
Reward Inside the Model: A Lightweight Hidden-State Reward Model for LLM’s Best-of-N sampling

Jizhou Guo¹ Zhaomin Wu² Philip S. Yu³

Abstract

High-quality reward models are crucial for unlocking the mathematical reasoning potential of large language models (LLMs), with best-of-N sampling demonstrating significant performance gains. While efficiency is crucial for mathematical discovery, current reward models, which typically operate on the textual output of LLMs, are computationally expensive and parameter-heavy. We introduce the Efficient Linear Hidden State Reward (ELHSR) model - a novel, highly parameter-efficient approach that leverages the rich information embedded in LLM hidden states to address these issues. **ELHSR systematically outperforms baselines with less than 0.005% of the parameters** of baselines, requiring only a few samples for training. ELHSR also achieves **orders-of-magnitude efficiency improvement** with significantly less time and fewer FLOPs per sample than baselines. Moreover, ELHSR exhibits robust performance even when trained only on logits, extending its applicability to some closed-source LLMs. In addition, ELHSR can also be combined with traditional reward models to achieve additional performance gains.

1. Introduction

The recent “reasoning era” (Chen et al., 2025a) in large language models (LLMs) has witnessed a remarkable surge in their ability to tackle complex mathematical reasoning tasks. Models such as GPT-o3-mini (OpenAI, 2024), DeepSeek-R1 (Guo et al., 2025), Grok 3 (xAI, 2025), Claude 3.7 Sonnet (Anthropic, 2025) and Gemini 2.5 (Google, 2025) have

demonstrated impressive capabilities in mathematical problem solving. One of the key breakthroughs in mathematical reasoning has been the application of best-of-N sampling under reward model guidance (Snell et al., 2025; Brown et al., 2024). This works particularly well for mathematical domains due to the verifiability of mathematical solutions and the frequency of near-miss errors (Welleck et al., 2021; Polu & Sutskever, 2020).

Efficiency is especially crucial for interactive mathematical discovery, automated theorem proving, and large-scale mathematical dataset generation (Halbach et al., 2010; Chen et al., 2025b; Azerbayev et al., 2023; Sun et al., 2024b). However, current methods incur significant overhead, as training and deploying large reward models consume substantial computational resources and require large amount of data during both the reward model training phase and the subsequent inference stage (Guo et al., 2025; Namgoong et al., 2024; Zhang et al., 2024). This exclusive reliance on text-based information constrains these models, necessitating large training sets and complex architectures to adequately capture the full spectrum of LLM errors.

However, during the reasoning process, the LLM possesses an inherent ‘inner knowledge’ reflecting its current certainty. Existing research demonstrates that the LLM’s internal states during reasoning contain high levels of information about answer confidence, often captured through *linear* representations (Zhang et al., 2025; Feucht et al., 2024; Xie et al., 2024). Our preliminary experiment in Section 3.1 also supports this observation by showing that hidden states can encode a *linear* representation for mathematical reasoning’s correctness with high accuracy.

While prior work has utilized hidden states to improve LLM’s reasoning performance, they often focus on the final answer token (Burns et al., 2023; Azaria & Mitchell, 2023; Xie et al., 2024) or on specific answer tokens within each reasoning step (Zhang et al., 2025). This approach has limitations: relying on a single token’s hidden state may not capture the full context of the reasoning process, and defining clear reasoning steps and identifying their corresponding answer tokens can be subjective and challenging. In addition, they do not provide direct reward signals, and their performance gains are limited compared to large-scale

¹Zhiyuan College, Shanghai Jiao Tong University ²National University of Singapore ³University of Illinois at Chicago. Correspondence to: Jizhou Guo <sjtu18640985163@sjtu.edu.cn>, Zhaomin Wu <zhaomin@nus.edu.sg>, Philip S. Yu <psyu@uic.edu>.

The second AI for MATH Workshop at the 42nd International Conference on Machine Learning, Vancouver, Canada. Copyright 2025 by the author(s).

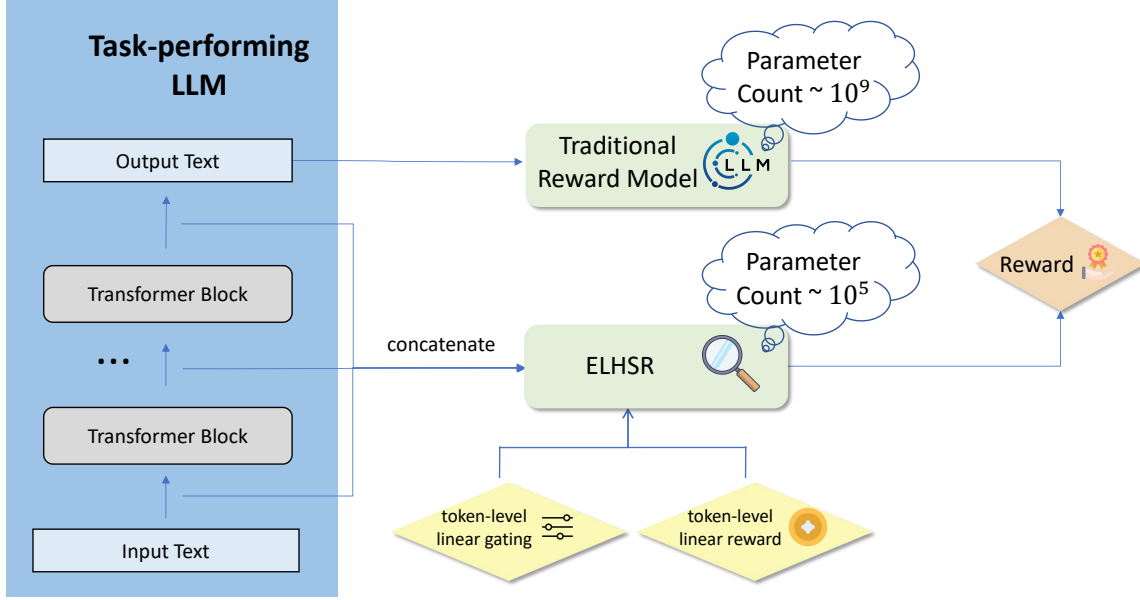


Figure 1: An illustration comparing traditional reward model and ELHSR.

reward models. Therefore, we propose the Efficient Linear Hidden State Reward (ELHSR) model (illustrated in Figure 1). ELHSR learns a linear gating and a linear reward projection per token. The final reward is computed as a weighted average of these token-level rewards, scaled by the gating values.

Our experiments demonstrate that ELHSR **systematically outperforms baseline reward models** on mathematical problem solving tasks while using **fewer than 0.005% their parameters**. Moreover, ELHSR model is data-efficient and can be trained using only a few instances. Critically, ELHSR achieves **orders-of-magnitude improvements of efficiency** in average time and FLOPs per sample compared to baselines, enabling efficient verification for mathematical problem solving that was previously computationally prohibitive.

In summary, this work makes three main contributions:

- We introduce ELHSR, a novel token-level reward model based on LLM hidden states. ELHSR achieves competitive mathematical reasoning performance with orders-of-magnitude less parameters and computational cost.
- ELHSR demonstrates exceptional data efficiency and scalability, performing well with limited data, demonstrating performance upgrade with training-time scaling and test-time scaling, generalizing to logit-only training for certain closed-source LLMs, and can be combined with conventional reward models to yield further performance improvements.

- We analyze ELHSR’s mechanisms by providing concrete examples, illustrating the way that it scores the critical components of reasoning paths.

2. Problem Statement

A common approach to enhance LLM reasoning is best-of-N sampling (Snell et al., 2025). Let \mathcal{V} be the token vocabulary and let D be a distribution over problem-answer pairs $(x, y) \in \mathcal{V}^n \times \mathcal{Y}$, where y is the hidden reference solution for question x . Conditioned on x , the base LLM π draws N reasoning paths independently (i.i.d. draws from $\pi(\cdot | x)$), producing $r_i = (r_{i,1}, \dots, r_{i,m_i}) \in \mathcal{V}^{m_i}$, $i = 1, \dots, N$, where m_i is the random length of the i -th path. We write $\mathcal{V}^* = \bigcup_{\ell=0}^{\infty} \mathcal{V}^{\ell}$. Simultaneously, π emits hidden states $h_i = (h_{i,1}, \dots, h_{i,m_i}) \in (\mathbb{R}^{L \times d})^{m_i}$, where L is the number of transformer layers, d is the per-layer hidden-state dimension, and each $h_{i,j} \in \mathbb{R}^{L \times d}$ is the stacked hidden state after emitting $r_{i,j}$. Define the correctness indicator $F : \mathcal{V}^* \times \mathcal{Y} \rightarrow \{0, 1\}$, s.t. $F(r, y) = 1 \iff r$ yields the reference answer y . We write $(\mathbb{R}^{L \times d})^* = \bigcup_{k=0}^{\infty} (\mathbb{R}^{L \times d})^k$. A parametric reward model $R_{\theta} : (\mathbb{R}^{L \times d})^* \rightarrow \mathbb{R}$ assigns a scalar score to each h_i (hence for each r_i). Note that neither π nor R_{θ} sees y . At test time we select $i^* = \arg \max_{1 \leq i \leq N} R_{\theta}(h_i)$ and return r_{i^*} . Our goal is to choose θ to maximize the probability that r_{i^*} is correct (expectation of $F(r_{i^*}, y)$) under D :

$$\max_{\theta} \mathbb{E}_{(x,y) \sim D} \mathbb{E}_{(r_1, \dots, r_N) \sim \pi(\cdot | x)} [F(r_{\arg \max_i R_{\theta}(h_i)}, y)]. \quad (1)$$

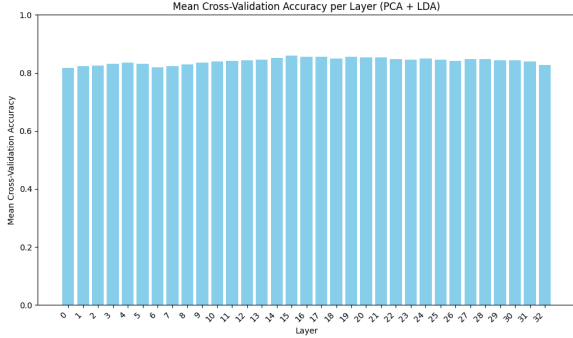


Figure 2: Cross-validation accuracy of the PCA+LDA pipeline for predicting correctness at each layer of Llama-3.1-8B. The results demonstrate that **hidden states contain information about reasoning correctness with linear representation**.

3. Methods

3.1. Motivation

We first present a preliminary experiment demonstrating the potential for enhancing mathematical reasoning using hidden states. Recent work has shown that LLM hidden states contain information about the correctness probability of a given reasoning path, often exhibiting *linear* representations (Xie et al., 2024; Zhang et al., 2025; Feucht et al., 2024). To verify this, we designed a simple pipeline to predict the correctness of reasoning steps based on the LLM’s hidden states.

Our pipeline extracts hidden states h_t^ℓ for each token t at each layer ℓ from the Llama-3.1-8B-Instruct model (Grattafiori et al., 2024) during reasoning on the MATH dataset (Hendrycks et al., 2021), averages the hidden states across all tokens within each reasoning path to obtain a single vector representation \bar{h}^ℓ for each layer ℓ , then reduces \bar{h}^ℓ for each ℓ to 50 dimensions using Principal Component Analysis (PCA) (Wold et al., 1987), and classifies the reasoning as correct or incorrect using Linear Discriminant Analysis (LDA) (Balakrishnama & Ganapathiraju, 1998). The pipeline is linear to verify that the hidden states can encode a *linear* representation for confidence of the answer. For more details, please refer to Appendix C.2.

We performed 5-fold cross-validation on 3000 instances of the MATH dataset. For each layer ℓ , we trained the pipeline on the training folds and evaluated the classification accuracy on the held-out test fold. Figure 2 presents the average cross-validation accuracy for each layer.

As shown in Figure 2, the accuracy is approximately 80% on each layer, indicating a strong signal for reasoning correctness within the hidden states. This result verifies the aforementioned theory and motivates the development of a novel, lightweight, *linear* reward model that directly lever-

ages this information, as described in the following sections.

3.2. Our Approach - ELHSR Model

Therefore, building upon the observation that LLM internal representations exhibit linear properties, we designed Efficient Linear Hidden State Reward (ELHSR) to be a parameter-efficient, linear model.

Our approach, ELHSR, learns a token-level reward function directly from the concatenated hidden states. ELHSR is designed to be highly parameter-efficient.

For each token t in a reasoning path, we concatenate and flatten the hidden states (after residual connection) from all L layers: $h_t = [h_t^1; h_t^2; \dots; h_t^L] \in \mathbb{R}^{Ld}$, where d is the dimension of the hidden state. We then apply a linear transformation to obtain a gating value g_t and a token-level reward r_t :

$$\begin{pmatrix} \tilde{g}_t \\ r_t \end{pmatrix} = W_{ELHSR} h_t + b_{ELHSR} \quad (2)$$

where $W_{ELHSR} \in \mathbb{R}^{2 \times Ld}$ is the weight matrix, $b_{ELHSR} \in \mathbb{R}^2$ is the bias vector, \tilde{g}_t is the pre-activation gating value, and r_t is the token-level reward. The gating value is then passed through a sigmoid function (Han & Moraga, 1995):

$$g_t = \sigma(\tilde{g}_t) = \frac{1}{1 + e^{-\tilde{g}_t}} \quad (3)$$

The final reward R for the entire reasoning path is the weighted average of the token-level rewards, using the gating values as weights:

$$R = \frac{\sum_{t=1}^T (g_t \cdot r_t)}{\max\left(\sum_{t=1}^T g_t, \epsilon\right)} \quad (4)$$

where T is the number of tokens, and ϵ is a small constant (e.g., 10^{-8}) for numerical stability. This model only contains $O(L \times d)$ parameters, which is significantly fewer than those in traditional reward models.

Since different tokens contribute differently to the overall correctness, the gating values allow the model to assign higher weights to more important tokens. In addition, we conduct ablation experiment to show that the gating mechanism can improve performance in Appendix E.2.

We use the binary cross-entropy with logits loss (Rafailov et al., 2024) for training. Let $y \in \{0, 1\}$ denote the correctness label of a reasoning path, where $y = 1$ indicates a correct path and $y = 0$ indicates an incorrect path. For a path with hidden state representation h , the model predicts a

scalar logit score $R(h) \in \mathbb{R}$. The binary cross-entropy with logits loss is:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^N \left[y_i \cdot \log(\sigma(R(h_i))) + (1 - y_i) \cdot \log(1 - \sigma(R(h_i))) \right] \quad (5)$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function.

