the decoder should produce the continuation with sufficient local decisiveness. Because both terms are computed from the same forward pass, ISLE does not require task-specific supervision or additional verifier calls.

As a sanity check, we examine whether the intrinsic stability evidence terms are associated with successful transitions in the inference traces. Figure 2 shows that successful transitions are more concentrated in regions with high Φ_{coh} and high Φ_{dec}, whereas transitions that appear only in failed trajectories are more diffuse and exhibit weaker residual evidence. This diagnostic provides empirical support for the coherent residual accumulation assumption behind S_{ISLE}.

3.2 Distributional Calibration for Heterogeneous Agents

The intrinsic score S_{ISLE} provides an agent-local step signal, but heterogeneous search requires comparing candidates generated by different agents. Let $\mathcal{M} = \{M_i\}_{i=1}^{B}$ be the agent pool, and denote the score of a step a generated by M_i at state x as $S_{\text{ISLE}}^{(i)}(x, a)$. Within a fixed agent, this score is expected to preserve local preference information because it measures whether the agent's residual evidence coherently supports the generated transition.

Proposition 1 (Agent-local preference consistency of S_{ISLE})**.** *Consider two candidate steps a and a' generated by the same agent M_i at state x. Let $q(x, a)$ denote the latent transition utility of a, and define $\Delta_q(a, a') = q(x, a) - q(x, a')$, $\Delta_S^{(i)}(a, a') = S_{\text{ISLE}}^{(i)}(x, a) - S_{\text{ISLE}}^{(i)}(x, a')$.*

Assume that, within agent M_i, $S_{\text{ISLE}}^{(i)}$ is an order-preserving noisy surrogate of q with sub-Gaussian perturbation. Then whenever $\Delta_q(a, a') > 0$, there exists an agent-dependent constant $\kappa_i > 0$ such

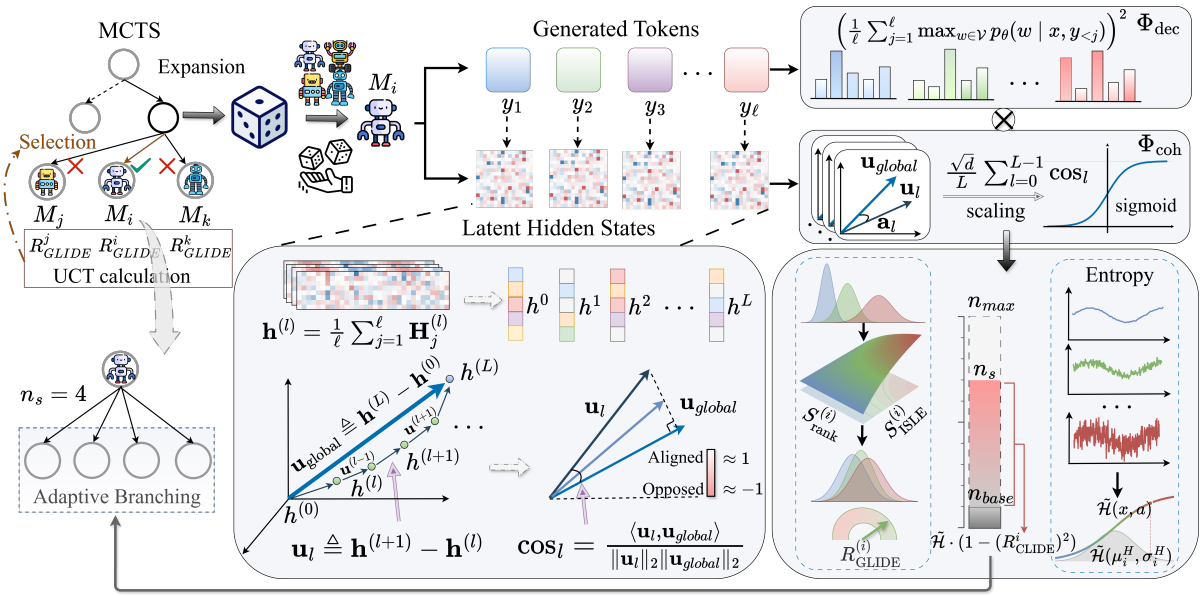

Figure 3: Overview of GLIDE

that,

$$\Pr\Big(\Delta_S^{(i)}(a,a') \le 0\Big) \le \exp\big(-\kappa_i \Delta_q(a,a')^2\big)\,, \qquad (5)$$

Proposition 1 states that the probability of a ranking reversal, $\Pr(\Delta_S^{(i)}(a,a') \le 0)$, decreases exponentially with the latent utility gap $\Delta_q(a,a')$. The full proof is provided in Appendix C.3.

However, agent-local consistency does not ensure cross-agent comparability, since different agents may exhibit different hidden-state dynamics, confidence scales, and intrinsic score distributions. Thus, heterogeneous candidate selection must retain the local preference information of S_{ISLE} while correcting agent-specific scale mismatch.

We reduce this mismatch by rank-normalizing scores within each agent. For each agent M_i, we maintain a FIFO window $\mathcal{W}_i^S$ of recent S_{ISLE} produced by that agent and define the empirical rank

$$S_{\text{rank}}^{(i)}(x,a) = \frac{1}{|\mathcal{W}_i^S|}\sum_{\chi\in\mathcal{W}_i^S}\mathbb{I}\Big[\chi \le S_{\text{ISLE}}^{(i)}(x,a)\Big]\,, \qquad (6)$$

which measures its standing within the generating agent's recent score distribution.

Rank normalization improves cross-agent comparability, but using rank alone would discard absolute intrinsic strength. A step may rank highly only because its agent has recently produced weak candidates, while another step may have strong residual evidence but appear less exceptional under a more confident generator. We therefore combine the raw score, which reflects absolute residual evidence, with the rank score, which reflects agent-relative standing. The GLIDE reward is defined by harmonic fusion:

$$R_{\text{GLIDE}}^{(i)}(x,a) = \frac{2S_{\text{ISLE}}^{(i)}(x,a)S_{\text{rank}}^{(i)}(x,a)}{S_{\text{ISLE}}^{(i)}(x,a) + S_{\text{rank}}^{(i)}(x,a)}. \qquad (7)$$

Because harmonic fusion is dominated by the smaller term, $R_{\text{GLIDE}}^{(i)}$ is pessimistic: a step receives a high reward only when it has both strong intrinsic evidence and high agent-relative standing. This converts the agent-local signal justified by Proposition 1 into a calibrated cross-agent comparison signal for value estimation and adaptive branching.

3.3 Adaptive Branching with Calibrated Rewards

Given the calibrated reward R_{GLIDE}, GLIDE uses it as the transition value in MCTS and further derives an adaptive branching rule for computation allocation. For a candidate step a generated by agent M_i at state x, $R_{\text{GLIDE}}^{(i)}(x,a)$ initializes the value of the expanded transition, which is then backpropagated through Eq. (12) and used by the UCT objective in Eq. (10). Thus, branch selection is driven by a verifier-free value signal that is intrinsic to the generator and calibrated within the corresponding agent distribution.

To support online calibration, GLIDE maintains two agent-wise FIFO windows: $\mathcal{W}_i^S$ for recent

Table 1: HotpotQA (Hot).

Method	Exact Match ↑
CoT	0.34
CoT-SC	0.38
ReAct	0.39
Reflexion	0.51
ToT	0.55
RAP	0.60
LATS	0.71
Beam Retrieval	0.73
MASTER	0.76
SYMPHONY-S	0.59
SYMPHONY-L	0.79
GLIDE	**0.73**

Table 2: WebShop (Web).

Method	Score ↑	SR ↑
IL	0.60	0.29
IL+RL	0.62	0.29
ReAct	0.54	0.32
Reflexion	0.64	0.35
MoA	0.31	0.42
WebGUM	0.68	0.45
AgentKit	0.70	–
LATS	0.76	0.38
MASTER	0.80	–
Human Expert	0.82	0.60
SYMPHONY-S	0.82	0.56
SYMPHONY-L	0.88	0.72
GLIDE	**0.84**	**0.57**

Table 3: Game of 24 (G24).

Method	Success Rate ↑
Llama-2(70B)	0.04
Gemini pro	0.08
Claude-3 Opus	0.07
GPT-4 turbo	0.09
CoT	0.08
Reflexion	0.12
ToT	0.20
GoT	0.21
MoA	0.13
RAP	0.40
MDToC	0.30
LATS	0.44
SYMPHONY-S	0.43
GLIDE	**0.54**

Note: Metrics are normalized to the [0,1] range; A dash (–) marks those not reported in the publication.

S_{ISLE} values and $\mathcal{W}_i^H$ for recent predictive entropies. The former is used for empirical rank calibration in Eq. (6), while the latter provides an agent-relative uncertainty scale. New observations are inserted only after the current reward and uncertainty are computed, avoiding self-normalization.

While R_{GLIDE} estimates the exploitation value of a transition, expansion width should also depend on whether its successor still requires exploration. We therefore allocate computation by coupling agent-relative predictive uncertainty with reward saturation. For a continuation $y_{1:\ell}$ generated by M_i, let p_i denote its token distribution and compute $\bar{\mathcal{H}}_i(x,a) = -\frac{1}{\ell}\sum_{j=1}^{\ell}\sum_{w\in\mathcal{V}} p_i(w \mid x, y_{<j}) \log p_i(w \mid x, y_{<j})$. Given the mean μ_i^H and standard deviation σ_i^H of $\mathcal{W}_i^H$, the normalized uncertainty is

$$\widetilde{\mathcal{H}}_i(x,a) = \text{sigmoid}\left((\bar{\mathcal{H}}_i(x,a) - \mu_i^H)/\sigma_i^H\right). \quad (8)$$

The branching width for the successor induced by (x,a) is then

$$n(x,a) = \Big\lfloor n_{\min} + (n_{\max} - n_{\min})\widetilde{\mathcal{H}}_i(x,a) \cdot \left[1 - \left(R_{\text{GLIDE}}^{(i)}(x,a)\right)^2\right] \Big\rfloor. \quad (9)$$

This formulation favors uncertain, insufficiently supported successors while suppressing redundant expansion near high-reward regions. Given $n(x,a)$, GLIDE uniformly samples generators from the heterogeneous pool to preserve diversity without a learned router.

Overall, GLIDE provides a calibrated intrinsic reward that makes generator-internal step evidence usable for heterogeneous MCTS control. The complete Algorithm is given in Appendix A.

4 Experiments

4.1 Experimental Settings

Following SYMPHONY-S (Zhu et al., 2026), we use a heterogeneous open-source model pool compatible with consumer-grade hardware: Qwen2.5-7B-Instruct-1M (Yang et al., 2024), Mistral-7B-Instruct-v0.3 (Jiang et al., 2023), and Llama-3.1-8B-Instruct (Grattafiori et al., 2024). Unless otherwise stated, all controlled task experiments follow prior protocols (Shinn et al., 2023; Yao et al., 2023a; Zhou et al., 2024; Gan et al., 2025; Zhu et al., 2026), using the same prompt formats, decoding settings, and search budgets as SYMPHONY-S. We report the average performance over three runs. For GLIDE, we set $|\mathcal{W}_i^S| = |\mathcal{W}_i^H| = 50$ and use $n_{\min} = 1$, $n_{\max} = 4$ in Eq. (9), matching the maximum fixed expansion width of SYMPHONY-S. Additional implementation details are provided in Appendix D.

4.2 Main Results

We evaluate GLIDE on three benchmarks covering distinct forms of agent inference: HotpotQA (Yang et al., 2018) for multi-hop reasoning, WebShop (Yao et al., 2022) for goal-directed sequential decision making, and Game of 24 (Yao et al., 2023a) for symbolic arithmetic reasoning. HotpotQA is evaluated by **Exact Match (EM)**, WebShop by **Success Rate (SR)** and **Average Score**, and Game of 24 by **Success Rate**. We follow the task-specific evaluation protocols and report baseline details in Appendix D.4.

Tables 1, 2, and 3 show that GLIDE achieves strong performance across all three settings. Un-

Table 4: **Step-level preference quality of R_{GLIDE}.** We report ranking metrics (P@1, MRR, NDCG@3) on expanded child nodes from HotpotQA (Hot), WebShop (Web), and Game of 24 (G24), with Avg denoting the mean. Efficiency reports per-candidate scoring overhead; ✓ indicates negligible overhead. Hetero-Judge randomly selects scorers from the heterogeneous agent pool. Baselines are detailed in Appendix E.4.1.

Category	Method	P@1				MRR				NDCG@3				Efficiency	
		Hot	Web	G24	**Avg**	Hot	Web	G24	**Avg**	Hot	Web	G24	**Avg**	**Time**	**Mem**
Intrinsic	Perplexity	0.17	0.37	0.38	0.31	0.52	0.66	0.64	0.61	0.59	0.71	0.65	0.65	✓	✓
	Energy	0.24	0.00	0.00	0.08	0.56	0.44	0.33	0.44	0.63	0.30	0.30	0.41	✓	✓
	LN-Entropy	0.33	0.00	0.00	0.11	0.61	0.44	0.33	0.46	0.69	0.30	0.30	0.43	✓	✓
	CoE-R	0.44	0.48	0.40	0.44	0.68	0.72	0.64	0.68	0.74	0.79	0.62	0.72	✓	✓
	CoE-C	0.46	0.47	0.41	0.45	0.69	0.71	0.65	0.68	0.75	0.78	0.66	0.73	✓	✓
Extrinsic	PRM-RLHFlow	0.36	0.47	0.36	0.40	0.63	0.72	0.63	0.66	0.68	0.77	0.63	0.69	0.10s	13.7k
	GenPRM-7B	0.41	0.36	0.30	0.36	0.66	0.66	0.60	0.64	0.72	0.73	0.60	0.68	0.10s	14.2k
	PRM-Skywork	0.32	0.40	0.48	0.40	0.62	0.67	0.70	0.66	0.68	0.74	0.67	0.70	0.04s	3.0k
Multi-Agent	Hetero-Judge	0.45	0.56	0.41	0.47	0.69	0.74	0.66	0.70	0.71	0.80	0.67	0.73	4.63s	15.5k
Ours	**GLIDE**	**0.59**	**0.48**	**0.43**	**0.50**	**0.77**	**0.73**	**0.67**	**0.72**	**0.83**	**0.79**	**0.71**	**0.78**	✓	✓

Table 5: **Controlled comparison across agent pools.** Pools use L (Llama-3.1-8B-Instruct), M (Mistral-7B-Instruct-v0.3), and Q (Qwen2.5-7B-Instruct-1M). Performance uses EM, SR, and Score; Cost denotes total input/output tokens in millions. Teal marks the better result between SYMPHONY (Sym) and GLIDE.

