KV-PRM: Efficient Process Reward Modeling via KV-Cache Transfer for Multi-Agent Test-Time Scaling

Peng Kuang[1], **Haibo Jin**[1], **Xiaoyu Han**[1], **Yanli Wang**[2], **Xiaopeng Yuan**[1], **Ye Yu**[1], **Kaidi Xu**[3], **Haohan Wang**[1]
[1]University of Illinois Urbana-Champaign, [2]Imperial College London,
[3]City University of Hong Kong
pengk3@illinois.edu, kaidixu@cityu.edu.hk, haohanw@illinois.edu

Abstract

Process Reward Models (PRMs) have been proven to be highly effective in guiding test-time scaling (TTS) methods, which significantly boost the capabilities of LLM-based multi-agent systems. However, existing PRMs are *text-based*: they re-encode the entire trajectory text from scratch. In long multi-agent rollouts, the scoring cost, growing quadratically with respect to sequence length L, creates a severe computational bottleneck, severely limiting PRMs' application in long-context scenarios. To resolve this, we introduce KV-PRM, a highly efficient process reward model that eliminates the heavy text re-encoding by directly reading the KV cache produced naturally during the LLM's generation phase. By processing a single "verify token" against the pre-existing KV cache, KV-PRM reduces the scoring cost from $O(L^2)$ to $O(L)$. We formally prove that the KV cache contains greater information capacity than text, and is more efficient for downstream reward modeling. Empirically, across the MATH, GSM8K, and AIME benchmarks and three model families, KV-PRM matches text-PRMs under various TTS methods such as Beam Search, MCTS, and Weighted Voting, with up to a 5,000$\times$ reduction in scoring FLOPs, a 37$\times$ reduction in latency, and a 34$\times$ reduction in per-sequence memory footprint compared to text-based PRMs. Our code is available at `https://github.com/KrumperIsOK/KV-PRM`.

1 Introduction

Scaling test-time compute has recently emerged as a highly effective direction for improving the capabilities of Large Language Models (LLMs) on complex reasoning tasks [1, 2, 3]. Instead of relying on a single greedy decoding pass, advanced reasoning pipelines leverage multi-agent systems (MAS) [4] and test-time search (TTS) algorithms [5], such as beam search and Monte Carlo Tree Search (MCTS) [6], to explore diverse solution trajectories. A critical component guiding this search is the Process Reward Model (PRM) [7, 8, 9]. By evaluating the correctness of intermediate reasoning steps, PRMs guide the search algorithm to allocate compute toward more promising partial solutions.

However, scaling test-time search exposes a critical computational bottleneck: the cost of the PRM itself. Existing PRMs are fundamentally *text-based*. For every candidate trajectory proposed by the generator, a text-PRM must encode the entire sequence of text tokens from scratch to output a reward. As self-attention scales quadratically with sequence length, this introduces an $O(L^2)$ computational cost per scoring call, where L is the sequence length. In modern MAS workflows, reasoning trajectories routinely span thousands or tens of thousands of tokens. When searching over dozens of candidates at multiple agent handoffs, the redundant text re-encoding by the PRM becomes an even heavier burden, often matching the FLOPs required for the generation itself. Such an architectural bottleneck limits the application of PRMs in long-context reasoning scenarios.

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

In this work, we identify a missed opportunity in current test-time scaling systems: the generation process of agents inherently computes a rich, high-dimensional representation of the trajectory, the Key-Value (KV) cache. As autoregressive generation requires caching historical states to predict the next token, a complete record of the model's intermediate representations across all layers and positions is produced naturally during the LLM's generation phase. In Section 3, we further theoretically prove the representational advantage of KV cache and its efficient exploitation compared to text. Specifically, we establish a theoretical framework demonstrating that the KV cache contains greater information capacity than text, and requires much less compute for downstream reward modeling. We formally show that the KV cache provides an $\Omega(d/\log|\mathcal{V}|)$ capacity advantage per position over text tokens, and prove that the approximation gap decays exponentially in readout depth k, with efficient $k=1$ capturing most of extractable reward information for practical architectures (Theorem 2). Re-encoding the decoded text from scratch discards this high-fidelity, continuous representation, forcing the verification model to rely solely on discrete text tokens that collapse the rich internal state.

Building upon this insight, we propose KV-PRM, a highly efficient architecture for process reward modeling. Instead of processing text, KV-PRM scores trajectories via *KV-cache transfer*. By appending a single "verify token" to the end of the sequence and processing it through a lightweight LoRA adapter [10] conditioned on the base model's pre-existing KV cache, KV-PRM achieves orders-of-magnitude complexity reduction. The per-call scoring cost drops from $O(L^2)$ to $O(L)$, resulting in up to a $5{,}000\times$ reduction in scoring FLOPs for typical MAS trajectories.

Empirically, we evaluate KV-PRM across three model scales (Qwen3-0.6B, 4B, 8B) and two additional model families (Llama-3.2-3B and Ministral-3-3B) on MATH, GSM8K, and AIME benchmarks in Section 5. We find that KV-PRM not only recovers the performance of text-based PRMs but frequently outperforms them, closing the theoretical approximation gap while accelerating wall-clock scoring latency by 15–37$\times$, and reducing the per-sequence memory footprint by 34.2$\times$. Finally, as KV-PRM evaluates the continuous KV cache rather than discrete text, its reward signal is fully differentiable. We explore this property through a proof-of-concept technique called KV Steering (Section 6), demonstrating that gradient-based optimization of the latent messages between agents is structurally possible and presents a promising direction for test-time scaling in the latent space. In summary, our contributions are:

- **Theoretical framework:** We formalize the representational advantage of the KV cache over text tokens for verification, and prove that the approximation gap decays exponentially in readout depth, formally justifying the efficiency of KV cache-based reward modeling.
- **Algorithm (KV-PRM):** We introduce a novel PRM architecture that reuses the generation KV cache, reducing scoring complexity from $O(L^2)$ to $O(L)$ by addressing the architectural bottleneck of text-based PRMs.
- **Extensive empirical validation:** We demonstrate that KV-PRM matches or exceeds the accuracy of text-based PRMs across multiple model scales, model families, and search algorithms, transfers to code generation, and extends to generators of different architectures, while delivering 37$\times$ wall-clock speedups and 34.2$\times$ reduction in per-sequence memory footprint.

2 Preliminaries and Notations

Multi-agent reasoning systems. A multi-agent system (MAS) decomposes a problem across D specialized LLM agents $a_1, \ldots, a_D$ [4, 11]. Each agent a_j generates output $\mathbf{o}_j$ conditioned on preceding agents' outputs and the original question, using a decoder-only transformer f_θ with N_L layers, n_h attention heads per layer, and hidden dimension d. Let L_j denote the total sequence length after agent a_j completes, and $L = L_D$ the final trajectory length.

Process reward models and test-time search. A process reward model (PRM) predicts a score $s_j \in [0, 1]$ for the trajectory up to agent step j, indicating the likelihood of reaching a correct answer [7, 8, 12]. PRM scores guide test-time search (TTS), such as beam search, MCTS [13], or weighted majority voting [14], to select high-quality trajectories [2]. With beam width W and D agent steps, the PRM is invoked at least $W \cdot D$ times. Existing PRMs are *text-based*: they perform a full forward pass over the trajectory text $\mathbf{x}_j$ of length L_j, incurring cost $F_{\text{forward}}(L_j)$ per call.

KV cache in autoregressive generation. During autoregressive generation, the transformer avoids redundant recomputation by caching the key and value projections from all previous tokens at every

layer. After generating L tokens, this *KV cache* $(\mathbf{K}, \mathbf{V})$ with $\mathbf{K}, \mathbf{V} \in \mathbb{R}^{N_L \times L \times d}$ constitutes a complete record of the model's intermediate representations across all layers and positions. Each new token is generated by attending to this cache at cost $O(d \cdot L)$ per token. Crucially, the KV cache is a *byproduct* of generation that exists whether or not it is used for any downstream purpose.

The computational bottleneck. The dominant cost in a forward pass over a sequence of length L is:

$$F_{\text{forward}}(L) = N_L \cdot \big(\underbrace{c_{\text{attn}} \cdot d \cdot L^2}_{\text{self-attention}} + \underbrace{c_{\text{ffn}} \cdot d^2 \cdot L}_{\text{feed-forward}} \big) \tag{1}$$

For the long trajectories in MAS ($L \gg d$), self-attention dominates, and each text-PRM scoring call costs $O(d \cdot L^2)$. When the PRM and the LLM are at the same scale, PRM scoring roughly *doubles* the total system compute. This motivates our central question: *can we reduce the per-call scoring cost from $O(d \cdot L^2)$ to $O(d \cdot L)$, a reduction of $L\times$, without a loss in scoring quality?* We begin by analyzing the information in the KV cache to establish that this reduction is not only computationally feasible but theoretically well-grounded in Section 3, before presenting our method in Section 4.

3 Theoretical Understanding

We begin by reviewing the representational advantage of the KV cache over text tokens in Section 3.1, which serves as the foundation for our main theoretical contributions: a verification error decomposition into a representation-dependent error floor and a readout approximation gap (Theorem 1), and a diminishing-returns analysis proving that a single-token readout readily captures most of the extractable reward information (Theorem 2). Full proof in Appendix A.3.

3.1 Representational Advantage of KV Cache

As a foundation for the results that follow, we first establish that the KV cache produced during generation carries richer information than decoded text.

Assumption 1 (Linear Representation Hypothesis [15, 16])**.** *The hidden embeddings $h \in \mathbb{R}^d$ produced by the transformer are linear combinations $h = \sum_{i=1}^{d} c_i s_i$ of a linearly independent semantic basis $\{s_1, \ldots, s_d\} \subset \mathbb{R}^d$ with ternary coefficients $c_i \in \{0, \pm 1\}$, where $c_i = 0$ indicates absence of semantic i, and $c_i = \pm 1$ its positive or negative presence.*

Proposition 1 (Representational Advantage of KV Cache)**.** *Let f_θ be an autoregressive transformer that, during generation, produces KV cache $\mathbf{H} = (\mathbf{K}, \mathbf{V})$ and text tokens $\mathbf{x} = (x_1, \ldots, x_L)$ via $\mathbf{x} = \text{decode}(\mathbf{H})$. Let Y denote any target variable of interest (e.g., trajectory correctness). Then, under Assumption 1, if the information in the KV cache of a length-L trajectory is to be expressed losslessly through text, the required number of tokens is at least*

$$m' \;=\; \Omega\!\left(\frac{d \cdot L}{\log |\mathcal{V}|}\right), \tag{2}$$

where $|\mathcal{V}|$ is the vocabulary size. Equivalently, the KV cache is $\Omega(d/\log|\mathcal{V}|)$ times more information-dense per position than text.

