HYPERGUIDE: HYPERBOLIC GUIDANCE FOR EFFICIENT MULTI-STEP REASONING IN LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Searching over alternative continuations lets large language models (LLMs) compare the downstream consequences of a reasoning step before committing to it, but expanding and evaluating many branches makes inference expensive. Training value models to score intermediate steps does not avoid this cost, as they still rank candidates at inference time. We introduce HyperGuide, a framework that treats reasoning as movement through a tree of states embedded in hyperbolic space. HyperGuide first trains a state encoder on the Poincaré ball, then trains a guidance head to predict the direction of a minimum-cost transition from the current state. The predicted direction is inserted into the decoder as a virtual token after each reasoning step, so inference follows a single autoregressive trajectory without expanding or reranking candidates. Because the supervision is ordinal rather than binary, it favors continuations that are both reliable and short, which steers generation toward successful, concise solutions. Experiments on competition mathematics and code generation show that HyperGuide matches or exceeds the accuracy of search- and verifier-based baselines while generating substantially fewer tokens than these baselines and about as many as few-shot prompting.

1 INTRODUCTION

Large language models (LLMs) increasingly solve complex tasks through extended reasoning trajectories (Chen et al., 2025; Li et al., 2025; Sui et al., 2025). Reasoning models such as DeepSeek-R1 (Guo et al., 2025) and s1 (Muennighoff et al., 2025) improve these trajectories through reasoning-focused training and additional deliberation at inference time. Generation nevertheless remains sequential: every next step commits to one continuation without comparing its downstream consequences against alternatives, and an unproductive choice may become apparent only several steps later (Yang et al., 2025b; Srivatsa et al., 2025; Zhu et al., 2025). As shown in Figure 1, after setting $\theta = \pi/7$ to compute $\cos\theta \cos 2\theta \cos 4\theta$, applying product-to-sum identities is the habitual next step, yet it only turns the product into a sum of cosines that is no easier to evaluate, and the trajectory needs five steps to reach the answer.

Search-based methods such as Tree of Thoughts (Yao et al., 2023) and Reasoning via Planning (Hao et al., 2023) make this comparison explicit by expanding several continuations from an intermediate state and evaluating their outcomes. A second line of work learns value models and step-level verifiers that guide later decoding (Liu et al., 2024; Lightman et al., 2024; Wang et al., 2024a; Yu et al., 2024). A learned scorer still needs several sampled candidates to rank at every step. Another line keeps a single decoding pass and intervenes in the model's continuous representations, feeding hidden states or soft thought tokens back into the decoder (Hao et al., 2025; Shen et al., 2025; Xu et al., 2025; Zhang et al., 2025). The injected signal is not supervised by what happens after a step is taken.

At any point in reasoning, the same partial solution may admit many plausible continuations. These alternatives are not equally useful: some bring the reasoning closer to a successful solution, while others lead toward unproductive paths. This suggests viewing reasoning not merely as a linear sequence, but as movement through a structured space of possible continuations. If this structure can be captured geometrically, it may provide a local signal for choosing the next step without explicitly exploring the full tree.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

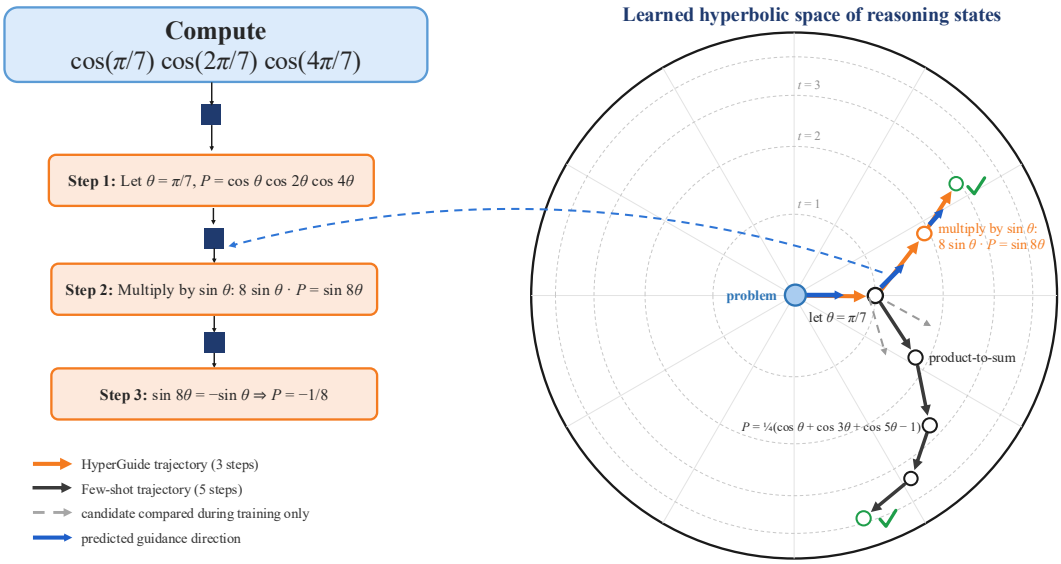


Figure 1: Reasoning example. **Left:** at inference time, each direction enters the decoder as a virtual token $\mathbf{v}_t$, and HyperGuide generates a single trajectory. **Right:** reasoning states in the learned hyperbolic space, where distance from the problem root follows reasoning depth and different branches separate in direction. Dashed candidates are compared only during training, and HyperGuide (orange) follows the predicted direction (blue arrows) of a lower-cost branch.

Hyperbolic space is a natural fit for this branching structure. As reasoning proceeds, each partial solution may give rise to multiple continuations, causing the number of possible paths to grow rapidly with depth. Hyperbolic volume likewise grows exponentially with radius, providing increasing room to distinguish these alternatives as they branch (Nickel & Kiela, 2017; 2018; Sala et al., 2018). At the same time, its metric can express the history shared by different continuations: those that diverge only recently can remain close, whereas those whose paths separated earlier can lie farther apart.

Building on this intuition, we propose HyperGuide (**Hyper**bolic **Guid**ance vector for LLM reasoning), which guides a single reasoning trajectory using directions learned in hyperbolic space. A state encoder maps representations produced by the frozen backbone into the Poincar'e ball. At each state encountered during training, we sample candidate next steps and rank their successor states by success-weighted continuation cost. A guidance head then learns, from the current state alone, to predict a direction aligned with one of the minimum-cost transitions, and the decoder learns to condition on this direction through a virtual token inserted after each completed step (Figure 1, left). We evaluate HyperGuide on MATH-500 and a temporally held-out split of LiveCodeBench using two backbone families. Across these settings, it matches or exceeds the accuracy of search- and verifier-based baselines while generating roughly as many tokens as few-shot prompting. With Qwen2.5-7B-Instruct on MATH-500, for example, HyperGuide achieves 77.5% accuracy at 359 generated tokens, compared with 74.6% at 4,754 tokens for value-model-guided search and 70.0% at 527 tokens for few-shot prompting. Our contributions are summarized as follows:

1. We formulate autoregressive reasoning as a prefix tree over partial trajectories and learn a hyperbolic representation of its states, where radius from the problem root tracks reasoning depth and pairwise distance reflects relative separation in the tree.
2. We derive ordinal supervision over candidate next steps from final-answer verification alone, combining empirical continuation success and length, and use it to learn a guidance direction that enters the decoder as a virtual token, so inference follows a single trajectory without candidate expansion or reranking.
3. Across math and code reasoning benchmarks with two backbone families, HyperGuide matches or exceeds search- and verifier-based baselines at roughly one tenth of their generated tokens, with ablations confirming the roles of hyperbolic geometry and directional guidance.

