MemReward: Graph-Based Experience Memory for LLM Reward Prediction with Limited Labels

Tianyang Luo[*1] **Tao Feng**[*1] **Zhigang Hua**[2] **Yan Xie**[2]
Shuang Yang[2] **Ge Liu**[1] **Jiaxuan You**[1†]
[1]University of Illinois Urbana-Champaign [2]Meta
[*]Equal contribution. [†]Correspondence to: jiaxuan@illinois.edu

ulab-uiuc/MemReward ulab-ai/memreward

Abstract

Reinforcement learning (RL) fine-tuning improves large language model (LLM) reasoning using reward signals on sampled rollouts. When reward labels are available for only a subset of rollouts, restricting policy updates to that subset leaves the remaining rollouts unused for policy optimization. Semi-supervised learning propagates labels from labeled to unlabeled samples, but existing methods either diffuse labels with fixed similarity weights or treat each sample independently, ignoring the relations among queries, thinking processes, and answers. We propose **MemReward**, a graph-based experience memory that propagates rewards to unlabeled rollouts during online policy optimization. MemReward stores queries and the rollouts (thinking processes and final answers) of an initial LLM policy as nodes of a heterogeneous graph connected by similarity and structural edges, and warms up a GNN on labeled rollouts to propagate rewards over this graph. During online RL fine-tuning, unlabeled rollouts are attached to the graph by query similarity, and the policy is updated with ground-truth rewards on labeled rollouts and GNN-predicted rewards on the rest. We evaluate MemReward on 13 benchmarks in mathematics, question answering, and code generation with verifiable rewards, and further with LLM-judge, human-preference, and continuous rewards. With ground-truth rewards on only 20% of rollouts, MemReward achieves 96.6% of the performance of fully supervised training (Oracle) on Qwen2.5-1.5B and 97.3% on Qwen2.5-3B, where it also outperforms label propagation and FixMatch-style self-training. It carries over to Llama backbones up to 8B and closely approaches Oracle on out-of-domain tasks.

1 Introduction

Recent advances in large language models (LLMs) have been driven by reinforcement-learning-based post-training, where rewards on sampled rollouts guide policy optimization [14, 46, 10, 19]. We study a limited-supervision setting in which reward labels are available for only a subset of sampled rollouts. Training solely on this labeled subset discards potentially useful unlabeled rollouts. Our goal is to exploit these unlabeled rollouts while keeping the reward-label budget fixed. A natural solution lies in semi-supervised learning, where models trained on labeled data have proven effective in propagating labels to unlabeled samples [21, 18, 38]. This suggests an analogous strategy for RL fine-tuning: treating labeled rollouts and their rewards as early experiences, linking unlabeled rollouts to these experiences, and propagating rewards from labeled to unlabeled rollouts. Therefore, we ask: *Can we leverage early experience memories to construct a shared reward model, enabling effective RL fine-tuning in label scarcity scenarios?*

Addressing this question presents several challenges. First, the relationships among experiences are complex and multi-faceted: semantically similar queries may share reward patterns, multiple

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

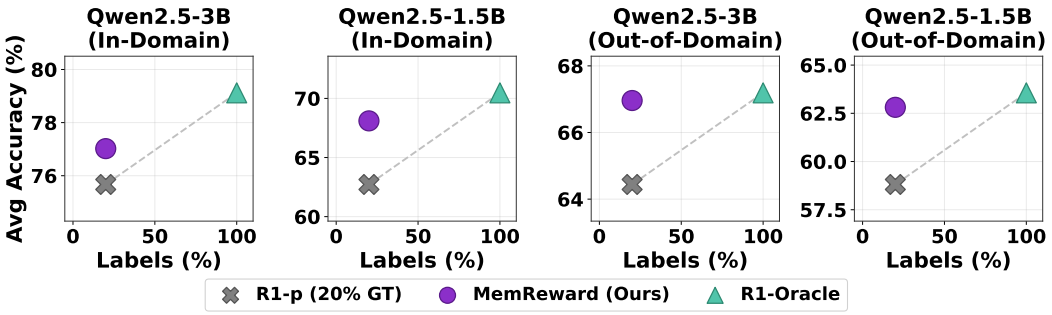

Figure 1: **MemReward approaches Oracle performance with only 20% labels.** Using the same 20% ground-truth labels, MemReward (purple) substantially outperforms partial labels (R1-p, gray), closely approaching fully-supervised Oracle performance (green) on both in-domain and out-of-domain tasks across both model scales.

rollouts for the same query compete in quality, and answer correctness depends on both the reasoning path and the query context. Modeling such intertwined dependencies calls for a relational structure such as a graph, rather than treating each experience independently [31, 17]. Second, the relevant relations are themselves heterogeneous: spanning query-query semantic similarity, query-to-reasoning association, and reasoning-to-answer correspondence—and a homogeneous graph cannot distinguish their differing semantics, motivating a heterogeneous design [42, 36].

To address these challenges, we introduce **MemReward**, a graph-based experience memory framework. We first use an initial LLM policy to generate rollouts for each labeled query, where each rollout consists of a thinking process and a final answer. Queries, thinking processes, and answers form three types of nodes, connected by three types of edges: query-query edges between semantically similar queries, query-thinking edges from each query to its rollouts, and thinking-answer edges within each rollout. A heterogeneous GNN is then trained jointly across multiple domains (mathematics, QA, and code generation) over this graph to predict rewards [12, 20]. During online RL fine-tuning, unlabeled rollouts are attached to the graph via embedding similarity, and the GNN predicts their rewards to augment the limited ground-truth signal during policy optimization.

We evaluate MemReward on Qwen2.5-1.5B and 3B across 13 benchmarks spanning 3 domains: mathematics, question answering, and code generation. As shown in Figure 1, MemReward—using ground-truth rewards on only 20% of rollouts and GNN-predicted rewards on the rest—achieves 96.6% of Oracle performance on 1.5B and 97.3% on 3B. On out-of-domain tasks, MemReward closely approaches Oracle on both model scales, showing that GNN-predicted rewards generalize effectively across domain boundaries. Performance scales smoothly with the label budget, reaching 99.4% of Oracle at 70% labels. Beyond these settings, MemReward improves over R1-p with Llama-3.2-3B and Llama-3.1-8B backbones, with LLM-judge (TruthfulQA), human-preference (Chatbot Arena), and continuous token-F1 (HotpotQA) rewards, and when the GNN is trained on one domain and used for RL in another. Across six random seeds, its gains over R1-p on both Qwen model sizes are statistically significant. Our conclusions concern label efficiency in the evaluated reward settings, while expert-graded mathematical proofs remain a direction for future work (Appendix A).

2 Preliminaries

Reinforcement Learning for Reasoning-based LLM Fine-tuning. In recent RL-based fine-tuning, an LLM policy π_θ parameterized by θ generates rollouts y for a query q [14]. Each rollout (also referred to as an *experience*) consists of a thinking process t and a final answer a. The correctness of the final answer a determines the reward, and the policy is updated through policy gradient methods to reinforce the entire thinking process t that produced a correct answer.

Group Relative Policy Optimization (GRPO). GRPO [14] is a widely adopted RL algorithm for reasoning-based LLM fine-tuning. For each query, it compares the results of multiple rollouts to obtain the advantage values. It can be formulated as follows:

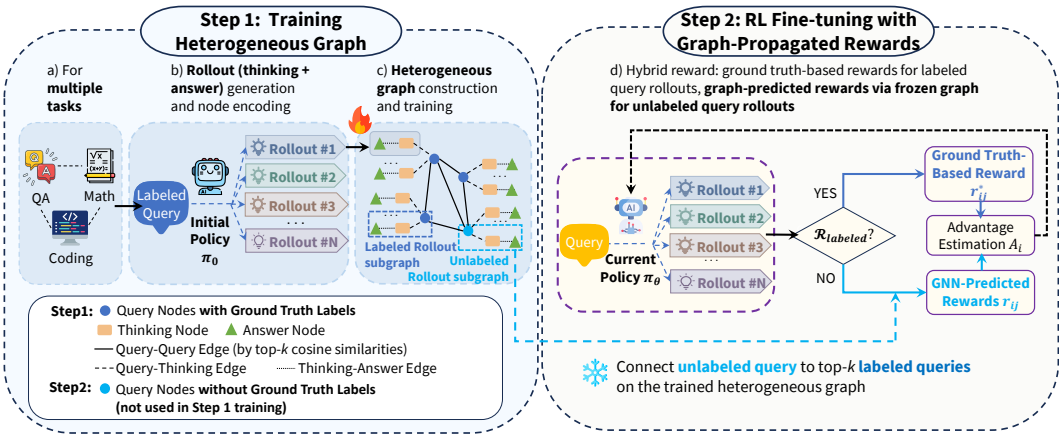

Figure 2: **Overview of MemReward.** Rollouts generated by the initial policy are stored as experience memory and organized into a heterogeneous graph for reward prediction. (Left) *Warmup Phase*: We construct a heterogeneous graph from labeled rollouts, where query nodes connect via embedding similarity, and each query links to its thinking and answer nodes. A GNN is trained to predict rewards through relational message passing. (Right) *Online Phase*: During GRPO training, labeled rollouts receive ground-truth rewards while unlabeled rollouts connect to the warmup graph via top-k similarity edges and obtain GNN-predicted rewards.

$$\mathcal{J}_{\text{GRPO}}(\pi_\theta) = \mathbb{E}\Big[\frac{1}{N}\sum_{i=1}^{N}\frac{1}{|y_i|}\sum_{j=1}^{|y_i|}\min\{\rho_{i,j}A_i, \text{clip}(\rho_{i,j}, 1-\epsilon, 1+\epsilon)A_i\} - \beta\text{KL}(\pi_\theta\|\pi_{\text{ref}})\Big] \quad (1)$$

where $q \sim \mathcal{D}$ is a query sampled from the data distribution; $y_{1\ldots N} \sim \pi_\theta(q)$ are N rollouts generated by the policy; $|y_i|$ is the token length of the rollout y_i; $\rho_{i,j} = \pi_\theta(y_{i,j}|q, y_{i,<j})/\pi_{\text{old}}(y_{i,j}|q, y_{i,<j})$ is the importance sampling ratio for the j-th token of rollout i; $A_i = \frac{R(q,y_i)-\text{mean}[R(q,y_{1\ldots N})]}{\text{std}[R(q,y_{1\ldots N})]}$ is the standardized advantage; $R(q, y_i)$ is the reward for the rollout y_i; ϵ is the clipping range; β is the KL penalty weight; and π_{ref} is the reference policy.

3 MemReward: Graph-Based Experience Memory for Reward Prediction

Overview. As illustrated in Figure 2, MemReward treats each rollout—consisting of a query, a thinking process, and an answer—as a labeled experience stored in a graph-based memory. It constructs a cross-domain heterogeneous graph from these labeled experiences and trains a GNN to predict rewards (Section 3.1). During online policy optimization, the trained GNN predicts rewards for unlabeled rollouts, which are combined with ground-truth rewards from labeled rollouts for scalable reward acquisition (Algorithm 1).

