ON-POLICY RL MEETS OFF-POLICY EXPERTS: HARMONIZING SUPERVISED FINE-TUNING AND REINFORCEMENT LEARNING VIA DYNAMIC WEIGHTING

Wenhao Zhang, Yuexiang Xie, Yuchang Sun, Yanxi Chen, Guoyin Wang, Yaliang Li,* Bolin Ding, Jingren Zhou
Alibaba Group

ABSTRACT

Supervised Fine-Tuning (SFT) and Reinforcement Learning (RL) are two prominent post-training paradigms for refining the capabilities and aligning the behavior of Large Language Models (LLMs). Existing approaches that integrate SFT and RL often face the risk of disrupting established model patterns and inducing overfitting to expert data. To address this, we present a novel investigation into the unified view of SFT and RL through an off-policy versus on-policy lens. We propose **CHORD**, a framework for the **C**ontrollable **H**armonization of **O**n- and Off-Policy **R**einforcement Learning via **D**ynamic Weighting, which reframes SFT not as a separate stage but as a dynamically weighted auxiliary objective within the on-policy RL process. Based on an analysis of off-policy expert data's influence at both holistic and granular levels, we incorporate a dual-control mechanism in CHORD. Specifically, the framework first employs a global coefficient to holistically guide the transition from off-policy imitation to on-policy exploration, and then applies a token-wise weighting function that enables granular learning from expert tokens, which preserves on-policy exploration and mitigates disruption from off-policy data. We conduct extensive experiments on widely used benchmarks, providing empirical evidence that CHORD achieves a stable and efficient learning process. By effectively harmonizing off-policy expert data with on-policy exploration, CHORD demonstrates significant improvements over baselines. We release the implementation at `https://github.com/modelscope/Trinity-RFT/tree/main/examples/mix_chord` to inspire further research.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities in a wide array of applications, including mathematical problem solving [9, 41, 50], code generation [20, 21, 22], tool use [57], and many more general agentic tasks [12, 32, 47]. This significant progress can be largely attributed to two critical post-tuning paradigms that enhance the performance of LLM in real-world scenarios, i.e., Supervised Fine-Tuning (SFT) [24, 44, 61] and Reinforcement Learning (RL) [35, 41, 42].

While both SFT and RL aim to refine the capabilities of LLM, each paradigm presents its pros and cons. SFT primarily relies on high-quality expert trajectories to effectively mimic response patterns, which can be sensitive to the quality and quantity of expert data [15, 52, 61]. Recent studies also point out that SFT may struggle to generalize beyond mere memorization [8] and is vulnerable to exposure bias [4, 58]. In contrast, RL encourages LLMs to actively explore and enhance their potentially superior behaviors, which enables better generalization through learning from direct feedback on their on-policy generations [6, 8]. However, such explorations can sometimes be inefficient, leading to risks such as policy degradation caused by entropy collapse or over-exploitation of suboptimal strategies [16].

A prevalent and straightforward approach for integrating the strengths of SFT and RL while mitigating their weaknesses is the sequential SFT-then-RL paradigm [25, 30]. Intuitively, on one hand, SFT offers reasoning patterns that can guide exploration in RL to escape local optima [16]. On the other hand, the on-policy learning mechanism in RL can reduce the exposure bias inherent in SFT and prevent overfitting to a limited set of static examples. However, empirical

*Corresponding author. Email to {zwh434786, yaliang.li}@alibaba-inc.com

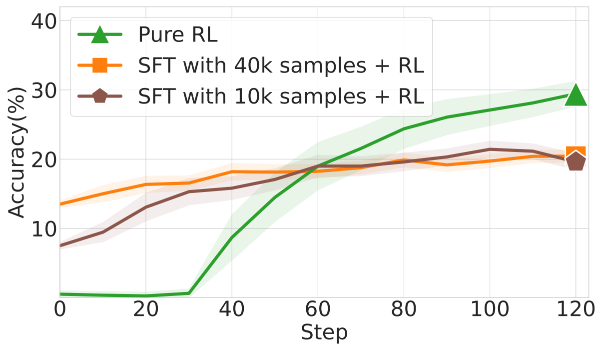

Figure 1: We train Qwen2.5-1.5B-Instruct on the Open-R1 dataset [10] and evaluate the model performance (accuracy) on a held-out validation set. These results show that the SFT-then-RL training paradigm can yield suboptimal performance compared to pure RL.

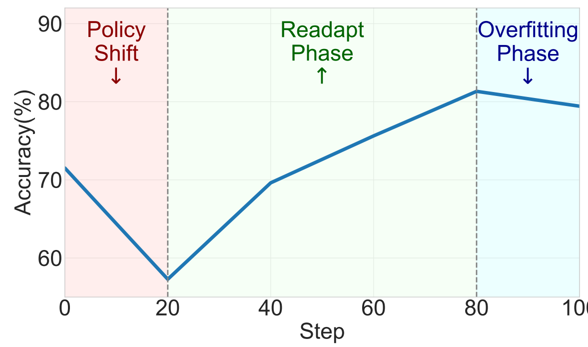

Figure 2: We perform SFT on Qwen2.5-7B-Instruct [51] using expert data generated by Deepseek-R1 [16]. The observed learning curve (measured by accuracy on the MATH-500 dataset) demonstrates a "shift-readapt-overfit" progression.

observations show that the SFT-then-RL paradigm does not consistently outperform the pure RL approach, as illustrated in Figure 1, which is also noted in recent studies [6, 57].

In this study, we further investigate this phenomenon and demonstrate that suboptimal performance may arise from training on expert data that significantly diverges from the model's own established patterns. Specifically, as shown in Figure 2, the learning curve reveals a "shift-readapt-overfit" progression consisting of three distinct phases. Firstly, there is an initial disruption in capability due to the sudden policy shift, which is followed by a readapt phase during which the model adapts to the expert's patterns and recover performance. Finally, we observe that the LLMs eventually overfit to the expert data. These observations highlight that while expert data can bring new capabilities, it may also *disrupt established patterns and induce overfitting during the training process*.

Drawing upon these insights, we propose a novel principle that views SFT and RL through a unified lens of off-policy versus on-policy learning, which reframes SFT not as a separate tuning stage, but as a dynamically weighted auxiliary objective within the on-policy RL process. We further design **CHORD**, a framework for the **C**ontrollable **H**armonization of **O**n- and Off-Policy **R**einforcement Learning via **D**ynamic Weighting. The overall architecture of CHORD can be found in Figure 3, featuring a global coefficient for adjusting the overall influence of expert data throughout the training process (Section 3.2), and a fine-grained, per-token weighting function that helps maintain stability by down-weighting highly divergent tokens from off-policy data that could disrupt on-policy training (Section 3.3). Through such a dynamic weighting mechanism, CHORD enables us to control the influence of off-policy expert data while ensuring training stability when learning from off-policy experts. Extensive experiments demonstrate that CHORD significantly outperforms the compared baselines, achieving a higher performance through a balanced and flexible integration between learning from expert data and maintaining the model's own exploration capabilities.

Our contributions can be summarized as follows:

- We provide a systematic and in-depth analysis of the training dynamics that arise when integrating off-policy expert data into models with established policies during the training process. We identify the "shift-readapt-overfit" progression, revealing how off-policy data disrupt the established reasoning patterns of LLMs.
- We introduce CHORD, a novel framework that unifies SFT and RL via a dynamically weighted auxiliary loss. It consists of a global coefficient and a token-wise weighting function, providing fine-grained control for implementing advanced post-tuning strategies of LLMs.
- Extensive experiments demonstrate that CHORD outperforms the SFT-then-RL paradigm and existing studies. We provide both quantitative and qualitative analysis to show that CHORD strategically navigate training dynamics to selectively absorb expert knowledge without stifling the model's reasoning capabilities. These results highlight the superiority and effectiveness of the proposed framework.

2 Preliminaries

The post-tuning of Large Language Models (LLMs) involves optimizing their policy, denoted as π_θ and parameterized by θ, to generate desirable responses. This typically follows two paradigms: Supervised Fine-Tuning (SFT) and

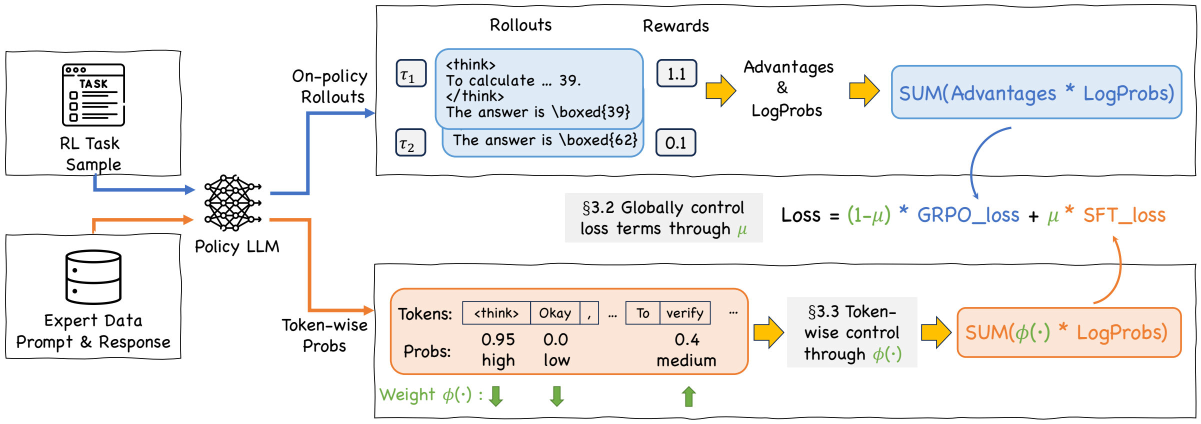

Figure 3: An overview of the proposed CHORD framework that unifies SFT and RL, featuring a global coefficient μ and a token-wise weighting function $\phi(\cdot)$.

