DON'T RELEARN WHAT YOU ALREADY KNOW: USING RESIDUAL REWARD LEARNING FOR RLHF

Anonymous authors
Paper under double-blind review

ABSTRACT

Human preferences are expensive, yet experts can often specify some of the components of a desired reward programmatically. Recent work has shown that such proxy rewards can be combined with preference learning by learning only their residual. But why does residual reward learning outperform simply adding a proxy to a conventionally learned reward model, and when should we expect it to do so?

We identify two mechanisms. First, residual learning preserves the decomposition of the true reward, whereas naive addition double-counts components already captured by the proxy. Second, when the proxy explains sufficient variation in the true reward, the residual constitutes a lower-variance and therefore simpler learning target. We theoretically characterize these conditions and test them through controlled experiments across eight RL environments. Finally, we demonstrate that the same principle extends to LLM alignment by combining a programmatic proxy with human preferences for summarization. Our results characterize when prior reward knowledge can reduce the burden placed on preference learning.

1 INTRODUCTION

Reward learning is often framed as a choice between specifying what we want and learning it from human feedback. In practice, however, we may be able to specify *part* of what we want. A summarization system, for example, can cheaply measure properties such as length, repetition, or formatting, even though no such heuristic captures summary quality completely. The question is then not whether to use a programmed reward or human preferences, but how to combine the information contained in both.

A common approach is to independently learn a reward model from preferences and add its predictions to a programmatic proxy reward (Mu et al., 2024; Tao et al., 2025; Liao et al., 2025). An alternative is *residual reward learning*: given a proxy $f^{\sim}$, learn only the missing component f^{Δ} such that $f^* = f^{\sim} + f^{\Delta}$. This approach was recently introduced by Cao et al. (2026), who demonstrated improved feedback efficiency in preference-based reinforcement learning.

In this work, we ask a complementary question: *why does learning the residual help, and when should it outperform naive reward addition?* We identify two mechanisms. First, the two approaches imply fundamentally different reward decompositions. If a conventional reward model recovers f^*, adding the proxy produces $f^* + f^{\sim}$, thereby double-counting whatever information the proxy already captures. Residual learning instead recovers $f^{\sim} + f^{\Delta} = f^*$ by construction. Second, residual learning can reduce the complexity of the learning problem itself. We show that

$$\mathrm{Var}(f^{\Delta}) < \mathrm{Var}(f^*) \quad \Longleftrightarrow \quad 2\mathrm{Cov}(f^*, f^{\sim}) > \mathrm{Var}(f^{\sim}),$$

providing a concrete condition under which the proxy removes variation that would otherwise need to be learned from preference feedback.

This perspective changes the role of a proxy reward: it is not merely an additional reward signal, but prior information that determines what remains to be learned. We test this hypothesis through controlled experiments across eight RL environments, where access to the ground-truth reward allows us to systematically manipulate proxy informativeness and evaluate the predicted relationship between proxy quality and the benefit of residual learning.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

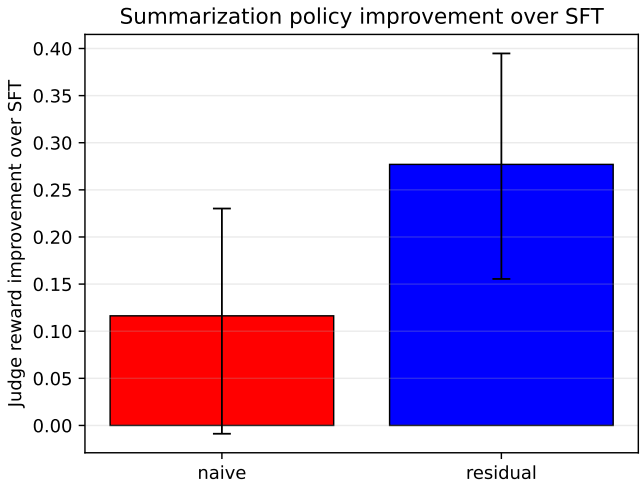


Figure 1: Improvement in mean judge reward over the common SFT policy for naive and residual reward composition across 200 held-out summarization prompts. Error bars show 95% bootstrap confidence intervals for each improvement. Residual additionally outperforms naive directly by 0.161 judge-reward points on average ($p = 0.0087$, paired Wilcoxon signed-rank test).

We then investigate whether this principle transfers to the setting in which RLHF is now most widely used: language-model alignment. We combine human preferences with a simple programmatic proxy while fine-tuning GPT-2 for summarization. According to an independently trained judge reward model, residual fine-tuning improves the base policy's reward by 0.277, compared with 0.116 for naive reward composition, more than doubling the observed improvement (Figure 1).

We make three main contributions:

1. We identify two mechanisms that explain why residual reward learning can outperform naive reward summation and formally characterize the conditions under which residual learning provides a simpler learning target.
2. We empirically validate these predictions across eight RL environments, including settings with incomplete and misspecified proxy rewards.
3. We extend residual reward learning to LLM fine-tuning and demonstrate its benefit over independently combining programmatic and learned rewards in a summarization task.

Together, our results suggest that preference learning need not reconstruct a reward from scratch when partial reward knowledge is already available. Instead, the learning problem can be restricted to explaining what the available proxy leaves unspecified or gets wrong.

2 Related work

2.1 Combining reward information in RL

Several lines of work combine multiple sources of information for reward learning. Preferences have, for example, been combined with demonstrations in robotics and Atari environments (Ibarz et al., 2018; Palan et al., 2019). These approaches differ from our setting because demonstrations provide behavioral evidence rather than an explicit proxy reward that can be evaluated during policy optimization. More generally, MAVRL (Baur et al., 2026) provides a probabilistic framework for learning a unified reward model from heterogeneous forms of feedback, including preferences, demonstrations, and ratings.

Most closely related, Cao et al. (2026) introduce Residual Reward Models (RRM) for preference-based RL. RRM decomposes the reward into a fixed prior and a learned residual and incorporates their sum into the Bradley–Terry objective. We use the same residual formulation but investigate a different question: *why and when should residual learning be preferable to independently learning*

a reward and combining it with a proxy? We identify two mechanisms—preservation of the reward decomposition and, under specific conditions, reduction in target variance—and empirically test the latter by systematically varying proxy informativeness.

CoRe (Ni et al., 2026) similarly decomposes rewards into a formal reward and a learned "residual" reward for robotic manipulation. Its residual reward is learned from VLM-generated video preferences, augmented with VLM-derived state-importance weights, while the formal reward is separately generated and refined using an LLM. Unlike RRM (Cao et al., 2026) and our residual condition, however, the formal reward does not enter the preference likelihood used to train the learned reward. CoRe therefore uses *residual* to describe the complementary role of the learned component, rather than explicitly training the model to predict the mathematical residual $f^* - f^{\sim}$.

2.2 Combining rewards for LLM alignment

Recent LLM post-training methods increasingly combine learned rewards with programmatic or automatically evaluated signals. Rule-Based Rewards (RBR) (Mu et al., 2024) use LLM-graded rules as direct rewards during RL, allowing explicit safety requirements to complement preference-based objectives. HERO (Tao et al., 2025) combines binary verifier rewards with continuous reward-model scores, using stratified normalization and variance-aware weighting to preserve correctness while providing denser feedback. For creative writing, RLMR (Liao et al., 2025) combines a reward model for subjective writing quality with a verifier for objective constraints, dynamically adjusting their contribution during online RL.

