NOT ALL ROLLOUTS ARE WORTH LEARNING: ON TRAJECTORY VALUATION FOR POST-TRAINING REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

We consider the problem of trajectory valuation in reinforcement learning: how to identify and mitigate detrimental trajectories during online training. Unlike classification, where data valuation relies on fixed training and validation sets, reinforcement learning involves dynamically generated trajectories without explicit validation signals, making conventional influence-based methods inapplicable. We propose Dynamic Trajectory Valuation (DTV), a simple and efficient framework that estimates trajectory utility at the mini-batch level and filters detrimental trajectories based solely on gradient information. By operating at the optimization level, DTV integrates seamlessly with existing reinforcement learning pipelines with minimal overhead. Extensive experiments across diverse settings, including GRPO, DPO, and PPO, demonstrate that DTV consistently improves performance, enhances data efficiency, and stabilizes optimization.

1 INTRODUCTION

Reinforcement learning is a paradigm in which an agent learns sequential decision-making policies through interaction with an environment, with the goal of maximizing long-term cumulative rewards (Sutton and Barto, 2018). Unlike classification with fixed labeled datasets, reinforcement learning data—commonly referred to as trajectories—is generated online through the agent's rollouts. As a result, the quality of collected trajectories directly affects learning performance, and not all trajectories contribute positively to policy optimization (Schaul et al., 2015; Shu et al., 2026).

To address this disparity in trajectory quality, adopting a data-centric perspective becomes essential—one that explicitly evaluates and curates trajectories prior to policy updates. Data-centric learning has recently emerged as a promising direction for improving model performance by assessing the contribution of individual samples to a target. A central task in this paradigm is data valuation, which aims to quantify the utility of each sample (Koh and Liang, 2017; Kwon et al., 2023; Park et al., 2023). A straightforward approach is to measure performance differences with and without a sample, but such retraining-based methods are computationally prohibitive. While influence functions offer an efficient alternative by estimating sample impact without retraining—albeit at the cost of inverse-Hessian approximations—their application has largely been restricted to classification, leaving their extension to reinforcement learning underexplored.

Extending influence-based data valuation from classification to reinforcement learning is non-trivial. Classification relies on fixed training and validation sets, enabling valuation through validation performance, whereas reinforcement learning involves dynamically generated trajectories without a fixed validation set. Moreover, trajectory utility in reinforcement learning is inherently policy-dependent and shaped by temporal credit assignment and reward dynamics, making it difficult to define consistent impact measures. Finally, reinforcement learning requires dynamic, online valuation as training progresses, raising additional challenges in efficiency.

Recent work (Hu et al., 2025; Shu et al., 2026; Li et al., 2026; Yang et al., 2026) has taken initial steps toward data valuation and selection in reinforcement learning. Existing methods rely on replay buffers and surrogate impact functions, verifiable feedback, or trusted validation gradients to estimate data utility. However, these setting-specific designs raise concerns about their generalizability across different reinforcement learning paradigms.

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

Contributions. In this paper, we study trajectory valuation in reinforcement learning, aiming to prevent detrimental trajectories from model updates. We summarize our key contributions as follows:

- We provide a principled formulation for trajectory valuation in general reinforcement learning settings, addressing the challenges of dynamic data generation, the absence of validation signals, and policy-dependent utility within a unified framework. This formulation establishes a unified perspective for trajectory valuation across diverse reinforcement learning pipelines.
- We propose Dynamic Trajectory Valuation (DTV), a simple and efficient method that dynamically identifies and filters detrimental trajectories at the mini-batch level. Our method operates purely on gradients and integrates seamlessly with existing reinforcement learning pipelines.
- We conduct extensive experiments to demonstrate the effectiveness of DTV across diverse reinforcement learning settings, including Group Relative Policy Optimization (GRPO), Direct Preference Optimization (DPO), and Proximal Policy Optimization (PPO). The results show consistent performance improvements, enhanced data efficiency, and more stable optimization, validating the benefits of dynamic trajectory valuation in practice.