	HotpotQA (Hot)				WebShop (Web)						Game of 24 (G24)			
	EM (↑)		Cost (M) (↓)		SR (↑)		Score (↑)		Cost (M) (↓)		SR (↑)		Cost (M) (↓)	
Pool Combo	Sym	**Ours**	Sym	**Ours**	Sym	**Ours**	Sym	**Ours**	Sym	**Ours**	Sym	**Ours**	Sym	**Ours**
Single: $\{L\}$	0.27	0.36	28.5	18.7	0.08	0.19	0.21	0.42	102.0	175.7	0.21	0.30	4.8	3.1
Single: $\{M\}$	0.32	0.44	11.1	14.9	0.35	0.35	0.65	0.67	52.9	48.0	0.15	0.23	25.5	11.5
Single: $\{Q\}$	0.30	0.49	53.3	12.1	0.39	0.46	0.66	0.71	54.4	20.2	0.14	0.14	9.6	4.6
Dual: $\{L, M\}$	0.52	0.53	24.4	12.0	0.37	0.45	0.73	0.77	82.5	44.0	0.31	0.40	6.4	6.5
Dual: $\{M, Q\}$	0.50	0.62	21.5	10.7	0.46	0.52	0.74	0.80	47.6	29.2	0.21	0.36	18.7	2.5
Dual: $\{L, Q\}$	0.47	0.60	19.9	13.2	0.40	0.53	0.74	0.81	68.7	25.6	0.30	0.47	4.7	2.8
Full: $\{L, M, Q\}$	0.59	0.73	18.7	10.5	0.56	0.57	0.82	0.84	139.3	26.6	0.43	0.54	6.3	4.6

der the same open-source heterogeneous-agent configuration, GLIDE consistently improves over SYMPHONY-S, indicating that the gain comes from more reliable candidate evaluation and computation allocation rather than from a stronger model pool. The improvement is observed across qualitatively different sources of difficulty, including evidence selection in multi-hop reasoning, goal maintenance in long-horizon interaction, and branch isolation in symbolic arithmetic reasoning.

Compared with broader task-specific baselines, GLIDE remains competitive while using only lightweight intrinsic evaluation. This suggests that calibrated intrinsic rewards and adaptive branching jointly improve candidate comparison and computation allocation in heterogeneous agents.

4.3 Preference Quality of Calibrated Rewards

To verify that GLIDE's task-level gains come from better candidate selection, we evaluate R_{GLIDE} on expanded child nodes from the three main tasks. Candidates are ranked by different signals and labeled by the final outcome of their continuations, testing whether successful continuations are prioritized. We report P@1, MRR, NDCG@3, per-candidate latency, and memory overhead.

Table 4 shows that R_{GLIDE} achieves the strongest overall ranking quality, outperforming probability-based intrinsic metrics, hidden-state baselines, extrinsic PRMs, and heterogeneous judge scores without task-specific supervision or additional verifier calls. Its gains over raw or uncalibrated signals show that distributional calibration turns S_{ISLE} from intrinsic evidence into a more reliable heterogeneous candidate-comparison signal.

4.4 Controlled Comparison and Ablation Study

We further examine whether GLIDE's gains hold under controlled agent-pool configurations and which components contribute to them. Table 5 compares GLIDE with SYMPHONY using single-

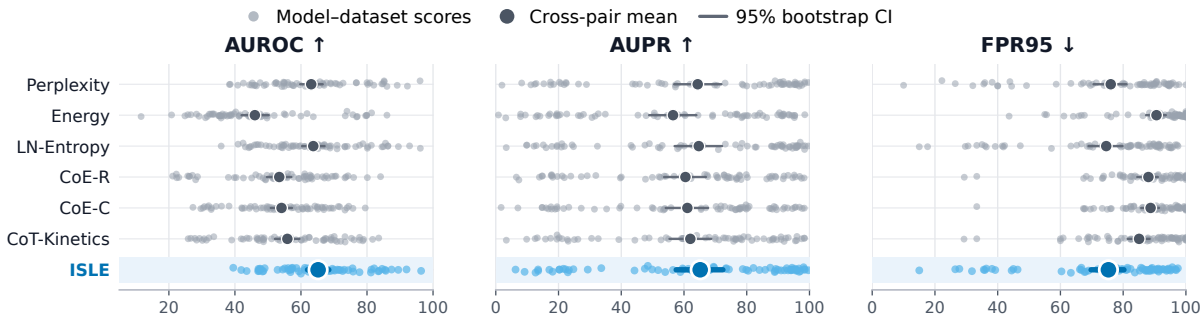

Figure 4: **Cross-task outcome separability of S_{ISLE}.** We report average AUROC, FPR95, and AUPR across MGSM, GPQA, HumanEval, and LLMs from different families and scales. Higher AUROC/AUPR and lower FPR95 indicate stronger separation. Full results are in Appendix F.3.

Table 6: **Ablation Study** (Full Pool; WebShop: SR).

Variant	Hot	Web	G24
w/o Adap. Branch.	0.68	0.50	0.52
Raw S_{ISLE}	0.63	0.52	0.47
Calib. Judge	0.66	0.56	0.50
Rand. Agent	0.59	0.56	0.43
UCB Dispatch	0.65	0.55	0.46
Full	**0.73**	**0.57**	**0.54**

agent, dual-agent, and full heterogeneous pools. Across these settings, GLIDE improves task performance and token efficiency, showing that the gains do not come from stronger base models but from calibrated intrinsic evaluation and adaptive computation allocation. The results also indicate that adding more agents alone is insufficient, since heterogeneous candidates can introduce noisy branch comparison without calibrated score alignment.

Table 6 isolates the effect of each component. Replacing R_{GLIDE} with raw S_{ISLE} weakens performance, confirming the need for distributional calibration. Removing adaptive branching reduces the benefit of uncertainty-aware computation allocation, while the calibrated judge variant shows that calibration alone is insufficient when the underlying signal comes from external judgment rather than generator-internal evidence. Alternative dispatching strategies remain competitive in some cases but do not match the full model. These results support the full design of GLIDE, where intrinsic evidence, agent-wise calibration, and adaptive branching work jointly within one controller. Appendix E further reports test-time compute scaling, parameter sensitivity, calibration-window analysis, fixed-search evaluation baselines, and failure cases.

4.5 Cross-Task Outcome Separability

Finally, we test whether S_{ISLE} remains outcome-relevant beyond the main agent-control setting. We evaluate multiple open-source LLMs on MGSM (Shi et al., 2023), GPQA (Rein et al., 2023), and HumanEval (Chen et al., 2021), covering multilingual mathematical reasoning, scientific question answering, and code generation. For each model–task pair, generated solutions are labeled by final task outcome, and each scoring method is evaluated by its ability to separate successful generations from failed ones. We report AUROC and AUPR for separability and FPR95 for false positives at high recall.

Figure 4 shows that S_{ISLE} maintains clear separability across models and tasks. This complements the step-level preference analysis by showing that the intrinsic signal is not only useful inside the calibrated control pipeline, but also remains associated with final task success under direct generation. Full results, layer-subset analysis, and memory-overhead evaluation are provided in Appendix F.

5 Conclusion

This paper introduces GLIDE, which transforms layer-wise residual coherence into a cross-agent calibrated reward for heterogeneous LLM-agent search. By aligning intrinsic step evidence with agent-specific score distributions, GLIDE enables verifier-free value estimation and adaptive branching for efficient inference-time control. Experiments show that this calibrated intrinsic reward improves task performance and step-level preference quality with low overhead, highlighting the promise of intrinsic signals for complex reasoning.

Limitations

GLIDE requires access to hidden states, which may not be available for closed-source models. Its calibration is also designed for settings where a heterogeneous model pool repeatedly contributes candidate reasoning steps, so a single black-box model deployment is outside the primary scope of this work. Future work should study how similar calibration signals can be approximated when only partial model traces or API-level confidence signals are available.

References

Richard Bellman. 1966. Dynamic programming. *science*, 153(3731):34–37.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and 1 others. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 17682–17690.

Jinhe Bi, Danqi Yan, Yifan Wang, Wenke Huang, Haokun Chen, Guancheng Wan, Mang Ye, Xun Xiao, Hinrich Schütze, Volker Tresp, and Yunpu Ma. 2025. Cot-kinetics: A theoretical modeling assessing LRM reasoning process. *CoRR*, abs/2505.13408.

Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, Xiaoyi Dong, Haodong Duan, Qi Fan, Zhaoye Fei, Yang Gao, Jiaye Ge, Chenya Gu, Yuzhe Gu, Tao Gui, and 52 others. 2024. Internlm2 technical report. *CoRR*, abs/2403.17297.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *CoRR*, abs/2107.03374.

Rémi Coulom. 2006. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pages 72–83. Springer.

Guy Dar, Mor Geva, Ankit Gupta, and Jonathan Berant. 2023. Analyzing transformers in embedding space. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16124–16170.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, and 6 others. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2021/framework/index.html.

Hiroki Furuta, Kuang-Huei Lee, Ofir Nachum, Yutaka Matsuo, Aleksandra Faust, Shixiang Shane Gu, and Izzeddin Gur. 2024. Multimodal web navigation with instruction-finetuned foundation models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Bingzheng Gan, Yufan Zhao, Tianyi Zhang, Jing Huang, Li Yusu, Shu Xian Teo, Changwang Zhang, and Wei Shi. 2025. MASTER: A multi-agent system with LLM specialized MCTS. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 9409–9426, Albuquerque, New Mexico. Association for Computational Linguistics.

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 conference on empirical methods in natural language processing*, pages 30–45.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, and 1 others. 2024. A survey on llm-as-a-judge. *The Innovation*.

Shibo Hao, Yi Gu, Haotian Luo, Tianyang Liu, Xiyan Shao, Xinyuan Wang, Shuhua Xie, Haodi Ma, Adithya Samavedhi, Qiyue Gao, and 1 others. 2024. Llm reasoners: New evaluation, library, and analysis of step-by-step reasoning with large language models. *arXiv preprint arXiv:2404.05221*.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8154–8173.

Jujie He, Tianwen Wei, Rui Yan, Jiacai Liu, Chaojie Wang, Yimeng Gan, Shiwen Tu, Chris Yuhao Liu, Liang Zeng, Xiaokun Wang, Boyang Wang, Yongcong Li, Fuxiang Zhang, Jiacheng Xu, Bo An, Yang Liu, and Yahui Zhou. 2024. Skywork-o1 open series.

Yuheng Huang, Jiayang Song, Zhijie Wang, Shengming Zhao, Huaming Chen, Felix Juefei-Xu, and Lei Ma. 2023. Look before you leap: An exploratory study of uncertainty measurement for large language models. *arXiv preprint arXiv:2307.10236*.

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. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Xinyan Jiang, Ninghao Liu, Di Wang, and Lijie Hu. 2026. Beyond scalars: Evaluating and understanding LLM reasoning via geometric progress and stability. *CoRR*, abs/2603.10384.

Zhenwen Liang, Ruosen Li, Yujun Zhou, Linfeng Song, Dian Yu, Xinya Du, Haitao Mi, and Dong Yu. 2025. CLUE: non-parametric verification from experience via hidden-state clustering. *CoRR*, abs/2510.01591.

Zhixiang Liang, Beichen Huang, Zheng Wang, and Minjia Zhang. 2026. Hidden states as early signals: Step-level trace evaluation and pruning for efficient test-time scaling. *CoRR*, abs/2601.09093.

Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. 2020. Energy-based out-of-distribution detection. *Advances in neural information processing systems*, 33:21464–21475.

Andrey Malinin and Mark Gales. 2020. Uncertainty estimation in autoregressive structured prediction. *arXiv preprint arXiv:2002.07650*.

Microsoft. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *CoRR*, abs/2404.14219.

Sungjin Park, Xiao Liu, Yeyun Gong, and Edward Choi. 2025. Ensembling large language models with process reward-guided tree search for better complex reasoning. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 10256–10277.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. GPQA: A graduate-level google-proof q&a benchmark. *CoRR*, abs/2311.12022.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. 2023. Language models are multilingual chain-of-thought reasoners. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

Chenglei Si, Zhe Gan, Zhengyuan Yang, Shuohang Wang, Jianfeng Wang, Jordan Boyd-Graber, and Lijuan Wang. 2022. Prompting gpt-3 to be reliable. *arXiv preprint arXiv:2210.09150*.

Tung Duong Ta and Tim Oates. 2025. Mdtoc: Metacognitive dynamic tree of concepts for boosting mathematical problem-solving of large language models. *arXiv preprint arXiv:2512.18841*.

Gemma Team. 2025. Gemma 3 technical report. *CoRR*, abs/2503.19786.

Roman Vershynin. 2018. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press.

Martina G. Vilas, Safoora Yousefi, Besmira Nushi, Eric Horvitz, and Vidhisha Balachandran. 2025. Tracing the traces: Latent temporal signals for efficient and accurate reasoning. *CoRR*, abs/2510.10494.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. 2024a. Mixture-of-agents enhances large language model capabilities. *arXiv preprint arXiv:2406.04692*.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024b. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Yiming Wang, Pei Zhang, Siyuan Huang, Baosong Yang, Zhuosheng Zhang, Fei Huang, and Rui Wang. 2025a. Sampling-efficient test-time scaling: Self-estimating the best-of-n sampling in early decoding. *arXiv preprint arXiv:2503.01422*.

Yiming Wang, Pei Zhang, Baosong Yang, Derek F. Wong, and Rui Wang. 2025b. Latent space chain-of-embedding enables output-free LLM self-evaluation. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances*

in neural information processing systems, 35:24824–24837.

Yue Wu, Yewen Fan, So Yeon Min, Shrimai Prabhumoye, Stephen McAleer, Yonatan Bisk, Ruslan Salakhutdinov, Yuanzhi Li, and Tom Mitchell. 2024. Agentkit: Structured llm reasoning with dynamic graphs. *arXiv preprint arXiv:2404.11483*.

Wei Xiong, Hanning Zhang, Nan Jiang, and Tong Zhang. 2024. An implementation of generative prm.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 2024. Qwen2.5 technical report. *CoRR*, abs/2412.15115.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Rui Ye, Xiangrui Liu, Qimin Wu, Xianghe Pang, Zhenfei Yin, Lei Bai, and Siheng Chen. 2025. X-mas: Towards building multi-agent systems with heterogeneous llms. *arXiv preprint arXiv:2505.16997*.

Runyang You, Hongru Cai, Caiqi Zhang, Qiancheng Xu, Meng Liu, Tiezheng Yu, Yongqi Li, and Wenjie Li. 2026. Agent-as-a-judge. *CoRR*, abs/2601.05111.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024. Rest-mcts*: Llm self-training via process reward guided tree search. *Advances in Neural Information Processing Systems*, 37:64735–64772.

Jiahao Zhang, Haiyang Zhang, Dongmei Zhang, Yong Liu, and Shen Huang. 2023. End-to-end beam retrieval for multi-hop question answering. *arXiv preprint arXiv:2308.08973*.

Yao Zhang, Yu Wu, Haowei Zhang, Weiguo Li, Haokun Chen, Jingpei Wu, Guohao Li, Zhen Han, and Volker Tresp. 2025. Groundedprm: Tree-guided and fidelity-aware process reward modeling for step-level reasoning. *arXiv preprint arXiv:2510.14942*.