2 Related Work

2.1 LLM Reasoning

Single-pass prompting methods such as Chain-of-Thought (Wei et al., 2022), Self-Consistency (Wang et al., 2023), and Least-to-Most (Zhou et al., 2023) generate reasoning trajectories without explicitly comparing alternative future continuations at each step. Search-based methods such as Tree of Thoughts (Yao et al., 2023) and Graph of Thoughts (Besta et al., 2024) provide this lookahead by exploring multiple branches. A parallel line scores intermediate steps with learned verifiers (Lightman et al., 2024; Wang et al., 2024a; Uesato et al., 2022) to guide beam or best-of-n decoding. Continuous-reasoning methods instead intervene in the model's representation space: Coconut (Hao et al., 2025) feeds the last hidden state back as the next input embedding, CODI (Shen et al., 2025) compresses explicit chain-of-thought into continuous thoughts, and SoftCoT (Xu et al., 2025) injects soft thought tokens from a fixed assistant model.

2.2 Learning from Search

Search can provide supervision beyond final answers by exposing which intermediate decisions lead to successful continuations. Existing LLM methods use search trajectories for supervised fine-tuning or train separate value models to guide later search (Liu et al., 2024; Tian et al., 2024; Zhang et al., 2024). HyperGuide instead converts verified continuations into ordinal supervision over candidate next steps and uses it to train guidance that is carried along a single autoregressive trajectory.

2.3 Hyperbolic Representations

Hyperbolic space embeds hierarchical or tree-like data with low distortion because ball volume grows exponentially with radius, matching tree branching. Poincaré (Nickel & Kiela, 2017), Lorentz (Nickel & Kiela, 2018), and entailment-cone (Ganea et al., 2018a) embeddings demonstrate this empirically, with theoretical guarantees given by Sala et al. (2018). A line of work generalizes standard neural-network layers to the Poincaré ball and other Riemannian manifolds (Ganea et al., 2018b; Shimizu et al., 2021; Chami et al., 2019; Chen et al., 2022), with applications in vision (Khrulkov et al., 2020), language (Chen et al., 2021), and recent transformer and LLM injections (Yang et al., 2025a). Raj (2026) uses hyperbolic classifiers to probe reasoning-relevant hierarchy in frozen LLM hidden states. HyperGuide instead learns the geometry of reasoning states and uses it to guide next-step generation.

3 Preliminaries

Reasoning states and continuation cost. A *state* s_t is the problem together with the reasoning steps generated so far. A next reasoning step a_t produces the successor state $s_t \oplus a_t$, where $\oplus$ appends the step to the current state. We assume a verifier $\mathcal{V}$ that recognizes a correct final answer and a task-specific rule for identifying a completed reasoning step. The states reachable through continuations generated by a frozen backbone form a reasoning tree $\mathcal{T}$ rooted at the initial problem state s_0. Its depth is $t(s) = d_{\mathcal{T}}(s_0, s)$, and $d_{\mathcal{T}}(s_i, s_j)$ is the shortest-path distance between two states, both measured in reasoning steps. The subscript of s_t denotes depth along a trajectory, whereas s_i, s_j, s_k label arbitrary states of the same problem. The remaining reasoning cost $d(s)$ is the expected cost of a continuation sampled from the frozen backbone: a verified continuation incurs its number of logical reasoning steps, while an unsuccessful continuation incurs a fixed penalty C_{fail}. We estimate this success-weighted cost from sampled continuations as $\hat{d}(s)$.

Hyperbolic geometry. We use the n-dimensional Poincaré ball $\mathbb{D}_c^n = \{\mathbf{x} \in \mathbb{R}^n : c\|\mathbf{x}\|^2 < 1\}$ with curvature $-c$, where $c > 0$. Its origin is $\mathbf{0}$, its geodesic distance is $d_{\mathbb{D}}$, and $T_{\mathbf{z}}\mathbb{D}_c^n$ denotes the tangent space at $\mathbf{z}$. The logarithmic map $\log_{\mathbf{z}}^c(\mathbf{z}') \in T_{\mathbf{z}}\mathbb{D}_c^n$ gives the tangent vector from $\mathbf{z}$ to $\mathbf{z}'$. The operator $\mathrm{PT}_{\mathbf{x}\to\mathbf{y}}^c : T_{\mathbf{x}}\mathbb{D}_c^n \to T_{\mathbf{y}}\mathbb{D}_c^n$ parallel-transports a tangent vector along the geodesic from $\mathbf{x}$ to $\mathbf{y}$. Both superscripts c refer to the same curvature parameter. For tangent vectors at the same point, $\mathrm{sim}(\mathbf{a}, \mathbf{b})$ denotes their cosine similarity.

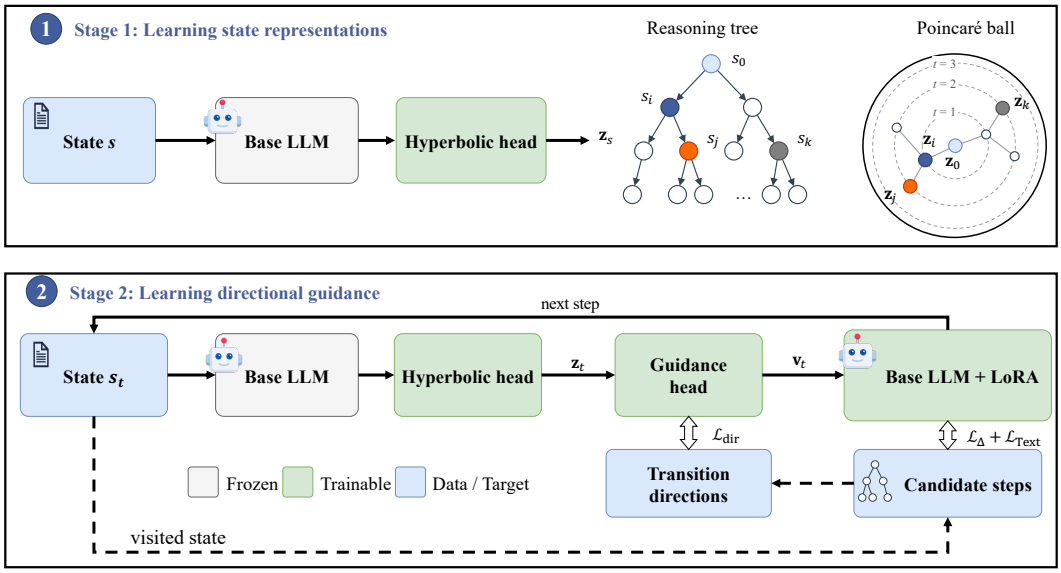


Figure 2: Overview of HyperGuide. **Stage 1 (top):** a hyperbolic head maps frozen LLM representations of reasoning states into the Poincaré ball. **Stage 2 (bottom):** candidate steps ranked by continuation cost provide transition targets for the guidance head and preference supervision for the LoRA-adapted decoder.