3.1 Cross-Domain Heterogeneous Graph for Reward Prediction

Initialize node/edge features. In MemReward, we have three types of nodes (query nodes, thinking nodes, and answer nodes) and three types of edges (query-query, query-thinking, and thinking-answer). Their embeddings at layer ℓ are denoted as $h_q^{(\ell)}$, $h_t^{(\ell)}$, and $h_a^{(\ell)}$, with direction-specific weight matrices W_{qq}, W_{tq}, W_{qt}, W_{at}, and W_{ta}.

For node initialization, for each query q with ground-truth labels in the training set, we use the initial policy π_0 to generate rollouts, each consisting of a thinking process t and a final answer a, and then encode the query, thinking, and answer to obtain embeddings e_q, e_t, e_a.

For edge construction, query-query edges are established using the top-k cosine similarity between query embeddings. Query-thinking edges directly connect each query node to its thinking nodes. Thinking-answer edges pair each thinking node one-to-one with the corresponding answer node. All edge weights are initialized to 1. Only query-query edges depend on embedding similarity; thinking

and answer nodes enter the graph solely through structural edges within their own rollout, so their length or template does not affect which experiences are retrieved.

The graph represents relations among experiences rather than temporal dependencies. A directed acyclic graph (DAG) variant could orient query-query edges by generation time or label confidence, but query similarity does not define a global acyclic ordering. Enforcing one would introduce an ordering heuristic and could remove reciprocal message-passing paths between related experiences. We therefore retain the heterogeneous graph without imposing acyclicity and leave DAG-based variants for future work.

Training the warmup GNN. We train a heterogeneous GNN on labeled rollouts to enable cross-query knowledge transfer, as shown in Figure 2.

For aggregating different types of nodes and edges, we employ heterogeneous aggregation with type-specific weights. The GNN derives expressive node representations h through iterative neighborhood aggregation with learnable weights. The update of the embedding of the node in the ℓ-th layer is as follows:

$$\mathbf{h}_q^{(\ell)} = \text{ReLU}\Big(\text{Mean}\big(\sum_{\mathcal{N}_q} \alpha^{qq} W_{qq}^{(\ell)} \mathbf{h}_q^{(\ell-1)}, \sum_{\mathcal{N}_t} \alpha^{tq} W_{tq}^{(\ell)} \mathbf{h}_t^{(\ell-1)}\big)\Big), \tag{2}$$

$$\mathbf{h}_t^{(\ell)} = \text{ReLU}\Big(\text{Mean}\big(\sum_{\mathcal{N}_q} \alpha^{qt} W_{qt}^{(\ell)} \mathbf{h}_q^{(\ell-1)}, \sum_{\mathcal{N}_a} \alpha^{at} W_{at}^{(\ell)} \mathbf{h}_a^{(\ell-1)}\big)\Big), \tag{3}$$

$$\mathbf{h}_a^{(\ell)} = \text{ReLU}\Big(\sum_{\mathcal{N}_t} \alpha^{ta} W_{ta}^{(\ell)} \mathbf{h}_t^{(\ell-1)}\Big), \tag{4}$$

where $\mathbf{h}^{(\ell)}$ is the embedding of the node after ℓ iterations. $\mathbf{h}_q^{(\ell)}$, $\mathbf{h}_t^{(\ell)}$, $\mathbf{h}_a^{(\ell)}$ are initialized as $\mathbf{h}_q^{(0)} = e_q$, $\mathbf{h}_t^{(0)} = e_t$, $\mathbf{h}_a^{(0)} = e_a$. α^{xy} denotes the attention weights for edge type $x \to y$, $W_{xy}^{(\ell)}$ denotes the learnable edge type-specific weights, and $\mathcal{N}_x$ denotes the neighbors of node type x.

Based on the final-layer embeddings for the query, thinking, and answer nodes, we predict the reward score for each rollout using scaled dot product scoring:

$$r_{ij} = \frac{\phi_q\big(h_{q_i}^{(L)}\big) \cdot \phi_r\big([h_{t_{ij}}^{(L)} \| h_{a_{ij}}^{(L)}]\big)}{\sqrt{d}} + b, \tag{5}$$

where r_{ij} is the predicted reward score for the j-th rollout of query i; ϕ_q and ϕ_r are learnable linear projections that map the query and rollout representations to a shared d-dimensional space, respectively; $h_{q_i}^{(L)}$ is the final-layer embedding of the i-th query; $h_{t_{ij}}^{(L)}$ and $h_{a_{ij}}^{(L)}$ are the final-layer embeddings of the thinking process and answer for the j-th rollout of query i; $\|$ denotes concatenation; and b is a learnable bias.

The GNN is trained to minimize the binary cross-entropy loss:

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{N} \sum_{i,j} \big[r_{ij}^* \log \sigma(r_{ij}) + (1 - r_{ij}^*) \log(1 - \sigma(r_{ij}))\big], \tag{6}$$

where $\sigma(\cdot)$ is the sigmoid function; $r_{ij}^* = r(a_{ij}, a_i^*) \in \{0, 1\}$ is the ground-truth reward, equal to 1 if the answer is correct and 0 otherwise; $r(\cdot, \cdot)$ is the reward function that compares an answer with the ground-truth label; a_{ij} is the j-th generated answer for the i-th query; a_i^* is its ground-truth label; and N is the total number of rollouts for each query. For continuous rewards such as token-level F1, we instead train the GNN with a regression loss to predict the reward directly.

3.2 Adaptive Reward Acquisition for Online Policy Optimization

Using the trained GNN. During online policy optimization, we employ a mixed reward acquisition strategy: labeled rollouts receive ground-truth rewards, while unlabeled rollouts leverage GNN-predicted rewards.

Since unlabeled rollouts and their associated queries are not part of the warmup graph, we connect them to the warmup graph (Section 3.1) at inference time. For each query q whose rollouts are unlabeled, we first encode it and its rollouts using the same encoder to obtain embeddings e_q, e_t,

Algorithm 1 Online GRPO with MemReward

Input: $\mathcal{R}_{\text{labeled}}$ (labeled rollouts), $\mathcal{R}_{\text{unlabeled}}$ (unlabeled rollouts), trained GNN, $\mathcal{G}_{\text{warmup}}$, LLM policy π_θ
Output: Fine-tuned policy π_{θ^*}
Prompt: Domain-specific system prompt (see Appendix E.5)

$\mathcal{R}_{\text{train}} \leftarrow \mathcal{R}_{\text{labeled}} \cup \mathcal{R}_{\text{unlabeled}}$
for each training iteration **do**
  Sample batch $\mathcal{B} \sim \mathcal{R}_{\text{train}}$
  for each query $q \in \mathcal{B}$ **do**
    Generate N rollouts: $\{(t_j, a_j)\}_{j=1}^{N} \sim \pi_\theta(\text{Prompt}, q)$
    if $(t_j, a_j) \in \mathcal{R}_{\text{labeled}}$ **then**
      $r_j \leftarrow r(a_j, a^*)$ // Labeled rollout: ground-truth reward
    else
      Encode: e_q, e_{t_j}, e_{a_j}
      Connect to $\mathcal{G}_{\text{warmup}}$ via top-k similarity
      Propagate through GNN
      $r_j \leftarrow \mathbb{I}[\sigma(\text{GNN}(q, t_j, a_j)) > 0.5]$ // Unlabeled rollout: GNN reward
    end if
  end for
  Compute advantages: $A_j = (r_j - \mu)/\text{std}$
  Update: $\theta \leftarrow \theta + \alpha \nabla_\theta \mathcal{J}_{\text{GRPO}}$
end for
Return π_{θ^*}

and e_a. We then establish query-query edges between the new query embedding and its top-k most similar warmup query embeddings by cosine similarity. The initial embeddings are then propagated through the trained GNN to obtain the final layer representations $h_q^{(L)}$, $h_t^{(L)}$, and $h_a^{(L)}$. Finally, we predict the reward of rollout j for query i as shown in Equation (5).

Integrating GNN with GRPO. During online policy optimization, for each query q_i, GRPO generates N rollouts $\{y_1, \ldots, y_N\}$ from the policy. The final reward for each rollout is computed as follows:

$$R_{ij} = \begin{cases} r_{ij}^* & \text{if } y_{ij} \in \mathcal{R}_{\text{labeled}}, \\ \mathbb{I}[\sigma(r_{ij}) > 0.5] & \text{otherwise}, \end{cases} \tag{7}$$

where R_{ij} is the reward for the j-th rollout of query i, serving as $R(\cdot)$ in Equation (1) for computing the standardized advantage; r_{ij}^* is the ground-truth reward and r_{ij} is the GNN-predicted reward score; $\mathcal{R}_{\text{labeled}}$ denotes the set of labeled rollouts (i.e., rollouts whose queries have ground-truth labels); and $\mathbb{I}[\cdot]$ is the indicator function. For continuous rewards, the predicted score is used directly in place of the indicator.

The policy is then optimized by maximizing the GRPO objective in Equation (1).

4 Experiments

4.1 Experimental Settings

Datasets. We evaluate 13 datasets in total: 10 in-domain and 3 out-of-domain, spanning three task categories (math reasoning, question answering, and code generation).

In-domain Benchmarks (10 datasets). We evaluate in three task categories: **Math**: GSM8K [8], GSM-Symbolic [29], MATH [16]; **Code**: MBPP+ [1, 26], HumanEval+ [5, 26]; **QA**: MMLU [15], CommonsenseQA [39], OpenBookQA [28], ARC-Challenge [7], GPQA [34]. Dataset statistics and detailed descriptions are given in the Appendices D and E.

Out-of-domain Benchmarks (3 datasets). To assess generalization, we evaluate three held-out datasets not seen during training: NuminaMath [23] (competition math), SIQA [35] (social reasoning) and PIQA [3] (physical common sense).

Baselines. All methods adopt GRPO [14] as the RL algorithm for policy optimization, but differ in how rewards are acquired. We compare against six baselines: (1) **R1-p (Partial)**, which trains

Table 1: **MemReward reaches 97.3% of Oracle performance on Qwen2.5-3B and 96.6% on Qwen2.5-1.5B across 10 benchmarks.** We compare reward acquisition strategies under a 20% label budget. On 3B, MemReward outperforms R1-p by +1.35, LLM-RM by +1.30, and kNN and Label Propagation by +1.05 each. Label Propagation is evaluated on Qwen2.5-3B. **Bold** and underline mark the best and second-best results among R1-p, MemReward, and R1-Oracle (a three-way tie is left unmarked). Avg is the mean over six random seeds, with the standard deviation as a subscript.

