The Butterfly Effect in Reasoning Branch-Structured Distributional Memory for Stochastic Agents

Ishan Jindal* **Vivek Dhamale*†** **Mahesh Chandran**
Fujitsu Research of India
{ishan.jindal, mahesh.chandran}@fujitsu.com

Abstract

Memory-augmented reasoning agents typically store a single realized reasoning trajectory as the unit of experience. This abstraction is brittle under stochastic decoding: the same task can induce multiple plausible reasoning paths with different decompositions, assumptions, and failure modes. We introduce **Branch-Structured Distributional Memory** (BranchDIME), a framework for constructing memory from branch-structured trajectory sets rather than isolated traces. BranchDIME samples short reasoning prefixes, clusters them into prefix-induced branches, expands representative prefixes into full trajectories, and distills the resulting set into reusable memories containing positive strategies, anti-patterns, and contrastive rules. Across GPQA-DIAMOND and MATH500, BranchDIME improves over single-trajectory memory and alternative trajectory-selection baselines at comparable inference latency, yielding relative gains of 8.7% and 2.2% over single-trajectory memory, respectively. Coverage analysis shows that BranchDIME spans all discovered branches, compared with 20.0% coverage for consensus selection and 64.9% for random selection, while also achieving the highest downstream accuracy. BranchDIME further transfers across tasks and models, improving over single-trajectory memory by up to 15.4% on AIME25 and OLYMMATH. These results support a shift from *trajectory as experience* to *trajectory distribution as experience*: memory should capture the local reasoning landscape induced by stochastic generation, not merely one sampled path.

1 Introduction

Modern large language models (LLMs) solve complex tasks by generating chains of intermediate reasoning at test time Wei et al. [2022], Grattafiori et al. [2024], Yang et al. [2025], Guo et al. [2025]. A growing line of work uses these reasoning and interaction traces as reusable memory, allowing agents to accumulate experience, refine strategies, and improve future behavior Yao et al. [2022], Han et al. [2026], Xu et al. [2025], Yuen et al. [2025], Du [2026]. Reasoning banks Ouyang et al. [2026], experience libraries Cai et al. [2025], and metacognitive handbooks Didolkar et al. [2025] demonstrate that storing past reasoning can provide substantial gains.

Despite this progress, most memory systems treat a single realized trajectory as the basic unit of experience. This is a natural abstraction when reasoning is deterministic, but modern LLM reasoning is inherently stochastic: under the same input, decoding induces a distribution over possible reasoning trajectories. A single chain of thought is therefore not the reasoning process itself, but one sample from a broader local reasoning distribution. In this sense, LLM reasoning exhibits a loose analogy

*Equal contribution.
†Vivek is currently affiliated with observe.ai

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

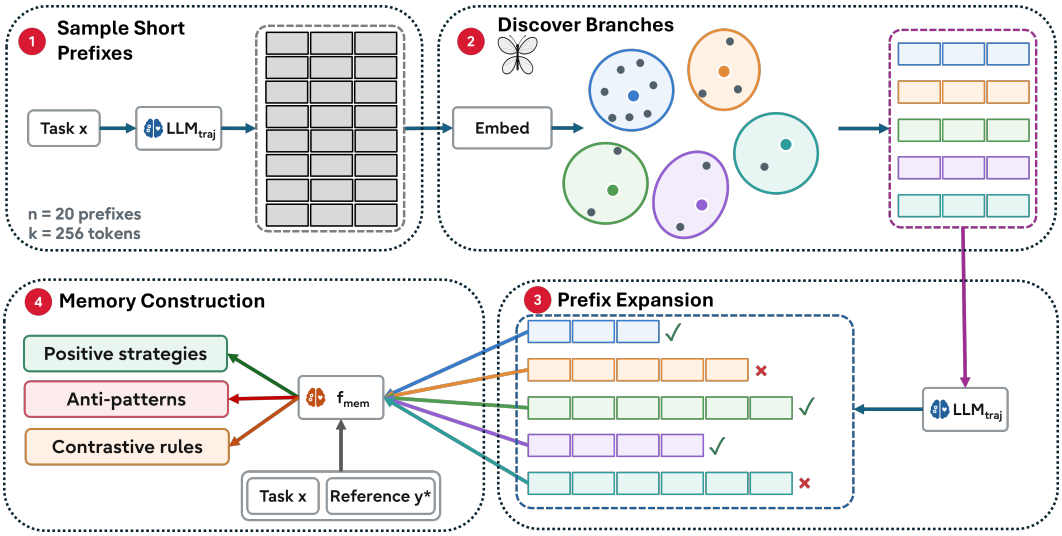


Figure 1: `BranchDIME` Overview.

to the "*butterfly effect*": small differences in early reasoning prefixes can steer generation toward qualitatively different solution paths, outcomes, and failure modes.

This distinction matters for memory. A single trajectory may contain useful reasoning principles, but it also contains sample-specific artifacts: incidental phrasing, arbitrary exploration order, local mistakes, and idiosyncratic solution choices. Memory constructed from one trace can therefore overfit to one realized path while missing alternative strategies and recurring failure modes. Sampling more trajectories can help, but unstructured multi-sampling is inefficient: many trajectories are redundant, and the memory generator can only consume a small number of full traces before context saturation.

We argue that the right object to store is not an isolated reasoning trace, but a structured approximation to the trajectory distribution induced by stochastic reasoning. We introduce **Branch-Structured Distributional Memory** (`BranchDIME`), a framework for constructing memory from branch-structured trajectory sets. For each memory-construction instance, `BranchDIME` first samples short reasoning prefixes from the trajectory distribution in Figure 1. It then clusters these prefixes to identify distinct prefix-induced branches and expands only representative prefixes into full trajectories. This decouples *branch discovery* from *trajectory expansion*: `BranchDIME` can explore a larger prefix pool while keeping the memory-generator input budget fixed. Given the expanded representative trajectories, `BranchDIME` distills memory as reusable reasoning constraints. The memory generator extracts three complementary forms of knowledge: *positive strategies* associated with successful reasoning, *anti-patterns* that characterize recurring failures, and *contrastive strategies* that distinguish successful and unsuccessful trajectories.

This view also explains why branch-structured distributional memory can transfer beyond the exact examples from which it is built. A memory constructed from only one trajectory captures a narrow solution path for one task. In contrast, a memory distilled from multiple branches captures a local neighborhood of reasoning strategies around the task: alternative approaches, common failure modes, and decision points that may remain relevant when future tasks differ from the original instance. Our cross-task and cross-model experiments test this hypothesis by constructing memories from MATH-$\mathcal{M}$ and evaluating them on disjoint competition-style benchmarks AIME25 and OLYMMATH.

Empirically, `BranchDIME` improves memory quality across reasoning benchmarks and model families. Under matched inference-time conditions, `BranchDIME` outperforms single-trajectory memory and available multi-trajectory baselines on GPQA-DIAMOND and MATH500. Coverage analyses show that `BranchDIME` selects trajectories spanning more discovered reasoning branches than random or consensus-based selection. Prefix clustering also closely approximates full-trajectory clustering while substantially reducing offline generation cost. Finally, branch-structured memories transfer across tasks, response models, and memory generators, suggesting that the method captures reusable reasoning abstractions rather than merely storing instance-specific traces. Our contributions are:

- We identify a limitation of trajectory-based memory: under stochastic reasoning, a single trajectory is an incomplete unit of experience.
- We propose `BranchDIME`, a memory-construction framework that discovers prefix-induced reasoning branches and expands representative prefixes into a structured trajectory set.
- We formulate memory construction as distillation over branch-structured trajectories, extracting positive strategies, anti-patterns, and contrastive strategies.
- We empirically show that `BranchDIME` improves over single-trajectory memory, covers more discovered reasoning branches, approximates full-trajectory clustering with lower offline generation cost, and transfers across tasks, response models, and memory generators.

We, `BranchDIME`, re-frame diversity in LLM reasoning. Multiple trajectories are not merely redundant samples to aggregate; they are observations of an underlying reasoning structure that can be discovered, compressed, and reused. This shift from *trajectory as experience* to *trajectory distribution as experience* offers a path toward more reliable and transferable memory for reasoning agents.

2 Branch-Structured Distributional Memory

We formalize `BranchDIME` as constructing memory from a *structured approximation to the distribution over reasoning trajectories* induced by a stochastic language model. We first define the trajectory distribution, then introduce prefix-based structure over partial trajectories, and finally describe memory construction from structured trajectory sets.

Trajectory Distribution. Let $x \in \mathcal{X}$ denote an input task. A stochastic LLM induces a distribution over *full reasoning trajectories* (chains of thought), represented as token sequences: $S = (w_1, w_2, \ldots, w_T), S \sim p_{\theta_{\text{trj}}}(S \mid x)$, where w_t are tokens and T is a stopping-time random variable determined by the emission of an end-of-sequence (EOS) token. The trajectory distribution factorizes autoregressively $p_{\theta_{\text{trj}}}(S \mid x) = \prod_{t=1}^{T} p_{\theta_{\text{trj}}}(w_t \mid x, w_{<t})$. We obtain a finite set of sampled trajectories by independent draws $\widehat{\mathcal{S}}(x) = \{S^{(i)}\}_{i=1}^{n}, S^{(i)} \sim p_{\theta_{\text{trj}}}(\cdot \mid x)$. Traditional memory methods construct memory $\mathcal{M}$ from a single sampled trajectory S. In contrast, our goal is to construct memory from a *structured set of trajectories* that better captures the diversity of reasoning under $p_{\theta_{\text{trj}}}(\cdot \mid x)$.

2.1 Prefix Space and Branch Structure

A direct way to construct memory from a trajectory distribution is to sample many complete trajectories from $p_{\theta_{\text{trj}}}(\cdot \mid x)$, cluster them into latent reasoning modes, and distill memory from representative trajectories. However, this is inefficient: full trajectories are expensive to generate, many continuations are redundant, and the memory generator can only consume a small fixed-size set of trajectories without context saturation.

`BranchDIME` therefore decouples *mode discovery* from *trajectory expansion*. We first sample short reasoning prefixes to estimate the branch structure of $p_{\theta_{\text{trj}}}(\cdot \mid x)$, motivated by prior work showing that reasoning trajectories often diverge into coherent modes early in generation Ji et al. [2025], Jindal et al. [2026]. Prefix-space clustering thus serves as a compute-efficient surrogate for full-trajectory clustering. After branch discovery, `BranchDIME` expands only representative prefixes into complete trajectories for memory construction. Consequently, all multi-trajectory memory methods can use the same memory-generator input budget of K full trajectories; they differ only in trajectories selection.

