# Can Reward Models Transfer Across Domains in Large Language Models?

**Changling Li** [*1] **David Dinucu-Jianu** [*1] **Michelle Chen** [*1] **Yuan Gao** [*1]

## Abstract

Reinforcement Learning from Human Feedback (RLHF) aligns language models with human preferences but is resource-intensive, often requiring separate reward models per domain. We investigate the transferability of reward models across domains in Large Language Models (LLMs) through fine-tuning. We investigate whether reward models can capture fundamental logic that transfers across tasks, enabling fine-tuning with minimal data. Additionally, we integrate an Active Learning framework to enhance data efficiency by by querying only the most uncertain samples. Our empirical investigations in mathematics and coding domains confirm the transferability of general purpose reward model and the potential of cross-domain adaptability. Our findings also reveal that the performance of cross-domain adaptation depends on the source and target domains hinting the asymmetric nature of the inherent characteristics of different domains.

## 1. Introduction

Reinforcement Learning from Human Feedback (RLHF) is a pivotal technique to align language models (LMs) with human preferences (Christiano et al., 2017), especially in training state-of-the-art models such as OpenAI's GPT-4 (Achiam et al., 2023) and Anthropic's Claude (Antropic, 2023). RLHF typically involves three stages: supervised fine-tuning (SFT) LM, training a reward model on preference data and optimizing the LM to maximize the reward. Reward model plays a crucial role in RLHF process. However, training an effective reward model remains challenging (Casper et al., 2023). It often requires substantial amounts of preference data, which can be abundant in some fields but scarce in others. On the other hand, separate reward models are trained for different tasks in RLHF, leading to inefficiencies in time and resource utilization.

While transferring reward models is difficult in general reinforcement learning due to differing state-action spaces (McKinney et al., 2023), RLHF for Large Language Models (LLMs) presents a potential: preference data across domains often shares a similar format involving human judgments on text. Successful reward model transfer can significantly reduce training time and improve data efficiency by minimizing the need for extensive preference data.

We investigate the transferability of reward models in RLHF, hypothesizing that reward models encode domain-independent logic and can be effectively fine-tuned in new domains. To validate our hypothesis, we conduct experiment in the mathematics and coding domains, comparing both fine-tuned general-purpose reward models and fine-tuned reward models from a different domain against ground truth reward models. Additionally, we incorporate margin-based active learning to further enhance data efficiency. Our findings demonstrate that fined-tuned reward models can match or even surpass the performance of ground-truth models with less training data, revealing the transferability of reward models and the potential to utilize data from other domains for better performance. On the other hand, the cross-domain transferability depends on the inherent characteristics of the source and target domain which is asymmetric. We provide all the finetuned models and datasets on our Huggingface and the code at https://github.com/RD211/eth-dl-rewards.

## 2. Related Work

**Reward transfer in RL:** Transfer learning in RL can enhance learning efficiency and performance by exploiting cross-task similarities (Feng et al., 2024; He et al., 2024). Reward transfer is one direction for transfer learning where reward function developed for one task is adapted to others. Methods like inverse reinforcement learning and imitation learning have been used for to obtain the reward model, but challenges arise due to differences in action and observation spaces across tasks (Metelli et al., 2021). Recent research addresses these issues by decoupling reward functions from task-specific dynamics (Luo et al., 2022; Yoo & Seo, 2022).

**Reward transfer in RLHF for LLMs:** Reward models in

[*]Equal contribution [1]Deep Learning, FS2024. Department of Computer Science, ETH Zürich.. Correspondence to: Changling Li <lichan@student.ethz.ch>, David Dinucu-Jianu <ddinucu@student.ethz.ch>, Michelle Chen <mcc@mailbox.org>, Yuan Gao <nouvaxs@gmail.com>.