2 Related Work

A line of work focuses on improving sample efficiency by prioritizing or filtering training data (Bengio et al., 2009; Schaul et al., 2015; Li et al., 2021; Jiang et al., 2021), most commonly at the level of individual transitions (i.e., experiences), as in prioritized experience replay, or via structured data scheduling such as curriculum learning. These approaches implicitly acknowledge that different data points contribute unequally to learning. However, they primarily operate on local signals—such as temporal-difference error or immediate rewards—to estimate importance, without explicitly modeling the contribution of entire trajectories. As a result, they may fail to capture long-range dependencies and the global effect of a trajectory on policy improvement.

Moving beyond transition-level selection, recent approaches in reinforcement learning from human feedback operate directly at the level of complete trajectories or response sequences (Christiano et al., 2017; Ouyang et al., 2022). Methods such as GRPO (Shao et al., 2024), DPO (Rafailov et al., 2023), and PPO (Schulman et al., 2017) treat each rollout as a training unit, where the quality of entire outputs determines the learning signal (See their preliminaries in Appendix A). In practice, data quality is often controlled via heuristic strategies such as rejection sampling or reward thresholding (Bai et al., 2022; Nakano et al., 2021). While effective, these approaches still rely on coarse signals and lack a principled framework for evaluating trajectories based on their contribution to learning.

Recent work (Hu et al., 2025; Shu et al., 2026; Li et al., 2026; Yang et al., 2026) has begun to explore the adaptation of data valuation and selection in reinforcement learning[1]. IIF (Hu et al., 2025) estimates data influence using replay buffers and algorithm-specific surrogate objectives; LearnAlign (Li et al., 2026) weights gradient alignment by learnability derived from verifiable ground-truth feedback; and GradAlign (Yang et al., 2026) evaluates candidate updates against gradients from a trusted validation set. While this line of work represents a promising direction, existing methods often rely on auxiliary signals, which may limit their general applicability across different reinforcement learning paradigms.

Taken together, existing approaches rely on local heuristics, coarse filtering, or setting-specific auxiliary signals. A unified and principled framework for trajectory-level data valuation remains lacking.

3 Method

In this section, we first highlight three challenges when adapting traditional influence functions to reinforcement learning. We then present Dynamic Trajectory Valuation (DTV), a unified framework that overcomes these issues and integrates across various reinforcement learning paradigms. Lastly, we offer an in-depth analysis of DTV by exploring its trajectory decomposition and leave-one-out extension.

Motivation and Challenges. Influence functions, as a crucial tool for data-centric learning, have been extensively studied in supervised learning (Kwon et al., 2023; Park et al., 2023; Pruthi et al.,

[1]We provide additional related work on data valuation and a comparison of these methods in Appendix B.

2020). The effect of an individual training sample can be characterized by infinitesimally perturbing its contribution to the training objective and tracing the resulting change in model behavior. Given a model parameterized by θ, let z denote an individual training sample and the empirical risk minimization objective be defined as $\hat{\theta}=\arg\min_{\theta\in\Theta}\frac{1}{n}\sum_{i=1}^{n}\ell(z_i;\theta)$. Following the classical influence-function formulation of Koh and Liang (2017), the effect of removing a training sample z_j on a validation objective can be approximated by infinitesimally perturbing its contribution to the training objective. This yields the following influence score:

$$\mathcal{I}(z_j;\hat{\theta}) = \nabla f(\mathcal{V};\hat{\theta})^\top \mathbf{H}_{\hat{\theta}}^{-1} \nabla \ell(z_j;\hat{\theta}), \tag{1}$$

where $\mathcal{V}$ denotes the validation set, $f(\cdot)$ represents the impact function that specifies the evaluation objective on $\mathcal{V}$, $\nabla\ell(z_j;\hat{\theta})$ denotes the gradient contribution of sample z_j, and $\mathbf{H}_{\hat{\theta}}=\sum_{i=1}^{n}\nabla^2\ell(z_i;\hat{\theta})$ is the Hessian matrix. This formulation provides a principled measure of how individual training samples affect model behavior through their contribution to the optimization process.