Method	Math			QA					Code		Avg	ΔScore
	GSM8K	GSM-sym	MATH	MMLU	CSQA	OBQA	ARC-C	GPQA	HumanEval+	MBPP+		
Qwen2.5-1.5B-Instruct												
R1-p	77.11	62.89	44.44	53.33	70.22	68.67	71.56	20.00	38.46	55.00	$62.72_{\pm 1.75}$	-7.75
LLM-RM	78.22	64.00	45.35	54.44	70.34	68.79	72.22	18.33	41.03	55.00	$63.29_{\pm 1.91}$	-7.18
kNN	79.33	65.11	46.49	55.33	71.91	70.62	73.74	20.00	41.03	52.50	$64.49_{\pm 1.58}$	-5.98
MemReward (Ours)	**88.67**	**77.78**	50.89	54.67	72.44	70.00	72.67	**23.33**	43.59	55.00	$68.10_{\pm 1.62}$	-2.37
R1-Oracle	86.44	75.33	**53.11**	**66.44**	**74.44**	**74.00**	**74.89**	15.00	**53.85**	**56.25**	$\mathbf{70.47}_{\pm 1.54}$	0
Qwen2.5-3B-Instruct												
R1-p	92.89	84.67	54.67	71.78	77.33	78.44	80.00	21.67	64.10	65.00	$75.67_{\pm 0.78}$	-3.45
LLM-RM	92.44	83.56	58.73	71.78	75.73	77.00	81.06	25.00	56.41	67.50	$75.72_{\pm 0.89}$	-3.40
kNN	91.78	81.56	64.40	72.22	73.48	78.82	80.56	21.67	56.41	66.25	$75.97_{\pm 0.65}$	-3.15
Label Propagation	92.67	85.78	58.00	71.78	74.22	80.00	80.22	23.33	58.97	62.50	$75.97_{\pm 0.72}$	-3.15
MemReward (Ours)	92.89	86.44	**61.11**	72.00	74.44	81.78	80.44	**30.00**	61.54	63.75	$77.02_{\pm 0.67}$	-2.10
R1-Oracle	92.89	**90.22**	60.44	**72.22**	**79.11**	**83.11**	**84.00**	**30.00**	**71.79**	**73.75**	$\mathbf{79.12}_{\pm 0.46}$	0

with only 20% labeled rollouts and discards the remaining 80%; (2) **R1-Oracle**, which trains with 100% ground-truth labels, serving as a fully supervised reference; (3) **kNN**, which predicts rewards for unlabeled rollouts by majority voting over the nearest labeled query embeddings; (4) **LLM-RM**, which fine-tunes Qwen2.5-3B as a binary reward classifier via SFT on the 20% labeled data and uses the frozen model to predict rewards for the remaining 80%; (5) **Label Propagation** [48], which uses the same query embeddings, similarity graph, labeled rollouts, and GRPO procedure as MemReward but replaces learned heterogeneous message passing with fixed label diffusion; and (6) **FixMatch-style self-training** [38], which assigns pseudo-rewards to the unlabeled rollouts on which the policy is most confident (Appendix K). In contrast, **MemReward** uses the same 20% labeled rollouts as R1-p, but augments the remaining 80% with GNN-predicted rewards.

Settings. We use Qwen2.5-1.5B-Instruct and Qwen2.5-3B-Instruct as backbone models, and additionally Llama-3.2-3B-Instruct and Llama-3.1-8B-Instruct [13] (Section 4.4). For optimization of the online policy, we adopt GRPO with hyperparameters in Table 5. The GNN training configurations are shown in Table 6 (complete implementation details are given in Appendix C). Case studies illustrating model output are provided in Appendix E.5, and compute overhead is analyzed in Appendix G.

Evaluation Metrics. We use task-specific metrics across task categories: **Exact Match** for mathematical reasoning requiring precise numerical answers; **Accuracy** for question answering involving multiple-choice or classification; and **Pass@1** for code generation, measuring functional correctness.

4.2 MemReward Approaches Oracle Performance with Only 20% Labels

Here, labeled and unlabeled queries are random splits of the same datasets, so this setting tests within-distribution reward propagation; Sections 4.3 and 4.4 evaluate held-out datasets and cross-domain transfer. Table 1 presents the performance of 10 benchmarks on two model scales. In Qwen2.5-3B, MemReward achieves a mean score of 77.02%, reaching 97.3% of Oracle performance while outperforming R1-p by 1.35 points. In Qwen2.5-1.5B, MemReward improves over R1-p by 5.38 points, reaching 96.6% of Oracle. On both scales, MemReward reduces the gap between partial labels and full supervision, showing that graph-based reward propagation provides an effective training signal regardless of model capacity (comparative case studies in Appendix E.6). Across six random seeds, MemReward's gains over R1-p are statistically significant on both model sizes, in-domain and out-of-domain (two-tailed paired t-test, $p < 0.05$; Appendix J). R1-Oracle is a fully supervised reference rather than a per-benchmark upper bound: MemReward exceeds it on several benchmarks, notably GSM8K and GSM-Symbolic for 1.5B and MATH for 3B, where GNN reward predictions are most accurate (Appendix Q).

MemReward outperforms all reward prediction baselines. On Qwen2.5-3B, MemReward outperforms LLM-RM by +1.30 and kNN by +1.05. On Qwen2.5-1.5B, the advantage widens: MemReward exceeds LLM-RM by +4.81 and kNN by +3.61, consistent with graph-based reward propagation being more valuable when the policy is weaker. At the reward prediction level, the GNN achieves 83.8% test-set accuracy, compared with 77.5% for kNN and 73.5% for MLP (Appendix H). On 3B,

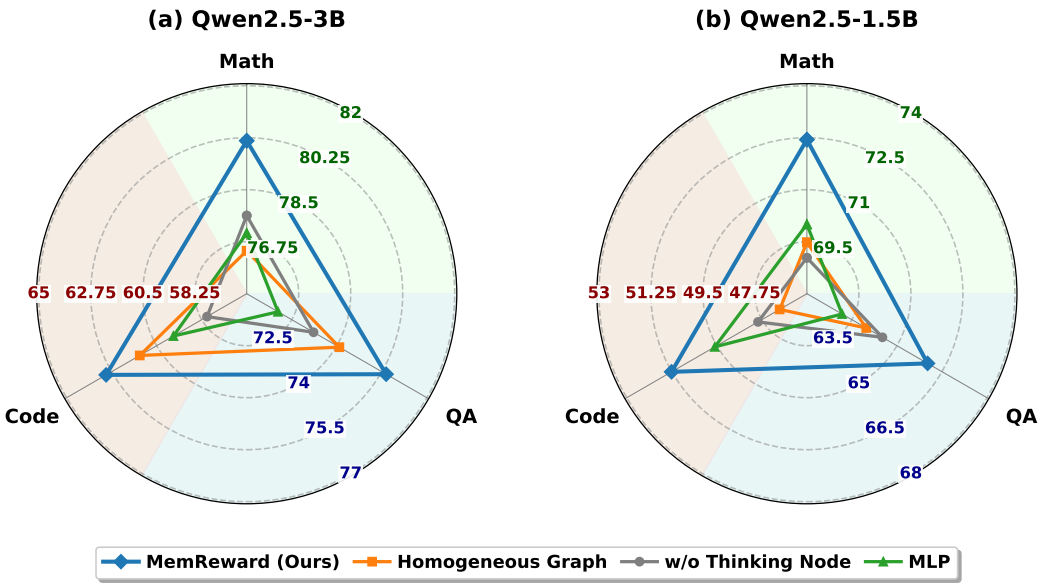

Figure 3: **Ablation studies on (a) Qwen2.5-3B and (b) Qwen2.5-1.5B show each architectural component contributes to performance.** The full model consistently outperforms all ablated variants on both scales across all three task categories.

Label Propagation reaches 75.97, only 0.30 above R1-p (75.67), and FixMatch-style self-training falls below R1-p at 72.55, whereas MemReward reaches 77.02. The gain thus comes from learned propagation over the graph rather than from adding pseudo-rewards (Appendix K).

GNN reward prediction quality. On held-out validation data, the GNN achieves 0.917 ROC-AUC across all domains, with the highest prediction quality on mathematical reasoning (ROC-AUC 0.936–0.946; see Appendix E.4 for a per-dataset breakdown and Appendix E.7 for propagation traces).

Robustness to reward-prediction noise. Flipping each GNN-predicted reward with probability p (ground-truth rewards untouched; Qwen2.5-1.5B) lowers the average score by 0.72 at $p = 30\%$ and 1.01 at $p = 50\%$. At $p = 70\%$ and 90%, where flipped predictions become anti-correlated with the target by construction, the score drops by 10.00 and 21.00 points. Separately, the frozen GNN's accuracy on Qwen2.5-3B stays between 74.9% and 77.3% over 410 GRPO steps (Appendix P).

Mathematical reasoning benefits the most from the predicted rewards of the GNN; on 1.5B, GSM8K improves by 11.56 points and GSM-Symbolic by 14.89 over R1-p; on 3B, MATH improves by 6.44. This is consistent with MemReward's query-query similarity edges: mathematical queries with similar problem structures share solution strategies, enabling effective reward propagation. Quantitatively, the top-7 nearest neighbors share the same reward label 80% of the time on average (66–80% for math, 80–86% for QA). The sentence encoder only proposes candidate neighbors; the GNN can learn to down-weight reward-inconsistent ones (Appendix E.7).

MemReward scales with label budget. Figures 4 and 5 summarize per-benchmark improvements and label ratio scaling. On Qwen2.5-1.5B, MemReward improves or matches R1-p on all 13 benchmarks, with the largest gains in mathematical reasoning (GSM-Sym +14.9, GSM8K +11.6). Performance scales smoothly with the ground-truth label ratio: increasing from 20% to 70% labels narrows the gap to Oracle from 2.10 to just 0.48 points (99.4% of Oracle), with diminishing returns at higher ratios (Appendix B).

4.3 MemReward Approaches Oracle on Out-of-Domain Tasks

GNN rewards generalize to out-of-domain tasks. Table 2 presents the results of the out-of-domain evaluation on NuminaMath, SIQA, and PIQA. In this setting, the GNN is trained exclusively on in-domain data (10 benchmarks) during the warmup phase. During online GRPO training, new

Table 2: **Out-of-domain evaluation. MemReward closely approaches R1-Oracle on out-of-domain tasks using only 20% ground-truth labels, narrowing the gap to just 0.23 on 3B and 0.75 on 1.5B.** Avg is the mean over six random seeds, with the standard deviation as a subscript.

Method	NuminaMath	SIQA	PIQA	Avg	ΔScore
Qwen2.5-1.5B-Instruct					
R1-p	31.56	72.67	72.22	$58.81_{\pm 2.20}$	-4.75
MemReward	**34.67**	74.44	79.33	$62.81_{\pm 2.05}$	-0.75
R1-Oracle	33.78	**75.78**	**81.11**	$\mathbf{63.56}_{\pm 1.94}$	0
Qwen2.5-3B-Instruct					
R1-p	36.44	74.67	**82.22**	$64.44_{\pm 1.05}$	-2.75
MemReward	42.22	76.89	81.78	$66.96_{\pm 0.86}$	-0.23
R1-Oracle	**42.44**	**77.11**	82.00	$\mathbf{67.19}_{\pm 0.72}$	0

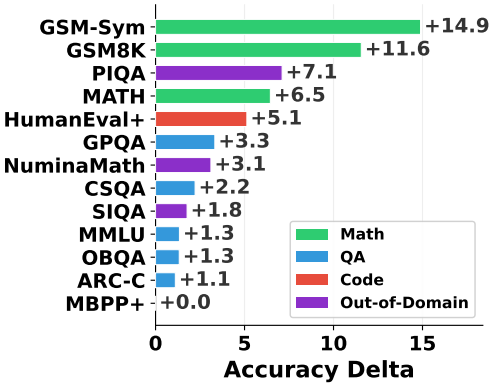

Figure 4: **MemReward improves or matches R1-p across all 13 benchmarks on Qwen2.5-1.5B,** with the largest gains on mathematical reasoning (GSM-Sym +14.9, GSM8K +11.6) and the smallest on well-saturated tasks (MBPP+ 0.0).