These approaches differ in how they compose rewards, but share a distinction from residual reward learning: the programmatic or automatically evaluated signal does not determine the target learned by the preference reward model. Instead, the reward signals are obtained separately and combined during policy optimization. In residual reward learning, the proxy enters the preference likelihood itself. Consequently, the reward model is optimized to explain preferences *conditional on* the information already represented by the proxy.

3 Method

We study how an imperfect proxy reward $f^{\sim}$ can be combined with preference-based reward learning. We compare three approaches, illustrated in Figure 2: *vanilla* learns a reward $\hat{f}$ exclusively from preferences; *naive* independently learns $\hat{f}$ and subsequently combines it with the proxy, $f^{+}_{\text{naive}} = f^{\sim} + \hat{f}$; and *residual* decomposes the target reward as

$$f^* = f^{\sim} + f^{\Delta} \tag{1}$$

and trains $\hat{f}^{\Delta}$ to model only the residual. Crucially, residual learning incorporates $f^{\sim}$ into the preference likelihood during reward-model training, such that $f^{+}_{\text{resid}} = f^{\sim} + \hat{f}^{\Delta}$. We provide the complete preference-learning objectives for all three approaches in Appendix A.1.

3.1 Why learn the residual?

Given that naive and residual composition use the same two sources of reward information, why should incorporating the proxy during reward-model training make a difference? Through an analysis of the reward decomposition and the statistical properties of the learning target, we identify two distinct reasons why residual reward learning can outperform naive summation. The first concerns the composition of the resulting reward; the second concerns the variance of the function being learned.

Preserving the reward decomposition. Consider first an idealized setting in which the preference-learned reward recovers the target reward f^* exactly. Naive summation then gives

$$f^{+}_{\text{naive}} = f^{\sim} + f^* = 2f^{\sim} + f^{\Delta}, \tag{2}$$

whereas a perfectly learned residual gives

$$f^{+}_{\text{resid}} = f^{\sim} + f^{\Delta} = f^*. \tag{3}$$

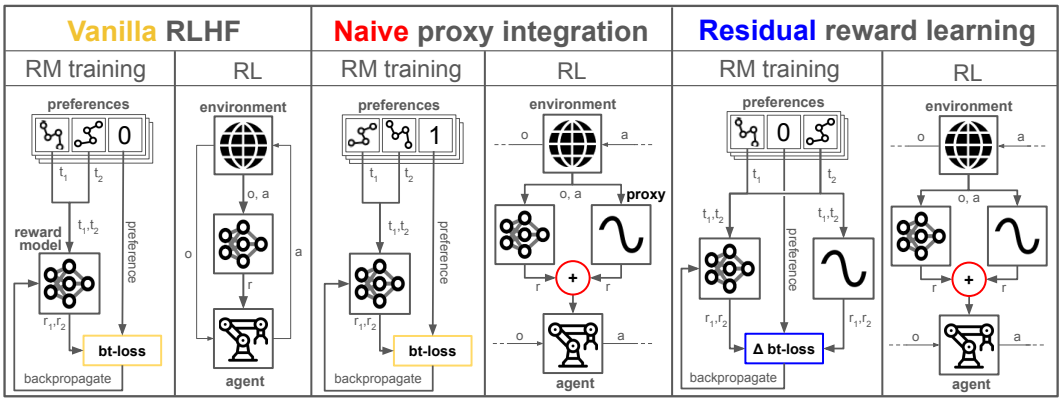


Figure 2: Training the reward model in *vanilla* RLHF is done by minimizing the Bradley-Terry loss (bt-loss) of the predicted reward on a preference dataset. During RL, the reward model predicts the state-wise reward for an agent. When integrating a proxy, hand-crafted reward function into RLHF, we have the choice to *naive*ly sum: train the reward model in the same way and add the proxy reward to the model's reward during RL, or use the proxy reward function also for learning a *residual* reward model by minimizing the adapted Bradley-Terry loss (Δ bt-loss) before adding the two rewards during RL.

Thus, naive summation double-counts the component represented by the proxy, while residual composition preserves the intended decomposition.

More generally, define estimation errors by

$$\hat{f} = f^* - \epsilon \qquad \text{and} \qquad \hat{f}^\Delta = f^\Delta - \epsilon^\Delta.$$

The two composed rewards become

$$f^+_{\text{naive}} = f^* + f^\sim - \epsilon, \; f^+_{\text{resid}} = f^* - \epsilon^\Delta. \tag{4}$$

Naive summation therefore contains an additional $f^\sim$ term even in the absence of estimation error, whereas residual composition does not. This distinction is particularly important when the proxy is misspecified. Suppose $f^\sim$ contains an erroneous component h. Because the residual is defined as $f^\Delta = f^* - f^\sim$, it contains the corresponding correction $-h$, allowing residual composition to cancel this error. In contrast, the naive reward model is trained independently of $f^\sim$ to approximate f^* and therefore has no reason to compensate for h; when $f^\sim$ is subsequently added, the erroneous component passes directly into f^+_{naive}. Residual learning can therefore not only avoid double-counting correct information in the proxy, but also correct information that the proxy gets wrong.

This argument concerns the reward decomposition itself and should be interpreted subject to the usual identifiability limitations of preference-based reward learning: preferences constrain differences in trajectory return rather than uniquely identifying a pointwise reward function. Our theoretical analysis in Appendix A.4 states the assumptions under which the comparison holds.

Reducing variation in the learning target. Residual learning may provide an additional advantage when the proxy explains substantial variation in the target reward. Since

$$f^\Delta = f^* - f^\sim,$$

the residual variance is

$$\text{Var}(f^\Delta) = \text{Var}(f^*) + \text{Var}(f^\sim) - 2\text{Cov}(f^*, f^\sim). \tag{5}$$

Consequently,

$$\text{Var}(f^\Delta) < \text{Var}(f^*) \iff 2\text{Cov}(f^*, f^\sim) > \text{Var}(f^\sim). \tag{6}$$

Equation 6 gives a precise condition under which subtracting the proxy reduces the variance of the target function. Intuitively, a sufficiently informative proxy removes systematic variation that the learned component would otherwise need to represent. Importantly, lower target variance alone does not guarantee lower generalization error for an arbitrary function approximator. Rather, it motivates the hypothesis that, under appropriate assumptions on the learner and data distribution, the residual can constitute an easier estimation problem. We state these assumptions formally in Appendix A.4 and empirically test the predicted relationship between proxy informativeness and residual-learning performance in Section 4.2.

3.2 Residual reward learning for LLM alignment

While residual reward learning has primarily been studied in conventional RL, we apply the same principle to LLM alignment, where preference-based reward learning has become an important component of post-training.