However, their application to reinforcement learning remains largely underexplored. Extending these principles to reinforcement learning is non-trivial due to several fundamental differences:

- *Input Trajectories*: Supervised learning assumes fixed training and validation datasets, enabling data valuation through each sample's influence on validation performance. In contrast, reinforcement learning generates trajectories online through environment interactions, with no fixed training set for valuation, rendering conventional valuation methods inapplicable.
- *Impact Function*: Classical influence functions rely on an explicit evaluation function defined on a validation set to quantify the contribution of individual samples. In reinforcement learning, trajectory usefulness is inherently policy-dependent and shaped by temporal credit assignment and reward dynamics, making such definitions elusive.
- *Dynamic Mechanism*: Unlike static, post-hoc evaluation in traditional data valuation, reinforcement learning requires efficient, dynamic valuation as trajectories evolve online without incurring prohibitive computational overhead.

Recent efforts (Hu et al., 2025; Shu et al., 2026; Li et al., 2026; Yang et al., 2026) extend data valuation and selection to reinforcement learning. Existing methods rely on setting-specific designs, including replay buffers and surrogate impact functions for PPO, verifiable-feedback-based learnability, and trusted validation gradients. However, they lack a unified and principled formulation across reinforcement learning paradigms, raising concerns about their generalizability.

Proposed Dynamic Trajectory Valuation. In this paper, we consider a principled formulation for general reinforcement learning by addressing the above three challenges in a unified framework. If we take a close look at Eq. (1), the first and second terms are constants across candidate samples, even given the unknown validation set and impact function. Therefore, the sample gradient is decisive for trajectory valuation, which casts the trajectory valuation problem into a sample gradient analysis problem. Suppose that learning is effective, beneficial samples should constitute the majority within a mini-batch, which leads the following assumption.

Assumption 1 (Majority Gradient Alignment). *Within a mini-batch, the majority of sample gradients collectively provide a locally informative reference direction for the current update. Accordingly, a sample whose gradient is negatively aligned with this reference direction is regarded as detrimental to the current optimization step.*

Under Assumption 1, we propose Dynamic Trajectory Valuation (DTV) to dynamically identify detrimental training units during optimization. A *training unit* denotes the basic entity for which a per-unit gradient contribution is computed, with its form depending on the underlying learning paradigm: a completion trajectory generated for a prompt in GRPO, and a prompt with a chosen–rejected response pair in DPO, and a trajectory segment in PPO. Given a mini-batch $\mathcal{B}=\{z_j\}_{j=1}^{b}$, let $g_j^{(t)}=\nabla_\theta\ell(z_j;\theta^{(t)})$ denote the gradient contribution of training unit z_j, where $\ell(z_j;\theta^{(t)})$ is the training loss under the underlying algorithm. We define the DTV score as

$$\mathcal{I}^{\mathrm{DTV}}(z_j;\theta^{(t)}) = \frac{1}{b}\sum\nolimits_{i=1}^{b}\big(g_i^{(t)}\big)^\top g_j^{(t)}. \tag{2}$$

DTV requires no separate impact function or predefined validation set. Instead, its reference direction is derived directly from the current mini-batch gradients. Training units with negative DTV scores are identified as alignment-opposing and filtered out dynamically before executing the optimizer update.

Measuring alignment in gradient space directly reflects how each training unit impacts parameter trajectory. Prior work (Chhabra et al., 2024) also follows Assumption 1 and considers gradient-based outlier analysis for static data valuation in classification (e.g., Isolation Forests), while DTV adapts a simpler inner product for outlier filtering and addresses the dynamic nature of reinforcement learning. By leveraging the majority gradient as an online reference, DTV adapts to evolving policy checkpoints without global convergence or local convexity assumptions (Wang et al., 2024; Yang et al., 2025).

A key advantage of our DTV is that it enables direct and efficient valuation without additional gradient approximations. Prior implementations rely on batch-level gradients and require surrogate techniques such as ghost influence or layer-wise estimation (Wang et al., 2024; Yang et al., 2025). In contrast, our JAX (Bradbury et al., 2018) implementation leverages vectorized automatic differentiation to directly compute training-unit-level gradients, enabling efficient evaluation of Eq. (2) and seamless integration into diverse reinforcement learning pipelines.