Jian Zhao, Runze Liu, Kaiyan Zhang, Zhimu Zhou, Junqi Gao, Dong Li, Jiafei Lyu, Zhouyi Qian, Biqing Qi, Xiu Li, and Bowen Zhou. 2025. Genprm: Scaling test-time compute of process reward models via generative reasoning. *arXiv preprint arXiv:2504.00891*.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. 2024. Language agent tree search unifies reasoning, acting, and planning in language models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Rui-Jie Zhu, Tianhao Peng, Tianhao Cheng, Xingwei Qu, Jinfa Huang, Dawei Zhu, Hao Wang, Kaiwen Xue, Xuanliang Zhang, Yong Shan, and 1 others. 2025. A survey on latent reasoning. *arXiv preprint arXiv:2507.06203*.

Wei Zhu, Zhiwen Tang, and Kun Yue. 2026. SYMPHONY: synergistic multi-agent planning with heterogeneous language model assembly. *CoRR*, abs/2601.22623.

Mingchen Zhuge, Changsheng Zhao, Dylan Ashley, Wenyi Wang, Dmitrii Khizbullin, Yunyang Xiong, Zechun Liu, Ernie Chang, Raghuraman Krishnamoorthi, Yuandong Tian, and 1 others. 2024. Agent-as-a-judge: Evaluate agents with agents. *arXiv preprint arXiv:2410.10934*.

APPENDIX

A GLIDE Algorithm 12

B Theoretical Background 12

B.1 Markov Decision Processes and LLM Agents 12

B.2 Monte Carlo Tree Search 12

C Theoretical Analysis 13

C.1 Setup and Notation 13

C.2 Analysis of Assumption 1 13

C.2.1 Low-dimensional Residual Subspace 14

C.2.2 Positive Local-to-global Coherence Margin 14

C.3 Proof of Proposition 1 15

D Main Experimental Details 16

D.1 Infrastructure and Model Configuration 16

D.2 MCTS and GLIDE Hyperparameters 16

D.3 Task-Specific Prompting and Settings 16

D.4 Main Benchmark Baselines 16

E Supplementary Analyses for Main Experiments 18

E.1 Test-Time Compute Scaling 19

E.2 Algorithmic Parameter Analysis 19

E.3 MCTS Evaluation Baselines under Fixed Search 20

E.4 Step-Level Preference and Efficiency Analysis 20

E.5 Failure Case Analysis 21

F Cross-Task Outcome Separability 22

F.1 Datasets and Models 22

F.2 Generation Protocol 22

F.3 Outcome-Separability Results 23

F.4 Layer-Subset Analysis 23

F.5 Memory Overhead of Hidden-State Scoring 23

G Use of AI Assistants 24

H Artifacts Statements 24

A Algorithm Pseudocode

Algorithm 1 presents the pseudocode of the proposed GLIDE framework.

Algorithm 1 GLIDE: Intrinsic Reward-Guided Heterogeneous MCTS

Input: initial state s_0, agent pool $\mathcal{M} = \{M_i\}_{i=1}^{B}$, MCTS budget K, UCT constant c, branching bounds $n_{\min}, n_{\max}$.
Initialize: tree $\mathcal{T} \leftarrow \{s_0\}$, node capacity $b(s_0) \leftarrow n_{\max}$, and FIFO windows $\mathcal{W}_i^S, \mathcal{W}_i^H$ for each $M_i \in \mathcal{M}$.
for $k = 1$ **to** K **do**
  $s \leftarrow s_0$
  {**1. Selection**}
  while s is not terminal **and** $|\mathrm{ch}(s)| \geq b(s)$ **do**
    $s \leftarrow \mathrm{argmax}_{s' \in \mathrm{ch}(s)} \mathrm{UCT}(s')$ {Eq. (10)}
  end while
  {**2. Expansion and intrinsic evaluation**}
  while s is not terminal **and** $|\mathrm{ch}(s)| < b(s)$ **do**
    Sample agent $M_i \sim \mathrm{Uniform}(\mathcal{M})$.
    Generate step a from state s using M_i, yielding successor s'.
    Extract hidden states and token distributions from the generation pass.
    Compute $S_{\mathrm{ISLE}}^{(i)}(s,a)$ using Eq. (4).
    Compute $S_{\mathrm{rank}}^{(i)}(s,a)$ from $\mathcal{W}_i^S$ using Eq. (6).
    Compute $R_{\mathrm{GLIDE}}^{(i)}(s,a)$ using Eq. (7).
    Compute mean entropy $\bar{\mathcal{H}}_i(s,a)$ and normalized uncertainty $\widetilde{\mathcal{H}}_i(s,a)$ using Eq. (8).
    Set successor capacity $b(s') \leftarrow n(s,a)$ using Eq. (9).
    Add s' to $\mathcal{T}$ as a child of s and store $R_{\mathrm{GLIDE}}^{(i)}(s,a)$.
    Backpropagate using Eq. (12).
    Insert $S_{\mathrm{ISLE}}^{(i)}(s,a)$ into $\mathcal{W}_i^S$ and $\bar{\mathcal{H}}_i(s,a)$ into $\mathcal{W}_i^H$.
  end while
end for
Return the best reasoning path from $\mathcal{T}$.

B Theoretical Background

B.1 Markov Decision Processes and LLM Agents

The sequential decision-making problem is formally defined by the Markov Decision Process (MDP) tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ (Bellman, 1966). Here, $\mathcal{S}$ denotes the state space, $\mathcal{A}$ represents the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ defines the transition dynamics, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1]$ serves as the discount factor. At each timestep t, the agent observes a state $s_t \in \mathcal{S}$, selects an action $a_t \in \mathcal{A}$, transitions to a new state $s_{t+1} \sim \mathcal{P}(\cdot \mid s_t, a_t)$, and receives a scalar reward $r_t = \mathcal{R}(s_t, a_t)$. The objective is to find a policy π that maximizes the expected cumulative discounted return $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$.

Large Language Models (LLMs) can be naturally integrated into this framework to support high-level reasoning. Specifically, an LLM can serve as a *policy* by generating actions (tokens or reasoning steps) conditioned on language-based state representations; as a *value function* by estimating expected returns from textual trajectories; or as a *world model* by predicting future states and rewards through learned internal knowledge. Unlike traditional reinforcement learning agents that rely on explicit environment modeling and manually designed signals, LLM-based agents leverage pre-training on large corpora to internalize commonsense, domain knowledge, and structured reasoning. This allows them to operate effectively in complex, open-ended environments with minimal task-specific engineering.

B.2 Monte Carlo Tree Search (MCTS)

Monte Carlo Tree Search (MCTS) (Coulom, 2006) is a sample-based planning algorithm that balances exploration (trying under-sampled actions) and exploitation (refining high-reward actions) through iterative tree search. When integrated with

LLM-based agents, MCTS leverages the language model's prior knowledge to guide efficient exploration in sequential decision making.

Formally, given the MDP setup described above, MCTS incrementally builds a partial search tree rooted at the initial state s_0. Each node in the tree represents a state s, and edges represent actions a. The algorithm maintains empirical statistics for each node-action pair: the visit count $N(s, a)$ and the estimated mean action value $Q(s, a)$. The search process repeats the following four phases until a computational budget is exhausted:

1. Selection. Starting from the root, the algorithm traverses the tree by recursively selecting the optimal child node. This step typically employs the Upper Confidence Bound for Trees (UCT) criterion to handle the exploration-exploitation trade-off:

$$a_t = \operatorname*{argmax}_{a \in \mathcal{A}(s_t)} \left[Q(s_t, a) + c\sqrt{\frac{\ln N(s_t)}{N(s_t, a)}} \right], \tag{10}$$

where $N(s_t) = \sum_a N(s_t, a)$ is the total visit count of the parent node, and $c > 0$ is the exploration constant that controls the degree of exploration.

2. Expansion. Upon reaching a leaf node s_L that is not terminal and has unvisited children, the tree is expanded by instantiating a new child node. In the context of LLMs, this corresponds to sampling a new token or reasoning step a from the policy π and appending the resulting state $s' \sim \mathcal{P}(\cdot \mid s_L, a)$ to the tree.

3. Simulation (Rollout). From the newly expanded node s', a simulation is executed to estimate its potential value. This typically involves generating a full trajectory of length T using a lightweight rollout policy π_{rollout} until a terminal state is reached:

$$\begin{aligned} R_{\text{sim}} &= \sum_{t=0}^{T} \gamma^t \mathcal{R}(s_t, a_t), \\ a_t &\sim \pi_{\text{rollout}}(\cdot \mid s_t). \end{aligned} \tag{11}$$

The cumulative reward R_{sim} serves as a Monte Carlo estimate of the state value $V(s')$. In recent LLM reasoning frameworks, this costly rollout is sometimes substituted by a learned value function or heuristic evaluation.

4. Backpropagation. The estimated value R_{sim} is propagated backward from the leaf s' up to the root. For every edge (s, a) along the traversal path, the statistics are updated via incremental averaging:

$$\begin{aligned} N(s, a) &\leftarrow N(s, a) + 1, \\ Q(s, a) &\leftarrow Q(s, a) + \frac{R_{\text{sim}} - Q(s, a)}{N(s, a)}. \end{aligned} \tag{12}$$

Under the assumption that every action is eventually explored infinitely often, the UCT update guarantees that $Q(s, a)$ converges to the optimal value function, guiding the agent toward the optimal policy.

C Theoretical Analysis

This appendix provides detailed derivations for the theoretical claims used in Section 3.1 and Section 3.2. The layer-wise analysis is local to a fixed transition (x, a), and the dependence on (x, a) is omitted unless otherwise stated. The preference-consistency analysis fixes an agent and a search state, and compares candidate steps generated from that state.

C.1 Setup and Notation

Let $\mathbf{h}^{(l)} \in \mathbb{R}^d$ denote the pooled hidden representation of a candidate step at layer l, where $l = 0, \ldots, L$. The local residual update is

$$\mathbf{u}_l = \mathbf{h}^{(l+1)} - \mathbf{h}^{(l)}, \qquad l = 0, \ldots, L-1. \tag{13}$$

The global residual update is

$$\mathbf{u}_{\text{global}} = \mathbf{h}^{(L)} - \mathbf{h}^{(0)} = \sum_{l=0}^{L-1} \mathbf{u}_l. \tag{14}$$

The residual-coherence term Φ_{coh} in Eq. (2) is based on the accumulated local-to-global agreement $\frac{1}{L}\sum_{l=0}^{L-1} \cos(\mathbf{u}_l, \mathbf{u}_{\text{global}})$.

C.2 Analysis of Assumption 1

Assumption 1 states that, for a fixed candidate step, layer-wise residual updates concentrate near a shared low-dimensional residual subspace, within which a dominant component accumulates consistently across layers. We analyze these two implications separately. The shared subspace yields an approximate low-rank structure, while the stable dominant component yields a positive local-to-global coherence margin.

C.2.1 Low-dimensional residual subspace

A sufficient formalization of the shared-subspace part of Assumption 1 is

$$\mathbf{u}_l = \mathbf{G}\boldsymbol{\zeta}_l + \boldsymbol{\epsilon}_l, \qquad l = 0, \dots, L-1, \tag{15}$$

where $\mathbf{G} \in \mathbb{R}^{d\times k}$ spans a step-local residual subspace, $\boldsymbol{\zeta}_l \in \mathbb{R}^k$ are layer-specific coefficients, $k < \min\{d, L\}$, and $\|\boldsymbol{\epsilon}_l\|_2 \le \delta$.

Let

$$\mathbf{U} = [\mathbf{u}_0, \dots, \mathbf{u}_{L-1}] \in \mathbb{R}^{d\times L} \tag{16}$$

collect the layer-wise residual updates, and define

$$\mathbf{Z} = [\boldsymbol{\zeta}_0, \dots, \boldsymbol{\zeta}_{L-1}] \in \mathbb{R}^{k\times L}, \tag{17}$$

$$\mathbf{E} = [\boldsymbol{\epsilon}_0, \dots, \boldsymbol{\epsilon}_{L-1}] \in \mathbb{R}^{d\times L}. \tag{18}$$

Stacking Eq. (15) over layers gives

$$\mathbf{U} = \mathbf{G}\mathbf{Z} + \mathbf{E}. \tag{19}$$

If $\boldsymbol{\epsilon}_l = \mathbf{0}$ for all l, then $\mathbf{U} = \mathbf{G}\mathbf{Z}$ and

$$\begin{aligned} \operatorname{rank}(\mathbf{U}) &= \operatorname{rank}(\mathbf{G}\mathbf{Z}) \\ &\le \min\{\operatorname{rank}(\mathbf{G}), \operatorname{rank}(\mathbf{Z})\} \\ &\le k. \end{aligned} \tag{20}$$

Thus, in the noiseless case, the layer-wise residual updates lie in a subspace of rank at most k.

With bounded perturbations, $\operatorname{rank}(\mathbf{G}\mathbf{Z}) \le k$. By the variational characterization of singular values,

$$\begin{aligned} \sigma_{k+1}(\mathbf{U}) &= \min_{\operatorname{rank}(\mathbf{A})\le k} \|\mathbf{U} - \mathbf{A}\|_2 \\ &\le \|\mathbf{U} - \mathbf{G}\mathbf{Z}\|_2 = \|\mathbf{E}\|_2. \end{aligned} \tag{21}$$

Since the spectral norm is bounded by the Frobenius norm,

$$\begin{aligned} \|\mathbf{E}\|_2 &\le \|\mathbf{E}\|_F \\ &= \left(\sum_{l=0}^{L-1} \|\boldsymbol{\epsilon}_l\|_2^2\right)^{1/2} \\ &\le \sqrt{L}\,\delta. \end{aligned} \tag{22}$$

Combining Eq. (21) and Eq. (22) gives

$$\sigma_{k+1}(\mathbf{U}) \le \sqrt{L}\,\delta. \tag{23}$$

This shows that the residual-update matrix remains close to a low-rank structure whenever the perturbations around the shared residual subspace are bounded. It formalizes the first part of Assumption 1: layer-wise residual updates are not arbitrary high-dimensional fluctuations, but concentrate near a common low-dimensional structure.

C.2.2 Positive local-to-global coherence margin

We next formalize the stable-accumulation part of Assumption 1. A sufficient condition is that the residual updates contain a shared dominant direction:

$$\mathbf{u}_l = \eta_l \mathbf{g} + \boldsymbol{\epsilon}_l, \qquad l = 0, \dots, L-1, \tag{24}$$

where $\|\mathbf{g}\|_2 = 1$, $0 < \eta_{\min} \le \eta_l \le \eta_{\max}$, and $\|\boldsymbol{\epsilon}_l\|_2 \le \delta$. The dominant direction $\mathbf{g}$ can be viewed as a stable component inside the shared residual subspace.