Reinforcement Learning (RL). They fundamentally differ in the learning dynamics: SFT is an *off-policy* paradigm driven by a static dataset of expert demonstrations, whereas RL is an *on-policy* paradigm guided by dynamic feedback.

Specifically, SFT adjusts the policy π_θ to mimic a high-quality, static dataset of N expert demonstrations, $\mathcal{D}_{\text{SFT}} = \{(x_i, y_i^*)\}_{i=1}^N$. Here, x_i is a prompt and $y_i^* = (y_{i,1}^*, \dots, y_{i,|y_i^*|}^*)$ is the corresponding expert response of $|y_i^*|$ tokens. The SFT objective is to minimize the negative log-likelihood of the expert responses. In practice, this is optimized using an empirical estimate obtained by averaging the loss across all trajectories in a mini-batch of size B

$$\mathcal{L}_{\text{SFT}}(\theta) = -\frac{1}{\sum_{i=1}^B |y_i^*|} \sum_{i=1}^{B} \sum_{t=1}^{|y_i^*|} \log \pi_\theta(y_{i,t}^* | x_i, y_{i,<t}^*). \tag{1}$$

In contrast, RL optimizes the policy π_θ by maximizing the expected reward from a reward function $R(\tau)$, where τ represents a generated trajectory (x, y^*). A particularly effective setting for tasks with objective correctness criteria, such as code generation or mathematical reasoning, is Reinforcement Learning from Verifiable Rewards (RLVR) [25, 41]. In RLVR, the reward $R(\tau)$ is determined by an automated and verifiable oracle (e.g., a unit test or a symbolic solver).

A prominent policy gradient algorithm for RLVR is *Group Relative Policy Optimization* (GRPO) [41]. For a given prompt x, the algorithm first samples a group of K candidate responses $\{\tau_1, \dots, \tau_K\}$ from a sample policy, π_{sample}, which could be the current policy being optimized (π_θ) or an older policy from a previous optimization step (π_{old}), depending on the settings. Then, each sampled response τ_k is evaluated by the verifiable reward function to obtain its reward, $R(\tau_k)$. The policy LLM π_θ is updated to maximize a PPO-style [40] clipped surrogate objective. Consistent with recent advancements [5, 17, 54], our formulation does not include the KL divergence term, avoiding restricting the performance of the policy LLM. The formal objective is:

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\frac{1}{\sum_{i=1}^{\hat{B}} \sum_{k=1}^{K} |\tau_{i,k}|} \sum_{i=1}^{\hat{B}} \sum_{k=1}^{K} \sum_{t=1}^{|\tau_{i,k}|} \min\left(r_{i,k,t}(\theta) A_{i,k},\ \text{clip}(r_{i,k,t}(\theta), 1-\epsilon, 1+\epsilon) A_{i,k}\right). \tag{2}$$

where $\hat{B}$ is the number of prompts in the mini-batch. The advantage A_k for each response is computed by $A_k = \frac{R(\tau_k) - \mu_{\mathcal{R}}}{\sigma_{\mathcal{R}} + \epsilon_z}$, where $\mu_{\mathcal{R}}$ and $\sigma_{\mathcal{R}}$ are the mean and standard deviation of rewards $\{R(\tau_j)\}_{j=1}^K$ within the group, and ϵ_z is a small constant for numerical stability.

The $r_{i,k,t}(\theta) = \frac{\pi_\theta(\tau_{i,k,t} | x, \tau_{i,k,<t})}{\pi_{\text{sample}}(\tau_{i,k,t} | x, \tau_{i,k,<t})}$ is the token-wise importance sampling (IS) ratio, which serves as a correction factor by re-weighting the probability of actions sampled under π_{sample} to simulate on-policy sampled distribution. The clipping mechanism constrains the policy update to a trusted region around the reference policy. For "strict on-policy training" [30], the ratio should always be 1, and the gradient of $r_{i,k,t}(\theta)$ should be equivalent to $\nabla_\theta \log \pi_\theta(\tau_{i,k,t}^* | x_i, \tau_{i,k,<t}^*)$.

3 CHORD: A Unified Framework that Learns from Off-Policy Expert and On-Policy RL

3.1 The Shift-Readapt-Overfit Progression When Utilizing Off-policy Data

Before introducing CHORD, we first take a close look at the training dynamics of the SFT process, demonstrating how training on off-policy expert data can disrupt the established patterns of LLMs, ultimately leading to the failure of the SFT-then-RL paradigm [6, 57] (such as those results shown in Figure 1).

We train Qwen2.5-7B-Instruct [51] on expert data generated by Deepseek-R1 [16] and monitor the changes in test accuracy on the MATH-500 dataset. The experimental results are shown in Figure 2, from which we can observe that the model performance declines during the first few epochs, followed by a continuous increase to a higher level than that before training, and then it decreases slightly again.

The performance curve reveals a "shift-readapt-overfit" progression that can be understood through three phases:

- *Shift*: The performance initially declines since the model is forced to follow off-policy expert demonstrations whose patterns are significantly different, *disrupting its established patterns and causing a significant performance drop*. Such degradation is further exacerbated by exposure bias [4, 39, 58], as the model, trained exclusively on ground-truth expert data, struggles to navigate the self-generated contexts it encounters during inference.
- *Readapt*: As SFT continues, the model policy π_θ begins to integrate the expert's reasoning patterns and generates responses similar to those of the expert, i.e., entering the readapt phase. The issue of exposure bias can be mitigated by reducing the model's reliance on its own reasoning patterns, allowing performance to steadily rise to a level close to that of the expert. However, it can also hinder the exploration of self-induced paths.
- *Overfit*: Extended training on the limited expert data ultimately leads to overfitting, resulting in a decline in generalization and a significant loss of output diversity. Such overfitting can also restrict the exploratory capacity that is crucial for the following RL optimization.

The aforementioned progression makes it challenging to control the influence of the off-policy expert data. Carefully determining when to transit from SFT to RL requires significant effort; yet such a two-stage paradigm may still yield suboptimal solutions due to the inherent separation of the training phases. This highlights the fragility and limitations of the SFT-then-RL paradigm, especially when the reasoning patterns of expert data diverge significantly from the model's own established pattern.

Drawing upon the above insights, we propose CHORD, a novel framework that effectively unifies SFT and RL. Specifically, the proposed framework consists of a dual-control mechanism: we first introduce a dynamic loss coefficient to balance learning from on- and off-policy data (refer to Section 3.2), then further design a token-wise weighting function that provides fine-grained stability control (refer to Section 3.3). The overall architecture of CHORD is shown in Figure 3.

3.2 Controlling the Influence of Off-policy Expert Data via μ

Firstly, in order to control the influence of off-policy expert data, we propose to reframe SFT as a dynamically weighted auxiliary objective within the on-policy RL process, rather than a separate tuning stage as the SFT-then-RL paradigm. Specifically, we design a combined loss function that minimizes a weighted sum of the RL and SFT losses:

$$\mathcal{L}_{\text{Hybrid}}(\theta) = (1-\mu)\mathcal{L}_{\text{GRPO}}(\theta) + \mu\mathcal{L}_{\text{SFT}}(\theta), \tag{3}$$

where $\mathcal{L}_{\text{GRPO}}(\theta)$ is the empirical GRPO loss defined in Equation (2), $\mathcal{L}_{\text{SFT}}(\theta)$ is the SFT loss defined in Equation (1), and $\mu \in [0, 1]$ is a hyperparameter that balances the influence.

When using a fixed value of μ, the influence of the off-policy expert data remains constant throughout the entire tuning process. Besides, μ can be changed to allow a dynamic balance between off-policy and on-policy learning. For example, the SFT-then-RL pipeline can be regarded as a special case with a binary schedule (initially setting $\mu = 1$ then transitioning to $\mu = 0$), while previous studies [13, 31] that utilize interleaved SFT and RL can be interpreted as employing a periodic μ schedule.

Moving a step forward, applying a decay schedule of μ provides a more graceful and flexible transition from off-policy imitation to on-policy optimization compared to the rigid and binary switch. Such a decay schedule has also proven successful in mitigating exposure bias [4], effectively bridging the distributional gap between training on off-policy samples and performing inference-time rollouts. As shown in Figure 4, the training begins with a high μ, encouraging the model to learn more from off-policy expert data. As training progresses, μ gradually decays to a lower value, shifting the training focus towards on-policy exploration and annealing the influence of the off-policy expert data before overfitting on them.

Beyond the Loss Coefficient μ Empirical comparisons (refer to Section 4 for more details) demonstrate that applying a decay schedule to μ yields significant performance gains over the SFT-then-RL paradigm. At the same time, our observations inspire us to extend beyond μ. Firstly, as shown in Figure 4, the learning curve still reveals a progress similar to "shift-readapt", where the reward initially declines before subsequently increasing. Secondly, the behavior of the trained model, including its reasoning pattern (as shown in Appendix D) and response length (as shown in Table 2), appears to converge to those of the expert. These observations indicate that, despite some improvements, to a certain extent, the utilization of off-policy expert data might still disrupt established patterns and stifle the model's capacity for genuine exploration during on-policy training.

Towards the goal of utilizing off-policy data as an incentive and guidance for the model to explore novel and effective reasoning paths, rather than merely as a target to imitate, we further integrate CHORD with a token-wise, fine-grained weighting function $\phi(\cdot)$, forming a dual-control mechanism together with the global coefficient μ for controlling the influence of the off-policy expert data.

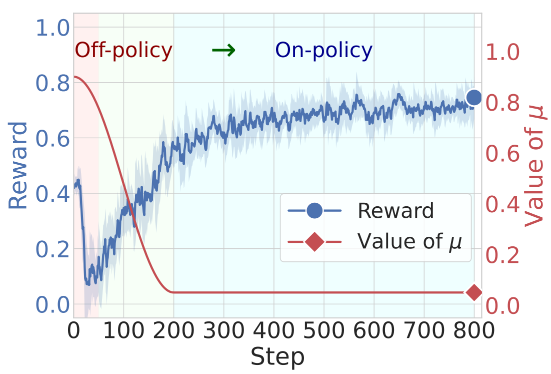

Figure 4: CHORD-μ enables a smooth transition from off-policy imitation to on-policy optimization by decaying the value of μ during the learning process, resulting in a gradual improvement in reward.