DTV Decomposition and Leave-One-Out Extension. To clarify the information captured by DTV, we decompose its score into the contribution from the training unit itself and its interactions with the remaining mini-batch. Expanding the mini-batch mean gives $\mathcal{I}^{\mathrm{DTV}}(z_j;\theta^{(t)})=\frac{1}{b}\|g_j^{(t)}\|^2+\frac{1}{b}\sum_{i\neq j}(g_i^{(t)})^\top g_j^{(t)}$. We define $s_j^{(t)}=\frac{1}{b}\|g_j^{(t)}\|^2$ and $c_j^{(t)}=\frac{1}{b}\sum_{i\neq j}(g_i^{(t)})^\top g_j^{(t)}$ denote the *self-term* and *cross-term*, respectively. The self-term captures self-alignment, whereas the cross-term measures the gradient agreement with the remaining training units.

Remark 1 (Self-Protection Effect). Since $s_j^{(t)}\geq 0$, DTV filters a training unit only when $c_j^{(t)}<-s_j^{(t)}$. A sufficiently large self-term can therefore offset a negative cross-term and retain a training unit even when it is strongly negatively aligned with the remaining mini-batch, an effect we refer to as self-protection. Nevertheless, the self-term should not be interpreted as universally detrimental, as its effect may depend on the underlying optimization setting.

The self-protection effect motivates a leave-one-out extension that isolates the cross-term. Specifically, we average over the remaining $b-1$ units and define DTV-Loo as

$$\mathcal{I}^{\text{DTV-Loo}}(z_j;\theta^{(t)}) = \frac{1}{b-1}\sum\nolimits_{i\neq j}\big(g_i^{(t)}\big)^\top g_j^{(t)} = \frac{b}{b-1}\big(\mathcal{I}^{\mathrm{DTV}} - s_j^{(t)}\big). \tag{3}$$

More generally, DTV and DTV-Loo can be unified into a broader DTV-λ family ($\lambda \in [0,1]$) by weighting the self-term. Here, $\lambda{=}1$ recovers DTV, whereas $\lambda{=}0$ completely removes the self-term to yield DTV-Loo. Despite this broader parameterization, we retain DTV and DTV-Loo as the practical instantiations, as both already achieve strong empirical performance without requiring additional tuning of λ. Empirically, DTV-Loo performs better than DTV in most evaluated settings, whereas DTV can be more effective under specific regimes, demonstrating their complementary strengths across different optimization scenarios. Beyond empirical comparisons, we also provide practical guidance for choosing between the two variants. Algorithm 1 in Appendix C summarizes the filtering procedure.

4 Experiments

To evaluate the versatility and practical applicability of DTV and DTV-Loo, we apply our methods[2] across diverse reinforcement learning and LLM post-training settings, in terms of GRPO, DPO, and PPO[3] scenarios. Beyond algorithmic performance, we further investigate the self-protection effect underlying the distinction between DTV and DTV-Loo, optimization dynamics, and data efficiency.

4.1 DTV for Group-Relative Completion Valuation with GRPO

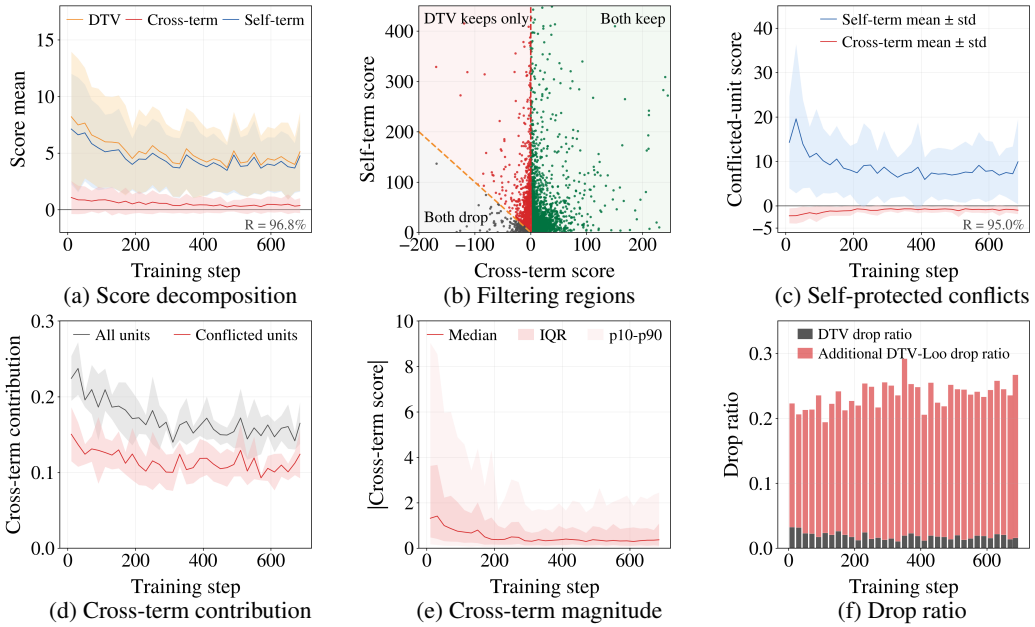
We examine DTV and DTV-Loo in the GRPO setting on two mathematical reasoning settings of different difficulty, ranging from grade-school reasoning on *GSM8K* (Cobbe et al., 2021) to more challenging competition-level reasoning on the American Invitational Mathematics Examination (AIME) (Hugging Face H4). GRPO generates multiple completion trajectories for each prompt and

