Proxy Exploration and Reusable Guidance: A Modular LLM Post-Training Paradigm via Proxy-Guided Update Signals

Daocheng Fu[*,1,2], **Rong Wu**[*,1,3], **Yu Yang**[*,1,3], **Xuemeng Yang**[1], **Jianbiao Mei**[1,3], **Licheng Wen**[1,4,5], **Pinlong Cai**[1], **Yong Liu**[3], **Botian Shi**[†,1], **Yu Qiao**[1]

[1] Shanghai AI Laboratory, [2] Fudan University, [3] Zhejiang University,
[4] Shanghai Innovation Institute, [5] Shanghai Jiao Tong University

[*] Equal Contribution, [†] Corresponding Author

Abstract

Post-training is essential for refining the domain-specific capabilities of large language models (LLMs), yet existing reward optimization and distribution matching methods tightly couple policy exploration with distribution alignment. This coupling forces expensive exploration directly on the policy model and severely hinders the asynchronous generation, reuse, and cross-model transfer of optimization signals. In this paper, we propose **Proxy-guided Update Signal Transfer (PUST)**, a novel post-training framework that fundamentally decouples update-signal exploration from distribution alignment. Instead of utilizing the primary model for costly exploration, PUST employs a lightweight proxy model as an efficient testbed to discover high-reward behaviors. We extract the relative improvement signal between the proxy's initial and optimized states, transferring this directional update to the primary model to guide its policy alignment. This decoupled pipeline, comprising proxy exploration, update-signal extraction, and signal transfer, significantly reduces computational overhead and enables optimization signals to be asynchronously generated, cached, and reused. Crucially, by transferring relative improvements rather than absolute policy distributions, PUST naturally supports weak-to-strong improvement and seamless cross-model transfer. Systematic evaluations on Qwen3-family models across math and code domains demonstrate that update signals extracted from substantially weaker proxies can robustly and adjustably enhance stronger primary models. Ultimately, PUST transforms post-training from a monolithic online optimization process into a highly modular, reusable, and cost-efficient paradigm.

Date: July 14, 2026
Github Repo: https://github.com/KnowledgeXLab/PUST
Huggingface Page: https://huggingface.co/KnowledgeXLab/PUST-Experiments

1 Introduction

Post-training is essential for refining the usability and domain-specific capabilities of large language models (LLMs) [1], yet increasing model scale and domain diversity demand higher training efficiency and lower computational costs. Existing PPO-like methods (e.g., PPO [2], GRPO [3]) optimize models via on-policy rollouts and reward feedback. However, adapting them to multiple domains typically necessitates a sequential and costly optimization process across different settings (Fig. 1(a)). The emergence of OPD [4, 5] introduces a new paradigm: instead of generating reward signals directly, it uses distribution matching to align a student model with optimized expert teachers, enabling cross-domain data-level parallelization (Fig. 1(b)). This shift implies that reward-oriented optimization and distribution alignment serve distinct roles, motivating our investigation into how policy-improvement signals are explored and applied for optimization.

As depicted in Fig. 1(b), this parallel pipeline comprises two distinct stages: **policy exploration** and **policy alignment**. It initially utilizes reward-oriented exploration to discover high-reward behaviors and construct expert

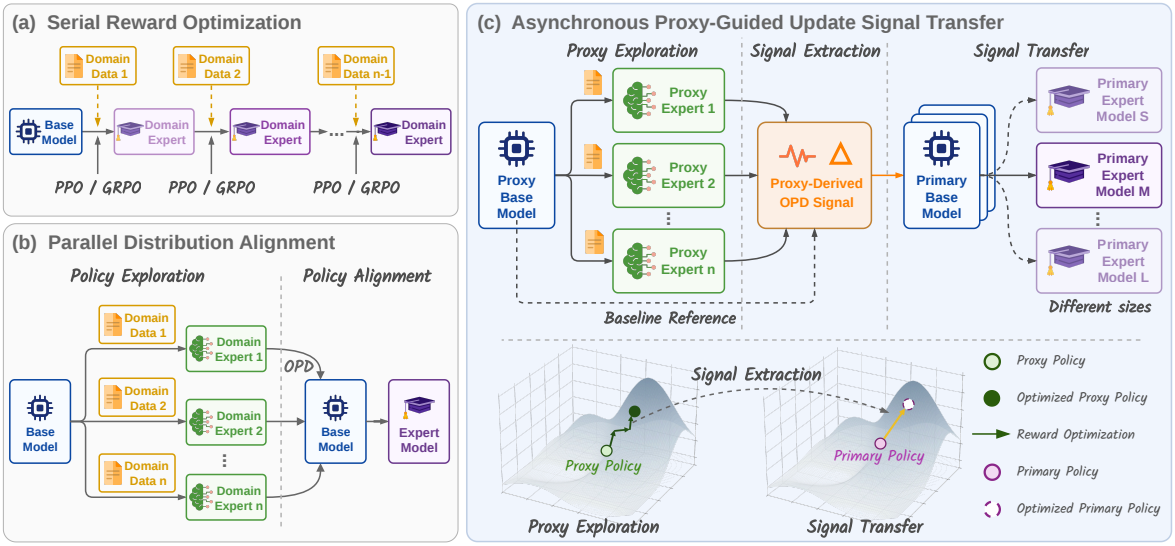

Figure 1: **Comparison of post-training pipelines. (a) Serial**: Sequential domain exploration, risking catastrophic forgetting. **(b) Parallel**: Parallel domain exploration followed by unified policy alignment. **(c) Proxy Asynchronous (Ours)**: A proxy model conducts asynchronous exploration; extracted signals are subsequently transferred to various primary models. Decoupling these update signals from the base model enables seamless propagation and reuse.

teachers, followed by aligning the student model with their distributions via distillation. While OPD enhances parallel alignment efficiency across multiple experts, its exploration phase inherently depends on the model being trained. Consequently, exploration and alignment remain tightly coupled within the same model, restricting the asynchronous generation, caching, reuse, and cross-model transfer of the discovered high-reward distributions. This limitation raises a natural question: *can we decouple exploration and alignment from the policy model to enable asynchronous exploration and seamless cross-model transfer?*

To address this issue, we propose **Proxy-guided Update Signal Transfer (PUST)**, a novel post-training framework that decouples update-signal exploration from distribution alignment. Instead of performing costly exploration on the primary model being trained, we introduce **proxy exploration** [6], where a proxy model serves as a low-cost testbed for exploration to discover high-reward behaviors. Furthermore, rather than directly distilling the optimized proxy, our framework introduces **update-signal extraction and transfer**. Specifically, we extract the relative improvement signal between the initial and optimized states of the proxy model, and subsequently transfer this update signal to the primary model to guide policy updates via alignment. Through the sequential pipeline of *proxy exploration, update-signal extraction, and signal transfer*, exploration results can be asynchronously generated and seamlessly transferred across different primary models, enabling efficient and scalable cross-model post-training.

This decoupled design offers several key advantages. First, it significantly reduces exploration costs by delegating expensive sampling and reward evaluation to a smaller proxy model, allowing the primary model to focus solely on applying the extracted signals for optimization. Second, it enhances signal reusability, allowing update signals to be asynchronously generated, cached, scaled, and reused across distinct training runs. Third, it improves scalability and supports weak-to-strong improvement [7, 8]; since the framework transfers the proxy's relative improvement rather than its absolute policy distribution, signals explored by a weaker proxy can be flexibly applied to various primary models, even substantially stronger ones. Overall, this decoupling transforms post-training from a monolithic online optimization process into a highly modular and reusable pipeline for signal exploration, extraction, and transfer.