Define the average global update

$$\bar{\mathbf{u}} = \frac{1}{L}\mathbf{u}_{\text{global}} = \bar{\eta}\mathbf{g} + \bar{\boldsymbol{\epsilon}}, \tag{25}$$

where

$$\bar{\eta} = \frac{1}{L}\sum_{l=0}^{L-1} \eta_l, \qquad \bar{\boldsymbol{\epsilon}} = \frac{1}{L}\sum_{l=0}^{L-1} \boldsymbol{\epsilon}_l. \tag{26}$$

By construction,

$$\eta_{\min} \le \bar{\eta} \le \eta_{\max}, \qquad \|\bar{\boldsymbol{\epsilon}}\|_2 \le \delta. \tag{27}$$

Assume that the stable component dominates the perturbation scale:

$$\eta_{\min}^2 > 2\eta_{\max}\delta + \delta^2. \tag{28}$$

This condition implies $\eta_{\min} > \delta$, so $\mathbf{u}_l$ and $\bar{\mathbf{u}}$ are nonzero and the cosine similarities are well-defined. We show that

$$\frac{1}{L}\sum_{l=0}^{L-1} \cos(\mathbf{u}_l, \mathbf{u}_{\text{global}}) \ge \frac{\eta_{\min}^2 - 2\eta_{\max}\delta - \delta^2}{(\eta_{\max} + \delta)^2} > 0. \tag{29}$$

Because cosine similarity is invariant to positive rescaling of the second argument,

$$\cos(\mathbf{u}_l, \mathbf{u}_{\text{global}}) = \cos(\mathbf{u}_l, \bar{\mathbf{u}}), \tag{30}$$

where $\bar{\mathbf{u}} = \mathbf{u}_{\text{global}}/L$. For each layer l, expand the numerator:

$$\begin{aligned} \langle \mathbf{u}_l, \bar{\mathbf{u}} \rangle &= \langle \eta_l \mathbf{g} + \boldsymbol{\epsilon}_l, \bar{\eta}\mathbf{g} + \bar{\boldsymbol{\epsilon}} \rangle \\ &= \eta_l \bar{\eta} + \eta_l \langle \mathbf{g}, \bar{\boldsymbol{\epsilon}} \rangle \\ &\quad + \bar{\eta}\langle \boldsymbol{\epsilon}_l, \mathbf{g} \rangle + \langle \boldsymbol{\epsilon}_l, \bar{\boldsymbol{\epsilon}} \rangle. \end{aligned} \tag{31}$$

Using Cauchy–Schwarz and Eq. (27),

$$\begin{aligned} \langle \mathbf{u}_l, \bar{\mathbf{u}} \rangle &\ge \eta_l \bar{\eta} - \eta_l \|\bar{\boldsymbol{\epsilon}}\|_2 - \bar{\eta}\|\boldsymbol{\epsilon}_l\|_2 \\ &\quad - \|\boldsymbol{\epsilon}_l\|_2 \|\bar{\boldsymbol{\epsilon}}\|_2 \\ &\ge \eta_{\min}^2 - 2\eta_{\max}\delta - \delta^2. \end{aligned} \tag{32}$$

Under Eq. (28), this lower bound is strictly positive.

The denominator is upper bounded by

$$\begin{aligned}\|\mathbf{u}_l\|_2 &= \|\eta_l \mathbf{g} + \boldsymbol{\epsilon}_l\|_2 \\ &\le \eta_l + \|\boldsymbol{\epsilon}_l\|_2 \le \eta_{\max} + \delta, \qquad (33)\\ \|\bar{\mathbf{u}}\|_2 &= \|\bar{\eta}\mathbf{g} + \bar{\boldsymbol{\epsilon}}\|_2 \\ &\le \bar{\eta} + \|\bar{\boldsymbol{\epsilon}}\|_2 \le \eta_{\max} + \delta. \qquad (34)\end{aligned}$$

Thus,

$$\|\mathbf{u}_l\|_2 \, \|\bar{\mathbf{u}}\|_2 \le (\eta_{\max} + \delta)^2. \qquad (35)$$

Combining Eq. (32) and Eq. (35), for every layer l,

$$\begin{aligned}\cos(\mathbf{u}_l, \mathbf{u}_{\text{global}}) &= \cos(\mathbf{u}_l, \bar{\mathbf{u}}) \\ &\ge \frac{\eta_{\min}^2 - 2\eta_{\max}\delta - \delta^2}{(\eta_{\max} + \delta)^2}. \qquad (36)\end{aligned}$$

Averaging Eq. (36) over layers gives Eq. (29).

This analysis formalizes the second part of Assumption 1: when a stable component dominates the perturbation scale, local residual updates have positive average agreement with the global residual update. This supports the residual-coherence term Φ_{coh} in Eq. (2) and, together with the decisiveness gate Φ_{dec}, the intrinsic stability score S_{ISLE} in Eq. (4).

C.3 Proof of Proposition 1

We provide the detailed derivation of Proposition 1. Fix an agent M_i and a search state x. Consider two candidate steps a and a' generated by M_i at state x. Let

$$\Delta_q(a, a') = q(x, a) - q(x, a') \qquad (37)$$

denote the latent utility gap, and let

$$\Delta_S^{(i)}(a, a') = S_{\text{ISLE}}^{(i)}(x, a) - S_{\text{ISLE}}^{(i)}(x, a') \qquad (38)$$

denote the intrinsic-score gap.

We use the following sufficient formulation of the noisy monotone surrogate condition. For any candidate step b generated by agent M_i at state x,

$$S_{\text{ISLE}}^{(i)}(x, b) = \mu_i(q(x, b)) + \xi_b, \qquad (39)$$

where μ_i is order-preserving with margin $\lambda_i > 0$:

$$\begin{aligned}q(x, a) > q(x, a') \quad \Rightarrow \quad & \mu_i(q(x, a)) - \mu_i(q(x, a')) \\ & \ge \lambda_i \Delta_q(a, a'). \qquad (40)\end{aligned}$$

We further assume that the noise difference

$$Z = \xi_{a'} - \xi_a$$

is mean-zero and τ_i-sub-Gaussian:

$$\mathbb{E}\exp(\nu Z) \le \exp\left(\frac{\tau_i^2 \nu^2}{2}\right), \qquad \forall \nu \in \mathbb{R}. \quad (41)$$

This condition holds, for example, when ξ_a and $\xi_{a'}$ are independent, mean-zero, and σ_i-sub-Gaussian, in which case $\tau_i = \sqrt{2}\sigma_i$.

Assume $\Delta_q(a, a') > 0$. The event $\Delta_S^{(i)}(a, a') \le 0$ implies

$$\begin{aligned}S_{\text{ISLE}}^{(i)}(x, a) &\le S_{\text{ISLE}}^{(i)}(x, a'), \\ \xi_{a'} - \xi_a &\ge \mu_i(q(x, a)) - \mu_i(q(x, a')) \\ &\ge \lambda_i \Delta_q(a, a'). \qquad (42)\end{aligned}$$

Therefore,

$$\Pr\Big(\Delta_S^{(i)}(a, a') \le 0\Big) \le \Pr\big(Z \ge \lambda_i \Delta_q(a, a')\big). \qquad (43)$$

Applying the one-sided sub-Gaussian tail bound gives

$$\Pr\big(Z \ge \lambda_i \Delta_q(a, a')\big) \le \exp\left(-\frac{\lambda_i^2 \Delta_q(a, a')^2}{2\tau_i^2}\right). \qquad (44)$$

Combining Eq. (43) and Eq. (44) yields

$$\Pr\Big(\Delta_S^{(i)}(a, a') \le 0\Big) \le \exp\left(-\frac{\lambda_i^2 \Delta_q(a, a')^2}{2\tau_i^2}\right). \qquad (45)$$

This matches Eq. (5) with

$$\kappa_i = \frac{\lambda_i^2}{2\tau_i^2}. \qquad (46)$$

Under the independent σ_i-sub-Gaussian sufficient condition above, this reduces to

$$\kappa_i = \frac{\lambda_i^2}{4\sigma_i^2}.$$

C.4 Summary

The analysis is aligned with the main text. Assumption 1 implies that layer-wise residual updates remain close to a shared low-dimensional residual subspace and can accumulate through a stable dominant component. The first implication yields the approximate low-rank bound in Eq. (23), while the second yields the positive local-to-global coherence margin in Eq. (29). These consequences justify the residual-coherence term Φ_{coh} used by S_{ISLE}. Proposition 1 further shows that, within each agent, the probability that S_{ISLE} reverses the

order of two candidate steps decays exponentially with their latent utility gap.

These results do not claim that all Transformer computations exactly satisfy Assumption 1. Rather, they identify a structural regime in which layer-wise residual coherence is mathematically meaningful as an intrinsic step-level preference signal for heterogeneous search.

D Main Experimental Details

This section provides implementation details for the main experiments on HotpotQA, WebShop, and Game of 24. These experiments evaluate GLIDE within the MCTS-based heterogeneous agent setting and use the same open-source model pool as SYMPHONY-S for controlled comparison.

D.1 Infrastructure and Model Configuration

Model pool. Following SYMPHONY-S (Zhu et al., 2026), we use open-source models compatible with consumer-grade hardware: Qwen2.5-7B-Instruct-1M (Yang et al., 2024), Mistral-7B-Instruct-v0.3 (Jiang et al., 2023), and Llama-3.1-8B-Instruct (Grattafiori et al., 2024). The same model pool is used for both GLIDE and the reproduced SYMPHONY-S baseline.

Compute environment. All main experiments are conducted on a local server with 3 NVIDIA RTX 4090 GPUs, each with 24GB VRAM. This setup is used to evaluate whether heterogeneous search can achieve competitive performance under accessible compute constraints.

D.2 MCTS and GLIDE Hyperparameters

Unless otherwise stated, all main experiments follow a unified search and generation protocol.

MCTS parameters. The maximum number of MCTS iterations is set to $K = 10$ across HotpotQA, WebShop, and Game of 24. The UCT exploration constant is set to $c = \sqrt{2}$, following common MCTS practice and prior LLM-agent search settings (Zhou et al., 2024; Zhu et al., 2026).

GLIDE parameters. For distributional calibration, the score window and entropy window are both set to 50: $|\mathcal{W}_i^S| = |\mathcal{W}_i^H| = 50$. For adaptive branching, we set $n_{\min} = 1$ and $n_{\max} = 4$ in Eq. (9). The upper bound $n_{\max} = 4$ matches the fixed expansion width used by SYMPHONY, keeping the maximum branching budget comparable.

Generation parameters. We use temperature 0.2, nucleus sampling with top-$p = 0.9$, and a maximum length of 200 tokens for each atomic reasoning step.

D.3 Task-Specific Prompting and Settings

HotpotQA. HotpotQA is a multi-hop question answering benchmark requiring evidence aggregation and logical synthesis. We use a 3-shot Chain-of-Thought prompting format, where demonstrations guide the model to decompose complex questions into intermediate reasoning steps.

WebShop. WebShop is a simulated e-commerce environment requiring goal-directed browser interactions. We use a 1-shot interaction prompt to demonstrate the valid action space, including search, click, and purchase actions.

Game of 24. Game of 24 is a symbolic arithmetic task where the model must combine four numbers to reach 24. We use a 2-shot prompting format and instruct the model to propose one atomic arithmetic operation at a time, providing explicit state transitions for MCTS expansion.

D.4 Main Benchmark Baselines

To rigorously evaluate the effectiveness of GLIDE, we compare it with a broad set of representative baselines across the three main benchmarks. These baselines cover five methodological paradigms, reflecting different reasoning structures, interaction modes, and supervision requirements.

Linear and Standard Prompting. This category represents the foundational capabilities of LLMs without explicit search or multi-agent control.

- *Standard Prompting (IO) & Single LLM:* These baselines use direct input-output generation with vanilla prompts, serving as basic reference points for evaluating the benefit of structured search and heterogeneous agent control.

- *Chain-of-Thought (CoT) (Wei et al., 2022):* CoT prompts the model to generate intermediate reasoning steps before producing the final answer, decomposing complex problems into linear reasoning traces.

- *Self-Consistency (CoT-SC) (Wang et al., 2023):* CoT-SC samples multiple reasoning paths and aggregates them through majority

Table 7: **Test-time compute scaling on HotpotQA.** We vary the MCTS budget K and report per-question averages. Results for ToT, RAP, and LATS are taken from LATS (Zhou et al., 2024), while SYMPHONY results are taken from SYMPHONY (Zhu et al., 2026). GLIDE maintains strong performance while using fewer nodes and substantially fewer tokens than verifier-based search baselines at comparable or larger search budgets.

Budget / Method	Exact Match ↑	# Nodes ↓	Token Consumption ↓
$K = 10$			
ToT (Yao et al., 2023a)	0.34	33.97	–
RAP (Hao et al., 2023)	0.44	31.53	–
LATS (Zhou et al., 2024)	0.44	28.42	–
SYMPHONY-S (Zhu et al., 2026)	0.59	16.39	–
SYMPHONY-L (Zhu et al., 2026)	**0.79**	**9.47**	7,906
GLIDE	0.73	15.74	**6,052**
$K = 30$			
ToT (Yao et al., 2023a)	0.39	47.54	–
RAP (Hao et al., 2023)	0.50	37.71	–
LATS (Zhou et al., 2024)	0.52	34.12	–
GLIDE	**0.76**	**26.21**	**9,872**
$K = 50$			
ToT (Yao et al., 2023a)	0.49	84.05	210,215
RAP (Hao et al., 2023)	0.54	70.60	176,500
LATS (Zhou et al., 2024)	0.61	66.65	173,290
GLIDE	**0.78**	**54.10**	**31,504**

voting. It provides a strong baseline for evaluating the effect of sampling-based consensus without explicit tree search or agent-level calibration.

Feedback-Driven and Interactive Reasoning. These methods extend linear generation by incorporating interaction, critique, or iterative refinement.

- *ReAct (Yao et al., 2023b):* ReAct interleaves reasoning and acting, allowing the model to produce thought-action trajectories and interact with external environments. This design helps reduce hallucination by grounding reasoning in observations.

- *Reflexion (Shinn et al., 2023):* Reflexion uses verbal reinforcement feedback to support iterative self-correction. It stores reflections over previous failures and uses them to improve subsequent trials without updating model parameters.

Structured Search and Planning. These approaches formulate reasoning as state-space search, using tree, graph, beam, or MCTS structures to explore multiple solution paths.

- *Tree, Graph, and Beam Search:* Methods such as ToT (Yao et al., 2023a), GoT (Besta et al., 2024), MDToC (Ta and Oates, 2025), and AgentKit (Wu et al., 2024) generalize linear reasoning chains into non-linear search structures, enabling multiple candidate branches to be explored and compared. For retrieval-intensive reasoning, we also include Beam Retrieval (Zhang et al., 2023), which applies beam-search-style selection to iteratively rank and select evidence chains.

- *MCTS-based Agents:* RAP (Hao et al., 2023) and LATS (Zhou et al., 2024) integrate Monte Carlo Tree Search with LLM agents. LATS, in particular, unifies reasoning, acting, and planning by using an external LLM-as-a-Judge to estimate state values and guide search toward high-reward trajectories.

Multi-Agent Frameworks. These systems leverage multiple LLM agents to diversify reasoning trajectories and reduce single-model bias.