4 Methodology

4.1 Overview

HyperGuide is trained in two stages, as summarized in Figure 2. Stage 1 learns hyperbolic representations of reasoning states (Section 4.2). Stage 2 trains the model to predict a guidance direction from the current state and use it to condition next-step generation (Section 4.3). Section 4.4 then describes how these components guide a single reasoning trajectory at inference time.

4.2 Learning State Representations

In the first stage, we train an encoder h_ϕ to learn a representation for each reasoning state. Let $f_\theta(s)$ be the hidden representation of state s from a frozen LLM backbone, and let h_ϕ map that representation into the Poincaré ball, giving the state embedding $\mathbf{z}_s = h_\phi(f_\theta(s)) \in \mathbb{D}_c^n$. Embeddings inherit the indices of their states, e.g., $\mathbf{z}_t = \mathbf{z}_{s_t}$ and $\mathbf{z}_i = \mathbf{z}_{s_i}$; in particular, $\mathbf{z}_0$ is the embedding of the problem root. Write $r(s) = d_{\mathbb{D}}(\mathbf{z}_0, \mathbf{z}_s)$ for the radius of s from its root. For two states from the same problem with $t(s_i) < t(s_j)$ and a margin $\gamma_{\text{rank}} > 0$, the radial ranking objective is

$$\mathcal{L}_{\text{rank}} = \mathbb{E}\big[\max\big(0, r(s_i) - r(s_j) + \gamma_{\text{rank}}\big(t(s_j) - t(s_i)\big)\big)\big]\,. \tag{1}$$

Thus deeper states are trained to lie farther from the root of their reasoning tree.

We also preserve relative distances in the reasoning tree. For triplets (s_i, s_j, s_k) from the same problem satisfying $d_{\mathcal{T}}(s_i, s_j) < d_{\mathcal{T}}(s_i, s_k)$ and a margin $\gamma_{\text{metric}} > 0$, let

$$\mathcal{L}_{\text{metric}} = \mathbb{E}[w_{ijk} \max(0, d_{\mathbb{D}}(\mathbf{z}_i, \mathbf{z}_j) - d_{\mathbb{D}}(\mathbf{z}_i, \mathbf{z}_k) + \gamma_{\text{metric}})]\,, \tag{2}$$

where the importance weight w_{ijk} down-weights triplets with large tree distances. This keeps nearby reasoning states closer in the learned geometry than states on more distant branches. The complete Stage 1 objective is

$$\mathcal{L}_{\text{Geom}} = \mathcal{L}_{\text{rank}} + \lambda_{\text{metric}} \mathcal{L}_{\text{metric}}, \tag{3}$$

where $\lambda_{\text{metric}} \geq 0$ weights the tree-metric objective.

4.3 LEARNING DIRECTIONAL GUIDANCE

In the second stage, starting from these state representations, we train a guidance head to predict a direction for the next reasoning step. The predicted direction is converted into a virtual token, and we train the decoder to favor lower-cost next steps when the token is present.

Guidance token construction. Consider a state s_t visited during training. We form a candidate set $\mathcal{C}(s_t) = \{a_t^{(i)}\}_i$, where each next reasoning step gives a successor state $s_{t+1}^{(i)} = s_t \oplus a_t^{(i)}$. The continuation estimator assigns each successor a remaining-cost estimate $\hat{d}_i = \hat{d}(s_{t+1}^{(i)})$. We embed each successor and express its transition from the current state in the tangent space at $\mathbf{z}_t$ as $\mathbf{u}_t^{(i)} = \log_{\mathbf{z}_t}^c(\mathbf{z}_{t+1}^{(i)}) \in T_{\mathbf{z}_t}\mathbb{D}_c^n$, the geometric direction taken by candidate step i. A guidance head predicts a direction $\hat{\mathbf{u}}_t = q_\omega(\mathbf{z}_t, \mathbf{z}_0)$ from the current state and its problem root. Because $\hat{\mathbf{u}}_t$ lies in a state-specific tangent space, we parallel-transport it to the tangent space at the origin and map it to the backbone hidden width:

$$\mathbf{v}_t = g_\psi\big(\mathrm{PT}^c_{\mathbf{z}_t \to \mathbf{0}}(\hat{\mathbf{u}}_t)\big) = g_\psi\big(\mathrm{PT}^c_{\mathbf{z}_t \to \mathbf{0}}\big(q_\omega(\mathbf{z}_t, \mathbf{z}_0)\big)\big)\,. \tag{4}$$

The resulting $\mathbf{v}_t$ is the virtual guidance token inserted into the decoder.

Guidance objective. We order candidate transitions by successor cost, $\mathcal{R}(s_t) = \{(i,j) : \hat{d}_i < \hat{d}_j\}$, excluding ties, and collect the minimum-cost candidates $\mathcal{P}(s_t) = \{i : \hat{d}_i = \min_k \hat{d}_k\}$. We train the predicted direction to align with at least one member of $\mathcal{P}(s_t)$, with a margin $\gamma_{\mathrm{dir}} > 0$:

$$\mathcal{L}_{\mathrm{dir}} = \mathbb{E}_{s_t, j \notin \mathcal{P}(s_t)}\left[\max\left(0, \mathrm{sim}(\hat{\mathbf{u}}_t, \mathbf{u}_t^{(j)}) - \max_{i \in \mathcal{P}(s_t)} \mathrm{sim}(\hat{\mathbf{u}}_t, \mathbf{u}_t^{(i)}) + \gamma_{\mathrm{dir}}\right)\right]. \tag{5}$$

For $(i,j) \in \mathcal{R}(s_t)$, define

$$\begin{aligned} \Delta_g^{ij} &= \ell(a_t^{(i)} \mid s_t, \mathbf{v}_t) - \ell(a_t^{(j)} \mid s_t, \mathbf{v}_t), \\ \Delta_0^{ij} &= \ell(a_t^{(i)} \mid s_t, \varnothing) - \ell(a_t^{(j)} \mid s_t, \varnothing), \end{aligned} \tag{6}$$

where ℓ is sequence log-likelihood under the same adapted decoder, and $\varnothing$ denotes that no virtual guidance token is inserted. With logistic sigmoid $\sigma(x) = 1/(1+e^{-x})$, we optimize

$$\mathcal{L}_{\mathrm{Align}} = \mathcal{L}_{\mathrm{dir}} + \lambda_\Delta \mathcal{L}_\Delta, \qquad \mathcal{L}_\Delta = -\mathbb{E}_{s_t, (i,j) \in \mathcal{R}(s_t)}\left[\log \sigma\left(\Delta_g^{ij} - \Delta_0^{ij}\right)\right]. \tag{7}$$

Here $\lambda_\Delta \geq 0$ weights the guidance-preference term. The directional loss selects one minimum-cost transition, while $\mathcal{L}_\Delta$ requires the virtual token to increase the decoder's relative preference for lower-cost transitions. The latter ordering reflects both empirical continuation success and the lengths of successful continuations.

We additionally use an auxiliary text loss $\mathcal{L}_{\mathrm{Text}}$ on minimum-cost candidate steps and verified final-answer steps while retaining the state-geometry objective (Appendix B.4). The complete Stage 2 objective is

$$\mathcal{L}_{\mathrm{Guide}} = \mathcal{L}_{\mathrm{Align}} + \lambda_{\mathrm{text}} \mathcal{L}_{\mathrm{Text}}, \tag{8}$$

where $\lambda_{\mathrm{text}} \geq 0$ weights the auxiliary training term.