We systematically evaluate Proxy-guided Update Signal Transfer on Qwen3-family models [9] across math and code domains. Our experiments reveal that update signals extracted from a substantially weaker proxy can still effectively improve a stronger primary model. Additionally, this transfer intensity is adjustable via a scaling coefficient, yielding robust performance across diverse settings. Crucially, the reusability of these signals across models and configurations indicates that useful post-training insights are not strictly bound to the proxy's absolute capability, but can instead be abstracted as transferable, adjustable distributional improvement signals.

Our main contributions are summarized as follows:

- We reinterpret post-training as distinct policy exploration and alignment stages, revealing how their tight coupling in existing methods inflates exploration costs and hinders cross-model adaptability.
- We propose the Proxy-guided Update Signal Transfer framework comprising proxy exploration, update-signal extraction, and signal transfer, which uniquely transfers the proxy's relative improvement signal to facilitate flexible policy updates.

- We demonstrate that these update signals can be seamlessly scaled, reused, and transferred across models to enable an asynchronous and cost-efficient post-training paradigm.

2 Preliminary Analysis: Reward Optimization vs. Distribution Matching

As discussed in Section 1, post-training fundamentally comprises two distinct stages: policy exploration and policy alignment. Reward optimization heavily emphasizes the former by actively seeking high-reward behaviors, whereas distribution matching executes the latter by aligning the model to a predetermined distribution. To elucidate their underlying differences and training efficiencies, this section compares two representative algorithms: GRPO and OPD.

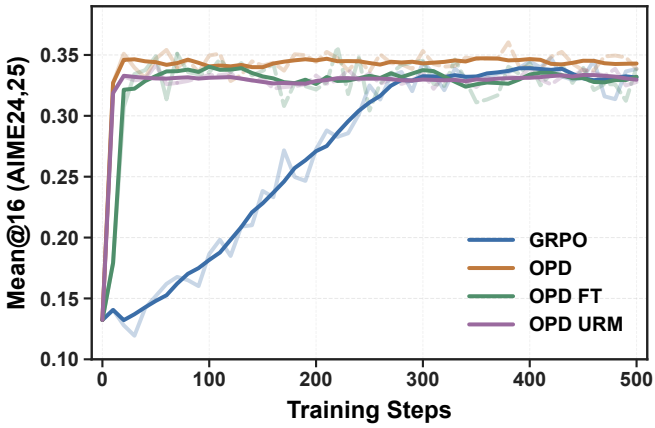

Figure 2: Mean@16 trajectories on AIME2024 and AIME2025. OPD variants converge within 70 steps and are extended to 500 steps based on converged-region statistics (i.e. mean and std).

To systematically evaluate these approaches using Qwen3-1.7B as the base model, we configure four training pipelines: **(1) standard GRPO**, optimized for 500 steps to serve as the expert teacher; and **(2) standard OPD**, which directly aligns the student model with this teacher. Furthermore, to explicitly verify that OPD is inherently reward-agnostic, we introduce two reward-related variants: **(3) OPD FT (Filtered Trajectories)**, which discards training groups comprising entirely correct trajectories of the student; and **(4) OPD URM (Update Reward Mask)**, which applies a token-level mask to discard tokens whose update direction contradicts the overall reward signal.

As illustrated in Fig. 2, our comparative analysis yields three key insights: (1) regarding convergence efficiency, all OPD variants achieve rapid convergence (at around 20 steps), whereas GRPO requires a significantly longer exploratory phase to identify optimal update directions; (2) regarding final efficacy, all four configurations ultimately converge to a comparable performance level, indicating that the FT and URM mechanisms do not alter overall effectiveness, though OPD FT converges slightly slower due to reduced update signal; and (3) regarding mechanistic limitations, the results reveal that OPD lacks intrinsic reward awareness, indiscriminately aligning the student's distribution with the teacher's. Even with URM, token probabilities are merely adjusted based on trajectory-level correctness, which cannot guarantee strict token-level reward relevance.

In conclusion, while distribution-matching algorithms are highly efficient during policy alignment, they inherently rely on active policy exploration to acquire an expert teacher with a high-quality target distribution. The genuine directional signal for policy improvement fundamentally stems from the active exploration inherent in reward optimization. Consequently, to further enhance post-training efficiency, it is imperative to investigate how to decouple, reuse, and seamlessly transfer these extracted update signals across models.

3 Methodology

As illustrated in Section 2, while distribution matching efficiently condenses multi-step reward optimization into direct policy alignment, its reliance on a pre-existing target distribution often forces a restrictive strong-to-weak or human demonstration paradigm.

To break this dependency, we reformulate post-training around the concept of the **update signal**. Instead of requiring a stronger target policy, we utilize a proxy model to explore reward-induced update signals that are then extracted and transferred to the primary model, decoupling the process into a scalable, reusable, and cross-model pipeline: *proxy exploration, update-signal extraction, and signal transfer*.

3.1 Setup

Let x denote an input prompt and $y = (y_1, \ldots, y_T)$ denote a corresponding generated response. At decoding step t, we define the token-level state as $s_t = (x, y_{<t})$.

Unlike standard on-policy alignment, our framework involves two distinct models sharing a common vocabulary $\mathcal{V}$: a lightweight proxy model designed for low-cost trial-and-error exploration, and a more capable primary model that

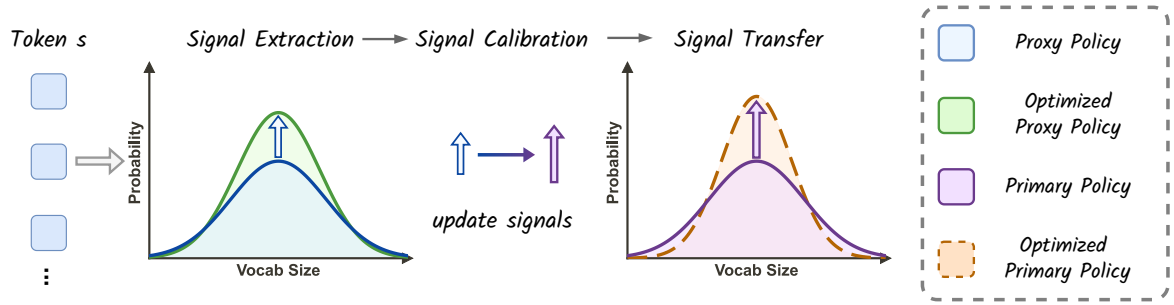

Figure 3: **Illustration of the PUST mechanism.** This pipeline decouples exploration from alignment via three sequential stages. First, a relative update signal is extracted from the explored proxy model pair. Next, to prevent over-updating, the signal is dynamically calibrated against the primary model's anchor. Finally, the calibrated signal is transferred to guide the primary model's alignment, ensuring stable and flexible policy optimization.

we ultimately aim to improve. Correspondingly, we define four policies essential to our framework: (1) the initial proxy policy before optimization, π_ϕ; (2) the reward-optimized proxy policy, π_ϕ^+; (3) the frozen anchor policy of the primary model, π_{ref}; (4) the primary policy to be updated, π_θ.

3.2 Stage 1: Proxy Exploration

Directly applying reward optimization algorithms (e.g., PPO or GRPO) to the primary model incurs prohibitive sampling costs and lacks flexibility, particularly when adapting to diverse reward distributions. To decouple this active exploration from the primary policy, we introduce *proxy exploration*.