- *MoA (Wang et al., 2024a):* MoA aggregates responses from multiple agents through layered collaboration, aiming to improve answer quality via multi-agent response refinement.

- *MASTER (Gan et al., 2025):* MASTER is a multi-agent MCTS framework that combines agent collaboration with tree-structured search, using multi-agent interactions to guide reasoning and decision making.

- *SYMPHONY (Zhu et al., 2026):* SYMPHONY optimizes query dispatching across a heterogeneous model pool. We compare with

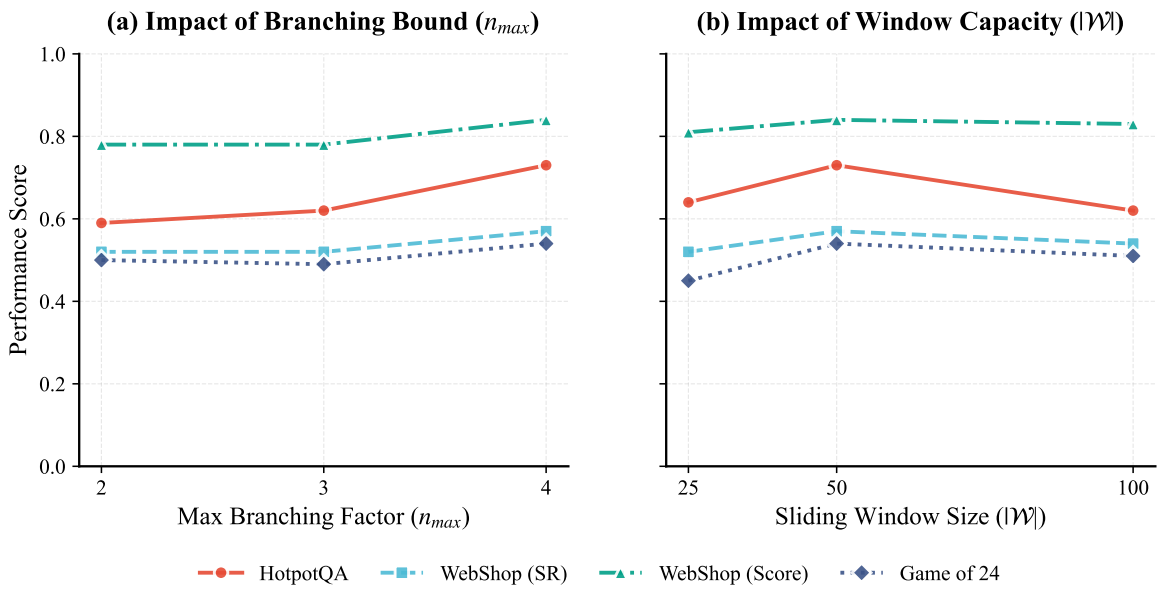

Figure 5: **Hyperparameter Sensitivity Analysis.** Performance trends across three datasets (four metrics) under varying configurations. **(a)** Impact of the search branching bound $n_{\max}$. The results show that performance remains generally stable across different branching bounds, with moderate gains obtained when allowing a slightly larger search space. **(b)** Impact of the sliding window capacity $|\mathcal{W}|$. The sliding window balances statistical stability and adaptation to local distribution shifts. The results show a broad performance plateau around $|\mathcal{W}| = 50$, indicating that this default setting robustly covers typical context shifts without task-specific tuning.

two variants when available: SYMPHONY-L, which uses a high-cost pool including proprietary or high-capacity models, and SYMPHONY-S, which uses open-source models. SYMPHONY-S serves as the primary controlled baseline because it uses the same open-weight model pool as GLIDE.

Task-Specific Methods for WebShop. For WebShop, we additionally include task-native and supervised methods. These include Imitation Learning (IL), IL+RL, and Human Expert baselines from the original WebShop benchmark (Yao et al., 2022), as well as WebGUM (Furuta et al., 2024), a multimodal agent fine-tuned on web navigation and instruction-following data.

Result Sources and Backbone Fairness. To ensure transparent comparison, we follow the reporting protocols and backbone settings of prior work, while using SYMPHONY-S as the primary controlled baseline for GLIDE.

- *HotpotQA.* Baselines are based on the GPT-4 backbone. Except for SYMPHONY, reported results are cited from Gan et al. (2025).
- *WebShop.* Most reported baselines use the GPT-4 backbone. SYMPHONY results are cited from the original SYMPHONY paper, while other GPT-4-based baselines are adopted from Gan et al. (2025).
- *Game of 24.* Baselines predominantly use the GPT-3.5-Turbo backbone. Single-LLM results are cited from Hao et al. (2024), GoT and MDToC results from Ta and Oates (2025), and the remaining baselines from Zhou et al. (2024).
- *Controlled comparison.* SYMPHONY-S is reproduced with the same open-weight model pool and comparable search budget as GLIDE, making it the primary controlled baseline across all three main benchmarks.

E Supplementary Analyses for Main Experiments

This section provides additional analyses for the main experiments on HotpotQA, WebShop, and Game of 24. These analyses further examine the efficiency, parameter behavior, and step-level preference quality of GLIDE under the same MCTS-based heterogeneous agent setting used in the main benchmark results.

E.1 Test-Time Compute Scaling

A major limitation of tree-structured reasoning frameworks is the inference overhead introduced by expanding and evaluating many candidate nodes. We evaluate the test-time compute efficiency of GLIDE on HotpotQA by scaling the MCTS budget from $K = 10$ to $K = 50$. For each setting, we report Exact Match, the average number of expanded nodes, and average token consumption per question. Following prior reporting protocols, the results of ToT, RAP, and LATS are taken from LATS (Zhou et al., 2024), and the SYMPHONY results are taken from SYMPHONY (Zhu et al., 2026).

Many structured reasoning methods follow a generate-then-verify pipeline, where node expansion is coupled with additional verifier or LLM-as-a-Judge calls. This design increases both token consumption and latency, especially when the search budget grows. In contrast, GLIDE computes its value signal from hidden states and predictive distributions already produced during generation, avoiding additional verifier-generated tokens.

Table 7 shows that GLIDE scales favorably with test-time compute. At $K = 10$, GLIDE substantially improves over SYMPHONY-S while using fewer tokens than SYMPHONY-L. When the budget is increased to $K = 50$, GLIDE outperforms ToT, RAP, and LATS while using far fewer tokens. These results indicate that verifier-free intrinsic evaluation and adaptive branching provide a more cost-effective scaling path than external-judge-based search.

E.2 Algorithmic Parameter Analysis

We analyze the main hyperparameters governing GLIDE's inference behavior. The analysis focuses on two aspects: the search budget that controls MCTS complexity, and the online distributional horizon used for calibration.

E.2.1 Search Budget and Adaptive Branching

In MCTS, inference cost is primarily controlled by the number of search iterations and the branching width. High-resource baselines such as RAP (Hao et al., 2023) and LATS (Zhou et al., 2024) often rely on larger search budgets to improve exploration and reduce value-estimation variance, but this brute-force scaling strategy is costly. GLIDE instead uses a constrained search budget and allocates branching adaptively according to predictive uncertainty and calibrated reward saturation.

Trajectory budget. The maximum number of MCTS iterations is set to $K = 10$ across HotpotQA, WebShop, and Game of 24. This budget follows the efficient SYMPHONY-S setting and keeps inference cost controlled across all tasks. Under this constraint, performance improvements must come from more effective candidate prioritization and search control rather than from simply increasing the number of simulations.

Branching width. The branching width determines how many candidate continuations are expanded from a search state. Standard fixed-width methods allocate the same expansion budget to all states, regardless of whether the current state is ambiguous, already promising, or unlikely to benefit from further exploration. In GLIDE, the branching width $n(x, a)$ is adapted using Eq. (9). We set $n_{\min} = 1$ and $n_{\max} = 4$, where $n_{\max} = 4$ matches the fixed expansion width used by SYMPHONY. Thus, GLIDE never exceeds the maximum branching width of the controlled baseline, while its average branching width can be lower because easy or saturated states receive fewer expansions. Figure 5(a) further analyzes the effect of different $n_{\max}$ settings.

E.2.2 Calibration Horizon and Stability

The sliding windows $\mathcal{W}_i^S$ and $\mathcal{W}_i^H$ define the online horizon for distributional calibration. We set both window sizes to 50:

$$|\mathcal{W}_i^S| = |\mathcal{W}_i^H| = 50.$$

The score window $\mathcal{W}_i^S$ is used for the distributional rank score in Eq. (6), while the entropy window $\mathcal{W}_i^H$ is used for relative uncertainty estimation in Eq. (8).

These windows serve as online, agent-specific calibrators rather than static storage buffers. Unlike offline normalization methods that require pre-computed statistics on a held-out set, GLIDE builds each agent's local score and entropy profiles during inference. This design is important in multi-step reasoning, where the distribution of generated tokens and intrinsic scores can shift as context length and task difficulty evolve.

The window also provides a forgetting mechanism. A very small window reacts quickly but can be sensitive to transient outliers, causing unstable rank and entropy estimates. A very large window provides smoother statistics but may introduce distributional lag, making the calibration slow to adapt

Table 8: **MCTS ablation with different evaluation baselines.** All variants are evaluated under the same fixed MCTS framework. Hot denotes HotpotQA EM, Web denotes WebShop SR, and G24 denotes Game of 24 success rate.

Variant	Hot	Web	G24
Perplexity	0.61	0.37	0.18
Energy	0.54	0.38	0.53
LN-Entropy	0.63	0.31	0.45
CoE-R	0.61	0.47	0.20
CoE-C	0.71	0.45	0.51
GenPRM-7B	0.61	0.34	0.47
Rand. Agent	0.59	0.56	0.43
GLIDE	**0.73**	**0.57**	**0.54**

to changes in reasoning depth or local task complexity. Figure 5(b) shows that a window size of 50 provides a stable trade-off between sensitivity and robustness.

E.3 MCTS Evaluation Baselines under Fixed Search

As an extension of the ablation study in Table 6, we evaluate different scoring functions under the same fixed MCTS framework. This experiment isolates the effect of the evaluation signal while keeping the search procedure unchanged. The compared baselines include probability-based confidence metrics, hidden-state-based metrics, supervised process reward models, and random heterogeneous-agent evaluation.

Table 8 shows that GLIDE achieves the best performance across all three domains. Compared with probability-based metrics such as perplexity, energy, and length-normalized entropy, GLIDE provides a more effective search-control signal. It also outperforms CoE variants and GenPRM-7B, suggesting that the proposed combination of S_{ISLE} and distributional calibration is better suited for heterogeneous MCTS selection.

E.4 Step-Level Preference and Efficiency Analysis

The step-level preference analysis in Section 4.3 is derived from the search traces of the main MCTS experiments. For each decision point, we collect the expanded child nodes and evaluate whether different scoring signals rank successful continuations ahead of unsuccessful ones. A child node is labeled according to the final environment outcome reached from that node. This protocol tests whether a scoring method provides a useful preference signal for MCTS selection, rather than whether it certifies the correctness of an isolated reasoning step.

E.4.1 Scoring Baselines

We compare R_{GLIDE} with probability-based intrinsic metrics, hidden-state-based metrics, uncalibrated intrinsic scores, heterogeneous judge scores, and process reward models. Unless otherwise specified, all baselines are converted to a common higher-is-better preference orientation before ranking child nodes.

Intrinsic uncertainty metrics. These baselines are training-free confidence metrics computed from token probabilities or logits during generation.

- *Perplexity (Liu et al., 2020).* Perplexity summarizes sequence-level uncertainty from the likelihood assigned to the generated continuation. Since lower perplexity indicates higher model confidence, we use its negative value as the ranking score.

- *Length-normalized entropy (Malinin and Gales, 2020).* Length-normalized entropy averages the Shannon entropy of the predictive distribution over decoding steps. Lower entropy indicates a more concentrated next-token distribution, so we use its negative value as the ranking score.

- *Energy score (Huang et al., 2023).* The energy score is computed from unnormalized model logits and aggregated over the generated continuation. We choose the sign convention so that larger values correspond to stronger confidence, making it comparable with other preference scores.

Hidden-state-based metrics. These baselines use internal representations rather than only token probabilities.

- *Chain-of-Embedding Metrics (CoE) (Wang et al., 2025b).* CoE measures hidden-state trajectories across layers and uses their magnitude and angular changes as indicators of generation reliability. We compare with both CoE-R and CoE-C variants following the original implementation. CoE-R combines normalized magnitude and angle changes in real space, while CoE-C maps layer-wise changes into a complex-plane representation to capture their joint evolution.

- *CoT-Kinetics (Bi et al., 2025).* CoT-Kinetics measures reasoning reliability by formulating the layer-wise evolution of reasoning-token hidden states as a kinetic process. It computes a scalar energy score from semantic momentum, semantic curvature, and output uncertainty, thereby assessing the soundness of the reasoning trajectory using only internal model states. We do not include CoT-Kinetics in the MCTS-based experiments, since it is primarily designed for long chain-of-thought reasoning in LRMs, whereas our method focuses on step-level decision control.

Heterogeneous and reward-model baselines. These baselines evaluate whether performance gains come from agent heterogeneity, calibrated intrinsic scoring, or supervised process supervision.

- *Raw S_{ISLE}.* This baseline directly uses the uncalibrated intrinsic stability score from each generating agent. It tests whether agent-wise distributional calibration is necessary.
- *Rank-only score.* This baseline uses only $S^{(i)}_{\text{rank}}$ from Eq. (6), discarding the absolute intrinsic score. It evaluates whether relative standing alone is sufficient for robust search control.
- *Hetero-Raw.* This control uses heterogeneous agent outputs without calibrated intrinsic reward. It tests whether gains arise merely from access to multiple models rather than from GLIDE's distributional calibration.
- *GenPRM-7B (Zhao et al., 2025).* GenPRM-7B is a process reward model designed to provide granular step-level feedback.
- *PRM-RLHFlow (Xiong et al., 2024).* We use the Llama3.1-8B-PRM-Deepseek-Data variant, which leverages preference data for process-level verification.
- *PRM-Skywork (He et al., 2024).* We use the Skywork-o1-Open-PRM-Qwen-2.5-1.5B variant, which provides a lightweight process reward model for step evaluation.

Unlike PRM-based baselines, GLIDE does not require task-specific process supervision or additional verifier forward passes. Its calibrated reward R_{GLIDE} is computed from the same generation pass used to produce the candidate continuation.

E.4.2 Ranking Metrics

We report P@1, MRR, and NDCG@3 for step-level preference evaluation. Let $\mathcal{Q}$ denote the set of evaluated selection points. For a selection point $q \in \mathcal{Q}$, let $\pi_q(k)$ be the child node ranked at position k by a scoring method, and let $g_q(v) \in \{0, 1\}$ indicate whether child node v eventually reaches a successful final outcome.