Although binary cross-entropy with logits loss generally yields good results, our experiments detailed in Appendix E.1 demonstrate that DPO (Rafailov et al., 2023) and hinge loss (Rosasco et al., 2004) can also perform well, sometimes even surpassing cross-entropy. However, given that cross-entropy performs best on average, we adopt it as our primary loss function in the main text. A comparison study among several loss functions is provided in Appendix E.1

4. Experiments

4.1. Experimental Setup

We evaluate ELHSR on three mathematical problem solving datasets: MATH (Hendrycks et al., 2021), GSM8K (Cobbe et al., 2021), and AQuA_RAT (Ling et al., 2017). The task-performing models are Llama-3.2-3B-Instruct (Meta, 2024), Llama-3.1-8B-Instruct (Grattafiori et al., 2024), and Ministral-8B-Instruct (Mistral AI, 2024). The generation temperature is 1.0 and `top_p` is 0.9. We use 6000 instances (each with 8 reasoning paths) for training and 500 instances for evaluation. When training ELHSR, we use early stopping (Yao et al., 2007) to mitigate overfitting. For more details on generation process, training and evaluation, please refer to Appendix C.3, C.4 and C.5.

We compare our method against four recent open-source reward models: EurusRM-7B (Yuan et al., 2025), Skywork-Reward-Llama-3.1-8B-v0.2 (Liu et al., 2024), Starling-7B (Zhu et al., 2024) and UltraRM-13B (Cui et al., 2024). These baselines are very parameter heavy and trained on extensive datasets, while achieving leading performance among open-source reward models. For links and brief descriptions of baselines, please refer to Appendix A. For parameter count and training data size of baselines compared against ELHSR, see Table 2 for details.

To provide a more comprehensive comparison, we also evaluated a fine-tuned version of EurusRM-7B, detailed in Appendix C.6, specifically fine-tuned on the relevant datasets and model-generated solutions used in our experiments. This is despite EurusRM-7B already being pre-trained on MATH and GSM8K, which gives it a potential advantage.

4.2. Accuracy

Table 1 reports the Best-of-N performance of each reward model on the MATH, GSM8K and AQuA_RAT datasets. Our ELHSR model consistently **outperforms all baseline reward models** across most BoN settings, demonstrating the effectiveness of the proposed linear architecture and hinge-loss training objective. A notable observation is that while many baseline reward models exhibit strong performance on specific subsets of the data, ELHSR demonstrates consistent and robust performance across the entire dataset. While the fine-tuned EurusRM-7B delivers respectable results, it still lags behind ELHSR in accuracy and incurs substantially higher computational costs during both training and inference. In contrast, ELHSR not only achieves the best performance but does so with a parameter footprint less than 0.005% that of the baselines and is trained on only 6,000 samples per dataset (see Section 4.3 for detailed comparison). This underscores the exceptional efficiency of our approach.

It is noteworthy that the results presented here were achieved **without extensive hyperparameter tuning**. Further performance gains could likely be realized through more refined optimization strategies. For example, exploring alternative loss functions, as demonstrated in Appendix Table 6, increasing the training dataset size based on the scalability analysis in Section 4.4, or training ELHSR on a subset of layers as explored in Section 4.5 (Table 3), could yield further improvements. This suggests that ELHSR has significant potential for further optimization and improvement.

Additionally, we further assessed ELHSR’s performance on code comprehension tasks using the Imbue Code Comprehension dataset (Imbue Team, 2024) to broaden our evaluation scope. The results, detailed in Appendix D (Appendix Table 5), demonstrate that ELHSR consistently outperforms the baselines on this task as well.

4.3. Efficiency Analysis

We demonstrate ELHSR’s significant parameter efficiency and data efficiency in Table 2. Moreover, we investigate the computational efficiency of ELHSR by comparing the average time and FLOPs required to assign a reward to each sample. Figure 3 presents a comparative analysis of ELHSR and baseline reward models across various datasets and task-performing models. The results reveal that ELHSR achieves orders-of-magnitude improvements in both time and FLOPs compared to the baselines. It is noteworthy that as shown in Section 4.4, even a much smaller training dataset leads to satisfactory performance. This substantial reduction in computational cost, combined with its parameter efficiency and data efficiency, makes ELHSR a highly practical solution for reward modeling in both resource-constrained environments and large-scale applications.

Table 1: Different methods’ best-of-N sampling performance on MATH, GSM8K, and AQuA_RAT datasets. Results represent accuracy (percentages omitted). Our method outperforms baselines. **Bold**: best performance; Underline: second-best performance; @k: indicates the number of reasoning paths used in best-of-N sampling.

| MATH Dataset | | | | | | | | | | |
|---------------------|-----------------------------|-------------|-------------|-----------------------------|-------------|-------------|-----------------------------|-------------|-------------|-------------|
| Reward Model | Llama-3.2-3B BoN@1: 39.0 | | | Llama-3.1-8B BoN@1: 47.2 | | | Ministral-8B BoN@1: 51.0 | | | Avg. |
| | @4 | @16 | @64 | @4 | @16 | @64 | @4 | @16 | @64 | |
| Eurus-7B | 42.6 | 48.2 | 46.8 | 50.8 | 52.0 | 52.2 | 54.6 | 56.8 | 55.0 | 51.0 |
| Skywork-Llama3.1-8B | 43.8 | <u>48.4</u> | <u>48.8</u> | 52.2 | 52.4 | <u>53.4</u> | <u>56.8</u> | <u>59.0</u> | <u>61.6</u> | <u>52.9</u> |
| Starling-7B | 39.6 | 41.2 | 39.8 | 50.4 | 49.0 | 49.0 | 53.8 | 50.2 | 47.0 | 46.7 |
| Ultra-13B | <u>44.6</u> | 47.4 | 44.4 | <u>53.0</u> | 50.6 | 50.4 | 53.6 | 53.0 | 54.0 | 50.1 |
| Fine-tuned Eurus-7B | 42.8 | 44.6 | 45.8 | 52.0 | <u>52.8</u> | 50.4 | 54.2 | 57.4 | 57.0 | 50.8 |
| ELHSR (ours) | 49.8 | 54.6 | 53.6 | 55.4 | 59.4 | 62.6 | 57.8 | 61.6 | 62.8 | 57.5 |
| GSM8K Dataset | | | | | | | | | | |
| Reward Model | Llama-3.2-3B BoN@1: 53.0 | | | Llama-3.1-8B BoN@1: 80.4 | | | Ministral-8B BoN@1: 79.6 | | | Avg. |
| | @4 | @16 | @64 | @4 | @16 | @64 | @4 | @16 | @64 | |
| Eurus-7B | 69.8 | 76.2 | 82.4 | 87.0 | 89.0 | 90.4 | 87.0 | 90.8 | 90.4 | 84.8 |
| Skywork-Llama3.1-8B | 77.2 | 81.8 | 85.2 | 89.6 | 89.4 | 89.2 | 83.0 | 85.8 | 87.4 | 85.4 |
| Starling-7B | 62.4 | 65.6 | 71.4 | 83.0 | 87.2 | 85.4 | 78.4 | 81.2 | 80.0 | 77.2 |
| Ultra-13B | 72.6 | 78.6 | 82.4 | 88.0 | 88.6 | 86.8 | 82.6 | 84.6 | 82.0 | 82.9 |
| Fine-tuned Eurus-7B | <u>79.0</u> | <u>84.4</u> | <u>86.6</u> | <u>90.0</u> | 90.6 | 90.6 | <u>88.2</u> | <u>91.6</u> | <u>92.4</u> | <u>88.2</u> |
| ELHSR (ours) | 80.0 | 86.4 | 87.6 | 90.6 | <u>89.6</u> | 90.6 | 89.8 | 93.2 | 93.4 | 89.0 |
| AQuA_RAT Dataset | | | | | | | | | | |
| Reward Model | Llama-3.2-3B BoN@1: 42.0 | | | Llama-3.1-8B BoN@1: 53.6 | | | Ministral-8B BoN@1: 56.6 | | | Avg. |
| | @4 | @16 | @64 | @4 | @16 | @64 | @4 | @16 | @64 | |
| Eurus-7B | 44.6 | 42.8 | 44.6 | 58.6 | 60.6 | 56.8 | 48.4 | 34.6 | 27.2 | 46.5 |
| Skywork-Llama3.1-8B | <u>60.4</u> | <u>68.0</u> | <u>69.4</u> | <u>69.0</u> | 77.0 | 79.2 | 65.6 | 71.2 | <u>72.0</u> | <u>70.2</u> |
| Starling-7B | 51.4 | 55.0 | 56.6 | 64.2 | 70.8 | 69.8 | 56.4 | 48.6 | 42.8 | 57.3 |
| Ultra-13B | 58.4 | 61.6 | 64.8 | 67.4 | 72.8 | 71.8 | 58.4 | 51.4 | 48.0 | 61.6 |
| Fine-tuned Eurus-7B | 57.6 | 63.0 | 57.8 | 67.8 | 72.6 | 70.2 | <u>70.8</u> | <u>71.6</u> | 70.8 | 66.9 |
| ELHSR (ours) | 61.0 | 70.4 | 70.8 | 69.6 | 77.0 | <u>77.0</u> | 75.2 | 75.8 | 78.0 | 72.8 |

4.4. Scalability with Training Set Size and Reasoning Paths

Scaling large language models (LLMs) can be achieved through various strategies, broadly categorized as training-time scaling and test-time scaling (Snell et al., 2025; Kaplan et al., 2020; Liu et al., 2025a;b). In this section, we investigate the scalability of ELHSR with respect to both training set size and the number of reasoning paths used during inference.

From Figure 4, our results demonstrate that ELHSR’s accu-

racy consistently improves with both the number of training samples and the size of the reasoning paths set. Specifically, under different BoN@k settings, we find that ELHSR’s accuracy increases monotonically with training set size. Likewise, for any given training set size, ELHSR’s accuracy also improves as the number of reasoning paths used in the BoN sampling process increases.

It is interesting to note that, as illustrated in Figure 4, the performance of ELHSR with training-time scaling has not appeared to saturate even when there are 6000 training samples, which is the one used in the experiments of Section

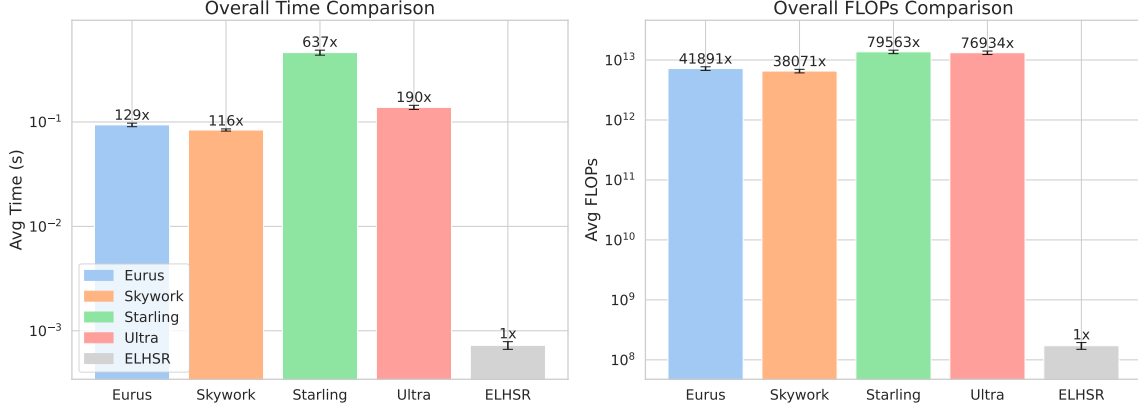


Figure 3: Comparison of average time and FLOPs per sample for ELHSR and baselines, averaged across different datasets and task-performing models with 1-sigma error bars using standard error of the mean. The y-axis is plotted in log-scale. The results show that ELHSR achieves orders-of-magnitude higher efficiency than the baselines. See Appendix Figure 6 for full details.

Table 2: Comparison of Reward Model Parameters and Training Data Size (number of questions \times number of answers). Our method is remarkably efficient in terms of parameters and data.

| Reward Model | Parameters | Training data size |
|----------------------------|----------------------|----------------------------|
| Eurus-7B | 7.1×10^9 | $5.8 \times 10^5 \times 2$ |
| Skywork-Llama3.1-8B | 7.5×10^9 | $8.0 \times 10^4 \times 2$ |
| Starling-7B | 6.7×10^9 | $1.8 \times 10^5 \times 7$ |
| Ultra-13B | 1.3×10^{10} | $7.5 \times 10^5 \times 2$ |
| ELHSR (on Llama-3.2-3B) | 1.8×10^5 | $6.0 \times 10^3 \times 8$ |
| ELHSR (on Llama-3.1-8B) | 2.7×10^5 | $6.0 \times 10^3 \times 8$ |
| ELHSR (on Ministral-8B) | 3.0×10^5 | $6.0 \times 10^3 \times 8$ |

4.2. This suggests that further increasing the training set size may lead to even greater performance improvements, highlighting the scalability of ELHSR. This indicates that ELHSR is a compelling way to scale reward modeling and hence is a practical solution for real-world applications.

4.5. Maintaining Performance with Selective Layer Utilization for Higher Efficiency

While ELHSR leverages hidden states from all layers of the LLM, we investigated the potential for further parameter reduction by selectively utilizing hidden states from a subset of layers. Specifically, for the Llama-3.1-8B model, which comprises 32 layers (plus an embedding layer), we explored training ELHSR using only the hidden states from layers 16, 24, 28, and 32, following the setting in Azaria & Mitchell (2023). This configuration reduces the parameter count to

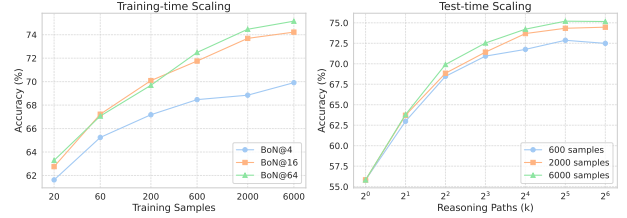


Figure 4: ELHSR has positive scaling with increased number of training samples and with the number of reasoning paths for inference. The result is averaged across datasets and task-performing models. See Appendix Figure 7 for full results.

approximately 4/33 of the full-layer ELHSR model. The results, presented in Table 3, demonstrate that performance remains comparable to the full-layer ELHSR, with certain settings even showing improved accuracy. Notably, on the GSM8K dataset, the BoN@16 and BoN@64 accuracies surpass those of the full-layer ELHSR. As shown in Table 1, the performance of the layer-selected ELHSR also **consistently exceeds that of baselines** with significantly larger parameter counts, further underscoring the efficiency and effectiveness of our approach.