4.4 INFERENCE

At inference time, HyperGuide generates a single autoregressive trajectory without using continuation costs. After each completed reasoning step, the accumulated state s_t is encoded as $\mathbf{z}_t$, and the guidance head predicts a direction from $\mathbf{z}_t$ and the problem-root embedding $\mathbf{z}_0$. This direction is converted into the virtual token $\mathbf{v}_t$ by Equation 4 and appended to the decoder stream before generation continues. The process repeats until the model produces a final answer. No candidate branches are sampled, scored, or reranked at inference time.

Table 1: Main results on MATH-500 and the temporally held-out LiveCodeBench code-generation split, with Qwen2.5-7B-Instruct and Llama-3.1-8B-Instruct as backbones. Tokens is the mean number of generated tokens. Self-Consistency is evaluated on MATH-500 only. The best result in each column is bolded and second best is underlined.

	MATH-500				LiveCodeBench			
	Qwen2.5-7B		Llama-3.1-8B		Qwen2.5-7B		Llama-3.1-8B	
Method	Acc. ↑	Tokens ↓	Acc. ↑	Tokens ↓	Pass@1 ↑	Tokens ↓	Pass@1 ↑	Tokens ↓
Training-free								
Few-shot	70.0	527.4	43.4	325.8	22.9	376.8	11.1	319.9
Self-Consistency	71.3±0.6	2504.3±57.3	46.7±0.7	1549.0±16.1	–	–	–	–
Tree of Thoughts	66.2±2.5	5532.0±94.6	40.2±1.3	3859.4±96.4	20.9±1.8	3965.2±85.2	11.3±1.1	3441.5±16.7
Soft Thinking	71.0	506.4	44.0	313.6	21.1	357.1	15.2	307.2
Training-based								
PT-SFT	69.5±1.4	532.1±6.5	41.1±1.6	337.9±2.0	19.3±1.9	379.1±8.8	13.0±2.1	313.0±4.3
OVM	74.6±1.7	4754.3±27.4	46.4±1.1	3968.2±22.6	**24.9**±1.5	3716.7±87.2	17.8±1.8	3118.5±19.7
Coconut	26.2±0.7	**36.0**±0.3	17.5±0.5	**25.3**±0.6	3.0±0.4	**196.3**±4.4	2.1±0.3	**159.2**±3.9
SoftCoT	70.8±0.8	528.9±7.7	45.3±1.8	322.0±6.3	23.2±1.5	371.6±7.6	13.6±1.1	319.7±4.4
HyperGuide (ours)	**77.5**±1.4	358.9±7.1	**48.6**±0.9	310.8±4.5	24.4±0.7	365.7±7.8	**19.3**±0.9	306.7±7.2

5 Experiments

5.1 Experimental Settings

Implementation Details. We instantiate HyperGuide on **Qwen2.5-7B-Instruct** (Qwen et al., 2024) and **Llama-3.1-8B-Instruct** (Grattafiori et al., 2024), two model families at a similar scale. Stage 1 trains the state encoder on top of the frozen backbone. Stage 2 jointly refines the state encoder and trains the guidance head, the up-projector, and a LoRA adapter on the decoder. We train HyperGuide and every ablation variant with three random seeds (4, 44, and 444). Hyperparameters and other implementation details are provided in Appendix A.

Baselines. We compare HyperGuide with two types of baselines. *Training-free* methods include **Few-shot** prompting (Brown et al., 2020), **Self-Consistency** (Wang et al., 2023) (majority voting over sampled trajectories, MATH-500 only), **Tree of Thoughts** (Yao et al., 2023) (search over candidate steps scored by the backbone), and **Soft Thinking** (Zhang et al., 2025) (continuous concept tokens in a single pass). *Training-based* methods include **PT-SFT** (Wang et al., 2024b) (fine-tuning with learned planning tokens), **OVM** (Yu et al., 2024) (beam search guided by an outcome value model), and the latent-reasoning methods **Coconut** (Hao et al., 2025) and **SoftCoT** (Xu et al., 2025).

Datasets and Metrics. We evaluate on all 500 problems in **MATH-500** (Lightman et al., 2024) and on 341 code-generation problems in a temporally held-out **LiveCodeBench** test window (Jain et al., 2025). The LiveCodeBench split is defined by release date so that every test problem postdates the training and validation problems (Table 5). We report two metrics over each complete test set: (1) *Accuracy* on MATH-500 and *Pass@1* on LiveCodeBench, the percentage of problems with a verified final answer or passing code; and (2) *Tokens*, the mean number of generated tokens. For seed-dependent methods, we report the mean and standard deviation over the three seeds.

5.2 Main Results

Table 1 reports the main results. HyperGuide attains the highest mean accuracy in three of the four settings. On MATH-500, it leads the strongest baseline by more than one standard deviation of either method; on LiveCodeBench, the gap between HyperGuide and OVM is within one standard deviation of OVM with both backbones.

Comparison with search- and verifier-based methods. OVM is the strongest baseline, but it scores several candidate steps at every state and generates 10 to 13 times as many tokens as HyperGuide. HyperGuide exceeds its accuracy by 2.2 to 2.9 points on MATH and is on par with it on LiveCodeBench, while following a single trajectory. Tree of Thoughts generates 11 to 15 times as

Table 2: Ablations on Qwen2.5-7B-Instruct. The best result in each column is bolded.

	MATH-500		LiveCodeBench	
Variant	Accuracy ↑	Tokens ↓	Pass@1 ↑	Tokens ↓
HyperGuide	**77.5**±1.4	**358.9**±7.1	**24.4**±0.7	**365.7**±7.8
Euclidean geometry	74.4±1.5	366.7±6.3	20.0±1.8	378.2±7.6
w/o $\mathcal{L}_{\text{rank}}$	72.6±1.2	453.7±4.7	21.6±1.9	392.5±6.4
w/o $\mathcal{L}_{\text{metric}}$	74.2±1.6	362.1±2.2	22.2±1.4	371.2±8.8
w/o guidance token	70.4±0.7	377.4±2.6	20.8±0.8	370.3±3.3

many tokens and still falls below Few-shot in three of the four settings and only matches it in the fourth, which suggests that the self-evaluation of a small backbone is not reliable enough.

Comparison with single-pass methods. Soft Thinking, SoftCoT, and PT-SFT also decode one trajectory, but they stay within 4.1 points of Few-shot. Coconut replaces most textual steps with latent ones and produces very short outputs, and its accuracy falls far below Few-shot on both benchmarks. HyperGuide exceeds the best single-pass baseline by 3.3 to 6.5 points in three settings; on LiveCodeBench with Qwen2.5-7B, its 1.2-point lead over SoftCoT is within the standard deviation of SoftCoT. The difference lies in what supervises the injected signal: the planning tokens and soft thoughts of these baselines are not trained on what happens after a step is taken, whereas the guidance token is trained to raise the decoder's preference for the successor with the lower remaining cost.

5.3 Ablations

We ablate the components behind the two parts of HyperGuide. Each variant changes one component and keeps the rest of the training and inference procedure fixed. Table 2 reports the results. Several additional ablations are reported in Appendix C.1.