3.2 Efficient Verification via KV-Cache Readout

Proposition 1 establishes that the KV cache is an information-rich representation for verification. We now formalize how this richness can be exploited efficiently.

Definition 1 (Depth-k Readout)**.** *A* depth-k readout $\mathcal{R}_k : (\mathbf{K}, \mathbf{V}) \mapsto s \in [0, 1]$ *processes k query vectors against the pre-existing KV cache of length L through a parameterized attention function f_ψ. We denote by $\mathcal{F}_k$ the function class of all depth-k readouts with bounded parameter capacity $\|\psi\| \le B$. Text re-encoding corresponds to $k = L$; KV-cache readout uses $k \ll L$.*

Definition 2 (Approximation Gap)**.** *For a function class $\mathcal{F}_k$ operating on KV cache $\mathbf{H}$, the* approximation gap *is:*

$$\text{Gap}(\mathcal{F}_k,\, \mathbf{H}) \;=\; \inf_{f \in \mathcal{F}_k} \mathbb{E}[\ell(f(\mathbf{H}),\, Y)] \;-\; \inf_{f} \mathbb{E}[\ell(f(\mathbf{H}),\, Y)], \tag{3}$$

where the second infimum is over all measurable functions. $\text{Gap}(\mathcal{F}_k, \mathbf{H})$ *is monotonically non-increasing in k.*

Theorem 1 (Verification Error Decomposition)**.** *For any depth-k readout $\mathcal{R}_k \in \mathcal{F}_k$, the verification error decomposes as:*

$$\epsilon(\mathcal{R}_k) = \epsilon_{\text{Bayes}}(\mathbf{H}) + \text{Gap}(\mathcal{F}_k, \mathbf{H}), \tag{4}$$

where $\epsilon_{\text{Bayes}}(\mathbf{H}) \leq \epsilon_{\text{Bayes}}(\mathbf{x})$ by Proposition 1. That is, the Bayes-optimal predictor from the KV cache achieves equal or lower error than the Bayes-optimal predictor from text. The approximation gap $\text{Gap}(\mathcal{F}_k, \mathbf{H})$ is controlled by the readout depth k and is monotonically non-increasing in k.

Remark 1. Theorem 1 decompose the verification error to a lower bound determined by the information richness of the input and a gap between the empirical and optimal verifier, where the former favors KV cache given its representational adventage proven in Proposition 1.

Assumption 2 (Low-Rank Reward Structure)**.** *The reward Y depends on the KV cache $\mathbf{H}$ through a linear projection $\Phi(\mathbf{H}) = W_\phi \,\text{pool}(\mathbf{H}) \in \mathbb{R}^r$, where r is the effective reward dimensionality and $W_\phi \in \mathbb{R}^{r \times d}$. The singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$ of the covariance of $\Phi(\mathbf{H})$ projected onto the reward-relevant subspace decay as $\sigma_j = O(e^{-\alpha j})$ for some spectral decay rate $\alpha > 0$ [17, 18].*

Theorem 2 (Diminishing Returns of Readout Depth)**.** *Under Assumptions 1 and 2, let $\mathcal{R}_k \in \mathcal{F}_k$ be a depth-k readout over KV cache $\mathbf{H}$, and define the marginal information gain $\Delta I_k = I(\mathcal{R}_k(\mathbf{H}); Y \mid \mathcal{R}_{k-1}(\mathbf{H}))$. Then:*

(a) ***Exponential decay.*** *The marginal gain decays exponentially:*

$$\Delta I_k \;\leq\; C_0 \cdot e^{-\alpha \cdot n_h N_L \cdot (k-1)}, \tag{5}$$

where $C_0 = \sum_{j=1}^{r} \frac{1}{2} \log(1 + \text{SNR} \cdot \sigma_j^2)$ is the total extractable information, with $\text{SNR} = \text{Var}(Y)/\sigma_\epsilon^2$ the signal-to-noise ratio.

(b) $k=1$ ***near-optimality.*** *The fraction of reward information captured at $k = 1$ satisfies:*

$$\frac{I(\mathcal{R}_1(\mathbf{H}); Y)}{I(\mathbf{H}; Y)} \;\geq\; 1 - \frac{e^{-\alpha \cdot n_h N_L}}{1 - e^{-\alpha \cdot n_h N_L}}. \tag{6}$$

(c) ***Cost–information tradeoff.*** *The information gained per additional FLOP decays as:*

$$\frac{\Delta I_k}{F_{\text{readout}}(k, L) - F_{\text{readout}}(k-1, L)} = O\left(\frac{e^{-\alpha \cdot n_h N_L \cdot (k-1)}}{d \cdot L}\right). \tag{7}$$

Remark 2. Theorem 2 shows that reward modeling on KV cache is cost-efficient, where using a single query token is the most cost-efficient operating point, readily capturing most of the information.

4 Method

Motivated by the theoretical analysis in Section 3, we introduce KV-PRM, a process reward model that instantiates the KV-cache readout framework (Definition 1) at $k = 1$, achieving the maximum $\Omega(L)$ speedup by scoring trajectories through the KV cache already produced during generation.

4.1 KV-PRM: Scoring via KV-Cache Transfer

During generation, the LLM naturally produces a KV cache, encoding the full reasoning trajectory. KV-PRM reuses this pre-existing cache to score trajectory quality, without re-encoding.

Architecture. Let f_θ denote the base language model with parameters θ, and let $f_{\theta+\Delta\theta}$ denote the same model with LoRA adapter parameters $\Delta\theta$ [10]. We designate a *verify token* v (the "?" token), and two *judgment tokens*: "+" (positive) and "−" (negative). We use text tokens to keep the output aligned with the base model's pre-trained vocabulary distribution [19].

Scoring procedure. As shown in Figure 1, after agent a_j completes generation, the accumulated KV cache $(\mathbf{K}, \mathbf{V})$ already exists as a byproduct of the generation process. KV-PRM scores the trajectory by processing a single verify token with the LoRA adapter activated:

$$\mathbf{z} = f_{\theta+\Delta\theta}(v \mid \mathbf{K}, \mathbf{V}), \qquad s_j = P(+) = \text{softmax}\big([\mathbf{z}_{[-]},\, \mathbf{z}_{[+]}]\big)_1 \tag{8}$$

The LoRA adapter learns to interpret the base model's KV cache, which encodes position information, attention patterns, and hidden state interactions of the full trajectory, and outputs a quality judgment through the verify token's logits. Crucially, the KV cache is produced by the base model with the

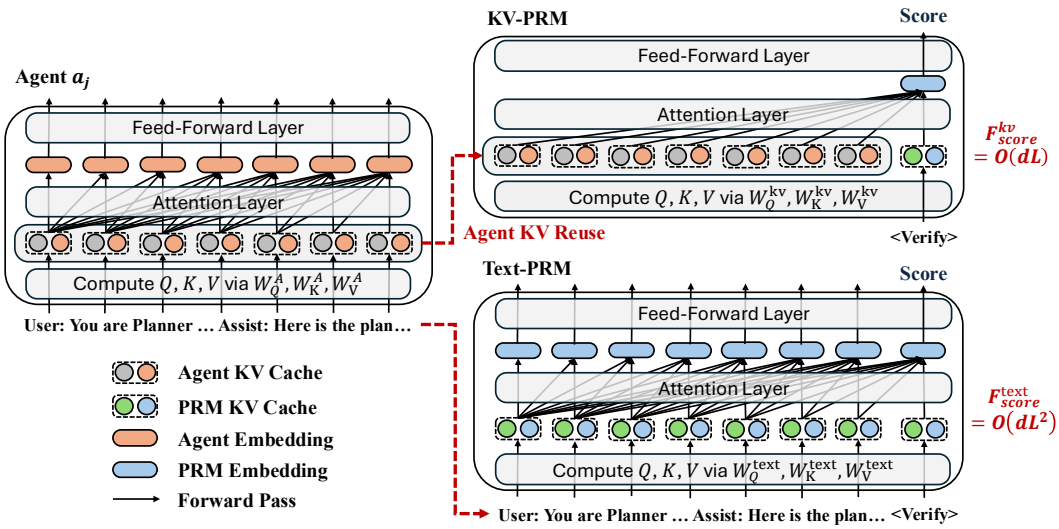

Figure 1: Text-PRM re-encodes the full trajectory for each scoring call at $O(dL^2)$ FLOPs. KV-PRM reuses the KV cache from generation and scores with a single verify token at $O(dL)$ FLOPs, a reduction of $L\times$, which is three orders of magnitude for typical MAS trajectories.

adapter *disabled* during normal generation, so KV-PRM introduces no overhead to the generation process itself.

Cost analysis. The verify-token forward pass processes a single query token against L cached positions across N_L layers:

$$F_{\text{score}}^{\text{kv}} = N_L \cdot (c_{\text{attn}} \cdot d \cdot L + c_{\text{ffn}} \cdot d^2) = O(d \cdot L) \tag{9}$$

A text-based PRM must re-encode the full trajectory from scratch, costing $F_{\text{score}}^{\text{text}} = O(d \cdot L^2)$ (Equation 1). The per-call speedup is:

$$\frac{F_{\text{score}}^{\text{text}}}{F_{\text{score}}^{\text{kv}}} = \frac{O(d \cdot L^2)}{O(d \cdot L)} = O(L) \tag{10}$$

For typical MAS trajectories ($L \approx 5{,}000$ tokens), this yields $\sim 5{,}000\times$ fewer FLOPs per scoring call. The same factor-of-L reduction extends to peak activation memory: Text-PRM materializes $O(L^2)$ attention maps per layer, while KV-PRM requires only $O(L)$. In test-time search, where the PRM is invoked $O(W \cdot D)$ times, this per-call reduction makes the total scoring cost negligible relative to generation, eliminating the computational bottleneck identified in Section 2. This instantiation corresponds to $k = 1$ in the readout framework, achieving the maximum theoretical speedup.

Inference. KV-PRM is a drop-in replacement for text-based PRMs in all search algorithms described in Section 2: beam search, MCTS, and weighted majority voting. The scoring mechanism changes: KV-PRM reads the pre-existing KV cache instead of re-encoding text, but the search algorithms themselves remain unchanged.