The residual model is trained using the Bradley-Terry loss, like conventional RLHF.

$$loss_{\Delta} = - \sum_{(\boldsymbol{o},\boldsymbol{a}^1,\boldsymbol{a}^2,\mu)\in\mathcal{D}} \left[\mu \log P^{+}(\boldsymbol{a}^1 \succ \boldsymbol{a}^2 \mid \boldsymbol{o}) + (1-\mu)\log P^{+}(\boldsymbol{a}^2 \succ \boldsymbol{a}^1 \mid \boldsymbol{o})\right]. \tag{7}$$

But with the important distinction that the preference probability P^{+} is not just given from the difference in the rewards predicted by the reward model $\hat{f}$, but by the sum of the proxy and model rewards $f^{\sim} + \hat{f}^{\Delta}$.

$$P^{+}(\boldsymbol{a}^1 \succ \boldsymbol{a}^2 \mid \boldsymbol{o}) = \sigma\left(f^{\sim}(\boldsymbol{o},\boldsymbol{a}^1) + \hat{f}^{\Delta}(\boldsymbol{o},\boldsymbol{a}^1) - f^{\sim}(\boldsymbol{o},\boldsymbol{a}^2) - \hat{f}^{\Delta}(\boldsymbol{o},\boldsymbol{a}^2)\right) \tag{8}$$

where σ denotes the Sigmoid function.

Since $f^{\sim}$ is fixed and gradients flow only through $\hat{f}^{\Delta}$, the residual model is optimized to complement or correct the contribution of the proxy in explaining the observed preferences.

This formulation allows explicitly measurable properties, such as length, formatting, repetition, or task-specific constraints, to be encoded programmatically while preference feedback captures properties that are harder to specify. Importantly, the distinction from naive composition is not which information is available, but where it is incorporated: naive composition combines independently obtained rewards during policy optimization, whereas residual learning incorporates the proxy during reward-model training.

4 Empirical evaluation

Our evaluation tests residual reward learning from three complementary perspectives. First, controlled RL environments provide access to the ground-truth reward, allowing us to construct known proxies and directly compare residual learning with naive composition. Second, this access lets us test our proposed explanation for the difference by relating residual target variance to reward-model prediction error. Finally, we test whether the benefit transfers beyond conventional RL by applying residual reward learning to LLM fine-tuning.

4.1 Main evaluation

We compare four reward formulations across eight environments (four MuJoCo, two Box2D, and two Atari) using 10 seeds per condition. Policy performance is evaluated using the ground-truth reward, which is not available during reward learning.

For each environment, we construct an imperfect proxy from the ground-truth reward. Five proxies are incomplete, retaining only a subset of the ground-truth reward terms, while three additionally contain a term with an inverted sign to introduce proxy misspecification. The exact reward decompositions are provided in Appendix A.3.

We compare *residual* and *naive* reward composition as our primary conditions. We additionally report *vanilla* preference learning and training exclusively on the *proxy* as baselines. All preference-based conditions use the same preference data and query budget.

	Method							
	proxy		Vanilla		Naive		Residual	
Environment	mean	std	mean	std	mean	std	mean	std
Hopper	987.4	4.8	2623.3	369.2	3113.1	712.8	**3510.0**	189.2
Walker2d	443.3	24.7	3381.6	487.6	2637.7	609.3	**3491.9**	460.3
Reacher	−11.8	0.7	−60.6	12.8	−9.3	0.7	**-9.1**	0.8
Pusher	−57.9	1.1	−93.9	40.7	−46.6	2.0	**-41.0**	3.4
LunarLander	−15 914.9	1479.8	−834.3	738.6	−70.6	32.4	**48.5**	118.8
BipedalWalker	−33.0	24.3	95.2	94.8	147.3	91.2	**166.2**	48.6
Qbert	218.3	59.6	545.0	243.6	528.6	233.0	**579.6**	210.2
SpaceInvaders	237.7	29.1	239.1	86.5	250.6	89.5	**262.8**	79.3

Table 1: The means of the final ground-truth training rewards for the four reward formulations across eight environments, with the bold means showing the maximum in that environment across 10 seeds.

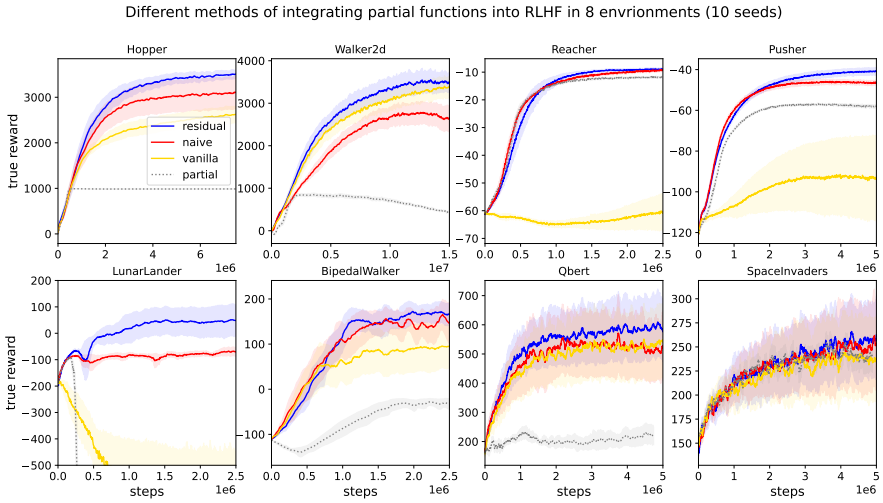


Figure 3: Comparing four different approaches of reward modeling in 8 environments with the lines showing the mean true reward across 10 seeds and the shaded region showing +- the standard deviation.

Results. Residual reward learning achieves the highest mean ground-truth return in all eight environments (table 1 and figure 3). Naive composition generally improves over either source alone, although vanilla preference learning outperforms naive composition in Walker2d and Qbert. Notably, residual learning also outperforms naive composition in the three environments containing a deliberately misspecified proxy component, consistent with residual learning being able to compensate for errors in the proxy.

To aggregate performance across environments with different reward scales, we min–max normalize returns within each environment, mapping the mean return of the worst random policy to 0 and the mean return observed with the best ground-truth policy to 1. Across the resulting normalized returns, residual achieves a mean of 0.68, compared with 0.59 for naive composition and 0.30 for vanilla preference learning. Residual therefore improves normalized return over naive composition by 0.09 absolute (15.3% relative). A paired Wilcoxon signed-rank test across the 80 environment-seed pairs finds higher returns for residual than naive ($W = 837$, $p = 2.8 \times 10^{-4}$), and for naive than vanilla ($W = 669$, $p = 5.1 \times 10^{-6}$).

4.2 TARGET VARIANCE AND PREDICTION ERROR

Our second proposed advantage of residual learning is that an informative proxy can reduce the variance of the learning target (Equation 6). We test whether this reduction is associated with lower reward-model prediction error.

For each of the eight environments, we train a conventional reward model $\hat{f}$ and a residual reward model $\hat{f}^{\Delta}$ on the same 1,000 randomly selected preference queries. We then construct an independent evaluation dataset of 100,000 transitions generated by a random policy. Access to the ground-truth reward allows us to compute both targets for every transition:

$$r^* = f^*(\boldsymbol{o}, \boldsymbol{a}), \qquad r^{\Delta} = f^*(\boldsymbol{o}, \boldsymbol{a}) - f^{\sim}(\boldsymbol{o}, \boldsymbol{a}). \tag{9}$$

Before computing prediction error, we affine-align each model's predictions to its respective ground-truth target by choosing $(a\hat{f} + b)$ to minimize mean squared error. This accounts for the scale and offset ambiguity of rewards learned from preferences.