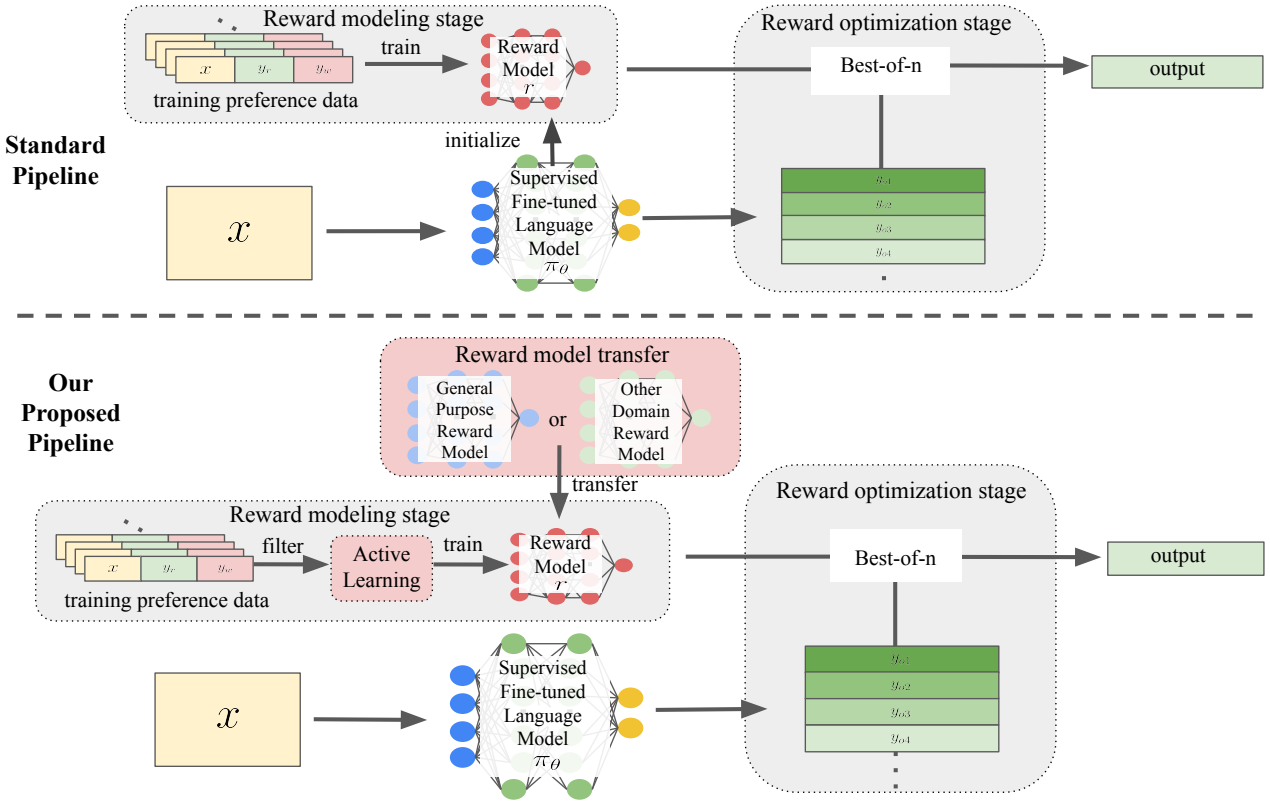



*Figure 1.* **Comparison between the standard reward modeling and optimization pipeline and our proposed pipeline.** The top shows the common practice where the reward model is initialized from the supervised fine-tuning model. The bottom shows our improved process where the reward model is transferred from other domains and then fine-tuned with filtered preference data using active learning.

RLHF often struggle to generalize across different scenarios and perform poorly on out-of-distribution data. To address this, recent studies have explored techniques such as contrastive learning to enhance model robustness (Wang et al., 2024) and tree-structured preference data to better capture complex human preferences (Qiu et al., 2024). Recent work (Szép et al., 2024) provides comprehensive insights into fine-tuning LLMs with limited data, emphasizing transfer learning techniques that mitigate overfitting while enhancing performance across domains. Our exploration differs from those works in that we investigate the transferability of reward models across domains through fine-tuning reward models. One work that is close to our objective is (Wu et al., 2024). However, they limit their scope in reward transferability across different languages without fine-tuning.

## 3. Method

Our explorations focus on the reward modeling (RM) and reward optimization of RLHF. The standard RM pipeline trains a model $r : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ as a proxy for human judgment of $y$ given $x$. It is initialized from the SFT model and trained using the preference data as shown in Figure 1.

Reward optimization adjusts the model outputs using the human feedback captured by RM. Common methods are reinforcement learning (RL) and best-of-n. RL changes the underlying model by optimizing the output while best-of-n is an inference-time procedure that reranks the generations sampled from SFT model using the reward model to decide the final output (Ouyang et al., 2022; Bai et al., 2022).

### 3.1. Reward Model Transfer

Different from the common practice of the RM stage, we propose to adapt a predefined reward model to the task domain as shown on the bottom part of Figure 1. We hypothesize that reward models encode domain-independent logic and can be effectively fine-tuned in new domains using much less data. The predefined reward model can be a general purpose reward model or a domain specific reward model from other fields. It is then fine-tuned using the preference data from the task domain. Particularly, we focus on pairwise feedback to train the reward model. Pairwise feedback chooses a better generation out of two. The reward model is obtained by maximizing the following objective using the preference data pairs

$$\mathbb{E}_{(x,y_r,y_w)\sim D^{RM}}[\log \sigma(r(x,y_r) - r(x,y_w))] \quad (1)$$

where $y_r$ is the chosen answer and $y_w$ denotes the other.

### 3.2. Active Learning

Additionally, we propose to incorporate active learning (AL) for data selection to further improve data efficiency. It has been shown that active learning is a strong baseline for data subset selection (Park et al., 2022). Let $\mathcal{D} = \{x_i, y_r, y_w\}$ be the given preference dataset. AL firstly chooses a small subset of training data $\mathcal{L}_0$ and then train on the subset. This process repeats for $t$ rounds until attaining the target number of samples by adding new samples to the selected set at each round, i.e. $\mathcal{L} \leftarrow \mathcal{L} \cup \mathcal{L}_t$. Margin-based AL is particularly suitable due to the interdependencies of the answer pair (Roth & Small, 2006). Margin-based AL uses a querying function dependent on the margin function $\rho(x, y, r)$. Since we use pairwise feedback, the margin function is naturally defined as the reward difference between answer pair:

$$\rho(x, y, r) = |r(x, y_r) - r(x, y_w)| \quad (2)$$

which measures the uncertainty of the reward model in judging the difference between the pair of responses. The corresponding querying function is then defined as

$$\mathcal{Q} : x_* = argmin_{x\in\mathcal{D}}[|r(x, y_r) - r(x, y_w)|] \quad (3)$$

The detailed margin-based AL algorithm is shown in Appendix B.

## 4. Experiments

**Experiment settings** We consider the domains of mathematics and coding for their wide interest and application. For each domain, we curate separate datasets for training and evaluation. The detailed information of each dataset, preprocessing and the procedure of generating preference data are included in Appendix A.

We choose the InternLM2-7B reward model (Cai et al., 2024) as our general-purpose reward model. The ground truth reward models are initialized from the InternLM2-7B base LLM fine-tuned on domain data and trained on the full training set. We note that the data for SFT has no overlap with the data for fine-tuning reward models. The training process is detailed in Appendix B. The fine-tuned reward models are compared with two baselines:

- **Direct reward transfer** applies the reward model to a new domain without any fine-tuning.
- **Ground-truth reward model** is trained specifically on each domain's dataset from scratch, serving as an ideal benchmark for performance.

Performance is measured by deploying the reward model on the evaluation set to distinguish the preferred from undesired answer serving as a proxy for the best-of-n method.