Instead of updating the primary model directly, we deploy a proxy model, π_ϕ, to serve as an efficient and flexible testbed. In this stage, we optimize π_ϕ using standard reward optimization to maximize a given reward function $r(x, y)$, yielding the optimized proxy policy π_ϕ^+.

By offloading the trial-and-error exploration to the proxy, we achieve highly efficient policy exploration without repetitively burdening the primary model. More importantly, this decoupled design enables the deployment of multiple proxy models in parallel to explore diverse high-reward behaviors and optimal distributions.

3.3 Stage 2: Update-Signal Extraction

Following proxy exploration, standard distillation typically aligns the primary model directly to the optimized proxy, π_ϕ^+. However, mimicking its absolute distribution inherently binds the explored updates to the proxy itself, preventing the signals from being reused across different primary models. To decouple these learned signals and enable flexible cross-model transfer, we introduce *update-signal extraction*.

Instead of absolute distillation, PUST extracts the relative improvement signal. For each state s_t and token $a \in \mathcal{V}$, the proxy-induced token-level update direction is defined as:

$$\Delta_\phi(a \mid s_t) = \log \pi_\phi^+(a \mid s_t) - \log \pi_\phi(a \mid s_t) = \log \frac{\pi_\phi^+(a \mid s_t)}{\pi_\phi(a \mid s_t)}. \tag{1}$$

The term $\Delta_\phi(a \mid s_t)$ precisely encapsulates the reward-induced signal: if $\Delta_\phi > 0$, the proxy's tendency to select token a has been encouraged by the reward; if $\Delta_\phi < 0$, it has been suppressed.

This relative signal represents a generalized directional improvement, making it highly transferable regardless of the proxy's absolute capability. More importantly, the extracted update signals can be easily cached and reused, establishing a highly scalable foundation for subsequent cross-model signal transfer.

3.4 Stage 3: Signal Transfer

In the final stage, we introduce *Signal Transfer* to apply the extracted relative improvements to the primary model. Unlike standard distillation that forces the primary model to blindly mimic a fixed, absolute distribution, signal transfer dynamically transplants the proxy-explored update signal onto the primary model's intrinsic policy space. This decoupled mechanism not only bypasses the capability bottleneck of weaker proxies, but also establishes a highly stable and scalable paradigm for cross-model alignment.

3.4.1 Anchor-Based Update Calibration

During signal transfer, the extracted update signal Δ_ϕ remains fixed post-exploration. Consequently, repeatedly transferring this static signal to the primary policy π_θ across multiple steps could induce severe over-updating. Once the primary model shifts toward the proxy-suggested direction, continuing to amplify the same tokens indiscriminately becomes detrimental.

To dynamically measure the update already absorbed by the primary model, we define the primary-anchor log-ratio:

$$\Delta_\theta(a \mid s_t) = \log \frac{\pi_\theta(a \mid s_t)}{\pi_{\text{ref}}(a \mid s_t)}. \tag{2}$$

To prevent over-updating, we introduce Δ_θ as a penalty term and define the token-level transferred utility with anchor calibration as:

$$r_\lambda(a \mid s_t) = \Delta_\phi(a \mid s_t) - \lambda \Delta_\theta(a \mid s_t), \tag{3}$$

where the calibration coefficient $\lambda > 0$ dictates the alignment conservatism. A larger λ imposes a stricter penalty against deviations from the anchor policy, yielding more conservative updates, whereas a smaller λ allows the primary model to aggressively exploit the proxy's update signal.

3.4.2 Signal-Guided Objective Formulation

Given a set of token-level states $\mathcal{D}$, we optimize π_θ by maximizing the expected transferred utility under its current token distribution:

$$\mathcal{J}_{\text{proxy}}(\theta) = \mathbb{E}_{s_t \sim \mathcal{D}} \left[\sum_{a \in \mathcal{V}} \pi_\theta(a \mid s_t) \left(\Delta_\phi(a \mid s_t) - \lambda \log \frac{\pi_\theta(a \mid s_t)}{\pi_{\text{ref}}(a \mid s_t)} \right) \right]. \tag{4}$$

Equivalently, this maximization can be cast as minimizing the following objective:

$$\mathcal{L}_{\text{proxy}}(\theta) = -\mathbb{E}_{s_t \sim \mathcal{D}} \left[\sum_{a \in \mathcal{V}} \pi_\theta(a \mid s_t) \left(\log \frac{\pi_\phi^+(a \mid s_t)}{\pi_\phi(a \mid s_t)} - \lambda \log \frac{\pi_\theta(a \mid s_t)}{\pi_{\text{ref}}(a \mid s_t)} \right) \right]. \tag{5}$$

Notably, π_ϕ, π_ϕ^+, and π_{ref} remain strictly frozen, and gradients are exclusively applied to the primary policy π_θ.

Mathematically, minimizing Eq. (5) is equivalent to minimizing the Kullback-Leibler (KL) divergence between the primary policy and a dynamically induced target distribution. This reveals PUST's core mechanism: it acts as an efficient distribution-matching algorithm that, instead of mimicking a pre-existing absolute distribution, aligns to a dynamic target constructed by transferring the proxy-explored update signal onto the primary model's anchor space.

4 Experiments

In this section, we design experiments to answer the following questions:

- Can update signals obtained from a proxy model improve the performance of the primary model (Section 4.1)?
- Can update signals explored by a proxy model be reused across different primary models (Section 4.2)?
- How do update signals explored by a proxy model change after multiple rounds of transfer (Section 4.3)?
- How do different proxy models and correction coefficients affect the final performance of the primary model (Section 4.4)?
- How large is the gap between update signals explored by a proxy model and those explored directly by the primary model (Section 4.5)?

4.1 Performance Improvement of Proxy-guided Update Signal Transfer

To evaluate the effectiveness of PUST, we conduct experiments on transferring update signals in both the Math and Code domains. In the Math domain, the proxy model (Qwen3-4B/1.7B) is trained for 500 steps using GRPO on the DeepMath-103K dataset [18] [2] to explore update signals, ultimately yielding the optimized proxy model, Qwen3-4B/1.7B Math (RL). Similarly, in the Code domain, the proxy model Qwen3-4B is trained for 300 steps via

[1]The lower LCB score of Qwen3-8B relative to Qwen3-4B is an inherent property of the base models. We strictly followed the evaluation protocol of [17] to ensure reproducibility.

[2]We filter this dataset using the selection strategy proposed in prior work.

Table 1: **Mean@16 results of math benchmarks**. Absolute scores and gains (Δ) over respective base models are reported in parentheses. **Bold+Teal** indicates that the best gains. In PUST, → indicates that update signals are extracted from the proxy model (🎓) on the left and transferred to the primary model (👤) on the right.

Model	AIME 24 [10]	AIME 25 [11]	HMMT 25 (Feb) [12]	HMMT 25 (Nov) [12]	Average
1.7B Parameter Scale					
Qwen3-1.7B (Base)	14.2	12.3	4.4	4.8	8.9
Qwen3-1.7B Math (RL)	35.6 (+21.4)	**31.7** (+19.4)	19.6 (+15.2)	15.2 (+10.4)	25.5 (+16.6)
4B Parameter Scale					
Qwen3-4B (Base)	23.2	22.1	10.6	7.9	16.0
Qwen3-4B Math (RL)	58.8 (+35.6)	**55.8 (+33.7)**	33.1 **(+22.5)**	38.3 (+30.4)	46.5 **(+30.5)**
8B Parameter Scale					
Qwen3-8B (Base)	24.4	22.5	12.3	10.0	17.3
PUST (1.7B 🎓 → 👤 8B)	54.4 (+30.0)	40.2 (+17.7)	22.1 (+9.8)	31.3 (+21.3)	37.2 (+19.9)
PUST (4B 🎓 → 👤 8B)	**62.5 (+38.1)**	52.7 (+30.2)	**34.2** (+21.9)	**40.4 (+30.4)**	**47.5** (+30.2)

Table 2: **Mean@16 results of code benchmarks**. Absolute scores and gains (Δ) over respective base models are reported in parentheses. **Bold+Teal** indicates that the best gains.