For each environment, we compare the prediction-error ratio

$$\frac{\epsilon_{\Delta}}{\epsilon_*} = \frac{\mathrm{MSE}(\hat{f}^{\Delta}, r^{\Delta})}{\mathrm{MSE}(\hat{f}, r^*)} \tag{10}$$

with the target-variance ratio

$$\frac{\sigma^2_{\Delta}}{\sigma^2_*} = \frac{\mathrm{Var}(r^{\Delta})}{\mathrm{Var}(r^*)}. \tag{11}$$

Ratios below one respectively indicate lower prediction error and lower target variance for residual learning.

Results. The residual model has lower prediction error in seven of the eight environments, with BipedalWalker as the only exception. As shown in Figure 4, reductions in target variance closely correspond to reductions in prediction error. Across the eight environments, the variance and prediction-error ratios are strongly correlated (Spearman's $\rho = 0.905$, exact two-sided permutation $p = 0.0046$). For example, Pusher and Reacher exhibit large reductions in both target variance and prediction error, whereas Walker2d and BipedalWalker show little reduction in either. This relationship supports our hypothesis that when the proxy reduces the variation of the learning target, the residual can be easier to estimate from preferences. However, the association does not establish target variance as the sole cause of the reduction in prediction error.

4.3 LLM FINE-TUNING

Our final experiment tests whether the benefit of residual reward learning transfers from conventional RL environments to LLM alignment.

Setup. We fine-tune GPT-2 for Reddit TL;DR summarization using human preferences from the Summarize from Feedback dataset and a programmatic summary-quality proxy. The naive and residual reward models are trained on the same 1,000 preference comparisons. Starting from the same supervised fine-tuned policy, we then train separate policies with PPO, optimizing either the naive or residual reward composition. Full training and proxy details are provided in Appendix A.5.

For evaluation, we independently train a GPT-2 judge reward model on 5,000 preference comparisons using disjoint training, validation, and test splits. The judge is not used during policy training. It achieves 71.0% accuracy on 500 held-out human preference comparisons

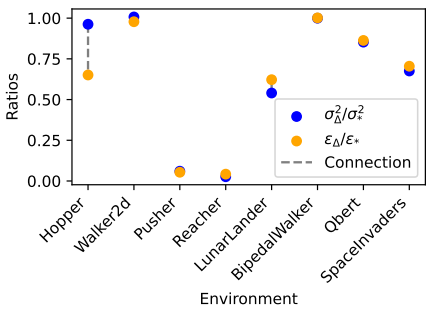


Figure 4: Ratio of residual to full-reward prediction MSE and ratio of residual to full-reward target variance across eight environments. Values below 1 indicate an advantage for the residual target.

and is subsequently used to score summaries generated by the resulting policies.

Baselines. Both policies optimize a reward consisting of the same programmatic proxy and a learned reward-model prediction. In the *naive* condition, the reward model is trained independently from the proxy and the two rewards are combined only during PPO. In the *residual* condition, the proxy is incorporated into the Bradley–Terry objective during reward-model training. Thus, the conditions differ in whether the proxy conditions reward-model learning.

Results. On 200 held-out summarization prompts, the SFT policy obtains a mean judge reward of -0.515. Naive reward composition improves this by 0.116 to -0.399, whereas residual reward learning improves it by 0.277 to -0.238 (Figure 1). Thus, the improvement over the common SFT policy is more than twice as large under residual learning. Directly comparing the two policies on the same prompts, residual learning achieves a 0.161 higher judge reward on average (bootstrap 95% CI $[0.052, 0.270]$; paired Wilcoxon signed-rank test, two-sided $p = 0.0087$). These results provide evidence that the benefit of conditioning reward learning on an available proxy extends to LLM alignment.

5 Discussion

Our results highlight an important distinction between *adding* a proxy reward and *conditioning reward learning* on it. In naive composition, the reward model is unaware of the proxy and therefore cannot account for information that the proxy already captures or gets wrong. Residual learning can do both: it can avoid relearning useful components and learn corrections for incorrect ones. This is reflected in our experiments: three of our proxies contain adversarial reward terms, yet residual learning still outperforms naive composition.

Our results further suggest that target simplification provides a second, distinct benefit. Across our environments, residual targets have lower variance than the full rewards, and this reduction is strongly associated with lower prediction error. Thus, residual learning can benefit both from composing the reward correctly and from reducing what must be inferred from preferences. Importantly, the latter advantage is conditional: a proxy does not necessarily produce a simpler residual.

Finally, our LLM experiment suggests that this principle is not specific to conventional RL environments. LLM objectives often combine properties that can be evaluated programmatically with qualities that are difficult to specify explicitly. Residual reward learning provides a way to let preference feedback model the latter while accounting for the former during reward-model training.

While our controlled RL experiments use synthetic proxies derived from known reward functions, our LLM experiment shows that residual learning can also benefit from an independently specified programmatic proxy. Evaluating how the quality and structure of such proxies affect residual learning across a broader range of real-world tasks remains important future work.

6 Conclusion

When part of a desired reward is already known, a reward model should not have to learn the entire reward from scratch. In this work, we investigated why incorporating this prior knowledge through residual reward learning can outperform learning a reward independently and adding the two afterward. We identified two complementary advantages: residual learning preserves the intended reward decomposition and can correct components that the proxy gets wrong, while an informative proxy can also reduce the variation of the remaining learning target.

Across eight diverse RL environments, residual reward learning consistently outperformed naive reward composition, including when proxies contained deliberately misspecified components. We further found a strong relationship between reductions in target variance and reward-model prediction error, providing empirical support for our proposed explanation of when residual learning is particularly beneficial. Finally, our LLM summarization experiment demonstrates that the principle extends beyond conventional RL to preference-based LLM fine-tuning.

When partial knowledge of a desired reward is available, preference learning can focus on precisely what that knowledge leaves missing or gets wrong.

Author Contributions If you'd like to, you may include a section for author contributions as is done in many journals. This is optional and at the discretion of the authors.

Acknowledgments Use unnumbered third level headings for the acknowledgments. All acknowledgments, including those to funding agencies, go at the end of the paper.

AI use disclaimer LLMs (Claude Sonnet 5, GPT-5.6 Luna) were used for 3 purposes:

1. As search engines to find related work papers.
2. For helping to draft the math theorems in Appendix A.4 and A.6.
3. Refining the writing of the paper.

In addition, Grammarly was used to check the correctness of the writing.

REFERENCES

Raphaël Baur, Yannick Metz, Maria Gkoulta, Mennatallah El-Assady, Giorgia Ramponi, and Thomas Kleine Buening. MAVRL: Learning Reward Functions from Multiple Feedback Types with Amortized Variational Inference, June 2026. URL http://arxiv.org/abs/2602.15206. arXiv:2602.15206 [cs.LG].

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The Arcade Learning Environment: An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research*, 47:253–279, June 2013. ISSN 1076-9757. doi: 10.1613/jair.3912. URL http://arxiv.org/abs/1207.4708. arXiv:1207.4708 [cs.AI].