**Evaluation metric** We employ two metrics to assess the transferability of reward models. *Judgment accuracy* evaluates the model's ability to select the chosen answer from the evaluation set. Denoting the number of evaluation data as $N_E$ and the accurately distinguished data as $N_A$, we define judgment accuracy as

$$S_{\mathcal{JA}} = \frac{N_A}{N_E} \quad (4)$$

*Transferability* measures the effectiveness of transferring the reward model across domains by determining the reduction in training data. We denote the required number of fine-tuning data as $N_F$ and the number of full training data as $N_T$. The transferability is defined as

$$S_{Transferability} = 1 - \frac{N_F}{N_T} \quad (5)$$

### 4.1. Reward model transferability

We denote the ground truth reward model as $R_{truth}$, the fine-tuned general purpose reward model as $R_{gen}$, the fine-tuned math model as $R_{math}$ and the fine-tuned code model as $R_{code}$. The comparisons between the fine-tuned models and the ground-truth models are shown in Table 1.

**General purpose reward model** achieves a comparable performance in math and a noticeably better performance in code with an increase of $0.0157$ in $S_{\mathcal{JA}}$ with minimal data of 20k, giving a transferability score of the $R_{gen}$ at $0.6667$. For math, $R_{gen}$ obtains best performance when fine-tuned on the full training set (60k) with an increase of $0.0228$ in $S_{\mathcal{JA}}$. For code, however, more data does not give the best performance. With the full training set, $R_{gen}$ performs slightly worse than $R_{gen}$ fine-tuned on 20k data which may be caused by low quality data and forgetting.

**Domain-specific reward models** show similar performance trend. While the fine-tuned math reward model on code domain shows better performance than $R_{truth}$ of code, the reverse transfer does not yield similar benefits. With 20k code data, $R_{math}$ achieves a comparable performance to the code $R_{truth}$. The best performance is obtained by fine-tuning math reward model with 60k code data which gives $S_{\mathcal{JA}} = 0.7905$. For math, the performance of $R_{code}$ even though increases with more data but only achieves a comparable performance to the math $R_{truth}$.

Overall, the comparison highlights the transferability of the general purpose reward model and the potential for cross-domain adaptability. However, the asymmetric nature of transferability across different domains indicates the effectiveness of such transfers may depend on the inherent characteristics of the source and target domains.

| Fine-tuning general purpose reward model | | | | | |
|---|---|---|---|---|---|
| Domain | $R_{truth}$ | $R_{gen}$ (0k) | $R_{gen}$ (20k) | $R_{gen}$ (40k) | $R_{gen}$ (60k) |
| math | 0.8681 | 0.7346 | 0.8786 | 0.8853 | **0.8909** |
| code | 0.7843 | 0.6519 | **0.8000** | 0.7907 | 0.7970 |
| Fine-tuning math reward model on code dataset | | | | | |
| Domain | $R_{truth}$ | $R_{math}$ (0k) | $R_{math}$ (20k) | $R_{math}$ (40k) | $R_{math}$ (60k) |
| code | 0.7843 | 0.5668 | 0.7894 | 0.7894 | **0.7905** |
| Fine-tuning code reward model on math dataset | | | | | |
| Domain | $R_{truth}$ | $R_{code}$ (0k) | $R_{code}$ (20k) | $R_{code}$ (40k) | $R_{code}$ (60k) |
| math | **0.8681** | 0.6883 | 0.8358 | 0.8588 | 0.8613 |

*Table 1.* **Comparison of the judgment accuracy across different reward models and different amount of fine-tuning data for code and math.** The number in the brackets indicates the amount of training data used to train the reward model. A higher score indicates better performance. The highest score for each scenario is highlighted in bold front.

| Active learning for fine-tuning general purpose reward model | | | | |
|---|---|---|---|---|
| Domain | $R_{truth}$ | $R_{gen}$ (2.5k) | $R_{gen}$ (5k) | $R_{gen}$ (7.5k) |
| math | 0.8700 | 0.8560(0.8577) | 0.8643(**0.8707**) | 0.8615(0.8692) |
| code | 0.7746 | **0.8012**(0.7901) | 0.7930(0.7855) | 0.7951(0.7953) |

*Table 2.* **Comparison of the judgment accuracy between the ground truth reward models and the fine-tuned general purpose reward models using margin-based AL.** The numbers in the brackets in the header indicate the amount of fine-tuning data selected using margin-based AL. The scores in the brackets are the performance obtained by fine-tuning on the same amount of data using random selection. A higher score indicates better performance. The highest score for each scenario is highlighted in bold front.

### 4.2. Active learning for efficient fine-tuning

We incorporate margin-based AL in our pipeline to further improve the data efficiency. We focus on fine-tuning general purpose reward model since it gives best performances on both domains in fine-tuning. The performance comparisons are shown in Table 2. Fine-tuning with active learning reduce tuning data to 2.5k while achieving better results with $S_{\mathcal{JA}} = 0.8012$ in code domain. This gives a transferability score at $0.9583$. It achieves better performance than random selection for both 2.5k and 5k data. With 7k data, it shows similar performance to random selection which is likely due to the performance saturation. The same technique, however, does not improve the performance for math, achieving slightly worse results than ground truth model and fine-tuned model with randomly selected data. We further explored the potential cause by looking into the selected data by AL, finding that margin-based AL tends to select data with answer pairs that are both correct but the reject answer has the wrong format. We show an example in Appendix C. We expect that a higher quality preference dataset can showcase the advantages of our proposed pipeline.