Precision at rank 1 is

$$\text{P@1} = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} g_q(\pi_q(1)). \tag{47}$$

MRR measures how early the first successful child appears:

$$\text{MRR} = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \frac{1}{r_q}. \tag{48}$$

$$r_q = \min\{k : g_q(\pi_q(k)) = 1\}. \tag{49}$$

If no successful child exists for a selection point, the reciprocal-rank contribution is set to zero. NDCG@3 measures the quality of the top-ranked child nodes:

$$\text{NDCG@3} = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \frac{\text{DCG@3}_q}{\text{IDCG@3}_q}, \tag{50}$$

where

$$\text{DCG@3}_q = \sum_{k=1}^{3} \frac{g_q(\pi_q(k))}{\log_2(k+1)}. \tag{51}$$

We also report inference latency and peak GPU memory to quantify the efficiency of each scoring method.

E.5 Failure Case Analysis

We further analyze cases where GLIDE assigns high scores to incorrect steps. The most common failure pattern is a *confident semantic mismatch*: the generated step is internally coherent and lexically well supported by the current context, but it fails to satisfy an external task constraint. In such cases, the model's layer-wise residual evolution can remain stable, causing S_{ISLE} to assign a high intrinsic score even though the step is ultimately wrong.

Table 9 shows a representative example from WebShop. Given the instruction "vegetarian smoked peppered bacon," the agent clicked a product titled "Smoked Bacon Sea Salt 3-Pack." The step matches salient surface cues such as "smoked,"

Table 9: **Representative failure case.** GLIDE can overscore internally coherent but externally mismatched steps.

Instruction	High-scoring step	Failure mode
Vegetarian smoked peppered bacon	Click "Smoked Bacon Sea Salt 3-Pack"	Strong lexical overlap makes the step internally coherent, but the product is not a vegetarian bacon substitute.

"bacon," and related product descriptors, leading to a confident and coherent internal trajectory. However, the clicked item is a seasoning product rather than a vegetarian bacon substitute, making the action semantically incorrect.

This failure mode highlights a natural boundary of intrinsic evaluation. S_{ISLE} measures whether the model's internal computation evolves coherently within the current generation step; it does not directly verify external semantic grounding or task-specific constraints. Therefore, high intrinsic coherence should be interpreted as a preference signal for search control rather than a certificate of correctness.

In practice, the MCTS framework can partially mitigate such errors. If later observations expose the mismatch, subsequent node evaluations may decrease and alternative branches can still be explored. Nevertheless, recovery is not guaranteed when the incorrect branch remains superficially plausible or the search budget is limited. This suggests a promising direction for future work: combining calibrated intrinsic evidence with lightweight grounding checks or environment-aware constraint signals, while preserving the low-overhead advantages of GLIDE.

F Cross-Task Outcome Separability

This section provides details for the auxiliary cross-task outcome-separability analysis in Section 4.5. Unlike the main search experiments, this evaluation does not use search traces, tree expansion, or adaptive branching. Each model directly generates one solution for each input, and the solution is labeled by its final task outcome. The goal is to test whether S_{ISLE} remains associated with final outcomes under direct generation, outside the main MCTS control pipeline.

F.1 Datasets and Models

We evaluate outcome separability across three complementary settings: multilingual mathematical reasoning, scientific knowledge reasoning, and code generation.

Multilingual mathematical reasoning. MGSM (Shi et al., 2023) is a subset of GSM8K with 11 language versions. We evaluate all languages: English, Bengali, German, Spanish, French, Japanese, Russian, Swahili, Telugu, Thai, and Chinese. Unless otherwise specified, MGSM results report the average over these 11 languages. Averaging across languages helps reduce dependence on language-specific surface forms, testing whether the scoring signal reflects model-internal outcome evidence rather than superficial linguistic variation.

Scientific knowledge reasoning. GPQA Diamond (Rein et al., 2023) is a PhD-level multiple-choice question answering benchmark covering physics, chemistry, and biology. It evaluates outcome separability under specialized domain knowledge and difficult scientific reasoning.

Code generation. HumanEval (Chen et al., 2021) consists of programming problems paired with unit tests. A generated solution is labeled successful if it passes the corresponding tests, allowing us to evaluate outcome separability for executable program synthesis.

We use open-source LLMs spanning different model families and parameter scales: Mistral-7B-Instruct-v0.3 (Jiang et al., 2023), Gemma-3-12B-IT (Team, 2025), Phi-3-medium-128k-instruct (Microsoft, 2024), InternLM2-Chat-20B (Cai et al., 2024), and Qwen2.5-32B-Instruct (Yang et al., 2024). This model set allows us to test whether the scoring signal remains outcome-relevant across architectures, training recipes, and model sizes.

F.2 Generation Protocol

We use deterministic greedy decoding with `do_sample=False`. The maximum number of new tokens is set to 512 for MGSM and GPQA, and 1024 for HumanEval. This protocol is designed to test outcome separability under a bounded single-pass generation setting, rather than to improve accuracy through additional test-time scaling such as repeated sampling, self-consistency, or search.

This design is consistent with the role of S_{ISLE} in the main method. GLIDE uses S_{ISLE} as a step-level preference signal during sequential search, where the goal is to rank candidate continuations rather than to elicit long free-form chains. Accordingly, this auxiliary experiment uses direct generations with moderate token limits, allowing the model to produce a complete answer while keeping the evaluation focused on the intrinsic score's outcome relevance. The larger budget for HumanEval accounts for full Python implementations, whereas MGSM and GPQA require only concise reasoning and final-answer extraction.

The task-specific prompts are shown in Figure 16. They standardize the final-answer format for MGSM and GPQA and enforce executable Python output for HumanEval, enabling consistent outcome labeling across tasks.

F.3 Outcome-Separability Results

We compare S_{ISLE} with the scoring baselines described in Appendix E.4.1, including probability-based confidence metrics, hidden-state-based metrics, and output-based confidence baselines when applicable. Unlike the step-level analysis in Section 4.3, the evaluation unit here is a complete generated solution rather than an expanded child node in the search tree.

For each model–dataset pair, each generated solution is labeled as successful or failed according to the final task outcome. A scoring method is effective if it assigns higher scores to successful generations than to failed ones. We report AUROC, AUPR, and FPR95. AUROC measures the probability that a randomly selected successful generation receives a higher score than a randomly selected failed generation. AUPR summarizes the precision–recall trade-off and is especially informative when the numbers of successful and failed samples are imbalanced. FPR95 reports the false-positive rate when the true-positive recall is fixed at 95%, measuring how many failed generations are incorrectly accepted when most successful generations are retained. Thus, higher AUROC and AUPR, and lower FPR95, indicate stronger outcome separability.

Full model-wise and dataset-wise results are reported in Table 10. For MGSM, the table reports the average over 11 language versions, while language-level results for each model are visualized in Figures 6–10. Across datasets and model families, S_{ISLE} provides consistent separation between successful and failed generations, supporting the outcome relevance of the intrinsic signal outside the main MCTS control pipeline.

F.4 Layer-Subset Analysis

The default computation of S_{ISLE} uses all Transformer layers. This choice follows directly from the theoretical formulation in Section C, where the score compares local residual updates with the global step update $\mathbf{u}_{\text{global}} = \mathbf{h}^{(L)} - \mathbf{h}^{(0)}$ accumulated across the full network depth. Using only a subset of layers weakens this cumulative-trajectory interpretation and introduces a large, hard-to-interpret combinatorial design space.

To assess whether partial trajectories still retain useful outcome-related evidence, we conduct an additional layer-subset analysis on MGSM. Building on the same outcome-separability protocol, we evaluate four deterministic layer subsets: First 50%, Last 50%, Odd, and Even layers. For each subset, S_{ISLE} is recomputed using only the selected layer transitions, while the generation outputs and success/failure labels remain unchanged. This isolates the effect of layer coverage from generation quality.

The results show that partial layer subsets can still capture useful residual-coherence signals, remaining competitive with several scoring baselines in the full outcome-separability analysis. However, the all-layer configuration remains the default because it best matches the cumulative residual-stream assumption underlying S_{ISLE}. Language-level MGSM results for the layer-subset analysis are visualized for each model in Figures 11–15.

F.5 Memory Overhead of Hidden-State Scoring

A potential concern is that computing layer-wise hidden-state dynamics may require storing high-dimensional hidden states across layers, candidate branches, or search trajectories, thereby introducing substantial memory overhead. In GLIDE, this is not the case. The intrinsic score is computed on the fly during the generation forward pass: hidden states are extracted for the current candidate, pooled across generated tokens, used to compute the layer-wise residual updates and the corresponding S_{ISLE} score, and then immediately discarded. The search tree stores only scalar quantities such as S_{ISLE} and R_{GLIDE}, rather than full hidden-state trajectories.

To empirically verify this, we measure the peak VRAM increase during the expansion phase on GPQA Diamond. For each model, we compare

Table 10: **Cross-task outcome separability across models and datasets.** Each cell reports AUROC/AUPR/FPR95, with higher AUROC/AUPR and lower FPR95 indicating stronger separability. MGSM results are averaged over 11 language versions; language-level results for each model are shown in Figures 6–10. Metric-wise best and second-best values are highlighted per model–dataset column using Best and Second Best; the Average column uses Best and Second Best.

Method	Mistral-7B Inst. v0.3	Gemma-3 12B IT	Phi-3 Medium 128k	InternLM2 Chat 20B	Qwen2.5 32B Inst.	Average
MGSM (AUROC↑/AUPR↑/FPR95↓)						
Perplexity	65.55/38.72/74.98	85.30/94.67/**33.53**	57.06/63.00/86.18	46.38/42.98/93.10	65.93/91.28/81.06	64.04/66.13/73.77
Energy	31.37/21.48/98.49	77.05/92.26/66.68	38.41/55.10/96.85	43.62/40.42/95.17	39.03/80.91/94.61	45.90/58.03/90.36
LN-Entropy	67.66/39.46/74.03	**85.61**/**95.05**/34.76	58.96/63.55/85.24	46.20/42.29/92.73	65.80/**92.03**/**72.57**	64.85/66.48/**71.87**
CoE-R (Wang et al., 2025b)	67.31/36.28/73.48	26.42/73.83/97.58	58.59/63.37/88.32	**50.39**/**46.07**/92.67	62.65/89.80/84.42	53.07/61.87/87.29
CoE-C (Wang et al., 2025b)	63.95/36.46/77.93	32.93/77.51/97.58	58.31/63.11/88.50	49.52/44.37/93.89	63.08/89.81/84.11	53.56/62.25/88.40
CoT-Kinetics	68.76/**41.20**/71.87	30.86/76.26/97.58	**61.54**/**64.59**/**85.02**	48.57/42.65/92.96	65.92/90.28/75.99	55.13/63.00/84.68
ISLE	**69.17**/40.83/**71.66**	85.45/94.83/35.48	59.92/63.67/85.68	48.53/43.51/**92.66**	**65.99**/89.77/79.60	**65.81**/**66.52**/73.02
GPQA Diamond (AUROC↑/AUPR↑/FPR95↓)						
Perplexity	52.10/12.40/89.58	53.15/20.29/96.86	62.24/78.88/**94.44**	52.46/64.77/90.41	**76.73**/89.66/**73.91**	59.34/53.20/89.04
Energy	40.24/7.92/95.59	61.84/30.67/76.10	55.31/77.02/97.22	44.44/60.37/100.00	33.25/64.47/100.00	47.02/48.09/93.78
LN-Entropy	53.36/12.17/91.32	51.80/19.55/97.48	62.58/78.50/**94.44**	54.07/66.05/90.41	76.57/**89.86**/82.61	59.68/53.23/91.25
CoE-R (Wang et al., 2025b)	**62.16**/13.91/95.04	67.68/29.15/73.58	55.86/78.19/97.22	55.89/67.59/97.26	60.87/76.74/82.61	60.49/53.12/89.14
CoE-C (Wang et al., 2025b)	60.60/**14.79**/90.51	**69.78**/**31.45**/**67.92**	57.72/78.88/97.22	**64.21**/**74.69**/90.41	58.70/75.34/82.61	62.20/55.03/85.73
CoT-Kinetics	53.64/13.24/91.32	69.15/30.54/75.47	**66.97**/**85.43**/**94.44**	60.03/72.60/**87.67**	75.28/87.27/78.26	**65.01**/**57.82**/**85.43**
ISLE	54.46/12.68/**87.65**	65.30/29.32/78.74	62.52/78.73/**94.44**	61.12/73.45/91.78	74.32/88.91/91.30	63.54/56.62/88.78
HumanEval (AUROC↑/AUPR↑/FPR95↓)						
Perplexity	66.75/48.33/87.78	47.64/70.54/**80.56**	54.81/18.45/88.72	57.57/65.13/**89.86**	56.01/78.53/93.02	56.56/56.20/87.99
Energy	32.63/19.47/94.87	47.82/74.81/88.89	51.90/19.30/89.47	54.92/58.10/**89.86**	45.94/69.14/88.37	46.64/48.16/90.29
LN-Entropy	67.68/48.96/81.79	47.04/**77.92**/86.11	50.46/16.04/90.23	57.60/64.60/91.30	54.85/76.45/90.70	55.53/56.79/88.03
CoE-R (Wang et al., 2025b)	61.64/39.70/88.89	38.02/70.57/97.22	53.30/17.80/100.00	53.34/62.24/97.10	45.08/70.86/93.02	50.28/52.23/95.25
CoE-C (Wang et al., 2025b)	67.31/45.02/94.87	38.22/70.80/97.22	53.14/17.81/99.25	55.67/65.48/95.65	46.46/71.64/90.70	52.16/54.15/95.54
CoT-Kinetics	**77.74**/48.88/**71.79**	39.02/71.17/97.22	51.10/17.48/95.49	60.16/66.87/92.75	47.33/73.21/88.37	55.07/55.52/89.12
ISLE	75.80/**49.84**/79.49	**47.87**/74.33/88.89	**57.41**/**20.28**/**87.22**	**61.99**/**69.44**/95.65	**57.28**/**79.84**/**83.72**	**60.07**/**58.75**/**86.99**

MGSM reports the average over 11 language versions: en, bn, de, es, fr, ja, ru, sw, te, th, and zh. The "Average" column reports the arithmetic mean over the five listed models.

Table 11: **Peak VRAM overhead of GLIDE hidden-state scoring.** We report the relative peak memory increase during expansion on GPQA Diamond.

Model	Peak Memory Increase (%)
Llama-3.1-8B-Instruct	+1.06
Llama-3.3-70B-Instruct	+2.39

peak memory usage with and without GLIDE's hidden-state scoring under the same generation setting. As shown in Table 11, the additional memory footprint is negligible, even when scaling from an 8B model to a 70B model. This confirms that GLIDE's intrinsic evaluation introduces little memory overhead in practice.

G Use of AI Assistants

We utilized AI assistants to help with language editing and writing refinement. All technical content, experimental results, and scientific claims were verified by the authors.

H Artifacts Statements

H.1 Model Artifacts

All model artifacts are publicly available and used from official open-source releases or public repositories in compliance with their licenses, model cards, and usage terms. We use them only for research evaluation and analysis, without redistributing modified checkpoints; configuration details are provided in Section F.1, E.4.1, and 4.1.