Motivated by this, we operate on *prefixes* of reasoning trajectories rather than complete trajectories. A prefix of length k is defined as $\pi_k = (w_1, \ldots, w_k)$, corresponding to the first k tokens of a trajectory S. Let Π_k denote the set of all valid prefixes of length k. The induced prefix distribution is

$$p_{\theta_{\text{trj}}}(\pi_k \mid x) = \sum_{S : S_{1:k} = \pi_k} p_{\theta_{\text{trj}}}(S \mid x).$$

For branch discovery, we sample prefixes directly from this distribution: $\pi_k^{(i)} \sim p_{\theta_{\text{trj}}}(\pi_k \mid x), i = 1, \ldots, n$. This yields the empirical prefix set $\widehat{\Pi}_k = \{\pi_k^{(i)}\}_{i=1}^{n}$. Our goal is to replace this unstructured set of prefixes with a small collection of *representative branches* that capture distinct modes of reasoning. To do so, each prefix π_k is mapped to a sparse vector representation $\varphi(\pi_k) \in \mathbb{R}^d$ using

TF–IDF features over prefix tokens[3]. We normalize each vector to unit norm, $\|\varphi(\pi_k)\|_2 = 1$, and use cosine similarity in this normalized space to cluster prefixes.

Clustering as branch discovery. We partition $\widehat{\Pi}_k$ into B clusters $\mathcal{C}_k = \{C_1, \ldots, C_B\}, \bigcup_{b=1}^{B} C_b = \widehat{\Pi}_k, C_b \cap C_{b'} = \varnothing \ (b \neq b')$. Each cluster is treated as an empirical approximation to a reasoning branch.

Fixed-B branch discovery. Our default procedure uses spherical K-means clustering with $B = 5$ clusters. Let $z_{ib} \in \{0, 1\}$ denote the assignment of prefix $\pi_k^{(i)}$ to cluster b, and $\mu_b \in \mathbb{R}^d$ denote the centroid. We solve:

$$\max_{\{z_{ib}\},\{\mu_b\}} \sum_{i=1}^{n} \sum_{b=1}^{B} z_{ib}\, \varphi(\pi_k^{(i)})^\top \mu_b \quad \text{s.t. } \|\mu_b\|_2 = 1,\ z_{ib} \in \{0,1\},\ \sum_{b=1}^{B} z_{ib} = 1.$$

After clustering, we select a representative prefix from each cluster by choosing the sample closest to the centroid: $i_b^* = \arg\max_{i: z_{ib}=1} \varphi(\pi_k^{(i)})^\top \mu_b, \widetilde{\pi}_b := \pi_k^{(i_b^*)}$.

Adaptive branch discovery. We also consider agglomerative clustering, which does not require pre-specifying B. Let $d(\pi, \pi')$ denote the distance in embedding space with $d(\pi, \pi') = 1 - \varphi(\pi)^\top \varphi(\pi')$. Agglomerative clustering iteratively merges clusters $(A^*, B^*) = \arg\min_{A \neq B} \mathrm{Link}(A, B)$, where $\mathrm{Link}(A, B)$ is a linkage criterion, such as average linkage with cosine distance. For Ward linkage, we apply Euclidean distance to the normalized TF–IDF vectors. Cutting the resulting dendrogram at threshold δ yields: $\mathcal{C}_k^{(\delta)} = \mathrm{CutDendrogram}(\widehat{\Pi}_k, d, \delta), |\mathcal{C}_k^{(\delta)}| = B_\delta$. This adaptive variant allocates more branches to inputs with diverse reasoning structure and fewer branches when reasoning is concentrated.

Representative-prefix expansion. Rather than expanding all prefixes, we expand only the representative prefixes. For each branch b, we generate full trajectories conditioned on $\widetilde{\pi}_b$:

$$S_b^{(j)} \sim p_{\theta_{\mathrm{trj}}}(S \mid x, S_{1:k} = \widetilde{\pi}_b), \qquad j = 1, \ldots, m_b.$$

In practice, this is implemented by prefix conditioning, i.e., initializing generation with $\widetilde{\pi}_b$ and continuing autoregressively. The resulting structured trajectory set $\mathcal{T}(x) = \bigcup_{b=1}^{B} \{S_b^{(1)}, \ldots, S_b^{(m_b)}\}$. Under the default configuration ($B = 5$, $m_b = 1$, so $K = 5$), `BranchDIME` produces up to five diverse trajectories per input for fixed-B branch discovery. The LLM used to generate trajectories for memory is labeled as $\mathrm{LLM}_{\mathrm{trj}}$.

2.1.1 Theoretical Justification: Mode Coverage

We provide a simple mode-coverage analysis explaining why prefix-based branch selection can be more informative than random multi-trajectory memory under the same expansion budget. Suppose that, for a fixed input x, the trajectory distribution decomposes into B latent reasoning modes $\mathcal{M}_1, \ldots, \mathcal{M}_B$ with probabilities $p_1, \ldots, p_B$, where $\sum_b p_b = 1$. A memory-construction method covers mode b if at least one selected trajectory comes from $\mathcal{M}_b$.

Proposition 1 (Coverage advantage of prefix branching)**.** *Consider a random-K trajectory memory that selects K independent full trajectories from $p_{\theta_{\mathrm{trj}}}(\cdot \mid x)$. The probability that it covers mode b is $P_{\mathrm{rand}}(b) = 1 - (1 - p_b)^K$. Now consider `BranchDIME` with n sampled prefixes. Assume that prefix clustering recovers every mode that appears in the prefix sample, and that one representative prefix from each discovered mode is expanded. Then the probability that `BranchDIME` covers mode b is $P_{\mathrm{BB}}(b) = 1 - (1 - p_b)^n$. Therefore, if $n \geq K$, then $P_{\mathrm{BB}}(b) \geq P_{\mathrm{rand}}(b)$ for every mode b, with strict inequality whenever $n > K$ and $p_b \in (0, 1)$.*

Corollary 1 (Rare-mode advantage)**.** *For rare modes with $p_b \ll 1$,*

$$1 - (1 - p_b)^m \approx m p_b.$$

Thus, `BranchDIME` provides an approximately n/K multiplicative advantage over random-K selection in discovering rare reasoning modes. In our default setting, $n = 20$ and $K = 5$, giving an idealized $4\times$ rare-mode coverage advantage.

[3]We use TF-IDF following Jindal et al. [2026]. While simple, we find that the resulting clusters correspond to distinct downstream outcomes and reasoning modes, as shown in Section 4 and Appendix E.

This analysis explains why BranchDIME is especially useful when the local trajectory distribution is skewed: random selection can miss minority modes, while prefix branching gives each mode more opportunities to be discovered before the limited full-trajectory expansion budget is spent. Appendix B gives proofs and extends the analysis to additive memory utility.

2.2 Memory Construction from Trajectory Sets

Given the structured trajectory set $\mathcal{T}(x)$, we construct memory by extracting reusable reasoning constraints. We formalize memory construction as: $\mathcal{M}(x) = f_{\text{mem}}\big(x, y^*, \mathcal{T}(x)\big)$, where y^* is the reference answer and f_{mem} is an LLM-based abstraction operator applied to the entire trajectory set.

Next, we extract reusable knowledge by comparing multiple reasoning trajectories. By jointly analyzing $\mathcal{T}(x)$, the model identifies: **Positive strategies:** reasoning patterns associated with correct outcomes; **Anti-patterns:** recurring failure modes in incorrect reasoning; **Contrastive strategies:** structural differences between successful and failing trajectories.

The memory function f_{mem} can be viewed as a compression operator over the trajectory set, where compression aims to capture reasoning-relevant invariants and failure modes while discarding trajectory-specific noise. Specifically, the quality of $\mathcal{M}(x)$ depends on the diversity and structure of $\mathcal{T}(x)$, which BranchDIME constructs via prefix-based branching.

GPT-5-Mini Singh et al. [2025] is the by default memory generator (unless otherwise stated) with the memory extraction prompt described in Appendix I.1. We hold both the generator and the prompt fixed across all memory-based methods. This design isolates the effect of trajectory-set structure on memory quality. Offline memory construction cost is reported separately because memories are built once and reused at inference time.

2.3 Inference with Retrieved Memory

We store the generated memories $\mathcal{M}(x)$ during training. At inference time, we retrieve memories conditioned on the input task and use them as global guidance for response generation. Given an input x, we retrieve the top-r most relevant memories using cosine similarity: $\mathcal{M}^*(x) = \text{Retrieve}(x)$, with $r = 3$ unless otherwise specified. The retrieved memories are incorporated by augmenting the model context during generation:

$$p_{\theta_{\text{res}},\mathcal{M}}(w_{t+1} \mid x, w_{\leq t}) = p_{\theta_{\text{res}}}(w_{t+1} \mid x, w_{\leq t}, \mathcal{M}^*(x)). \tag{1}$$

This provides high-level guidance in the form of positive strategies, anti-patterns, and contrastive rules distilled from prior trajectory distributions. The LLM used to generate the final response is denoted by LLM_{res}.

3 Experiments

We evaluate BranchDIME to understand the role of trajectory structure in memory construction. Our experiments are designed to answer three questions: **Q1:** Does modeling trajectory sets improve performance over trajectory-centric memory? **Q2:** How does trajectory diversity affect reasoning under stochastic decoding? **Q3:** Does structured exploration provide better efficiency than unstructured sampling?

Backbone LLMs. We evaluate the efficiency and generality of our method across diverse open-source LLMs spanning different architectures, scales, and training paradigms. Specifically, we use **DeepSeek-R1-Distill-Qwen-7B (DSQ-7B)**, **DeepSeek-R1-Distill-Llama-8B (DSL-8B)** Guo et al. [2025], distilled from reasoning-specialized LLMs; **QWQ-32B** Team [2025b], Yang et al. [2024a], a Qwen2.5 variant trained with reinforcement learning for problem solving; and **Qwen2.5-instruct-7B (Q2.5-7B)** Yang et al. [2024b], a math-specialized instruction-tuned model. These choices cover architectures (Qwen, DeepSeek), parameter scales (7B–32B), and training paradigms (distillation, RL, supervised fine-tuning).

Benchmarks. We evaluate on multi-step reasoning tasks: MATH500 Lightman et al. [2024], a set of 500 challenging math problems; AIME25 AIME [2025], high-school olympiad-level math problems; GPQA-DIAMOND Rein et al. [2024], a graduate-level STEM reasoning benchmark covering physics, chemistry, and biology; and OLYMMATH Sun et al. [2026].

Table 1: Performance comparison of BranchDIME against different baselines. Here $\text{LLM}_{\text{trj}} = \text{LLM}_{\text{res}}$ and η represents response generation latency.

	GPQA-$\mathcal{M}$ on GPQA-DIAMOND					MATH-$\mathcal{M}$ on MATH500				
Memory	DSQ-7B	Q2.5-7B	DSL-8B	avg acc	avg η(s)	DSQ-7B	Q2.5-7B	DSL-8B	avg acc	avg η(s)
Single Response (CoT)										
No Memory	0.491	0.333	0.406	0.410	8.68	0.808	0.728	0.831	0.789	6.55
Self-Consistency@5 (CoT)										
No Memory	0.525	0.343	0.470	0.446	21.66	0.865	0.756	0.866	0.829	11.22
Single $\mathcal{T}$	0.533	0.349	0.505	0.462	23.20	0.882	0.764	0.890	0.845	17.34
Random $\mathcal{T}$	0.535	0.342	0.512	0.463	23.60	0.890	0.767	0.886	0.848	17.52
Consensus $\mathcal{T}$	0.530	0.347	0.485	0.454	95.60	0.891	0.754	0.889	0.845	71.04
BranchDIME	**0.581**	**0.392**	**0.534**	**0.502**	23.84	**0.896**	**0.795**	**0.902**	**0.864**	17.91

Baselines. We compare BranchDIME against controlled memory-construction baselines. All memory-based methods use the same memory data, retrieval mechanism, memory generator, and inference protocol; they differ in how trajectories are selected for memory construction. **No Memory** uses standard chain-of-thought reasoning without external memory. **Single** $\mathcal{T}$ constructs memory from one sampled trajectory per instance. **Random** $\mathcal{T}$ constructs memory from $K = 5$ randomly sampled trajectories, testing whether gains come merely from using multiple samples. **Consensus** $\mathcal{T}$ selects trajectories from the most frequent final-answer group among a pool of generated trajectories, testing whether the model's dominant answer mode provides sufficient memory signal. BranchDIME, samples $n = 20$ prefixes, clusters them into prefix-induced branches, and expands representative prefixes to obtain $K = 5$ trajectories. Detailed descriptions of these plus additional baselines are provided in Appendix C Table 10.