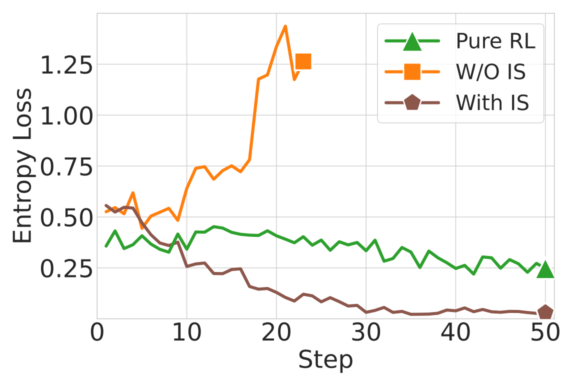

Figure 5: Comparisons of entropy loss between pure RL and mixed RL training that integrates expert data (with or without the IS strategy).

3.3 Enhancing the Stability of Off-policy Learning via $\phi(\cdot)$

A feasible solution for controlling the influence of off-policy expert data from a fine-grained perspective is to differentiate the tokens based on their generation probabilities $\pi(y_t^*|x, y_{<t}^*)$. For example, Importance Sampling (IS) [40] has been widely used for stably integrating off-policy data in RL, which suggests re-weighting the objective by the probability ratio between the target policy π_θ and the behavior policy π_{sample} that generated the expert data. Formally, the objective function can be given as:

$$\mathcal{L}_{\text{SFT-IS}}(\theta) = \mathbb{E}_{(x,y^*)\sim\mathcal{D}_{\text{SFT}}}\left[-\sum_{t=1}^{|y^*|} \text{sg}\left(\frac{\pi_\theta(y_t^*|x, y_{<t}^*)}{\pi_{\text{sample}}(y_t^*|x, y_{<t}^*)}\right) \cdot \log \pi_\theta(y_t^*|x, y_{<t}^*)\right], \tag{4}$$

where the probabilities $\pi_{\text{sample}}(y_t^*|\dots)$ for the expert data $\mathcal{D}_{\text{SFT}}$ are often unknown. It is a common practice [48, 49] to assume that the denominator is 1, treating the expert data as the ground-truth distribution.

From a token-wise perspective, IS enhances training stability by down-weighting low-probability tokens that could disrupt the established policy. As empirical observations shown in Figure 5, mixing off-policy data without IS leads to a sharp rise in entropy, which implies that the model's established patterns are quickly disrupted by the unweighted off-policy data. However, we notice that IS can lead to a sharp collapse in policy entropy compared to pure RL, which implies that it can limit the exploration essential for the RL phase and trap the model in a stable but suboptimal solution. The underlying reason is that IS prevents disruptive shifts in the policy distribution by down-weighting low-probability tokens, but it also aggressively reinforces existing high-probability tokens while ignoring novel but low-probability ones, thus causing the policy to become overconfident.

Stabilize Off-policy Data Training with $\phi(\cdot)$ To tackle this, we propose a novel fine-grained, per-token weighting function $\phi(y_t^*; \pi_\theta)$ for **down-weighting the learning signal for tokens at both ends of the probability spectrum**, i.e., down-weighting those tokens that are already highly probable (to prevent entropy collapse) and those that are extremely improbable (to avoid disruption). To be more specific, we define the weight based on the policy's probability for a given expert token, i.e., $p_t = \pi_\theta(y_t^*|x, y_{<t}^*)$, as follows:

$$\phi(y_t^*; \pi_\theta) = p_t(1 - p_t), \tag{5}$$

which naturally forms a parabolic curve that peaks at $p_t = 0.5$ and decays to zero as p_t approaches 0 or 1. The SFT objective function can be updated as:

$$\mathcal{L}_{\text{SFT-}\phi}(\theta) = -\mathbb{E}_{(x,y^*)\sim\mathcal{D}_{\text{SFT}}}\left[\sum_{t=1}^{|y^*|}\phi(y_t^*; \pi_\theta)\cdot\log\pi_\theta(y_t^*|x, y_{<t}^*)\right], \tag{6}$$

where $\phi(y_t^*; \pi_\theta)$ modulates the gradient contribution of each token in the expert trajectory.

From an information-theoretic perspective, the term $p_t(1-p_t)$ can be viewed as a measure of the policy's uncertainty [45] for the binary event of generating token y_t^*. Therefore, this approach biases learning towards tokens where the policy is most uncertain, and creates a "learning sweet spot" that focuses the off-policy learning on tokens that are novel enough to be informative but not so divergent as to disrupt the established policy.

By replacing the static $\mathcal{L}_{\text{SFT}}$ in the proposed hybrid loss function (defined in Equation (3)) with $\mathcal{L}_{\text{SFT-}\phi}$, we obtain the final objective function of CHORD, which applies a global coefficient μ for adjusting the overall influence of expert data and a fine-grained weighting function $\phi(\cdot)$ that helps enhance the stability when learning from off-policy data.

4 Experiments

4.1 Setup

Datasets and Models We conduct experiments on the OpenR1-Math-220k dataset[2], which includes mathematical problems sourced from NuminaMath1.5 [27] and expert trajectories generated by Deepseek-R1 [16]. We sample 5k instances for SFT and 20k instances for RL, ensuring no overlap between these two sets. More details can be found in Appendix A.

We perform post-tuning on the Qwen2.5-Instruct series [51], specifically using Qwen2.5-7B-Instruct unless otherwise specified, as the response patterns of Qwen2.5-Instruct series (served as the policy models) differ significantly from those of Deepseek-R1 (provided the off-policy expert data).

Evaluations We evaluate the models across the following two categories of capabilities. (i) **In-domain Reasoning Capability**: We measure model performance on widely used mathematics benchmarks, including AIME24, AIME25, and AMC [27], to assess the in-domain generalization ability of models. (ii) **General Reasoning Capability**: We evaluate the models on the MMLU-pro [46] benchmark to assess potential improvements or degradations in general reasoning capability after post-tuning.

Baselines We compare the proposed CHORD with a comprehensive set of baselines, including:

- **Original Model**: The Qwen2.5-7B-Instruct model without any tuning.
- **SFT-only**: The model fine-tuned on the 5k SFT dataset. We focus on two specific configurations: *SFT-light*, trained for a single epoch, and *SFT-best*, the peak-performing checkpoint on the test set found by searching over different learning rates and training epochs.
- **RL-only**: The model fine-tuned directly on the 20k RL dataset using the GRPO algorithm.
- **SFT+RL**: The sequential SFT-then-RL paragidm.
- **LUFFY [49]**: A method that integrates expert demonstrations within the GRPO rollout groups and reshapes the importance sampling ratio. Note that this approach requires a one-to-one mapping between RL samples and expert responses, thus we implement it using the 20k RL dataset and the corresponding 20k expert demonstrations.

We adopt Trinity-RFT [37] as our RL framework, for its ease of implementation and flexible design. Specific hyperparameters and implementation details of the proposed method and baselines can be found in Appendix A.

[2] `https://huggingface.co/datasets/open-r1/OpenR1-Math-220k`

4.2 Comparisons

The proposed approaches implemented based on CHORD include (i) **CHORD-μ**: We employ a decay schedule for the loss coefficient μ to gradually transition from off-policy to on-policy learning, as detailed in Section 3.2; and (ii) **CHORD-ϕ**: We fix the value of μ and further integrate the token-wise weighting function $\phi(\cdot)$ to achieve a dual-control mechanism on the influence of off-policy expert data, as introduced in Section 3.3.

Model Performance Overall, the comparisons regarding model performance, as summarized in Table 1, demonstrate the effectiveness of the proposed CHORD in both in-domain reasoning capability and general reasoning capability.

Table 1: Performance comparisons on mathematical reasoning and general reasoning benchmarks.

	MATH			*General*
Method	**AMC**	**AIME24**	**AIME25**	**MMLU-Pro**
Qwen2.5-7B-Instruct	43.8	11.7	6.66	24.7
SFT-light	42.5	8.54	7.80	28.0
SFT-best	55.9	15.8	15.2	38.4
SFT-light + RL	52.5	11.9	11.6	44.6
SFT-best + RL	58.4	17.1	16.3	51.3
CHORD-μ	60.8	18.1	**17.9**	43.3
GRPO (Pure RL)	52.1	13.2	8.54	45.8
LUFFY[3]	52.8	16.6	14.3	44.0
CHORD-ϕ	**62.5**	**18.2**	17.2	**56.2**

Specifically, from the comparisons between the two different configurations of SFT, we observe that SFT-light, which was trained for one epoch with a learning rate of 1×10^{-6}, performs even worse than the basic policy model. After exhaustively optimizing hyperparameters, we produce SFT-best, which is trained for three epochs with a learning rate of 1×10^{-5} and outperforms Qwen2.5-7B-Instruct. Furthermore, following the SFT-then-RL paradigm, we observe that both SFT-light+RL and SFT-best+RL achieve further improvements over SFT-only, with SFT-best+RL demonstrating a noticeable advantage over SFT-light+RL. These experimental results are consistent with our analysis in Section 3.1, highlighting the necessity and importance of on-policy optimization and the determination of when to transition from SFT to RL.

However, it is worth noting that these SFT-then-RL approaches still achieve suboptimal performance compared to the proposed CHORD-μ, which allows a smooth transition between off-policy and on-policy learning rather than a rigid switch. With a decay μ strategy, CHORD-μ outperforms the strong baseline SFT-best+RL baseline across all MATH benchmarks, achieving improvements of +2.4 on AMC, +1.0 on AIME24, and +1.6 on AIME25, respectively. These results demonstrate the superiority of the design of μ in CHORD for unifying SFT and RL, compared to the SFT-then-RL paradigm.