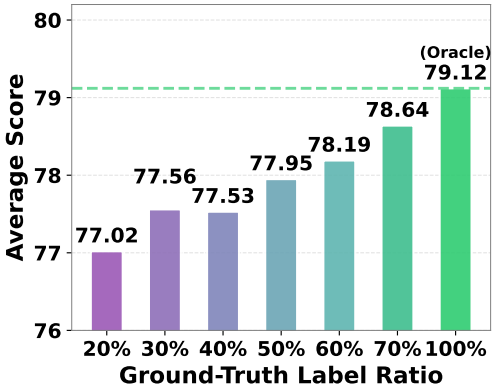

Figure 5: **MemReward performance scales with ground-truth label ratio on Qwen2.5-3B.** Each bar shows the overall average score. Even at 20% GT, MemReward reaches 97.3% of Oracle.

queries from the held-out tasks are connected to the warmup graph through top-k similarity edges, and the GNN directly predicts rewards without any domain-specific fine-tuning.

MemReward closely approaches Oracle on both model scales: on 3B, the gap narrows to just 0.23 points; on 1.5B, to 0.75 points. On NuminaMath, MemReward closely matches Oracle on both scales, trailing by only 0.22 on 3B and exceeding it by +0.89 on 1.5B, while the remaining benchmarks remain close to Oracle. This shows that the cross-domain graph structure (Section 3.1), where the GNN aggregates reward signals from semantically similar in-domain experiences through query-query edges, transfers learned reward patterns across domain boundaries. Sensitivity to the query-query connectivity parameter k is evaluated in Appendix F.

4.4 MemReward Generalizes Across Backbones, Reward Types, and Domains

Table 3 evaluates MemReward beyond the Qwen backbones and verifiable rewards, keeping the 20% label budget. With Llama-3.2-3B-Instruct and Llama-3.1-8B-Instruct, MemReward improves over R1-p by 2.64 and 3.00 points, reaching 96.1% and 97.8% of R1-Oracle. With Qwen2.5-3B-Instruct, it also improves over R1-p when rewards come from an LLM judge on TruthfulQA [25] (+2.86), from human pairwise preferences on Chatbot Arena [6] (+1.67), and from continuous token-level F1 on HotpotQA [45] (+2.68), where the GNN is trained as a regressor. Finally, a GNN trained only on QA labels and used for RL on math raises the math average from 77.36 to 80.79, and the reverse direction raises the QA average from 73.95 to 76.80 (Appendix N).

Table 3: **MemReward generalizes beyond Qwen backbones and verifiable rewards.** All settings use ground-truth rewards for only 20% of the training queries. The TruthfulQA, Chatbot Arena, and HotpotQA rows use Qwen2.5-3B-Instruct; each row reports the metric in parentheses.

Setting (metric)	Reward signal	R1-p	MemReward	R1-Oracle
Llama-3.2-3B, 10 benchmarks (Avg)	Verifiable	67.41	70.05	72.91
Llama-3.1-8B, 10 benchmarks (Avg)	Verifiable	74.62	77.62	79.39
TruthfulQA (truthful & informative)	LLM judge (binary)	36.73	39.59	44.90
Chatbot Arena (preference acc.)	Human preference	57.33	59.00	63.00
HotpotQA (token F1)	Continuous (F1)	39.42	42.10	47.17

4.5 Each Architectural Component Contributes to Reward Prediction

We ablate three key architectural components of MemReward by comparing against three variants: (1) **MLP**, which replaces the GNN with a multi-layer perceptron that predicts rewards from query embeddings alone, without graph structure; (2) **Homogeneous Graph**, which removes edge type distinctions and treats all nodes uniformly; (3) **w/o Thinking Node**, which removes the intermediate thinking nodes from the heterogeneous graph.

Figure 3 presents results per-category on both model scales, where the category-level accuracy is computed by averaging over all benchmarks within each task category. On 3B, the full model achieves 80.1% in Math, 75.6% in QA, and 63.0% in Code, consistently outperforming all ablated variants. On 1.5B, the full model achieves 72.4% in Math, 66.0% in QA, and 51.3% in Code.

Graph structure captures inter-query reward patterns. The MLP baseline, which predicts rewards from query embeddings alone without graph structure, suffers the largest drop on QA: 72.0% on 3B (a gap of 3.6 points) and 63.2% on 1.5B (a gap of 2.8 points). QA queries span diverse topics, including science, social reasoning, and physical commonsense, where individual embeddings provide limited reward signals. This shows that the edges of query-query similarity (Section 3.1) capture exploitable structural dependencies between experiences, addressing the challenge identified in Section 1.

Heterogeneous edge types benefit structured reasoning. The homogeneous graph variant, which treats all edges uniformly, retains 74.1% on QA but drops to 76.4% on Math, a gap of 3.7 points from the full model. Math problems exhibit distinct structural relationships at different levels: query-query edges capture problem similarity (e.g., two quadratic equations), while query-thinking and thinking-answer edges capture solution correctness. Collapsing these into a single edge type loses this hierarchical signal, demonstrating that the heterogeneous edge design (Section 3.1) preserves task-specific structural relationships essential for accurate reward prediction.

Thinking nodes benefit multi-step reasoning tasks. Removing thinking nodes degrades Code to 58.0% on 3B (a drop of 5.0 points) and Math to 77.6% (a drop of 2.5 points). In math and code, the thinking process encodes intermediate reasoning steps (e.g., equation manipulations, algorithmic logic) that determine whether the final answer is correct. Without thinking nodes, the GNN can only compare queries and answers, losing the fine-grained signal about *how* the model reasons. On 1.5B, the w/o Thinking variant similarly falls to 69.0% in Math (a gap of 3.4 points). The consistency of component contributions across both model scales shows that the heterogeneous three-node design (Section 3.1) captures reasoning structure that generalizes across model capacities. We also evaluate reward threshold sensitivity in Appendix I.

5 Related Work

Reinforcement Learning for LLMs. Early work established the RLHF paradigm [30], training reward models from human evaluations and optimizing via PPO [37]. Subsequent efforts reduced pipeline complexity through AI feedback [22] and direct preference optimization [32]. Process-level supervision further improved reward quality via step-by-step verification [24] and automated labeling [41]. More recently, DeepSeek-R1 [14] used GRPO with group-based advantage estimation for reasoning. These methods assume full reward supervision, whereas MemReward addresses limited labels. Recent work further addresses reward model reliability: ensemble-based uncertainty penalization mitigates overoptimization [9], and noise-corrected gradient estimation handles noisy

verifiers [27]. MemReward is complementary, addressing the orthogonal challenge of label scarcity rather than reward model calibration.

Graph Neural Networks for Language Tasks. GNNs model relational structures through iterative message passing [12], with foundational architectures including GCN [20], GAT [40], and GATv2 [4]. Heterogeneous extensions such as HAN [42] and R-GCN [36] handle multiple node and edge types. Our work extends heterogeneous GNNs to reward prediction, constructing graphs over queries, thinking processes, and answers to propagate reward labels.

Semi-supervised Learning and Label Propagation. Semi-supervised learning addresses label scarcity through graph-based label propagation [48, 18] and consistency-based methods such as pseudo-labeling [21] and FixMatch [38]. In the LLM context, Constitutional AI [2] uses AI feedback to reduce labeling. Self-training approaches iteratively expand labeled sets using model predictions [43], but risk error accumulation without structural constraints. More broadly, these approaches either rely on fixed similarity metrics or operate on individual samples without exploiting structural dependencies. MemReward bridges these paradigms, predicting rewards through learnable message passing over query-thinking-answer structures. We compare against both label propagation and FixMatch-style self-training in Section 4.2.

6 Conclusion

We proposed MemReward, a graph-based experience memory framework that addresses reward label scarcity in reinforcement learning for LLMs. By organizing queries, thinking processes, and answers into a heterogeneous graph, MemReward propagates rewards from a small set of labeled rollouts to unlabeled ones during online policy optimization. Across 13 benchmarks spanning mathematics, question answering, and code generation on two Qwen2.5 model scales, MemReward achieves near-Oracle performance with only 20% ground-truth labels and closely approaches Oracle on out-of-domain tasks; the gains carry over to Llama backbones up to 8B and to LLM-judge, human-preference, and continuous rewards. Ablation studies confirm that each architectural component contributes to prediction quality. Overall, MemReward demonstrates that graph-based experience memory is an effective and scalable approach for enabling label-efficient RL fine-tuning of LLMs. We discuss limitations and future directions in Appendix A.

Acknowledgments and Disclosure of Funding

We sincerely appreciate the research gift from Meta and the research grant from Samsung GRO that made this project possible.

References

[1] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[2] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[3] Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439, 2020.

[4] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*, 2021.

[5] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[6] Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E. Gonzalez, et al. Chatbot Arena: An open platform for evaluating LLMs by human preference. *arXiv preprint arXiv:2403.04132*, 2024.

[7] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

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

[9] Thomas Coste, Usman Anwar, Robert Kirk, and David Krueger. Reward model ensembles help mitigate overoptimization. *arXiv preprint arXiv:2310.02743*, 2023.

[10] Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

[11] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

[12] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. Pmlr, 2017.

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

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

[15] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

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

[17] Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin R Benson. Combining label propagation and simple models out-performs graph neural networks. *arXiv preprint arXiv:2010.13993*, 2020.

[18] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. Label propagation for deep semi-supervised learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5070–5079, 2019.

[19] Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

[20] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[21] Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896. Atlanta, 2013.

[22] Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, et al. Rlaif vs. rlhf: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*, 2023.

[23] Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13(9):9, 2024.

[24] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The twelfth international conference on learning representations*, 2023.

[25] Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, 2022.

[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. *Advances in neural information processing systems*, 36:21558–21572, 2023.

[27] Omar El Mansouri, Mohamed El Amine Seddik, and Salem Lahlou. Noise-corrected grpo: From noisy rewards to unbiased gradients. *arXiv preprint arXiv:2510.18924*, 2025.

[28] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2381–2391, 2018.

[29] Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. *arXiv preprint arXiv:2410.05229*, 2024.

[30] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[31] Tianyi Alex Qiu, Fanzhi Zeng, Jiaming Ji, Dong Yan, Kaile Wang, Jiayi Zhou, Yang Han, Josef Dai, Xuehai Pan, and Yaodong Yang. Reward generalization in rlhf: A topological perspective. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 15884–15930, 2025.

[32] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

[33] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 3982–3992, 2019.

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

[35] Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 4463–4473, 2019.

[36] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer, 2018.

[37] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[38] Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in neural information processing systems*, 33:596–608, 2020.

[39] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, 2019.

[40] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

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

[42] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. Heterogeneous graph attention network. In *The world wide web conference*, pages 2022–2032, 2019.

[43] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10687–10698, 2020.

[44] Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

[45] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, 2018.

[46] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

[47] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, volume 36, pages 46595–46623, 2023.

[48] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107, Carnegie Mellon University, 2002.