4.6. Logit-Based Training for Closed-Source LLMs

While ELHSR offers significant advantages in terms of efficiency and performance, making it well-suited for deployment by large model providers and end-users, a limitation arises when dealing with closed-source LLMs where hidden states are inaccessible. To overcome this constraint, we leverage the fact that some closed-source models still provide access to their logits, such as GPT-3.5-turbo (OpenAI, 2023a) and GPT-4 (Achiam et al., 2023), as illustrated

Table 3: ELHSR’s high Best-of-N sampling performance is maintained even when trained only on layers 16, 24, 28, and 32 for Llama-3.1-8B, leading to a substantial reduction in parameters and improved efficiency.

| Method | MATH BoN@1: 47.2 | | | GSM8K BoN@1: 80.4 | | | AQuA_RAT BoN@1: 53.6 | | | Avg. |
|-------------------------|---------------------|------|------|----------------------|------|------|-------------------------|------|------|------|
| | @4 | @16 | @64 | @4 | @16 | @64 | @4 | @16 | @64 | |
| ELHSR on partial layers | 55.8 | 59.0 | 60.0 | 89.4 | 90.8 | 92.0 | 69.6 | 76.8 | 77.4 | 74.5 |
| ELHSR on all layers | 55.4 | 59.4 | 62.6 | 90.6 | 89.6 | 90.6 | 69.6 | 77.0 | 77.0 | 74.6 |

Table 4: Training ELHSR solely on logits can also yield high performance and can outperform many baselines with orders-of-magnitude higher efficiency. For results on GSM8K and AQuA_RAT dataset, please refer to Appendix Table 9.

| MATH Dataset | | | | | | | | | | |
|-----------------|-----------------------------|------|------|-----------------------------|------|------|-----------------------------|------|------|------|
| Method | Llama-3.2-3B BoN@1: 39.0 | | | Llama-3.1-8B BoN@1: 47.2 | | | Ministral-8B BoN@1: 51.0 | | | Avg. |
| | @4 | @16 | @64 | @4 | @16 | @64 | @4 | @16 | @64 | |
| ELHSR on logits | 44.6 | 49.6 | 50.6 | 54.2 | 54.4 | 55.8 | 53.4 | 55.6 | 57.4 | 52.8 |

in Finlayson et al. (2024) and OpenAI (2023b). We train ELHSR directly on these logits, resulting in a model with a parameter count that is orders of magnitude smaller than traditional reward models - and even smaller than ELHSR trained on all layers. This allows for effortless deployment on personal devices such as smartphones and personal computers. As demonstrated in Table 4 and Appendix Table 9, the resulting logit-based ELHSR retains strong performance. Furthermore, as shown in Table 1, its performance even surpasses that of many baselines. These results highlight the versatility and adaptability of ELHSR, making it a viable solution even when access to hidden states is restricted.

4.7. Combining ELHSR with External Reward Models

ELHSR offers a highly parameter-efficient approach to reward modeling, enabling its seamless integration with existing external reward models. Given ELHSR’s minimal parameter footprint (less than 0.005% of baseline models), the additional computational overhead introduced by incorporating ELHSR is negligible compared to relying solely on the external reward model. We explore two straightforward methods for combining ELHSR with external rewards: rank selection and scaled averaging. In rank selection, we choose the sample with the lowest (i.e. best) rank as determined by either ELHSR or the external reward model. In scaled averaging, we average the rewards from both models after normalizing them to the $[0, 1]$ range. Detailed descriptions of these methods are provided in Appendix B.

We empirically evaluated the effectiveness of these combination strategies by integrating ELHSR with various traditional reward models across different datasets and task-

performing models, assessing performance using best-of-N sampling. Figure 5 presents the average results across different datasets for both ranking and scaling methods, applied to various task-performing models. Detailed results for each dataset and task-performing model can be found in Appendix Figure 8. Our findings indicate that combining ELHSR with external reward models can lead to further improvements in accuracy, particularly on the GSM8K and AQuA-RAT datasets. The MATH dataset did not exhibit significant performance gains, potentially because ELHSR already significantly outperforms the external reward models on this benchmark. This performance boost is likely due to ELHSR and external reward models capturing different types of errors, resulting in improved overall accuracy - particularly with a higher number of best-of-N reasoning paths. This combination strategy offers a pathway to achieve further performance gains without incurring significant computational overhead.

4.8. Analysis on ELHSR

To investigate the mechanism of ELHSR, we present illustrative examples of its scoring of reasoning paths in Appendix F. The examples with the highest and lowest gating \times reward values are highlighted, along with their corresponding numerical scores. Notably, ELHSR assigns relatively high or low gating \times reward values to specific numerical and mathematical symbols, as well as particular tokens such as “answer”, “boxed” and the special token ‘<|eot_id|>’. This behavior suggests that ELHSR possesses the ability to identify and weigh key components within the reasoning process.

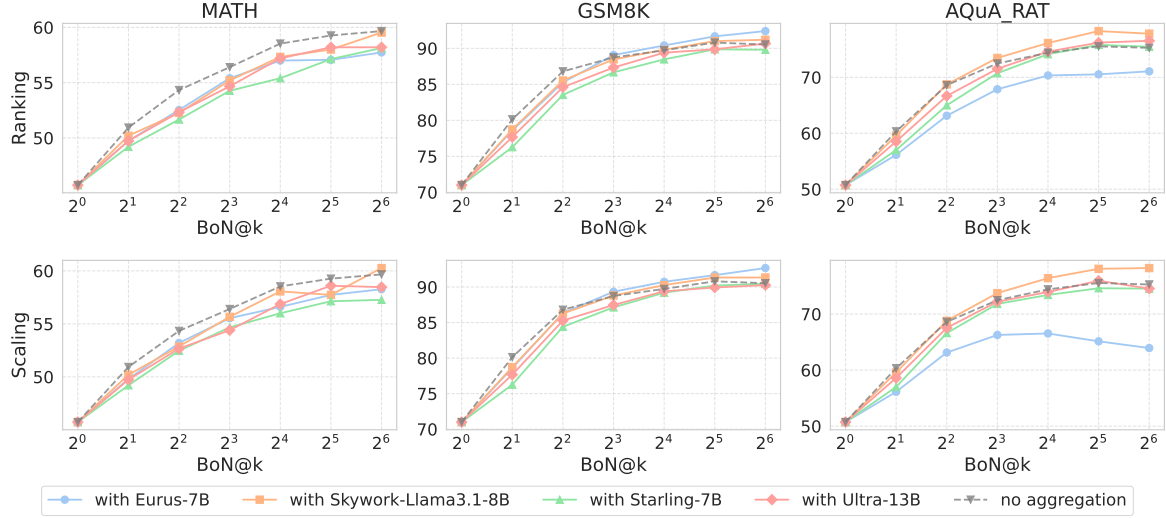


Figure 5: Average performance of combined ELHSR and external reward models across different datasets and task-performing models, for both rank selection and scaled averaging. See Appendix Figure 8 for full details.

5. Related Work

Reward Modeling for Language Models’ Reasoning

The rapid advancement of Large Language Models (LLMs) in complex reasoning tasks (Wei et al., 2022; Chen et al., 2025a; Hao et al., 2023; Kumar et al., 2025; Xia et al., 2025; Hao et al., 2024a) has highlighted the pivotal role of reward modeling in both training and post-training inference (Cui et al., 2024; Cao et al., 2024; Lambert et al., 2025; Sun et al., 2024a; Scheurer et al., 2023; Ouyang et al., 2022). However, reward modeling faces several challenges. These include high computational costs (Guo et al., 2025; Namgoong et al., 2024; Zhang et al., 2024) and potentially suboptimal performance (Wen et al., 2025; Gao et al., 2023; Coste et al., 2024). Although prior work has proposed various approaches to mitigate these issues (Chan et al., 2024; Fisch et al., 2024; Meng et al., 2024), achieving a satisfactory balance between efficiency and performance remains a challenge. This work introduces a novel perspective by leveraging internal representations to achieve a significant improvement in efficiency with satisfactory performance.

Understanding and Leveraging the Internal Representations of Language Models

The rapid development of LLMs has spurred considerable research into their internal representations (Ferrando et al., 2024; Li et al., 2023a; Gur-Arieh et al., 2025; Pan et al., 2024; Gupta et al., 2024; Nanda et al., 2023). Hidden states of LLMs have been effectively utilized in diverse fields, including safety alignment (Zou et al., 2023; Barez et al., 2025; Zhou et al., 2024; Jiang et al., 2025; Wichers et al., 2024), faithfulness and hallucination detection (Burns et al., 2023; Li et al., 2023b; Qi et al., 2024; Ferrando et al., 2025), knowledge editing (Meng et al., 2022; Yu et al., 2024; Fang et al., 2025; Gu et al., 2024), knowl-

edge distillation or transfer (Dasgupta & Cohn, 2025; Wu et al., 2024) and reasoning within the latent space (Wang et al., 2025; Hao et al., 2024b). By contrast, we effectively leverage the hidden states in intermediate layers of LLMs as a direct signal for reward modeling in reasoning tasks.

6. Conclusion

In this paper, we presented the Efficient Linear Hidden State Reward (ELHSR) model, a novel reward modeling approach for large language models (LLMs) that leverages the rich information in the LLM’s internal hidden states with significant efficiency and efficacy, and can be scaled up efficiently in the training and inference process. We also demonstrated that comparable performance can be achieved by training ELHSR on a subset of layers or even on logits alone, further lowering computational costs and making it more versatile. The success of employing ELHSR in combination with standard reward models further illustrates its versatility and potential.

The linear nature of ELHSR opens up exciting avenues for future research. Its simplicity may contribute to improved interpretability of LLMs by providing insights into their reasoning process. Furthermore, the ELHSR framework could potentially be extended to multi-modal LLMs, enabling the development of efficient reward models for a broader range of applications. Additionally, future work could explore the use of ELHSR to directly guide the reinforcement learning training of LLMs to enhance its mathematical problem-solving skills. We believe that ELHSR represents a significant step towards more efficient and effective reward modeling for LLMs in the mathematical domain, paving the way for future advancements in the field.

Impact Statement

The development of ELHSR presents both opportunities and challenges for society. On the positive side, its efficiency could democratize access to high-performing LLMs, enabling smaller labs, startups, and academic groups to iterate more rapidly on large language model (LLM) applications without requiring massive compute budgets. In addition, its remarkable efficiency can also promote more sustainable AI practices through reduced computational demands, cutting energy consumption and carbon footprint across the machine-learning lifecycle. However, the efficiency of ELHSR might also lower the bar for malicious use, possibly towards the development of harmful applications like disinformation attacks.

To mitigate these harms, we recommend proactive measures in the form of robust security measures, transparent reporting and auditing, clearly defined ethical guidelines for responsible use, and increased investments in AI safety and alignment research. By critically analyzing and taking into account these broader social implications, we can strive to ensure that ELHSR and other similar technologies are deployed to bring maximum benefit with minimal potential harm.

Acknowledgment

We gratefully acknowledge Mrs. Jiahe Jin from Zhiyuan College, Shanghai Jiao Tong University, for her invaluable assistance in creating the initial version of our illustrations. We also thank Mr. Zhihui Xie from The University of Hong Kong for offering precious suggestions.

References

- Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- Aminabadi, R. Y., Rajbhandari, S., Awan, A. A., Li, C., Li, D., Zheng, E., Ruwase, O., Smith, S., Zhang, M., Rasley, J., et al. Deepspeed-inference: enabling efficient inference of transformer models at unprecedented scale. In *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–15. IEEE, 2022.
- Anthropic. Claude 3.7 sonnet and claude code. <https://www.anthropic.com/news/claude-3-7-sonnet>, 2025.
- Azaria, A. and Mitchell, T. The internal state of an LLM knows when it’s lying. In Bouamor, H., Pino, J., and Bali, K. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 967–976, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.68. URL <https://aclanthology.org/2023.findings-emnlp.68/>.
- Azerbayev, Z., Schoelkopf, H., Paster, K., Santos, M. D., McAleer, S., Jiang, A. Q., Deng, J., Biderman, S., and Welleck, S. Llemma: An open language model for mathematics. *arXiv preprint arXiv:2310.10631*, 2023.
- Balakrishnama, S. and Ganapathiraju, A. Linear discriminant analysis-a brief tutorial. *Institute for Signal and information Processing*, 18(1998):1–8, 1998.
- Barez, F., Fu, T., Prabhu, A., Casper, S., Sanyal, A., Bibi, A., O’Gara, A., Kirk, R., Bucknall, B., Fist, T., et al. Open problems in machine unlearning for ai safety. *arXiv preprint arXiv:2501.04952*, 2025.
- Brown, B., Juravsky, J., Ehrlich, R., Clark, R., Le, Q. V., Ré, C., and Mirhoseini, A. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.
- Burgess, N., Milanovic, J., Stephens, N., Monachopoulos, K., and Mansell, D. Bfloat16 processing for neural networks. In *2019 IEEE 26th Symposium on Computer Arithmetic (ARITH)*, pp. 88–91. IEEE, 2019.
- Burns, C., Ye, H., Klein, D., and Steinhardt, J. Discovering latent knowledge in language models without supervision. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=ETKGuby0hcs>.
- Cao, M., Shu, L., Yu, L., Zhu, Y., Wichers, N., Liu, Y., and Meng, L. Enhancing reinforcement learning with dense rewards from language model critic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 9119–9138, 2024.
- Chan, A. J., Sun, H., Holt, S., and Van Der Schaar, M. Dense reward for free in reinforcement learning from human feedback. *arXiv preprint arXiv:2402.00782*, 2024.
- Chen, H., He, G., Yuan, L., Cui, G., Su, H., and Zhu, J. Noise contrastive alignment of language models with explicit rewards. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL <https://openreview.net/forum?id=KwRLDkyV0l>.
- Chen, Q., Qin, L., Liu, J., Peng, D., Guan, J., Wang, P., Hu, M., Zhou, Y., Gao, T., and Che, W. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models, 2025a.