4.2 Training

Training data generation. We generate training labels using Monte Carlo Tree Search (MCTS) [6] over the multi-agent reasoning process. For each training question, we run R MCTS rollouts with C candidates per agent step. Terminal trajectories are evaluated by exact-match against the ground truth, producing binary rewards. Backpropagation through the search tree assigns Q-values $y \in [-1, 1]$ to each intermediate agent step. Training data is generated separately for each (topology, execution mode) combination. Each (trajectory, agent step) pair produces one training sample $(\mathbf{x}_j, y_j)$.

KV-PRM training. For each sample $(\mathbf{x}, y)$, the base model (adapter disabled) processes $\mathbf{x}$ to produce KV cache $(\mathbf{K}, \mathbf{V})$ without gradients. The model (adapter active) processes the verify token conditioned on $(\mathbf{K}, \mathbf{V})$, producing score $P(+)$ via Equation 8. The loss is:

$$\mathcal{L}_{\text{KV}} = \text{MSE}\left(P(+),\ \frac{y+1}{2}\right) \tag{11}$$

Algorithm 1 KV-PRM Training

Require: Training set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$ with trajectories and MCTS Q-values, base model f_θ, LoRA parameters $\Delta\theta$, learning rate η

for each mini-batch $\mathcal{B} \subset \mathcal{D}$ **do**
  for each $(\mathbf{x}, y) \in \mathcal{B}$ **do**
    $(\mathbf{K}, \mathbf{V}) \leftarrow f_\theta(\mathbf{x})$ *// Encode trajectory with adapter off (no grad)*
    $\mathbf{z} \leftarrow f_{\theta+\Delta\theta}(v \mid \mathbf{K}, \mathbf{V})$ *// Forward verify token with adapter on*
    $P(+) \leftarrow \text{softmax}\big([\mathbf{z}_{[-]},\ \mathbf{z}_{[+]}]\big)_1$
    $\mathcal{L} \leftarrow \text{MSE}\big(P(+),\ (y+1)/2\big)$
  end for
  $\Delta\theta \leftarrow \Delta\theta - \eta \nabla_{\Delta\theta} \frac{1}{|\mathcal{B}|} \sum \mathcal{L}$ *// Update only LoRA parameters*
end for

Only the LoRA parameters $\Delta\theta$ are updated. Gradients flow only through the single verify-token forward pass. Algorithm 1 summarizes the procedure. The KV cache is generated on the fly rather than stored offline; the resulting training cost is analyzed in Appendix E.4.

5 Experiments

5.1 Setup

Datasets and metrics. We evaluate on four mathematical reasoning benchmarks: MATH [20], GSM8K [21], AIME 2024, and AIME 2025, reporting exact-match accuracy with numeric equivalence checking. We evaluate three search algorithms: step-level beam search (SBS), MCTS, and weighted majority voting, with per-dataset configurations reported in Tables 1–2. Scoring cost is normalized to Text-PRM $= 1\times$. Both KV-PRM and Text-PRM are trained with LoRA ($r = 256$, $\alpha = 32$) on MCTS-generated labels ($R = 64$ rollouts, $C = 4$ candidates per step) from the MATH training split. Full hyperparameters and implementation details are in Appendix B.

Models and MAS configurations. We use three scales of the Qwen3 family [22]: 0.6B, 4B, and 8B, with thinking mode disabled, under two representative multi-agent topologies: **Sequential** (Reader $\rightarrow$ Planner $\rightarrow$ Solver $\rightarrow$ Verifier) and **Hierarchical** (Math/Science/Code Agents $\rightarrow$ Task Summarizer). To test generalization across model families, we additionally evaluate Llama-3.2-3B-Instruct [23] and Ministral-3-3B-Instruct [24].

Baselines. We compare KV-PRM against three scoring strategies: (1) **No PRM** (Random Sampling / Majority Voting), where the MAS generates without process reward guidance; (2) **Policy Log-prob**, which ranks trajectories by the generator's own token-level log-probabilities at negligible additional cost; and (3) **Text-PRM**, a LoRA adapter trained with the same architecture, data, and objective as KV-PRM (Equation 11), but scoring by re-encoding the full trajectory $[\mathbf{x}; v]$ in a single forward pass with the adapter active on all tokens at $O(dL^2)$ cost per call. Text-PRM training requires gradient checkpointing due to full-sequence gradient computation.

5.2 Main Results

KV-PRM outperforms Text-PRM across a broad range of settings while reducing scoring cost by 3 orders of magnitude. Table 1 and Table 2 show that KV-PRM achieves this accuracy-efficiency trade-off with dramatically lower scoring FLOPs per call, with relative cost reductions ranging from roughly 9×10^2 to $4.9 \times 10^3\times$ compared to Text-PRM, depending on the benchmark, model scale, and search configuration. The accuracy results show that replacing full-text re-encoding with single-token KV-cache readout improves, or at least preserves, scorer quality while making verification far cheaper.

KV-PRM significantly reduces the inference latency by 1-2 orders of magnitude. We validate the theoretical complexity reduction with wall-clock latency measurements on an NVIDIA GH200 GPU. As shown in the left panel of Figure 3. KV-PRM consistently delivers substantial per-sequence speedups over Text-PRM, ranging from $15\times$ to $37\times$ depending on model scale and sequence length. The gap widens as sequences become longer and models become larger, in line with the expected $O(L)$ vs. $O(L^2)$ scoring-cost difference. At 8B and $L = 4096$, a single KV-PRM scoring call takes

Table 1: Test-time search accuracy (%) and relative scoring cost on **MATH** and **AIME 2024** under sequential multi-agent execution. Scoring cost normalized to Text-PRM $= 1\times$; Policy Log-prob cost is negligible (—); KV-PRM cost $= 1/L$ where L is the maximum mean per-step trajectory length.

			MATH						**AIME 2024**				
			Qwen3-0.6B		**Qwen3-4B**		**Qwen3-8B**		**Qwen3-4B**		**Qwen3-8B**		
Search	**Config**	**Scorer**	Acc	Cost	Acc	Cost	Acc	Cost	Acc	Cost	Acc	Cost	**Avg.**
Random Sampling	$n=1$	—	28.00		54.15		56.70		20.00		20.00		35.77
Majority Voting	$n=40$	—	34.65		59.20		60.55		36.67		33.33		44.88
Beam Search	$W=1$	Policy Log-prob	32.50	—	58.70	—	63.70	—	23.33	—	23.33	—	40.31
		Text-PRM	32.75	$1\times$	62.00	$1\times$	66.62	$1\times$	26.67	$1\times$	20.00	$1\times$	41.61
		KV-PRM	31.90	$\frac{1}{2409}$	**65.55**	$\frac{1}{2543}$	**67.02**	$\frac{1}{2782}$	20.00	$\frac{1}{2007}$	**30.00**	$\frac{1}{2367}$	**42.89**
MCTS	$n=50$	Policy Log-prob	29.40	—	63.90	—	61.90	—	20.00	—	23.33	—	39.71
		Text-PRM	35.55	$1\times$	65.95	$1\times$	68.92	$1\times$	30.00	$1\times$	36.67	$1\times$	47.42
		KV-PRM	35.10	$\frac{1}{3598}$	**67.15**	$\frac{1}{3771}$	**69.57**	$\frac{1}{4367}$	**33.33**	$\frac{1}{3652}$	**40.00**	$\frac{1}{3944}$	**49.03**
Weighted Voting	$n=10$	Policy Log-prob	33.75	—	58.45	—	59.95	—	30.00	—	33.33	—	43.10
		Text-PRM	34.20	$1\times$	58.75	$1\times$	60.51	$1\times$	36.67	$1\times$	23.33	$1\times$	42.69
		KV-PRM	**34.40**	$\frac{1}{2444}$	**59.35**	$\frac{1}{2749}$	60.46	$\frac{1}{3137}$	33.33	$\frac{1}{2360}$	**33.33**	$\frac{1}{2575}$	**44.17**
	$n=200$	Policy Log-prob	35.35	—	59.40	—	60.85	—	36.67	—	33.33	—	45.12
		Text-PRM	36.05	$1\times$	59.95	$1\times$	61.26	$1\times$	36.67	$1\times$	33.33	$1\times$	45.45
		KV-PRM	**36.15**	$\frac{1}{2441}$	**60.15**	$\frac{1}{2721}$	61.06	$\frac{1}{3086}$	**36.67**	$\frac{1}{2298}$	**36.67**	$\frac{1}{2539}$	**46.14**

Table 2: Test-time search accuracy (%) and relative scoring cost on **GSM8K** and **AIME 2025** under sequential multi-agent execution. Same format as Table 1.

			GSM8K						**AIME 2025**				
			Qwen3-0.6B		**Qwen3-4B**		**Qwen3-8B**		**Qwen3-4B**		**Qwen3-8B**		
Search	**Config**	**Scorer**	Acc	Cost	Acc	Cost	Acc	Cost	Acc	Cost	Acc	Cost	**Avg.**
Random Sampling	$n=1$	—	50.80		91.28		91.43		16.67		20.00		54.04
Majority Voting	$n=40$	—	62.77		93.10		93.03		20.00		16.67		57.11
Beam Search	$W=10$	Policy Log-prob	52.99	—	90.98	—	93.40	—	13.33	—	16.67	—	53.47
		Text-PRM	58.68	$1\times$	93.78	$1\times$	95.00	$1\times$	26.67	$1\times$	26.67	$1\times$	60.16
		KV-PRM	58.00	$\frac{1}{1341}$	92.80	$\frac{1}{1311}$	**95.07**	$\frac{1}{1295}$	**26.67**	$\frac{1}{4266}$	**30.00**	$\frac{1}{4392}$	**60.51**
MCTS	$n=30$	Policy Log-prob	52.92	—	89.99	—	91.13	—	13.33	—	16.67	—	52.81
		Text-PRM	57.77	$1\times$	93.33	$1\times$	95.15	$1\times$	23.33	$1\times$	23.33	$1\times$	58.58
		KV-PRM	54.74	$\frac{1}{1357}$	92.87	$\frac{1}{1386}$	94.62	$\frac{1}{1402}$	**23.33**	$\frac{1}{4877}$	**26.67**	$\frac{1}{4840}$	58.45
Weighted Voting	$n=20$	Policy Log-prob	61.64	—	92.72	—	92.87	—	16.67	—	16.67	—	56.11
		Text-PRM	62.40	$1\times$	93.18	$1\times$	93.18	$1\times$	13.33	$1\times$	16.67	$1\times$	55.75
		KV-PRM	**63.61**	$\frac{1}{908}$	92.87	$\frac{1}{1006}$	93.03	$\frac{1}{1036}$	**20.00**	$\frac{1}{3402}$	**20.00**	$\frac{1}{3415}$	**57.90**
	$n=100$	Policy Log-prob	63.00	—	92.80	—	92.95	—	16.67	—	16.67	—	56.42
		Text-PRM	63.38	$1\times$	93.18	$1\times$	93.25	$1\times$	20.00	$1\times$	16.67	$1\times$	57.30
		KV-PRM	**63.76**	$\frac{1}{934}$	**93.18**	$\frac{1}{997}$	**93.25**	$\frac{1}{1025}$	**23.33**	$\frac{1}{3428}$	**16.67**	$\frac{1}{3380}$	**58.04**

only 4.6 ms, compared with 172.0 ms for Text-PRM. This confirms that KV-cache transfer translates the theoretical FLOP savings into large practical reductions in end-to-end scorer latency.