Model	HumanEval+ [13, 14]	MBPP+ [15]	LCB [16]	Average
4B Parameter Scale				
Qwen3-4B (Base)	79.4	64.1	18.0	53.8
Qwen3-4B Code (RL)	82.5 **(+3.1)**	68.7 **(+4.6)**	19.0 (+1.0)	56.7 (+2.9)
8B Parameter Scale				
Qwen3-8B (Base)	80.5	70.6	16.7[1]	55.9
PUST (4B 🎓 → 👤 8B)	**83.1** (+2.6)	**73.5** (+2.9)	**25.0 (+8.3)**	**60.5 (+4.6)**

GRPO on the Eurus-RL-Code dataset [19], resulting in Qwen3-4B Code (RL). The update signals explored in both domains are then transferred to the primary model Qwen3-8B. During signal transfer, to account for the differences between the proxy and primary models, we apply a calibration coefficient of $\lambda = 1.5$ for PUST (1.7B 🎓 → 👤 8B), and a calibration coefficient of $\lambda = 1.0$ for PUST (4B 🎓 → 👤 8B).

The experimental results are presented in Table 1 and Table 2. In both domains, the update signals explored by the Qwen3-4B/1.7 proxy model successfully improve the Qwen3-8B primary model, which achieves consistent performance gains across all evaluated datasets. Notably, on certain benchmarks such as AIME2024, HMMT25(NOV), and LCB, the performance improvements observed in the primary model even surpass the gains achieved by the proxy model itself. This demonstrates that update signals can be effectively transferred across models. Furthermore, it suggests that the efficacy of the transferred signals is influenced by the capacity of the primary model, sometimes yielding even better results than when applied directly to the proxy model.

4.2 Reusability of Proxy Update Signal

PUST decouples the exploration of update signals from the primary model. This decoupling allows the update signals to be stored within a proxy model pair in the form of ***on-policy distributional differences***, enabling the update signals to be independently saved and reused. As shown in Table 3, the proxy model Qwen3-4B explores update signals through 500 steps of GRPO training. These signals are inherently captured by the proxy pair comprising Qwen3-4B Math (RL) and the base Qwen3-4B, and can be transferred to different primary models in an on-policy manner. It can be observed that applying the exact same update signals to various primary models—namely Qwen3-1.7B, Qwen3-4B, and Qwen3-8B—consistently yields performance improvements. In terms of training steps, the cost of reusing update signals is substantially lower than the initial cost of exploring them, thereby significantly enhancing the efficiency of the post-training phase. When transferring the signals to these three primary models, we use a calibration coefficient of $\lambda = 1.0$ in all cases.

4.3 Transitivity of Proxy Update Signal

The decoupling of update signals from the primary model not only enables their reusability but also allows them to be transitively transferred across different primary models. As shown in Table 4, we first employ Qwen3-4B as the

Table 3: **Mean@16 results of math benchmarks**. Absolute scores and gains (Δ) over respective base models are reported in parentheses. **Bold+Teal** indicates that the best gains. Update signal explored by one proxy model can be transferred to different primary models in few training steps.

Model	AIME 2024	AIME 2025	HMMT 25 (Feb)	HMMT 25 (Nov)	AVG.	Training Steps
Update signal obtained by GRPO						
Qwen3-4B Math (RL)	58.8 (+35.6)	**55.8** **(+33.7)**	33.1 **(+22.5)**	38.3 (+30.4)	46.5 (+30.5)	500
Update signal transferred to different models						
PUST (4B → 1.7B)	36.5 (+22.3)	30.0 (+17.7)	18.3 (+13.9)	17.5 (+13.1)	25.6 (+16.7)	50
PUST (4B → 4B)	60.7 (+37.5)	55.0 (+32.9)	32.4 (+21.8)	37.9 (+30.0)	46.5 **(+30.5)**	50
PUST (4B → 8B)	**62.5** **(+38.1)**	52.7 (+30.2)	**34.2** (+21.9)	**40.4** **(+30.4)**	**47.5** (+30.2)	50

Table 4: **Mean@16 results of math benchmarks**. Absolute scores and gains (Δ) over the base model are reported in parentheses. The best and second best absolute scores are highlighted in **bold** and underline.

Model	AIME 2024	AIME 2025	HMMT 25 (Feb)	HMMT 25 (Nov)	Average
Qwen3-8B (Base)	24.4	22.5	12.3	10.0	17.3
PUST (1.7B → 8B)	50.8 (+26.4)	42.1 (+19.6)	22.9 (+10.6)	30.6 (+20.6)	36.6 (+19.3)
PUST (4B → 8B)	**62.5** (+38.1)	**52.7** (+30.2)	**34.2** (+21.9)	**40.4** (+30.4)	**47.4** (+30.1)
PUST (4B → 1.7B → 8B)	60.6 (+36.2)	49.6 (+27.1)	28.5 (+16.2)	36.7 (+26.7)	43.9 (+26.6)

proxy model to explore update signals and transfer them to Qwen3-1.7B. Subsequently, we transfer these signals from Qwen3-1.7B to Qwen3-8B, resulting in the PUST (4B → 1.7B → 8B) model. Experimental results demonstrate that even after successive transfers, the update signals still effectively enhance the performance of the final primary model. However, it is worth noting that the update signals may experience a certain degree of drift or deviation during this transitive process. Throughout the transfer process from the 4B model to the 1.7B model and subsequently to the 8B model, we use the same calibration coefficient of $\lambda = 1.0$.

4.4 Sensitivity Analysis of Proxy Models and Calibration Coefficients

Building on the previous experiments, we show that proxy-explored update signals can improve the primary model and be transferred or reused. To study the transfer behavior, we conduct a sensitivity analysis over model scale and calibration coefficient. Specifically, we use Qwen3-1.7B and Qwen3-4B as proxy models, train each with GRPO on DeepMath103K for 500 steps, and transfer the resulting signals to Qwen3-8B with different calibration coefficients.[3]

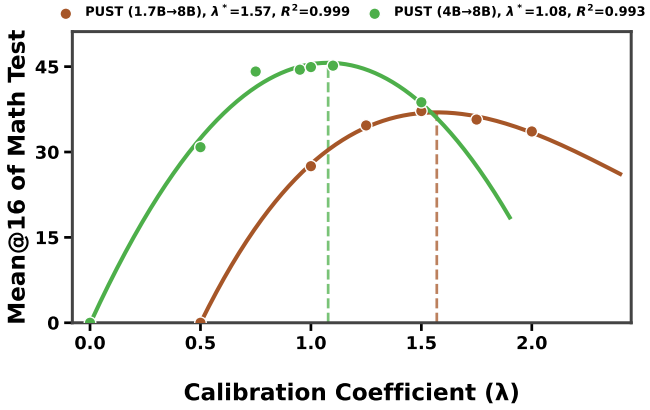

Figure 4: **Sensitivity analysis on different proxy models and calibration coefficients**. Each curve is fitted to the experimental results using a cubic polynomial. A value of R_2 closer to 1 indicates a better fit.