H.2 Data Artifacts

We employ publicly available benchmarks and task environments for evaluation, including HotpotQA, WebShop, Game of 24, MGSM, GPQA, and HumanEval. These benchmarks are widely used in the open research community. We use them strictly for non-commercial research purposes and comply with their respective licenses, terms of use, and citation requirements.

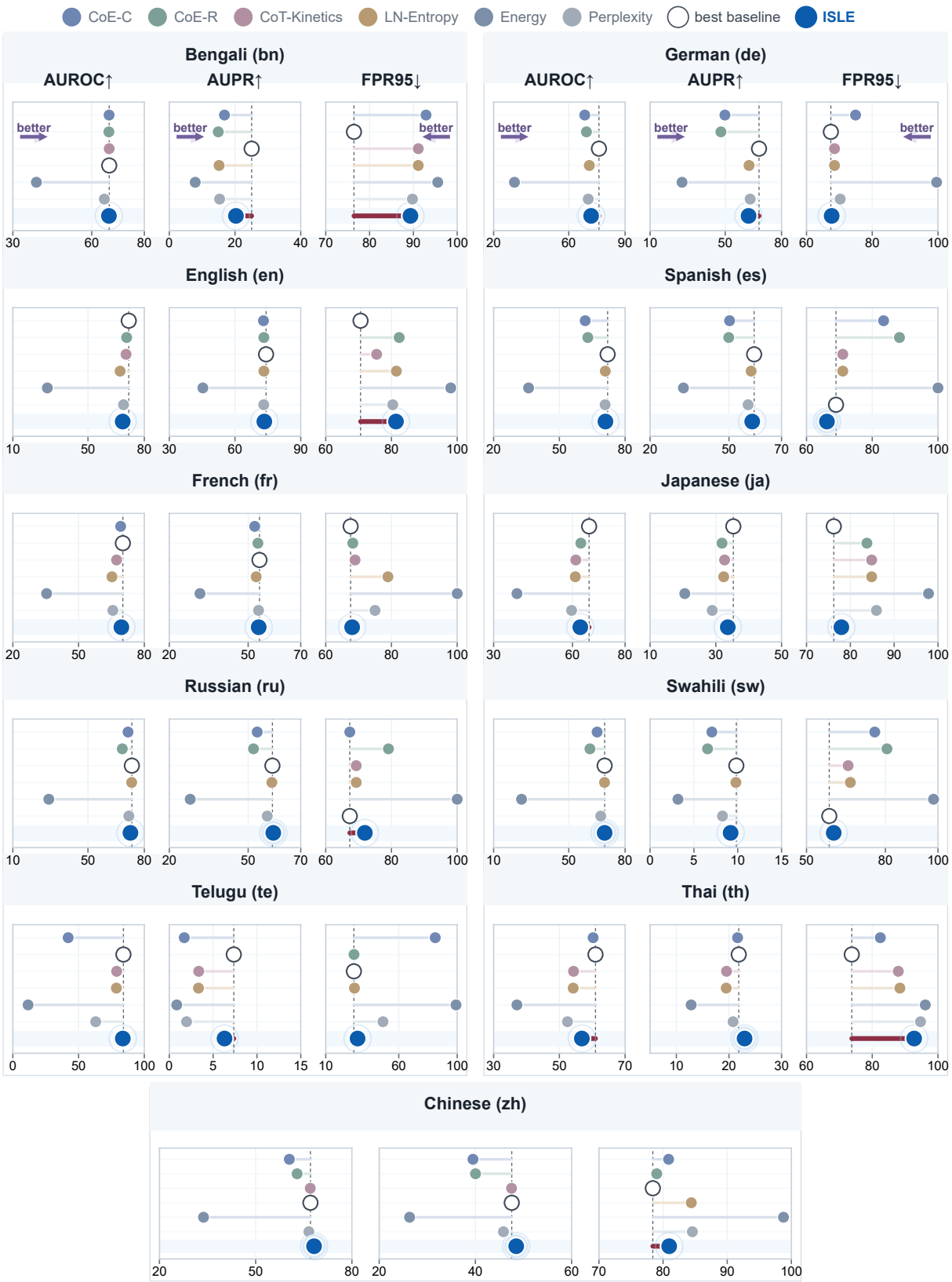

Figure 6: **Language-level MGSM outcome separability for Mistral-7B-Instruct-v0.3.** The figure shows the performance of different scoring methods across the 11 MGSM language versions. Higher AUROC/AUPR and lower FPR95 indicate better separation between successful and failed generations.

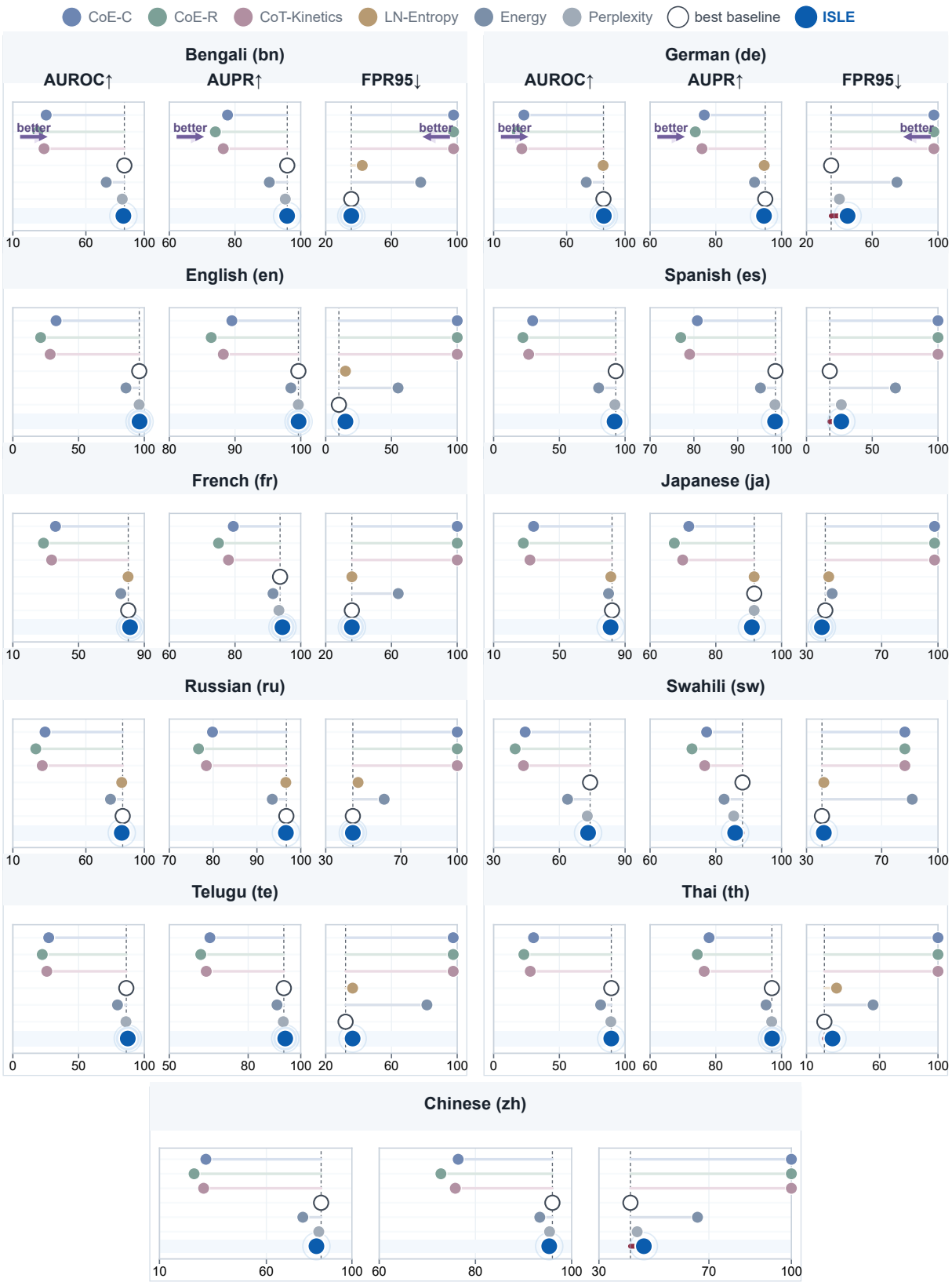

Figure 7: **Language-level MGSM outcome separability for Gemma-3-12B-IT.** The figure shows the performance of different scoring methods across the 11 MGSM language versions. Higher AUROC/AUPR and lower FPR95 indicate better separation between successful and failed generations.

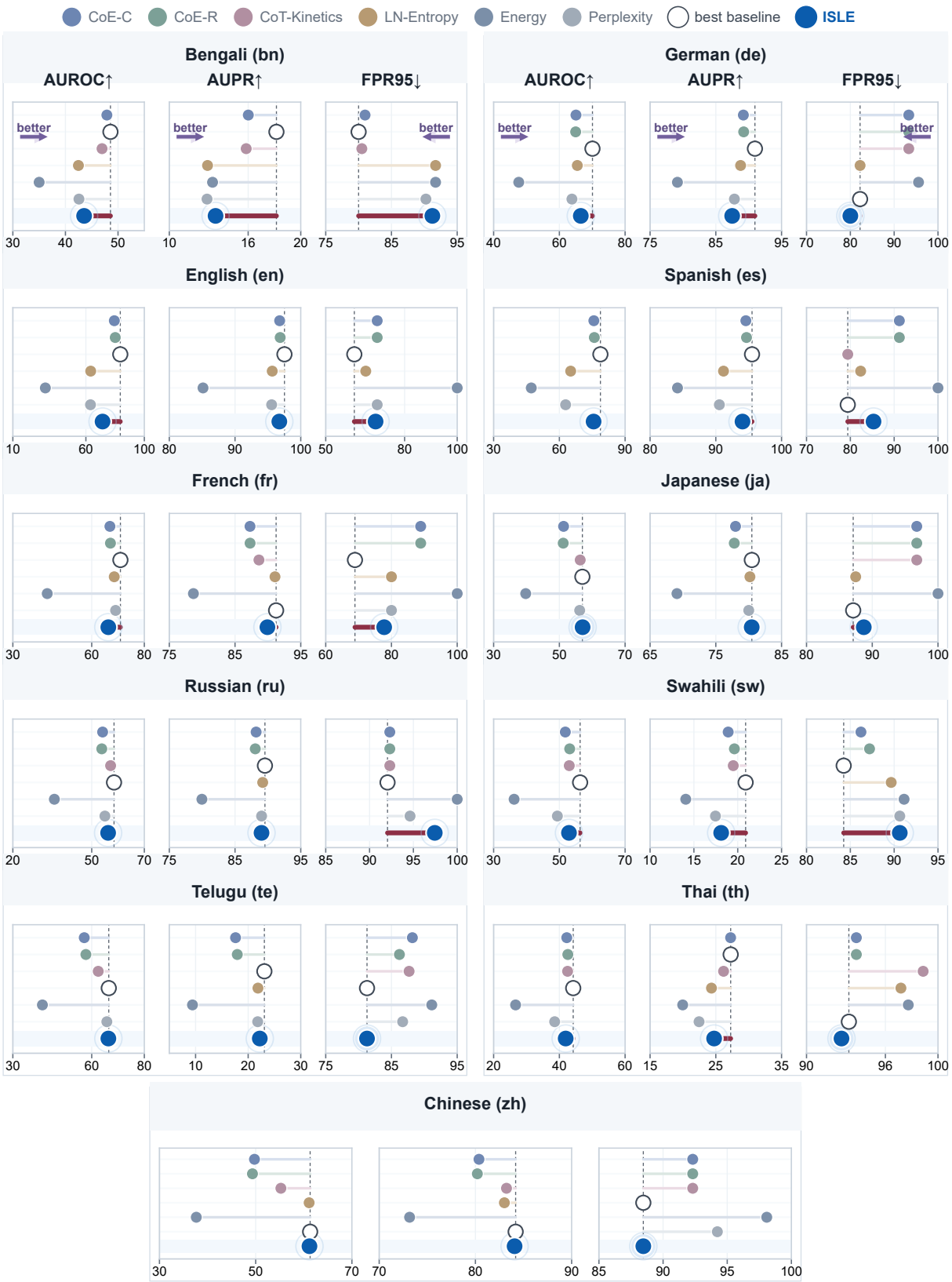

Figure 8: **Language-level MGSM outcome separability for Phi-3-medium-128k-instruct.** The figure shows the performance of different scoring methods across the 11 MGSM language versions. Higher AUROC/AUPR and lower FPR95 indicate better separation between successful and failed generations.

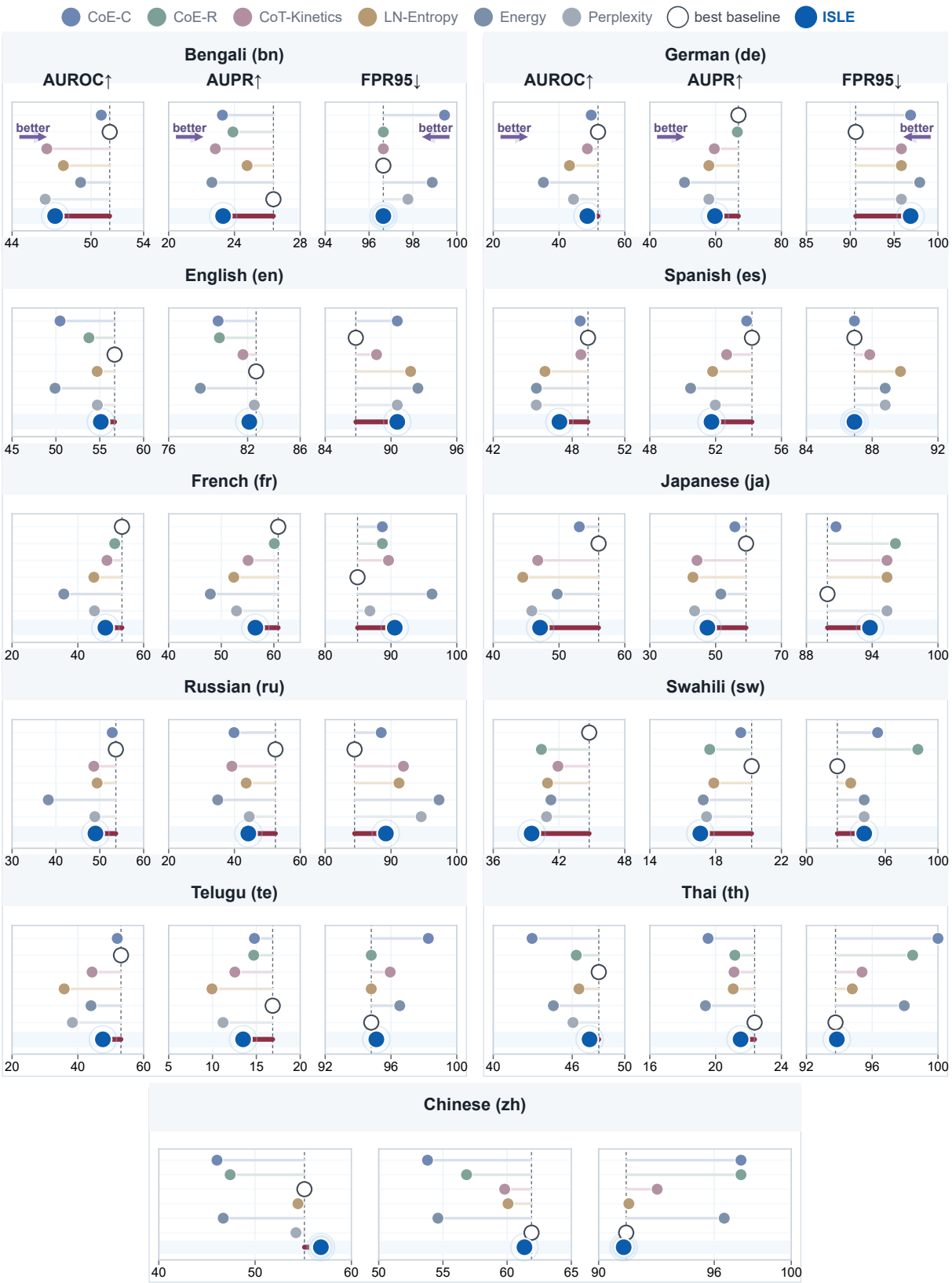

Figure 9: **Language-level MGSM outcome separability for InternLM2-Chat-20B.** The figure shows the performance of different scoring methods across the 11 MGSM language versions. Higher AUROC/AUPR and lower FPR95 indicate better separation between successful and failed generations.