KV-PRM significantly reduces the memory consumption during inference by 1-2 orders of magnitude. As shown in the right panel of Figure 3, KV-PRM reduces the average memory increase per sequence by up to $34.2\times$, relative to Text-PRM under matched batch sizes. This gain comes from avoiding full-sequence re-encoding and the associated attention-map materialization, allowing KV-PRM to score trajectories with substantially lower activation overhead. In practice, the smaller memory footprint enables larger verification batches on the same hardware, providing a complementary systems advantage beyond latency alone.

5.3 Analysis

Can smaller Text-PRMs close the efficiency gap? A natural strategy for reducing Text-PRM scoring cost is to use a smaller verifier model, which is a common practice in the literature [2]. We evaluate this on the MATH dataset, using Qwen3-8B as the base generator and scoring the same search trees with Text-PRMs of sizes 0.6B, 4B, and 8B alongside KV-PRM of size 8B. Figure 2(Left) plots accuracy against relative scoring cost per call for SBS $W=2$. Scaling up the Text-PRM from 0.6B to 4B to 8B improves accuracy, but with diminishing returns and increasing cost. KV-PRM achieves the *highest* accuracy (68.0%), surpassing all Text-PRM sizes, at $\sim$3,000$\times$ lower cost than

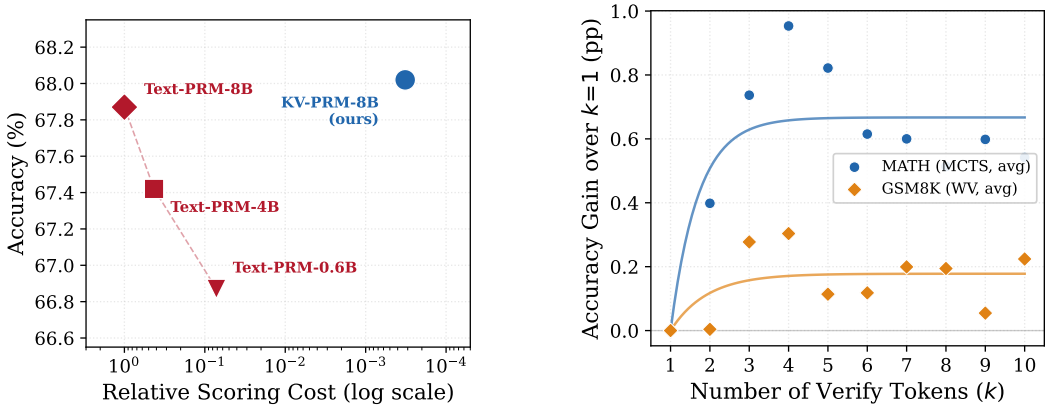

Figure 2: Analysis of KV-PRM efficiency and readout depth. *Left*: Scaling up the Text-PRM from 0.6B to 8B yields diminishing returns; KV-PRM surpasses all sizes at $\sim 10^3\times$ lower cost. *Right*: Accuracy gain over $k = 1$ as a function of the number of verify tokens k averaged across search hyperparameters. Fitted curves $a(1 - e^{-c(k-1)})$ confirm exponentially diminishing returns.

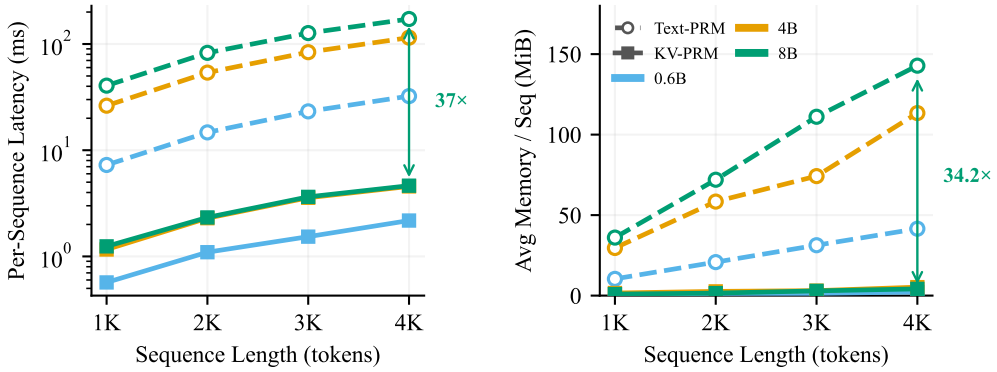

Figure 3: Wall-clock efficiency on NVIDIA GH200 120GB GPU. *Left*: Per-sequence scoring latency (log scale). KV-PRM (solid) is 1–2 orders of magnitude faster than Text-PRM (dashed), reaching $37\times$ at 8B ($L = 4{,}096$). *Right*: Per-sequence average memory increase during PRM inference. KV-PRM reduces memory by up to $34.2\times$, enabling substantially larger batches on the same hardware.

Text-PRM-8B, demonstrating that KV-cache transfer is a fundamentally more efficient strategy than reducing the size of the PRM.

What happens with more verify tokens ($k > 1$)? Theorem 2 makes two concrete predictions: (a) the marginal information gain ΔI_k from each additional verify token decays exponentially in k at rate $\alpha \cdot n_h N_L$ (Equation 5), and (b) a single token ($k = 1$) already captures most extractable reward information (Equation 6). Figure 2(Right) tests these predictions on Qwen3-0.6B ($n_h \cdot N_L = 448$), plotting accuracy gain over $k = 1$ averaged across search hyperparameters. Both curves are well fit by $a(1 - e^{-c(k-1)})$, consistent with the exponential decay form $e^{-\alpha \cdot n_h N_L \cdot (k-1)}$ predicted by Theorem 2(a). The rapid saturation validates the $k = 1$ near-optimality bound (Theorem 2b): even at the smallest model scale, a single verify token captures most of the available reward signal. By the cost, information tradeoff (Theorem 2c), the information gained per additional FLOP decays as $O(e^{-\alpha \cdot n_h N_L \cdot (k-1)}/(dL))$, so even at $k = 4$ the total scoring cost of $4 \cdot O(dL)$ remains $\sim 10^3\times$ cheaper than text re-encoding at $O(dL^2)$.

Does KV-PRM work on more MAS topologies? We additionally evaluate KV-PRM under a hierarchical multi-agent topology. Table 3 reports results across all four datasets. KV-PRM matches or outperforms Text-PRM in the majority of search configurations. On MATH, the advantage is consistent but modest (+0.5–0.8 pp), while on AIME benchmarks the gaps are more pronounced (up to +6.7 pp), reflecting greater headroom for scorer quality on harder problems. These results confirm that the KV-cache scoring mechanism works effectively for non-sequential MAS topologies.

Table 3: Hierarchical MAS accuracy (%) across datasets and model scales. KV-PRM matches or exceeds Text-PRM on all dataset–model combinations. Δ = KV-PRM $-$ Text-PRM.

Dataset	Model	Text-PRM	KV-PRM	Δ
MATH	Qwen3-4B	58.7	**59.2**	+0.5
	Qwen3-8B	60.2	**61.0**	+0.8
GSM8K	Qwen3-4B	93.7	**93.9**	+0.2
	Qwen3-8B	94.2	**94.5**	+0.3
AIME 2024	Qwen3-4B	23.3	**26.7**	+3.3
	Qwen3-8B	23.3	**30.0**	+6.7
AIME 2025	Qwen3-4B	23.3	**26.7**	+3.3
	Qwen3-8B	16.7	**20.0**	+3.3

Table 4: KV Steering proof-of-concept: accuracy with gradient-based KV-cache optimization at agent handoffs, without any search. This is structurally impossible with text-based PRMs.

Dataset	Model	No steering	KV Steering	Δ
MATH	Qwen3-4B	60.23	**61.84**	+1.61
	Qwen3-8B	60.79	**62.00**	+1.21
GSM8K	Qwen3-4B	91.81	**92.42**	+0.61
	Qwen3-8B	93.86	**94.09**	+0.23
AIME 2024	Qwen3-4B	10.00	**13.33**	+3.33
	Qwen3-8B	13.33	**16.67**	+3.33
AIME 2025	Qwen3-4B	10.00	10.00	+0.00
	Qwen3-8B	13.33	**16.67**	+3.33

Table 5: Test-time search accuracy (%) on **Llama-3.2-3B-Instruct** and **Ministral-3-3B-Instruct**. KV-PRM: native verifier; KV-PRM-X: verifier of family X reading the generator's KV cache through a cross-architecture adapter.