As shown in Fig. 4, the best performance is not achieved at $\lambda = 1.0$ for either proxy model. The optimal coefficients are $\lambda^* = 1.08$ for Qwen3-4B and $\lambda^* = 1.57$ for Qwen3-1.7B, both greater than 1, which suggests that the signals should be downscaled during transfer. In addition, Qwen3-4B achieves a higher peak score than Qwen3-1.7B, indicating that different proxy models can produce signals of different quality.

Although we obtain the best coefficient for each proxy model via polynomial fitting, the relationship between the proxy and primary models varies across sampled states. Exploring adaptive calibration coefficients is therefore an important direction for future work.

[3]Here we report the average results over all test sets; see Section B.2 for details.

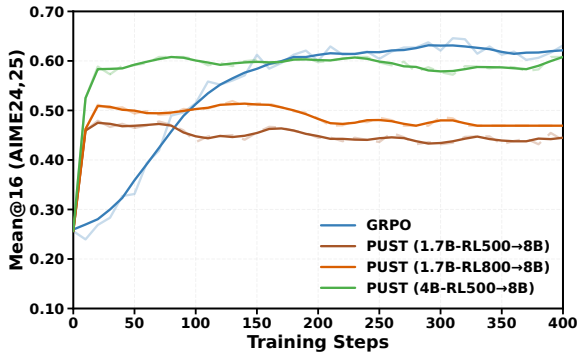

Figure 5: **Mean@16 trajectories on AIME2024 and AIME2025**. All PUST converge within 70 steps and are extended to 400 steps based on converged-region statistics (i.e. mean and std).

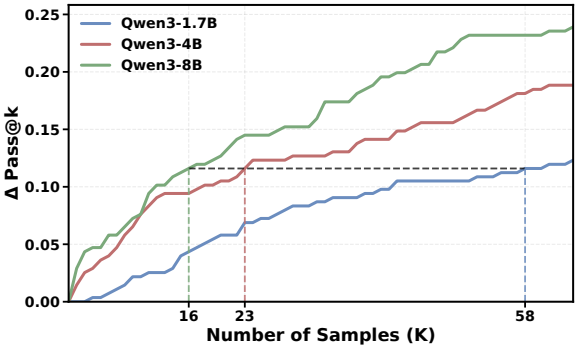

Figure 6: **Δ Pass@K (pass@k - pass@1) over the number of samples k**. As the number of model samples increases, accuracy steadily improves, indicating that more useful information is being extracted.

4.5 Comparison of Exploration Quality Between Proxy and Primary Models

The update signals explored by a proxy model improve the primary model, but an important question remains: how large is the gap between signals explored by a proxy model and those explored directly by the primary model? To answer this, we conduct a comparison on Qwen3-8B. The primary model is trained with GRPO for 400 steps on DeepMath-103K. The proxy model Qwen3-4B is trained for 500 GRPO steps, while Qwen3-1.7B is trained in two variants, with 500 and 800 GRPO steps, respectively. To account for the scale gap between proxy and primary models, we use a calibration coefficient of $\lambda = 1.0$ for Qwen3-4B and $\lambda = 1.5$ for Qwen3-1.7B when transferring update signals to Qwen3-8B. The mean@16 trajectories are shown in Fig. 5. Overall, signals explored by proxy models remain slightly weaker than those discovered directly by the primary model, and larger proxy models tend to transfer better.

Under the same training data and a comparable exploration budget, Qwen3-4B achieves performance close to Qwen3-8B, suggesting diminishing returns from increasing model scale. Since the training data is fixed, the quality of discoverable update signals is inherently bounded. Meanwhile, the 800-step Qwen3-1.7B model outperforms its 500-step counterpart, indicating that longer exploration improves signal quality. The results in Fig. 6 further support this conclusion: with more exploration, smaller models can approach the performance of larger ones.

These results suggest that both **training data** and **exploration strategy** are critical to post-training performance. Because PUST decouples signal exploration from the primary model, it enables more flexible strategies such as multi-model and parallel exploration. In addition, signal reusability lowers the cost of exploration, making it more practical to invest in stronger search over a fixed dataset.

5 Conclusion

In this work, we introduced Proxy-guided Update Signal Transfer (PUST), an asynchronous and highly reusable framework for large language model post-training. By fundamentally decoupling the exploration of update signals from the primary model being trained, PUST enables these signals to be independently extracted, cached, reused, and seamlessly transferred across diverse target models. Systematic evaluations on the Qwen3 family across math and code domains validate the efficacy of this approach. Crucially, our experiments demonstrate that update signals explored by substantially weaker proxy models can effectively improve stronger primary models, with the transfer intensity remaining flexibly adjustable via a calibration coefficient. Ultimately, PUST transforms post-training into a highly modular, scalable, and cost-efficient paradigm, offering flexibility in signal exploration and alignment.

6 Discussion and Future Works

PUST provides a novel lens to revisit the relationship between training data and policy models. As illustrated in Fig. 7, it abstracts post-training into two sequential processes: extracting environmental data into a transferable signal, and subsequently applying this signal to a target primary model.

Update-Signal Extraction (Data / Environment → Transferable Update Signal):

Figure 7: Revisiting the relationship between data/environment and models. By decoupling the exploration process from the model, update signals can be stored and reused independently. To maximally compress the useful information in the data, optimizing the exploration strategy of proxy models becomes a valuable research direction. Moreover, the adaptive correction of update signals during transfer is crucial, as it directly affects transfer efficiency.

- **Mechanism:** This stage utilizes a proxy model to explore the environment via reward optimization. It functions as task-specific information compression, distilling useful supervision from the data into reusable distributional improvement signals.
- **Bottleneck:** Signal quality is intrinsically bounded by the data distribution, the proxy exploration strategy, and the signal aggregation mechanism. Optimizing these factors, e.g., via multi-model parallel exploration or ensemble multi-source voting, can significantly raise this upper bound.
- **Advantage:** Ultimately, this phase unlocks the highly efficient paradigm of "compressing once and reusing many times."

Adaptive Signal Transfer (Update Signal $\rightarrow$ Primary Model):

- **Mechanism:** This stage transfers the extracted signal to the primary model to realize policy optimization.
- **Bottleneck:** The primary challenge is the distributional gap between the proxy and primary models. Although a constant calibration coefficient provides a solid baseline, variations in the dynamic trajectory, such as logit overlap and local entropy, significantly affect transfer efficiency. Overcoming this requires trajectory-adaptive calibration methods that dynamically adjust transfer intensity based on local distributional relationships.
- **Advantage:** Despite these challenges, this mechanism successfully enables robust cross-model transfer and high signal reusability.

Overall, our framework demonstrates the theoretical and practical feasibility of asynchronous exploration and the reusable transfer of update signals. We leave the design of stronger *proxy exploration strategies* and more sophisticated *adaptive calibration mechanisms* as promising directions for future research.

References

[1] Hanyu Lai, Xiao Liu, Junjie Gao, Jiale Cheng, Zehan Qi, Yifan Xu, Shuntian Yao, Dan Zhang, Jinhua Du, Zhenyu Hou, et al. A survey of post-training scaling in large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2771–2791, 2025.

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

[3] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[4] Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, volume 2024, pages 21246–21263, 2024.

[5] Mingyang Song and Mao Zheng. A survey of on-policy distillation for large language models. *arXiv preprint arXiv:2604.00626*, 2026.