Contents of Appendix

A Limitations, Future Work, and Broader Impact **15**

B Ground-Truth Label Ratio Analysis **16**

C Implementation Details **16**

D Dataset Statistics **17**

E Dataset Descriptions **17**

E.1 Mathematical Reasoning 17

E.2 Question Answering 18

E.3 Code Generation 18

E.4 GNN Reward Prediction Quality 19

E.5 Case Studies 19

E.6 Comparative Case Studies: R1-p vs. MemReward 19

E.7 GNN Reward Propagation Trace 19

F Top-k Sensitivity on OOD Benchmarks **20**

G Computational Overhead **20**

H GNN Reward Prediction Accuracy on the Test Set **21**

I Reward Threshold Sensitivity **21**

J Statistical Reliability **21**

K Semi-Supervised Baselines **21**

L Llama Backbones **22**

M Additional Reward Sources **22**

N Cross-Domain Transfer **23**

O Thinking Length and Similarity **24**

P Robustness to Reward-Prediction Noise **25**

Q When MemReward Exceeds R1-Oracle **27**

A Limitations, Future Work, and Broader Impact

Limitations. We evaluate verifiable rewards on 13 benchmarks with Qwen2.5-1.5B and 3B, and on 10 benchmarks with Llama-3.2-3B and Llama-3.1-8B. Additional experiments with Qwen2.5-3B use LLM-judge (TruthfulQA), human-preference (Chatbot Arena), and continuous token-F1 (HotpotQA)

Table 4: **MemReward maintains near-Oracle performance across ground-truth label ratios on Qwen2.5-3B.** We vary the proportion of ground-truth labels from 20% to 70%, with the remaining queries receiving GNN-predicted rewards. Avg is the mean over six random seeds, with the standard deviation as a subscript.

GT Ratio	Math			QA					Code		Avg	ΔScore
	GSM8K	GSM-sym	MATH	MMLU	CSQA	OBQA	ARC-C	GPQA	HumanEval+	MBPP+		
20%	92.89	86.44	61.11	72.00	74.44	81.78	80.44	**30.00**	61.54	63.75	$77.02_{\pm 0.67}$	-2.10
30%	**93.56**	90.67	61.78	69.56	78.22	79.56	81.11	25.00	56.41	62.50	$77.56_{\pm 0.63}$	-1.56
40%	92.67	91.78	57.78	**72.44**	76.22	81.56	83.11	23.33	58.97	55.00	$77.53_{\pm 0.56}$	-1.59
50%	92.44	**92.89**	63.56	69.11	76.22	80.44	82.22	20.00	64.10	65.00	$77.95_{\pm 0.53}$	-1.17
60%	93.11	92.44	56.89	70.00	78.44	82.44	82.89	28.33	66.67	71.25	$78.19_{\pm 0.50}$	-0.93
70%	92.00	89.33	**66.22**	72.00	78.00	82.22	81.78	20.00	**71.79**	63.75	$78.64_{\pm 0.49}$	-0.48
100% (Oracle)	92.89	90.22	60.44	72.22	**79.11**	**83.11**	**84.00**	**30.00**	**71.79**	**73.75**	$\mathbf{79.12}_{\pm 0.46}$	0

rewards. Two design choices define the current scope. (1) The GNN is frozen after warmup, which avoids a moving-target feedback loop between the reward predictor and the policy, while the 20% ground-truth rewards continuously anchor optimization. Its accuracy stays stable over 410 GRPO steps, and training remains robust when up to half of the predicted rewards are randomly flipped (Appendix P). This study characterizes robustness to random prediction errors; systematic prediction bias and policy-dependent reward exploitation lie outside it. (2) Query-query edges are built with a fixed general-purpose sentence encoder (all-MiniLM-L6-v2) that is not trained for reward relevance. Its neighborhoods nonetheless carry reward-relevant signal: the top-7 nearest queries share the reward label about 80% of the time, and a GNN trained on one domain predicts rewards in another 12.95 (QA→Math) and 7.35 (Math→QA) points more accurately than a graph-free MLP (Appendix N).

Future Work. Natural next steps include extending MemReward to further forms of supervision, such as graded expert assessment of mathematical proofs and multi-level human ratings, and to larger backbones; characterizing systematic reward bias and policy-dependent reward exploitation, together with safeguards against them; learning reward-aware similarity jointly with the GNN; and exploring richer graph topologies, such as directed or hierarchical experience graphs that order experiences by generation time or label confidence. Scaling graph construction to larger rollout pools is another promising avenue.

Broader Impact. By reducing the reliance on costly human labels through graph-based reward propagation, MemReward has the potential to democratize access to RL-based LLM training for researchers with limited labeling budgets. We do not foresee specific negative societal consequences beyond those generally associated with improving LLM reasoning capabilities. As with all advances in LLM performance, downstream applications should be deployed responsibly with appropriate safeguards.

B Ground-Truth Label Ratio Analysis

Table 4 examines how MemReward performs as the proportion of ground-truth labels varies from 20% to 70% on Qwen2.5-3B, with the remaining queries receiving GNN-predicted rewards (see also Figure 5 in the main text). Even at the lowest label budget (20% GT + 80% GNN), MemReward already achieves 97.3% of Oracle performance (77.02 vs. 79.12). As the GT ratio increases to 60%, MemReward reaches 98.8% of Oracle (78.19 vs. 79.12), and at 70% the gap narrows to just 0.48 points (78.64 vs. 79.12), reaching 99.4% of Oracle.

C Implementation Details

We implement the heterogeneous GNN using PyTorch Geometric [11] with `HeteroConv` as the heterogeneous message-passing wrapper and `GATv2Conv` [4] as the per-relation convolution operator, combined with `torch_scatter` for efficient sparse aggregation. The GNN architecture consists of 2 layers with a hidden dimension of 512, 4 attention heads, and a dropout rate of 0.1. For in-domain and out-of-domain experiments, we encode queries, thinking processes, and answers using `all-MiniLM-L6-v2` [33] (384-dim). All embeddings are projected to the hidden dimension via type-specific linear transformations. Query-query edges are constructed using top-k cosine similarity with $k = 7$; the kNN baseline uses the same $k = 7$ for a fair comparison. The GNN is trained with

Table 5: **GRPO training hyperparameters.** All RL fine-tuning experiments use these settings with Qwen2.5-1.5B-Instruct, Qwen2.5-3B-Instruct, Llama-3.2-3B-Instruct, and Llama-3.1-8B-Instruct as the backbone models.

Hyperparameter	Value
Actor learning rate	$1e{-}6$
train_batch_size	128
mini_batch_size	128
micro_batch_size	32
Training steps	410
Max response length	1024
Num of rollouts	8
Rollout temp (τ)	1.0
KL penalty (β)	$1e{-}3$
Advantage clip (ϵ)	0.2

Table 6: **GNN hyperparameters.** Configuration of the heterogeneous graph neural network used for reward prediction.

Hyperparameter	Value
Input dimension	384
Hidden dimension	512
Number of GNN layers	2
Number of attention heads	4
Dropout	0.1
Learning rate	$1e{-}3$
Training epochs	150
Query-Query top-k	7

the Adam optimizer (learning rate 1×10^{-3}) using binary cross-entropy loss for 150 epochs with early stopping (patience 20).

For GRPO training, we use the veRL framework with the following hyperparameters: actor learning rate 1×10^{-6}, batch size 128, micro-batch size 32, and maximum response length 1024. We enable KL loss regularization ($\beta = 1 \times 10^{-3}$) and advantage clipping ($\epsilon = 0.2$). Response generation uses a temperature of 1.0 for diverse sampling. Each query generates $N = 8$ responses for advantage estimation. Training runs for 410 steps, with model selection based on validation performance. The Llama-3.2-3B-Instruct and Llama-3.1-8B-Instruct experiments use the same settings. All experiments are conducted on NVIDIA RTX PRO 6000 GPUs with BF16 precision.

D Dataset Statistics

Table 7 summarizes the data splits used across all in-domain benchmarks.

E Dataset Descriptions

We provide detailed descriptions of all datasets used in our evaluation, organized by task category.

E.1 Mathematical Reasoning

GSM8K [8] is a dataset of 8.5K grade school math word problems requiring multi-step arithmetic reasoning. Problems involve basic operations (addition, subtraction, multiplication, and division) and require 2-8 reasoning steps to solve. We evaluate using an exact match on the final numerical answer.

GSM-Symbolic [29] is a variant of GSM8K generated from symbolic templates that vary names and numerical values while preserving the underlying problem structure. This variant helps assess whether models truly understand mathematical reasoning or merely memorize numerical patterns.

Table 7: **Dataset statistics for in-domain evaluation.** For each dataset, we randomly sample 1,500 instances and split them into training, validation, and testing sets. R1-Oracle uses all training queries with 100% ground-truth labels; MemReward uses all training queries with 20% GT labels and 80% GNN-predicted rewards; R1-p uses only the 20% labeled rollouts and discards the rest. Validation labels are available in all settings for model selection. Note that GPQA, HumanEval+, and MBPP+ have fewer than 1,500 available instances due to their original dataset sizes.

Split	Math			QA					Code	
	GSM8K	GSM-sym	MATH	MMLU	CSQA	OBQA	ARC-C	GPQA	HumanEval+	MBPP+
R1-Oracle	750	750	750	750	750	750	750	99	65	132
MemReward	750	750	750	750	750	750	750	99	65	132
R1-p	150	150	150	150	150	150	150	19	13	26
Validation	300	300	300	300	300	300	300	39	26	52
Testing	450	450	450	450	450	450	450	60	39	80

MATH [16] contains 12.5K challenging competition mathematics problems spanning algebra, geometry, number theory, counting and probability, and precalculus. Problems are drawn from AMC, AIME, and other prestigious competitions, requiring sophisticated multi-step reasoning and domain knowledge.

NuminaMath [23] (out-of-domain) is a large-scale mathematical reasoning dataset with problems collected from various mathematical olympiads and competitions worldwide. It provides diverse problem types and difficulty levels beyond standard benchmarks, serving as a challenging test of mathematical generalization.

E.2 Question Answering

MMLU [15] (Massive Multitask Language Understanding) covers 57 subjects across STEM, humanities, social sciences, and other domains. Questions are multiple-choice and test both world knowledge and reasoning ability, ranging from elementary to professional difficulty.

CommonsenseQA [39] contains 12.2K multiple-choice questions requiring commonsense reasoning about everyday concepts and relationships. Questions are generated from the ConceptNet knowledge graph and require an understanding of implicit world knowledge not stated in the question.

OpenBookQA [28] presents 5.9K elementary science questions modeled after OpenBook exams. Each question requires combining a core science fact with additional commonsense knowledge, testing multi-hop reasoning over scientific concepts.

ARC [7] (AI2 Reasoning Challenge) contains 7.8K natural science questions from standardized tests. We use the Challenge set (ARC-C), which contains questions that simple retrieval and word co-occurrence methods fail to answer correctly, requiring genuine reasoning.

GPQA [34] (Graduate-level Google-Proof QA) is a highly challenging benchmark of 448 multiple-choice questions in biology, physics, and chemistry. Questions are designed to be "Google-proof," answerable by domain experts but not easily searchable online, testing deep domain expertise.

SIQA [35] (Social Interaction QA, out-of-domain) tests reasoning about people's actions and their social implications. Questions require understanding of emotional reactions, motivations, and social dynamics in everyday situations, assessing social commonsense reasoning beyond factual knowledge.