		Generator: Llama-3.2-3B					Generator: Ministral-3-3B					
Search	**Scorer**	MATH	GSM8K	AIME 24	AIME 25	**Avg.**	**Scorer**	MATH	GSM8K	AIME 24	AIME 25	**Avg.**
Random Sampling ($n=1$)	—	45.35	74.07	13.33	0.00	33.19	—	50.25	87.49	10.00	6.67	38.60
Majority Voting ($n=160$)	—	54.85	84.53	16.67	0.00	39.01	—	57.45	92.87	23.33	20.00	48.41
Beam Search $W=4$	Policy Log-prob	52.80	80.82	10.00	0.00	35.90	Policy Log-prob	56.10	90.83	16.67	16.67	45.06
	Text-PRM	58.00	86.73	13.33	0.00	39.52	Text-PRM	63.45	92.80	26.67	20.00	50.73
	KV-PRM-Qwen3	52.55	85.37	6.67	0.00	36.15	KV-PRM-Qwen3	60.60	91.96	23.33	16.67	48.14
	KV-PRM-Ministral	54.65	85.29	6.67	0.00	36.65	KV-PRM-Llama	58.45	91.66	23.33	20.00	48.36
	KV-PRM	57.95	86.35	**13.33**	**0.00**	39.41	KV-PRM	**63.50**	**93.03**	20.00	**26.67**	**50.80**
MCTS $n=160$	Policy Log-prob	52.25	78.39	6.67	0.00	34.33	Policy Log-prob	55.85	91.51	20.00	16.67	46.01
	Text-PRM	59.35	88.10	16.67	0.00	41.03	Text-PRM	65.95	93.48	20.00	23.33	50.69
	KV-PRM-Qwen3	55.70	86.88	6.67	0.00	37.31	KV-PRM-Qwen3	61.80	93.10	20.00	20.00	48.73
	KV-PRM-Ministral	57.00	87.26	10.00	0.00	38.57	KV-PRM-Llama	60.70	93.03	20.00	23.33	49.26
	KV-PRM	59.25	**88.25**	**16.67**	**3.33**	**41.87**	KV-PRM	65.90	92.87	**23.33**	**23.33**	**51.36**
Weighted Voting $n=200$	Policy Log-prob	55.05	83.55	16.67	0.00	38.82	Policy Log-prob	57.45	93.03	23.33	20.00	48.45
	Text-PRM	54.65	85.67	13.33	0.00	38.41	Text-PRM	57.90	93.40	23.33	20.00	48.66
	KV-PRM-Qwen3	55.15	84.84	10.00	0.00	37.50	KV-PRM-Qwen3	57.25	93.10	23.33	20.00	48.42
	KV-PRM-Ministral	55.35	85.37	10.00	0.00	37.68	KV-PRM-Llama	57.15	93.03	23.33	20.00	48.38
	KV-PRM	**55.30**	**85.75**	**13.33**	**0.00**	**38.60**	KV-PRM	**57.90**	**93.48**	**23.33**	**20.00**	**48.68**

Does KV-PRM generalize to other model families? Table 5 reports results on Llama-3.2-3B and Ministral-3-3B. The native KV-PRM surpasses Text-PRM in average accuracy in five of the six backbone–search combinations and trails it by only 0.11 pp in the remaining one, while its scoring cost remains three orders of magnitude lower, as on Qwen3 (Appendix E.1). This confirms that KV-cache transfer is not specific to the Qwen3 architecture.

Can KV-PRM score a generator of a different architecture? To relax the requirement of a shared architecture, we train a lightweight adapter that converts the generator's KV cache into the cache geometry of a frozen KV-PRM of another family, which then scores it with a single verify token without re-prefilling the context (details in Appendix E.2). As shown in Table 5, such cross-architecture variants trail the native KV-PRM by 2.1–4.6 pp in average accuracy under beam search and MCTS, yet still surpass Policy Log-prob in these settings for both generators, demonstrating the feasibility of extending KV-PRM to heterogeneous generator–verifier pairs. Under weighted voting, where all scorers lie within 1.4 pp of each other, they fall slightly below Policy Log-prob.

Does KV-PRM transfer beyond math? We apply the math-trained KV-PRM and Text-PRM, without retraining, to guide test-time search on HumanEval [25] and HumanEval+ [26]. As shown in Table 6, KV-PRM raises HumanEval accuracy from 71.34% (random sampling) to 80.49% with MCTS, outperforms Policy Log-prob in all settings, and stays within 0.61 pp of, or above, Text-PRM.

Additional analyses. Comparisons with a last-hidden-state value head and with end-only verification, as well as score calibration and training cost, are provided in Appendix E.

6 Discussion on Broader Applications of KV-PRM

As KV-PRM scores are differentiable with respect to the KV cache, we explore whether gradient-based optimization of latent inter-agent messages [27] can improve generation quality, a capability structurally impossible with text-based PRMs. KV Steering performs gradient ascent on the accu-

Table 6: Test-time search accuracy (%) on code generation with PRMs trained on MATH only.

		Beam Search ($W=2$)			**MCTS** ($n=40$)		
Dataset	**Random Sampling**	Policy Log-prob	Text-PRM	KV-PRM	Policy Log-prob	Text-PRM	KV-PRM
HumanEval	71.34	75.00	78.66	78.05	76.83	79.27	**80.49**
HumanEval+	62.20	66.46	70.12	69.51	70.73	72.56	71.95

mulated KV cache at each agent handoff to maximize the KV-PRM score before the next agent begins generation (algorithm details in Appendix C). Table 4 presents proof-of-concept results. KV Steering improves accuracy on MATH by $+1.6$ pp (Qwen3-4B) and $+1.2$ pp (Qwen3-8B) without any search. On AIME benchmarks with Qwen3-8B, steering yields $+3.3$ pp gains on both AIME 2024 and AIME 2025. While preliminary, these results demonstrate that the KV-native architecture opens a qualitatively new direction for inference-time optimization beyond search.

7 Related Work

Process reward models. PRMs [9, 28, 29, 30] provide step-level supervision for mathematical reasoning, pioneered by Lightman et al. [7] who demonstrated that step-level human feedback improves over outcome-based reward models. Math-Shepherd [8] automates PRM label generation using MCTS. OmegaPRM [31] further scales automated PRM training. GenRM [32] frames verification as generation. All existing PRMs operate on text representations; KV-PRM is the first to score via KV-cache transfer, reducing the computational cost by orders of magnitude.

LLM-based multi-agent systems. Multi-agent systems have been explored in debate frameworks [33], collaborative problem-solving [34, 35], and structured pipelines [11]. MASPRM [12] introduced PRMs for multi-agent systems using text-based scoring. We address the computational bottleneck that MASPRM's text-based approach creates in test-time search.

KV-cache methods. Prior work has explored KV-cache compression [36], eviction [37], and memory management via paging [38] for efficient inference. These methods aim to reduce the *size* or improve the *allocation* of the KV cache; our work instead uses the KV cache as an *input representation* for reward modeling, a fundamentally different application.

Test-time compute scaling. Scaling test-time compute through search [2], self-consistency [14], tree-of-thoughts [39], MCTS [6], and reinforcement learning for reasoning [3] has shown consistent gains. However, the cost of scoring candidates, particularly ones with long trajectories, has received limited attention. Our work directly addresses this bottleneck.

8 Limitations and Conclusion

We presented KV-PRM, a process reward model that scores trajectories through a single verify-token forward pass over the generator's pre-existing KV cache, reducing per-call verifier cost from $O(dL^2)$ to $O(dL)$. Our theoretical framework proves that the KV cache is a richer representation than decoded text for verification (Proposition 1), and that the marginal information gain from additional readout tokens decays exponentially, making the efficient $k=1$ design near-optimal (Theorem 2). Empirically, KV-PRM matches or outperforms text-based PRMs across beam search, MCTS, and weighted voting on four benchmarks, three model families, and two MAS topologies, while delivering up to $\sim$5,000$\times$ fewer scoring FLOPs, $37\times$ lower latency, and $34\times$ less memory per sequence. The differentiability of KV-cache scoring further enables KV Steering, a proof-of-concept for gradient-based inter-agent message optimization that is structurally impossible with text-based PRMs. Limitations include the requirement that generator and verifier share the same architecture, which cache adaptation only partially relaxes, the need for provider-side KV-cache access (Appendix D), reliance on the Linear Representation Hypothesis in the theoretical bounds, and the preliminary nature of KV Steering. We hope this work encourages a shift from redundant text re-encoding toward representation reuse in verifier design.

Acknowledgments and Disclosure of Funding

This work was partially supported by the National Artificial Intelligence Research Resource (NAIRR) Pilot under awards NAIRR250400 and NAIRR240283, and Standing Up to POTS.

References

[1] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

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

[3] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[4] Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: A survey of progress and challenges. *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2024.

[5] Peng Kuang, Yanli Wang, Xiaoyu Han, Yaowenqi Liu, Kaidi Xu, and Haohan Wang. Optimal aggregation of LLM and PRM signals for efficient test-time scaling. In *The Fourteenth International Conference on Learning Representations*, 2026. URL `https://openreview.net/forum?id=x85kiYqL4y`.

[6] Xidong Feng, Ziyu Wan, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. Alphazero-like tree-search can guide large language model decoding and training. *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.

[7] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

[8] Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. *arXiv preprint arXiv:2312.08935*, 2024.

[9] Congmin Zheng, Jiachen Zhu, Zhuoying Ou, Yuxiang Chen, Kangning Zhang, Rong Shan, Zeyu Zheng, Mengyue Yang, Jianghao Lin, Yong Yu, and Weinan Zhang. A survey of process reward models: From outcome signals to process supervisions for large language models. *arXiv preprint arXiv:2510.08049*, 2025.

[10] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2022.

[11] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*, 2024.

[12] Milad Yazdani, Mahdi Mostajabdaveh, Zirui Zhou, and Ying Xiong. MASPRM: Multi-agent system process reward model. *arXiv preprint arXiv:2510.24803*, 2025.

[13] Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *European Conference on Machine Learning (ECML)*, pages 282–293. Springer, 2006.

[14] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2023.

[15] Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.

[16] Jiaru Zou, Xiyuan Yang, Ruizhong Qiu, Gaotang Li, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, James Zou, Mengdi Wang, and Ling Yang. Latent collaboration in multi-agent systems, 2025. URL https://arxiv.org/abs/2511.20639.

[17] Shi Fu, Yingjie Wang, Shengchao Hu, Peng Wang, and Dacheng Tao. Theoretical guarantees for iterative alignment of self-rewarding language models, 2026. URL https://openreview.net/forum?id=Dhd7nHdAaf.

[18] Yi Liu. The spectral geometry of thought: Phase transitions, instruction reversal, token-level dynamics, and perfect correctness prediction in how transformers reason, 2026. URL https://arxiv.org/abs/2604.15350.

[19] Wei Xiong, Hanning Zhang, Nan Jiang, and Tong Zhang. An Implementation of Generative PRM, 2024. URL https://github.com/RLHFlow/RLHF-Reward-Modeling. Publication Title: GitHub repository.