[6] Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A Smith. Tuning language models by proxy. *arXiv preprint arXiv:2401.08565*, 2024.

[7] Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, et al. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. *arXiv preprint arXiv:2312.09390*, 2023.

[8] Shiyuan Feng, Huan-ang Gao, Haohan Chi, Hanlin Wu, Zhilong Zhang, Zheng Jiang, Bingxiang He, Wei-Ying Ma, Ya-Qin Zhang, and Hao Zhou. Weak-to-strong generalization via direct on-policy distillation. *arXiv preprint arXiv:2607.05394*, 2026.

[9] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

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

[11] OpenCompass Contributors. Opencompass: A universal evaluation platform for foundation models. https://github.com/open-compass/opencompass, 2023.

[12] Mislav Balunovic, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions. *Advances in Neural Information Processing Systems*, 38, 2026.

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

[14] Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

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

[16] Naman Jain, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, volume 2025, pages 58791–58831, 2025.

[17] Wenkai Yang, Weijie Liu, Ruobing Xie, Kai Yang, Saiyong Yang, and Yankai Lin. Learning beyond teacher: Generalized on-policy distillation with reward extrapolation. *arXiv preprint arXiv:2602.12125*, 2026.

[18] Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, et al. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. *arXiv preprint arXiv:2504.11456*, 2025.

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

[20] Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

[21] Bo Adler, Niket Agarwal, Ashwath Aithal, Dong H Anh, Pallab Bhattacharya, Annika Brundyn, Jared Casper, Bryan Catanzaro, Sharon Clay, Jonathan Cohen, et al. Nemotron-4 340b technical report. *arXiv preprint arXiv:2406.11704*, 2024.

[22] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

[23] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pages 39578–39601, 2024.

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

[25] Runze Liu, Junqi Gao, Jian Zhao, Kaiyan Zhang, Xiu Li, Biqing Qi, Wanli Ouyang, and Bowen Zhou. Can 1b llm surpass 405b llm? rethinking compute-optimal test-time scaling. *arXiv preprint arXiv:2502.06703*, 2025.

[26] Deqing Fu, Tong Xiao, Rui Wang, Wang Zhu, Pengchuan Zhang, Guan Pang, Robin Jia, and Lawrence Chen. Tldr: Token-level detective reward model for large vision language models. In *International Conference on Learning Representations*, volume 2025, pages 45205–45236, 2025.

[27] Eunseop Yoon, Hee Suk Yoon, SooHwan Eom, Gunsoo Han, Daniel Nam, Daejin Jo, Kyoung-Woon On, Mark Hasegawa-Johnson, Sungwoong Kim, and Chang Yoo. Tlcr: Token-level continuous reward for fine-grained reinforcement learning from human feedback. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14969–14981, 2024.

[28] Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Ren Lu, Thomas Mesnard, Johan Ferret, Colton Bishop, Ethan Hall, Victor Carbune, and Abhinav Rastogi. Rlaif: Scaling reinforcement learning from human feedback with ai feedback. 2023.

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

[30] Guanting Dong, Hongyi Yuan, Keming Lu, Chengpeng Li, Mingfeng Xue, Dayiheng Liu, Wei Wang, Zheng Yuan, Chang Zhou, and Jingren Zhou. How abilities in large language models are affected by supervised fine-tuning data composition. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 177–198, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[31] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[32] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International journal of computer vision*, 129(6):1789–1819, 2021.

[33] Yoon Kim and Alexander M Rush. Sequence-level knowledge distillation. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 1317–1327, 2016.

[34] Shicheng Tan, Weng Lam Tam, Yuanchun Wang, Wenwen Gong, Shu Zhao, Peng Zhang, and Jie Tang. Gkd: A general knowledge distillation framework for large-scale pre-trained language model. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, pages 134–148, 2023.

[35] Dan Busbridge, Amitis Shidani, Floris Weers, Jason Ramapuram, Etai Littwin, and Russ Webb. Distillation scaling laws. *arXiv preprint arXiv:2502.08606*, 2025.

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

[37] Yaxuan Li, Yuxin Zuo, Bingxiang He, Jinqian Zhang, Chaojun Xiao, Cheng Qian, Tianyu Yu, Huan-ang Gao, Wenkai Yang, Zhiyuan Liu, et al. Rethinking on-policy distillation of large language models: Phenomenology, mechanism, and recipe. *arXiv preprint arXiv:2604.13016*, 2026.

[38] Dongxu Zhang, Zhichao Yang, Sepehr Janghorbani, Jun Han, Andrew Ressler II, Qian Qian, Gregory D Lyng, Sanjit Singh Batra, and Robert E Tillman. Fast and effective on-policy distillation from reasoning prefixes. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 25553–25569, 2026.

[39] Tianzhu Ye, Li Dong, Zewen Chi, Xun Wu, Shaohan Huang, and Furu Wei. Black-box on-policy distillation of large language models. *arXiv preprint arXiv:2511.10643*, 2025.

[40] Jitao Sang, Yuhang Wang, Jing Zhang, Yanxu Zhu, Chao Kong, Junhong Ye, Shuyu Wei, and Jinlin Xiao. Improving weak-to-strong generalization with scalable oversight and ensemble learning. *arXiv preprint arXiv:2402.00667*, 2024.

[41] Yige Yuan, Teng Xiao, Shuchang Tao, Xue Wang, Jinyang Gao, Bolin Ding, and Bingbing Xu. Incentivizing strong reasoning from weak supervision. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7138–7156, 2026.

[42] Wenhong Zhu, Zhiwei He, Xiaofeng Wang, Pengfei Liu, and Rui Wang. Weak-to-strong preference optimization: Stealing reward from weak aligned model. In *International Conference on Learning Representations*, volume 2025, pages 37116–37144, 2025.

Appendix

Contents

A Related Works

Post-training is a crucial stage in the development of large language models, and post-training algorithms have advanced substantially in recent years [1]. Existing efforts have improved these algorithms from different perspectives, aiming to enhance either training efficiency or final model performance. In this section, we provide a high-level taxonomy and analysis of representative approaches.

A.1 Reward Optimization

Reward-optimization methods constitute an important family of post-training algorithms. Early approaches rely on reward models to provide feedback signals for policy optimization [20, 21, 22]. A central focus of this line of work is improving the granularity and accuracy of reward signals. As the feedback evolves from sequence-level rewards to step-wise rewards [23, 24, 25] and further to token-level rewards [26, 27], the supervision becomes increasingly fine-grained, leading to improved model performance. However, training reliable reward models requires a large amount of high-quality data, which substantially increases the cost of post-training. Constructing training data with the help of AI systems [28, 29] alleviates this burden to some extent and has therefore been widely adopted. Nevertheless, these methods are typically PPO-like algorithms [2], which require loading multiple models during training to stabilize variance and advantage estimation. GRPO [3] reduces this cost by estimating stable advantages through group-wise relative comparisons, thereby significantly lowering the training overhead. Despite these improvements, reward-optimization methods still require on-policy autoregressive trajectory sampling, and each sampled trajectory must be evaluated.