## 5. Conclusion

We empirically investigated whether reward models in RLHF capture fundamental reasoning principles that can transfer across domains. By fine-tuning both general purpose and domain-specific reward models on new domain data, we found consistently comparable or improved performance in judgment accuracy and data efficiency over standard RLHF reward model training. Notably, fine-tuned general purpose reward models outperformed ground truth models in both mathematics and coding with less data. Furthermore, a math reward model fine-tuned on code data surpassed the code ground-truth model, highlighting the potential of cross-domain knowledge transfer. However, the reverse transfer did not yield similar gains, indicating asymmetries in domain characteristics. We also integrated margin-based AL to enhance data efficiency, which produced notable gains in coding but was less effective in math, likely due to limitations in the preference data.

**Limitation** Due to computational constraints, our experiments were limited to mathematics and coding. Future research could investigate additional domains to more comprehensively assess reward-model transferability and asymmetry of knowledge. Our approach employed LoRA-based fine-tuning and relied on a best-of-n strategy for reward optimization; it would be interesting to explore alternative methods such as RL-based optimization. Finally, while our margin-based active learning strategy showed promise, data quality issues in the preference annotations limited its effectiveness. Future work can focus on constructing higher-quality preference datasets and further exploring the benefits of active learning in the RLHF pipeline.

## 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.

Antropic. Introducing claude. 2023. URL https://www.anthropic.com/news/introducing-claude.

Bai, Y., Jones, A., Ndousse, K., Askell, A., Chen, A., DasSarma, N., Drain, D., Fort, S., Ganguli, D., Henighan, T., et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

Cai, Z., Cao, M., Chen, H., Chen, K., Chen, K., Chen, X., Chen, X., Chen, Z., Chen, Z., Chu, P., Dong, X., Duan, H., Fan, Q., Fei, Z., Gao, Y., Ge, J., Gu, C., Gu, Y., Gui, T., Guo, A., Guo, Q., He, C., Hu, Y., Huang, T., Jiang, T., Jiao, P., Jin, Z., Lei, Z., Li, J., Li, J., Li, L., Li, S., Li, W., Li, Y., Liu, H., Liu, J., Hong, J., Liu, K., Liu, K., Liu, X., Lv, C., Lv, H., Lv, K., Ma, L., Ma, R., Ma, Z., Ning, W., Ouyang, L., Qiu, J., Qu, Y., Shang, F., Shao, Y., Song, D., Song, Z., Sui, Z., Sun, P., Sun, Y., Tang, H., Wang, B., Wang, G., Wang, J., Wang, J., Wang, R., Wang, Y., Wang, Z., Wei, X., Weng, Q., Wu, F., Xiong, Y., Xu, C., Xu, R., Yan, H., Yan, Y., Yang, X., Ye, H., Ying, H., Yu, J., Yu, J., Zang, Y., Zhang, C., Zhang, L., Zhang, P., Zhang, P., Zhang, R., Zhang, S., Zhang, S., Zhang, W., Zhang, W., Zhang, X., Zhang, X., Zhao, H., Zhao, Q., Zhao, X., Zhou, F., Zhou, Z., Zhuo, J., Zou, Y., Qiu, X., Qiao, Y., and Lin, D. Internlm2 technical report, 2024.

Casper, S., Davies, X., Shi, C., Gilbert, T. K., Scheurer, J., Rando, J., Freedman, R., Korbak, T., Lindner, D., Freire, P., et al. Open problems and fundamental limitations of reinforcement learning from human feedback. *arXiv preprint arXiv:2307.15217*, 2023.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Feng, J., Chen, M., Pu, Z., Qiu, T., and Yi, J. Efficient multi-task reinforcement learning via task-specific action correction. *arXiv preprint arXiv:2404.05950*, 2024.

He, J., Li, K., Zang, Y., Fu, H., Fu, Q., Xing, J., and Cheng, J. Efficient multi-task reinforcement learning with cross-task policy guidance. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Hendrycks, D., Basart, S., Kadavath, S., Mazeika, M., Arora, A., Guo, E., Burns, C., Puranik, S., He, H., Song, D., et al. Measuring coding challenge competence with apps. *arXiv preprint arXiv:2105.09938*, 2021.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

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.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

LI, J., Beeching, E., Tunstall, L., Lipkin, B., Soletskyi, R., Huang, S. C., Rasul, K., Yu, L., Jiang, A., Shen, Z., Qin, Z., Dong, B., Zhou, L., Fleureau, Y., Lample, G., and Polu, S. Numinamath. [https://github.com/project-numina/aimo-progress-prize](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf), 2024.

Luo, F.-M., Cao, X., Qin, R.-J., and Yu, Y. Transferable reward learning by dynamics-agnostic discriminator ensemble. *arXiv preprint arXiv:2206.00238*, 2022.

McKinney, L., Duan, Y., Krueger, D., and Gleave, A. On the fragility of learned reward functions. *arXiv preprint arXiv:2301.03652*, 2023.

Metelli, A. M., Ramponi, G., Concetti, A., and Restelli, M. Provably efficient learning of transferable rewards. In *International Conference on Machine Learning*, pp. 7665–7676. PMLR, 2021.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Park, D., Papailiopoulos, D., and Lee, K. Active learning is a strong baseline for data subset selection. In *Has it Trained Yet? NeurIPS 2022 Workshop*, 2022.

Qiu, T., Zeng, F., Ji, J., Yan, D., Wang, K., Zhou, J., Han, Y., Dai, J., Pan, X., and Yang, Y. Reward generalization in rlhf: A topological perspective. *arXiv preprint arXiv:2402.10184*, 2024.

Roth, D. and Small, K. Margin-based active learning for structured output spaces. In *Machine Learning: ECML 2006: 17th European Conference on Machine Learning Berlin, Germany, September 18-22, 2006 Proceedings 17*, pp. 413–424. Springer, 2006.

Szép, M., Rueckert, D., von Eisenhart-Rothe, R., and Hinterwimmer, F. A practical guide to fine-tuning language models with limited data, 2024. URL https://arxiv.org/abs/2411.09539.