[20] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[21] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[22] An Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[23] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin

Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

[24] Alexander H. Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, Alexandre Sablayrolles, Amélie Héliou, Amos You, Andy Ehrenberg, Andy Lo, Anton Eliseev, Antonia Calvi, Avinash Sooriyarachchi, Baptiste Bout, Baptiste Rozière, Baudouin De Monicault, Clémence Lanfranchi, Corentin Barreau, Cyprien Courtot, Daniele Grattarola, Darius Dabert, Diego de las Casas, Elliot Chane-Sane, Faruk Ahmed, Gabrielle Berrada, Gaëtan Ecrepont, Gauthier Guinet, Georgii Novikov, Guillaume Kunsch, Guillaume Lample, Guillaume Martin, Gunshi Gupta, Jan Ludziejewski, Jason Rute, Joachim Studnia, Jonas Amar, Joséphine Delas, Josselin Somerville Roberts, Karmesh Yadav, Khyathi Chandu, Kush Jain, Laurence Aitchison, Laurent Fainsin, Léonard Blier, Lingxiao Zhao, Louis Martin, Lucile Saulnier, Luyu Gao, Maarten Buyl, Margaret Jennings, Marie Pellat, Mark Prins, Mathieu Poirée, Mathilde Guillaumin, Matthieu Dinot, Matthieu Futeral, Maxime Darrin, Maximilian Augustin, Mia Chiquier, Michel Schimpf, Nathan Grinsztajn, Neha Gupta, Nikhil Raghuraman, Olivier Bousquet, Olivier Duchenne, Patricia Wang, Patrick von Platen, Paul Jacob, Paul Wambergue, Paula Kurylowicz, Pavankumar Reddy Muddireddy, Philomène Chagniot, Pierre Stock, Pravesh Agrawal, Quentin Torroba, Romain Sauvestre, Roman Soletskyi, Rupert Menneer, Sagar Vaze, Samuel Barry, Sanchit Gandhi, Siddhant Waghjale, Siddharth Gandhi, Soham Ghosh, Srijan Mishra, Sumukh Aithal, Szymon Antoniak, Teven Le Scao, Théo Cachet, Theo Simon Sorg, Thibaut Lavril, Thiziri Nait Saada, Thomas Chabal, Thomas Foubert, Thomas Robert, Thomas Wang, Tim Lawson, Tom Bewley, Tom Bewley, Tom Edwards, Umar Jamil, Umberto Tomasini, Valeriia Nemychnikova, Van Phung, Vincent Maladière, Virgile Richard, Wassim Bouaziz, Wen-Ding Li, William Marshall, Xinghui Li, Xinyu Yang, Yassine El Ouahidi, Yihan Wang, Yunhao Tang, and Zaccharie Ramzi. Ministral 3, 2026. URL `https://arxiv.org/abs/2601.08584`.

[25] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021.

[26] Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL `https://openreview.net/forum?id=1qvx610Cu7`.

[27] Haibo Jin, Peng Kuang, Ye Yu, Xiaopeng Yuan, and Haohan Wang. Agent primitives: Reusable latent building blocks for multi-agent systems. In *Forty-third International Conference on Machine Learning*, 2026. URL `https://openreview.net/forum?id=CzShhpY2qU`.

[28] Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. Process reward models that think. *arXiv preprint arXiv:2504.16828*, 2025.

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

[30] Peng Kuang, Xiangxiang Wang, Wentao Liu, Jian Dong, and Kaidi Xu. Tim-prm: Verifying multimodal reasoning with tool-integrated prm, 2025. URL `https://arxiv.org/abs/2511.22998`.

[31] Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*, 2024.

[32] Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. *arXiv preprint arXiv:2408.15240*, 2024.

[33] Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*, 2023.

[34] Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: Communicative agents for "mind" exploration of large language model society. *Advances in Neural Information Processing Systems*, 2023.

[35] Yexiong Lin, Shanshan Ye, Yu Yao, Zhen Fang, Bo Han, and Tongliang Liu. Squidagent: Parallelize wisely, coordinate efficiently, 2026. URL `https://arxiv.org/abs/2610.08647`.

[36] Haoyang Li et al. A survey on large language model acceleration based on KV cache management. *arXiv preprint arXiv:2412.19442*, 2024.

[37] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

[38] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles (SOSP)*, 2023.

[39] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.

[40] Thomas M Cover and Joy A Thomas. *Elements of Information Theory*. Wiley-Interscience, 2nd edition, 2006.

A Theoretical Proofs

A.1 Proof of Proposition 1

Part (a): Information sufficiency. The generation process defines the Markov chain $Y \to \mathbf{H} \to \mathbf{x}$: for the KV cache $\mathbf{H} = (\mathbf{K}, \mathbf{V})$ and any target variable Y, text $\mathbf{x} = \mathrm{decode}(\mathbf{H})$ is a deterministic function of $\mathbf{H}$ (each token $x_t = \arg\max_v [W_{\mathrm{out}} h_t]_v$ or is sampled from $\mathrm{softmax}(W_{\mathrm{out}} h_t)$, where h_t is determined by $\mathbf{H}$). By the Data Processing Inequality [40, Theorem 2.8.1], for any Markov chain $Y \to \mathbf{H} \to \mathbf{x}$:

$$I(\mathbf{H};\, Y) \;\geq\; I(\mathbf{x};\, Y). \tag{12}$$

Equality holds if and only if $\mathbf{x}$ is a sufficient statistic of $\mathbf{H}$ for Y, i.e., $Y \perp \mathbf{H} \mid \mathbf{x}$. Since $\mathrm{decode}(\cdot)$ maps from $\mathbb{R}^{N_L \times L \times d}$ to $\{1, \ldots, |\mathcal{V}|\}^L$—a many-to-one projection at each position—distinct hidden states that carry different information about Y are generically collapsed to the same token, so equality does not hold in general. □

Part (b): Capacity bound. Under Assumption 1, the set of hidden embeddings is

$$\mathcal{H} = \left\{ \sum_{i=1}^{d} c_i s_i : c_1, \ldots, c_d \in \{0, \pm 1\} \right\}, \tag{13}$$

with $|\mathcal{H}| = 3^d$ (since the semantic basis is linearly independent). For a trajectory of length L, the KV cache encodes L such hidden states (considering the final-layer representation per position), giving $|\mathcal{H}^L| = 3^{dL}$ distinct configurations.

To represent these losslessly via text tokens from vocabulary $\mathcal{V}$, a text sequence of length m' can encode at most $|\mathcal{V}|^{m'}$ distinct configurations. Lossless representation requires:

$$|\mathcal{V}|^{m'} \;\geq\; 3^{dL}. \tag{14}$$

Taking logarithms:

$$m' \;\geq\; \frac{dL \cdot \log 3}{\log |\mathcal{V}|} \;=\; \Omega\!\left(\frac{d \cdot L}{\log |\mathcal{V}|} \right). \quad \square \tag{15}$$

A.2 Proof of Theorem 1

Verification error decomposition. For any representation R and function class $\mathcal{F}$, the expected loss of the best function in $\mathcal{F}$ decomposes as:

$$\inf_{f \in \mathcal{F}} \mathbb{E}[\ell(f(R), Y)] = \underbrace{\inf_{f} \mathbb{E}[\ell(f(R), Y)]}_{\epsilon_{\mathrm{Bayes}}(R)} + \underbrace{\inf_{f \in \mathcal{F}} \mathbb{E}[\ell(f(R), Y)] - \inf_{f} \mathbb{E}[\ell(f(R), Y)]}_{\mathrm{Gap}(\mathcal{F}, R)}. \tag{16}$$

This is a direct decomposition of the achievable error. Setting $R = \mathbf{H}$ and $\mathcal{F} = \mathcal{F}_k$ yields Equation 4. The inequality $\epsilon_{\mathrm{Bayes}}(\mathbf{H}) \leq \epsilon_{\mathrm{Bayes}}(\mathbf{x})$ follows from Proposition 1(a): since $I(\mathbf{H}; Y) \geq I(\mathbf{x}; Y)$, a predictor with access to $\mathbf{H}$ can achieve at least as low a Bayes error as one with access to $\mathbf{x}$ only.

Monotonicity of $\mathrm{Gap}(\mathcal{F}_k, \mathbf{H})$ in k: since $\mathcal{F}_k \subseteq \mathcal{F}_{k'}$ for $k \leq k'$ (a depth-k' readout can ignore $k' - k$ of its query tokens), we have $\inf_{f \in \mathcal{F}_{k'}} \leq \inf_{f \in \mathcal{F}_k}$, so the gap is non-increasing. □

A.3 Proof of Theorem 2

We prove each part of Theorem 2 under Assumptions 1 and 2.

Setup. Under Assumption 2, the reward Y depends on $\mathbf{H}$ through r features $\phi_1(\mathbf{H}), \ldots, \phi_r(\mathbf{H})$ with $\phi_j = (W_\phi)_j \, \mathrm{pool}(\mathbf{H})$, where the covariance eigenvalues satisfy $\sigma_j = O(e^{-\alpha j})$. Under a Gaussian channel model, the mutual information contributed by the j-th feature is:

$$I_j = \tfrac{1}{2} \log(1 + \mathrm{SNR} \cdot \sigma_j^2), \tag{17}$$

where $\mathrm{SNR} = \mathrm{Var}(Y)/\sigma_\epsilon^2$. The total extractable information is $C_0 = \sum_{j=1}^{r} I_j$.

Feature extraction capacity of depth-k readouts. A depth-k readout processes k query tokens through N_L transformer layers with n_h attention heads each. Each attention head computes a linear

combination of the cached value vectors via its attention weights: $\text{head}_i = \text{softmax}(q_i^\top K/\sqrt{d_k})V$, which is a linear function of $\mathbf{H}$ for fixed query q_i. With k query tokens, the readout has access to $k \cdot n_h \cdot N_L$ such linear projections of $\mathbf{H}$. Under Assumption 1, these suffice to extract $k \cdot n_h \cdot N_L$ independent linear features from the reward-relevant subspace.