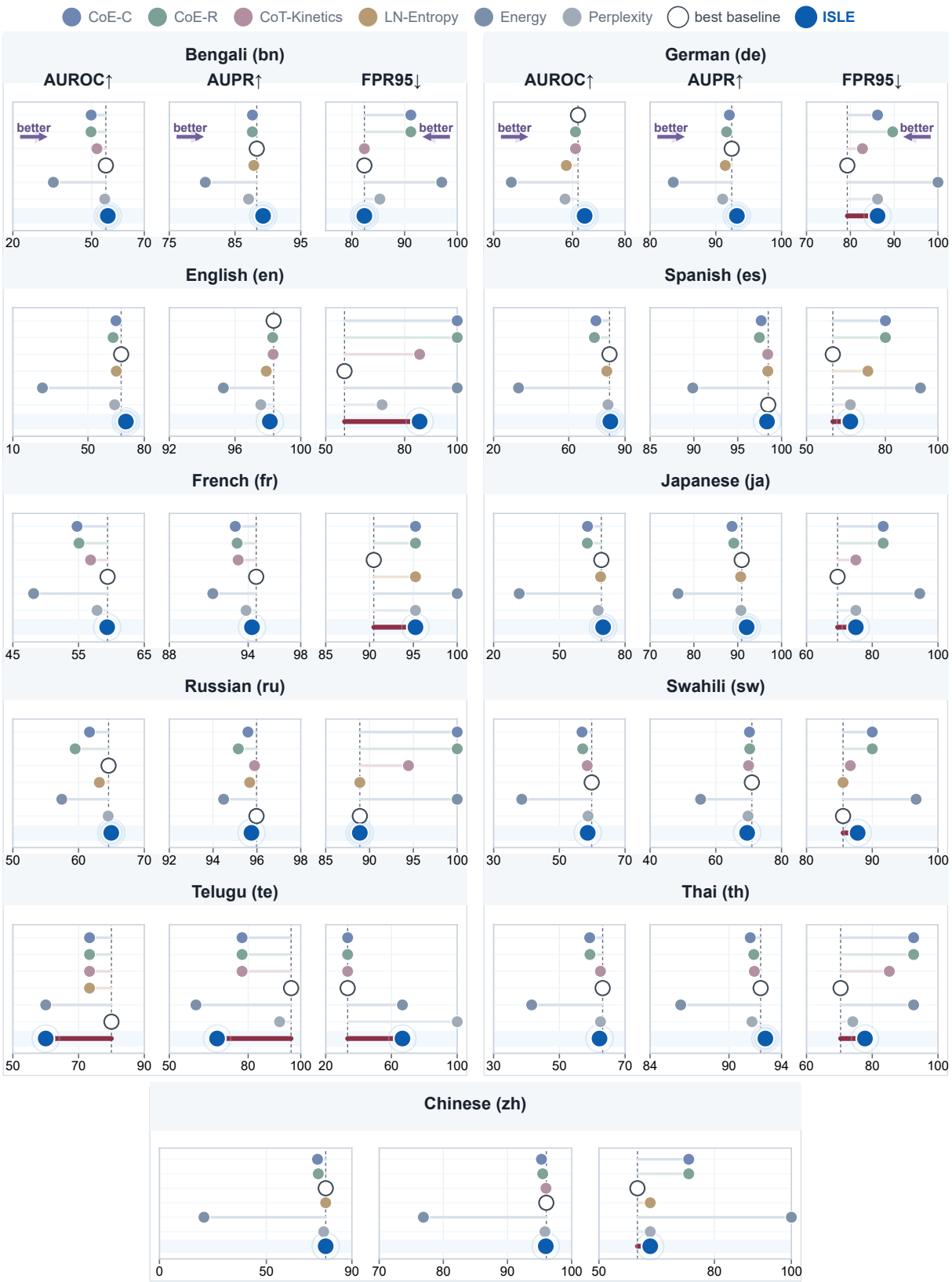

Figure 10: **Language-level MGSM outcome separability for Qwen2.5-32B-Instruct.** The figure shows the performance of different scoring methods across the 11 MGSM language versions. Higher AUROC/AUPR and lower FPR95 indicate better separation between successful and failed generations.

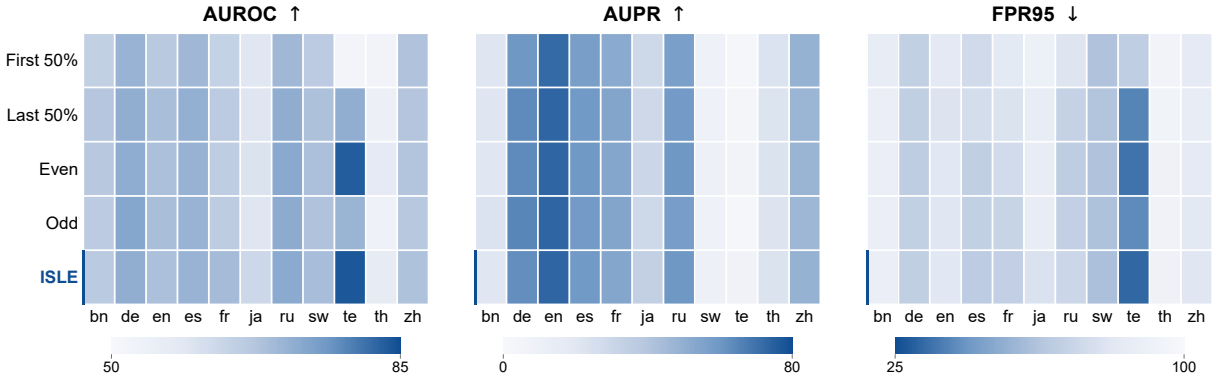

Figure 11: **Language-level MGSM layer-subset analysis for Mistral-7B-Instruct-v0.3.**

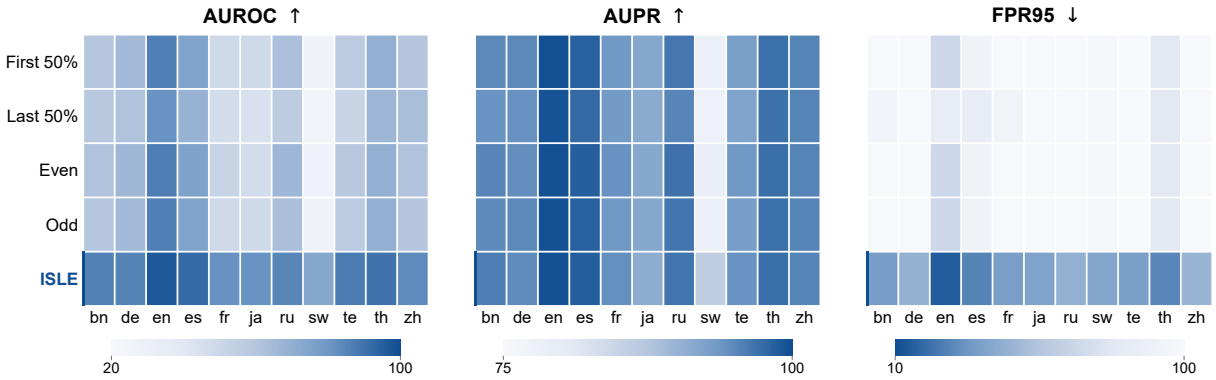

Figure 12: **Language-level MGSM layer-subset analysis for Gemma-3-12B-IT.**

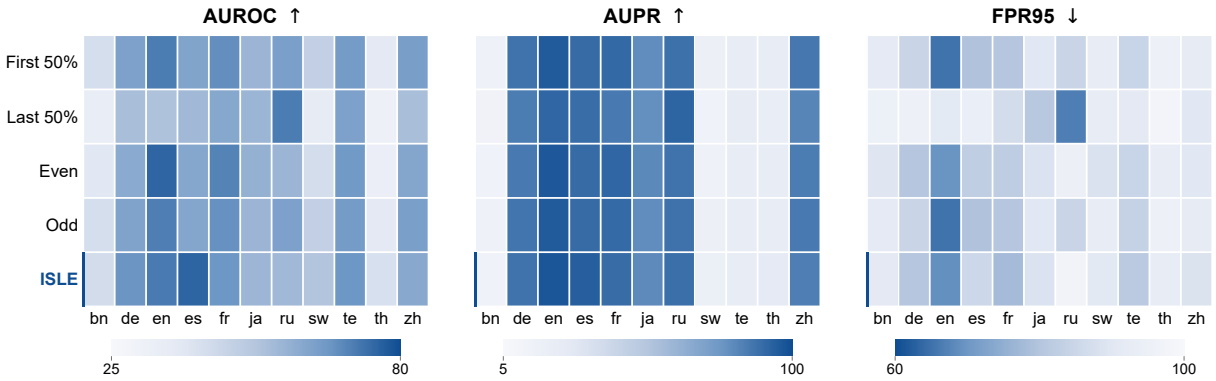

Figure 13: **Language-level MGSM layer-subset analysis for Phi-3-medium-128k-instruct.**

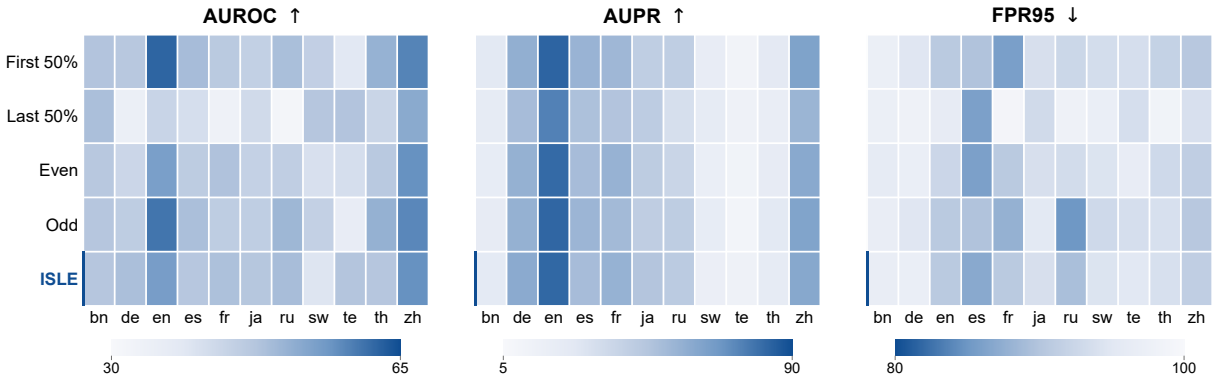

Figure 14: **Language-level MGSM layer-subset analysis for InternLM2-Chat-20B.**

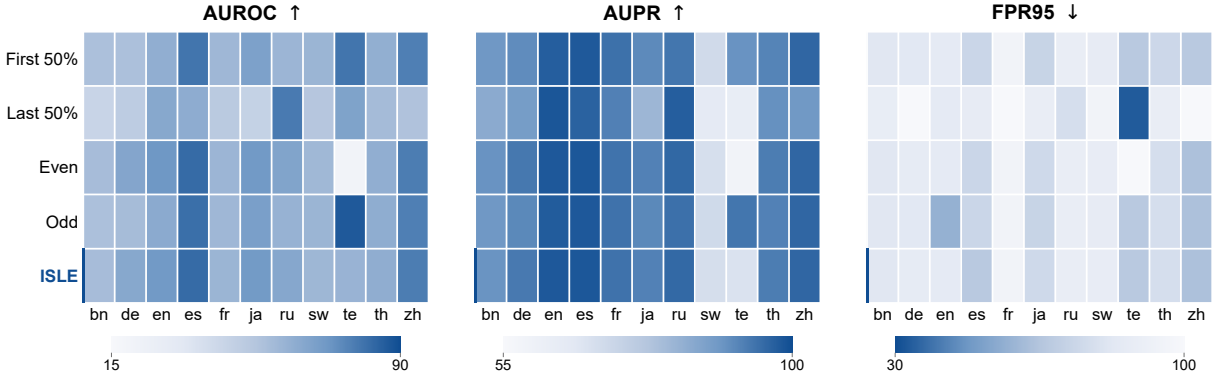

Figure 15: **Language-level MGSM layer-subset analysis for Qwen2.5-32B-Instruct.**

Figure 16: **Prompt templates for cross-task outcome separability.** We use task-specific prompts to standardize answer extraction across mathematical reasoning, scientific multiple-choice reasoning, and code generation.

MGSM Prompt ***Multilingual mathematical reasoning***

```
Answer the following math problem. The last line of your response should be of the following
format: 'Answer: $NUMBER' (without quotes) where NUMBER is the final integer answer.

{input_data}
```

GPQA Diamond Prompt ***Scientific knowledge reasoning***

```
Answer the following multiple choice question. The last line of your response should be of
the following format: 'Answer: $LETTER' (without quotes) where LETTER is one of ABCD. Think
step by step before answering.

{input_data}
```

HumanEval Prompt ***Code generation***

```
You are an AI that only responds with python code, NOT ENGLISH. You will be given a function
signature and its docstring. Write your full implementation (restate the function signature).

IMPORTANT: If the function uses type hints like List, Dict, etc., make sure to include "from
typing import *" at the beginning.

Use a Python code block to write your response. For example:
```python
from typing import *

def hello():
    print("Hello world!")
{input_data}
```
```