Ralph Allan Bradley and Milton E. Terry. Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons. *Biometrika*, 39(3/4):324–345, 1952. ISSN 0006-3444. doi: 10.2307/2334029. URL https://www.jstor.org/stable/2334029.

Chenyang Cao, Miguel Rogel-García, Mohamed Nabail, Xueqian Wang, and Nicholas Rhinehart. Residual Reward Models: Leveraging Prior Knowledge for Efficient Preference-based Reinforcement Learning in Robotics, August 2026. URL http://arxiv.org/abs/2507.00611. arXiv:2507.00611 [cs.LG].

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep Reinforcement Learning from Human Preferences. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/hash/d5e2c0adad503c91f91df240d0cd4e49-Abstract.html.

Kavita Ganesan. ROUGE 2.0: Updated and Improved Measures for Evaluation of Summarization Tasks, March 2018. URL http://arxiv.org/abs/1803.01937. arXiv:1803.01937 [cs.IR].

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The Curious Case of Neural Text Degeneration, February 2020. URL http://arxiv.org/abs/1904.09751. arXiv:1904.09751 [cs.CL].

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models, October 2021. URL http://arxiv.org/abs/2106.09685. arXiv:2106.09685 [cs.CL].

Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. Reward learning from human preferences and demonstrations in Atari, November 2018. URL http://arxiv.org/abs/1811.06521. arXiv:1811.06521 [cs.LG].

Jianxing Liao, Tian Zhang, Xiao Feng, Yusong Zhang, Rui Yang, Haorui Wang, Bosi Wen, Ziying Wang, and Runzhi Shi. RLMR: Reinforcement Learning with Mixed Rewards for Creative Writing, August 2025. URL http://arxiv.org/abs/2508.18642. arXiv:2508.18642 [cs.AI].

Chin-Yew Lin. ROUGE: A Package for Automatic Evaluation of Summaries. 2004.

Tong Mu, Alec Helyar, Johannes Heidecke, Joshua Achiam, Andrea Vallone, Ian Kivlichan, Molly Lin, Alex Beutel, John Schulman, and Lilian Weng. Rule Based Rewards for Language Model Safety, November 2024. URL http://arxiv.org/abs/2411.01111. arXiv:2411.01111 [cs.AI].

Hexian Ni, Tao Lu, and Yinghao Cai. CoRe: Combined Rewards with Vision-Language Model Feedback for Preference-Aligned Reinforcement Learning, July 2026. URL http://arxiv.org/abs/2607.01721. arXiv:2607.01721 [cs.RO].

Malayandi Palan, Nicholas C. Landolfi, Gleb Shevchuk, and Dorsa Sadigh. Learning Reward Functions by Integrating Human Demonstrations and Preferences, June 2019. URL http://arxiv.org/abs/1906.08928. arXiv:1906.08928 [cs.RO].

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. 2019.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, August 2017. URL http://arxiv.org/abs/1707.06347. arXiv:1707.06347 [cs.LG].

H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pp. 287–294. Publ by ACM, 1992. doi: 10.1145/130385.130417. URL https://collaborate.princeton.edu/en/publications/query-by-committee/.

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. Learning to summarize from human feedback, February 2022. URL http://arxiv.org/abs/2009.01325. arXiv:2009.01325 [cs.CL].

Leitian Tao, Ilia Kulikov, Swarnadeep Saha, Tianlu Wang, Jing Xu, Sharon Li, Jason E. Weston, and Ping Yu. Hybrid Reinforcement: When Reward Is Sparse, It's Better to Be Dense, October 2025. URL http://arxiv.org/abs/2510.07242. arXiv:2510.07242 [cs.CL].

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A Standard Interface for Reinforcement Learning Environments, November 2025. URL http://arxiv.org/abs/2407.17032. arXiv:2407.17032 [cs.LG].

A APPENDIX

CONTENTS

A.1 Approches to preference learning 11
A.2 Gymnasium experiments training setup 12
A.3 Environments and proxy functions 12
A.4 Simpler learning target 13
A.5 LLM fine-tuning configuration 15
A.6 Active learning 17
A.6.1 Experiment 17

A.1 APPROCHES TO PREFERENCE LEARNING

Preference-based reward learning Following Christiano et al. (2017), we assume a dataset

$$\mathcal{D} = \{(\tau_i^1, \tau_i^2, \mu_i)\}_{i=1}^N$$

of preferences between trajectory segments, where $\mu_i = 1$ indicates a preference for τ_i^1, $\mu_i = 0$ a preference for τ_i^2, and $\mu_i = 0.5$ represents an indifferent preference encoded as equal probability mass on both outcomes.

A reward model $\hat{f}(\boldsymbol{o}, \boldsymbol{a})$ predicts a scalar reward for an observation–action pair. We denote its predicted return for a trajectory τ by

$$\hat{F}(\tau) := \sum_{(\boldsymbol{o}_t, \boldsymbol{a}_t) \in \tau} \hat{f}(\boldsymbol{o}_t, \boldsymbol{a}_t). \tag{12}$$

Under the Bradley & Terry (1952) preference model, the predicted probability that τ^1 is preferred to τ^2 is

$$\hat{P}(\tau^1 \succ \tau^2) = \sigma\left(\hat{F}(\tau^1) - \hat{F}(\tau^2)\right), \tag{13}$$

where σ denotes the sigmoid function.

The reward model is trained by minimizing binary cross-entropy,

$$\mathcal{L}_{\mathrm{BT}} = -\sum_{(\tau^1, \tau^2, \mu) \in \mathcal{D}} \left[\mu \log \hat{P}(\tau^1 \succ \tau^2) + (1 - \mu) \log \hat{P}(\tau^2 \succ \tau^1)\right]. \tag{14}$$

We refer to a reward model trained exclusively using this objective as *vanilla* preference-based reward learning.

Naive proxy integration Suppose that, in addition to preferences, we have access to an imperfect proxy reward

$$f^{\sim}(\boldsymbol{o}, \boldsymbol{a}),$$

which captures some information about the unknown target reward $f^*(\boldsymbol{o}, \boldsymbol{a})$. A straightforward way to incorporate this information is to independently train $\hat{f}$ using Equation 14 and then add its output to the proxy reward:

$$f^+_{\mathrm{naive}}(\boldsymbol{o}, \boldsymbol{a}) = f^{\sim}(\boldsymbol{o}, \boldsymbol{a}) + \hat{f}(\boldsymbol{o}, \boldsymbol{a}). \tag{15}$$

We refer to this approach as *naive summation*. Variants of additive reward composition have been used to combine learned and programmatic reward signals in prior work (Mu et al., 2024; Tao et al., 2025; Liao et al., 2025; Ni et al., 2026).

The key property of naive summation is that the learned component is trained independently of the proxy. Consequently, $\hat{f}$ is optimized to explain the complete preference signal even when part of that signal is already captured by $f^{\sim}$.

Residual reward learning Residual reward learning instead conditions the reward-learning problem on the information already provided by the proxy (Cao et al., 2026). We decompose the target reward as

$$f^* = f^{\sim} + f^{\Delta}, \tag{16}$$

where

$$f^{\Delta} := f^* - f^{\sim}$$

is the residual reward.