A.2 Distribution Matching

Distribution-matching methods offer a different perspective. Since trajectory sampling is expensive and trajectory evaluation imposes high demands on training data, these methods directly align distributions without explicitly optimizing over sampled trajectories. Supervised Fine-Tuning (SFT) [30] and Knowledge Distillation (KD) [31, 32, 33, 34, 35] are two classical distribution-matching approaches: they align the student model with a target distribution, using either hard targets or soft targets, to improve student performance. DPO [36] further constructs an implicit target distribution from pairs of preferred and dispreferred responses, thereby enhancing the model through preference-based distribution alignment. However, these approaches are offline distribution-matching methods. While they are efficient to train, they may suffer from distribution shift. OPD [4, 37, 17, 5, 38, 39] addresses this issue with an online distribution-matching framework, aligning the student model with the teacher model on trajectories sampled from the student itself. Although OPD still requires autoregressive sampling, distribution-matching methods generally provide more precise credit assignment and thus often converge within fewer training steps. Nevertheless, these methods all depend on access to a stronger target distribution, which introduces an implicit and often substantial training cost.

Table 5: Training hyper-parameters of PUST signal transfer on math domain.

Hyper-parameter	Value
Train Batch Size	256
Rollout n	8
Max Prompt Length	1024
Max Response Length	16,384
Temperature	1.0
Top-p	1.0
LR	1×10^{-6}
KL Coefficient	0.0

Table 6: Training hyper-parameters of PUST signal transfer on code domain.

Hyper-parameter	Value
Train Batch Size	256
Rollout n	8
Max Prompt Length	2048
Max Response Length	16,384
Temperature	1.0
Top-p	1.0
LR	1×10^{-6}
KL Coefficient	0.0

A.3 Weak-to-strong Signal Transfer

Weak-to-strong generalization studies whether supervision from weaker models can elicit stronger capabilities from more capable target models [7]. Recent work extends this question to alignment and reasoning, including scalable oversight or ensemble-based improvements [40], weakly supervised reasoning incentives [41], and preference optimization that reuses implicit rewards from weak aligned models [42]. Direct-OPD [8] is especially related to our setting: it transfers an RL-induced policy shift from a weak model pair to stronger students rather than imitating the weak post-RL teacher's final policy. Our work shares the goal of transferring improvement signals across model scales, while emphasizing proxy-guided signal extraction, cacheable and reusable update signals, and calibrated transfer to different primary models.

Both reward optimization and distribution matching ultimately rely on a high-quality target distribution. Moreover, as the base model becomes stronger, the requirements on the target distribution become increasingly stringent, posing greater challenges to post-training efficiency and scalability. To address this issue, we propose to decouple model optimization from the reward exploration process. Specifically, we optimize the model by extracting and transferring training signals, enabling update signals to be explored asynchronously, stored independently, and reused effectively. This design improves the overall efficiency of post-training.

B Experiment Details

B.1 Implementation Details

Models and framework. All experiments are conducted on the Qwen3 family [9] (1.7B, 4B, 8B), all operating in non-thinking mode. Our implementation is built upon verl [22], and all training is performed on 8×NVIDIA A100 80GB.

Proxy exploration. The proxy experts are trained with GRPO until convergence; hyper-parameters are listed in Table 7. We train on DeepMath-103K [18] for the math domain (filtered following prior work) and on Eurus-RL-Code [19] for the code domain, using a rule-based verifiable reward (1 for correct answers and 0 otherwise).

Signal transfer (PUST). The training hyper-parameters for signal transfer are summarized in Table 5 (math) and Table 6 (code). The calibration coefficient λ is set as described in Section 4.5.

Evaluation protocol. For math benchmarks, we report Mean@16 (the average accuracy over 16 independently sampled responses) across a comprehensive suite of eight datasets: AIME 2024, AIME 2025, AIME 2026, CMIMC 2025, HMMT 2025 (Feb), HMMT 2025 (Nov), HMMT 2026 (Feb), and SMT 2025 [18]. Due to space constraints, the main text reports a representative subset of four benchmarks, while the full results on all eight datasets are detailed in the appendix. For code benchmarks, we report Mean@8 and strictly follow the evaluation scripts of [17].

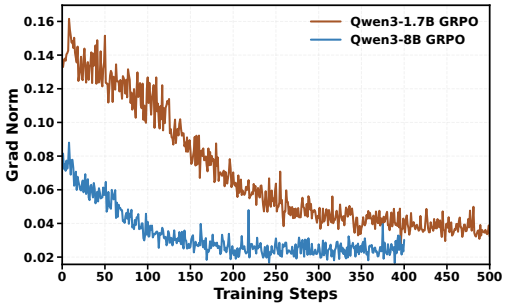

Figure 8: Gradient norm during GRPO training of the proxy expert models.

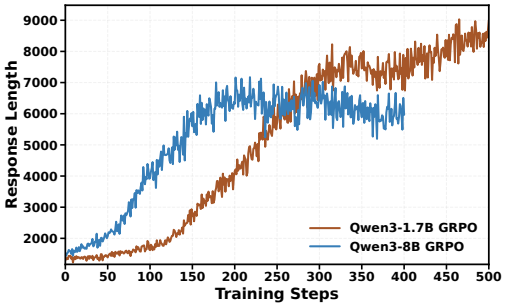

Figure 9: Response length during GRPO training of the proxy expert models.

Table 8: Mean@16 results on different math benchmarks of PUST (4B→8B). Performance comparison across different calibration coefficients λ. The best results are highlighted in **bold**

λ	AIME 24	AIME 25	CMIMC 25	AIME 26	HMMT 25 (Feb)	HMMT 25 (Nov)	SMT 25	HMMT 26 (Feb)	AVG.
0	0	0	0	0	0	0	0	0	0
0.5	57.29	41.88	8.82	45.63	23.75	26.67	12.31	30.43	30.85
0.75	61.25	50.63	27.81	**56.88**	**34.17**	37.71	**52.48**	32.20	44.14
0.95	60.83	50.21	29.22	56.67	31.88	**41.04**	**52.48**	33.33	44.46
1.0	62.50	52.71	28.44	56.67	**34.17**	40.42	50.94	33.52	44.92
1.1	**63.54**	**52.92**	29.53	55.63	32.08	40.83	52.01	34.85	**45.17**
1.5	60.42	48.33	**31.88**	52.29	27.92	37.50	19.76	31.82	38.74

Proxy Expert Model Training Details To ensure that the update signal transferred by PUST is both reliable and high-quality, we train each proxy expert model with GRPO until full convergence, rather than terminating at a fixed budget. The complete set of training hyper-parameters is summarized in Table 7, which we keep consistent across all proxy model scales.

Table 7: Training hyper-parameters of GRPO in math RL.

Hyper-param.	**Value**
Train Batch Size	128
Micro Batch Size	128
Rollout n	8
Max Prompt Length	2048
Max Response Length	16,384
Temperature	1.0
Top-p	1.0
LR	1×10^{-6}
KL Coefficient	0.0

Figure 8 and Figure 9 present the training dynamics of the proxy experts (Qwen3-1.7B and Qwen3-8B) throughout the GRPO process. As shown in Figure 8, the gradient norm decreases steadily and stabilizes at a low plateau ($\sim$0.03) in the later training phase, indicating that the optimization has reached a stable, converged regime. Meanwhile, Figure 9 shows that the response length grows and gradually saturates, reflecting that the model has developed stable reasoning behaviors rather than still actively evolving its output distribution. Together, these curves confirm that our proxy experts are trained to convergence, providing a well-formed update signal for cross-scale transfer.

B.2 Effect of the Calibration Coefficient λ

As discussed in Section 3, the calibration coefficient λ controls how strongly the primary model compensates for the update it has already absorbed: a smaller λ follows the proxy-induced direction more aggressively, while a larger λ yields more conservative transfer. To understand its practical effect, we sweep λ when transferring update signals to the same 8B primary model from two proxy experts of different scales: 4B proxy and 1.7B proxy. We report validation accuracy, policy entropy, gradient norm, and training loss throughout transfer.