Greedy ordering. An optimal readout extracts features in decreasing order of information contribution. After k depth steps, the readout has extracted features $\phi_1, \ldots, \phi_{k \cdot n_h N_L}$ (or all r features if $k \cdot n_h N_L \geq r$). The residual (unexplained) information is:

$$R_k = \sum_{j=k \cdot n_h N_L + 1}^{r} I_j \leq \sum_{j=k \cdot n_h N_L + 1}^{r} \tfrac{1}{2} \log(1 + \text{SNR} \cdot \sigma_j^2). \tag{18}$$

Part (a): Exponential decay. The marginal gain at depth k is bounded by the residual:

$$\Delta I_k \leq R_{k-1} - R_k = \sum_{j=(k-1)n_h N_L + 1}^{k \cdot n_h N_L} I_j. \tag{19}$$

Since $\sigma_j = O(e^{-\alpha j})$, we have $I_j = O(e^{-2\alpha j})$ for large j (using $\log(1+x) \leq x$ for small x). Thus:

$$\Delta I_k \leq \sum_{j=(k-1)n_h N_L + 1}^{k \cdot n_h N_L} O(e^{-2\alpha j}) = O(e^{-2\alpha(k-1)n_h N_L}) \cdot \sum_{i=1}^{n_h N_L} O(e^{-2\alpha i}) = O(e^{-2\alpha(k-1)n_h N_L}). \tag{20}$$

Since $e^{-2\alpha} \leq e^{-\alpha}$, we obtain the stated bound $\Delta I_k \leq C_0 \cdot e^{-\alpha \cdot n_h N_L \cdot (k-1)}$, where the constant C_0 absorbs the geometric series prefactor. □

Part (b): $k=1$ near-optimality. The information captured at $k = 1$ is $I(\mathcal{R}_1; Y) \geq C_0 - R_1$. The residual after $k = 1$ is:

$$R_1 = \sum_{j=n_h N_L + 1}^{r} I_j \leq C_0 \cdot \sum_{m=1}^{\infty} e^{-\alpha \cdot n_h N_L \cdot m} = C_0 \cdot \frac{e^{-\alpha \cdot n_h N_L}}{1 - e^{-\alpha \cdot n_h N_L}}. \tag{21}$$

Therefore:

$$\frac{I(\mathcal{R}_1; Y)}{I(\mathbf{H}; Y)} \geq \frac{C_0 - R_1}{C_0} \geq 1 - \frac{e^{-\alpha \cdot n_h N_L}}{1 - e^{-\alpha \cdot n_h N_L}}. \quad \square \tag{22}$$

Part (c): Cost–information tradeoff. The incremental cost of increasing depth from $k-1$ to k is:

$$F_{\text{readout}}(k, L) - F_{\text{readout}}(k-1, L) = N_L \cdot (c_{\text{attn}} \cdot d \cdot L + c_{\text{ffn}} \cdot d^2) = \Theta(d \cdot L). \tag{23}$$

Dividing part (a) by this constant-per-step cost:

$$\frac{\Delta I_k}{\Theta(d \cdot L)} = O\left(\frac{e^{-\alpha \cdot n_h N_L \cdot (k-1)}}{d \cdot L}\right). \quad \square \tag{24}$$

B Implementation Details

B.1 Training Hyperparameters

Table 7: Training hyperparameters for KV-PRM and Text-PRM.

Hyperparameter	KV-PRM	Text-PRM
LoRA rank r	256	256
LoRA α	32	32
LoRA dropout	0.05	0.05
Max sequence length	8192	8192
Effective global batch	2048	2048
Precision	bf16	bf16

B.2 Verify Token Details

The verify token is "?" (token ID determined by the tokenizer). The judgment tokens are "+" and "-". During scoring, we extract logits at these two positions and apply softmax to obtain $P(+)$. Labels are mapped from $y \in [-1, 1]$ to $\hat{y} = (y + 1)/2 \in [0, 1]$.

C KV Steering Algorithm

KV Steering exploits the differentiability of KV-PRM's KV-cache scoring to optimize inter-agent messages via gradient ascent on the process reward. At each agent handoff, the accumulated KV cache is refined before the next agent begins generation.

Algorithm 2 KV Steering at agent handoff $j \to j + 1$

Require: KV cache kv_j after agent a_j, step size η, number of steps T

$\mathrm{kv}^{(0)} \leftarrow \mathrm{kv}_j$
for $t = 0$ to $T - 1$ **do**
  $\widetilde{\mathrm{kv}}^{(t)} \leftarrow \mathrm{clone}(\mathrm{kv}^{(t)})$, set requires_grad $\leftarrow$ True
  $\ell \leftarrow -\log P(+)(\widetilde{\mathrm{kv}}^{(t)})$ *// Negative log-prob of positive judgment*
  $\mathrm{kv}^{(t+1)} \leftarrow \widetilde{\mathrm{kv}}^{(t)} - \eta \cdot \nabla_{\widetilde{\mathrm{kv}}} \ell$ *// Gradient ascent on reward*
end for
return $\mathrm{kv}^{(T)}$

D Deployment Scope

API-only models. As with most KV-cache methods, KV-PRM is not designed for users who only have API access to the generator. It is instead most useful on the model-provider side, including providers that serve closed-source frontier models.

KV-cache compression. Since KV-PRM shares the same architecture as the generator, it can consume any compressed cache, e.g., a quantized one, as long as the compressed cache can be consumed by the generator itself. Compatibility with KV-cache compression techniques [36] is thus guaranteed by the structural identity between KV-PRM and the generator.

KV-cache eviction. A KV cache does not persist and is evicted once it is deemed less important for future queries [37]. The ideal time to apply KV-PRM is therefore immediately after generation, while the cache is still in memory. Scoring then amounts to one additional single-token decoding step during generation, with negligible cost, especially in long-horizon settings, which cleanly decouples KV-PRM from, and makes it compatible with, existing KV-cache eviction methods.

E Additional Experimental Results

E.1 Full Results on Llama and Ministral

Tables 8 and 9 extend Table 5 with per-dataset scoring costs, a last-hidden-state baseline, end-only weighted voting, and an additional weighted voting configuration ($n = 40$) on Ministral-3-3B. End-only weighted voting scores only the output of the final agent.

Is the last hidden state sufficient for scoring? We compare KV-PRM with LHS-PRM, a value head trained on the generator's last hidden state, which also reuses generation activations at $O(dL)$ cost but only from the final layer. Although LHS-PRM and KV-PRM incur nearly identical scoring costs, LHS-PRM improves over Policy Log-prob in seven of the nine configurations, yet underperforms KV-PRM in all nine, by up to 3.8 pp in average accuracy. This indicates that earlier layers carry reward-relevant signal that a readout over the full KV cache can exploit.

Does intermediate scoring help over end-only verification? With KV-PRM, step-wise MCTS at $n = 160$ outperforms end-only weighted voting at $n = 200$ on both Llama-3.2-3B (41.87 vs. 40.36)

and Ministral-3-3B (51.36 vs. 49.05), achieving higher accuracy with less compute. The calibration of KV-PRM's step-level scores is analyzed in Appendix E.3.

Table 8: Test-time search accuracy (%) and relative scoring cost on **Llama-3.2-3B-Instruct**. Same format as Table 1.

			MATH		GSM8K		AIME 2024		AIME 2025		
Search	**Config**	**Scorer**	Acc	Cost	Acc	Cost	Acc	Cost	Acc	Cost	**Avg.**
Random Sampling	$n=1$	—	45.35		74.07		13.33		0.00		33.19
Majority Voting	$n=160$	—	54.85		84.53		16.67		0.00		39.01
Beam Search	$W=4$	Policy Log-prob	52.80	—	80.82	—	10.00	—	0.00	—	35.90
		LHS-PRM	56.65	$\frac{1}{3169}$	85.52	$\frac{1}{1412}$	10.00	$\frac{1}{2726}$	0.00	$\frac{1}{3133}$	38.04
		Text-PRM	58.00	$1\times$	86.73	$1\times$	13.33	$1\times$	0.00	$1\times$	39.52
		KV-PRM	57.95	$\frac{1}{3418}$	86.35	$\frac{1}{1411}$	**13.33**	$\frac{1}{2762}$	**0.00**	$\frac{1}{3133}$	39.41
MCTS	$n=160$	Policy Log-prob	52.25	—	78.39	—	6.67	—	0.00	—	34.33
		LHS-PRM	58.30	$\frac{1}{4269}$	87.26	$\frac{1}{2151}$	6.67	$\frac{1}{3402}$	0.00	$\frac{1}{4288}$	38.06
		Text-PRM	59.35	$1\times$	88.10	$1\times$	16.67	$1\times$	0.00	$1\times$	41.03
		KV-PRM	59.25	$\frac{1}{4266}$	**88.25**	$\frac{1}{2162}$	**16.67**	$\frac{1}{3403}$	**3.33**	$\frac{1}{4293}$	**41.87**
Weighted Voting	$n=200$	Policy Log-prob	55.05	—	83.55	—	16.67	—	0.00	—	38.82
		LHS-PRM	55.10	$\frac{1}{3058}$	85.44	$\frac{1}{1431}$	13.33	$\frac{1}{2450}$	0.00	$\frac{1}{2953}$	38.47
		Text-PRM	54.65	$1\times$	85.67	$1\times$	13.33	$1\times$	0.00	$1\times$	38.41
		KV-PRM	**55.30**	$\frac{1}{3058}$	**85.75**	$\frac{1}{1431}$	**13.33**	$\frac{1}{2450}$	**0.00**	$\frac{1}{2953}$	**38.60**
End-only Weighted Voting	$n=200$	Policy Log-prob	55.00	—	83.70	—	13.33	—	0.00	—	38.01
		LHS-PRM	54.85	$\frac{1}{4621}$	85.82	$\frac{1}{2270}$	16.67	$\frac{1}{3604}$	0.00	$\frac{1}{4545}$	39.33
		Text-PRM	55.30	$1\times$	86.43	$1\times$	16.67	$1\times$	0.00	$1\times$	39.60
		KV-PRM	**55.45**	$\frac{1}{4621}$	85.97	$\frac{1}{2270}$	**20.00**	$\frac{1}{3604}$	**0.00**	$\frac{1}{4545}$	**40.36**

Table 9: Test-time search accuracy (%) and relative scoring cost on **Ministral-3-3B-Instruct**. Same format as Table 1.