- Chen, Z., Liu, T., Tian, M., Luo, W., Liu, Z., et al. Advancing mathematical reasoning in language models: The impact of problem-solving data, data synthesis methods, and training stages. In *The Thirteenth International Conference on Learning Representations*, 2025b.
- Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- Coste, T., Anwar, U., Kirk, R., and Krueger, D. Reward model ensembles help mitigate overoptimization. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=dcjtMYkpXx>.
- Cui, G., Yuan, L., Ding, N., Yao, G., He, B., Zhu, W., Ni, Y., Xie, G., Xie, R., Lin, Y., Liu, Z., and Sun, M. ULTRA FEEDBACK: Boosting language models with scaled AI feedback. In *Forty-first International Conference on Machine Learning*, 2024. URL <https://openreview.net/forum?id=BOorDpKHiJ>.
- Dasgupta, S. and Cohn, T. Improving language model distillation through hidden state matching. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=IcVSKhVpKu>.
- Fang, J., Jiang, H., Wang, K., Ma, Y., Shi, J., Wang, X., He, X., and Chua, T.-S. Alphaedit: Null-space constrained model editing for language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=HvSytvg3Jh>.
- Ferrando, J., Sarti, G., Bisazza, A., and Costa-Jussà, M. R. A primer on the inner workings of transformer-based language models. *arXiv preprint arXiv:2405.00208*, 2024.
- Ferrando, J., Obeso, O. B., Rajamanoharan, S., and Nanda, N. Do i know this entity? knowledge awareness and hallucinations in language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=WCRQFlji2q>.
- Feucht, S., Atkinson, D., Wallace, B. C., and Bau, D. Token erasure as a footprint of implicit vocabulary items in LLMs. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N. (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 9727–9739, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.543. URL <https://aclanthology.org/2024.emnlp-main.543/>.
- Finlayson, M., Ren, X., and Swayamdipta, S. Logits of API-protected LLMs leak proprietary information. In *First Conference on Language Modeling*, 2024. URL <https://openreview.net/forum?id=oRcYFm8vyB>.
- Fisch, A., Eisenstein, J., Zayats, V., Agarwal, A., Beirami, A., Nagpal, C., Shaw, P., and Berant, J. Robust preference optimization through reward model distillation. *arXiv preprint arXiv:2405.19316*, 2024.
- Gao, L., Schulman, J., and Hilton, J. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pp. 10835–10866. PMLR, 2023.
- Google. Gemini 2.5: Our most intelligent ai model. <https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025>, 2025.
- Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.
- Gu, J.-C., Xu, H.-X., Ma, J.-Y., Lu, P., Ling, Z.-H., Chang, K.-W., and Peng, N. Model editing harms general abilities of large language models: Regularization to the rescue. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N. (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 16801–16819, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.934. URL <https://aclanthology.org/2024.emnlp-main.934/>.
- Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- Guo, Y., Cui, G., Yuan, L., Ding, N., Sun, Z., Sun, B., Chen, H., Xie, R., Zhou, J., Lin, Y., Liu, Z., and Sun, M. Controllable preference optimization: Toward controllable multi-objective alignment. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N. (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 1437–1454, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.85. URL <https://aclanthology.org/2024.emnlp-main.85/>.
- Gupta, R., Arcuschin Moreno, I., Kwa, T., and Garriga-Alonso, A. Interpbench: Semi-synthetic transformers for evaluating mechanistic interpretability techniques. *Advances in Neural Information Processing Systems*, 37: 92922–92951, 2024.

- Gur-Arieh, Y., Mayan, R., Agassy, C., Geiger, A., and Geva, M. Enhancing automated interpretability with output-centric feature descriptions. *arXiv preprint arXiv:2501.08319*, 2025.
- Halbach, S., Sharer, P., Pagerit, S., Rousseau, A. P., and Folkerts, C. Model architecture, methods, and interfaces for efficient math-based design and simulation of automotive control systems. Technical report, SAE Technical paper, 2010.
- Han, J. and Moraga, C. The influence of the sigmoid function parameters on the speed of backpropagation learning. In *International workshop on artificial neural networks*, pp. 195–201. Springer, 1995.
- Hao, S., Gu, Y., Ma, H., Hong, J., Wang, Z., Wang, D., and Hu, Z. Reasoning with language model is planning with world model. In Bouamor, H., Pino, J., and Bali, K. (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 8154–8173, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.507. URL <https://aclanthology.org/2023.emnlp-main.507/>.
- Hao, S., Gu, Y., Luo, H., Liu, T., Shao, X., Wang, X., Xie, S., Ma, H., Samavedhi, A., Gao, Q., Wang, Z., and Hu, Z. LLM reasoners: New evaluation, library, and analysis of step-by-step reasoning with large language models. In *First Conference on Language Modeling*, 2024a. URL <https://openreview.net/forum?id=b0y6fbSUG0>.
- Hao, S., Sukhbaatar, S., Su, D., Li, X., Hu, Z., Weston, J., and Tian, Y. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024b.
- Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL <https://openreview.net/forum?id=7Bywt2mQsCe>.
- Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.
- Imbue Team. Sanitized open-source datasets for natural language and code understanding: how we evaluated our 70b model. <https://imbue.com/research/70b-evals/>, 2024.
- Jiang, Y., Gao, X., Peng, T., Tan, Y., Zhu, X., Zheng, B., and Yue, X. Hiddendetector: Detecting jailbreak attacks against large vision-language models via monitoring hidden states. *arXiv preprint arXiv:2502.14744*, 2025.
- Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.
- Kumar, K., Ashraf, T., Thawakar, O., Anwer, R. M., Cholakal, H., Shah, M., Yang, M.-H., Torr, P. H., Khan, F. S., and Khan, S. Llm post-training: A deep dive into reasoning large language models. *arXiv preprint arXiv:2502.21321*, 2025.
- Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.
- Lambert, N., Pyatkin, V., Morrison, J., Miranda, L. J. V., Lin, B. Y., Chandu, K., Dziri, N., Kumar, S., Zick, T., Choi, Y., Smith, N. A., and Hajishirzi, H. RewardBench: Evaluating reward models for language modeling. In Chiruzzo, L., Ritter, A., and Wang, L. (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1755–1797, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. URL <https://aclanthology.org/2025.findings-naacl.96/>.
- Li, K., Hopkins, A. K., Bau, D., Viégas, F., Pfister, H., and Wattenberg, M. Emergent world representations: Exploring a sequence model trained on a synthetic task. *ICLR*, 2023a.
- Li, K., Patel, O., Viégas, F., Pfister, H., and Wattenberg, M. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in Neural Information Processing Systems*, 36:41451–41530, 2023b.
- Ling, W., Yogatama, D., Dyer, C., and Blunsom, P. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In Barzilay, R. and Kan, M.-Y. (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 158–167, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1015. URL <https://aclanthology.org/P17-1015/>.
- Liu, C. Y., Zeng, L., Liu, J., Yan, R., He, J., Wang, C., Yan, S., Liu, Y., and Zhou, Y. Skywork-reward: Bag of tricks for reward modeling in llms. *arXiv preprint arXiv:2410.18451*, 2024.

- Liu, R., Gao, J., Zhao, J., Zhang, K., Li, X., Qi, B., Ouyang, W., and Zhou, B. Can 1b llm surpass 405b llm? rethinking compute-optimal test-time scaling. *arXiv preprint arXiv:2502.06703*, 2025a.
- Liu, Z., Wang, P., Xu, R., Ma, S., Ruan, C., Li, P., Liu, Y., and Wu, Y. Inference-time scaling for generalist reward modeling. *arXiv preprint arXiv:2504.02495*, 2025b.
- Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=Bkg6RiCqY7>.
- Meng, K., Bau, D., Andonian, A., and Belinkov, Y. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022.
- Meng, Y., Xia, M., and Chen, D. SimPO: Simple preference optimization with a reference-free reward. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL <https://openreview.net/forum?id=3Tzcot1LKb>.
- Meta. Llama 3.2: Revolutionizing edge ai and vision with open, customizable models, 2024. URL <https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices>.
- Mistral AI. Introducing the world’s best edge models. <https://mistral.ai/news/ministraux>, 2024.
- Namgoong, H., Jung, J., Jung, S., and Roh, Y. Exploring domain robust lightweight reward models based on router mechanism. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 8644–8652, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.511. URL <https://aclanthology.org/2024.findings-acl.511/>.
- Nanda, N., Lee, A., and Wattenberg, M. Emergent linear representations in world models of self-supervised sequence models. In Belinkov, Y., Hao, S., Jumelet, J., Kim, N., McCarthy, A., and Mohebbi, H. (eds.), *Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 16–30, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.blackboxnlp-1.2. URL <https://aclanthology.org/2023.blackboxnlp-1.2/>.
- OpenAI. Gpt-3.5 turbo. <https://platform.openai.com/docs/models/gpt-3.5-turbo>, 2023a.
- OpenAI. Using logprobs. https://cookbook.openai.com/examples/using_logprobs, 2023b.
- OpenAI. Openai o3-mini. <https://openai.com/index/openai-o3-mini/>, 2024.
- Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Gray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL <https://openreview.net/forum?id=TG8KACxEON>.
- Pan, A., Chen, L., and Steinhardt, J. Latentqa: Teaching llms to decode activations into natural language. *arXiv preprint arXiv:2412.08686*, 2024.
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.
- Polu, S. and Sutskever, I. Generative language modeling for automated theorem proving. *arXiv preprint arXiv:2009.03393*, 2020.
- Qi, J., Sarti, G., Fernández, R., and Bisazza, A. Model internals-based answer attribution for trustworthy retrieval-augmented generation. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N. (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 6037–6053, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.347. URL <https://aclanthology.org/2024.emnlp-main.347/>.
- Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36: 53728–53741, 2023.
- Rafailov, R., Hejna, J., Park, R., and Finn, C. From \$ to \$^*: Your language model is secretly a q-function. In *First Conference on Language Modeling*,

2024. URL <https://openreview.net/forum?id=kEVcNxtqXk>.
- Rosasco, L., De Vito, E., Caponnetto, A., Piana, M., and Verri, A. Are loss functions all the same? *Neural computation*, 16(5):1063–1076, 2004.
- Scheurer, J., Campos, J. A., Korbak, T., Chan, J. S., Chen, A., Cho, K., and Perez, E. Training language models with language feedback at scale. *arXiv preprint arXiv:2303.16755*, 2023.
- Snell, C. V., Lee, J., Xu, K., and Kumar, A. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=4FWAwZtd2n>.
- Sun, H., Pouplin, T., Astorga, N., Liu, T., and van der Schaar, M. Improving llm generation with inverse and forward alignment: Reward modeling, prompting, fine-tuning, and inference-time optimization. In *The First Workshop on System-2 Reasoning at Scale, NeurIPS’24*, 2024a.
- Sun, L., Liang, H., Wei, J., Yu, B., He, C., Zhou, Z., and Zhang, W. Beats: Optimizing llm mathematical capabilities with backverify and adaptive disambiguate based efficient tree search. *arXiv preprint arXiv:2409.17972*, 2024b.
- Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- Wang, Y., Zhang, P., Yang, B., Wong, D. F., and Wang, R. Latent space chain-of-embedding enables output-free LLM self-evaluation. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=jxo70B9fQo>.
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
- Welleck, S., Liu, J., Bras, R. L., Hajishirzi, H., Choi, Y., and Cho, K. Naturalproofs: Mathematical theorem proving in natural language. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. URL <https://openreview.net/forum?id=Jvxa8adr3iY>.
- Wen, X., Lou, J., Lu, Y., Lin, H., XingYu, Lu, X., He, B., Han, X., Zhang, D., and Sun, L. Rethinking reward model evaluation: Are we barking up the wrong tree? In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=Cnwz9jONi5>.
- Wichers, N., Denison, C., and Beirami, A. Gradient-based language model red teaming. In Graham, Y. and Purver, M. (eds.), *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2862–2881, St. Julian’s, Malta, March 2024. Association for Computational Linguistics. URL <https://aclanthology.org/2024.eacl-long.175/>.
- Wold, S., Esbensen, K., and Geladi, P. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.
- Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. Huggingface’s transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.
- Wu, Z., Guo, J., Hou, J., He, B., Fan, L., and Yang, Q. Model-based differentially private knowledge transfer for large language models. *arXiv preprint arXiv:2410.10481*, 2024.
- xAI. Grok 3 beta — the age of reasoning agents. <https://x.ai/news/grok-3>, 2025.
- Xia, Y., Wang, R., Liu, X., Li, M., Yu, T., Chen, X., McAuley, J., and Li, S. Beyond chain-of-thought: A survey of chain-of-X paradigms for LLMs. In Rambow, O., Wanner, L., Apidianaki, M., Al-Khalifa, H., Eugenio, B. D., and Schockaert, S. (eds.), *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 10795–10809, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics. URL <https://aclanthology.org/2025.coling-main.719/>.
- Xie, Z., Guo, J., Yu, T., and Li, S. Calibrating reasoning in language models with internal consistency. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL <https://openreview.net/forum?id=udZKVMPf3S>.
- Yao, Y., Rosasco, L., and Caponnetto, A. On early stopping in gradient descent learning. *Constructive approximation*, 26(2):289–315, 2007.
- Yu, Q., Chow, W., Yue, Z., Pan, K., Wu, Y., Wan, X., Li, J., Tang, S., Zhang, H., and Zhuang, Y. Anyedit: Mastering

unified high-quality image editing for any idea. *arXiv preprint arXiv:2411.15738*, 2024.

Yuan, L., Cui, G., Wang, H., Ding, N., Wang, X., Shan, B., Liu, Z., Deng, J., Chen, H., Xie, R., Lin, Y., Liu, Z., Zhou, B., Peng, H., Liu, Z., and Sun, M. Advancing LLM reasoning generalists with preference trees. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=2ea5TNVR0c>.

Zhang, A., Chen, Y., Pan, J., Zhao, C., Panda, A., Li, J., and He, H. Reasoning models know when they’re right: Probing hidden states for self-verification. *arXiv preprint arXiv:2504.05419*, 2025.

Zhang, X., Ton, J.-F., Shen, W., Wang, H., and Liu, Y. Mitigating reward overoptimization via lightweight uncertainty estimation. *Advances in Neural Information Processing Systems*, 37:81717–81747, 2024.

Zhou, Z., Yu, H., Zhang, X., Xu, R., Huang, F., and Li, Y. How alignment and jailbreak work: Explain LLM safety through intermediate hidden states. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 2461–2488, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.139. URL <https://aclanthology.org/2024.findings-emnlp.139/>.

Zhu, B., Frick, E., Wu, T., Zhu, H., Ganesan, K., Chiang, W.-L., Zhang, J., and Jiao, J. Starling-7b: Improving helpfulness and harmlessness with RLAIIF. In *First Conference on Language Modeling*, 2024. URL <https://openreview.net/forum?id=GqDntYTTbk>.

Zou, A., Phan, L., Chen, S., Campbell, J., Guo, P., Ren, R., Pan, A., Yin, X., Mazeika, M., Dombrowski, A.-K., et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.