4B→8B transfer. Figures 10–13 and Table 8 show the training dynamics when the 4B proxy guides the 8B model. All configurations improve rapidly in the early steps, but the smallest coefficient ($\lambda = 0.5$) is clearly too aggressive: its validation accuracy peaks early and then degrades, and its entropy stays the lowest, indicating premature over-updating. Notably, setting $\lambda = 0$ leads to a

Table 9: Mean@16 results on different math benchmarks of PUST (1.7B→8B). Performance comparison across different calibration coefficients λ. The best results are highlighted in **bold**

λ	AIME 24	AIME 25	CMIMC 25	AIME 26	HMMT 25 (Feb)	HMMT 25 (Nov)	SMT 25	HMMT 26 (Feb)	AVG.
0.5	0	0	0	0	0	0	0	0	0
1.0	41.04	31.88	16.41	32.29	13.54	25.00	35.85	24.05	27.51
1.25	47.71	38.33	23.75	43.13	20.83	**31.88**	42.93	28.79	34.67
1.5	**54.38**	**40.21**	**26.41**	**46.25**	22.08	31.25	**48.23**	28.60	**37.19**
1.75	51.67	39.17	25.47	43.13	**22.50**	30.00	44.10	**29.55**	35.70
2.0	47.71	36.88	25.00	36.88	21.88	29.38	42.22	28.98	33.61

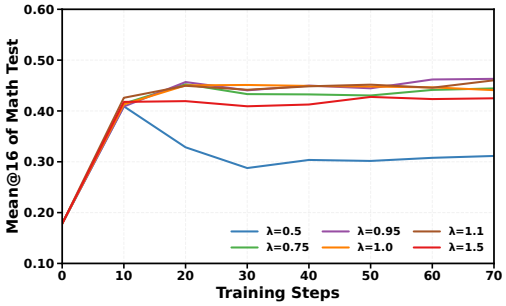

Figure 10: Mean@16 of Math Test under different calibration coefficient λ (4B → 8B).

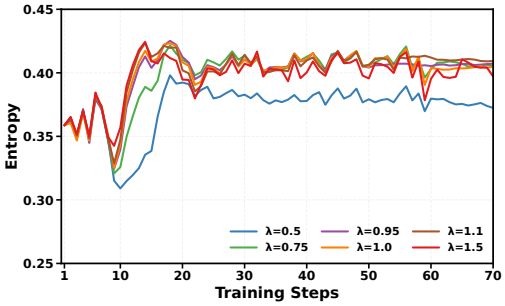

Figure 11: Conditioned entropy of primary model under different λ (4B → 8B).

complete performance collapse, which is likely due to the compensation coefficient being too low to prevent over-optimization. In contrast, moderate-to-large coefficients ($\lambda \in \{0.75, 0.95, 1.0, 1.5\}$) remain stable and converge to consistently higher accuracy, with entropy and gradient norm settling into a smooth plateau. This suggests that a sufficient amount of calibration is needed to avoid over-following the fixed proxy direction.

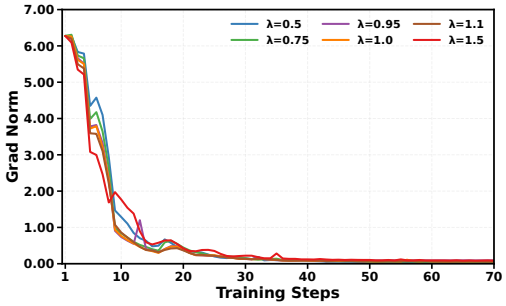

Figure 12: Gradient norm under different calibration coefficient λ (4B → 8B).

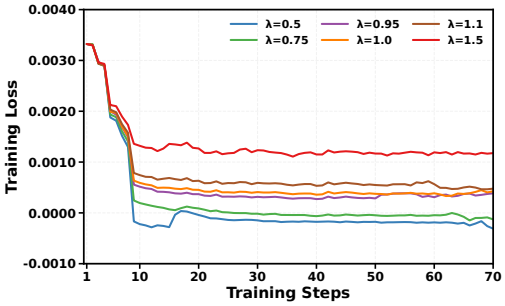

Figure 13: Training loss under different calibration coefficient λ (4B → 8B).

1.7B→8B transfer. Figures 14–17 and Table 9 report the results when the signal comes from the smaller 1.7B proxy. The overall accuracy is lower than in the 4B proxy case, which is expected given the larger capability gap between proxy and primary. Here the trend with respect to λ is even sharper: the most conservative setting ($\lambda = 1.5$) achieves the best and most stable accuracy, while $\lambda = 0.5$ directly leads to a performance collapse. This severe degradation further confirms that an excessively low compensation coefficient cannot effectively regularize the noisier proxy direction, requiring stronger calibration.

Across both settings, the training loss and gradient norm converge stably for all λ, confirming that the transfer objective is well-behaved. The key takeaway is that overly aggressive transfer (small λ) consistently hurts performance, and the optimal degree of calibration increases as the proxy–primary gap widens.

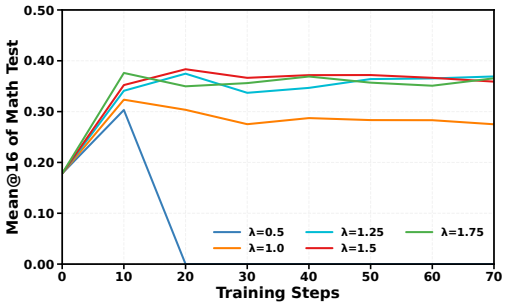

Figure 14: Validation accuracy under different calibration coefficient λ (1.7B → 8B).

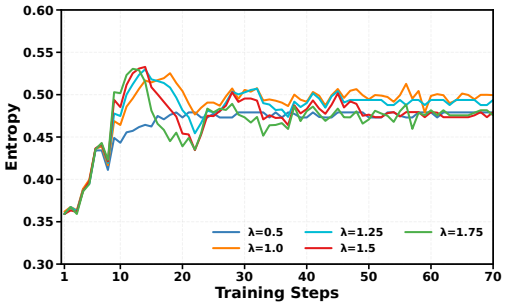

Figure 15: Conditioned entropy of primary model under different λ (1.7B → 8B).

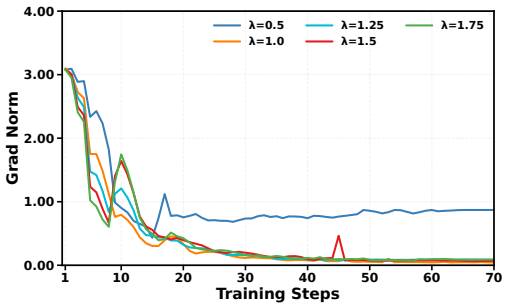

Figure 16: Gradient norm under different calibration coefficient λ (1.7B → 8B).

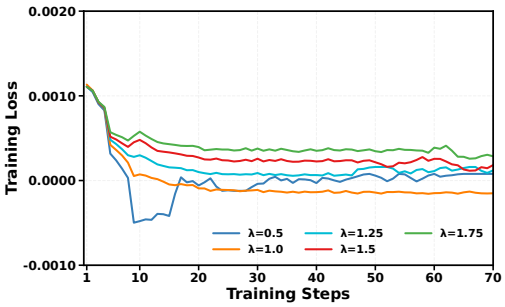

Figure 17: Training loss under different calibration coefficient λ (1.7B → 8B).