Euclidean geometry. In this variant, we replace the Poincaré ball with $\mathbb{R}^n$ and keep both Stage 1 objectives, with Euclidean distances in place of hyperbolic ones. Accuracy drops by 3.1 points on MATH and Pass@1 by 4.4 points on LiveCodeBench, the largest drop on that benchmark, while the token counts change by less than 4%. The same supervision is therefore less effective in flat space.

w/o $\mathcal{L}_{\text{rank}}$. This variant removes the radial depth-ranking objective, so distance from the problem root is no longer explicitly ordered by reasoning depth. Accuracy drops by 4.9 points on MATH-500 and Pass@1 by 2.8 points on LiveCodeBench. It also clearly lengthens trajectories, from 358.9 to 453.7 tokens on MATH-500 and from 365.7 to 392.5 on LiveCodeBench.

w/o $\mathcal{L}_{\text{metric}}$. This variant removes the tree-metric objective, so pairwise distances are no longer trained to follow branch structure. Accuracy drops by 3.3 points on MATH-500 and Pass@1 by 2.2 points on LiveCodeBench, and the token counts stay within 2% of the full model. Branch structure thus mainly affects whether the model reaches a correct solution, not how long that solution is.

w/o guidance token. In this variant, we train the decoder adapter on the same data but neither predict nor insert $\mathbf{v}_t$, so the decoder receives no signal from the learned geometry. This yields the largest accuracy drop on MATH-500, 7.1 points, and a 3.6-point drop in Pass@1 on LiveCodeBench. The gain of HyperGuide therefore does not come from adapter training alone: the learned representation improves generation only when its direction is passed to the decoder.

5.4 Representation of Reasoning Depth and Prefix-Tree Structure

The ablations show that the two Stage 1 objectives matter for accuracy, but not whether the learned space has the structure they are meant to impose. We therefore test the deployed state encoder on problems it has never seen: does radius from the problem root follow reasoning depth, and does pairwise distance follow distance in the reasoning tree?

Setup. On Qwen2.5-7B-Instruct and MATH-500, we build a reasoning tree per evaluation problem with the training-time continuation sampling, which yields held-out states with depth $t(s)$, tree distances $d_{\mathcal{T}}$, and candidate steps with costs $\hat{d}$. We encode every state with the HyperGuide encoder

Table 3: Structure of the learned space on held-out MATH-500 states (Qwen2.5-7B-Instruct), in %. Chance is 50% for every column.

Encoder	Depth order ↑	Triplet accuracy ↑	
		All	Across branches
Hyperbolic (HyperGuide)	94.5	88.7	82.1
Euclidean	87.3	78.2	69.8

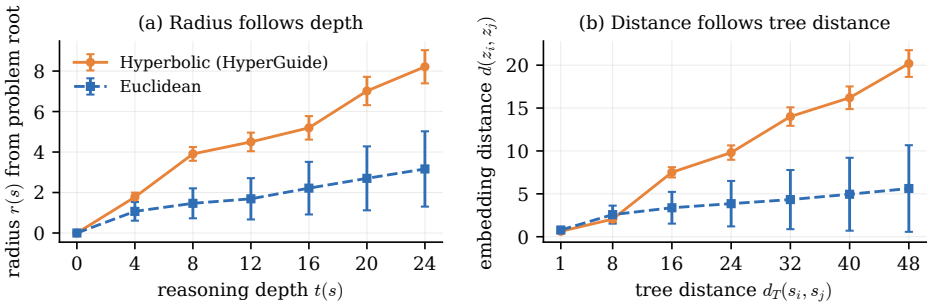


Figure 3: Learned geometry on held-out MATH-500 states. **(a)** Mean radius from the problem root at each reasoning depth. **(b)** Mean embedding distance between two states at each tree distance. Error bars show one standard deviation. The hyperbolic encoder keeps both relations monotone; the Euclidean encoder compresses deep states and distant branches.

and with the *Euclidean geometry* encoder of Table 2, both trained with seed 4. Two metrics mirror the two objectives: *depth order*, the share of same-problem pairs with $t(s_i) < t(s_j)$ for which $r(s_i) < r(s_j)$, and *triplet accuracy*, the share of triplets with $d_{\mathcal{T}}(s_i, s_j) < d_{\mathcal{T}}(s_i, s_k)$ whose embedding distances are ordered the same way; chance is 50% for both. We report triplet accuracy over all triplets and over the subset whose states lie on different branches, where the metric tests branch separation rather than step counting along one path.

Radius follows depth. As shown in Table 3, both encoders order most held-out pairs by depth (94.5% against 87.3%), since $\mathcal{L}_{\text{rank}}$ trains this directly. The difference appears at larger depth (Figure 3(a)): in the hyperbolic encoder the mean radius keeps growing with $t(s)$, whereas in the Euclidean encoder it flattens, so deep states from the same problem become hard to tell apart by radius alone.

Distance follows tree distance. Triplet accuracy separates the two geometries more clearly: 88.7% against 78.2% over all triplets, and the gap widens to 12.3 points (82.1% against 69.8%) on triplets across branches. In flat space, states that diverged several steps ago are placed at nearly the same distance as states that diverged recently (Figure 3(b)), while the hyperbolic encoder keeps the distance growing with $d_{\mathcal{T}}$.

5.5 Causal Effect of the Guidance Token on Next-Step Selection

Section 5.4 does not show whether the guidance head can select a lower-cost branch from the current state, or whether the decoder uses the resulting token. We therefore keep the trained HyperGuide model fixed and change only the token it receives at inference time.

Setup. We compare four conditions on Qwen2.5-7B-Instruct and MATH-500: *Predicted* inserts $\mathbf{v}_t$; *No token* inserts nothing; *Shuffled* inserts the token predicted at a random state of a different test problem, which is in-distribution but carries the direction of an unrelated state; *Negated* replaces $\hat{\mathbf{u}}_t$ with $-\hat{\mathbf{u}}_t$ in Equation 4. We report accuracy and tokens as in Table 1, and two step-level metrics on the held-out states of Section 5.4 with at least two candidate steps of different cost, where the decoder only scores fixed candidates. *Pair shift* is the share of pairs $(i, j) \in \mathcal{R}(s_t)$ with $\Delta_g^{ij} > \Delta_0^{ij}$, that is, the token moves likelihood toward the lower-cost candidate; chance is 50%. *Min-cost top-1* is the share of states at which the most likely candidate has minimum cost.

Table 4: Inference-time interventions on the guidance token of the trained HyperGuide model (Qwen2.5-7B-Instruct, MATH-500). Pair shift and Min-cost top-1 are in % and are computed on held-out states with fixed candidate steps. Pair shift compares against the same decoder without a token, so it is undefined for *No token*. The best result in each column is bolded.