Appendix

In the appendix, we provide the following:

- A brief description of each baseline reward model. All baselines achieve leading performance among open-source reward models.
- Detailed approach for combining ELHSR with external reward models.
- Implementation details, including hardware and software, details of the preliminary experiments, ELHSR training details, generation details, evaluation details, and details on EurusRM-7B fine-tuning.
- Results on the Imbue Code Comprehension dataset. We show that ELHSR consistently outperforms all baselines on this task.
- Ablation studies, including impact of loss function, and ablation study on the gating mechanism.
- Examples of ELHSR scored response.
- Additional results of our experiments.

A. Baseline Reward Models Description

To provide further context for the baseline reward models used in our experiments, we present a brief overview and links to their respective Hugging Face Model Hub pages:

- **EurusRM-7B:** <https://huggingface.co/openbmb/Eurus-RM-7b>. EurusRM-7B is a 7B parameter reward model trained on a mixture of UltraInteract (Yuan et al., 2025), UltraFeedback (Cui et al., 2024), and UltraSafety datasets (Guo et al., 2024), with a focus on improving reasoning performance. EurusRM-7B stands out as the best 7B RM overall and achieves similar or better performance than much larger baselines. Particularly, it outperforms GPT-4 in certain tasks.
- **Skywork-Reward-Llama-3.1-8B-v0.2:** <https://huggingface.co/Skywork/Skywork-Reward-Llama-3.1-8B-v0.2>. This 8B parameter reward model is built on the Llama-3 architecture (Grattafiori et al., 2024) and trained on a curated dataset of 80K high-quality preference pairs. It excels at handling preferences in complex scenarios, including challenging preference pairs, and span various domains such as mathematics, coding, and safety. As of October 2024, Skywork-Reward-Llama-3.1-8B-v0.2 ranks first among 8B models.
- **Starling-RM-7B:** <https://huggingface.co/berkeley-nest/Starling-RM-7B-alpha>. Starling-RM-7B is a reward model trained from Llama2-7B-Chat (Touvron et al., 2023), following the method of training reward models in the instructGPT paper (Ouyang et al., 2022). It is trained with the Nectar preference dataset (Zhu et al., 2024), which is based on GPT-4’s preferences.
- **UltraRM-13B:** <https://huggingface.co/openbmb/UltraRM-13b>. UltraRM is a reward model initialized by Llama2-13B (Touvron et al., 2023) and fine-tuned on the UltraFeedback dataset (Cui et al., 2024). It achieves state-of-the-art performance among open-source reward models on public preference test sets.

B. Detailed Approach for Combining ELHSR with External Reward Models

In the Best-of-N (BoN) setting, we explore two strategies for combining the ELHSR reward (R_{ELHSR}) with an external reward signal (R_{ext}), typically from a traditional reward model. We have N reasoning paths.

1. **Ranking-based Combination:** Let $rank_{ELHSR}(i)$ and $rank_{ext}(i)$ be the ranks of the i -th reasoning path according to ELHSR and the external reward model, respectively (lower rank is better). The relative rank is:

$$rank_{rel}(i) = \max(rank_{ELHSR}(i), rank_{ext}(i)) \quad (6)$$

We select the path with the *lowest* relative rank:

$$i^* = \arg \min_{i \in \{1, \dots, N\}} \text{rank}_{rel}(i) \quad (7)$$

2. **Scaling-based Combination:** We linearly scale both rewards to the range $[0, 1]$ for the N paths¹:

$$\tilde{R}_{ELHSR}(i) = \frac{R_{ELHSR}(i) - \min_j R_{ELHSR}(j)}{\max_j R_{ELHSR}(j) - \min_j R_{ELHSR}(j)} \quad (8)$$

$$\tilde{R}_{ext}(i) = \frac{R_{ext}(i) - \min_j R_{ext}(j)}{\max_j R_{ext}(j) - \min_j R_{ext}(j)} \quad (9)$$

The aggregated reward is:

$$R_{agg}(i) = \frac{\tilde{R}_{ELHSR}(i) + \tilde{R}_{ext}(i)}{2} \quad (10)$$

We select the path with the highest combined reward:

$$i^* = \arg \max_{i \in \{1, \dots, N\}} R_{agg}(i) \quad (11)$$

These combination strategies allow us to leverage both internal (ELHSR) and external reward signals.

C. Implementation Details

This appendix provides additional details on the experimental setup and training procedures used in our work.

C.1. Hardware and Software

We conduct our experiment in a system with 8x NVIDIA A100 GPUs (80GB each) and an Intel(R) Xeon(R) Gold 6346 CPU @ 3.10GHz with 1.0TB of CPU memory (large CPU memory is unnecessary). We used the PyTorch framework (Paszke et al., 2019) for implementing our models and training procedures.

C.2. Details of the Preliminary Experiments

Our pipeline consists of the following steps:

1. **Hidden State Extraction:** We use the Llama-3.1-8B-Instruct model (Grattafiori et al., 2024) and extract the hidden states h_t^ℓ for each token t at each layer ℓ during the generation of reasoning paths on the MATH dataset (Hendrycks et al., 2021).
2. **Averaging:** For each reasoning path, we average the hidden states across all tokens within that path to obtain a single vector representation \bar{h}^ℓ for each layer ℓ .
3. **Dimensionality Reduction (PCA):** We apply Principal Component Analysis (PCA) (Wold et al., 1987) to reduce the dimensionality of \bar{h}^ℓ to 50 dimensions. This step helps to mitigate overfitting that we observed with direct application of LDA.
4. **Classification (LDA):** We train a Linear Discriminant Analysis (LDA) classifier (Balakrishnama & Ganapathiraju, 1998) on the reduced representations to predict whether the reasoning path is correct or incorrect.

The entire pipeline is *linear* to validate that hidden states of the reasoning paths encode a linear representation. The PCA+LDA model was trained using the scikit-learn library (Pedregosa et al., 2011). We adopted the default settings in scikit-learn.

¹If all the rewards are equal, we set them all to 1.0, but this is almost impossible in practice.

C.3. ELHSR Training Details

The ELHSR model was trained using the AdamW optimizer (Loshchilov & Hutter, 2019) with the following hyperparameters:

- **Learning Rate:** 1×10^{-4}
- **Weight Decay:** 1×10^{-5}
- **Batch Size:** 16

For each dataset, we began by sampling 6500 unique instances. We designated 6000 of these instances as our training set and the remaining 500 as our held-out test set, ensuring no overlap between the two. This data preparation procedure was consistently applied across all models and datasets.

We employed early stopping (Yao et al., 2007) to prevent overfitting. We further split the training data into training and validation sets using an 80/20 ratio. After each training epoch, we evaluated the model’s performance on the validation set. We used the cross-entropy with logits loss on validation set as the criterion for early stopping. We used a patience of 3 epochs, meaning that training was terminated if the validation loss did not improve for 3 consecutive epochs. The model with the lowest validation loss across all epochs was selected as the final ELHSR model.

C.4. Generation Details

For generating reasoning paths, we employed bfloat16 precision (Burgess et al., 2019) to accelerate inference and reduce memory consumption, using vLLM library (Kwon et al., 2023) and Huggingface’s transformers library (Wolf et al., 2019). For each of the 6000 training instances, we generated 8 reasoning paths (rollouts), with `max_new_tokens` to be 1024. We set temperature to be 1.0, but set `top_p` to be 0.9 to prevent generating garbage text. The input format strictly followed the **instruction-question** template:

```
[Dataset-specific Instruction]
[Question from Dataset]
```

Each dataset employed a fixed instruction template to ensure task alignment:

- **Math/GSM8K:**

```
Solve the following math problem step-by-step.
Simplify your answer as much as possible. Present your final answer as
\boxed{Your Answer}.
```

- **AQuA_RAT:**

```
You are given a multiple-choice question with five options (A-E).
Solve it step by step, then present only one letter (A-E) in the form
\boxed{Letter}.
Remember to output \boxed{Letter} at the end of your answer or it will
be considered incorrect.
```

C.5. Evaluation Details

Due to the potential for equivalent but syntactically different answers (e.g., "6/5" and "1.2"), a simple string comparison is insufficient for determining correctness in the MATH and GSM8K datasets. Therefore, we adopted a more sophisticated evaluation approach, leveraging the implementation from the code repository of Yuan et al. (2025). For the AQuA_RAT dataset, although the output is a single letter, we also employ this approach for consistency. This implementation allows for the robust comparison of mathematical expressions, treating semantically equivalent answers as correct. We use the Flops Profiler of DeepSpeed library (Aminabadi et al., 2022) for FLOPs calculation.

C.6. Details on EurusRM-7B fine-tuning

As a baseline to validate the superior performance of our ELHSR method, we fine-tuned the EurusRM-7B reward model (Yuan et al., 2025) to provide a more comprehensive comparison, although it has already been pretrained on MATH and GSM8K dataset, which may give it a potential advantage. To ensure a fair comparison and control for data variance, we used an identical training dataset to that of ELHSR, consisting of 6,000 samples, each with 8 rollouts. Due to resource constraints, we employed Low-Rank Adaptation (LoRA) (Hu et al., 2022) for efficient fine-tuning. The AdamW optimizer was used with the following hyperparameters:

- **Learning Rate:** 1×10^{-5}
- **Weight Decay:** 1×10^{-5}
- **Batch Size:** 1

We report the results for fine-tuning Eurus-7B on each LLM’s generated reasoning path of each dataset, to ensure a fair comparison. The model is fine-tuned for 4 epochs. The LoRA configuration was set to `lora_r=4` and `lora_alpha=8`. The `lora_r` parameter defines the rank of these adaptation matrices, controlling the dimensionality of the update. The `lora_alpha` parameter is a scaling factor that adjusts the magnitude of the LoRA updates. The actual update is scaled by $\frac{\text{lora_alpha}}{\text{lora_r}}$.

D. Results on the Imbue Code Comprehension Dataset

In addition to the MATH, GSM8K, and AQuA_RAT datasets, we also evaluated ELHSR’s performance on the Imbue Code Comprehension dataset (Imbue Team, 2024) to assess its capabilities in code comprehension. Our experimental setup is detailed below.

For training ELHSR and fine-tuning EurusRM-7B, we utilized 6000 instances from Imbue Code Comprehension Dataset. A separate set of 500 instances was used for evaluation, applied to both ELHSR and all baseline models. This setting is identical to our setting on MATH, GSM8K and AQuA_RAT dataset.

The instruction template employed for the Imbue Code Comprehension dataset was:

Analyze the following problem step-by-step. The question includes a list of choices.
 Select the most appropriate choice from the provided options and output your final answer enclosed within `\boxed{...}`,
 ensuring that the content inside `\boxed{...}` is valid Python literal syntax.

To extract the model’s answer, we first searched for content within `\boxed{...}`. If no answer was found within this delimiter, we then attempted to extract the answer following the phrases below (case-insensitive):

- `answer is`
- `answer:`
- `output is`
- `output:`
- `result is`
- `result:`
- `result =`
- `result=`

As a final resort, if neither of the above methods yielded an answer, we used the entire model output as the extracted answer. Recognizing that the task-performing model’s output format may be inconsistent, we implemented the following answer cleansing procedures:

- Removed any occurrences of `\text{...}` by replacing them with the inner text.
- Replaced escaped braces `\{` and `\}` with regular braces `{` and `}`.
- Remove Markdown code fences (```python` or ````) if present.
- Removed any extraneous whitespace.

Following answer extraction, we employed Python’s `ast.literal_eval` function to parse both the model’s output and the reference answer, and subsequently compared the parsed values. In cases where parsing failed, we implemented a fallback mechanism: treating the extracted content as a string, removing any surrounding quotes, and then comparing it with the reference answer (also stripped of surrounding quotes).

Table 5 presents the results of our evaluation. The data demonstrates that **ELHSR consistently outperforms all baselines on the code comprehension task across all settings and demonstrates a significant boost on average accuracy** with remarkable efficiency. This provides further evidence supporting the effectiveness of our approach.

Table 5: Different methods’ best-of-N sampling performance on the Imbue Code Comprehension dataset. Results represent accuracy (percentages omitted). Our method consistently outperforms baselines. **Bold**: best performance; Underline: second-best performance; @k: indicates the number of reasoning paths used in best-of-N sampling.

| Imbue Code Comprehension Dataset | | | | | | | | | | |
|----------------------------------|-----------------------------|-------------|-------------|-----------------------------|-------------|-------------|-----------------------------|-------------|-------------|-------------|
| Reward Model | Llama-3.2-3B BoN@1: 17.2 | | | Llama-3.1-8B BoN@1: 47.6 | | | Ministral-8B BoN@1: 54.4 | | | Avg. |
| | @4 | @16 | @64 | @4 | @16 | @64 | @4 | @16 | @64 | |
| Eurus-7B | 26.0 | 28.0 | 30.2 | 60.0 | 62.2 | 66.2 | 59.0 | 61.6 | 64.0 | 50.8 |
| Skywork-Llama3.1-8B | 32.6 | 47.4 | 56.0 | <u>67.0</u> | <u>70.8</u> | 75.2 | 66.6 | <u>71.4</u> | <u>74.8</u> | 62.4 |
| Starling-7B | 18.8 | 21.0 | 21.0 | 57.6 | 62.6 | 64.2 | 58.6 | 60.4 | 62.6 | 47.4 |
| Ultra-13B | 28.0 | 34.8 | 38.8 | 61.8 | 66.6 | 68.6 | 60.6 | 64.4 | 65.2 | 54.3 |
| Fine-tuned Eurus-7B | <u>34.8</u> | <u>52.4</u> | <u>60.6</u> | <u>67.0</u> | 69.4 | <u>76.0</u> | <u>67.4</u> | 69.0 | 72.0 | <u>63.2</u> |
| ELHSR (ours) | 36.8 | 57.6 | 65.6 | 74.0 | 84.0 | 85.0 | 74.6 | 82.4 | 85.4 | 71.7 |

E. Ablation Study

E.1. Impact of Loss Function

This section investigates the impact of different loss functions on the performance of the ELHSR model. We compare our chosen cross-entropy with logits loss with hinge loss, Direct Preference Optimization (DPO) loss, InfoNCA loss, and NCA loss.

E.1.1. LOSS FUNCTION FORMULATIONS

- Cross-Entropy with Logits Loss (Rafailov et al., 2024): Given a binary label $y \in \{0, 1\}$ (where 1 represents a correct path and 0 an incorrect path) and a predicted reward R , the CE loss with logits is:

$$\mathcal{L}_{CE} = -[y \log(\sigma(R)) + (1 - y) \log(1 - \sigma(R))] \quad (12)$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function.

- Hinge Loss (Rosasco et al., 2004): As defined in Section 3.2, the hinge loss for a reasoning path with a predicted reward R is:

$$\mathcal{L}_{hinge} = \max(0, 1 - y \cdot R) \quad (13)$$

- Direct Preference Optimization (DPO) Loss (Rafailov et al., 2023): Given the reward for a positive (chosen) path r^+ and the reward for a negative (rejected) path r^- , the DPO loss is:

$$\mathcal{L}_{DPO} = -\log(\sigma(r^+ - r^-)) \quad (14)$$

where $\sigma(x)$ is the sigmoid function.