Evaluation Metrics. We follow standard metrics from prior reasoning literature: **Pass@1** on MATH500, AIME25 and OLYMMATH; **Accuracy** (binary correctness) on GPQA-DIAMOND. We report inference latency η, measured as retrieval plus final response generation time.

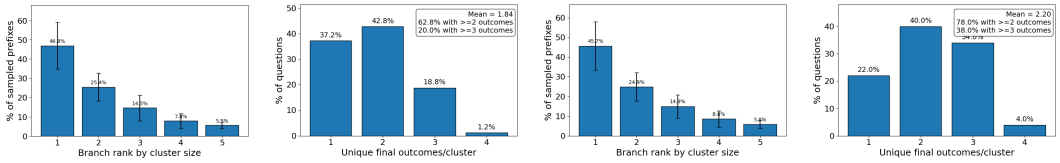
BranchDIME **Configuration.** All experiments are repeated 3 times[4]. with different random seeds and we report mean performance across runs for all metrics: accuracy, EM, Pass@1, and latency. Unless otherwise specified, BranchDIME uses: Prefix length $k = 256$; Number of sampled prefixes $n = 20$; Fixed clustering with $B = 5$; One trajectory expansion per cluster ($m_b = 1$). We additionally evaluate an adaptive clustering variant in ablations. Detailed experimental setting is in Appendix D

Datasets for Memory Construction. We construct memory using datasets that are disjoint from evaluation benchmarks to avoid leakage. For GPQA, we start from the GPQA main dataset and remove all samples that overlap with the GPQA-DIAMOND evaluation set. The remaining subset consists of 250 samples, which are used exclusively for memory construction, denoted as GPQA-$\mathcal{M}$. For mathematics, we construct the memory set from Hendrycks et al. [2021]. To ensure coverage across problem types, we perform stratified sampling over subject categories and levels (e.g., algebra, geometry, combinatorics) and randomly select 200 samples, denoted as MATH-$\mathcal{M}$. We explicitly ensure that no samples from the MATH500 evaluation set appear in the memory construction set.

4 Results

$\Longrightarrow$ **Main Results.** Table 1 compares memory construction from different parts of the agent's trajectory distribution on GPQA-DIAMOND and MATH500. We evaluate a standard single-response chain-of-thought baseline, and several memory-construction strategies under the same self-consistency setting. Across both benchmarks, BranchDIME **achieves the strongest performance**. On GPQA-DIAMOND, BranchDIME improves average accuracy to **0.502**, compared to 0.462 for Single $\mathcal{T}$ and 0.446 for self-consistency without memory. On MATH500, BranchDIME reaches **0.864** average accuracy, compared to 0.845 for Single $\mathcal{T}$ and 0.829 for self-consistency without memory. These gains come with similar inference latency to single-trajectory memory: 23.20s Vs 23.84s on GPQA-

[4]For readability, the main table reports mean performance, while standard deviations across three runs are provided in Appendix F.



(a) Branch mass (b) Outcome diversity (d) Branch mass (e) Outcome diversity

(c) DSL-8B (f) Q2.5-7B

Figure 2: **Prefix branches capture nontrivial mass and distinct outcomes.** For each GPQA-$\mathcal{M}$ instance, across both models, the largest branch contains less than half of sampled prefixes on average, while selected representatives often yield multiple final outcomes.

DIAMOND, and 17.34s Vs 17.91s on MATH500. Thus, the improvement is not explained by substantially higher inference-time cost.

The results show that **memory quality depends on which parts of the trajectory distribution are used**. Single $\mathcal{T}$ improves over self-consistency without memory, confirming that trajectory-derived memory is useful. However, Random $\mathcal{T}$ gives only marginal gains over Single $\mathcal{T}$ despite using multiple trajectories. This suggests that simply adding more trajectories is not sufficient: randomly sampled trajectories can be redundant or fail to cover distinct reasoning modes. Consensus $\mathcal{T}$ also fails to improve over Single $\mathcal{T}$ and in some cases underperforms Random $\mathcal{T}$, indicating that the agent's highest-probability behaviors do not necessarily provide the most informative memory.

In contrast, `BranchDIME` **consistently outperforms both Random $\mathcal{T}$ and Consensus $\mathcal{T}$**. By selecting representative trajectories from distinct prefix-induced branches, `BranchDIME` captures a broader set of possible reasoning modes for the same task. This produces memories that encode not only what the agent happened to do in one trajectory, but how different plausible trajectories succeed or fail. These results support the central hypothesis of `BranchDIME`: under stochastic reasoning, **a single trajectory is an incomplete unit of experience**. Memory improves when it is constructed from a structured approximation to the trajectory distribution, rather than from one realized trajectory, random additional samples, or only the Consensus reasoning mode.

Memory construction cost. Memory construction is an offline, one-time cost amortized across inference. `BranchDIME` samples n prefixes of length k and continues K selected prefixes, giving cost $nk + \sum_{i=1}^{K}(t_i - k)$, where t_i is the total token length of the i-th expanded trajectory. Random $\mathcal{T}$ samples K full trajectories directly, with cost $\sum_{i=1}^{K} t_i$. Single $\mathcal{T}$ samples one full trajectory, with cost t_1. Consensus $\mathcal{T}$ requires a pool of n fully generated trajectories to identify the most frequent final-answer group, with cost $\sum_{i=1}^{n} t_i$. Thus, `BranchDIME` is more expensive than Random by approximately $(n-K)k$ prefix-exploration tokens, but substantially cheaper than full-trajectory selection methods that generate all n candidates to completion.

$\Longrightarrow$ **Prefix Clustering Approximates Full-Trajectory Clustering.** A natural alternative to `BranchDIME` is to generate the full pool of trajectories, cluster complete trajectories, and select representative trajectories for memory construction. This directly implements trajectory-distribution memory, but pays full generation cost for every candidate before selection. We evaluate this full-trajectory clustering baseline against `BranchDIME`, which clusters prefixes and expands only the selected representatives. As shown in Table 2, prefix clustering closely matches full-trajectory clustering, trailing by only 0.005 accuracy while reducing pre-memory generation tokens from 26.8M to 8.0M, a 70.2% reduction. This supports the view that **prefix clustering is an efficient surrogate for full-trajectory clustering**: it preserves nearly all of the memory-construction benefit while avoiding redundant full-trajectory generation for candidates that are not ultimately selected.

Table 2: Prefix versus full-trajectory clustering on GPQA-DIAMOND.

Method	Tokens	DSL-8B
Full-Traj Cluster $\mathcal{T}$	26.8M	0.539
`BranchDIME`	8.0M	0.534

$\Longrightarrow$ `BranchDIME` **Covers Diverse and Nontrivial Reasoning Branches.** We next examine whether `BranchDIME` covers the discovered prefix-induced branch structure. We measure coverage rate, pairwise cosine distance among embeddings of the selected full trajectories, and normalized entropy

over selected branch assignments. Table 3 shows that Single $\mathcal{T}$ and Consensus $\mathcal{T}$ collapse to one branch, while Random $\mathcal{T}$ covers more branches but remains incomplete. `BranchDIME` covers all discovered branches by construction, achieving the highest coverage, entropy, and pairwise diversity. This ordering mirrors downstream accuracy on GPQA-DIAMOND: Consensus $\mathcal{T}$ performs worst among memory methods, Random $\mathcal{T}$ improves with broader coverage, and `BranchDIME` performs best. Pairwise diversity is also positively associated with accuracy across the three multi-trajectory strategies (Pearson $r = 0.955$).

Since `BranchDIME` is designed to select one representative per discovered branch, full coverage alone could be vacuous. We therefore analyze whether the discovered branches have nontrivial mass and lead to different downstream behavior. Figure 2 shows that the prefix distribution is skewed but not collapsed: the largest branch contains only 46.8% of prefixes for DSL-8B and 45.7% for Q2.5-7B on average. Thus, selecting only the consensus branch would discard more than half of the observed prefix distribution.

Table 3: Trajectory-distribution coverage on the memory-construction set on GPQA-$\mathcal{M}$ DSL-8B.

Method	Coverage Rate ↑	Pairwise Diversity ↑	Entropy ↑	Acc. ↑
Single $\mathcal{T}$	0.200	–	0.000	0.505
Random $\mathcal{T}$	0.649	0.273	0.657	0.512
Consensus $\mathcal{T}$	0.200	0.141	0.000	0.485
`BranchDIME`	**1.000**	**0.301**	**1.000**	**0.534**

The selected branch representatives also induce diverse outcomes. Among the five expanded representatives, DSL-8B produces an average of 1.84 unique final outcomes per instance, with 62.8% of instances producing at least two outcomes and 20.0% producing three or more. Q2.5-7B produces an average of 2.20 unique outcomes, with 78.0% of instances producing at least two outcomes and 38.0% producing three or more. These results suggest that the discovered branches capture non-dominant behaviors that often correspond to different final outcomes, supporting the use of structured trajectory sets for memory construction. Appendix E provides qualitative examples.

$\Longrightarrow$ **Structured Memory Transfers Across Tasks and Models.** We next evaluate whether `BranchDIME` memories capture reusable reasoning structure beyond the examples from which they are constructed. We build memories from MATH-$\mathcal{M}$ and evaluate them on the disjoint competition-style benchmarks AIME25 and OLYMMATH. We also vary the trajectory-generation model, LLM_{trj}, and the response model, LLM_{res}, to test whether branch-structured memories transfer across model boundaries.

Table 4: Cross-model transfer from MATH-$\mathcal{M}$ to AIME25 and OLYMMATH.

	LLM_{trj}	LLM_{res}	No Memory	Single $\mathcal{T}$	`BranchDIME`
AIME25	DSQ-7B	DSQ-7B	0.358	0.383	**0.442**
		QWQ-32B	0.633	0.633	**0.666**
	QWQ-32B	DSQ-7B	0.358	0.433	**0.458**
		QWQ-32B	0.633	0.700	**0.733**
OLYMMATH	DSQ-7B	DSQ-7B	0.430	0.450	**0.515**
		QWQ-32B	0.780	**0.800**	**0.800**
	QWQ-32B	DSQ-7B	0.430	0.455	**0.525**
		QWQ-32B	0.780	0.810	**0.840**

Table 4 shows that `BranchDIME` matches or improves over Single $\mathcal{T}$ in every transfer setting. The gains are largest when DSQ-7B is used as the response model, suggesting that smaller models benefit more from branch-structured guidance than from a single stored trajectory. Memories generated by the stronger QWQ-32B also transfer effectively to DSQ-7B, indicating that stronger trajectory generators can provide useful branch-level abstractions for weaker response models.