Wang, B., Zheng, R., Chen, L., Liu, Y., Dou, S., Huang, C., Shen, W., Jin, S., Zhou, E., Shi, C., et al. Secrets of rlhf in large language models part ii: Reward modeling. *arXiv preprint arXiv:2401.06080*, 2024.

Wu, Z., Balashankar, A., Kim, Y., Eisenstein, J., and Beirami, A. Reuse your rewards: Reward model transfer for zero-shot cross-lingual alignment. *arXiv preprint arXiv:2404.12318*, 2024.

Yoo, S.-W. and Seo, S.-W. Learning multi-task transferable rewards via variational inverse reinforcement learning. In *2022 International Conference on Robotics and Automation (ICRA)*, pp. 434–440. IEEE, 2022.

# A. Details of Dataset, Preprocessing and Preference Data Generation

We used the NuminaMath-CoT dataset (LI et al., 2024) and the CodeParrot/APPS dataset (Hendrycks et al., 2021) for our evaluation of reward model transferability. NuminaMath-CoT dataset contains 859,608 diverse mathematics problems spanning a wide range of categories. The CodeParrot/APPS dataset comprises 10,000 coding problems extracted from competitive programming platforms like Codeforces and technical interviews. We note that both datasets do not preference data and we generate the preference data pairs ourselves. Below, we present the data preprocessing details and the unified preference data generation process.

## A.1. Data preprocessing

**Selection criteria:** For easy evaluation of correct response generation, we firstly select data with responses that include easy-to-compare answers such as numbers or letters. In addition, the answers need to be wrapped in an indicator such as \boxed{}. An example of the data is shown in Table 3.

**Dataset curation:** Due to the limitation of both compute and budget, we use subsets of the whole dataset. For both math and code datasets, we set 60k as our standard for full training set and 20k and 40k as our fine-tuning sets which are subsets of the training set. In addition, we set a separate set of 9k data as our evaluation set. We use random selection with shuffling to ensure unbiased data selection. We note that the code dataset does not contain enough data. We expand the code dataset by generating diverse preference pairs.

## A.2. Preference Data Generation

The preference data generation process involves creating pairs of candidate solutions to form training data for the reward model. We generate a set of candidate solutions using the Llama-3.1-8b-Instruct model (Dubey et al., 2024) for each problem in the dataset. This model was chosen due to its strong performance and common usage. The temperature for generation was set to 0.9 for to promote diversity in the generated responses. For mathematics problems, 8 candidate solutions are generated, and pairs are formed between correct and incorrect solutions, where correctness is determined using the `latex2sympy` equivalence check. For coding problems, 16 candidate solutions are generated. Pairs are formed not only between solutions that pass all test cases and those that fail, but also between solutions that pass more or fewer test cases to capture varying levels of correctness. Below we present an example of the preference data from each domain in Table 3 and 4.

The generation of preference data pairs is computationally intensive. To accelerate this process, we utilized the vLLM inference acceleration library, which enables efficient batching of responses (Kwon et al., 2023). By leveraging vLLM, we significantly reduced the time required for generation while maintaining high-quality candidate solutions across both domains.

| Math preference data | | |
|---|---|---|
| Question | Accept | Reject |
| What is the value of $x^6 - 3x$ when $x = 3$? | To find the value of $x^6 - 3x$ when $x = 3$, we will substitute $x$ with 3 in the expression. $x^6-3x = (3)^6-3(3) = 729 - 9 = 720$ So, the value of $x^6-3x$ when $x = 3$ is \boxed{720}. | To solve this problem, we will substitute the value of x into the expression given. Given expression: $x^6 - 3x$ Value of x: 3 Substituting x = 3 into the expression, we get: $3^6 - 3(3)$ Calculate the exponent first: $3^6 = 3 * 3 * 3 * 3 * 3 * 3$ $3^6 = 729$ Now multiply the result by 3: $729 * 3 = 2187$ Now subtract 3(3), which is 9: $2187-9$ $2187-9 = 2178$ So the value of the expression $x^6-3x$ when $x = 3$ is: \boxed{2178}. |

*Table 3.* Math and code preference data examples with reject responses generated using our preference data generation pipeline.

| Code preference data | | |
|---|---|---|
| Question | Accept | Reject |
| You are given a string S. Each character of S is uppercase or lowercase English letter. Determine if S satisfies all of the following conditions: - The initial character of S is an uppercase A. - There is exactly one occurrence of C between the third character from the beginning and the second to last character (inclusive). - All letters except the A and C mentioned above are lowercase. ——Constraints—— - 4 —S— 10 (—S— is the length of the string S.) - Each character of S is uppercase or lowercase English letter. ——Input—— Input is given from Standard Input in the following format: S ——Output—— If S satisfies all of the conditions in the problem statement, print AC; otherwise, print WA. ——Sample Input—— AtCoder ——Sample Output—— AC The first letter is A, the third letter is C and the remaining letters are all lowercase, so all the conditions are satisfied. | ```python Take the string input from the user s = input() Check if the first character is uppercase 'A' if s[0] != 'A': print('WA') else: Find the index of 'C' c_index = s.find('C') Check if the index of 'C' is within the correct range (3 to second last character) if not 2 < c_index < len(s) - 1: print('WA') else: Check if all other characters are lowercase if all('a' <= char <= 'z' for i, char in enumerate(s) if i != 0 and i != c_index + 1): print('AC') else: print('WA') ```This code checks each condition in the problem statement and prints 'AC' if all conditions are met, and 'WA' otherwise. | ```python s = input() if len(s) < 4 or len(s) > 10: print("WA") elif s[0] != 'A': print("WA") elif s[1:4].count('C') != 1 or s[:3].islower() or s[3:-2].isupper(): print("WA") else: print("AC") ``` |

*Table 4.* Code preference data examples with reject responses generated using our preference data generation pipeline.

## B. Active Learning and Training Details

In this section, we present our detailed implementation of margin-based active learning algorithm and training details.