Rather than learning f^* directly, we train a model $\hat{f}^{\Delta}$ such that the combined reward is

$$f^{+}_{\text{resid}}(\boldsymbol{o}, \boldsymbol{a}) = f^{\sim}(\boldsymbol{o}, \boldsymbol{a}) + \hat{f}^{\Delta}(\boldsymbol{o}, \boldsymbol{a}). \tag{17}$$

Let

$$F^{\sim}(\tau) = \sum_{(\boldsymbol{o}_t, \boldsymbol{a}_t) \in \tau} f^{\sim}(\boldsymbol{o}_t, \boldsymbol{a}_t) \quad \text{and} \quad \hat{F}^{\Delta}(\tau) = \sum_{(\boldsymbol{o}_t, \boldsymbol{a}_t) \in \tau} \hat{f}^{\Delta}(\boldsymbol{o}_t, \boldsymbol{a}_t).$$

The preference probability is then

$$P^{+}(\tau^1 \succ \tau^2) = \sigma\left(F^{\sim}(\tau^1) + \hat{F}^{\Delta}(\tau^1) - F^{\sim}(\tau^2) - \hat{F}^{\Delta}(\tau^2)\right). \tag{18}$$

The residual model is trained using the same binary cross-entropy objective as Equation 14, replacing $\hat{P}$ with P^{+}. Thus, the proxy is incorporated *during reward-model training*, rather than only being added after training. Figure 2 sketches the difference between the 3 approaches to RLHF.

A.2 Gymnasium experiments training setup

A.3 Environments and proxy functions

The environments taken from gymnasium (Towers et al., 2025; Bellemare et al., 2013) and their respective reward functions with the terms used for the proxy function highlighted in **boldface** are:

- LunarLander - distance, speed, **tilt**, main engine, side engine, legs, **crash**, **land**
- BipedalWalker - forward, **control**
- Hopper - forward, **alive**, control
- Walker2d - forward, **alive**, control
- Reacher - distance, **control**
- Pusher - **-1.0 distance**, **control**, ball_distance
- SpaceInvaders - **-0.1 score**, **live_lost**
- Qbert - **-0.1 score**, **live_lost**

The first 5 of the reward functions are just incomplete. They are missing important terms needed for learning the full goal. The latter 3 reward functions have adversarial components. Features that should be rewarded are punished and vice versa.

Reward models training setup The policies are trained with PPO (Schulman et al., 2017) (hyperparameters can be found below in A.3). The policy and value models have two layers of 256 neurons [256, 256] with PReLU as an activation function.

The reward models have the same size [256, 256], use LeakyReLU activation between the two layers, and the logits are put through a hyperbolic tangent. The reward model is an ensemble of 5 networks with k-fold cross-validation and use early stopping with a patience=10. Each network is trained with an Adam optimizer with learning rate=1e-2 and regularized with L1 regression of a lambda of 1e-5 except for Atari where a lambda of 1e-3 is used.

The arguments for the RLHF are 10 rounds with linear queries per round (except for the first round which has double the queries of the others), a fragment length of 25 frames 50k policy training steps in between rounds, and then a large amount of final training steps. The environment specific final training timesteps and query amounts can be seen in figure 3.

Each proxy reward function is min-max-normalized with the min reward given for a state mapped to -1 and the max reward for a state mapped to +1 in order to coincide with the reward model's scale.

PPO hyperparameters The PPO hyperparameters used were:

n steps=2048,

batch size=256,

gamma=0.995, (0.99 for BipedalWalker)

gae lambda=0.97,

learning rate=2.5e-4,

n epochs=10,

ent coef=0.015, (0.03 for BipedalWalker)

target kl=0.02,

clip range=0.1;

A.4 SIMPLER LEARNING TARGET

We formalize the intuition that learning the residual can be easier than learning the complete reward function. Let the true reward be decomposed as

$$f^* = f^{\sim} + f^{\Delta},$$

where $f^{\sim}$ is the known proxy reward and f^{Δ} is the component that must be inferred from human preferences.

Residual variance Consider the reward values induced by a distribution over state-action pairs, and denote the corresponding random variables by f^*, $f^{\sim}$, and f^{Δ}. Since

$$f^{\Delta} = f^* - f^{\sim},$$

its variance is

$$\mathrm{Var}(f^{\Delta}) = \mathrm{Var}(f^*) + \mathrm{Var}(f^{\sim}) - 2\mathrm{Cov}(f^*, f^{\sim}). \tag{19}$$

Consequently,

$$\mathrm{Var}(f^{\Delta}) < \mathrm{Var}(f^*)$$

whenever

$$\mathrm{Var}(f^{\sim}) < 2\mathrm{Cov}(f^*, f^{\sim}).$$

Thus, a proxy reward that captures sufficiently much of the variation in the true reward reduces the variation that remains to be learned.

An equivalent interpretation follows by writing

$$f^* = f^{\sim} + f^{\Delta}.$$

The covariance between the proxy reward and the true reward is

$$\mathrm{Cov}(f^*, f^\sim) = \mathrm{Var}(f^\sim) + \mathrm{Cov}(f^\Delta, f^\sim).$$

Substituting into Eq. 19 gives

$$\mathrm{Var}(f^\Delta) = \mathrm{Var}(f^*) - \mathrm{Var}(f^\sim) - 2\mathrm{Cov}(f^\Delta, f^\sim).$$

Hence, when $f^\sim$ captures a substantial and systematic component of f^*, the residual contains less variation than the complete reward.

Implications for estimation Lower variance does not by itself guarantee lower sample complexity. To make the connection explicit, consider a simplified regression setting in which the reward model estimates the conditional mean from n preference-derived observations with independent zero-mean estimation noise of variance σ^2. For a target f, the variance of the empirical estimate scales as

$$\mathrm{Var}(\hat{f}) \propto \frac{\sigma_f^2}{n},$$

where σ_f^2 characterizes the variation of the target.

If

$$\mathrm{Var}(f^\Delta) < \mathrm{Var}(f^*),$$

then, for the same number of observations,

$$\mathrm{Var}(\hat{f}^\Delta) < \mathrm{Var}(\hat{f}).$$

Equivalently, achieving a fixed estimation variance ϵ requires, under this simplified model,

$$n_\Delta \propto \frac{\mathrm{Var}(f^\Delta)}{\epsilon} < \frac{\mathrm{Var}(f^*)}{\epsilon} \propto n_*.$$

This provides a formal basis for the intuition that the known proxy reward can reduce the amount of structure that the preference model needs to infer. The argument depends on $f^\sim$ explaining a meaningful portion of the variation in f^*; if $f^\sim$ is uninformative or poorly aligned with f^*, the residual need not be simpler.