	Decoding		Step level	
Inserted token	Accuracy ↑	Tokens ↓	Pair shift ↑	Min-cost top-1 ↑
Predicted $\mathbf{v}_t$ (HyperGuide)	**77.4**	**352.7**	**77.8**	**61.8**
No token	70.2	380.0	–	45.2
Shuffled	71.4	389.2	52.0	47.6
Negated	65.2	405.3	34.4	40.8

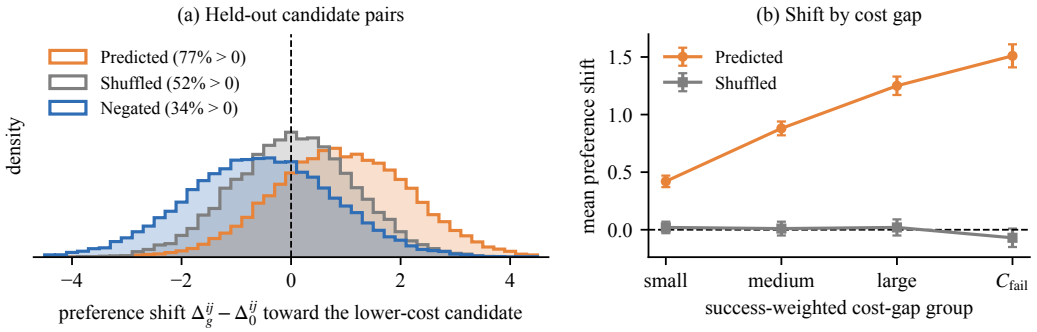


Figure 4: Effect of the inserted token on the decoder's preference between two candidate steps at held-out MATH-500 states. The preference shift $\Delta_g^{ij} - \Delta_0^{ij}$ is positive when the token moves likelihood toward the lower-cost candidate. **(a)** Distribution over all candidate pairs for the three inserted tokens. **(b)** Mean shift with 95% confidence intervals, grouped by success-weighted cost gap; C_{fail} denotes pairs in which the higher-cost candidate has no verified continuation.

The decoder reads the content of the token. With the predicted token, the likelihood margin moves toward the lower-cost candidate in 77.8% of the pairs, and the most likely candidate is a minimum-cost one at 61.8% of the states, against 45.2% without a token. A shuffled token has no such effect. Its pair shift stays near chance (52.0%), its distribution in Figure 4(a) is centered near zero, and its accuracy differs by 1.2 points from decoding without a token. The shuffled token therefore does not reproduce the effect of the predicted token. The 7-point accuracy difference between the predicted token and *No token* depends on carrying a direction specific to the current state.

The effect follows the cost gap. Figure 4(b) groups the pairs by the difference in remaining cost between the two candidates. The shift caused by the predicted token is smallest for small finite cost gaps, grows with the gap, and is largest when the higher-cost candidate has no verified continuation. The token therefore acts most strongly on the decisions that matter most, and it also separates candidates that both lead to a solution, which binary supervision cannot do.

6 Conclusion

This work introduced HyperGuide, which represents the branching structure of reasoning in hyperbolic space and uses it to guide a single autoregressive trajectory. By learning from alternative continuations during training, HyperGuide avoids expanding or reranking candidate steps at inference time. Across MATH-500 and LiveCodeBench with two backbone families, it matches or exceeds the accuracy of search- and verifier-based baselines while using roughly one tenth of their generated tokens. Ablations and inference-time interventions further show that both the learned geometry and the resulting directional guidance contribute to these gains. More broadly, our results suggest that the geometry of reasoning prefixes provides a useful coordinate system for guiding generation.

AI USE STATEMENT

In this work, we used generative AI tools to analyse existing literature, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, modify scientific figures, create or edit software code, identify relevant literature, format references, provide feedback on research methodology and experiments. We have not used generative AI tools for generating data, proposing idea, or interpreting results; translation, dataset cleaning are not applicable to this work. We have reviewed all AI-assisted work, and every AI-suggested edit was checked by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Anonymized source code for training and evaluation is available at `https://anonymous.4open.science/r/anonymous-D2A8`. Model architecture, training objectives, hyperparameters, and dataset construction are described in Section 4 and Appendices A and B. All seed-dependent results are reported as the mean and standard deviation over seeds 4, 44, and 444.

REFERENCES

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of Thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17682–17690, 2024.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901, 2020.

Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. Hyperbolic graph convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Boli Chen, Yao Fu, Guangwei Xu, Pengjun Xie, Chuanqi Tan, Mosha Chen, and Liping Jing. Probing BERT in hyperbolic spaces. In *International Conference on Learning Representations*, 2021.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*, 2025.

Weize Chen, Xu Han, Yankai Lin, Hexu Zhao, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Fully hyperbolic neural networks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5672–5686, 2022.

Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic entailment cones for learning hierarchical embeddings. In *International conference on machine learning*, pp. 1646–1655. PMLR, 2018a.

Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. In *Advances in Neural Information Processing Systems*, volume 31, 2018b.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

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

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason E Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. In *Second Conference on Language Modeling*, 2025.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, 2025.

Valentin Khrulkov, Leyla Mirvakhabova, Evgeniya Ustinova, Ivan Oseledets, and Victor Lempitsky. Hyperbolic image embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6417–6427, 2020.

Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, Yingying Zhang, Fei Yin, Jiahua Dong, Zhiwei Li, Bao-Long Bi, Ling-Rui Mei, Junfeng Fang, Xiao Liang, Zhijiang Guo, Le Song, and Cheng-Lin Liu. From System 1 to System 2: A survey of reasoning large language models. *arXiv preprint arXiv:2502.17419*, 2025.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024.

Jiacheng Liu, Andrew Cohen, Ramakanth Pasunuru, Yejin Choi, Hannaneh Hajishirzi, and Asli Celikyilmaz. Don't throw away your value model! Generating more preferable text with value-guided Monte-Carlo tree search decoding. In *Conference on Language Modeling*, 2024.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20286–20332, 2025.

Maximilian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Maximilian Nickel and Douwe Kiela. Learning continuous hierarchies in the Lorentz model of hyperbolic geometry. In *International conference on machine learning*, pp. 3779–3788. PMLR, 2018.

Qwen, 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, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Arnav Raj. Hyperbolic geometry of reasoning: Probing LLM hidden states. In *ICLR 2026 Workshop on Geometry-grounded Representation Learning and Generative Modeling (GRaM), Tiny Paper Track*, 2026.

Frederic Sala, Chris De Sa, Albert Gu, and Christopher Ré. Representation tradeoffs for hyperbolic embeddings. In *International conference on machine learning*, pp. 4460–4469. PMLR, 2018.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. CODI: Compressing chain-of-thought into continuous space via self-distillation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 677–693, 2025.

Ryohei Shimizu, YUSUKE Mukuta, and Tatsuya Harada. Hyperbolic neural networks++. In *International Conference on Learning Representations*, 2021.

Kv Aditya Srivatsa, Kaushal Kumar Maurya, and Ekaterina Kochmar. Llms cannot spot math errors, even when allowed to peek into the solution. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 10925–10939, 2025.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, Hanjie Chen, and Xia Hu. Stop overthinking: A survey on efficient reasoning for large language models. *Transactions on Machine Learning Research*, 2025.

Ye Tian, Baolin Peng, Linfeng Song, Lifeng Jin, Dian Yu, Lei Han, Haitao Mi, and Dong Yu. Toward self-improvement of llms via imagination, searching, and criticizing. In *Advances in Neural Information Processing Systems*, volume 37, pp. 52723–52748, 2024.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 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)*, pp. 9426–9439, 2024a.

Xinyi Wang, Lucas Caccia, Oleksiy Ostapenko, Xingdi Yuan, William Yang Wang, and Alessandro Sordoni. Guiding language model reasoning with planning tokens. In *Conference on Language Modeling*, 2024b.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. SoftCoT: Soft chain-of-thought for efficient reasoning with LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23336–23351, 2025.