### B.1. Margin-based active learning

Our implementation of the margin-based active learning is shown as Algorithm 1. Margin-based active learning selects question data with pairs of the smallest margin (highest uncertainty) as the training subset. This iterative process ensures that the model focuses on the most informative samples, improving its performance while minimizing required fine-tuning data. To avoid overfitting, we employ softmax sampling method to sample the data according to the probability defined in Equation 6.

For each iteration, we set $n = 2.5$k and repeat for three iterations.

### B.2. Training

All experiments were conducted using an H200 GPU from `Vast.ai`. The reward models are trained using the `TRL` library with `LoRA` (Low-Rank Adaptation) to improve parameter efficiency (Hu et al., 2021). The specific parameters for LoRA were:

- `rank: 32`
- `alpha: 64`
- `dropout: 0.01`

Leveraging LoRA reduces the required training parameters from 7B of the full model to only around 75M.

**Algorithm 1** Active Learning for Reward Model Fine-Tuning (Sampling with Softmax)

**Require:** Initial labeled dataset $\mathcal{L}$, unlabeled dataset $\mathcal{U}$, reward model $r$, query budget $B$, temperature $T$

**Ensure:** Fine-tuned reward model $r$

1: Train reward model $r$ on updated $\mathcal{L}$

2: UncertaintyScores[$x$] $\leftarrow \emptyset$

**while** $|\mathcal{L}| < B$ **do**

  **for all** $x \in \mathcal{U}$ **do**

    1: Compute uncertainty $\rho(x, y, r) = -|r(x, y_r) - r(x, y_w)|$

    2: UncertaintyScores[$x$] $\leftarrow \rho(x, r)$

  **end for**

  1: Normalize uncertainties with softmax:

$$P(x) = \frac{\exp(\rho(x, y, r)/T)}{\sum_{x' \in \mathcal{U}} \exp(\rho(x', y, r)/T)} \tag{6}$$

  2: Sample $n$ examples $\mathcal{Q}$ from $\mathcal{U}$ according to $P(x)$

  3: $\mathcal{L} \leftarrow \mathcal{L} \cup \mathcal{Q}$

  4: $\mathcal{U} \leftarrow \mathcal{U} \setminus \mathcal{Q}$

  5: Retrain reward model $r$ on updated $\mathcal{L}$

**return** Updated labeled dataset $\mathcal{L}$, fine-tuned reward model $r$

**Learning rate and scheduler** are controlled different for fine-tuning and active learning iterations. For fine-tuning, we used a cosine learning rate scheduler with an initial learning rate of 0.00005. For active learning, a constant learning rate was used to avoid destabilizing the model when starting a new iteration.

**Batch size** was set to 32 during training and all experiments were done with one epoch.

## C. Results

### C.1. Reward model transferability

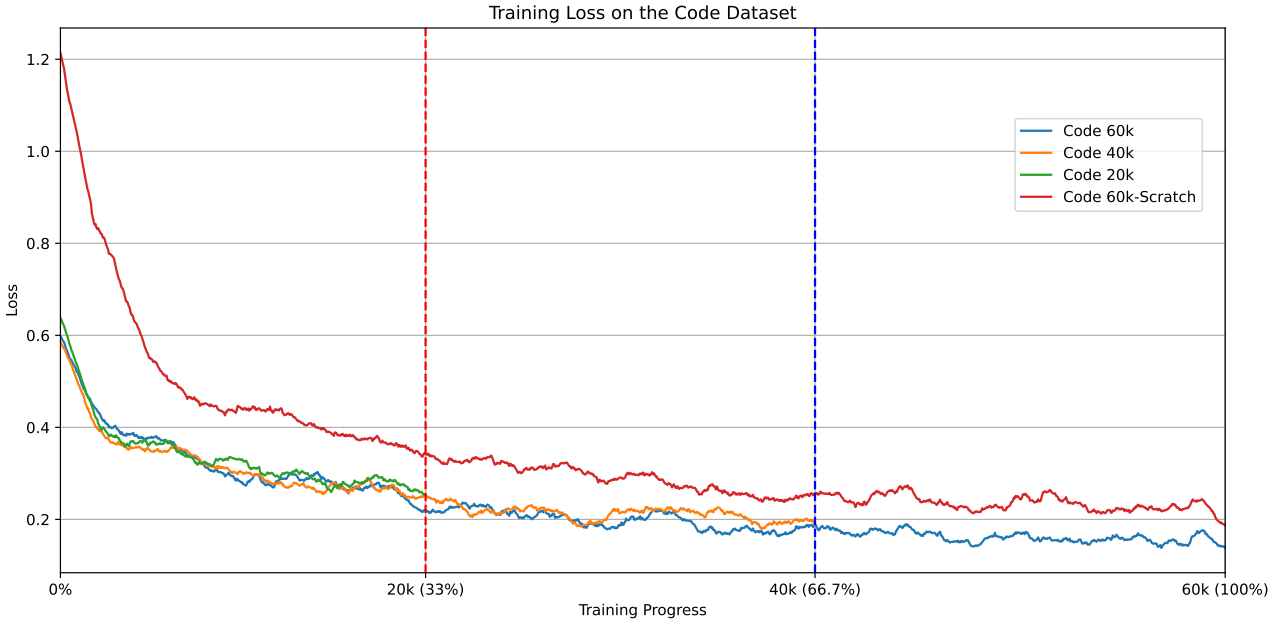
As an additional performance comparison between the fine-tuned reward models and the ground truth models, we include the training curves as shown in Figure 2 and Figure 3.