When QWQ-32B is used as the response model, `BranchDIME` still improves over Single $\mathcal{T}$ in most settings, but the gains are smaller. This suggests that stronger response models may already recover some reasoning structure during generation, reducing the marginal benefit of additional memory diversity. Overall, these results indicate that `BranchDIME` **memories encode transferable reasoning abstractions** rather than only instance-specific or model-specific trajectory artifacts.

5 Ablations

$\Longrightarrow$ **Prefix Pool Size Ablation.** We study how the prefix pool size n affects `BranchDIME` while keeping the expansion budget fixed at $K = 5$ for all $n \geq 5$. The $n = 0$ and $n = 1$ rows correspond to no-memory and single-trajectory memory baselines, respectively. Increasing n gives `BranchDIME` more opportunities to discover rare or minority reasoning modes, while fixing K keeps the number of expanded trajectories and the memory-generator input size unchanged. Thus, improvements for $n \geq 5$ reflect better branch discovery rather than providing more full trajectories.

Table 5: Prefix pool size ablation on GPQA-DIAMOND.

n	DSQ-7B	DSL-8B
0	0.525	0.470
1	0.533	0.505
5	0.535	0.512
20	0.581	0.534
50	0.576	0.531
70	0.576	0.540
100	0.596	0.531

Table 5 shows that increasing the prefix pool substantially improves performance from $n = 5$ to $n = 20$. At $n = 5$, the prefix pool equals the expansion budget, leaving little opportunity for selective branch discovery. With $n = 20$, both DSQ-7B and DSL-8B improve, suggesting that a larger prefix pool exposes more useful reasoning modes. Increasing n beyond 20 does not yield consistent gains: DSQ-7B reaches its best result at $n = 100$, while DSL-8B peaks at $n = 70$ and then decreases slightly. We therefore use $n = 20$ as a cost-effective default that captures most of the benefit while limiting offline prefix-sampling cost.

$\Longrightarrow$ **Open-Weight Memory Generator Ablation.** We evaluate whether `BranchDIME`'s gains depend on the proprietary memory generator used in the main experiments. Table 6 repeats the GPQA-DIAMOND memory-construction experiment while replacing GPT-5-Mini with open-weight memory generators: **Qwen3-30B-A3B-Thinking-2507** (`Q3-30B`) Team [2025a] and **Llama-3.1-70B-Instruct** (`L3-70B`) Grattafiori et al. [2024]. The trajectory-generation and response-generation models are fixed.

Table 6: Ablation of the memory generator on GPQA-DIAMOND. Here $\text{LLM}_{\text{trj}} = \text{LLM}_{\text{res}}$ while varying the memory generator.

f_{mem}	**Memory**	DSQ-7B	DSL-8B
`GPT-5-Mini`	Single $\mathcal{T}$	0.533	0.505
	`BranchDIME`	**0.581**	**0.534**
`Q3-30B`	Single $\mathcal{T}$	0.527	0.502
	`BranchDIME`	**0.539**	**0.520**
`L3-70B`	Single $\mathcal{T}$	0.515	0.515
	`BranchDIME`	**0.530**	**0.520**

`BranchDIME` improves over Single $\mathcal{T}$ for all memory generators across both response models. The gains are largest with `GPT-5-Mini`, but the open-weight results show that `BranchDIME`'s benefit is not solely due to the proprietary memory generator. Instead, branch-structured trajectory selection remains useful across different memory generators, though the magnitude depends on the generator used.

$\Longrightarrow$ **Adaptive Clustering Ablation.** We evaluate whether every memory-construction instance requires the same number of expanded branches. The default `BranchDIME` configuration uses fixed-B clustering with $B = 5$, yielding $250 \times 5 = 1250$ expanded trajectories on the 250-instance memory-construction set.

Table 7: Adaptive clustering ablation on the memory-construction set.

Method	δ	Total reps.	Avg. branches	DSL-8B
No Memory	–	–	–	0.470
Single $\mathcal{T}$	–	250	1.00	0.505
`BranchDIME`				
Fixed-B	–	1250	5.00	0.534
Agglomerative	0.20	1125	4.50	0.532
Agglomerative	0.30	956	3.82	0.523

Table 7 shows that adaptive clustering preserves most of `BranchDIME`'s benefit while reducing the number of expanded trajectories. With $\delta = 0.20$, agglomerative `BranchDIME` reduces expansions by 10.0% and nearly matches fixed-B performance, achieving 0.532 versus 0.534. With $\delta = 0.30$, it reduces expansions by 23.5% while retaining 0.523 accuracy, still above Single $\mathcal{T}$ at 0.505. Thus, adaptive clustering provides a useful accuracy–cost trade-off, while fixed-B remains the best configuration.

$\Longrightarrow$ **Sensitivity to Prefix Representation.** We examine whether `BranchDIME`'s gains depend on the representation used for prefix clustering. We replace TF–IDF features with Jina semantic embeddings Sturua et al. [2024], while keeping the clustering procedure, trajectory-expansion budget, and remaining pipeline fixed.

Table 8: Prefix representation ablation on GPQA-Diamond with Self-Consistency@5. We compare TF–IDF and Jina embeddings for BranchDIME while holding the remaining pipeline fixed. The last row reports the mean across response models. Best accuracy in each row is shown in bold.

Response Model	No Memory	TF–IDF	Jina
DSQ-7B	0.525	**0.581**	0.574
DSL-8B	0.470	0.534	**0.539**
Average	0.498	**0.558**	0.557

Table 8 shows comparable downstream accuracy with both representations: mean accuracy is 0.558 with TF–IDF and 0.557 with Jina, a difference of 0.1 percentage points. Jina improves DSL-8B accuracy by 0.005 but reduces DSQ-7B accuracy by 0.007 relative to TF–IDF, providing no consistent advantage across the two models. Both representations outperform the no-memory baseline. These results suggest that BranchDIME's gains in this setting are not specific to TF–IDF. We retain TF–IDF as a lightweight default because it achieves comparable accuracy without requiring an additional neural encoder.

$\Longrightarrow$ **Contribution of Memory Types.** BranchDIME jointly uses positive, negative, and contrastive memories. To assess the contribution of this combination, we compare the full method with variants that use only one memory type, keeping the retriever and inference procedure fixed.

Table 9: Contribution of memory types on GPQA-Diamond and MATH500. Positive, Negative, and Contrastive denote variants using only the specified memory type; BranchDIME uses all three. Best accuracy in each row is shown in bold.

Benchmark	Model	Positive	Negative	Contrastive	BranchDIME
GPQA-Diamond	DSQ-7B	0.529	0.540	0.544	**0.581**
	Q2.5-7B	0.355	0.372	0.391	**0.392**
	DSL-8B	0.510	0.505	0.517	**0.534**
MATH500	DSQ-7B	0.884	0.872	0.884	**0.896**
	Q2.5-7B	0.763	0.751	0.733	**0.795**
	DSL-8B	0.888	0.880	0.890	**0.902**

Combining memory types yields the highest accuracy in all six settings. As shown in Table 9, the full method improves over the strongest individual memory type by 0.1–3.7 percentage points. The largest gains occur for DSQ-7B on GPQA-Diamond (0.581 vs. 0.544) and Q2.5-7B on MATH500 (0.795 vs. 0.763). Among individual types, contrastive memory performs best across all models on GPQA-Diamond. On MATH500, positive memory performs best for Q2.5-7B, contrastive memory performs best for DSL-8B, and the two tie for DSQ-7B. Thus, no individual memory type consistently matches the joint configuration, supporting the use of all three types in BranchDIME.

6 Conclusion and Future Work

We introduced BranchDIME, a framework that constructs reasoning memory from structured trajectory distributions rather than isolated reasoning traces. By turning reasoning diversity into reusable positive, negative, and contrastive memories, BranchDIME depicts a shift from *trajectory as experience* to *trajectory distribution as experience*.

Our results show that prefix-level branching is an effective and compute-efficient way to approximate the local distribution of stochastic reasoning trajectories under a fixed memory-construction budget. At the same time, BranchDIME should be viewed as one instantiation of a broader design space for **distributional memory** construction. Future work should further study how different prefix-sampling strategies, clustering objectives, representative-selection methods, and branch-expansion policies affect memory quality, transferability, and downstream reasoning behavior.

References

2025 AIME. Art of problem solving. https://artofproblemsolving.com/wiki/index.php/2025_AIME_I, 2025. [Online; accessed 2025].

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

Zhicheng Cai, Xinyuan Guo, Yu Pei, Jiangtao Feng, Jinsong Su, Jiangjie Chen, Ya-Qin Zhang, Wei-Ying Ma, Mingxuan Wang, and Hao Zhou. Flex: Continuous agent evolution via forward learning from experience. *arXiv preprint arXiv:2511.06449*, 2025.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation, 2024.

Aniket Didolkar, Nicolas Ballas, Sanjeev Arora, and Anirudh Goyal. Metacognitive reuse: Turning recurring llm reasoning into concise behaviors. *arXiv preprint arXiv:2509.13237*, 2025.

Pengfei Du. Memory for autonomous llm agents: Mechanisms, evaluation, and emerging frontiers. *arXiv preprint arXiv:2603.07670*, 2026.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Nathan Habib, Clémentine Fourrier, Hynek Kydlíček, Thomas Wolf, and Lewis Tunstall. Lighteval: A lightweight framework for llm evaluation, 2023. URL https://github.com/huggingface/lighteval.

Dongge Han, Camille Couturier, Daniel Madrigal Diaz, Xuchao Zhang, Victor Rühle, and Saravan Rajmohan. Legomem: Modular procedural memory for multi-agent llm systems for workflow automation. AAMAS '26, page 3116–3118, Richland, SC, 2026. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9798400723179. doi: 10.65109/VLUA1303. URL https://doi.org/10.65109/VLUA1303.

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

Ke Ji, Jiahao Xu, Tian Liang, Qiuzhi Liu, Zhiwei He, Xiaoyuan Liu, Xingyu Chen, Junying Chen, Benyou Wang, Zhaopeng Tu, et al. The first few tokens are all you need: An efficient and effective unsupervised prefix fine-tuning method for reasoning models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Ishan Jindal, Jayant Taneja, Chandana Badrinath, Vikas Kapur, and Sachin Dev Sharma. Offloaded reasoning: Efficient inference for large language models via modular reasoning and refinement. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 7450–7458, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.393. URL https://aclanthology.org/2025.findings-emnlp.393/.

Ishan Jindal, Sai Prashanth Akuthota, Jayant Taneja, and Sachin Sharma. The path of least resistance: Guiding llm reasoning trajectories with prefix consensus. In *International Conference on Learning Representations*, volume 2026, pages 54914–54934, 2026.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pages 39578–39601, 2024.

Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, et al. Reasoningbank: Scaling agent self-evolving with reasoning memory. In *International Conference on Learning Representations*, volume 2026, pages 94327–94354, 2026.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22, 2023.

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

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Saba Sturua, Isabelle Mohr, Mohammad Kalim Akram, Michael Günther, Bo Wang, Markus Krimmel, Feng Wang, Georgios Mastrapas, Andreas Koukounas, Nan Wang, and Han Xiao. jina-embeddings-v3: Multilingual embeddings with task lora, 2024. URL https://arxiv.org/abs/2409.10173.

Haoxiang Sun, Yingqian Min, Zhipeng Chen, Wayne Xin Zhao, and Ji-Rong Wen. Challenging the boundaries of reasoning: An olympiad-level math benchmark for large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 17438–17457, 2026.

Qwen Team. Qwen3 technical report, 2025a. URL https://arxiv.org/abs/2505.09388.

Qwen Team. Qwq-32b: Embracing the power of reinforcement learning, March 2025b. URL https://qwenlm.github.io/blog/qwq-32b/.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=ehfRiF0R3a.

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 *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 17577–17604. Curran Associates, Inc., 2025. doi: 10.52202/085713-0593. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/19909c36f51abc4856b4560aff3d36d6-Paper-Conference.pdf.

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*, 2024a.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024b.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv e-prints*, pages arXiv–2505, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022.

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

Sizhe Yuen, Francisco Gomez Medina, Ting Su, Yali Du, and Adam J Sobey. Intrinsic memory agents: Heterogeneous multi-agent llm systems through structured contextual memory. *arXiv preprint arXiv:2508.08997*, 2025.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 19724–19731, 2024.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=WZH7099tgfM`.

A Algorithms

In this appendix we provide pseudocode for both the memory generation and response generation in 1 and 2, respectively.

B Theoretical Analysis

This appendix provides the proofs for the mode-coverage analysis in Section 2.1.1. The goal is not to fully characterize the semantics of prefix clusters, but to formalize the basic advantage of `BranchDIME`: it uses a larger number of inexpensive prefix samples to discover local reasoning modes before spending the limited budget of full trajectory expansions.

B.1 Setup

Fix an input x. We assume that the trajectory distribution $p_{\theta_{\mathrm{trj}}}(S \mid x)$ decomposes into B latent reasoning modes

$$\mathcal{M}_1, \ldots, \mathcal{M}_B,$$

Algorithm 1 `BranchDIME` Memory Generation

Require: Memory-construction dataset $\mathcal{D}_{\mathrm{mem}} = \{(x_i, y_i^\star)\}_{i=1}^N$
Require: Trajectory-generation model $\mathrm{LLM}_{\mathrm{trj}}$
Require: Memory generator f_{mem}
Require: Prefix pool size n, prefix length k, number of branches B, expansion budget K. By Default $B = K$.
Ensure: Memory bank $\mathcal{M}$

Initialize memory bank $\mathcal{M} \leftarrow \emptyset$
for each memory-construction instance $(x_i, y_i^\star) \in \mathcal{D}_{\mathrm{mem}}$ **do**
  Sample n reasoning prefixes:

$$\widehat{\Pi}_k(x_i) = \{\pi_k^{(1)}, \ldots, \pi_k^{(n)}\}, \qquad \pi_k^{(j)} \sim p_{\theta_{\mathrm{trj}}}(\pi_k \mid x_i)$$

  Embed each prefix using $\phi(\cdot)$:

$$z_j \leftarrow \phi(\pi_k^{(j)}), \qquad j = 1, \ldots, n$$

  Cluster prefix embeddings into B prefix-induced branches:

$$\mathcal{C}_i = \{C_1, \ldots, C_B\}$$

  for each branch $C_b \in \mathcal{C}_i$ **do**
    Select representative prefix closest to the branch centroid:

$$\widetilde{\pi}_b \leftarrow \arg\max_{\pi_k^{(j)} \in C_b} \phi(\pi_k^{(j)})^\top \mu_b$$

  end for
  Let $\widetilde{\Pi}_i = \{\widetilde{\pi}_1, \ldots, \widetilde{\pi}_B\}$
  Select at most K representative prefixes from $\widetilde{\Pi}_i$
  Expand each selected representative prefix into a full trajectory:

$$S_b \sim p_{\theta_{\mathrm{trj}}}(S \mid x_i, S_{1:k} = \widetilde{\pi}_b)$$

  Form the branch-structured trajectory set:

$$\mathcal{T}(x_i) = \{S_b\}_{b=1}^K$$

  Distill reusable memory from the structured trajectory set:

$$M_i \leftarrow f_{\mathrm{mem}}(x_i, y_i^\star, \mathcal{T}(x_i))$$

  Add generated memory to the memory bank:

$$\mathcal{M} \leftarrow \mathcal{M} \cup \{M_i\}$$

end for
return $\mathcal{M}$

with probabilities

$$p_b = \Pr[S \in \mathcal{M}_b \mid x], \qquad \sum_{b=1}^{B} p_b = 1.$$

A memory-construction method is said to *cover* mode b if at least one of the selected trajectories used for memory construction comes from $\mathcal{M}_b$.

We compare two procedures under a fixed full-trajectory expansion budget K.

Random $\mathcal{T}$. This method samples or selects K full trajectories independently from $p_{\theta_{\mathrm{trj}}}(\cdot \mid x)$.

`BranchDIME`**.** `BranchDIME` samples n prefixes, clusters them into prefix-induced branches, and expands one representative prefix from each discovered branch. In this analysis, we use the following idealized assumption.

Algorithm 2 Autoregressive Inference with Retrieved `BranchDIME` Memory

Require: Evaluation input x
Require: Memory bank $\mathcal{M}$
Require: Response-generation model $\mathrm{LLM}_{\mathrm{res}}$
Require: Number of retrieved memories r
Require: Maximum response length $T_{\max}$
Ensure: Final response $\hat{S}$

Retrieve relevant memories:

$$\mathcal{M}^{\star}(x) \leftarrow \mathrm{Retrieve}(x, \mathcal{M}, r)$$

Initialize response prefix $w_{1:0} \leftarrow \emptyset$
for $t = 1, \ldots, T_{\max}$ **do**
    Sample next token with memory-augmented context:

$$w_t \sim p_{\mathrm{res}}(w_t \mid x, \mathcal{M}^{\star}(x), w_{<t})$$

    if w_t is EOS **then**
        break
    end if
end for
Set $\hat{S} \leftarrow (w_1, \ldots, w_t)$
Extract final answer $\hat{y}$ from $\hat{S}$ if required
return $\hat{S}$ or $\hat{y}$

Assumption 1 (Mode recovery by prefix clustering)**.** *If at least one sampled prefix comes from mode $\mathcal{M}_b$, then prefix clustering discovers a corresponding branch and `BranchDIME` expands one representative prefix from that branch.*

This assumption abstracts away clustering error. In the main paper, we empirically test whether discovered branches have nontrivial mass and lead to distinct downstream outcomes.

B.2 Coverage Advantage of Prefix Branching

Proposition 2 (Coverage advantage of prefix branching)**.** *Consider a random-K trajectory memory that selects K independent full trajectories from $p_\theta(\cdot \mid x)$. The probability that it covers mode b is*

$$P_{\mathrm{rand}}(b) = 1 - (1 - p_b)^K.$$

Now consider `BranchDIME` with n sampled prefixes. Under Assumption 1, the probability that `BranchDIME` covers mode b is

$$P_{\mathrm{BB}}(b) = 1 - (1 - p_b)^n.$$

Therefore, if $n \geq K$,

$$P_{\mathrm{BB}}(b) \geq P_{\mathrm{rand}}(b)$$

for every mode b, with strict inequality whenever $n > K$ and $p_b \in (0, 1)$.

Proof. A random-K trajectory memory fails to cover mode b exactly when none of its K sampled trajectories lies in $\mathcal{M}_b$. Since each sample belongs to $\mathcal{M}_b$ with probability p_b, this failure event has probability

$$(1 - p_b)^K.$$

Therefore,

$$P_{\mathrm{rand}}(b) = 1 - (1 - p_b)^K.$$

For `BranchDIME`, mode b is discovered if at least one of the n sampled prefixes comes from $\mathcal{M}_b$. The probability that no sampled prefix comes from $\mathcal{M}_b$ is

$$(1 - p_b)^n.$$

Under Assumption 1, every discovered mode is represented by an expanded prefix, so

$$P_{\mathrm{BB}}(b) = 1 - (1 - p_b)^n.$$

Finally, for any $p_b \in [0, 1]$, the function

$$m \mapsto 1 - (1 - p_b)^m$$

is nondecreasing in m. Hence, if $n \geq K$,

$$1 - (1 - p_b)^n \geq 1 - (1 - p_b)^K.$$

The inequality is strict whenever $n > K$ and $p_b \in (0, 1)$. □

B.3 Expected Mode Coverage

Let C denote the number of covered modes. For a selected trajectory set A, define

$$C(A) = \sum_{b=1}^{B} \mathbf{1}\{b \in A\},$$

where $\mathbf{1}\{b \in A\}$ indicates whether mode b is covered.

Corollary 2 (Expected mode coverage)**.** *Under the assumptions of Proposition 2, the expected number of modes covered by* `BranchDIME` *is at least that of random-K trajectory memory:*

$$\mathbb{E}[C_{\mathrm{BB}}] = \sum_{b=1}^{B} (1 - (1 - p_b)^n) \geq \sum_{b=1}^{B} \left(1 - (1 - p_b)^K\right) = \mathbb{E}[C_{\mathrm{rand}}].$$

Proof. By linearity of expectation,

$$\mathbb{E}[C] = \sum_{b=1}^{B} \Pr[b \text{ is covered}].$$

Substituting the coverage probabilities from Proposition 2 gives

$$\mathbb{E}[C_{\mathrm{BB}}] = \sum_{b=1}^{B} (1 - (1 - p_b)^n)$$

and

$$\mathbb{E}[C_{\mathrm{rand}}] = \sum_{b=1}^{B} \left(1 - (1 - p_b)^K\right).$$

Since each term in the `BranchDIME` sum is at least the corresponding term in the random-K sum when $n \geq K$, the result follows. □

B.4 Rare-Mode Advantage

Corollary 3 (Rare-mode advantage)**.** *For a rare mode with $p_b \ll 1$,*

$$1 - (1 - p_b)^m \approx m p_b.$$

Therefore, the ratio between `BranchDIME`*'s mode-discovery probability and random-K selection's mode-discovery probability is approximately*

$$\frac{P_{\mathrm{BB}}(b)}{P_{\mathrm{rand}}(b)} \approx \frac{n}{K}.$$

Proof. For small p_b, the first-order Taylor approximation gives

$$(1 - p_b)^m = 1 - m p_b + O(p_b^2).$$

Thus,

$$1 - (1 - p_b)^m = mp_b + O(p_b^2).$$

Applying this approximation with $m = n$ and $m = K$ yields

$$P_{\mathrm{BB}}(b) \approx np_b, \qquad P_{\mathrm{rand}}(b) \approx Kp_b.$$

Taking the ratio gives

$$\frac{P_{\mathrm{BB}}(b)}{P_{\mathrm{rand}}(b)} \approx \frac{n}{K}.$$

□

In our default setting, `BranchDIME` samples $n = 20$ prefixes and expands $K = 5$ representatives. Under the idealized model above, this gives an approximately $4\times$ advantage in discovering rare reasoning modes before the full-trajectory expansion budget is spent.