			MATH		GSM8K		AIME 2024		AIME 2025		
Search	**Config**	**Scorer**	Acc	Cost	Acc	Cost	Acc	Cost	Acc	Cost	**Avg.**
Random Sampling	$n=1$	—	50.25		87.49		10.00		6.67		38.60
Majority Voting	$n=160$	—	57.45		92.87		23.33		20.00		48.41
Beam Search	$W=4$	Policy Log-prob	56.10	—	90.83	—	16.67	—	16.67	—	45.06
		LHS-PRM	62.70	$\frac{1}{4091}$	92.04	$\frac{1}{2046}$	13.33	$\frac{1}{3396}$	20.00	$\frac{1}{4479}$	47.02
		Text-PRM	63.45	$1\times$	92.80	$1\times$	26.67	$1\times$	20.00	$1\times$	50.73
		KV-PRM	**63.50**	$\frac{1}{4067}$	**93.03**	$\frac{1}{2014}$	20.00	$\frac{1}{3426}$	**26.67**	$\frac{1}{4509}$	**50.80**
MCTS	$n=160$	Policy Log-prob	55.85	—	91.51	—	20.00	—	16.67	—	46.01
		LHS-PRM	65.30	$\frac{1}{5604}$	92.27	$\frac{1}{2710}$	20.00	$\frac{1}{4705}$	16.67	$\frac{1}{6139}$	48.56
		Text-PRM	65.95	$1\times$	93.48	$1\times$	20.00	$1\times$	23.33	$1\times$	50.69
		KV-PRM	65.90	$\frac{1}{5602}$	92.87	$\frac{1}{2709}$	**23.33**	$\frac{1}{4704}$	**23.33**	$\frac{1}{6135}$	**51.36**
Weighted Voting	$n=40$	Policy Log-prob	57.30	—	92.80	—	20.00	—	20.00	—	47.52
		LHS-PRM	57.20	$\frac{1}{3807}$	93.10	$\frac{1}{1759}$	23.33	$\frac{1}{2901}$	20.00	$\frac{1}{3886}$	48.41
		Text-PRM	57.45	$1\times$	93.03	$1\times$	23.33	$1\times$	20.00	$1\times$	48.45
		KV-PRM	57.30	$\frac{1}{3807}$	**93.25**	$\frac{1}{1759}$	**23.33**	$\frac{1}{2901}$	**23.33**	$\frac{1}{3886}$	**49.30**
	$n=200$	Policy Log-prob	57.45	—	93.03	—	23.33	—	20.00	—	48.45
		LHS-PRM	57.80	$\frac{1}{3784}$	93.40	$\frac{1}{1731}$	23.33	$\frac{1}{2877}$	20.00	$\frac{1}{3863}$	48.63
		Text-PRM	57.90	$1\times$	93.40	$1\times$	23.33	$1\times$	20.00	$1\times$	48.66
		KV-PRM	**57.90**	$\frac{1}{3784}$	**93.48**	$\frac{1}{1731}$	**23.33**	$\frac{1}{2877}$	**20.00**	$\frac{1}{3863}$	**48.68**
End-only Weighted Voting	$n=200$	Policy Log-prob	57.50	—	93.10	—	23.33	—	20.00	—	48.48
		LHS-PRM	58.85	$\frac{1}{5881}$	93.71	$\frac{1}{2981}$	20.00	$\frac{1}{5025}$	20.00	$\frac{1}{6545}$	48.14
		Text-PRM	58.95	$1\times$	93.86	$1\times$	23.33	$1\times$	23.33	$1\times$	49.87
		KV-PRM	**59.00**	$\frac{1}{5881}$	**93.86**	$\frac{1}{2981}$	20.00	$\frac{1}{5025}$	**23.33**	$\frac{1}{6545}$	49.05

E.2 Cross-Architecture KV-Cache Adapter

To apply KV-PRM when the generator and the verifier do not share the same architecture, we train a lightweight adapter that converts the generator's KV cache into the cache geometry of a receiver KV-PRM while preserving sequence length. Each receiver layer is paired with a generator layer by normalized-depth mapping, and its keys and values are transformed independently by lightweight

bottleneck residual projectors. Keys are first rotated out of the generator's RoPE coordinate system, projected to the receiver's KV-head dimensions, and then re-encoded with the receiver's RoPE; values are projected directly. The resulting receiver-shaped cache is scored by the frozen receiver KV-PRM using only the verify token, without re-prefilling the original context, so the scoring cost remains comparable to that of the native KV-PRM.

The results are reported in Table 5 and discussed in Section 5.3.

E.3 Score Calibration

Table 10 compares KV-PRM scores with the ground-truth value of each step, defined as the probability of reaching a correct answer from that step. The scores are well calibrated on MATH, which matches the training distribution. On GSM8K and AIME, KV-PRM tends to be under-confident and over-confident, respectively, in line with their relative difficulty compared to the MATH training data. Calibration is thus necessary if the scores are to be interpreted as probabilities, but not for test-time search, which only requires a sufficient ranking signal, as reflected by the positive Pearson correlation with the ground-truth value. The only exception is Llama-3.2-3B on AIME 2025, where the correlation is zero because the generator's accuracy on this dataset is zero.

Table 10: Calibration of KV-PRM scores against ground-truth step values (probability of reaching a correct answer from the step). ECE: expected calibration error.

Dataset	Backbone	Mean score	Mean value	ECE	Pearson
MATH	Qwen3-4B	0.6291	0.6198	0.0189	0.3816
	Llama-3.2-3B	0.6123	0.4960	0.1163	0.4515
	Ministral-3-3B	0.5806	0.5294	0.0524	0.5769
GSM8K	Qwen3-4B	0.7065	0.9155	0.2090	0.3303
	Llama-3.2-3B	0.6932	0.7818	0.0893	0.4730
	Ministral-3-3B	0.6771	0.8775	0.2004	0.3590
AIME 2024	Qwen3-4B	0.4145	0.1753	0.2427	0.4491
	Llama-3.2-3B	0.4063	0.0653	0.3410	0.1619
	Ministral-3-3B	0.2911	0.1021	0.1890	0.5649
AIME 2025	Qwen3-4B	0.4035	0.1419	0.2616	0.4950
	Llama-3.2-3B	0.3681	0.0052	0.3629	0.0000
	Ministral-3-3B	0.3089	0.1114	0.1975	0.6568

E.4 Training Cost

We compare the training cost of KV-PRM and Text-PRM using the same Ministral-3-3B backbone, optimizer, and input batches on a single NVIDIA A100 GPU. As shown in Table 11, KV-PRM achieves up to a $2.93\times$ training-step speedup and up to a 14.3% peak memory reduction, with both advantages growing with sequence length and batch size. Table 12 breaks down a training step at sequence length 8,192. KV-PRM spends 77.4% of its CUDA time in the gradient-free reference forward that produces the KV cache, 10.6% in the verify-token forward, and 11.5% in the backward pass, whereas Text-PRM spends 29.8% in the full forward and 70.2% in the backward pass. The end-to-end speedup thus mainly stems from the avoided full-sequence backward pass: the Text-PRM backward pass is $17.9\times$ slower than that of KV-PRM.

Table 11: Average time and peak memory per training step of KV-PRM and Text-PRM (Ministral-3-3B, one NVIDIA A100 GPU).

		Step time (ms)			Peak memory (MiB)		
Factor	**Setting**	KV-PRM	Text-PRM	**Speedup**	KV-PRM	Text-PRM	**Δ Peak**
Sequence length (batch 1)	1,024	162.28	217.20	1.34×	6,954	7,128	−2.4%
	2,048	206.20	314.37	1.52×	7,136	7,481	−4.6%
	4,096	294.93	659.49	2.24×	7,501	8,190	−8.4%
	8,192	487.82	1,431.67	2.93×	8,231	9,605	−14.3%
Batch size (length 1,024)	2	199.20	303.72	1.52×	7,136	7,481	−4.6%
	4	277.96	594.29	2.14×	7,500	8,188	−8.4%
	8	424.07	1,125.95	2.66×	8,228	9,604	−14.3%

Table 12: Breakdown of a training step at sequence length 8,192 (Ministral-3-3B, one NVIDIA A100 GPU).

Method	**Operation**	**CUDA time (ms)**	**Avg. allocated (MiB)**	**Peak allocated (MiB)**
KV-PRM	Reference forward (no grad)	377.55	7,608.17	8,231.49
	Verify forward	51.70	7,623.62	7,658.51
	Loss	0.15	7,642.46	7,642.71
	Backward	56.05	7,242.23	7,804.62
	Optimizer step	2.13	6,842.92	6,842.92
	Zero gradient	0.18	6,807.17	6,842.92
	Complete step	**487.80**	7,559.98	8,231.49
Text-PRM	Full forward	426.55	8,034.01	8,635.77
	Loss	0.02	8,119.80	8,119.93
	Backward	1,004.51	7,790.03	9,605.14
	Optimizer step	0.51	6,842.92	6,842.92
	Zero gradient	<0.01	6,807.17	6,842.92
	Complete step	**1,431.63**	7,915.88	9,605.14

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: The abstract and introduction clearly state three contributions (theoretical framework, KV-PRM algorithm, empirical validation) that are supported by the theoretical results in Section 3, the method description in Section 4, and the experiments in Section 5. Quantitative claims (e.g., 3,000× FLOP reduction, 37× latency speedup) are backed by Tables 1–2 and Figure 3.
   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: Section 6 explicitly discusses limitations: the requirement for generator–verifier KV-cache compatibility, reliance on simplifying assumptions such as the Linear Representation Hypothesis, and the preliminary nature of KV Steering with open questions around stability and interpretability.
   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 (or correct) proof?
   Answer: [Yes]
   Justification: All theorems (Theorems 1–4) and definitions (Definitions 1–2) are numbered and cross-referenced. Assumptions 1–2 are clearly stated before being invoked. Complete proofs for all results are provided in Appendix A.

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: Section 5.1 specifies all datasets (MATH, GSM8K, AIME 2024/2025), model families and scales (Qwen3-0.6B/4B/8B), MAS topologies, and search algorithm configurations. Appendix B provides full training hyperparameters (Table 5), LoRA configuration, verify token details, and the KV Steering algorithm (Appendix C).

   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: [No]

   Justification: Code and trained adapters will be released upon acceptance. All evaluation benchmarks (MATH, GSM8K, AIME) are publicly available. The paper provides sufficient algorithmic and hyperparameter detail to reproduce the method independently.

   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: Section 5.1 describes the datasets, model scales, MAS topologies, search algorithms, and LoRA configuration. Appendix B (Table 5) provides full training hyperparameters including LoRA rank, alpha, dropout, sequence length, batch size, and precision.
   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: Error bars are not reported due to the prohibitive computational cost and time required to evaluate massive, multi-billion-parameter models across extensive benchmarks multiple times. This is a common practice for LLM papers, and we believe it's suitable.
   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: Latency and memory benchmarks are conducted on an NVIDIA GH200 120GB GPU, as stated in Section 5.2 and Figure 3. Training uses bf16 precision (Table 5 in Appendix B).
Guidelines:
- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**
Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?
Answer: [Yes]
Justification: The research uses publicly available models and benchmarks, does not involve human subjects, and conforms with the NeurIPS Code of Ethics.
Guidelines:
- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**
Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?
Answer: [N/A]
Justification: This work proposes a more efficient scoring mechanism for process reward models. It is foundational infrastructure research that reduces compute cost and does not introduce new capabilities with direct negative societal impact.
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: The paper trains lightweight LoRA adapters for reward scoring on publicly available mathematical reasoning benchmarks. The released artifacts (adapters) do not pose risks for misuse.
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: All datasets (MATH, GSM8K, AIME) and models (Qwen3) are cited with their original papers. The Qwen3 models are used under their publicly available licenses.
Guidelines:
- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**
Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?
Answer: [N/A]
Justification: No new datasets or pre-trained models are released in this submission. The trained LoRA adapters and code will be released upon acceptance with accompanying documentation.
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 paper does not involve crowdsourcing or research with human subjects.
Guidelines:

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

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**
    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?
    Answer: [N/A]
    Justification: The paper does not involve crowdsourcing nor research with human subjects.
    Guidelines:
    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.
16. **Declaration of LLM usage**
    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.
    Answer: [N/A]
    Justification: The LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research.
    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.