Further, CHORD-ϕ achieves consistent outperformance over the baselines across both in-domain reasoning datasets and general reasoning datasets. These results demonstrate the effectiveness of our dual-control mechanism in flexibly controlling the influence of off-policy expert data. CHORD-ϕ selectively applies the SFT loss to non-disruptive tokens, integrating expert knowledge without compromising foundational abilities. This enables robust learning from both off-policy expert data and on-policy exploration.

Reasoning Patterns To investigate the influence of expert data on reasoning patterns across different approaches, we summarize the average response length of the proposed methods and baselines, as shown in Table 2 and Figure 6.

Firstly, we observe that the length of the expert data and the responses generated by Qwen2.5-7B-Instruct differ significantly. Both SFT-light and SFT-best show a shift from the original reasoning patterns and tend to generate long responses. After that, RL helps mitigate the issues of overly lengthy responses by training the models to conduct on-policy exploration based on their original and learned reasoning patterns, thereby encouraging the formulation of more efficient answers. We notice that the response length produced by SFT-light+RL is significantly shorter than that of SFT-best+RL, as fewer epochs of SFT allow the model to retain its original reasoning patterns to a certain extent. From Figure 6, we can observe that CHORD-μ exhibits a similar trend to the SFT-then-RL paradigm. The average

[3] The performance using Qwen2.5-7B-Instruct reported in original paper [49] utilizes more data (45k instead of 20k), achieving scores of 50.9 on AMC, 17.7 on AIME24, and 14.8 on AIME25, respectively.

Table 2: Statistics of the average response length for different methods.

Methods	Avg. Length
Expert Data	6,132
Qwen2.5-7B-Instruct	659
SFT-light	9,966
SFT-best	8,442
SFT-light + RL	1,322
SFT-best + RL	4,830
CHORD-μ	6,081
GRPO (Pure RL)	1,423
CHORD-ϕ	2,444

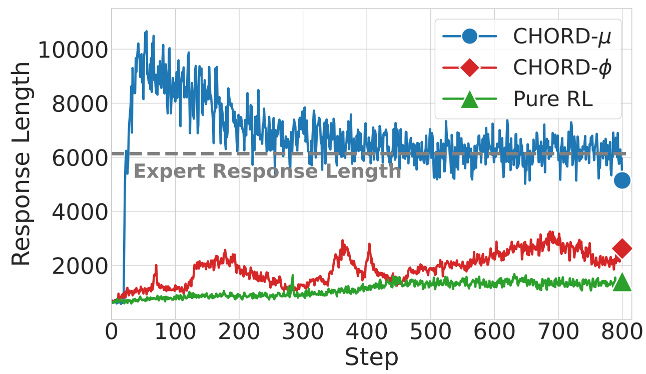

Figure 6: Comparisons of response length for CHORD-μ, CHORD-ϕ, and pure RL.

response length initially increases to align with expert patterns and then gradually converges to a similar length as on-policy training progresses.

On the other hand, instruction models directly trained with RL (i.e., Pure RL denoted in the table and figure) show a relatively slight increase in response length, from 659 to 1,423, a moderate growth compared to "ZeroRL" methods [56] that start from a base model. The proposed CHORD-ϕ exhibits a balance in response length that lies between the SFT-then-RL paradigm and Pure RL. By adjusting the token-wise weight through CHORD-ϕ, the model learns to selectively integrate reasoning patterns from the off-policy expert data. Qualitative analysis shown in Appendix D confirms the effectiveness of such a flexible design, preventing the model from generating overly lengthy responses while strategically incorporating verification steps within its Chain-of-Thought process. The proposed CHORD-ϕ can **go beyond simply mimicking the expert, and can learn to selectively absorb reasoning patterns from the expert, such as verification steps, while exploring its own response strategies**.

4.3 Analysis on the Effects of μ

We provide further discussions and analysis on the effects of the coefficient μ for promoting a flexible transition from off-policy to on-policy learning.

Decreasing Schedule for μ As shown in Figure 7, we decrease the value of μ from 0.9 to 0.05 over the first 200 training steps and keep it unchanged in the following steps. The training curve is depicted in Figure 8. The initial high value of μ ensures that the optimization of the SFT loss plays a dominant role in the early phases, forcing the model to align with off-policy expert data. Meanwhile, it causes a temporary disruption to the model's reward-seeking behavior, leading to an initial drop in rewards. As the training continues, the model begins to readapt the expert's patterns, resulting in a gradual increase in rewards. Notably, as the influence of the SFT loss is progressively reduced by the decreasing μ, the optimization seamlessly shifts focus to the on-policy RL. This allows the model to leverage the learned reasoning patterns for reward maximization while avoiding overfitting on expert data. Finally, with such a dynamic strategy for μ, the proposed method achieves a performance that significantly surpasses pure RL and the sequential SFT-then-RL paradigm.

Dynamic μ Versus Fixed μ In Figure 8, we compare the model performance when applying a dynamic schedule for μ against several fixed schedules in CHORD. The compared μ schedules are shown in Figure 7, and we can observe that applying a fixed μ consistently results in poorer performance compared to dynamic μ. This indicates that naively incorporating off-policy SFT data with a static weight does not effectively serve as a solution for simultaneously learning from off-policy data and on-policy exploration. In fact, it might fail to match Pure RL, which directly encourages an instruction model to follow its own reasoning patterns, highlighting the importance and necessity of controlling the influence of off-policy data.

From the figure, we can also observe that, while using a smaller value of μ (e.g., 0.02) can mitigate the performance degradation compared to larger values (e.g., 0.1 and 0.5), it still fails to provide a significant improvement over pure on-policy RL. With a fixed μ, the model is consistently required to accommodate two potentially divergent reasoning patterns, which might pull it in different directions and prevent it from converging to a stable and high-performance state.

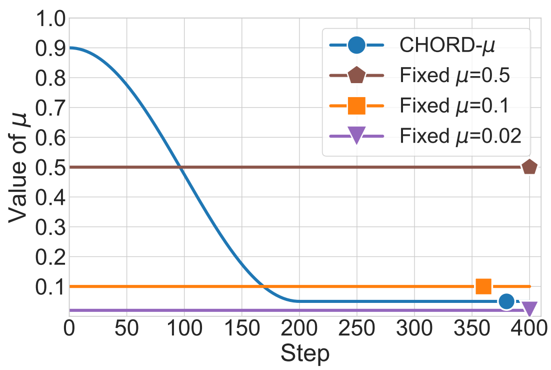

Figure 7: Value of μ versus training step for CHORD-μ and various fixed-μ strategies.

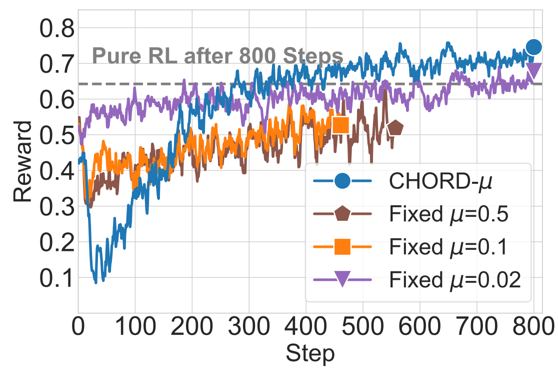

Figure 8: Reward versus training step for CHORD-μ and various fixed-μ strategies.

The decay schedule for μ effectively resolves this conflict by creating a smooth transition from off-policy supervision to on-policy exploration.

4.4 Analysis on the Effects of ϕ

In this subsection, we analyze the effects of the proposed token-wise weighting function $\phi(\cdot)$.

Training Curve of CHORD-ϕ In Figures 9 and Figures 10, we compare the entropy loss and rewards of Pure RL with those of CHORD-ϕ (with fixed μ to 0.1), to illustrate their training dynamics.

From the changes in entropy loss, we can observe that by applying the token-wise function $\phi(\cdot)$, the model maintains a great balance between exploration and exploitation while performing off-policy and on-policy learning simultaneously. On one hand, CHORD-ϕ prevents the entropy from collapsing prematurely, which can occur when the SFT loss forces the model to become over-confident on high-probability tokens from the expert data, showing no entropy collapse. On the other hand, it avoids large entropy spikes and training instability that occur when the off-policy expert data drastically conflict with the current policy's predictions, as the performance curve remains stable throughout the training process. The rewards curve further demonstrates the advantage of CHORD-ϕ, indicating that CHORD-ϕ achieves a stable and continuous increase in rewards, ultimately resulting in significantly better performance than Pure RL. These results further confirm our analysis in Section 3.3, demonstrating that the proposed token-wise weighting function is crucial for effectively unifying the SFT and RL phases.

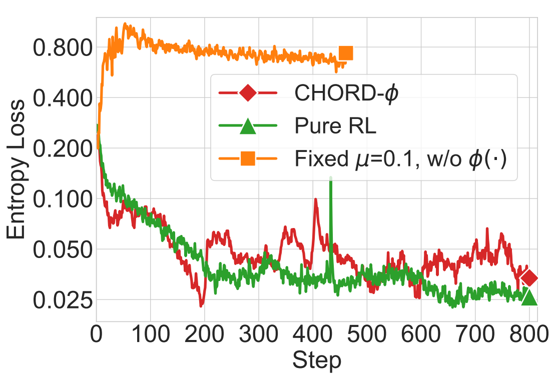

Figure 9: Entropy loss versus training step for CHORD-ϕ and baseline methods.