PIQA [3] (Physical Interaction QA, out-of-domain) evaluates physical commonsense reasoning about everyday objects and their interactions. Questions test intuitive physics knowledge, such as object affordances, material properties, and physical causality that humans acquire through embodied experience.

E.3 Code Generation

HumanEval+ [5, 26] is an extended version of the original HumanEval benchmark with additional test cases to reduce false positives. It contains 164 programming problems with function signatures and docstrings, requiring models to generate correct Python implementations that pass all test cases.

Table 8: **Per-dataset GNN reward prediction metrics on the validation set** (Qwen2.5-3B, 20% GT). The GNN achieves strong prediction quality across domains, with math achieving the highest ROC-AUC.

Dataset	Domain	Acc.	Prec.	Recall	ROC-AUC
GSM8K	Math	0.873	0.843	0.789	0.946
GSM-Sym	Math	0.883	0.725	0.669	0.936
MATH	Math	0.890	0.672	0.744	0.936
MMLU	QA	0.858	0.642	0.531	0.899
CSQA	QA	0.838	0.667	0.724	0.892
OBQA	QA	0.843	0.658	0.752	0.896
ARC-C	QA	0.868	0.756	0.663	0.913
GPQA	QA	0.923	0.826	0.861	0.843
HumanEval+	Code	0.692	0.485	0.532	0.721
MBPP+	Code	0.757	0.774	0.770	0.832
Overall		**0.861**	**0.717**	**0.704**	**0.917**

MBPP+ [1, 26] extends the Mostly Basic Python Problems benchmark with more rigorous test cases. The original MBPP dataset contains 974 crowd-sourced Python programming problems designed to be solvable by entry-level programmers, testing basic programming skills and common algorithmic patterns.

E.4 GNN Reward Prediction Quality

Table 8 presents per-dataset GNN prediction metrics on the validation set (used for model selection) for Qwen2.5-3B with 20% ground-truth labels. The GNN achieves 86.1% overall accuracy and 0.917 ROC-AUC, demonstrating that the heterogeneous graph structure (Section 3.1) learns discriminative reward patterns from limited labels. Mathematical reasoning datasets achieve the highest ROC-AUC (0.936–0.946), consistent with the strong downstream gains on math benchmarks reported in Section 4.2. QA datasets maintain robust prediction quality (ROC-AUC 0.84–0.91), while code datasets yield lower but still functional prediction quality (ROC-AUC 0.72–0.83), consistent with the smaller downstream gains on code benchmarks in Table 1. A separate comparison against MLP and kNN baselines on the held-out test set is provided in Appendix H.

E.5 Case Studies

We present representative model outputs across QA (Table 9), math (Table 10), and code (Table 11), illustrating the system prompt format and reasoning style used during GRPO training.

E.6 Comparative Case Studies: R1-p vs. MemReward

Across all 3,329 evaluation samples, we identify 246 cases where R1-p answers incorrectly but MemReward answers correctly, compared to 201 reverse cases, yielding a net advantage of 45 samples for MemReward. In 165 of the 246 cases, Oracle also answers correctly, showing that MemReward's GNN-predicted rewards guide the model toward the same solutions that full supervision produces. Tables 12 and 13 present two representative examples.

E.7 GNN Reward Propagation Trace

To illustrate how the GNN propagates reward signals through the experience graph, we trace the full inference path for two representative validation queries (Table 14 and Table 15). For each query, we show its top-7 nearest training neighbors (ranked by cosine similarity of query embeddings), the reward labels of those neighbors, and the GNN's predicted scores versus ground-truth labels for each response. These traces use the Qwen-2.5-3B model with 20% ground-truth labels.

Case 1 (Table 14) demonstrates clear reward propagation: 4 of the 7 neighbors have mostly correct responses ($\geq$7/8), and the GNN assigns uniformly high scores (0.78–0.98) to all 8 responses, matching

Table 9: **Case study in QA (Science).**

System Prompt: You are a helpful assistant skilled in mathematics, logical reasoning, and programming. Solve problems step by step, showing your work clearly. You MUST end your response with '#### ' followed by your final answer.

User Prompt: Solve this multiple-choice question:

The speed of a sound wave varies as it travels through different substances. Which factor will most affect the speed of a sound wave?
A. the frequency of the wave
B. the wavelength of the wave
C. the source that created the sound
D. the distance between molecules in the medium

Think step by step, then write your final answer as: #### [letter]

Ground Truth: D

To determine which factor most affects the speed of a sound wave, we need to understand the relationship between the speed of sound and the properties of the medium through which it travels.

Let's analyze each option:
A. the frequency of the wave - The speed of sound is independent of the frequency of the wave. Frequency affects the pitch but not its speed.
B. the wavelength of the wave - Similar to frequency, the wavelength is a characteristic of the wave itself and is not influenced by the medium's properties.
C. the source that created the sound - The source of the sound does not affect the speed once it has been generated.
D. the distance between molecules in the medium - The speed of sound is directly related to the distance between molecules. In a denser medium (molecules closer together), sound travels faster.

D

the ground truth perfectly (8/8). Because most neighbors share the correct label, the GNN produces confident, accurate predictions.

Case 2 (Table 15) demonstrates fine-grained discrimination under mixed neighbor signals: neighbor correctness rates range from 0/8 to 5/8, yet the GNN correctly separates the 5 correct responses (scores 0.58–0.77) from the 3 incorrect ones (scores 0.009–0.048), achieving 8/8 accuracy. This shows that the GNN does not simply average neighbor labels but leverages the thinking-process and answer embeddings to discriminate at the individual response level, producing well-separated scores even when neighbor-level signals are ambiguous.

Reward-inconsistent neighbors. Case 2 also shows how propagation handles semantically similar neighbors with conflicting rewards. The query's two cross-domain GSM8K neighbors have 0/8 and 4/8 correct responses, and its five same-domain neighbors have 5/8, 3/8, 3/8, 4/8, and 0/8. Neither similarity nor domain membership alone determines whether a neighbor is useful. The GNN instead combines the retrieved context with each response's thinking and answer nodes and separates the correct responses from the incorrect ones. Reward-inconsistent neighbors remain a potential source of error.

F Top-k Sensitivity on OOD Benchmarks

We evaluate the sensitivity of the query-query edge connectivity parameter k on the three out-of-domain benchmarks. All k values consistently outperform both R1-p (64.44%) and MLP (63.85%), and the default k=7 yields the best overall OOD performance.

G Computational Overhead

MemReward's offline pipeline (data loading, graph construction, and GNN training) runs in 173 seconds on a single NVIDIA RTX PRO 6000 GPU. During online GRPO training, GNN inference takes only 0.06s per step; together with encoding new queries and attaching them to the graph by query similarity, the online reward computation adds $\sim$180s over full training, i.e., 0.03% overhead, with GPU memory consumption of $\sim$0.3GB offline and $\sim$1.5GB online.

Table 10: **Case study in Math (GSM8K).**

System Prompt: You are a helpful assistant skilled in mathematics, logical reasoning, and programming. Solve problems step by step, showing your work clearly. You MUST end your response with '#### ' followed by your final answer.

User Prompt: Solve this math problem:

Leila spent $40 which is 1/4 of her money on a sweater. She was then left with $20 after spending the rest of her money on buying pieces of jewelry. How much more did Leila spend on jewelry than on the sweater?

Think step by step, showing your work clearly. Write your final answer as: #### [answer]

Ground Truth: 60

To determine how much more Leila spent on jewelry than on the sweater, we follow these steps:

1. Calculate the total amount of money Leila had initially.
   Since $40 is 1/4 of her total money:
   Total money = $40 \times 4 = 160$

2. Determine how much Leila spent on jewelry.
   After spending $40 on the sweater and being left with $20:
   Money spent on jewelry = $160 - 40 - 20 = 100$

3. Find the difference:
   Difference = $100 - 40 = 60$

60

H GNN Reward Prediction Accuracy on the Test Set

While Table 8 reports per-dataset metrics on the validation set used for model selection, Table 17 compares three reward prediction methods on the held-out test set. The GNN consistently outperforms both parametric (MLP) and non-parametric (kNN) baselines across all domains, with advantages over kNN of 5.6 points on Math, 6.0 on QA, and 4.8 on Code.

I Reward Threshold Sensitivity

We evaluate different thresholds for converting GNN sigmoid outputs to binary rewards on the standard 10 benchmarks (Qwen2.5-3B).

Threshold=0.3 introduces more false-positive rewards, causing the model to reinforce incorrect responses (-1.17 from default). Threshold=0.8 is overly conservative, discarding valid positive signals (-0.53). The default threshold of 0.5 achieves the best performance.

J Statistical Reliability

Table 19 reports six-seed statistics for the Qwen2.5 in-domain and out-of-domain comparisons among R1-p, MemReward, and R1-Oracle, and for the listed Qwen2.5-3B ground-truth-ratio conditions. The standard deviation is computed across the six per-run aggregate scores, and two-tailed paired t-tests compare the listed methods with R1-p using runs paired by seed ($df = 5$). The MemReward comparisons with R1-p reported in this table are significant at $p < 0.05$, and the separation grows as the ground-truth ratio increases. These statistical claims are limited to the conditions listed in the table.

K Semi-Supervised Baselines

Label Propagation [48] uses the same query embeddings, similarity graph, labeled examples, and mixed-reward GRPO procedure as MemReward; the only difference is that it uses fixed label diffusion instead of learned heterogeneous message passing. Its per-benchmark results on Qwen2.5-3B are included in Table 1. The FixMatch-style baseline [38] uses the same 20% labeled data and estimates

Table 11: **Case study in Coding (HumanEval+).**

System Prompt: You are an expert Python programmer. Solve coding problems step by step. You MUST end your response with '#### ' followed by your complete code in a ```python code block.

User Prompt: Task: Given an array representing a branch of a tree that has non-negative integer nodes, pluck the node with the smallest even value. If multiple nodes have the same smallest even value, return the one with the smallest index. Return [smallest_value, its index], or [] if no even values exist.

Example: pluck([5, 0, 3, 0, 4, 2]) == [0, 1]

Think through your approach, then write the Python function. REMEMBER: You MUST end with #### followed by your code in a ```python block.

Ground Truth:

```
def pluck(arr):
  if(len(arr) == 0):  return []
  evens = list(filter(lambda x:  x%2 == 0, arr))
  if(evens == []):  return []
  return [min(evens), arr.index(min(evens))]
```

We need to find the smallest even value and return it along with its index. If there are multiple nodes with the same smallest even value, we return the one with the smallest index.

1. Initialize variables to store the smallest even value and its index.
2. Iterate through the array to find the smallest even value.
3. Return the result or an empty list if no even values exist.

```
####
def pluck(arr):
  smallest_even = float('inf')
  smallest_index = -1
  for index, value in enumerate(arr):
    if value % 2 == 0 and value < smallest_even:
      smallest_even = value
      smallest_index = index
  if smallest_even == float('inf'):
    return []
  return [smallest_even, smallest_index]
```

the confidence of each unlabeled rollout by the policy's average answer-segment probability. Within each batch, only the 30% most confident unlabeled rollouts are retained and assigned pseudo-labels; the corresponding confidence threshold, $\tau = 0.058$, is selected by maximizing F1 on the validation set. Labeled rollouts continue to receive ground-truth rewards, and the baseline uses no graph structure or cross-query information.