Menglin Yang, Ram Samarth B B, Aosong Feng, Bo Xiong, Jiahong Liu, Irwin King, and Rex Ying. Hyperbolic fine-tuning for large language models. In *Advances in Neural Information Processing Systems*, volume 38, 2025a.

Zhaohui Yang, Chenghua He, Xiaowen Shi, Shihong Deng, Linjing Li, Qiyue Yin, and Daxin Jiang. Beyond the first error: Process reward models for reflective mathematical reasoning. In *EMNLP (Findings)*, pp. 4711–4728, 2025b.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of Thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 11809–11822, 2023.

Fei Yu, Anningzhe Gao, and Benyou Wang. OVM, outcome-supervised value models for planning in mathematical reasoning. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 858–875, 2024.

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

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, and Xin Wang. Soft thinking: Unlocking the reasoning potential of llms in continuous concept space. In *Advances in Neural Information Processing Systems*, volume 38, pp. 168990–169012, 2025.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *International Conference on Learning Representations*, 2023.

Kunlun Zhu, Zijia Liu, Bingxuan Li, Muxin Tian, Yingxuan Yang, Jiaxun Zhang, Pengrui Han, Qipeng Xie, Fuyang Cui, Weijia Zhang, Xiaoteng Ma, Xiaodong Yu, Gowtham Ramesh, Jialian Wu, Zicheng Liu, Pan Lu, James Zou, and Jiaxuan You. Where LLM agents fail and how they can learn from failures. *arXiv preprint arXiv:2509.25370*, 2025.

A Additional Experimental Settings

A.1 Datasets and Splits

Table 5: Dataset partitions. MATH-500 and the LiveCodeBench test window are not used for training or model selection.

Dataset	Train	Validation	Evaluation
MATH	MATH train	held-out MATH train	MATH-500
LiveCodeBench	May 2023–June 2024	July–September 2024	October 2024–April 2025

Training continuations, candidate steps, and all learned parameters use only the training partition in Table 5. Validation data select checkpoints and hyperparameters; evaluation data are used only for reporting results.

MATH and MATH-500. The MATH training split contains 7,500 competition problems across seven subjects (Hendrycks et al., 2021). We reserve a fixed subset for validation and use the remaining problems for both stages of HyperGuide training. Evaluation uses all 500 problems in MATH-500 (Lightman et al., 2024).

LiveCodeBench. We use the code-generation portion of LiveCodeBench release v6, which contains problems dated from May 2023 through April 2025 (Jain et al., 2025). Release dates define the chronological train, validation, and test windows in Table 5. Evaluation uses all 341 code-generation problems released during the October 2024–April 2025 test window.

Reasoning format and verification. For MATH, a reasoning step is one numbered mathematical step, and the final boxed answer is checked by normalized symbolic equivalence. For LiveCodeBench, the model produces numbered algorithm-design steps followed by executable code; a reasoning step is one numbered planning step, and the final code is checked with the benchmark test harness.

A.2 Hyperparameters

Table 6 summarizes the hyperparameters used across the two training stages and at inference. The same settings are used for MATH and LiveCodeBench unless stated otherwise, with early stopping on the corresponding validation partition.

B Training and Inference Details

B.1 HyperGuide Implementation

All architecture and optimization hyperparameters are listed in Table 6. During Stage 2 and inference, the state encoder processes the plain-text reasoning state with LoRA disabled, while the decoder uses the Stage 2 adapter. The state encoder remains trainable during Stage 2.

Table 6: Hyperparameters used for HyperGuide training and inference.

Architecture	
Backbones	Qwen2.5-7B-Instruct; Llama-3.1-8B-Instruct
LoRA rank	16
LoRA α	32
LoRA dropout	0.05
LoRA target modules	$\{q, k, v, o\}$_proj of every attention layer
Projection head h_ϕ	2-layer MLP, hidden 1024, GELU activation
Hyperbolic embedding dim n	64
Directional guidance head q_ω	predicts a tangent vector from the state and problem-root embeddings
Up-projector g_ψ	2-layer MLP, hidden 2048, GELU activation
Up-projector output dim	backbone hidden size (3584 for Qwen2.5; 4096 for Llama-3.1)
Curvature c init	1.0 (learnable scalar)
Training hardware	4× NVIDIA H200 GPUs
Random seeds	4, 44, 444
Stage 1: state-geometry training	
Optimizer	AdamW, β_1=0.9, β_2=0.999, ϵ=10^{-8}
Learning rate	1e−3
Weight decay	0.01
LR schedule	cosine with linear warmup
Warmup steps	500
Epochs	20
Effective batch size	64 (4 GPUs × 16 per-GPU)
Ranking-loss margin	0.1
Gradient clipping	1.0
Mixed precision	bf16
Continuations per state K	32
Sampling temperature τ_{mc}	0.8
Nucleus p	0.95
Max tokens per continuation	1024
Failure cost C_{fail}	one plus the longest verified training continuation (logical steps)
Importance-weight decay η	0.95
Stage 2: directional-guidance training	
Optimizer	AdamW
Learning rate	5e−5 (Stage 2 trainable parameters)
Weight decay	0.0
LR schedule	cosine, warmup ratio 0.03
Epochs	5
Reasoning trajectories per training problem	8
Sampling temperature τ	0.7
Nucleus p	0.9
Max tokens per reasoning trajectory	1024
Guidance-preference weight λ_Δ	1.0
Directional margin γ_{dir}	0.1
Auxiliary-text weight λ_{text}	0.5
Geometry-retention weight λ_{keep}	0.1
Effective batch size	256
Mixed precision	bf16
Inference (HyperGuide and single-pass baselines)	
Decoding	greedy
Max new tokens	1024
Re-encode interval	every step boundary

B.2 Hyperbolic State Geometry

Distance and projection. For the Poincaré ball defined in Section 3, the geodesic distance is

$$d_{\mathbb{D}}(\mathbf{x}, \mathbf{y}) = \frac{1}{\sqrt{c}} \cosh^{-1}\left(1 + \frac{2c\|\mathbf{x} - \mathbf{y}\|^2}{(1 - c\|\mathbf{x}\|^2)(1 - c\|\mathbf{y}\|^2)}\right). \tag{9}$$

Distance to the origin is $(2/\sqrt{c}) \tanh^{-1}(\sqrt{c}\|\mathbf{x}\|)$. The projection head maps Euclidean hidden representations into the ball using the exponential map at the origin (Ganea et al., 2018b).

Objectives. The radial ranking and tree-metric objectives are given in Equations 1 and 2. Appendix B.5 describes the construction and weighting of these triplets.

B.3 Directional Guidance Training

Transition directions. The state and successor embeddings define the transition directions $\mathbf{u}_t^{(i)}$ of Section 4.3, while the guidance head predicts a tangent vector at the current state. Gradients through these directions update the state encoder during Stage 2. The transport in Equation 4 is computed in closed form on the Poincaré ball using gyrovector operations (Ganea et al., 2018b).

Guidance preference. The two likelihood margins in Equation 6 are evaluated with the same adapted decoder. Consequently, $\mathcal{L}_\Delta$ in Equation 7 isolates the change in candidate preference caused by inserting the guidance token.

Reasoning states used in Stage 2. Each epoch generates reasoning trajectories with the current model and constructs candidate transitions at the resulting reasoning states. Refreshing these states after each epoch keeps the guidance data aligned with the trajectories produced by the adapted decoder.