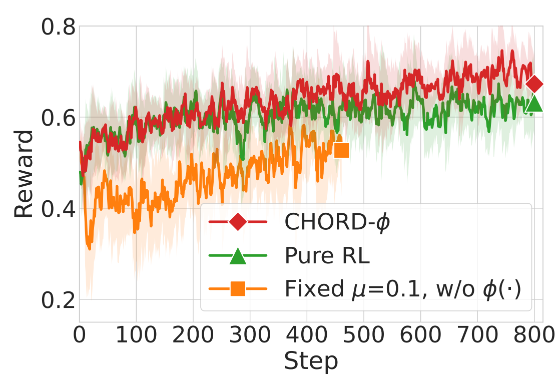

Figure 10: Reward versus training step for CHORD-ϕ and baseline methods.

Principle for Instantiating $\phi(\cdot)$ It is worth noting that the proposed $\phi(\cdot) = p_t * (1 - p_t)$ serves as a concrete and interpretable instantiation following a general principle: stabilizing off-policy integration requires down-weighting the learning signal for tokens at both ends of the probability spectrum. As grounded in our empirical observations, by assigning negligible weight to tokens that the policy is already certain about (where p_t is close to 0 or 1), the proposed method prevents off-policy data from disrupting the model's established reasoning patterns and focuses updates on tokens where the model is still uncertain. Therefore, beyond the specific formulation of $\phi(\cdot)$, this general principle that enables stable and selective learning from off-policy data can potentially inspire more advanced weighting schemes that are suitable for different scenarios.

5 Related Works

Reinforcement Learning for LLM Alignment Reinforcement Learning (RL) has become a key technique for enhancing LLMs, moving beyond early applications in human preference alignment [2, 36]. Recent advancements show significant success in complex reasoning tasks like mathematics and code generation, largely through a paradigm known as Reinforcement Learning from Verifiable Reward (RLVR) [16, 25, 41]. RLVR utilizes definitive outcomes, such as correct final answers or passing unit tests, as reward signals, achieving state-of-the-art results. However, a fundamental limitation persists: RL-based exploration is often constrained by the base model's initial knowledge, making it difficult for the model to discover fundamentally new or superior reasoning pathways on its own [55]. This challenge motivates the integration of external expert data to guide the learning process beyond the model's existing capabilities.

Combining On-policy and Off-policy Learning Incorporating off-policy expert data into the on-policy RL loop is a promising strategy to address the exploration challenge. Existing methods have explored several approaches. Some directly mix expert trajectories into the on-policy rollout groups [49], while others use expert data to guide generation, for instance, by using SFT demonstrations as prefixes for on-policy rollouts [19, 29, 59]. A third category interleaves RL updates with supervised fine-tuning (SFT) steps on expert data, either on a defined schedule [7] or for challenging examples [31]. More recently, SRFT [11] proposed a unified framework that combines data mixing with a sample-level SFT loss. Our work, however, addresses a distinct and more challenging scenario. The aforementioned state-of-the-art methods, including SRFT [11], LUFFY [49], and Reift [31], primarily initiate "ZeroRL" training from a base model. In contrast, we focus on training an instruct finetuned model that already possesses a well-developed policy. This advanced starting point introduces a much larger distributional gap between the model's own policy and the external expert data, significantly exacerbating the off-policy data introduced problems that our method is designed to solve.

For a more comprehensive literature review, please refer to Appendix B.

6 Conclusions

In this work, we identify that the existing SFT-then-RL paradigm can often lead to suboptimal performance due to the disruption of established patterns when utilizing off-policy expert data. This finding motivates us to re-evaluate the separated RL and SFT paradigms through a unified on- versus off-policy lens, framing them not as distinct stages but as integrated components. To realize this unified vision, we propose CHORD. By analyzing the influence of expert data at both the holistic and granular levels, CHORD first integrates a global coefficient μ to manage the overall influence of off-policy expert data, enabling a smoother transition from imitation to exploration. CHORD then introduces a token-wise weighting function, $\phi(\cdot)$, which strategically navigates the selective absorption of expert knowledge while maintaining the model's reasoning patterns, with a general principle of down-weighting tokens that are either already highly probable (to prevent pattern fixation and preserve exploration) or extremely improbable (to avoid policy disruption). We conduct a series of experiments providing both quantitative and qualitative analyses, demonstrating that CHORD selectively learns beneficial patterns from off-policy expert data while exploring its own behaviors throughout the tuning process, achieving significant outperformance compared to the existing SFT-then-RL paradigm.

References

[1] Charles Arnal, GaĂŤtan Narozniak, Vivien Cabannes, Yunhao Tang, Julia Kempe, and Remi Munos. Asymmetric reinforce for off-policy reinforcement learning: Balancing positive and negative rewards. *arXiv preprint arXiv:2506.20520*, 2025.

[2] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

[3] Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*, pages 1577–1594. PMLR, 2023.

[4] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28, 2015.

[5] Aili Chen, Aonian Li, Bangwei Gong, Binyang Jiang, Bo Fei, Bo Yang, Boji Shan, Changqing Yu, Chao Wang, Cheng Zhu, et al. Minimax-m1: Scaling test-time compute efficiently with lightning attention. *arXiv preprint arXiv:2506.13585*, 2025.

[6] Hardy Chen, Haoqin Tu, Fali Wang, Hui Liu, Xianfeng Tang, Xinya Du, Yuyin Zhou, and Cihang Xie. Sft or rl? an early investigation into training r1-like reasoning large vision-language models. *arXiv preprint arXiv:2504.11468*, 2025.

[7] Jack Chen, Fazhong Liu, Naruto Liu, Yuhan Luo, Erqu Qin, Harry Zheng, Tian Dong, Haojin Zhu, Yan Meng, and Xiao Wang. Step-wise adaptive integration of supervised fine-tuning and reinforcement learning for task-specific llms. *arXiv preprint arXiv:2505.13026*, 2025.

[8] Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*, 2025.

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

[10] Hugging Face. Open r1: A fully open reproduction of deepseek-r1, January 2025.

[11] Yuqian Fu, Tinghong Chen, Jiajun Chai, Xihuai Wang, Songjun Tu, Guojun Yin, Wei Lin, Qichao Zhang, Yuanheng Zhu, and Dongbin Zhao. Srft: A single-stage method with supervised and reinforcement fine-tuning for reasoning. *arXiv preprint arXiv:2506.19767*, 2025.

[12] Dawei Gao, Zitao Li, Xuchen Pan, Weirui Kuang, Zhijian Ma, Bingchen Qian, Fei Wei, Wenhao Zhang, Yuexiang Xie, Daoyuan Chen, et al. Agentscope: A flexible yet robust multi-agent platform. *arXiv preprint arXiv:2402.14034*, 2024.

[13] Dechen Gao, Hang Wang, Hanchu Zhou, Nejib Ammar, Shatadal Mishra, Ahmadreza Moradipari, Iman Soltani, and Junshan Zhang. In-ril: Interleaved reinforcement and imitation learning for policy fine-tuning. *arXiv preprint arXiv:2505.10442*, 2025.

[14] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[15] Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, et al. Openthoughts: Data recipes for reasoning models. *arXiv preprint arXiv:2506.04178*, 2025.

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

[17] Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *arXiv preprint arXiv:2503.24290*, 2025.

[18] Xiao Hu, Xingyu Lu, Liyuan Mao, YiFan Zhang, Tianke Zhang, Bin Wen, Fan Yang, Tingting Gao, and Guorui Zhou. Why distillation can outperform zero-rl: The role of flexible reasoning. *arXiv preprint arXiv:2505.21067*, 2025.

[19] Zeyu Huang, Tianhao Cheng, Zihan Qiu, Zili Wang, Yinghui Xu, Edoardo M Ponti, and Ivan Titov. Blending supervised and reinforcement fine-tuning with prefix sampling. *arXiv preprint arXiv:2507.01679*, 2025.

[20] Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

[21] Naman Jain, King Han, 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. *arXiv preprint arXiv:2403.07974*, 2024.

[22] Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.

[23] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[24] Andreas Köpf, Yannic Kilcher, Dimitri Von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, et al. Openassistant conversations-democratizing large language model alignment. *Advances in Neural Information Processing Systems*, 36:47669–47681, 2023.

[25] Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

[26] Jack Lanchantin, Angelica Chen, Janice Lan, Xian Li, Swarnadeep Saha, Tianlu Wang, Jing Xu, Ping Yu, Weizhe Yuan, Jason E Weston, et al. Bridging offline and online reinforcement learning for llms. *arXiv preprint arXiv:2506.21495*, 2025.

[27] 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, 2024.

[28] Tianjian Li and Daniel Khashabi. Simplemix: Frustratingly simple mixing of off-and on-policy data in language model preference learning. *arXiv preprint arXiv:2505.02363*, 2025.

[29] Mingyang Liu, Gabriele Farina, and Asuman Ozdaglar. Uft: Unifying supervised and reinforcement fine-tuning. *arXiv preprint arXiv:2505.16984*, 2025.

[30] Zihan Liu, Zhuolin Yang, Yang Chen, Chankyu Lee, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Acereason-nemotron 1.1: Advancing math and code reasoning through sft and rl synergy. *arXiv preprint arXiv:2506.13284*, 2025.

[31] Lu Ma, Hao Liang, Meiyi Qiang, Lexiang Tang, Xiaochen Ma, Zhen Hao Wong, Junbo Niu, Chengyu Shen, Runming He, Bin Cui, et al. Learning what reinforcement learning can't: Interleaved online fine-tuning for hardest questions. *arXiv preprint arXiv:2506.07527*, 2025.

[32] Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants. In *The Twelfth International Conference on Learning Representations*, 2023.

[33] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[34] Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

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

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

[37] Xuchen Pan, Yanxi Chen, Yushuo Chen, Yuchang Sun, Daoyuan Chen, Wenhao Zhang, Yuexiang Xie, Yilun Huang, Yilei Zhang, Dawei Gao, et al. Trinity-rft: A general-purpose and unified framework for reinforcement fine-tuning of large language models. *arXiv preprint arXiv:2505.17826*, 2025.