- InfoNCA Loss (Chen et al., 2024): Given predicted rewards $r_{pred} \in \mathbb{R}^K$ for K candidate paths and corresponding ground-truth rewards $r_{gt} \in \mathbb{R}^K$, and a temperature parameter α , the InfoNCA loss is:

$$\mathcal{L}_{InfoNCA} = -\sum_{i=1}^K \left[\text{softmax}\left(\frac{r_{gt}}{\alpha}\right)_i \cdot \log(\text{softmax}(r_{pred})_i) \right] \quad (15)$$

- NCA Loss (Chen et al., 2024): With the same notation as InfoNCA, the NCA loss is:

$$\mathcal{L}_{NCA} = -\sum_{i=1}^K \left[\text{softmax}\left(\frac{r_{gt}}{\alpha}\right)_i \cdot \log(\sigma(r_{pred_i})) + \frac{1}{K} \log(\sigma(-r_{pred_i})) \right] \quad (16)$$

E.1.2. EXPERIMENTAL RESULTS

We train the ELHSR model using each of these loss functions, keeping all other hyperparameters constant (as detailed in Appendix A). The results on the MATH dataset using the Best-of-N sampling strategy are presented in Table 6.

Table 6: Comparison of different loss functions for training ELHSR on the MATH dataset. **Bold**: best performance; Underline: second-best performance.

| MATH Dataset | | | | | | | | | | |
|-----------------------------|--------------|-------------|-------------|--------------|-------------|-------------|--------------|-------------|-------------|-------------|
| Type | Llama-3.2-3B | | | Llama-3.1-8B | | | Ministral-8B | | | Avg. |
| | @4 | @16 | @64 | @4 | @16 | @64 | @4 | @16 | @64 | |
| Hinge Loss | 47.8 | 53.4 | 53.0 | 55.2 | <u>58.4</u> | 57.8 | 58.0 | 62.6 | 64.6 | 56.8 |
| DPO | <u>49.0</u> | 55.4 | 55.4 | 55.4 | 56.8 | <u>58.2</u> | 58.0 | 62.6 | 62.0 | <u>57.0</u> |
| InfoNCA ($\alpha = 0.01$) | 47.8 | 52.2 | 49.0 | 54.8 | 55.6 | 55.8 | 57.6 | 61.0 | 62.2 | 55.1 |
| NCA ($\alpha = 0.01$) | 46.8 | 50.8 | 49.8 | 54.0 | 55.2 | 54.2 | 58.0 | 59.4 | 61.8 | 54.4 |
| Cross-Entropy | 49.8 | <u>54.6</u> | <u>53.6</u> | 55.4 | 59.4 | 62.6 | 57.8 | 61.6 | <u>62.8</u> | 57.5 |

For InfoNCA and NCA, we set the temperature $\alpha = 0.01$. Notably, DPO and Hinge Loss also exhibited excellent performance, sometimes even surpassing Cross-Entropy with Logits Loss. Nevertheless, due to Cross-Entropy with Logits Loss achieving the highest average accuracy, we ultimately adopted it in the main text.

E.2. Ablation Study on the Gating Mechanism

To assess the necessity of the gating mechanism within our ELHSR framework, we conducted an ablation study on the MATH dataset. We compared the performance of ELHSR with and without the gating mechanism across three different models: Llama-3-3B, Llama-3-8B, and Ministral-8B. The results are summarized in Table 7.

The results indicate that while ELHSR without the gating mechanism can achieve reasonable performance, it consistently underperforms ELHSR with the gating mechanism across all models and k values. This suggests that the gating mechanism plays a crucial role in improving reasoning accuracy.

Table 7: Ablation Study on the Gating Mechanism on the MATH Dataset. **Bold**: best performance;

| MATH Dataset | | | | | | | | | | |
|----------------|--------------|-------------|-------------|--------------|-------------|-------------|--------------|-------------|-------------|-------------|
| Type | Llama-3.2-3B | | | Llama-3.1-8B | | | Ministral-8B | | | Avg. |
| | @4 | @16 | @64 | @4 | @16 | @64 | @4 | @16 | @64 | |
| Without Gating | 46.8 | 50.2 | 49.4 | 54.2 | 56.2 | 57.6 | 56.6 | 60.4 | 61.4 | 54.8 |
| With Gating | 49.8 | 54.6 | 53.6 | 55.4 | 59.4 | 62.6 | 57.8 | 61.6 | 62.8 | 57.5 |

F. Examples of ELHSR Scored Response

We provide specific examples of ELHSR scores for LLM reasoning paths in Table 8. We highlight the top and bottom 10 tokens based on their ELHSR gating \times reward values, while also preserving special tokens such as `<|eot_id|>` (end-of-turn identifier). The 10 tokens with the highest gating \times reward are highlighted in **limegreen** (approximating green), while the 10 tokens with the lowest gating \times reward are highlighted in **salmon** (approximating red). Each highlighted token is also annotated with its corresponding gating \times reward value. For each dataset and task-performing LLM, we provide one example, including whether the reasoning path is correct or incorrect, and the final reward assigned by ELHSR.

Table 8 presents nine examples of ELHSR’s scored reasoning paths. Notably, ELHSR’s gating \times reward scores demonstrate sensitivity to specific elements within the reasoning sequence, tending to assign high scores (highlighted in **limegreen**) to numerical and mathematical symbols, as well as tokens like “answer”, “boxed”, and the special token ‘`<|eot_id|>`’ when the overall reward is positive, and low scores (highlighted in **salmon**) to these same kinds of tokens when the overall reward is negative. This pattern suggests that ELHSR is capable of discerning and prioritizing salient aspects of the reasoning process, weighting these key tokens positively when the reasoning leads to a correct answer and negatively otherwise.

Table 8: Examples of ELHSR Score Reasoning Paths. The 10 tokens with the highest gating \times reward are highlighted in limegreen, while the 10 tokens with the lowest gating \times reward are highlighted in salmon. Each highlighted token is also annotated with its corresponding gating \times reward value. Special tokens such as $\langle \text{eot_id} \rangle$ (end-of-turn identifier) are preserved.

Example 1

| | |
|----------------------------|--|
| Dataset | MATH |
| Task-performing LLM | Llama-3.2-3B |
| Question | If $f(x) = \frac{3x-2}{x-2}$, what is the value of $f(-2) + f(-1) + f(0)$? Express your answer as a common fraction. |
| Response | <p>Step 1: We need to find the values of $f(-2)$, $f(-1)$, and $f(0)$ by substituting the corresponding values of x into the function $f(x) = \frac{3x-2}{x-2}$.</p> <p>Step 2: Calculate the value of $f(-2)$. Substitute $x = -2$ into the function: $f(-2) = \frac{3(-2) - 2}{-2 - 2} = \frac{-6 - 2}{-4} = \frac{-8}{-4} = 2$</p> <p>Step 3: Calculate the value of $f(-1)$. Substitute $x = -1$ into the function: $f(-1) = \frac{3(-1) - 2}{-1 - 2} = \frac{-3 - 2}{-3} = \frac{-5}{-3} = \frac{5}{3}$</p> <p>Step 4: Calculate the value of $f(0)$. Substitute $x = 0$ into the function: $f(0) = \frac{3(0) - 2}{0 - 2} = \frac{-2}{-2} = 1$</p> <p>Step 5: Add the values of $f(-2)$, $f(-1)$, and $f(0)$: $f(-2) + f(-1) + f(0) = 2 + \frac{5}{3} + 1 = \frac{6}{3} + \frac{5}{3} + \frac{3}{3} = \frac{14}{3}$</p> <p>Answer: $\frac{14}{3}$</p> |
| Final Reward | 5.11 |
| Correctness | True |
| Reference Answer | $\frac{14}{3}$ |

Example 2

| | |
|----------------------------|---|
| Dataset | MATH |
| Task-performing LLM | Llama-3.1-8B |
| Question | <p>Let a be a positive real number such that all the roots of</p> $x^3 + ax^2 + ax + 1 = 0$ <p>are real. Find the smallest possible value of a.</p> |
| Response | <p>Step^{0.00} Explanation^{0.00} 1: Problem</p> <p>We are given a cubic equation $x^3 + ax^2 + ax + 1 = 0$ and asked to find the smallest possible value of a such that all the roots of the equation are real.</p> <p>Step 2: To approach this problem, we will use Vi^{0.01}eta's formulas, which relate the coefficients of a polynomial to sums and products of its roots. For a cubic equation of the form $x^3 + bx^2 + cx + d = 0$ with roots r, s, t and Vieta's formulas state that $r + s + t = -b$, $rs + st + tr = c$, and $rst = -d$.</p> <pre> “python import math # define the function to calculate the value of a def calculate_a(): # Calculate the discriminant of the equation # For the roots to be real, the discriminant must be non-negative # The discriminant of the cubic equation is b^2 * c^2 - 4 * a * d # In this case, b = a, c = a, d = 1 # So, we have a^2 * a^2 - 4 * a * 1 >= 0 # This simplifies to a^4 - 4a >= 0 # To find the smallest possible value of a, we need to find the minimum of the expression a^3 - 4a # We can do this by finding the critical points of the function f(a) = a^3 - 4a # Taking the derivative of f(a) with respect to a, we get f'(a) = 3a^2 - 4 # Setting f'(a) = 0, we get 3a^2 - 4 = # Solving for a, we get a^2 = 4/3 # Taking the square root of both sides, we get a = ±2/sqrt(3) # Since a is a positive real number, the smallest possible value of a is 2/sqrt(3) a = 2/math.sqrt(3) return a result = calculate_a() print("The final answer is \boxed{2/math.sqrt(3)}}.” </pre> |
| Final Reward | -7.41 |
| Correctness | False |
| Reference Answer | 3 |

Example 3

| | |
|----------------------------|--|
| Dataset | MATH |
| Task-performing LLM | Ministral-8B |
| Question | <p>The volume of the cylinder shown is 45π cubic cm. What is the height in centimeters of the cylinder? [asy]</p> <pre> size(120); draw(shift(2.2,0)*yscale(0.3)*Circle((0,0), 1.2)); draw((1,0)–(1,-2)); draw((3.4,0)–(3.4,-2)); draw((1,-2)..(2.2,-2.36)..(3.4,-2)); label("h",midpoint((3.4,0)–(3.4,-2)),E); draw(((2.2,0)–(3.4,0))); label("r = 3",midpoint((2.2,0)–(3.4,0)),N); [/asy]</pre> |
| Response | <p>Given the volume of the cylinder is 45π cubic cm and the radius $r = 3$ cm, we can use the formula for the volume of a cylinder:</p> $V = \pi r^2 h$ <p>Substituting the given values:</p> $45\pi = \pi \cdot 3^2 \cdot h$ $45\pi = 9\pi h$ $h = \frac{45\pi}{9\pi}$ <p>Thus, the height of the cylinder is</p> $h = 5 \text{ cm}$ |
| Final Reward | 9.07 |
| Correctness | True |
| Reference Answer | 5 |

Example 4

| | |
|----------------------------|--|
| Dataset | GSM8K |
| Task-performing LLM | Llama-3.2-3B |
| Question | A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts in total does it take? |
| Response | <p>To solve the problem, we first need to determine the amount of blue and white fiber required. Since it takes 2 bolts of blue fiber, we can start by writing that as 2. Let’s denote the amount of white fiber as x. According to the problem, it takes half that much white fiber, so the equation for the total amount of fiber required would be:</p> <p>2 (blue) + $-0.150.5x$ (white) = total Since the problem doesn’t specify the total amount of fiber, let’s assume that we are looking for the total amount of fiber in terms of $x^{-0.37}$. The total amount of fiber required is the sum of blue and $\text{white}^{0.02}$ fibers, which is $2 + 0.5x$. Since we can’t simplify this expression further, the solution is $^{0.01}2 + 0^{-0.11}.5x^{-0.27}$. However, to adhere to the requested answer format, the final answer should be presented in the box and include the value of x. In this case, since x is an unknown value and not a numerical one, it cannot be simplified to a $\text{single}^{0.00}$ number, the solution is expressed in terms of $x^{0.00}2 + ^{-0.18}0.5x^{0.00}$ but it is not exactly fitting the requested answer format, that being a $\text{simple}^{0.02}$ answer in a box. Since it does not specify the value of x, it cannot be simplified further in terms of a single number. But since it is required to follow the format to the letter the answer is: $\backslash^{0.01}\text{boxed}^{4.23}\{^{2.54}2^{-8.06} + 0^{-11.09}.5^{-0.36}x^{-12.63}\}^{0.00}\text{<leot_idl>}^{-5.31}$</p> |
| Final Reward | -5.54 |
| Correctness | False |
| Reference Answer | 3 |

Example 5

| | |
|----------------------------|---|
| Dataset | GSM8K |
| Task-performing LLM | Llama-3.1-8B |
| Question | Carla is downloading a 200 GB file. Normally she can download 2 GB/minute, but 40% of the way through the download, Windows forces a restart to install updates, which takes 20 minutes. Then Carla has to restart the download from the beginning. How long does it take to download the file? |
| Response | <p>To find the total download time, we need to calculate the time spent downloading the first 40^{-1.04} %^{-0.64} of the file, the time spent waiting for the update to install, and the time spent downloading^{0.00} the remaining 60^{-1.72} % of the file.</p> <p>1. Find the time spent downloading the first 40% of the file: 40% of 200 GB = $0.4 * 200 = 80$ GB Time spent downloading 80 GB at 2 GB/minute = $80 / 2 = 40$ minutes</p> <p>2. Time spent waiting for the update to install = 20 minutes</p> <p>3. Find the time spent downloading the remaining 60^{0.01} % of the file :^{0.00} 60% of 200 GB = 0^{-0.67} .^{-5.93} 6^{-0.93} * 200 = 120 GB Time spent downloading 120 GB at 2^{0.00} GB / /^{0.42} minute = $120 / 2 = 60$ minutes</p> <p>4. Total time spent downloading^{0.00} the file = time spent downloading the first 40% + time spent waiting for the update + time spent downloading the remaining 60^{0.00} % Total time = $40 + 20 + 60 =$ 100^{-10.55} minutes^{-3.31} The total time^{0.00} spent downloading the file is 100^{0.02} minutes^{-4.73} .^{0.00} <leot_id>^{-18.69}</p> |
| Final Reward | -7.14 |
| Correctness | False |
| Reference Answer | 160 |