Continuous decoding. Inference uses one live decoder cache for the entire reasoning sequence. The prompt is prefilled, and its root embedding is computed once. At each completed logical step, the accumulated plain-text state is encoded separately, and the resulting guidance direction is converted to one virtual token and appended as one additional cached position. Autoregressive decoding then continues without re-prefilling the prefix. Logical step completion is task-aware and does not split multiline mathematical blocks. Step count, per-step token count, and total generated tokens have separate stopping limits. No candidate expansion, scoring, or reranking is performed at inference.

B.4 Auxiliary Text Loss

Let $\mathcal{D}_{\text{text}}$ contain pairs (s_t, y) where y is either a minimum-cost candidate step $a_t^{(i)}$ with $i \in \mathcal{P}(s_t)$ (Section 4.3) or a verified final-answer step. Candidate-step targets are taken only from states with at least one candidate that has a sampled verified continuation. The weight $\mu(y)$ allows these target types to receive different weights. With the virtual guidance token $\mathbf{v}_t$ of Equation 4, the token-level cross-entropy is

$$\mathcal{L}_{\text{NLL}} = -\mathbb{E}_{(s_t,y)\sim\mathcal{D}_{\text{text}}}\left[\mu(y) \sum_{j=1}^{|y|} \log p(y_j \mid s_t, \mathbf{v}_t, y_{<j})\right]. \tag{10}$$

We retain the Stage 1 geometry while updating the state encoder with

$$\mathcal{L}_{\text{Text}} = \mathcal{L}_{\text{NLL}} + \lambda_{\text{keep}}\mathcal{L}_{\text{Geom}}. \tag{11}$$

For a final-answer target, y contains the complete required answer form.

B.5 Continuation and Tree Construction Details

This section specifies the continuation estimator used to order Stage 2 candidate transitions.

Sampling and success-weighted cost. For each state s, let $\rho_1, \ldots, \rho_K$ be independent continuations sampled from the frozen backbone using the settings in Table 6. Let $b_i \in \{0, 1\}$ indicate whether ρ_i reaches a final answer accepted by the verifier $\mathcal{V}$, and let $L_i = |\rho_i|$ count logical reasoning steps. We define

$$\hat{d}(s) = \frac{1}{K} \sum_{i=1}^{K} \left[b_i L_i + (1 - b_i) C_{\text{fail}} \right], \tag{12}$$

where C_{fail} is set to one plus the longest verified continuation in the corresponding training set. Equivalently, $\hat{d}(s)$ averages successful-continuation length and the failure penalty according to their empirical frequencies. Assigning the fixed penalty rather than the observed length of an unsuccessful rollout prevents an early incorrect termination from receiving a low cost. Thus, shorter successful continuations receive lower cost, while an isolated short success is penalized by the failures in the remaining samples.

Tree distances. Pairwise tree distances for $\mathcal{L}_{\text{metric}}$ in Equation 2 are extracted from generated continuations. Two types of pairs are used:

1. **Pairs within one continuation.** Consecutive states along a single continuation ρ satisfy $d_{\mathcal{T}}(s_t, s_{t+k}) = k$.
2. **Pairs across continuations.** Two continuations $\rho^{(1)}, \rho^{(2)}$ that share a common prefix up to step t and then diverge yield $d_{\mathcal{T}}\big(s^{(1)}_{t+j},\, s^{(2)}_{t+k}\big) = j + k$.

Importance weighting. Trajectory-local distances become less reliable as the offset from the shared prefix grows, because the inferred tree structure is based on a finite sample of continuations. To down-weight distant pairs, each triplet (s_i, s_j, s_k) in $\mathcal{L}_{\text{metric}}$ receives the multiplicative weight

$$w_{ijk} \;=\; \eta^{\,d_{\mathcal{T}}(s_i,\, s_j) + d_{\mathcal{T}}(s_i,\, s_k)}, \tag{13}$$

where the decay factor η is specified in Table 6. This ensures that the loss is dominated by nearby, high-confidence distance estimates while still receiving gradient from longer-range pairs.

B.6 Baseline Implementations

Few-shot greedy. Few-shot greedy performs one greedy decode from the task prompt.

Self-Consistency (MATH). Following Wang et al. (2023), we generate five reasoning trajectories at temperature 0.7 and nucleus p=0.95 and use majority voting over normalized final answers.

Tree of Thoughts. Following Yao et al. (2023), each active reasoning state is expanded with three candidate next steps and scored by the backbone using a task-specific value prompt. Beam search retains the five highest-scoring states at each depth.

PT-SFT. Following Wang et al. (2024b), each reasoning step is assigned a discrete latent planning variable inferred from its contextualized representation, and the corresponding learned planning token is inserted at the beginning of the step. The model is fine-tuned to autoregressively generate each planning token before its reasoning step and the final answer; at inference, it likewise predicts a planning token before generating each step.

Outcome Value Model. Following Yu et al. (2024), a scalar head is trained on generated reasoning trajectories labeled by final-answer correctness. At inference, the head scores candidate next steps in a width-five beam search.

Coconut and SoftCoT. We use the public training procedures of Coconut (Hao et al., 2025) and SoftCoT (Xu et al., 2025), replacing their task data with the MATH or LiveCodeBench training partition. Their adapters are selected on the same validation data as HyperGuide and decoded with the same answer format and length limit.

Table 7: Additional ablations on Qwen2.5-7B-Instruct. All conditions are trained and evaluated with seed 4.

	MATH-500		LiveCodeBench	
Variant	Accuracy ↑	Tokens ↓	Pass@1 ↑	Tokens ↓
HyperGuide	77.4	352.7	25.2	373.5
Binary Stage 2	73.2	522.3	20.5	377.1
Ordinal-preference LoRA (w/o geometry or token)	74.8	397.4	22.0	379.1

Soft Thinking. Following Zhang et al. (2025), Soft Thinking forms continuous concept tokens from probability-weighted mixtures of token embeddings and requires no task-specific training.

Random seeds. For the training-based baselines, the seed controls training and any sampling at inference; for Self-Consistency and Tree of Thoughts, it controls the sampling of trajectories and candidate steps. Few-shot and Soft Thinking decode greedily without training, so their outputs do not depend on the seed, and we report a single run.

C Additional Experiments

C.1 Additional Ablation Studies

Binary Stage 2. Here we replace ordinal continuation costs with binary labels that distinguish successors with and without a verified continuation. This variant has a 4.2-point accuracy drop on MATH-500 and a 4.7-point Pass@1 drop on LiveCodeBench. It also substantially increases length on MATH-500, from 352.7 to 522.3 tokens. Binary labels can steer the model away from steps with no verified continuation, but they give no preference among successors with different empirical success rates or successful-continuation lengths.

Ordinal-preference LoRA (w/o geometry or token). This variant removes the state geometry and guidance token and instead trains the same decoder LoRA directly on the same rollout-ranked candidate pairs, increasing the likelihood margin of the lower-cost step over the higher-cost one. It reaches 74.8% accuracy on MATH-500 and 22.0% Pass@1 on LiveCodeBench, outperforming Binary Stage 2 by 1.6 and 1.5 points, respectively. However, it remains 2.6 points below HyperGuide on MATH-500 and 3.2 points below it on LiveCodeBench, while also generating longer outputs.