[38] Nicolas Le Roux, Marc G Bellemare, Jonathan Lebensold, Arnaud Bergeron, Joshua Greaves, Alex Fréchette, Carolyne Pelletier, Eric Thibodeau-Laufer, Sándor Toth, and Sam Work. Tapered off-policy reinforce: Stable and efficient reinforcement learning for llms. *arXiv preprint arXiv:2503.14286*, 2025.

[39] Florian Schmidt. Generalization in generation: A closer look at exposure bias. *arXiv preprint arXiv:1910.00292*, 2019.

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

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

[42] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[43] Yunhao Tang, Taco Cohen, David W Zhang, Michal Valko, and Rémi Munos. Rl-finetuning llms from on-and off-policy data with a single algorithm. *arXiv preprint arXiv:2503.19612*, 2025.

[44] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7, 2023.

[45] Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning, 2025.

[46] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

[47] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.

[48] Yongliang Wu, Yizhou Zhou, Zhou Ziheng, Yingzhe Peng, Xinyu Ye, Xinting Hu, Wenbo Zhu, Lu Qi, Ming-Hsuan Yang, and Xu Yang. On the generalization of sft: A reinforcement learning perspective with reward rectification, 2025.

[49] Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. Learning to reason under off-policy guidance. *arXiv preprint arXiv:2504.14945*, 2025.

[50] An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

[51] Qwen An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxin Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yi-Chao Zhang, Yunyang Wan, Yuqi Liu, Zeyu Cui, Zhenru Zhang, Zihan Qiu, Shanghaoran Quan, and Zekun Wang. Qwen2.5 technical report. *ArXiv*, abs/2412.15115, 2024.

[52] Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning. *arXiv preprint arXiv:2502.03387*, 2025.

[53] Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Guoyin Wang, Heng Li, Jiangcheng Zhu, Jianqun Chen, et al. Yi: Open foundation models by 01. ai. *arXiv preprint arXiv:2403.04652*, 2024.

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

[55] Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025.

[56] Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *arXiv preprint arXiv:2503.18892*, 2025.

[57] Shaokun Zhang, Yi Dong, Jieyu Zhang, Jan Kautz, Bryan Catanzaro, Andrew Tao, Qingyun Wu, Zhiding Yu, and Guilin Liu. Nemotron-research-tool-n1: Exploring tool-using language models with reinforced reasoning. *arXiv preprint arXiv:2505.00024*, 2025.

[58] Wen Zhang, Yang Feng, Fandong Meng, Di You, and Qun Liu. Bridging the gap between training and inference for neural machine translation. *arXiv preprint arXiv:1906.02448*, 2019.

[59] Xuechen Zhang, Zijian Huang, Yingcong Li, Chenshun Ni, Jiasi Chen, and Samet Oymak. Bread: Branched rollouts from expert anchors bridge sft & rl for reasoning. *arXiv preprint arXiv:2506.17211*, 2025.

[60] Yaowei Zheng, Richong Zhang, Junhao Zhang, YeYanhan YeYanhan, and Zheyan Luo. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 400–410, 2024.

[61] Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36:55006–55021, 2023.

A Additional Experimental Setups and Results

A.1 Hyperparameters

The learning rates are tuned within $\{1 \times 10^{-6}, 5 \times 10^{-6}, 1 \times 10^{-5}\}$ across all the conducted experiments. The batch size for RL is set to 32, with 8 rollouts each prompt. The batch size for SFT is set to 64. For the update steps in RL, we perform "strict on-policy training" similar to the previous study [30], indicating that we generate $K = 8$ rollouts for $B = 32$ prompts, followed by a single policy gradient update. For SFT, we set the maximum number of training epochs to 5, while for RL, the maximum number of steps is set to 1,500. The temperature for both rollout and evaluation is configured at 1.0.

A.2 Implementation Details

In our experiments, a set of hierarchical rules is used to determine the reward for a generated response. A response receives a maximum positive reward of $+1.0$ if it yields the correct final answer. Besides, to encourage structural adherence without penalizing incorrect reasoning during exploration, we assign a neutral reward of 0.0 if the response follows the correct format (e.g., step-by-step reasoning ending with a boxed answer) but the final answer is incorrect. A small penalty of -0.1 is applied if the response is both factually incorrect and improperly formatted. Response exceeding a pre-defined token limit incurs a strong penalty of -1.0.

We implement SFT algorithms based on LLaMA-Factory [60][4], and implement RL algorithms based on Trinity-RFT [37][5]. All the experiments are conducted on 8 NVIDIA A100 GPUs.

For evaluation, we adopt accuracy as the metric. To ensure fair comparisons, we report avg@32 on AIME24 and AIME 25, and avg@8 on AMC, respectively.

A.3 Experimental Results on the MMLU-pro dataset

We provide experimental results on the MMLU-pro dataset in Table 3. The adopted prompt for generating these results can be found in Appendix A.4.

Table 3: Performance comparisons on the MMLU-Pro dataset.

	TAG (by category)														*Average*
Method	Business	Law	Psych.	Biology	Chemistry	History	Other	Health	Econ.	Math	Physics	Comp. Sci.	Philosophy	Engineering	Overall Acc.
Qwen2.5-7B-Instruct	31.18	11.72	23.81	26.22	26.15	20.73	22.40	22.74	25.95	35.75	26.48	25.12	21.84	20.02	24.71
SFT-light	40.56	8.17	21.05	25.52	36.22	14.44	23.38	24.57	27.01	44.63	37.34	28.29	17.43	21.47	28.01
SFT-best	54.50	13.90	31.70	41.98	49.12	21.78	30.84	27.51	40.76	59.29	47.96	42.93	22.85	28.79	38.42
SFT-light + RL	48.80	26.52	51.50	61.09	45.41	41.21	43.72	46.82	52.73	45.89	47.19	46.10	37.68	33.95	44.61
SFT-best + RL	60.84	26.34	51.75	64.02	56.18	40.16	49.57	49.27	57.94	62.10	57.35	51.46	43.09	39.22	51.29
CHORD-μ	55.64	18.71	31.95	43.38	56.18	30.71	34.20	34.60	45.14	64.03	54.81	47.80	28.66	35.81	43.28
GRPO (Pure RL)	56.91	18.35	44.74	58.58	52.30	34.38	41.23	40.22	54.86	57.88	52.19	46.10	37.07	36.02	45.77
LUFFY [49]	52.22	24.25	45.11	54.39	49.29	34.91	41.13	43.40	49.76	54.77	49.42	43.90	32.46	30.13	43.97
CHORD-ϕ	66.79	30.88	60.78	69.87	58.30	45.93	51.19	55.13	66.35	68.47	61.66	53.41	45.89	43.14	56.22

A.4 Prompts

Prompt for Math Problems The adopted prompt for math problems is shown below.

Example: Prompt for Math Problems

<|im_start|>system
You are a helpful assistant that solves MATH problems. You should first think about the reasoning process in mind and then provide the user with the answer. You should present your reasoning process using the format: <think>\n...your reasoning process here... </think>\n first. You should always include your final answer in \boxed{} as closed-form results.<|im_end|>
<|im_start|>user

[4] https://github.com/hiyouga/LLaMA-Factory
[5] https://github.com/modelscope/Trinity-RFT

1. A bus leaves the station at exactly 7:43 a.m. and arrives at its destination at exactly 8:22 a.m. on the same day. How long, in minutes, was the bus trip?<|im_end|>
<|im_start|>assistant

For the performance of the base model, we report the higher score achieved using either the above prompts for math problems or the default Qwen [50] prompt: "Please reason step by step, and put your final answer within \boxed{}".

Prompt for the MMLU-pro Dataset The adopted prompt for the MMLU-pro dataset is shown below. We use the same system prompt as for the math problems, except that for multiple-choice questions, we modify the answer format to require the corresponding integer as the response.

Example: Prompt for MMLU-pro Question

<|im_start|>system
You are a helpful assistant that solves MATH problems. You should first think about the reasoning process in mind and then provide the user with the answer. You should present your reasoning process using the format: <think>\n...your reasoning process here... </think>\n first. You should always include your final answer in \boxed{} as closed-form results.<|im_end|>
<|im_start|>user
Let V be the set of all real polynomials $p(x)$. Let transformations T, S be defined on V by $T : p(x) -> xp(x)$ and $S : p(x) -> p'(x) = d/dxp(x)$, and interpret $(ST)(p(x))$ as $S(T(p(x)))$. Which of the following is true? Below are multiple choice options. You should answer your choice by selecting the index of the option as a number:
0. $ST + TS$ is the identity map of V onto itself.
1. $TS = 0$
2. $ST = 1$
3. $ST - TS = 0$
4. $ST = T$
5. $ST = 0$
6. $ST = TS$
7. $ST - TS$ is the identity map of V onto itself.
8. $TS = T$
9. $ST = S$ <|im_end|>
<|im_start|>assistant

B Detailed Discussions of Related Works

B.1 Finetuning for LLMs

SFT for LLMs. SFT has established itself as a cornerstone for aligning LLMs, primarily due to its conceptual simplicity and cost-effectiveness, making it a favored approach within the open-source community for creating capable instruction-following models [24, 44]. Early work emphasized the power of high-quality datasets [53, 61], while the required expert curation is labor-intensive and costly. Moreover, to cover the diverse use cases of modern LLMs, the paradigm has shifted towards massive-scale SFT [14, 25]. This trend makes it computationally prohibitive for many to fine-tune from a base model, promoting continued tuning on pre-aligned instruction models instead. Furthermore, the interplay between SFT and RL has grown more complex, from recent methods like DFT [48] that incorporate RL-inspired importance sampling into SFT, to reasoning models like DeepSeek-R1 [16] that strategically integrate both paradigms, highlighting that the optimal, principled integration of these methods remains a critical and open area of research.