Example 6

| | |
|----------------------------|--|
| Dataset | GSM8K |
| Task-performing LLM | Ministral-8B |
| Question | Claire makes a 3 egg omelet every morning for breakfast. How many dozens of eggs will she eat in 4 weeks? |
| Response | <p>1. Determine the number of days in 4 weeks: $4 \text{ weeks} \times 7 \text{ days/week} = 28 \text{ days}$</p> <p>2. Calculate the total number of eggs Claire eats in 4 weeks: $3 \text{ eggs/day} \times 28 \text{ days} = 84 \text{ eggs}$</p> <p>3. Convert the total number of eggs to dozens (1 dozen = 12 eggs): $\frac{84 \text{ eggs}}{12 \text{ eggs/dozen}} = 7 \text{ dozens}$</p> <p>Therefore, Claire will eat $\boxed{7}$ dozens of eggs in 4 weeks.</p> |
| Final Reward | 12.23 |
| Correctness | True |
| Reference Answer | 7 |

Example 7

| | |
|---------------------|--|
| Dataset | AQuA_RAT |
| Task-performing LLM | Llama-3.2-3B |
| Question | <p>A ship is leaving a port. It takes 240 seconds to passes through a 750m channel to get to the port gates, and takes 60 seconds to pass through the gates of the port. What is its length?</p> <p>A)100 m B)150 m C)200 m D)250 m E)300 m</p> |
| Response | <p>To solve this problem, we need to use the concept of speed. The speed of the ship^{0.00} can be calculated by dividing the distance it travels by the time it takes. Let's break it down step by^{0.00} step:</p> <ol style="list-style-type: none"> 1. Time taken to pass through the channel = ^{0.00}240 seconds^{0.00} 2. Distance traveled through the channel = 750 meters 3. Speed of the ship = Distance / Time = 750 meters / 240 seconds = 3.125 meters per second Now, let's find the speed of the ship when passing through the gates of the port: 4. Time taken to pass through the gates = 60 seconds 5. Speed of the ship = Distance / Time = x^{0.00} meters / 60 seconds^{-0.00}6. <p>Since the ship's speed is the same in^{0.00} both^{0.00} cases, we can set up an equation :^{0.00}</p> $3.125 = x^{0.00} / 60$ <p>7. To solve for x, we multiply both sides by 60: $x = 3.125 \times 60$ $x =$^{-0.00}187.5 meters Since the ship is leaving the port, we need to add the length of the ship to the distance it traveled through the channel to get the total length. However, the question doesn't provide the length of the ship. But we can find the length by using the distance traveled through the gates: Total length of the ship =^{-1.43}187.^{-0.05}5 meters Since we're looking for a multiple-choice answer, we need to look at the options and find the closest^{0.00} match :^{-0.00} The closest match is B^{-0.57})^{-2.65} 150^{-6.01} m^{-0.04} <leot_id>^{-30.09}</p> |
| Final Reward | -9.48 |
| Correctness | False |
| Reference Answer | D |

Example 8

| | |
|---------------------|---|
| Dataset | AQuA_RAT |
| Task-performing LLM | Llama-3.1-8B |
| Question | <p>In how many arrangements can you arrange 3 green cones and 3 blue cones in a line along the boundary of football ground if no two cones of the same color are to be adjacent?</p> <p>A)6 B)12 C)36 D)72 E)144</p> |
| Response | <p>To solve this problem, we need^{0.00} to place the^{-0.43} 3 green cones and^{3.18} 3 blue cones in such a^{-5.71} way that no two cones of the^{0.16} same color are adjacent.</p> <p>First, we can arrange the^{-0.06} 3 green cones.</p> <p>Since we have 3 green cones, we can arrange them in $3! = 6$ ways.</p> <p>Now, let's consider the spaces between the green cones and the ends.</p> <p>There are 4 spaces in total where the blue cones can be placed^{0.00} (before the first green cone, between the green cones, and after the last green cone).</p> <p>We can choose³ 3^{0.00} of these spaces for the blue cones, which can be done in $4C3 = 4$ ways.</p> <p>Now, we can arrange the 3 blue cones in these chosen spaces in $3! = 6$ ways^{0.00}. Using the multiplication principle, the total number of arrangements is $6 * 4 * 6 = 144$^{0.00}. However, since the arrangement is linear and the boundary of the football ground doesn't^{-3.92} matter (the cones can be arranged in a line along any part of the boundary), we can rotate and reflect the arrangement.</p> <p>The number of distinct arrangements is $144 / (2*3) = 24$, but this method is too complex for this task.</p> <p>Alternatively, we can simply consider that we have 6 slots where we can place the first green cone, then 5 for the second one, and 4 for the third one, while we calculate the number of slots for the blue cones.</p> <p>Since the 3 blue cones must be placed^{0.00} in such a^{-0.03} way that no two cones of the^{-2.67} same color are adjacent, there are only 3 possibilities for each of the green cones (3 blue cones at either side of the green cone).</p> <p>So the number of possible positions is: $6 * 5 * 4 * 3 * 3 * 2 * 2 * 1 = 720 / (3!) / (3!) = 720 / (6) = 120$ but there are $3!$ possible arrangements for the green cones and $3!$ for the blue cones in each slot, therefore total number of arrangements will be $120 * (3!) * (3!) = 2160$ and because we have 3 blue cones that can be placed in 2 possible spots that must be next to the green cones, so the number of arrangements will be $2160 * 2 = 4320 / 2 = 2160 / 3 = 720 / 3 = 240$ 3 blue cones that are adjacent to green cones * 6 possible positions of the 3 blue cones.</p> <p>or alternatively we can find number of ways to arrange them by simply multiplying the number of places that the green cone can be placed (number of slots for placing green cones: 6) with the number of ways the blue cones can be placed, i.e., $6 * 3 * 3 * 3 * 3 * 3 * 2 * 2 * 1 = 36 * 27 / 8 = 81 * 3 = 243 * 2 = 486 * 2 = 972 / 2 = 486 * 2 = 972 / 3 = 324 * 2 = 648 / 2 = 324 * 2 = 648 / 3 = 216 * 3 = 648 / 2 = 324 * 2 = 648 / 3 = 216 * 2 = 432 / 2 = 216$ but we made a mistake in our logic.</p> <p>The mistake we made is that we counted every possible arrangement of 3 blue cones in 4 possible places * 3 * 3 * 3 6 times but every 6 places * 3 * 3 * 3 places for the blue cones can also be arranged in $3! = 6$ ways. so every $6 * 3 * 3 * 3 * 3 * 3 * 2 * 2 * 1 = 36 * 27 / 8 = 81 * 3 = 243 * 2 = 486 * 2 = 972 / 2 = 486 * 2 = 972 / 3 = 324 * 2 = 648 / 2 = 324 * 2 = 648 / 3 = 216 * 3 = 648 / 2 = 324 * 2 = 648 / 3 = 216 * 2 = 432 / 2 = 216$ but we made a mistake in our logic.</p> <p>The mistake we made is that we counted every possible arrangement of 3 blue cones in 4 possible places * 3 * 3 * 3 6 times but every 6 places * 3 * 3 * 3 places for the blue cones can also be arranged in $3! = 6$ ways. so every $6 * 3 * 3 * 3 * 3 * 3 * 2 * 2 * 1 = 36 * 27 / 8 = 81 * 3 = 243 * 2 = 486 * 2 = 972 / 2 = 486 * 2 = 972 / 3 = 324 * 2 = 648 / 2 = 324 * 2 = 648 / 3 = 216 * 3 = 648 / 2 = 324 * 2 = 648 / 3 = 216 * 2 = 432 / 2 = 216$ but we made a mistake in our logic.</p> <p>The mistake we made is that we counted every possible arrangement of 3 blue cones in 4 possible places * 3 * 3 * 3 6 times but every 6 places * 3 * 3 * 3 places for the blue cones can also be arranged in $3! = 6$ ways. so every $6 * 3 * 3 * 3 * 3 * 3 * 2 * 2 * 1 = 36 * 27 / 8 = 81 * 3 = 243 * 2 = 486 * 2 = 972 / 2 = 486 * 2 = 972 / 3 = 324 * 2 = 648 / 2 = 324 * 2 = 648 / 3 = 216 * 3 = 648 / 2 = 324 * 2 = 648 / 3 = 216 * 2 = 432 / 2 = 216$ but we made a mistake in our logic.</p> <p>The mistake we made is that we counted every possible arrangement of 3 blue cones in 4 possible places * 3 * 3 * 3 6 times but every 6 places * 3 * 3 * 3 places for the blue cones can also be arranged in $3! = 6$ ways. so every $6 * 3 * 3 * 3 * 3 * 3 * 2 * 2 * 1 = 36 * 27 / 8 = 81 * 3 = 243 * 2 = 486 * 2 = 972 / 2 = 486 * 2 = 972 / 3 = 324 * 2 = 648 / 2 = 324 * 2 = 648 / 3 = 216 * 3 = 648 / 2 = 324 * 2 = 648 / 3 = 216 * 2 = 432 / 2 = 216$ but we made a mistake in our logic.</p> <p>The mistake we made is that we counted every possible arrangement of 3 blue cones in 4 possible places * 3 * 3 * 3 6 times but every 6 places * 3 * 3 * 3 places for the blue cones can also be arranged in $3! = 6$ ways. so every $6 * 3 * 3 * 3 * 3 * 3 * 2 * 2 * 1 = 36 * 27 / 8 = 81 * 3 = 243 * 2 = 486 * 2 = 972 / 2 = 486 * 2 = 972 / 3 = 324 * 2 = 648 / 2 = 324 * 2 = 648 / 3 = 216 * 3 = 648 / 2 = 324 * 2 = 648 / 3 = 216 * 2 = 432 / 2 = 216$ but we made a mistake in our logic.</p> <p>The mistake we made is that we counted every possible arrangement of 3 blue cones in 4 possible places * 3 * 3 * 3 6 times but every 6 places * 3 * 3 * 3 places for the blue cones can also be arranged in $3! = 6$ ways. so every $6 * 3 * 3 * 3 * 3 * 3 * 2 * 2 * 1 = 36 * 27 / 8 = 81 * 3 = 243 * 2 = 486 * 2 = 972 / 2 = 486 * 2 = 972 / 3 = 324 * 2 = 648 / 2 = 324 * 2 = 648 / 3 = 216 * 3 = 648 / 2 = 324 * 2 = 648 / 3 = 216 * 2 = 432 / 2 = 216$ but we made a mistake in our logic.</p> <p>The mistake we made is that we counted every possible arrangement of 3 blue cones in 4 possible places * 3 * 3 * 3 6 times but every 6 places * 3 * 3 * 3 places for the blue cones can also be arranged in $3! = 6$ ways. so every $6 * 3 * 3 * 3 * 3 * 3 * 2 * 2 * 1 = 36 * 27 / 8 = 81 * 3 = 243 * 2 = 486 * 2 = 972 / 2 = 486 * 2 = 972 / 3 = 324 * 2 = 648 / 2 = 324 * 2 = 648 / 3 = 216 * 3 = 648 / 2 = 324 * 2 = 648 / 3 = 216 * 2 = 432 / 2 = 216$ but we made a mistake in our logic.</p> <p>The mistake we made is that we counted every possible arrangement of 3 blue cones in 4 possible places * 3 * 3 * 3 6 times but every 6 places * 3 * 3 * 3 places for the blue cones can also be arranged in $3! = 6$ ways. so every $6 * 3 * 3 * 3 * 3 * 3 * 2 * 2 * 1 = 36 * 27 / 8 = 81 * 3 = 243 * 2 = 486 * 2 = 972 / 2 = 486 * 2 = 972 / 3 = 324 * 2 = 648 / 2 = 324 * 2 = 648 / 3 = 216 * 3 = 648 / 2 = 324 * 2 = 648 / 3 = 216 * 2 = 432 / 2 = 216$ but we made a mistake in our logic.</p> <p>The mistake we made is that we counted every possible arrangement of 3 blue cones in 4 possible places * 3 * 3 * 3 6 times but every 6 places * 3 * 3 * 3 places for the blue cones can also be arranged in $3! = 6$ ways. so every $6 * 3 * 3 * 3 * 3 * 3 * 2 * 2 * 1 = 36 * 27 / 8 = 81 * 3 = 243 * 2 = 486 * 2 = 972 / 2 = 486 * 2 = 972 / 3 = 324 * 2 = 648 / 2 = 324 * 2 = 648 / 3 = 216 * 3 = 648 / 2 = 324 * 2 = 648 / 3 = 216 * 2 = 432 / 2 = 216$ but we made a mistake in our logic.</p> <p>The mistake we made is that we counted every possible arrangement of 3 blue cones in 4 possible places * 3 * 3 * 3 6 times but every 6 places * 3 * 3 * 3 places for the blue cones can also be arranged in $3! = 6$ ways. so every $6 * 3 * 3 * 3 * 3 * 3 * 2 * 2 * 1 = 36 * 27 / 8 = 81 * 3 = 243 * 2 = 486 * 2 = 972 / 2 = 486 * 2 = 972 / 3 = 324 * 2 = 648 / 2 = 324 * 2 = 648 / 3 = 216 * 3 = 648 / 2 = 324 * 2 = 648 / 3 = 216 * 2 = 432 / 2 = 216$ but we made a mistake in our logic.</p> <p>The mistake we made is that we counted every possible arrangement of 3 blue cones in 4 possible places * 3 * 3 * 3 6 times but every 6 places * 3 * 3 * 3 places for the blue cones can also be arranged in $3! = 6$ ways. so every $6 * 3 * 3 * 3 * 3 * 3 * 2 * 2 * 1 = 36 * 27 / 8 = 81 * 3 = 243 * 2 = 486 * 2 = 972 / 2 = 486 * 2 = 972 / 3 = 324 * 2 = 648 / 2 = 324 * 2 = 648 / 3 = 216 * 3 = 648 / 2 = 324 * 2 = 648 / 3 = 216 * 2 = 432 / 2 = 216$ but we made a mistake in our logic.</p> <p>The mistake we made is that we counted every possible arrangement of 3 blue cones in 4 possible places * 3 * 3 * 3 6 times but every 6 places * 3 * 3 * 3 places for the blue</p> |

Example 9

| | |
|----------------------------|---|
| Dataset | AQuA_RAT |
| Task-performing LLM | Ministral-8B |
| Question | <p>A rectangular piece of cloth 2 feet wide was cut lengthwise into two smaller rectangular pieces. The shorter piece was one-third of the length of the longer of the 2 new pieces and had an area of 12 square feet. What was the length Q in feet of the original piece of cloth before cutting?</p> <p>A)6 B)18 C)24 D)36 E)48</p> |
| Response | <p>1. Let (l) be the length of the longer piece and (w) be the width of the original piece. The shorter piece is $(\frac{1}{3}l)$.</p> <p>2. Given the width of the original piece is 2 feet, the shorter piece's dimensions are $(2 \times \frac{1}{3}l)$.</p> <p>3. The area of the shorter piece is given by: $2 \times \frac{1}{3}l = 12 \text{ square feet}$</p> <p>4. Solving for (l): $\frac{2l}{3} = 12 \implies l = 18$ feet</p> <p>5. The longer piece has a length $(l = 18)$ feet, and the shorter piece has a length $(\frac{1}{3}l = 6)$ feet.</p> <p>6. The original piece had a total length $(l + \frac{1}{3}l)$: $18 + 6 = 24$ feet. Therefore, the length (Q) of the original piece of cloth is 24 feet.</p> |
| Final Reward | -8.57 |
| Correctness | False |
| Reference Answer | C |

G. Additional Results

This section provides additional results of our experiments.