B.5 Connection to Memory Utility

The preceding results show that `BranchDIME` covers more reasoning modes in expectation. We now connect this coverage advantage to memory quality under a simple additive utility model.

Assume that each reasoning mode b contains a reusable memory signal with nonnegative utility $q_b \geq 0$. This signal may correspond to a positive strategy, an anti-pattern, or a contrastive rule that can be extracted when the mode is represented in the trajectory set. For a covered mode set $A \subseteq \{1, \ldots, B\}$, define additive memory utility as

$$Q(A) = \sum_{b \in A} q_b.$$

Proposition 3 (Coverage improves additive memory utility)**.** *Under Assumption 1, if $n \geq K$, then*

$$\mathbb{E}[Q(A_{\mathrm{BB}})] \geq \mathbb{E}[Q(A_{\mathrm{rand}})].$$

Proof. By definition,

$$Q(A) = \sum_{b=1}^{B} q_b \mathbf{1}\{b \in A\}.$$

Taking expectations and using linearity of expectation,

$$\mathbb{E}[Q(A)] = \sum_{b=1}^{B} q_b \Pr[b \in A].$$

From Proposition 2,

$$\Pr[b \in A_{\mathrm{BB}}] \geq \Pr[b \in A_{\mathrm{rand}}]$$

for every mode b when $n \geq K$. Since $q_b \geq 0$ for all b, multiplying each inequality by q_b and summing over b gives

$$\mathbb{E}[Q(A_{\mathrm{BB}})] \geq \mathbb{E}[Q(A_{\mathrm{rand}})].$$

□

This proposition formalizes the intuition that, when different reasoning modes expose useful and nonnegative memory signals, improving mode coverage can improve expected memory quality. The additive utility model is intentionally simple; in practice, the memory generator may combine signals nonlinearly, discard redundant information, or extract contrastive rules from interactions between modes. Nonetheless, the result captures the key mechanism behind `BranchDIME`: prefix-level branch discovery gives minority reasoning modes more opportunities to be represented before memory is distilled.

C Baseline Details

In Table 10, we provide a clear description of the all the baselines used in this work.

Table 10: Comparison of candidate generation, selection, and expansion strategies.

Method	Candidate generation	Selection	Expansion
Single $\mathcal{T}$	1 prefix or 1 full trajectory	random/single	1 full trajectory
Upper-cost diagnostic baselines			
Consensus $\mathcal{T}$	$(n = 20)$ full trajectories	frequent-answer group	already full
Full-Traj Cluster	$(n = 20)$ full trajectories	cluster full trajectories	already full
Cost-matched baselines			
Random $\mathcal{T}$	$(n = 20)$ prefixes	random $(K = 5)$ prefixes	5 full trajectories
Farthest-First $\mathcal{T}$	$(n = 20)$ prefixes	farthest prefixes	5 full trajectories
`BranchDIME`	$(n = 20)$ prefixes	one prefix per cluster	5 full trajectories

D Experimental details

We use three distinct language-model components in our experiments. LLM_{trj} denotes the model used to generate trajectories for constructing memory, LLM_{res} denotes the model used to generate the final response at inference time, and f_{mem} denotes the memory generator used to construct `BranchDIME` memories. Unless otherwise specified, $\text{LLM}_{\text{trj}} = \text{LLM}_{\text{res}}$ in the main experiments. For cross-model transfer experiments, LLM_{trj} and LLM_{res} may differ. We use `GPT-5-mini` as f_{mem} for memory generation. Table 11 and Table 12 provides the detailed description on all the components in our pipeline.

E Qualitative Case Study: Branch Contrast Reveals Reusable Memory

We include a qualitative example from GPQA-DIAMOND to illustrate how `BranchDIME` converts trajectory diversity into reusable memory. The problem asks how surface roughness changes the apparent contact angles of water and oil. The smooth coating has contact angles of 104° for water and 76° for oil. On the rough coating, the advancing and receding water contact angles are 133° and 125°, respectively. The correct option is A: water $\approx 128^\circ$, oil $\approx 48^\circ$, and water droplets roll off at low tilt angles.

The sampled trajectories agree on some intermediate facts, such as estimating the rough water contact angle from the advancing and receding angles. However, they diverge on the key modeling decision: whether the wetting state inferred for water should also be applied to oil. This produces distinct reasoning branches, summarized in Table 13. The successful branch applies roughness in a sign-sensitive way: since water has a smooth contact angle above 90°, roughness increases its apparent angle, while oil has a smooth contact angle below 90°, so Wenzel-type roughness decreases its apparent angle. Failing branches instead overgeneralize the water Cassie-like interpretation to oil or confuse high static contact angle with high roll-off tilt despite low hysteresis.

This case study highlights why memory from a single realized trajectory can be incomplete. A single correct trajectory may provide the final answer but not expose the plausible failure modes, while a single incorrect trajectory may encode a misleading rule such as applying the water Cassie interpretation to oil. By comparing branches, `BranchDIME` extracts contrastive memories about dynamic-angle averaging, Wenzel/Cassie model selection, sign-sensitive roughness transfer, and hysteresis-based roll-off reasoning. Thus, the reusable memory emerges most clearly from the structure of the trajectory distribution rather than from any individual trace alone.

F Tables with Standard Deviation

This appendix provides the standard deviations for the main benchmark results. Each experiment is repeated over three runs with different random seeds affecting trajectory sampling order and decoding stochasticity. The main text reports mean performance for readability, while Table 14 reports mean $\pm$ standard deviation. These values quantify run-to-run variability and complement the aggregate trends discussed in Section 4.

Table 11: Environment, memory, and language-model parameters. $\mathrm{LLM}_{\mathrm{trj}}$ denotes the trajectory-generation model, $\mathrm{LLM}_{\mathrm{res}}$ denotes the response-generation model, and f_{mem} denotes the memory generator.

Language Model Components	
Trajectory-generation model $\mathrm{LLM}_{\mathrm{trj}}$	Task-specific base LM used to sample prefixes and expand trajectories
Response-generation model $\mathrm{LLM}_{\mathrm{res}}$	Task-specific base LM used to generate final evaluation responses
Memory generator f_{mem}	`GPT-5-mini` by default; open-weight variants `Q3-30B`, `L3-70B` evaluated in ablations
Main-experiment setting	$\mathrm{LLM}_{\mathrm{trj}} = \mathrm{LLM}_{\mathrm{res}}$
Cross-model transfer setting	$\mathrm{LLM}_{\mathrm{trj}}$ and $\mathrm{LLM}_{\mathrm{res}}$ may differ
Decoding Settings	
Prefix-sampling temperature	*0.6*
Trajectory-expansion temperature	*0.6*
Final-response temperature	*0.6*
Memory-generation temperature	0
Top-p	*0.95*
Decoding strategy for f_{mem}	Greedy decoding
Maximum prefix length	$k = 256$ tokens
Maximum full trajectory length	*8192*
Maximum final response length	*32K*
Maximum memory-generation length	*32K*
Stop condition	EOS token or maximum generation length
Self-consistency samples	5 for Self-Consistency@5 baselines
Random seeds	Three independent runs with different sampling seeds
Prefix Sampling and Expansion	
Number of sampled prefixes	$n = 20$ per memory-construction instance
Prefix length	$k = 256$ tokens
Number of selected representatives	$K = 5$
Number of fixed branches	$B = 5$
Continuations per selected prefix	$m_b = 1$ for each branch b
Fixed-branch setting	$K = B = 5$; one representative prefix is expanded per branch
Prefix sampling	Prefixes are sampled independently from $\mathrm{LLM}_{\mathrm{trj}}$
Prefix expansion	Selected prefixes are continued autoregressively using $\mathrm{LLM}_{\mathrm{trj}}$
Clustering and Branch Discovery	
Prefix representation	TF–IDF features over prefix tokens
Tokenization	*whitespace*
Stop-word removal	*yes*
Vector normalization	L2 normalization to unit norm
Similarity metric	Cosine similarity
Clustering algorithm	Spherical k-means
Number of clusters	$B = 5$
Number of k-means initializations	*10*
Maximum k-means iterations	*25*
Representative selection	Select the prefix closest to the cluster centroid
Adaptive clustering linkage	Agglomerative clustering in adaptive ablations
Adaptive clustering threshold	$\delta \in \{0.20, 0.30\}$ in adaptive ablations

G Additional Experiments

G.1 Rank-Based Retrieval Utility

Since the benchmarks do not provide gold query–memory relevance labels, we evaluate retrieval quality through downstream utility. For each query, we retrieve the top-6 memories by cosine similarity and compare conditioning on ranks 1–3 versus ranks 4–6.

As shown in Table 15, using ranks 1–3 improves accuracy by 5.9, 1.9, and 3.2 percentage points, respectively, over ranks 4–6. This indicates that higher-ranked memories provide greater downstream

Table 12: Environment, memory, and language-model parameters. $\mathrm{LLM}_{\mathrm{trj}}$ denotes the trajectory-generation model, $\mathrm{LLM}_{\mathrm{res}}$ denotes the response-generation model, and f_{mem} denotes the memory generator.

Memory Generation	
Memory generator	$f_{\mathrm{mem}} =$ `GPT-5-mini` by default
Memory-generation prompt	Fixed prompt shown in Appendix I.1
Prompt includes question	Yes
Prompt includes reference answer	Yes; memory construction is supervised on memory-construction instances
Prompt includes trajectories	Yes; selected full trajectories are provided to f_{mem}
Number of trajectories given to f_{mem}	1 for Single $\mathcal{T}$; $K = 5$ for multi-trajectory memory methods
Maximum input context length	*Prompt + Max trajectory length*
Maximum memory output length	*32K*
Memory output format	JSON with memory type, statement, and evidence trajectory identifiers
Memory freezing	Memories are generated once from the memory-construction set and frozen before evaluation
Memory Bank and Retrieval	
Memory-construction datasets	GPQA-$\mathcal{M}$ for GPQA-DIAMOND; MATH-$\mathcal{M}$ for MATH500, AIME25, and OLYMMATH
Stored memory fields	Memory text, source question id, selected trajectory ids, and selection metadata
Whether source questions are stored	*yes*
Whether source answers are stored	*no*
Retrieval embedding model	`BAAI/bge-m3` Chen et al. [2024]
Text embedded for retrieval	*Memory only*
Retrieval query	Evaluation question text
Similarity function	Cosine similarity
Number of retrieved memories	$r = 3$ unless otherwise specified
Evaluation	
Evaluation datasets	GPQA-DIAMOND, MATH500, AIME25, and OLYMMATH
Train/evaluation separation	Memory-construction examples are disjoint from evaluation examples
Number of runs	3 independent runs
Reported statistic	Mean across runs; standard deviations reported separately
Evaluation metric	Accuracy for GPQA-DIAMOND; Pass@1 / exact match for math benchmarks
Answer extraction	Used LIGHTEVAL presets for individual task Habib et al. [2023]
Correctness evaluator	Used LIGHTEVAL presets for individual task
Self-consistency aggregation	Majority vote over final extracted answers
Standard deviations	Computed across runs that vary trajectory sampling and decoding stochasticity

utility. On DSQ–GPQA, ranks 4–6 achieve 0.522 accuracy, close to the 0.525 no-memory baseline, showing that lower-quality retrieval can largely eliminate the memory gain.