RL for LLMs. Recent applications of Reinforcement Learning (RL) for Large Language Models (LLMs) have expanded beyond traditional human preference alignment [2, 36], demonstrating significant progress in complex reasoning domains such as mathematics and code generation [16, 41, 50]. In particular, a surge of recent work has focused on Reinforcement Learning from Verifiable Rewards (RLVR) [16, 25], where rewards are derived from definitive outcomes like correct answers or passing unit tests. This paradigm has achieved remarkable results on various benchmarks. However, a fundamental challenge persists in how RL can facilitate effective exploration to surpass the inherent capabilities of its base model [55]. The search for novel solutions is often constrained by the model's pre-existing knowledge, limiting its discovery of superior reasoning pathways. To address this, introducing external expert data—either for distillation [15, 18, 30], cold start[16], or to guide exploration towards diverse, high-quality patterns [31, 49]—emerges as a promising approach to transcend these limitations and unlock new problem-solving frontiers.

B.2 On- and Off-policy Reinforcement Learning

Combining On-policy and Off-policy Data in Traditional RL. In traditional RL domains like robotics [23] or games [33], combining on-policy and off-policy data is a potent strategy. Methods ranging from alternating training phases [13], to mixing data from separate buffers [3], or directly augmenting on-policy replay buffers with expert trajectories [34] have been proven useful. While such methods yield good results in the traditional RL fields, the discrepancy arises from two fundamental distinctions of LLMs: their strong initial priors, where aggressive off-policy updates risk disrupting established reasoning patterns, and their vast, autoregressive action space that radically increases the off-policy degree of expert data, especially for long reasoning chains, and invalidates the assumptions underpinning conventional off-policy algorithms.

Combining On-policy and Off-policy Data in RL for LLM. Leveraging off-policy data to improve the sample efficiency is a well-established strategy in RL. Several studies have focused on leveraging stale, self-generated data by employing techniques such as refining importance sampling corrections [43], mixing on- and off-policy gradients [28], modifying the optimization loss objective [1, 38], or adjusting the synchronization frequency between online and target policies [26].

More closely related to our work are methods that leverage external expert data to guide the reinforcement learning process for LLMs. These methods can be broadly categorized. One strategy is direct data mixing, where LUFFY [49] incorporates off-policy expert trajectories directly into the on-policy rollout groups. While such an approach can also expose the model to expert data, it introduces significant constraints: it requires prompt alignment between datasets, imposing rigid data-sourcing requirements. Another strategy involves using expert data as guidance for generation. For instance, UFT [29] and BREAD [59] utilize supervised fine-tuning (SFT) trajectories as prefixes for on-policy rollouts; UFT progressively masks the suffix of the expert demonstration, while BREAD initiates new rollouts by branching from intermediate steps. A third category interleaves RL updates with SFT steps on expert data, either selectively for challenging examples [31] or based on a probabilistic schedule [7]. Most recently, SRFT [11] unifies these approaches into a single-stage framework by not only mixing SFT samples into the on-policy rollout groups but also applying a dedicated SFT loss whose influence is adjusted at the sample level.

Our work diverges from these methods in a crucial aspect. The aforementioned approaches, including state-of-the-art methods like SRFT [11], LUFFY [49], and Reift [31], primarily operate under a "zero-RL" paradigm, initiating training from a base model with a nascent policy. In stark contrast, our work addresses the challenge of fine-tuning a model that already possesses a well-developed, instruction-following policy. This advanced starting point inherently creates a more significant distributional shift between the model's existing policy and the external expert data, thereby exacerbating the off-policy correction problem that our method aims to solve.

C The Influence of Off-Policy Data on Base v.s. Instruction Models

The challenges of controlling the influence of off-policy data and maintaining training stability are significantly amplified when fine-tuning instruction models. This is largely due to the established policy inherent in these instruction models.

A base model, having been pre-trained solely with a language modeling objective, lacks a coherent, task-specific policy for instruction following. It often has not yet converged on a particular response pattern. When learning from off-policy expert data, the training process is akin to initial policy formation. The model learns a new skill without the risk of conflicting with an existing pattern, thus avoiding significant instability during training.

In contrast, an instruction model has already developed a sharply-peaked policy. Training these models on off-policy expert data that may reflect different reasoning patterns introduces a substantial *distributional mismatch*. The RL algorithm's efforts to reconcile this mismatch can result in large, disruptive policy updates, destabilizing the established policy and potentially leading to a collapse in performance.

Figure 11 provides empirical observation to support the above discussions. When learning from a mixture of on-policy and off-policy data, the reward of a base model improves monotonically, displaying none of the instability issues that can affect instruction models under similar conditions.

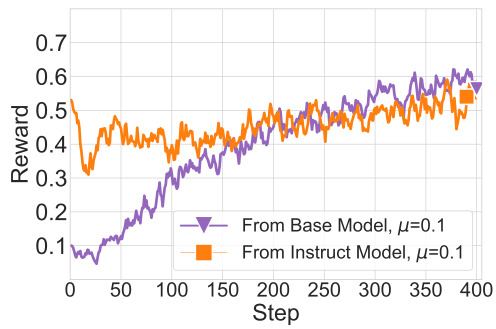

Figure 11: Reward curves for training the base or instruct model with fixed $\mu = 0.1$.

Different from most existing studies, which focus on the "Zero-RL" setting [11, 49, 56] that trains from a base model, this paper addresses a more challenging yet practical problem: how to effectively integrate knowledge from off-policy experts into a model that already possesses an established policy.

D Case Studies

For a better understanding, we compare the generation patterns of RL-only (i.e., pure RL), SFT-only, and the proposed CHORD.

- **RL-only**: The model trained solely with RL exhibits a concise and structured, yet ultimately rigid, reasoning pattern. It tends to follow a fixed template, such as beginning with "To..." and using connectors like "First" and "Next", and proceeds linearly without engaging in self-correction or exploring alternative solution paths. While this approach leads to efficient responses, it may result in less robust solutions.
- **SFT-only and CHORD-μ**: In contrast, the model trained solely on expert demonstrations can be verbose and exploratory. It exhibits a "think-aloud" style with frequent meta-commentary (e.g., "Let me think..."), backtracking, and verification of intermediate steps. The generated responses are comprehensive, but often lack a concise structure. Note that the proposed CHORD-μ has a similar reasoning pattern to SFT-only.
- **CHORD-ϕ**: It exhibits a hybrid reasoning style that retains a clean and logical structure while selectively incorporating the expert's sophisticated verification strategies. As shown in the example below, it develops patterns such as "Alternatively..." and "Both methods confirm..." to produce responses that are both well-structured and robust.

Example: Response generated by Qwen2.5-7B-Instruct

<think>
To calculate the duration of the bus trip, we need to subtract the departure time from the arrival time.

The departure time is 7:43 a.m.
The arrival time is 8:22 a.m.
Subtracting the two times:
- From the hours: 8 - 7 = 1 hour
- From the minutes: 22 - 43 = -21 minutes

Since we can't have negative minutes directly, we borrow 1 hour (60 minutes) from the hour part, making it 0 hours and adding 60 minutes to the 22 minutes. Then, 60 + 22 = 82 minutes. So, we subtract 21 minutes from 82 minutes:
82 - 21 = 61 minutes

Therefore, the bus trip duration is 1 hour and 61 minutes, which is simply expressed as 61 minutes.
</think>
\boxed{61} <|im_end|>

Example: Response generated by the model trained solely with RL

`<think>`
To determine the duration of the bus trip, we need to calculate the difference in time between the departure time and the arrival time. The departure time is 7:43 a.m. and the arrival time is 8:22 a.m. on the same day.

First, let's identify the departure and arrival times:
- Departure time: 7:43 a.m.
- Arrival time: 8:22 a.m.

Next, we need to find the difference between these two times. We can do this by subtracting the departure time from the arrival time.

From 7:43 a.m. to 8:00 a.m. is the time from 7:43 to 8:00. This is 17 minutes because 8:00 - 7:43 = 17 minutes (since 8:00 is 17 minutes after 7:43).

Now we need to add the time from 8:00 a.m. to 8:22 a.m., which is 22 minutes.

So, the total trip time is 17 minutes (from 7:43 to 8:00) + 22 minutes (from 8:00 to 8:22) = 39 minutes.

Therefore, the duration of the bus trip is `\boxed{39}` minutes. `</think>`
The duration of the bus trip is `\boxed{39}` minutes. `<|im_end|>`

Example: Response generated by the model trained solely with SFT

`<think>`
Okay, let me try to solve this problem step by step. Let me think aloud so I can explain my reasoning.

So, the problem is: A bus leaves the station at exactly 7:43 a.m. and arrives at its destination at exactly 8:22 a.m. on the same day. How long, in minutes, was the bus trip?

Let me visualize this. Let me think about the time elapsed from 7:43 a.m. to 8:22 a.m.

Let me break this down. Let me think of it as two parts: from 7:43 a.m. to 8:00 a.m., and then from 8:00 a.m. to 8:22 a.m.

Let me check that. Let me see if that makes sense.

So, from 7:43 a.m. to 8:00 a.m., how many minutes is that?

Let me calculate that. Let me think, from 7:43 to 8:00 is 17 minutes. Let me think, let me count:

7:43 to 8:00 is 17 minutes. Because 8:00 minus 7:43 is 17 minutes.

Let me confirm that. Let me think, 7:43 to 7:50 is 7 minutes, and then 7:50 to 8:00 is 10 minutes. So, 7 + 10 = 17 minutes. Okay, that seems right.

So, that's the first part: 17 minutes from 7:43 a.m. to 8:00 a.m.

Then, the second part is from 8:00 a.m. to 8:22 a.m., which is 22 minutes.

So, adding those two parts together: 17 minutes + 22 minutes = 39 minutes.

Let me check that. Let me think, 17 + 22. Let me do the addition:

17 + 22 = 39.

So, that seems correct.

Let me think if there is another way to calculate this.

Let me think, maybe using a different approach.

Let me think, the total time from 7:43 a.m. to 8:22 a.m.

Let me think, let me calculate the difference between 8:22 and 7:43.