### C.2. Active learning for efficient fine-tuning

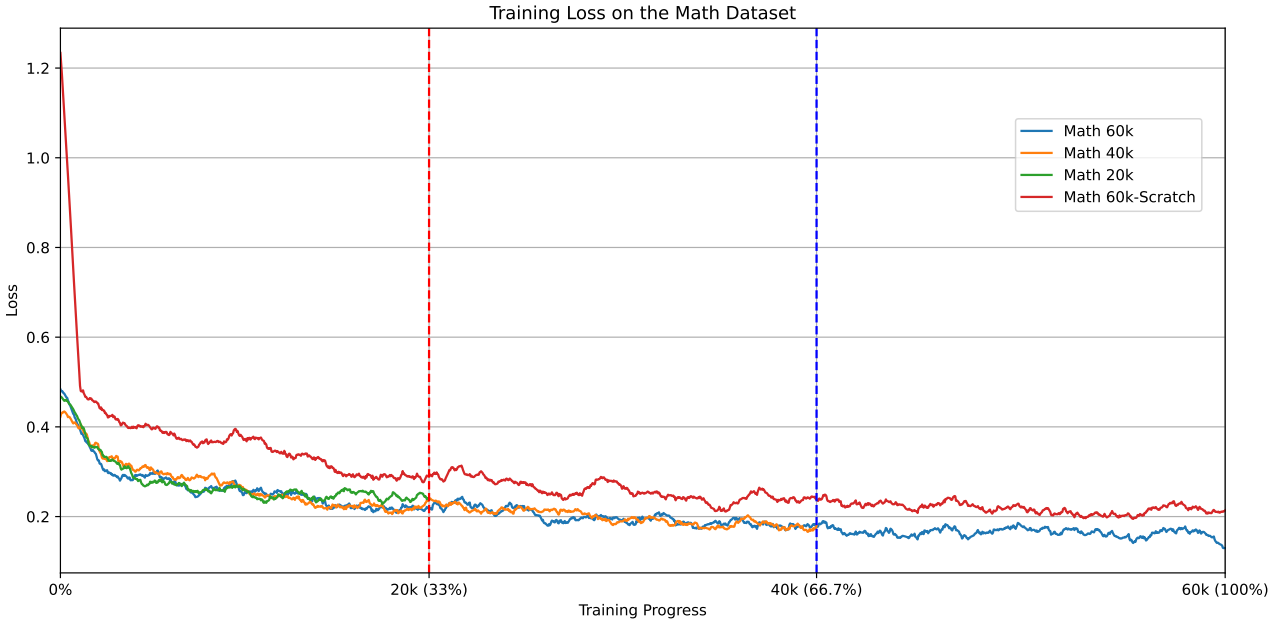
The results we obtained in active learning are unexpected particularly for the domain of math. We investigated the selected data by active learning to explore the potential cause. We found that margin-based active learning tends to select data with response pairs that are both correct but the reject answer has a wrong format because those answers are difficult to distinguish and thus have smaller margin (bigger uncertainty). One example of selected data in this scenario is shown in Table 5. Compared to the math data example in Table 3 which has better reject data, this preference pair can potentially cause more confusion. The generation of this kind of data is due to the limitation of our preference data generation process. We expect that a higher quality math preference dataset can showcase the advantages of our proposed pipeline with active learning. However, obtaining such dataset requires much higher budget involving using better LLMs or human annotators.

| Question | Accept | Reject |
|---|---|---|
| Jasmine, her older sister Alexa, their friend Clara and their cousin Molly went to the store to buy shoes. Alexa's shoes are twice as big as Jasmine's shoes, Clara's shoes are three times the size of Jasmine's shoes, and Molly's shoes are one and a half times the size of Jasmine's shoes, but Molly also bought a pair of sandals that were half a size less than her shoe size. If Jasmine's shoe size is 7, what is the combined shoe size for all the shoes they bought? | To find the combined shoe size, we first need to calculate each person's shoe size based on Jasmine's shoe size of 7. Alexa's shoe size is twice Jasmine's shoe size: 2 * 7 = 14 Clara's shoe size is three times Jasmine's shoe size: 3 * 7 = 21 Molly's shoe size is one and a half times Jasmine's shoe size: 1.5 * 7 = 10.5 For the sandals, Molly's shoe size is half a size less than her shoe size: 10.5 - 0.5 = 10 Now we need to add up the shoe sizes of all the individuals: - Jasmine's: 7 - Alexa's: 14 - Clara's: 21 - Molly's shoes: 10.5 - Molly's sandals: 10 The combined shoe size is: 7 + 14 + 21 + 10.5 + 10 = 62.5 \boxed{62.5} | To find the combined shoe size for all the shoes they bought, we need to find the sizes of all their shoes and add them together. Jasmine's shoe size is 7. Alexa's shoes are twice as big as Jasmine's shoes, so Alexa's shoe size is 7 * 2 = 14. Clara's shoes are three times the size of Jasmine's shoes, so Clara's shoe size is 7 * 3 = 21. Molly's shoes are one and a half times the size of Jasmine's shoes, but since Molly also bought a pair of sandals that were half a size less than her shoe size, we first need to find Molly's shoe size: Molly's shoe size is (7 * 1.5) - (7 * 1.5 * 0.5) = 10.5. Now, let's not forget the sandal size. Sandal size is 10.5 - 0.5 = 10. Molly's total shoe size is 10.5 + 10 = 20.5 Now we need to add all the shoe sizes together: 7 + 14 + 21 + 20.5 = 62.5 |

*Table 5.* Example of active learning selected math preference data.