MSE and variance ratios :

Hopper: $\epsilon_\Delta/\epsilon_*$: 0.651, $\sigma^2_\Delta/\sigma^2_*$, 0.962

Walker2d: $\epsilon_\Delta/\epsilon_*$: 0.978, $\sigma^2_\Delta/\sigma^2_*$, 1.007

Pusher: $\epsilon_\Delta/\epsilon_*$: 0.054, $\sigma^2_\Delta/\sigma^2_*$, 0.060

Reacher: $\epsilon_\Delta/\epsilon_*$: 0.042, $\sigma^2_\Delta/\sigma^2_*$, 0.026

LunarLander: $\epsilon_\Delta/\epsilon_*$: 0.622, $\sigma^2_\Delta/\sigma^2_*$, 0.540

BipedalWalker: $\epsilon_\Delta/\epsilon_*$: 1.002, $\sigma^2_\Delta/\sigma^2_*$, 0.999

Qbert: $\epsilon_\Delta/\epsilon_*$: 0.863, $\sigma^2_\Delta/\sigma^2_*$, 0.853

SpaceInvaders: $\epsilon_\Delta/\epsilon_*$: 0.705, $\sigma^2_\Delta/\sigma^2_*$, 0.675

A.5 LLM FINE-TUNING CONFIGURATION

We use GPT-2 (Radford et al., 2019) for both policy and reward models. We first perform supervised fine-tuning (SFT) on the `trl-lib/tldr` dataset to obtain the initial policy shared by both experimental conditions. Preference data are taken from the `comparisons` subset of the `openai/summarize_from_feedback` dataset (Stiennon et al., 2022). Each comparison consists of a Reddit post and two candidate summaries together with a human preference between them.

The experimental reward models are trained on the same 1,000 preference comparisons, with an additional 250 examples reserved for evaluation. Input posts and summaries are restricted to at most 6,000 and 1,200 characters, respectively.

Supervised fine-tuning. The common initial policy is trained for one epoch with a learning rate of 2×10^{-5}, a per-device batch size of 1, and 16 gradient-accumulation steps. We use a warm-up ratio of 0.03, a maximum sequence length of 768 tokens, and gradient checkpointing. The resulting SFT policy serves as the initialization and reference point for both PPO conditions.

Programmatic proxy reward. We construct a deterministic proxy from three automatically measurable properties of a generated summary: *coverage*, *length*, and *redundancy*. Let $\tau(\cdot)$ denote the implementation's tokenizer, which lowercases text and extracts word tokens with the regular expression `\b\w+\b`. For a source document x and generated summary y, let T_x be the set of source tokens and let $(y_1, \ldots, y_n) = \tau(y)$ be the summary tokens. The coverage term is the fraction of summary tokens that also appear in the source,

$$f_{\text{coverage}}(x, y) = \begin{cases} \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}[y_i \in T_x], & n > 0, \\ 0, & n = 0. \end{cases} \tag{20}$$

The length term is a Gaussian-shaped reward centered at 48 tokens with bandwidth 20,

$$f_{\text{length}}(y) = \exp\left(-\frac{1}{2} \left(\frac{n - 48}{20} \right)^2 \right). \tag{21}$$

Finally, the redundancy penalty is the fraction of repeated bigrams in the summary. With $B_y = \{(y_i, y_{i+1})\}_{i=1}^{n-1}$ as the multiset of summary bigrams and U_y as the corresponding set of unique bigrams, the signed redundancy component used by the proxy is

$$f_{\text{redundancy}}(y) = \begin{cases} -\left(1 - \frac{|U_y|}{|B_y|}\right), & n \geq 2, \\ 0, & n < 2. \end{cases} \tag{22}$$

The raw proxy reward is therefore

$$\tilde{f}(x, y) = f_{\text{coverage}}(x, y) + f_{\text{length}}(y) + f_{\text{redundancy}}(y). \tag{23}$$

Before reward composition, the raw proxy is multiplied by a scale factor of 1.0, normalized using a fixed mean of 1.0 and standard deviation of 0.5, and clipped to the interval $[-5.0, 5.0]$:

$$f_{\text{proxy}}(x, y) = \text{clip}\left(\frac{\tilde{f}(x, y) - 1.0}{0.5}, -5.0, 5.0 \right). \tag{24}$$

The same transformed proxy reward is used in both reward-composition conditions: in the naive condition it is added to the learned preference reward at PPO time, while in the residual condition the same proxy margin is included in the Bradley–Terry training objective and the transformed proxy is again added to the learned residual reward at PPO time.

We additionally report ROUGE (Lin, 2004; Ganesan, 2018) as a standard reference-based summarization metric (see A.5). ROUGE is computed only during policy evaluation and is not used in reward-model training, PPO reward computation, or to distinguish the naive and residual reward-learning conditions.

Reward-model training. The naive and residual reward models use the same GPT-2 architecture, preference data, and optimization parameters. Both are trained for one epoch with a learning rate of 10^{-5}, per-device batch size 1, 16 gradient-accumulation steps, warm-up ratio 0.03, and maximum sequence length 768. Gradient checkpointing is enabled.

The naive model is trained using the conventional Bradley–Terry preference objective. The residual model instead incorporates the fixed proxy into the Bradley–Terry likelihood as described in Equation 8. Learned reward-model outputs are transformed using $\tanh$ before reward composition. Thus, the two conditions differ in the reward-model training objective but otherwise use the same model architecture, preference data, and optimization settings.

Policy fine-tuning. Starting from the same SFT policy, we train separate naive and residual policies using Proximal Policy Optimization (PPO) (Schulman et al., 2017). Parameter-efficient fine-tuning is performed using Low-Rank Adaptation (LoRA) (Hu et al., 2021), with rank $r = 16$, scaling parameter $\alpha = 32$, and dropout 0.05. For GPT-2, adapters are applied to the `c_attn`, `c_proj`, and `c_fc` modules.

Both policies are trained for 500 PPO episodes using a learning rate of 10^{-7} and a per-device batch size of 1. Prompts are truncated to 384 tokens and responses to 48 newly generated tokens. During PPO generation, we use temperature 0.5 and nucleus sampling (Holtzman et al., 2020) with $p = 0.8$.

We use an initial KL coefficient of 1.0 with adaptive KL control targeting 0.5. The PPO policy and value clipping ranges are both 0.05, the maximum gradient norm is 0.5, and reward scores are clipped at 3.0. Training additionally monitors the reward over a 50-step window for plateau-based early stopping after a minimum of 150 steps, with a patience of four windows. Neither condition triggered early stopping, and both completed all 500 episodes.

The two policies differ only in the learned component of their training reward. Naive composition uses

$$R_{\text{naive}}(\boldsymbol{o}, \boldsymbol{a}) = f^{\sim}(\boldsymbol{o}, \boldsymbol{a}) + \hat{f}(\boldsymbol{o}, \boldsymbol{a}), \tag{25}$$

whereas residual composition uses

$$R_{\text{residual}}(\boldsymbol{o}, \boldsymbol{a}) = f^{\sim}(\boldsymbol{o}, \boldsymbol{a}) + \hat{f}^{\Delta}(\boldsymbol{o}, \boldsymbol{a}). \tag{26}$$

In the latter case, $\hat{f}^{\Delta}$ has been trained conditional on $f^{\sim}$ through the residual Bradley–Terry objective.

Independent judge model. To evaluate policy quality independently from the reward models used for PPO, we train a separate GPT-2 judge reward model on 5,000 human preference comparisons. The data are split into 80% training, 10% validation, and 10% test sets. The resulting held-out test set contains 500 comparisons.

The judge achieves an accuracy of 71.0%, precision of 70.0%, recall of 72.0%, and F_1 score of 70.9% on these held-out human preferences. The programmatic proxy is only moderately correlated with the judge: the Spearman correlation is 0.455 between absolute scores and 0.401 between preference margins. This indicates that the proxy captures information related to human preferences without being equivalent to the independently learned preference signal.