Table 20 compares the overall scores. Confidence-filtered self-training falls 3.12 points below R1-p, so the policy's own confidence is not calibrated enough to supply reliable rewards in this setting. Fixed graph propagation adds 0.30 points over R1-p, and learned heterogeneous propagation adds 1.35.

L Llama Backbones

We replicate the full pipeline (GNN warmup, GRPO training, and evaluation on the same 10 benchmarks) with Llama-3.2-3B-Instruct and Llama-3.1-8B-Instruct [13], keeping the 20% label budget, sentence encoder, GNN architecture, data splits, and GRPO hyperparameters unchanged. Table 21 reports per-benchmark results.

M Additional Reward Sources

The three experiments in Table 3 use Qwen2.5-3B-Instruct as the policy and provide ground-truth rewards for only 20% of the training queries; the remaining rollouts receive GNN-predicted rewards.

Table 12: **Comparative case study in QA (ARC-C).** R1-p incorrectly selects D (size) while MemReward correctly identifies B (rocky terrain). The R1-p model's reasoning contains a critical error: it dismisses option B despite both the Moon and Earth having rocky surfaces.

Question: The Moon and Earth have a similar A. surface atmosphere B. rocky terrain C. gravity D. size **Ground Truth**: B
R1-p (Answer: D, Incorrect): "To determine the correct answer, let's analyze each option: A. Surface atmosphere: The Moon does not have a significant atmosphere. It has a very thin exosphere, which is much less dense than Earth's atmosphere. Earth has a thick atmosphere composed of nitrogen, oxygen, and other gases. Therefore, the Moon and Earth do not have a similar surface atmosphere. B. Rocky terrain: The Moon has a rocky surface, but it is not as diverse or as varied as Earth's terrain. Earth has a diverse range of terrains including mountains, valleys, plains, and oceans. Therefore, the Moon and Earth do not have a similar rocky terrain. C. Gravity: The Moon's gravity is about 1/6th of Earth's gravity. Earth's gravity is approximately 9.8 m/s^2. The Moon's gravity is about 1.63 m/s^2. Therefore, the Moon and Earth do not have a similar gravity. D. Size: The Moon's diameter is about 3,475 km. Earth's diameter is about 12,742 km. The Moon is significantly smaller than Earth. Therefore, the Moon and Earth do not have a similar size. Given the analysis, none of the options A, B, C, or D describe a similarity between the Moon and Earth. However, the question asks for a similarity, and the only option that could be considered a similarity is the size, but it is not a similar size. Since the question asks for a similarity and the only option that could be considered a similarity is the size, but it is not a similar size, the correct answer is D." The model dismisses the correct option B by conflating terrain diversity with terrain composition, then selects D despite the Moon being roughly 1/4 Earth's diameter.
MemReward (Answer: B, Correct): "To determine the correct answer, let's analyze each option: A. Surface atmosphere: The Moon does not have a significant atmosphere. It has a very thin exosphere, which is much less dense than Earth's atmosphere. Earth, on the other hand, has a substantial atmosphere composed of nitrogen and oxygen. Therefore, the Moon and Earth do not have a similar surface atmosphere. B. Rocky terrain: Both the Moon and Earth have rocky surfaces. The Moon's surface is covered with craters and basaltic plains, while Earth's surface is composed of various types of rock, including mountains, valleys, and plains. This similarity in terrain suggests that they do have a similar rocky terrain. C. Gravity: The Moon's gravity is about 1/6th of Earth's gravity. This means that an object would weigh about 1/6th of its weight on Earth. Earth's gravity is approximately 9.8 m/s^2, while the Moon's gravity is about 1.625 m/s^2. Therefore, the Moon and Earth do not have similar gravity. D. Size: The Moon and Earth are quite different in size. The Moon is about 1/400th the size of Earth. Earth has a diameter of approximately 12,742 kilometers, while the Moon has a diameter of about 3,475 kilometers. Thus, the Moon and Earth do not have similar sizes. Based on the analysis, the correct answer is B. rocky terrain." The model correctly identifies that both bodies share rocky surface composition.

TruthfulQA (LLM-judge rewards). Responses to TruthfulQA [25] are scored by DeepSeek-V4-Flash [44], used as a fixed judge, on whether they are both truthful and informative. The data are split by query into training, validation, and test sets with a 5:2:3 ratio. The same judge and criterion provide the training rewards for the labeled queries and evaluate the policy on the held-out test set. **Chatbot Arena (human-preference rewards).** We sample 1,500 Chatbot Arena conversations [6, 47] after excluding ties and formulate each as a two-choice question: "Which response is better, A or B?" The data are split by query into 750/300/450 training/validation/test examples. The reward is whether the policy's choice agrees with the human preference, and we report preference accuracy on the 450 test queries.

HotpotQA (continuous rewards). We use HotpotQA [45] in the distractor setting with a 5:2:3 split. The reward is the token-level F1 between the predicted span and the gold answer, a continuous value in $[0, 1]$. The GNN is trained as a regression model to predict token-level F1, and its checkpoint is selected by validation token-level F1.

N Cross-Domain Transfer

To test transfer without target-domain labels, we train the GNN only on the QA datasets (MMLU, CSQA, OBQA, ARC-C, and GPQA) and evaluate it on the unseen math datasets (GSM8K, MATH, and GSM-Symbolic), and vice versa. An MLP with the same input features and training procedure but without graph structure serves as the comparison. Table 22 shows that the GNN predicts rewards 12.95 points more accurately than the MLP for QA→Math and 7.35 points for Math→QA. Table 23 then uses the cross-domain GNN to provide rewards for GRPO in the other domain; although the GNN never observes reward labels from that domain, its rewards improve the policy in both directions. Table 15 shows a cross-domain neighborhood: an MMLU query retrieves two GSM8K problems among its top-7 neighbors, with cosine similarities of 0.581 and 0.570.

Table 13: **Comparative case study in Math (GSM8K).** R1-p computes the base pay as $2 \times 2 = 4$ instead of $30 \times 2 = 60$, arriving at an incorrect answer of 22. MemReward correctly computes both the base pay ($60) and the bonus ($18) for a total of $78.

Question: Maurice is getting $2 for every finished task. And for every 10 tasks finished, he receives a $6 bonus. How much money would Maurice make for finishing 30 tasks?
Ground Truth: 78

R1-p (Answer: 22, Incorrect): "To determine how much money Maurice would make for finishing 30 tasks, we need to break down the problem into two parts: the base pay for the tasks and the bonus for completing multiples of 10 tasks. 1. Calculate the base pay: Maurice earns $2 for each task. Base pay $= 2$ tasks $\times$ 2 dollars/task $= 4$ dollars. 2. Calculate the bonus: Maurice receives a $6 bonus for every 10 tasks completed. Number of bonuses $= 30$ tasks / 10 tasks/bonus $=$ 3 bonuses. Total bonus $= 3$ bonuses $\times$ 6 dollars/bonus $= 18$ dollars. 3. Calculate the total earnings: Total earnings $=$ Base pay $+$ Total bonus $= 4$ dollars $+ 18$ dollars $= 22$ dollars. Therefore, Maurice would make $\boxed{22}$ dollars for finishing 30 tasks." The model uses "2 tasks" instead of "30 tasks" for the base pay calculation, producing an incorrect result.

MemReward (Answer: 78, Correct): "To determine how much money Maurice would make for finishing 30 tasks, we need to break down the problem into two parts: the money he earns from the tasks and the bonus he receives. First, let's calculate the money Maurice earns from the tasks. Maurice earns $2 for each task. If he finishes 30 tasks, the total money he earns from the tasks is: 30 tasks $\times$ 2 dollars/task $= 60$ dollars. Next, we need to calculate the bonus Maurice receives. Maurice receives a $6 bonus for every 10 tasks finished. To find out how many bonuses he gets for 30 tasks, we divide 30 by 10: $30 \div 10 = 3$. So, Maurice receives 3 bonuses. Since each bonus is $6, the total bonus he receives is: 3 bonuses $\times$ 6 dollars/bonus $= 18$ dollars. Now, we add the money earned from the tasks and the bonus to find the total amount of money Maurice makes: 60 dollars $+ 18$ dollars $= 78$ dollars. Therefore, the total amount of money Maurice would make for finishing 30 tasks is $\boxed{78}$." The model correctly computes both components of the payment.

Table 14: **GNN trace – Math (Type A: clear propagation).** Query from MATH dataset; all 8 responses are correct. Neighbors are predominantly correct, and the GNN assigns uniformly high scores.

Query: Expand the product $(2x + 3)(x + 5)$. **Ground-truth answer**: $2x^2 + 13x + 15$

Top-7 Training Neighbors

Rank	Dataset	Cosine Sim	Correct
1	math	0.817	8/8
2	math	0.720	6/8
3	math	0.519	8/8
4	math	0.518	7/8
5	math	0.495	0/8
6	math	0.465	5/8
7	math	0.456	8/8

GNN Predicted Scores vs. Ground Truth

Resp	Score	GT	Pred	Match
0	0.971	1	1	✓
1	0.976	1	1	✓
2	0.974	1	1	✓
3	0.956	1	1	✓
4	0.973	1	1	✓
5	0.776	1	1	✓
6	0.975	1	1	✓
7	0.971	1	1	✓

O Thinking Length and Similarity

Thinking texts are much longer than queries and answers, so one may ask whether their embeddings are dominated by length or by a shared reasoning template. In MemReward, only query embeddings determine graph connectivity; thinking and answer nodes are attached through structural edges within each rollout (Section 3.1). We also measured the effect directly with the deployed all-MiniLM-L6-v2 embeddings and the same top-7 neighborhoods. The correlation between thinking length and top-7 thinking similarity is -0.287 across the population and is negative in 7 of 9 benchmark-level

Table 15: **GNN trace – QA (Type B: mixed-signal discrimination).** Query from MMLU dataset; 5 of 8 responses are correct. Despite mixed neighbor labels (0/8 to 5/8), the GNN correctly separates correct from incorrect responses.

Query: Erin made 12 pints of juice. She drinks 3 cups of juice each day. How many days will Erin take to drink all of the juice she made? **Ground-truth answer**: C

Top-7 Training Neighbors

Rank	Dataset	Cosine Sim	Correct
1	mmlu	0.607	5/8
2	mmlu	0.585	3/8
3	gsm8k	0.581	0/8
4	mmlu	0.571	3/8
5	gsm8k	0.570	4/8
6	mmlu	0.568	4/8
7	mmlu	0.563	0/8

GNN Predicted Scores vs. Ground Truth

Resp	Score	GT	Pred	Match
0	0.676	1	1	✓
1	0.578	1	1	✓
2	0.733	1	1	✓
3	0.771	1	1	✓
4	0.048	0	0	✓
5	0.716	1	1	✓
6	0.019	0	0	✓
7	0.009	0	0	✓

Table 16: **Top-k sensitivity on OOD benchmarks** (Qwen2.5-3B). All k values outperform R1-p and MLP.