Let me break it down hour by hour.

From 7:43 a.m. to 8:00 a.m. is 17 minutes, as we already knew.

Then, from 8:00 a.m. to 8:22 a.m. is 22 minutes.

So, adding those together gives 17 + 22 = 39 minutes.

Let me think if there is a formula or a quicker way.

Let me think, maybe using the formula:

Total time = (End time hour - Start time hour) * 60 + (End time minute - Start time minute)

Let me try that.

So, let me denote the start time as 7:43 a.m.

End time as 8:22 a.m.

So, End time hour = 8

Start time hour = 7

End time minute = 22

Start time minute = 43

So, Total time = (8 - 7) * 60 + (22 - 43)

Let me calculate that.

First, (8 - 7) * 60 = 1 * 60 = 60 minutes

Then, (22 - 43) = -21 minutes

So, Total time = 60 + (-21) = 39 minutes

Oh, that's interesting! So, using this formula, we get 39 minutes as well.

Let me think, let me check if this makes sense.

So, when we subtract the start time from the end time, we get:

8:22 - 7:43

Let me think of this as 8 hours and 22 minutes minus 7 hours and 43 minutes.

But since we are subtracting times, let me think of it as:

8:22 - 7:43 = (8 - 7) hours + (22 - 43) minutes

But since 22 minutes is less than 43 minutes, we need to borrow an hour.

So, 8 hours is 7 hours and 60 minutes.

So, 7 hours and 60 minutes minus 7 hours and 43 minutes is 0 hours and (60 - 43) minutes = 17 minutes.

Then, the remaining hour is 1 hour.

So, total time is 1 hour + 17 minutes = 60 minutes + 17 minutes = 77 minutes.

Wait, but that doesn't match. Let me think if this is correct.

Let me think, maybe this method is complicating it. Let me think, maybe the first method is simpler and more accurate.

Let me think, let me think of it as total time = end time - start time.

So, end time is 8:22 a.m.

Start time is 7:43 a.m.

So, let me calculate the difference.

From 7:43 to 8:00 is 17 minutes.

From 8:00 to 8:22 is 22 minutes.

So, 17 + 22 = 39 minutes.

So, that seems correct.

Let me think, let me think of it another way.

Let me think, let me convert both times to minutes since midnight and subtract them.

Let me think, that might be complicated, but let me try.

Let me think, 7:43 a.m. is 7 hours and 43 minutes.

So, in minutes, that is 7 * 60 + 43 = 420 + 43 = 463 minutes since midnight.

Similarly, 8:22 a.m. is 8 hours and 22 minutes, which is 8 * 60 + 22 = 480 + 22 = 502 minutes since midnight.

So, the difference is 502 - 463 = 39 minutes.

Oh, so that also gives 39 minutes.

So, this method also confirms that the answer is 39 minutes.

Therefore, I think all methods lead to the same answer, so I can be confident that the bus trip lasted 39 minutes.

Final Answer
The bus trip lasted `\boxed{39}` minutes.

</think>

To determine the duration of the bus trip, we need to calculate the time difference between the departure time (7:43 a.m.) and the arrival time (8:22 a.m.).

1. Calculate the time from 7:43 a.m. to 8:00 a.m.:
- From 7:43 a.m. to 8:00 a.m. is 17 minutes.

2. Calculate the time from 8:00 a.m. to 8:22 a.m.:
- From 8:00 a.m. to 8:22 a.m. is 22 minutes.

3. Add the two intervals together:
- Total time = 17 minutes + 22 minutes = 39 minutes.

Thus, the bus trip lasted `(\boxed{39})` minutes.`<|im_end|>`

Example: Response generated by the model trained with CHORD-μ

<think>
Okay, let me try to solve this problem step by step. Let me think aloud so I can explain my reasoning.

So, the problem is: A bus leaves the station at exactly 7:43 a.m. and arrives at its destination at exactly 8:22 a.m. on the same day. How long, in minutes, was the bus trip?

Let me visualize this. I need to find the time elapsed between 7:43 a.m. and 8:22 a.m.

Let me break this down. A good way to handle time calculations that cross an hour is to use the top of the hour as a stepping stone. In this case, that's 8:00 a.m. So, I can think of the trip in two parts: from 7:43 a.m. to 8:00 a.m., and then from 8:00 a.m. to 8:22 a.m.

Let me check that. Let me see if that makes sense. Yes, that splits the problem into two easier calculations.

So, first part: from 7:43 a.m. to 8:00 a.m. How many minutes is that?
Let me calculate that. An hour has 60 minutes. So, the time from 7:43 to 8:00 is `60 - 43` minutes.
Let me think... `60 - 40` is 20, so `60 - 43` is 17. Yes, 17 minutes.
So, the first part is 17 minutes.

Then, the second part is from 8:00 a.m. to 8:22 a.m. That's straightforward, it's just 22 minutes.

So, to get the total time, I just need to add those two parts together: `17 minutes + 22 minutes`.
Let me do the addition: `17 + 22`.
`10 + 20` is 30, and `7 + 2` is 9. So `30 + 9 = 39`.
The total is 39 minutes. So, that seems correct.

Let me think if there is another way to calculate this to be sure.
Let me try converting both times to minutes past midnight and then subtracting.

Let me think, start time is 7:43 a.m.
In minutes, that is `(7 hours * 60 minutes/hour) + 43 minutes`.
`7 * 60 = 420`.
`420 + 43 = 463` minutes since midnight.

Now the end time: 8:22 a.m.
In minutes, that is `(8 hours * 60 minutes/hour) + 22 minutes`.
`8 * 60 = 480`.
`480 + 22 = 502` minutes since midnight.

So, the difference is `502 - 463`.
Let me calculate that. `502 - 400` is 102. `102 - 60` is 42. `42 - 3` is 39.
So, `502 - 463 = 39` minutes.

Oh, so that also gives 39 minutes. This method confirms my previous answer.

Let me think about one more method, direct subtraction with borrowing.
I want to calculate 8:22 - 7:43.
Let me write it as hours and minutes.
```
8 hours 22 minutes
- 7 hours 43 minutes
```
I can't subtract 43 from 22. So I need to borrow 1 hour (which is 60 minutes) from the 8 hours.
So, 8 hours and 22 minutes becomes 7 hours and `(60 + 22)` minutes, which is 7 hours and 82 minutes.
Now let me subtract:
```
7 hours 82 minutes
- 7 hours 43 minutes
----------
0 hours 39 minutes
```
So, the result is 39 minutes.

All three methods lead to the same answer, so I can be confident that the bus trip lasted 39 minutes.

Final Answer
The bus trip lasted `(\boxed{39})` minutes.
</think>
To determine the duration of the bus trip, we need to calculate the time difference between the departure time (7:43 a.m.) and the arrival time (8:22 a.m.).

We can solve this by breaking the duration into two parts using 8:00 a.m. as a reference point.

1. Calculate the time from 7:43 a.m. to 8:00 a.m.:
   - From 7:43 a.m. to 8:00 a.m. is `60 - 43 = 17` minutes.

2. Calculate the time from 8:00 a.m. to 8:22 a.m.:
   - From 8:00 a.m. to 8:22 a.m. is 22 minutes.

3. Add the two intervals together:
   - Total time = 17 minutes + 22 minutes = 39 minutes.

Thus, the bus trip lasted `(\boxed{39})` minutes.<|im_end|>

Example: Response generated by the model trained with CHORD-ϕ

<think>
To determine the duration of the bus trip, I need to calculate the time difference between the departure and arrival times. The departure time is 7:43 a.m. and the arrival time is 8:22 a.m. on the same day. The key is to find the difference between these two times.

First, I'll visualize the time on a clock face. Starting at 7:43 a.m., the bus arrives at 8:22 a.m. The difference between these two times is 39 minutes (from 7:43 to 8:22).

Alternatively, I can break down the calculation step by step:

1. From 7:43 a.m. to 8:00 a.m. is 17 minutes (since 8:00 - 7:43 = 0:17).
2. From 8:00 a.m. to 8:22 a.m. is 22 minutes.

3. Adding these two intervals together: 17 minutes + 22 minutes = 39 minutes.

Therefore, the total duration of the bus trip is 39 minutes. Let me confirm this with a different approach to ensure accuracy.

Another way to think about it is to calculate the total minutes between 7:43 a.m. and 8:22 a.m.:

1. Convert 7:43 a.m. to minutes since midnight: 7 hours * 60 minutes/hour + 43 minutes = 420 minutes + 43 minutes = 463 minutes.
2. Convert 8:22 a.m. to minutes since midnight: 8 hours * 60 minutes/hour + 22 minutes = 480 minutes + 22 minutes = 502 minutes.
3. The difference in minutes between 502 minutes and 463 minutes is 502 - 463 = 39 minutes.

Both methods confirm that the bus trip lasted 39 minutes. Therefore, the answer is `\boxed{39}`.
`</think>`
To determine the duration of the bus trip from departure at 7:43 a.m. to arrival at 8:22 a.m., we need to calculate the time difference between these two times.

1. First, let's break down the time from 7:43 a.m. to 8:00 a.m.:
- From 7:43 a.m. to 8:00 a.m. is 17 minutes (since 8:00 - 7:43 = 0:17).

2. Next, calculate the time from 8:00 a.m. to 8:22 a.m.:
- From 8:00 a.m. to 8:22 a.m. is 22 minutes.

3. Add these two intervals together:
- 17 minutes + 22 minutes = 39 minutes.

Alternatively, we can convert the times to minutes since midnight:
- 7:43 a.m. is 7 hours * 60 minutes/hour + 43 minutes = 463 minutes.
- 8:22 a.m. is 8 hours * 60 minutes/hour + 22 minutes = 502 minutes.
- The difference is 502 minutes - 463 minutes = 39 minutes.

Both methods confirm that the bus trip lasted 39 minutes. Therefore, the answer is `\boxed{39}.<|im_end|>`