Policy evaluation. We generate summaries from the SFT, naive, and residual policies on the same 200 held-out prompts and score them using the independent judge model. Evaluation generation uses a maximum of 128 new tokens, temperature 0.9, and nucleus sampling with $p = 0.95$.

The SFT policy obtains a mean judge reward of -0.515, compared with -0.399 for naive composition and -0.238 for residual learning. Relative to the common SFT policy, this corresponds to mean improvements of 0.116 for naive composition and 0.277 for residual learning. Bootstrap 95% confidence intervals are $[-0.009, 0.230]$ and $[0.155, 0.395]$, respectively.

As a complementary automatic evaluation, we compute ROUGE scores on 150 generated summaries. Both PPO policies improve over the SFT policy. The naive and residual policies obtain similar ROUGE-L scores (0.0849 and 0.0849, respectively), despite the substantially higher human-preference judge reward of the residual policy. This difference is consistent with the two metrics measuring different aspects of summary quality and motivates evaluating the policies using a reward model trained directly from human preferences.

A.6 ACTIVE LEARNING

Active learning can reduce the number of human queries by selecting comparisons expected to provide the most information to the reward model. We use Query by Committee (QBC) (Seung et al., 1992), in which an ensemble of reward models is trained and candidate comparisons are ranked by disagreement between their predicted preference probabilities.

For conventional preference learning, ensemble member i assigns the Bradley–Terry probability

$$P_i(\tau^1 > \tau^2) = \sigma\left(F_i(\tau^1) - F_i(\tau^2)\right). \tag{27}$$

The acquisition score is the variance of P_i across ensemble members.

Residual learning changes these preference probabilities because the fixed proxy contributes to the Bradley–Terry likelihood. Let

$$d^{\sim} := F^{\sim}(\tau^1) - F^{\sim}(\tau^2) \tag{28}$$

denote the difference in proxy return between two trajectories, and let

$$d_i^{\Delta} := F_i^{\Delta}(\tau^1) - F_i^{\Delta}(\tau^2) \tag{29}$$

denote the corresponding difference predicted by residual model i. The preference probability is then

$$P_i^{+}(\tau^1 > \tau^2) = \sigma\left(d^{\sim} + d_i^{\Delta}\right). \tag{30}$$

The proxy difference $d^{\sim}$ is identical across ensemble members and therefore does not directly change the variance of their predicted reward differences. It can, however, change disagreement after the nonlinear sigmoid transformation. The sensitivity of the preference probability to the residual prediction is

$$\frac{\partial P_i^{+}}{\partial d_i^{\Delta}} = P_i^{+}(1 - P_i^{+}). \tag{31}$$

This sensitivity is maximized at $P_i^{+} = 1/2$ and approaches zero as P_i^{+} approaches either 0 or 1. Consequently, if the combined proxy and residual prediction already strongly favors one trajectory, differences between the residual models have less influence on their predicted preference probabilities.

For small disagreement between ensemble members, a first-order Taylor expansion around the mean residual prediction $\bar{d}^{\Delta}$ gives

$$P_i^{+} \approx \sigma(d^{\sim} + \bar{d}^{\Delta}) + \sigma(d^{\sim} + \bar{d}^{\Delta})\left[1 - \sigma(d^{\sim} + \bar{d}^{\Delta})\right](d_i^{\Delta} - \bar{d}^{\Delta}). \tag{32}$$

The variance of the predicted preference probabilities is therefore approximately

$$\mathrm{Var}(P^{+}) \approx \left[\sigma(d^{\sim} + \bar{d}^{\Delta})\left(1 - \sigma(d^{\sim} + \bar{d}^{\Delta})\right)\right]^2 \mathrm{Var}(d^{\Delta}). \tag{33}$$

Thus, incorporating the proxy into the preference likelihood also changes the QBC acquisition criterion. Disagreement is attenuated for comparisons for which the combined reward predicts a confident preference and remains larger around uncertain comparisons. This suggests that residual QBC may direct queries away from comparisons already well explained by the available reward information. We empirically test whether this change in query selection improves downstream policy performance.

A.6.1 EXPERIMENT

Setup. We ablate the query-selection strategy while keeping residual reward learning fixed. We compare three conditions: *random*, which samples preference queries uniformly; *vanilla QBC*, which selects comparisons according to disagreement computed from the residual-model predictions alone; and *residual QBC*, which computes disagreement after incorporating the proxy into the preference probability as in Equation 30. We reuse the eight environments, proxy rewards, query budgets, and training setup from the main evaluation, with 10 seeds per condition. Thus, the conditions differ only in how preference queries are selected.

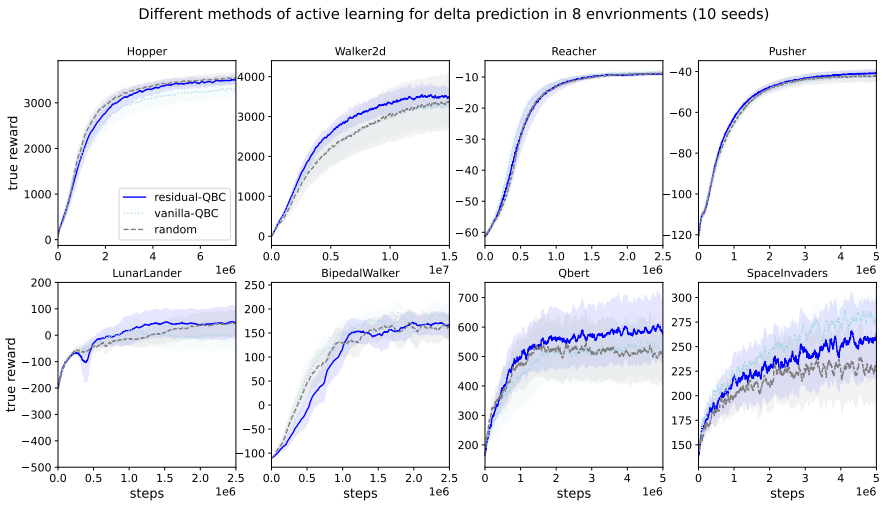


Figure 5: Ablation of the active-learning strategy while keeping residual reward learning fixed. Queries are selected randomly, using disagreement between residual-model predictions alone (*vanilla QBC*), or using disagreement after incorporating the proxy into the preference probabilities (*residual QBC*). Lines show the mean ground-truth policy return across 10 seeds and shaded regions show $\pm$ one standard deviation. We observe no statistically significant difference between the three query-selection strategies.

Results. Residual QBC achieves the highest mean return in three of the eight environments, vanilla QBC in three, and random selection in two. However, we find no statistically significant differences in normalized policy return between the three query-selection strategies. Figure 5 shows the corresponding training curves.

These results do not provide sufficient evidence that incorporating the proxy into the QBC acquisition criterion improves downstream policy performance in our setting. Although the analysis above shows that the proxy changes the disagreement measured by QBC, this change does not produce a statistically detectable performance improvement under our experimental conditions. In contrast, the main evaluation finds a clear difference between naive and residual reward-model training. Our results therefore provide substantially stronger evidence for the effect of incorporating the proxy during reward-model training than for its effect on active query selection.