All memory-construction variants use the same retriever, retrieval depth, and inference prompt. Thus, improvements from `BranchDIME` over Single, Random, and Consensus reflect differences in stored memory quality rather than the retrieval procedure.

H Additional Related Work

The ability of LLMs to reason through intermediate chains of thought has motivated a broad set of methods that improve inference by sampling, decomposing, or searching over reasoning trajectories Wei et al. [2022], Zhou et al. [2023], Zelikman et al. [2022], Wang et al. [2023], Jindal et al. [2025]. Self-consistency shows that multiple sampled chains can improve final-answer accuracy by aggregating over diverse reasoning paths Wang et al. [2023], while Tree-of-Thoughts and Graph-of-Thoughts generalize chain-based reasoning into explicit search structures over intermediate thoughts

Table 13: Qualitative example of branch-structured memory construction on GPQA-DIAMOND. For the same input, sampled trajectories disagree on how roughness should transfer from water to oil. `BranchDIME` exposes these distinct reasoning modes and distills reusable memories from their contrasts.

question text: A coating is applied to a substrate resulting in a perfectly smooth surface. The measured static contact angles of this smooth coating are 104° and 76° for water and oil respectively. The coating formulation is then modified and when now applied to the same type of substrate, a rough surface is produced. For the rough surface, the advancing and receding water contact angles are measured to be 133° and 125° respectively. What would be the best estimate of the static water and static oil contact angles of the rough surface, and would water droplets roll off of the surface at low or high tilt angles."

choices: ["Water = 128°, Oil = 48°, low tilt angle", "Water = 129°, Oil = 98°, low tilt angle", "Water = 128°, Oil = 48°, high tilt angle", "Water = 129°, Oil = 98°, high tilt angle"]

Branch	Reasoning mode	Predicted answer	Memory signal
Water-static-angle branch	Uses the advancing and receding water angles to estimate the rough-surface static water angle as approximately $(133^\circ + 125^\circ)/2 \approx 129^\circ$. This branch identifies the right water-angle estimate, but may still diverge later on the oil estimate.	Partially correct	If advancing and receding contact angles are given, estimate the static angle as lying between them rather than treating either dynamic extreme as the equilibrium value.
Correct Wenzel-transfer branch	Uses the smooth water angle above 90° and smooth oil angle below 90° to apply roughness with opposite effects: roughness increases the apparent water angle but decreases the oil angle under Wenzel-type penetration. This yields water $\approx$ 128°–129°, oil $\approx 48^\circ$, and low water roll-off tilt.	A	If roughness is transferred across liquids, check the sign of $\cos\theta$: roughness increases apparent angle for smooth angles above 90° but decreases it for smooth angles below 90° under Wenzel wetting.
Cassie-overgeneralization branch	Infers from the high water angle and low hysteresis that the rough surface is Cassie-like for water, then incorrectly transfers this same Cassie interpretation to oil. This raises the oil angle from 76° to 98° and leads to answer B.	B	If different liquids have smooth contact angles on opposite sides of 90°, do not transfer the wetting state inferred for one liquid directly to the other.
Tilt-confusion branch	Recognizes that the water contact angle is high, but confuses droplet mobility by selecting a high tilt angle despite low hysteresis, $(133^\circ - 125^\circ = 8^\circ)$. This leads to answer D in one trajectory.	D	If a problem asks about roll-off, use contact-angle hysteresis and wetting state, not the static angle alone, to infer low or high tilt behavior.

Yao et al. [2023], Besta et al. [2024]. More recent inference-time scaling approaches further suggest that reasoning trajectories are not arbitrary samples: early prefixes can induce coherent families of downstream reasoning behavior, revealing a branching structure in the model's local reasoning space Jindal et al. [2026], Ji et al. [2025]. These methods establish that diversity and structure are useful for inference. `BranchDIME` builds on this observation, but changes its role: rather than using multiple trajectories only to select or aggregate an answer for the current problem, we use their structure to construct reusable memory.

Memory-augmented agents provide a complementary mechanism for improving behavior over time. Reflexion stores verbal feedback from past trials and reuses it to guide later attempts Shinn et al.

Table 14: Performance comparison of BranchDIME against different baselines with standard deviation. Here $\text{LLM}_{\text{trj}} = \text{LLM}_{\text{res}}$.

	GPQA-$\mathcal{M}$ on GPQA-DIAMOND			MATH-$\mathcal{M}$ on MATH500		
Memory	DSQ-7B	Q2.5-7B	DSL-8B	DSQ-7B	Q2.5-7B	DSL-8B
Single Response (CoT)						
No Memory	0.491 ± 0.036	0.333 ± 0.027	0.406 ± 0.018	0.808 ± 0.024	0.728 ± 0.012	0.831 ± 0.022
Self-Consistency@5 (CoT)						
No Memory	0.525 ± 0.010	0.343 ± 0.020	0.470 ± 0.009	0.865 ± 0.019	0.756 ± 0.001	0.866 ± 0.001
Single $\mathcal{T}$	0.533 ± 0.000	0.349 ± 0.011	0.505 ± 0.005	0.882 ± 0.001	0.764 ± 0.003	0.890 ± 0.003
Random $\mathcal{T}$	0.535 ± 0.000	0.342 ± 0.012	0.512 ± 0.006	0.890 ± 0.002	0.767 ± 0.003	0.886 ± 0.007
Consensus $\mathcal{T}$	0.530 ± 0.031	0.347 ± 0.006	0.485 ± 0.002	0.891 ± 0.002	0.754 ± 0.011	0.889 ± 0.009
BranchDIME	$\mathbf{0.581 \pm 0.000}$	$\mathbf{0.392 \pm 0.021}$	$\mathbf{0.534 \pm 0.012}$	$\mathbf{0.896 \pm 0.005}$	$\mathbf{0.795 \pm 0.003}$	$\mathbf{0.902 \pm 0.000}$

Table 15: Downstream accuracy using higher- and lower-ranked memories.

Memories Passed	**DSQ–GPQA**	**DSQ–MATH500**	**DSL–GPQA**
Ranks 4–6	0.522	0.877	0.502
Ranks 1–3	**0.581**	**0.896**	**0.534**

[2023]; Generative Agents retrieve and synthesize natural-language memories for planning and interaction Park et al. [2023]; MemoryBank studies long-term memory for sustained interaction and personalization Zhong et al. [2024]; and Voyager accumulates executable skills that enable open-ended lifelong learning Wang et al. [2024]. These systems demonstrate that memory can convert past experience into future improvement. However, their memories typically correspond to episodes, reflections, interactions, or skills. BranchDIME focuses on a more specific problem: how to construct *reasoning memory* from the stochastic trajectories produced by an LLM-based agent.

The closest line of work uses reasoning traces themselves as reusable experience. ReasoningBank distills strategies from successful and failed agent trajectories and retrieves them at test time Ouyang et al. [2026]. FLEX builds experience libraries for gradient-free agent evolution, showing that accumulated experience can improve future behavior and transfer across agents Cai et al. [2025]. Metacognitive Reuse compresses recurring reasoning fragments into reusable behaviors that can be stored in a handbook or incorporated through fine-tuning Didolkar et al. [2025]. These approaches show that reasoning traces contain reusable knowledge, but they largely preserve a trajectory-centric abstraction: memory is extracted from realized traces, and each trace is treated as an independent unit of evidence. Under stochastic decoding, this is a limited view. A single trace is only one sample from a distribution over possible reasoning paths, and abstractions built from it can inherit sample-specific artifacts such as phrasing, exploration order, or local mistakes.

BranchDIME differs by changing the unit of experience. Instead of constructing memory from isolated trajectories, it constructs memory from a structured set of trajectories that approximates the local reasoning distribution for a task. This distinction is central: prior memory systems primarily accumulate experience across tasks or time, whereas BranchDIME models *within-task trajectory diversity*. By clustering early prefixes, selecting representative reasoning modes, and expanding them into full trajectories, BranchDIME exposes the branch-level organization of reasoning. Memory is then distilled from comparisons across these trajectories, producing positive strategies, anti-patterns, and contrastive strategies. This makes the memory less a record of what happened in one trace and more a compressed representation of how reasoning can succeed or fail across plausible alternatives.

This perspective also clarifies how BranchDIME relates to diversity-based inference. Self-consistency, tree search, graph search, and prefix-consensus methods can all be viewed as ways to obtain or organize multiple reasoning paths Wang et al. [2023], Yao et al. [2023], Besta et al. [2024], Jindal et al. [2026]. BranchDIME is orthogonal to these mechanisms: it can use them as trajectory generators, but its objective is memory construction rather than answer aggregation. In this sense, BranchDIME converts inference-time diversity from a heuristic for solving the current instance into a learning signal for future instances.

Finally, our work suggests a shift in perspective: from treating reasoning diversity as noise to be aggregated, to treating it as structure to be learned.

I Prompts

I.1 Prompt to construct Memories

Memory Extraction Prompt

```
You are a reasoning expert whose task is to extract reusable reasoning
    constraints ("memories") from multiple solution attempts to the same
     problem.
---

## Input

You will be given:
- a question
- a reference (gold) answer
- multiple reasoning trajectories of varying quality
  (correct, incorrect, or partially correct)

---

## Objective

Do NOT solve the problem.
Do NOT fix or rewrite any trajectory.
Do NOT produce a correct solution.

Your task is to analyze differences in reasoning across trajectories
and distill them into generalizable reasoning memories that can
improve future problem solving.

---

## What to Extract

Each memory should capture a reusable reasoning principle that reflects:
- a situation where reasoning succeeds or fails
- a required property that strong reasoning satisfies
- a pattern of reasoning that leads to failure (if observable)

Memories must reflect reasoning behavior, not final correctness.

---

## Memory Forms

Each memory must take one of the following forms:

1. Contrastive (preferred when clear differences exist)
   If [condition], ensure [required reasoning property],
   and do not [failure pattern].

2. Positive-only
   If [condition], ensure [required reasoning property].

3. Negative-only
   If [condition], do not [failure pattern].

Use the contrastive form only when both components are clearly supported
    .

---

## Analysis Focus
```

```
Examine reasoning at a structural level, including:
- how assumptions, constraints, and objectives are represented
- whether intermediate representations (variables, abstractions,
    decompositions) are coherent
- whether the chosen approach matches the structure of the problem
- whether reasoning proceeds in a logically consistent sequence
- whether intermediate conclusions are justified before being used
- whether conclusions are checked against initial conditions
- whether special cases, boundary conditions, or hidden constraints are
    handled
- whether consistency is maintained across transformations or steps

Use abstract reasoning concepts such as:
- constraint satisfaction
- decomposition and recomposition
- invariants and conservation
- conditional reasoning and case analysis
- consistency checking
- representation alignment
- verification and validation

---

## What to Ignore

Do not focus on:
- arithmetic or calculation mistakes without structural impact
- formatting, writing style, or verbosity
- final answers in isolation
- rewriting or correcting solutions
- problem-specific constants, values, or domain-specific details

---

## Extraction Rules

1. Only extract a memory if supported by clear patterns across
    trajectories.
2. Each memory must be abstract and transferable across domains.
3. Do not reference specific numbers, entities, or problem details.
4. Each memory must be written as exactly one sentence.
5. Do not include explanations or commentary.
6. Do not invent unsupported patterns.
7. Prefer precise partial memories over unsupported complete ones.

---

## Symbolic / Mathematical Forms Guidance

Memories may include symbolic or mathematical forms if they represent
general reasoning structure.

These must be:
- abstract (e.g., A x = b,  f (x) = 0, constraint systems)
- reusable across different problems

Do not include:
- specific numeric values
- problem-specific equations
- domain facts that do not generalize

---

## Quality Criteria

Prefer memories that:
```

````
- capture systematic reasoning failures or invariants
- apply across domains (science, logic, quantitative reasoning)
- describe how reasoning should be structured or validated

Avoid:
- vague statements (e.g., "be careful", "think step by step")
- trivial correctness statements
- narrow heuristics tied to a specific task

---

## Problem Instance

Question:
{question_text}

Reference (gold) answer:
{gold_letter}) {gold_answer_text}

Reasoning trajectories:
{trajectories}

---

## Output Format (strict)

Return only JSON:

```json
{
  "memories": [
    {
      "type": "contrastive | enforce_only | avoid_only",
      "statement": "If ..., ensure ..., and do not ...",
      "evidence": ["trajectory_id"]
    }
  ]
}
```
````

I.2 Prompt to generate response

GPQA Response Generation Prompt

```
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.

{question}

{choices}

Relevant memory guidance from prior GPQA questions:
Use the guidance only if it is relevant to this question. Do not let the
     guidance override the question details or answer choices,

{memories}
```

J Compute Resources

All experiments were run on GPU compute nodes. For trajectory-prefix generation, we used nodes equipped with 4 NVIDIA A100 GPUs with 80GB memory per GPU. For trajectory expansion, we used a combination of nodes equipped with 4 NVIDIA A30 GPUs with 24GB memory per GPU and nodes equipped with 4 NVIDIA H100 GPUs with 96GB memory per GPU. The latency numbers reported in the main experiments were measured on nodes equipped with 4 NVIDIA A30 GPUs with 24GB memory per GPU. For open-weight memory-generation experiments, we used nodes equipped with 4 NVIDIA H100 GPUs with 96GB memory per GPU.

Experiments using API-accessed models were executed through the corresponding provider APIs. For these runs, local compute was used only for orchestration, dataset preprocessing, memory retrieval, evaluation, and result aggregation.

The main evaluation experiments require generating stochastic reasoning prefixes, constructing memories, retrieving relevant memories, and running response generation on the evaluation benchmarks. For the default `BranchDIME` configuration, each memory-construction instance samples $n = 20$ prefixes, clusters them into $B = 5$ branches, and expands $K = 5$ representative trajectories. We report response-generation latency in the main results and separately analyze offline memory-construction cost in terms of generated tokens. We did not separately log total GPU-hours for each experiment. The released code includes scripts and configuration files specifying model names, decoding parameters, dataset preparation, memory construction, retrieval, and evaluation commands.

K Limitations

`BranchDIME` demonstrates that branch-structured distributional memory can improve reasoning agents by constructing memory from structured sets of stochastic reasoning trajectories. Our study focuses on a controlled setting designed to isolate the effect of trajectory-set construction. Several natural extensions remain open.

First, `BranchDIME` requires an offline memory-construction stage in which short prefixes are sampled, clustered, and selected representatives are expanded into full trajectories. This cost is incurred once during memory construction and is amortized across future inference, so it does not substantially affect response-time latency. Nevertheless, reducing the offline construction cost further, for example through adaptive prefix budgets, early stopping criteria, or dynamic branch allocation, is an important direction for scaling `BranchDIME` to very large memory banks.

Second, our implementation uses prefix-based clustering as a lightweight approximation to trajectory-distribution structure. This design is intentionally simple and efficient: early prefixes provide a low-cost signal for discovering diverse reasoning modes before full trajectory expansion. Richer representations, learned similarity functions, or task-adaptive clustering methods may further improve branch discovery, especially in domains where meaningful reasoning differences are not fully captured by surface-level prefix variation.

Third, `BranchDIME` introduces construction hyperparameters, including the prefix length, prefix pool size, number of branches, and expansion budget. Our ablations show that a moderate prefix pool captures most of the benefit, suggesting that the method is effective within a practical compute regime. Still, automatically selecting these parameters based on the observed prefix distribution could make the method more adaptive across tasks, models, and memory-construction budgets.

Fourth, the quality of the final memory depends on the memory generator used to distill reusable guidance from the selected trajectories. Our experiments show that `BranchDIME` remains beneficial across different memory generators, but the magnitude of improvement varies. This suggests that branch-structured trajectory selection and memory distillation are complementary components. Future work could improve the distillation step with more specialized memory-generation objectives, uncertainty-aware memory filtering, or verification procedures.

Finally, our evaluation focuses on mathematical and STEM reasoning benchmarks, where stochastic reasoning trajectories naturally exhibit branching behavior. These benchmarks provide a clean testbed for studying distributional memory, but they do not cover all forms of agentic behavior. Extending `BranchDIME` to long-horizon tool use, interactive environments, multilingual tasks, and settings with

delayed or partial feedback would further clarify the generality of branch-structured distributional memory.

Overall, these limitations concern the current instantiation and evaluation scope of BranchDIME rather than the central premise of the work: under stochastic reasoning, a single realized trajectory is an incomplete unit of experience, and memory can be improved by capturing structure in the local trajectory distribution.

L Societal Impact

BranchDIME studies how memory-augmented reasoning agents can construct reusable guidance from a structured distribution of stochastic reasoning trajectories rather than from a single sampled trajectory. The potential positive impact of this work is to improve the reliability and transferability of reasoning agents. By exposing multiple reasoning branches, including successful strategies, failure modes, and contrastive decision points, BranchDIME can help future systems make better use of past reasoning experience. This may be useful in domains such as education, scientific question answering, mathematical reasoning, and decision-support systems where robust intermediate reasoning and awareness of common failure patterns are valuable.

At the same time, more effective reasoning memory can also introduce risks if deployed without appropriate safeguards. Memories distilled from model-generated trajectories may preserve or amplify incorrect assumptions, spurious heuristics, or biases present in the underlying model or data. If such memories are retrieved in inappropriate contexts, they could lead to overconfident or misleading guidance. These risks are especially important in high-stakes domains such as medicine, law, finance, or safety-critical engineering, where incorrect reasoning can have significant consequences. In addition, while our experiments use public reasoning benchmarks, applying similar memory-construction methods to private user interactions or organizational data would require careful attention to privacy, consent, access control, and data retention.

Our work is intended as a research contribution toward understanding how stochastic reasoning structure can be used for memory construction. We do not advocate unsupervised deployment in high-stakes settings. Future deployments should include evaluation of memory faithfulness, mechanisms for filtering or revising unreliable memories, safeguards against inappropriate retrieval, and human oversight when the system is used in consequential decision-making contexts.

M LLM Usage

We used ChatGPT and Claude for grammar reviews and language style polishing. In certain cases we used these models in analyzing and summarizing tables. These summaries are then verified and updated manually for correctness.

N Technical appendices and supplementary material

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: Main claims in abstract and introductions are well grounded in the methodology Section 2 and experimentation section 4.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We discussed the limitations of this work in Appendix K.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: We have specified all the assumptions and clear description of the theorems and propositions in Section 2.1.1 and in Appendix B.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: We provide experiments details in Section 3. We further provided the fine-grained details about each design choice in Appendix D Table 11 and Table 12.

Guidelines:

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

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

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

   Answer: [Yes]

   Justification: We provide the minimal code necessary to reproduce the evaluation results reported in this work. All evaluations are conducted on publicly available datasets; therefore, rather than redistributing the datasets, we provide scripts to download and preprocess them from their original public sources.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We provide experiments details in Section 3. We further provided the fine-grained details about each design choice in Appendix D Table 11 and Table 12.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: We report error bars for the main experimental results. Each setting is repeated three times with different random seeds, and we report the mean and standard deviation in Appendix F.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We provide compute-resource details in Appendix J, including the GPU node types used for prefix generation, trajectory expansion, latency measurement, and open-weight memory generation. We also report GPU memory per device and describe which parts of the pipeline use local GPU execution versus API-accessed models. Response-generation latency is reported in the main results in Table 1, and offline memory-construction cost is analyzed separately in terms of generated tokens in Section 3.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: Yes. The research conforms to the NeurIPS Code of Ethics. The work uses publicly available reasoning benchmarks, does not involve human subjects or crowdsourcing, does not release new model weights or high-risk datasets, and includes discussion of societal impacts, asset usage, reproducibility, and limitations. We have also considered potential risks associated with memory-augmented reasoning systems and describe appropriate safeguards and deployment considerations in Appendix L.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: We discuss both potential positive and negative societal impacts in Appendix L. The appendix describes possible benefits of `BranchDIME` for improving reliable reasoning and reusable failure-mode awareness, as well as risks related to incorrect or biased memories, inappropriate retrieval, privacy concerns, and use in high-stakes domains.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [N/A]

    Justification: This work does not release new model weights, scraped datasets, or high-risk data artifacts. Our experiments use publicly available reasoning benchmarks, and we provide only minimal evaluation code together with scripts to download and preprocess the public datasets from their original sources. Therefore, responsible-release safeguards for high-risk data or model artifacts are not applicable.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: The paper cites the original datasets and models, but it does not yet explicitly summarize the corresponding licenses and terms of use.

    Guidelines:

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

13. **New assets**

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

    Answer: [Yes]

    Justification: The new asset introduced with this work is the minimal codebase for reproducing the evaluation pipeline. We provide documentation alongside the code, including instructions for environment setup, dataset downloading and preprocessing, running the evaluation scripts, and reproducing the reported results. Since all datasets used in the paper are public, we do not introduce or redistribute new datasets; instead, we provide scripts that download the required datasets from their original public sources.

    Guidelines:

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

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

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

    Answer: [N/A]

Justification: This work does not involve crowdsourcing experiments or research with human subjects. All experiments are conducted using publicly available reasoning benchmarks and automated model evaluations. Therefore, participant instructions, screenshots, and compensation details are not applicable.

Guidelines:

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

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

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

Answer: [N/A]

Justification: This work does not involve research with human subjects, user studies, participant data collection, crowdsourcing experiments, or interaction with study participants. All experiments are conducted using publicly available reasoning benchmarks and automated model evaluations. Therefore, discussion of participant risks, risk disclosure, and IRB or equivalent approval is not applicable.

Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [N/A]

Justification: As described in Appendix M, We used ChatGPT and Claude for grammar reviews and language style polishing. In certain cases we used these models in analyzing and summarizing tables. These summaries are then verified and updated manually for correctness.

Guidelines:

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