Top-k	**NuminaMath**	**SIQA**	**PIQA**	**Avg**
MLP (k=0)	37.78	73.56	80.22	63.85
k=3	40.00	75.33	80.44	65.26
k=7 (default)	**42.22**	**76.89**	**81.78**	**66.96**
k=10	41.33	76.44	80.89	66.22

measurements. Thinking similarity also does not track query similarity: of two MATH queries, the one with higher top-7 thinking similarity (0.6641 vs. 0.4702) has lower top-7 query similarity (0.4577 vs. 0.6530). The w/o Thinking and homogeneous-graph ablations (Section 4.5) show that modeling thinking as a separate node type improves reward prediction.

P Robustness to Reward-Prediction Noise

We randomly flip each GNN-predicted reward with probability $p \in \{0, 30, 50, 70, 90\}\%$; the 20% ground-truth rewards are never corrupted. We use Qwen2.5-1.5B-Instruct and otherwise keep the GRPO settings and 10-benchmark evaluation of the main experiments. The $p = 0$ condition already

Table 17: **GNN reward prediction accuracy on the test set** (Qwen2.5-3B, 20% GT). MemReward consistently outperforms both parametric (MLP) and non-parametric (kNN) baselines across all domains.

Method	**Overall**	**Math**	**QA**	**Code**
MLP	73.5%	77.0%	72.4%	64.8%
kNN (k=7)	77.5%	81.5%	76.2%	68.3%
MemReward (GNN)	**83.8%**	**87.1%**	**82.2%**	**73.1%**

Table 18: **Reward threshold sensitivity** (Qwen2.5-3B, 10 benchmarks).

Threshold	GSM8K	GSM-Sym	MATH	MMLU	CSQA	OBQA	ARC-C	GPQA	HE+	MBPP+	Avg
0.3	92.44	85.11	58.50	70.89	73.93	80.64	79.80	28.33	58.97	62.50	75.85
0.5 (default)	**92.89**	**86.44**	**61.11**	**72.00**	**74.44**	**81.78**	**80.44**	**30.00**	**61.54**	**63.75**	**77.02**
0.8	92.67	85.78	59.86	71.33	74.61	81.09	80.30	28.33	61.54	63.75	76.49

Table 19: **Standard deviation over six random seeds and paired t-tests against R1-p.** The standard deviation is computed on the aggregate score of each run.

Evaluation	Model	Method	Std ($n = 6$)	t vs. R1-p
In-domain (10 benchmarks)	Qwen2.5-3B	R1-p	0.78	–
		MemReward	0.67	3.32 ($p < 0.05$)
		R1-Oracle	0.46	9.33 ($p < 0.001$)
	Qwen2.5-1.5B	R1-p	1.75	–
		MemReward	1.62	5.48 ($p < 0.005$)
		R1-Oracle	1.54	8.12 ($p < 0.001$)
Out-of-domain (3 benchmarks)	Qwen2.5-3B	R1-p	1.05	–
		MemReward	0.86	4.57 ($p < 0.01$)
		R1-Oracle	0.72	5.52 ($p < 0.005$)
	Qwen2.5-1.5B	R1-p	2.20	–
		MemReward	2.05	4.82 ($p < 0.005$)
		R1-Oracle	1.94	5.38 ($p < 0.005$)
GT ratio (in-domain)	Qwen2.5-3B	MemReward, 20% GT	0.67	3.32
		MemReward, 30% GT	0.63	4.09
		MemReward, 40% GT	0.56	4.75
		MemReward, 50% GT	0.53	5.66
		MemReward, 60% GT	0.50	6.67
		MemReward, 70% GT	0.49	8.01

Table 20: **Semi-supervised baselines** (Qwen2.5-3B, 10 benchmarks, 20% ground-truth labels).

Method	Overall
FixMatch-style self-training	72.55
R1-p	75.67
Label Propagation	75.97
MemReward	**77.02**

Table 21: **Results with Llama backbones** (10 benchmarks, 20% ground-truth labels).

Method	GSM8K	GSM-sym	MATH	MMLU	CSQA	OBQA	ARC-C	GPQA	HumanEval+	MBPP+	Overall
Llama-3.2-3B-Instruct											
R1-p	87.78	69.33	48.89	59.33	66.44	76.44	74.67	16.67	58.97	47.50	67.41
MemReward	89.11	70.44	54.00	61.11	69.78	80.67	75.11	28.33	61.54	50.00	70.05
R1-Oracle	91.56	79.33	55.11	64.44	70.89	80.67	77.56	21.67	64.10	63.75	72.91
Llama-3.1-8B-Instruct											
R1-p	90.89	83.78	54.00	70.22	72.00	81.78	81.56	20.00	48.72	61.25	74.62
MemReward	91.33	87.56	58.89	73.33	74.00	85.78	83.56	26.67	58.97	62.50	77.62
R1-Oracle	94.00	91.33	59.56	73.56	76.44	87.11	85.11	21.67	69.23	63.75	79.39

Table 22: **Cross-domain reward prediction accuracy** (%). The GNN is trained on one domain and evaluated on the other.

Trained on QA, evaluated on Math			**Trained on Math, evaluated on QA**		
Dataset	MLP	GNN	Dataset	MLP	GNN
GSM8K	69.62	82.29	MMLU	72.33	80.92
MATH	74.12	87.12	CSQA	75.38	82.37
GSM-Sym	70.58	83.75	OBQA	75.17	82.04
			ARC-C	77.42	84.25
			GPQA	76.97	84.94
Overall	71.44	84.39	Overall	75.13	82.48

Table 23: **GRPO with cross-domain GNN rewards.** The GNN is trained on one domain; GRPO is trained and evaluated on the other.

GNN trained on Math; GRPO trained and evaluated on QA						
Method	MMLU	CSQA	OBQA	ARC-C	GPQA	**Overall**
R1-p	69.56	75.96	75.63	82.07	18.33	73.95
MemReward	73.78	77.53	81.09	81.06	30.00	76.80
GNN trained on QA; GRPO trained and evaluated on Math						
Method	GSM8K	MATH	GSM-Sym			**Mean**
R1-p	89.78	59.41	82.89			77.36
MemReward	94.22	63.27	84.89			80.79

includes the GNN's natural prediction errors. As Table 24 shows, performance degrades by about one point up to $p = 50\%$. Above 50%, flipped binary predictions become anti-correlated with the target by construction, so these points are stress tests rather than a general robustness threshold. We attribute this robustness to the uncorrupted ground-truth rewards that anchor policy optimization with reliable supervision, together with GRPO's normalized, group-relative advantages over eight rollouts per query. The latter reduce the influence of isolated reward mistakes when predictions remain positively informative at the group level. This study covers random corruption; systematic prediction bias is discussed in Appendix A.

Table 24: **Downstream performance under injected reward noise** (Qwen2.5-1.5B, 10 benchmarks). Noise is applied only to GNN-predicted rewards.

Flip probability p	**Overall**	**Change from** $p = 0$
0%	68.10	–
30%	67.38	−0.72
50%	67.09	−1.01
70%	58.10	−10.00
90%	47.10	−21.00

GNN prediction remains stable throughout training. Table 25 logs the frozen GNN's prediction accuracy across GRPO training stages. The accuracy remains stable between 74.9% and 77.3% over 410 steps.

Q When MemReward Exceeds R1-Oracle

R1-Oracle uses ground-truth rewards for all rollouts, but it is trained with a finite, stochastic, multi-domain GRPO objective, so full supervision does not guarantee the best result on every benchmark. In Table 1, MemReward exceeds R1-Oracle most notably on GSM8K and GSM-Symbolic for 1.5B and on MATH for 3B. For the math benchmarks, the explanation lies in domain-dependent reward quality: with Qwen2.5-1.5B and 20% labels, the frozen GNN reaches a ROC-AUC of 0.918 on math, 0.871 on QA, and 0.754 on code. Because math rollouts receive the most reliable predicted rewards, MemReward places relatively more effective emphasis on math and less on the noisier QA and code gradients. To test this explanation, we raise the ground-truth ratio for QA and code from 20% to 70% while keeping math at 20% (Table 26). QA and code improve, while math drops by 2.01 points and no longer exceeds R1-Oracle, which is consistent with a multi-task trade-off rather than predicted rewards being better than ground truth.

Table 25: **GNN prediction accuracy across GRPO training stages** (Qwen2.5-3B). The frozen GNN remains reliable throughout 410 training steps.

Training Stage	Steps	Overall Accuracy
Early	1–136	74.9%
Middle	137–273	76.1%
Ending	274–410	77.3%

Table 26: **Domain-dependent reward quality and the math advantage** (Qwen2.5-1.5B).

Domain	GNN ROC-AUC	MemReward (20% all)	MemReward (QA/Code 70%)	R1-Oracle	Change
Math	0.918	72.45	70.44	71.63	−2.01
QA	0.871	66.02	68.12	70.59	+2.10
Code	0.754	51.26	52.10	55.47	+0.84

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 state the evaluated settings (verifiable rewards on 13 benchmarks; LLM-judge, human-preference, and continuous rewards; Qwen2.5 and Llama backbones from 1.5B to 8B), which match Sections 4.2–4.4 and the appendices.

   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: Appendix A discusses the scope of the evaluation, the frozen GNN, and the fixed sentence encoder.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [N/A]

Justification: This is an empirical paper; no theoretical results or proofs are presented.

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: Appendix C provides complete hyperparameters for both GRPO training and GNN training. The GNN architecture is fully described in Section 3.1, and all datasets are publicly available with statistics in Appendix D.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: Code, data splits, and trained checkpoints are released at the GitHub and Hugging Face links on the first page.

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 4 describes the evaluation protocol; Appendix C provides all training hyperparameters (Tables 5 and 6); Appendix D provides data splits for every dataset.

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: For R1-p, MemReward, and R1-Oracle, Tables 1 and 2 report the average score over six random seeds with its standard deviation. Table 4 reports the same statistics for the ground-truth-ratio conditions. Appendix J reports two-tailed paired t-tests against R1-p for the conditions listed there.

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: Appendix C specifies NVIDIA RTX PRO 6000 GPUs with BF16 precision. Appendix G reports GNN training time (173s) and online inference overhead (0.03%).

   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 conforms to the NeurIPS Code of Ethics. No human subjects are involved and no harmful applications are pursued. Appendix A discusses broader impacts.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: Appendix A discusses the positive impact of reducing annotation cost for RL-based LLM training and notes that downstream deployments should incorporate appropriate safeguards.

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 proposes a training framework rather than releasing a new pretrained model or scraped dataset. The fine-tuned models are standard Qwen2.5 variants on public benchmarks.

    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 and models are properly cited. All benchmarks used are publicly available under permissive licenses. TruthfulQA, Chatbot Arena, and HotpotQA are cited at first use and used under their respective licenses and terms of use, and the Llama-3 models are used under the Llama 3.1 and 3.2 Community 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, `https://paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: The released code and data are documented in the GitHub and Hugging Face repositories linked on the first page.

    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: No crowdsourcing or human subjects research is involved.

    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: No human subjects research is involved.

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: [Yes]

    Justification: LLMs are the policies trained by MemReward (Section 4.1), and the TruthfulQA experiment uses DeepSeek-V4-Flash as a judge (Appendix M). LLMs were otherwise used only for editing the writing.

    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.