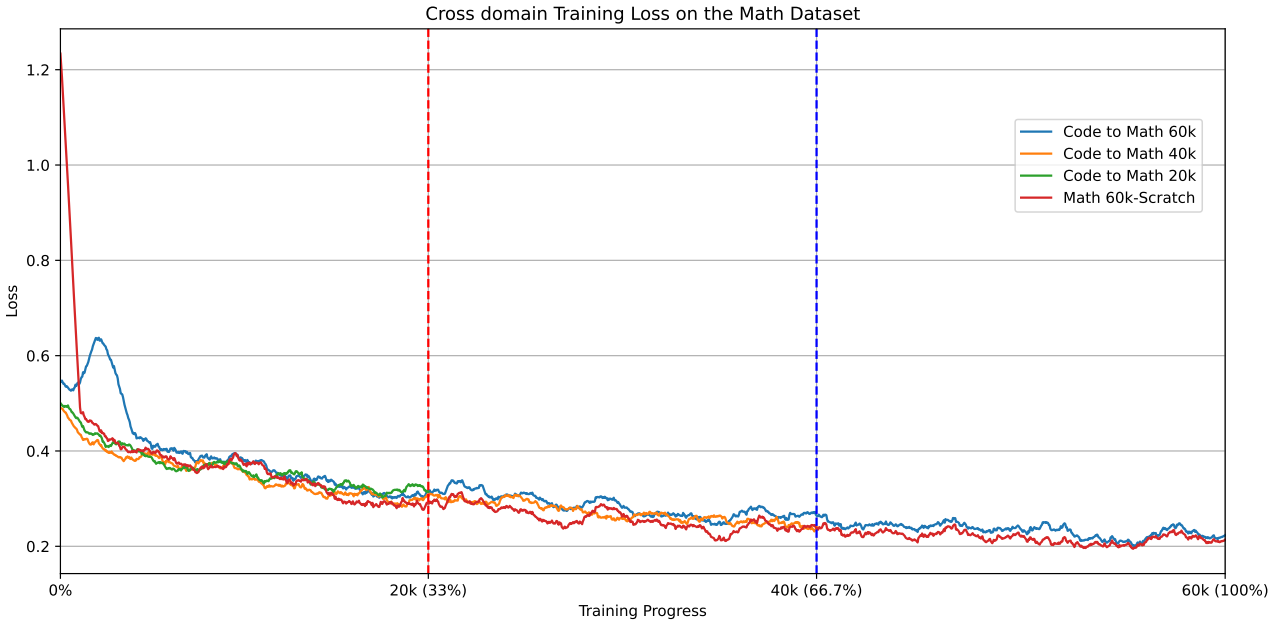



(a) The training loss curves on the code dataset.

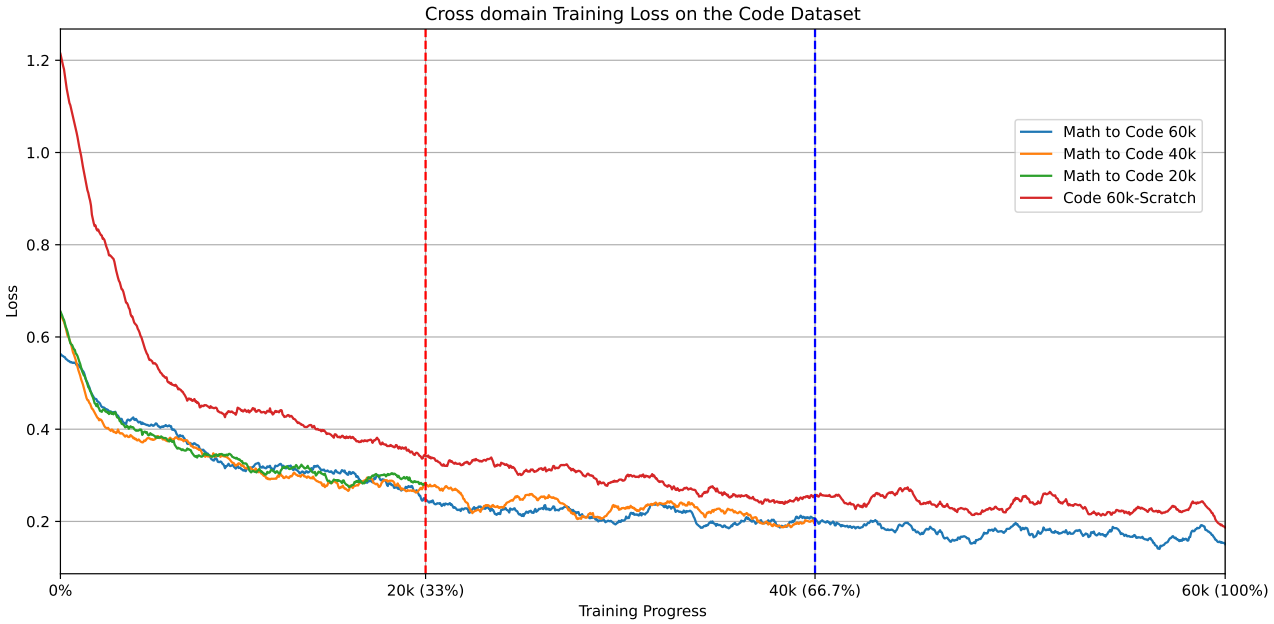



(b) The training loss curves on the math dataset.

*Figure 2.* This figure shows the training loss curves for (A) the code dataset and (B) the math dataset. Both datasets are evaluated using three training dataset sizes: 20k, 40k, and 60k data points. Models initialized from a general reward model are compared against a base LLM trained on the full 60k dataset (our ground truth reward model). For both datasets we can see that the models trained from scratch on the 60k data perform substantially worse than the models initialized from a base reward model.




(a) The training loss curves for fine-tuning code reward model on math data.




(b) The training loss curves for fine-tuning math reward model on code data.

*Figure 3.* This figure shows the training loss curves for cross-domain fine-tuning: (A) code-to-math and (B) math-to-code. For these fine-tunings, the models are initialized from the base LLM trained on the full 60k dataset of code or math, rather than a general reward model. We compare against the base LLM trained on the full 60k dataset (the ground truth reward models) for the two tasks and we discover that for the (B) math-to-code fine-tuning the performance of the cross-domain models is noticeable while it's less different on the (A) code-to-math scenario.