Table 9: Training ELHSR solely on logits can also yield high performance and can outperform many baselines with orders-of-magnitude higher efficiency.

| MATH Dataset | | | | | | | | | | |
|------------------|-----------------------------|------|------|-----------------------------|------|------|-----------------------------|------|------|------|
| Method | Llama-3.2-3B BoN@1: 39.0 | | | Llama-3.1-8B BoN@1: 47.2 | | | Ministral-8B BoN@1: 51.0 | | | Avg. |
| | @4 | @16 | @64 | @4 | @16 | @64 | @4 | @16 | @64 | |
| ELHSR on logits | 44.6 | 49.6 | 50.6 | 54.2 | 54.4 | 55.8 | 53.4 | 55.6 | 57.4 | 52.8 |
| GSM8K Dataset | | | | | | | | | | |
| Method | Llama-3.2-3B BoN@1: 39.0 | | | Llama-3.1-8B BoN@1: 47.2 | | | Ministral-8B BoN@1: 51.0 | | | Avg. |
| | @4 | @16 | @64 | @4 | @16 | @64 | @4 | @16 | @64 | |
| ELHSR on logits | 76.0 | 79.6 | 81.0 | 89.2 | 90.6 | 91.2 | 87.4 | 87.0 | 90.2 | 85.8 |
| AQuA_RAT Dataset | | | | | | | | | | |
| Method | Llama-3.2-3B BoN@1: 39.0 | | | Llama-3.1-8B BoN@1: 47.2 | | | Ministral-8B BoN@1: 51.0 | | | Avg. |
| | @4 | @16 | @64 | @4 | @16 | @64 | @4 | @16 | @64 | |
| ELHSR on logits | 58.8 | 63.4 | 65.2 | 68.6 | 77.2 | 78.0 | 69.6 | 74.4 | 74.0 | 69.9 |

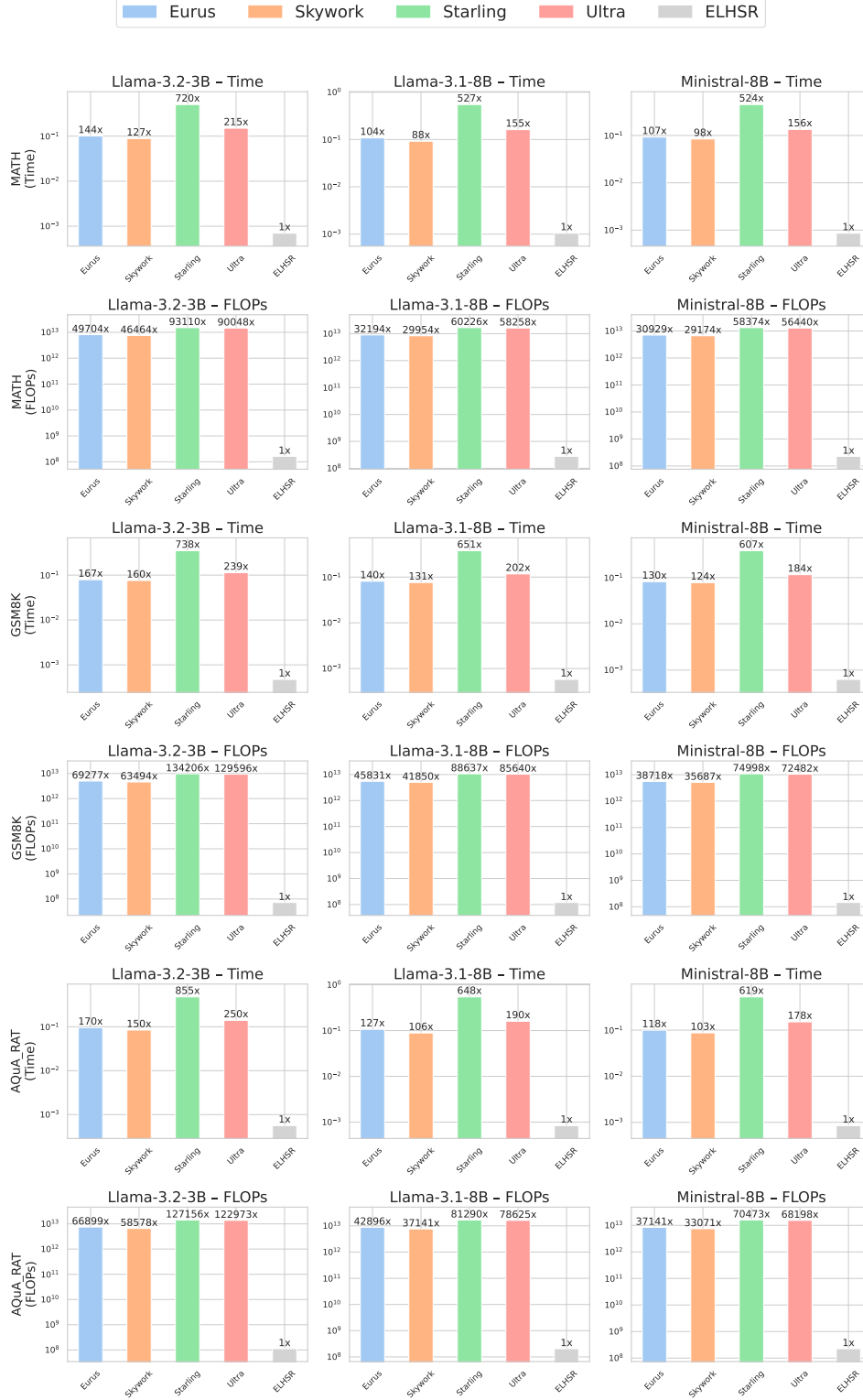


Figure 6: Comparison of average time and FLOPs per sample for the proposed ELHSR reward model and baseline reward models, evaluated on different datasets and task-performing models. The y-axis is plotted in log-scale. The results show that ELHSR achieves orders-of-magnitude higher efficiency than the baselines.

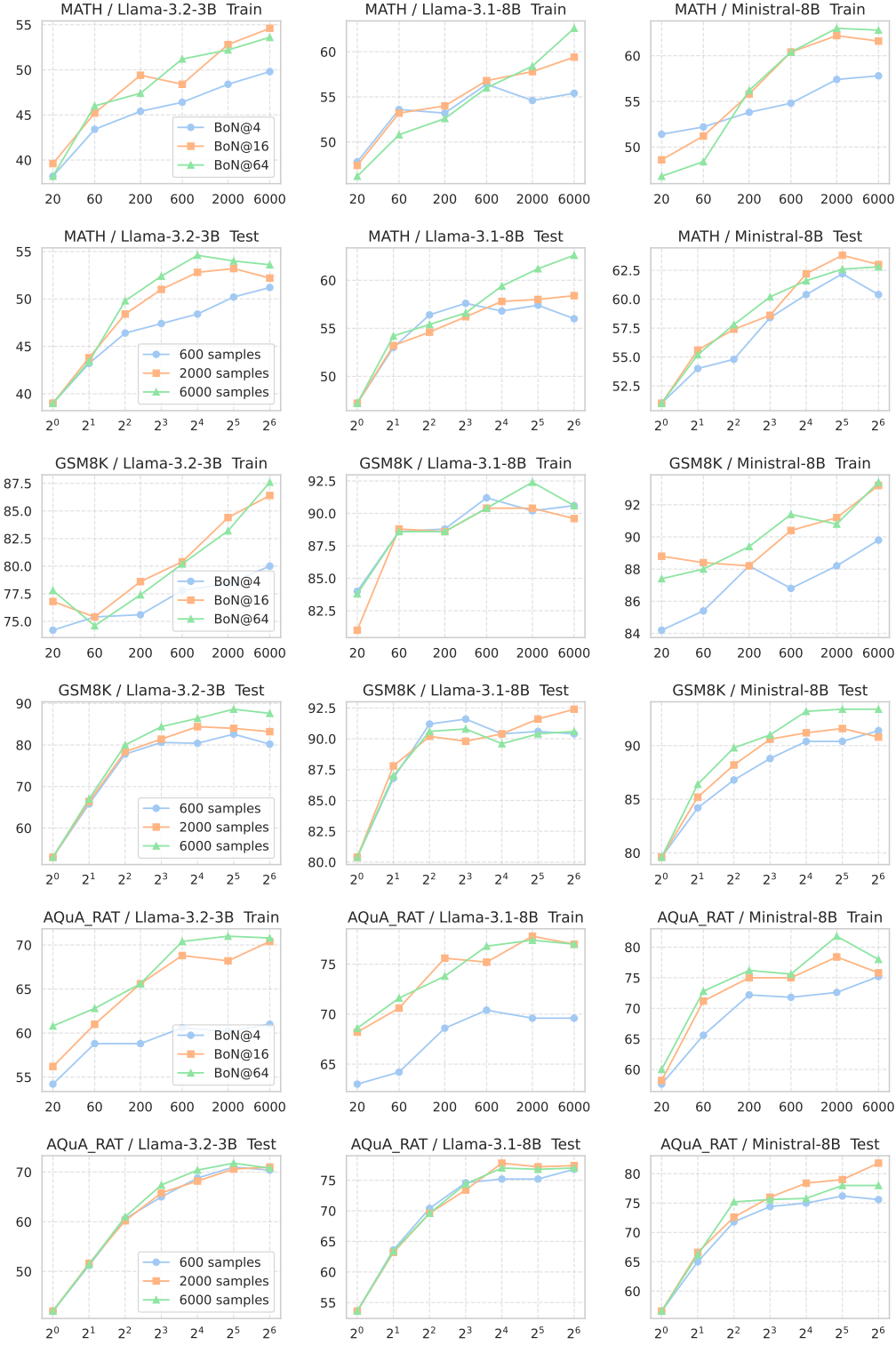


Figure 7: ELHSR has positive scaling with increased number of training samples and with the number of reasoning paths for inference.

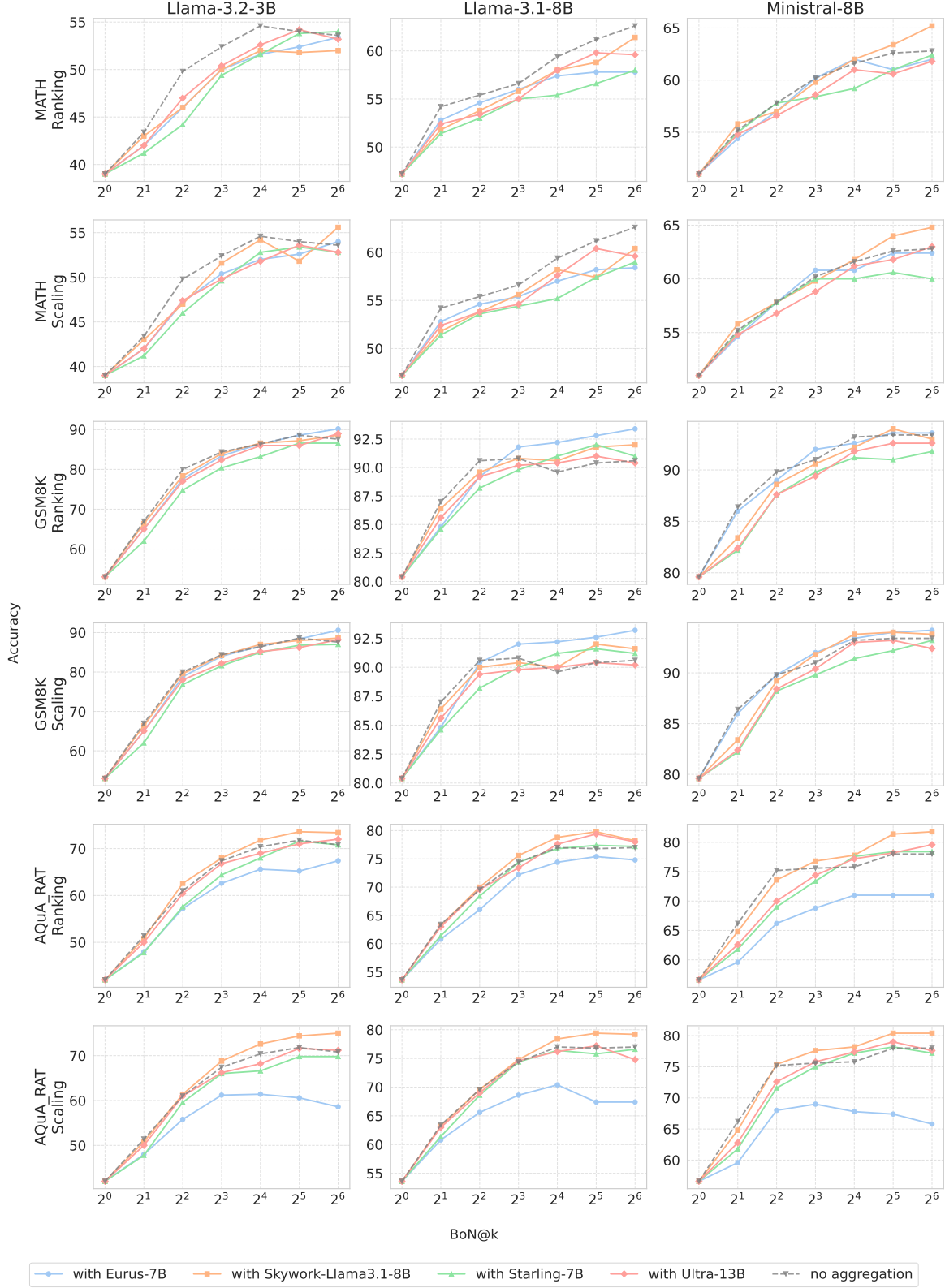


Figure 8: Performance of combined ELHSR and external reward models across different datasets and task-performing models, using both rank selection and scaled averaging.