[2]All experiments follow a JAX-based (Bradbury et al., 2018) implementation, with the GRPO and DPO pipelines further built on Google's open-source Tunix library (Bao et al., 2025), which we extend with vectorized training-unit-level gradient computation and dynamic filtering.

[3]Due to the page limitation, we present the complete experiments on PPO in Appendix D.

Table 1: **GRPO results on *GSM8K* under clean and mismatch settings.** Δ Acc. is measured relative to the shared pre-trained model. Results are reported as mean ± std over five seeds, with the best result shown in bold. Given the consistent margins, we omit additional pairwise t-tests.

	Clean			*Mismatch-20%*		
Method	Acc. ↑	Δ Acc. ↑	Partial ↑	Acc. ↑	Δ Acc. ↑	Partial ↑
Pre-trained	0.4723±0.0000	0.0000	0.4996±0.0000	0.4723±0.0000	0.0000	0.4996±0.0000
Vanilla GRPO	0.4828±0.0077	0.0105	0.5114±0.0089	0.4731±0.0094	0.0008	0.5001±0.0089
Random 5%	0.5207±0.0341	0.0484	0.5442±0.0307	0.4103±0.2026	-0.0620	0.4255±0.2086
Random 10%	0.4511±0.1175	-0.0212	0.4760±0.1127	0.5133±0.0431	0.0409	0.5318±0.0455
Reward 5%	0.4516±0.2104	-0.0208	0.4729±0.2181	0.4443±0.1691	-0.0281	0.4628±0.1733
Reward 10%	0.4973±0.1128	0.0250	0.5193±0.1094	0.3777±0.1981	-0.0946	0.4012±0.2000
LearnAlign (2026)	0.5014±0.0170	0.0291	0.5281±0.0160	0.4863±0.0069	0.0140	0.5130±0.0083
GradAlign (2026)	0.4876±0.0040	0.0153	0.5154±0.0046	0.4775±0.0064	0.0052	0.5060±0.0051
DTV (Ours)	**0.5483±0.0075**	**0.0760**	**0.5742±0.0099**	**0.5413±0.0139**	**0.0690**	**0.5660±0.0123**
DTV-Loo (Ours)	0.5416±0.0141	0.0693	0.5647±0.0146	0.5234±0.0103	0.0511	0.5460±0.0076



(a) Score decomposition (b) Filtering regions (c) Self-protected conflicts

(d) Cross-term contribution (e) Cross-term magnitude (f) Drop ratio

Figure 1: **Empirical analysis of the self-protection effect on *GSM8K* under *Mismatch-20%*.** We define *conflicted units* as training units for which DTV and DTV-Loo make different keep/drop decisions; R denotes the fraction retained for visualization after trimming extreme score ranges, and IQR denotes the interquartile range (25th–75th percentiles). (a) Decomposition of the DTV score into the self- and cross-terms. (b) Filtering regions of DTV and DTV-Loo. (c) Self- and cross-term scores of conflicted units. (d) Cross-term contribution $|c_j|/(s_j + |c_j|)$ for all and conflicted units. (e) Cross-term magnitude of conflicted units. (f) Drop ratios of DTV and the additional filtering induced by DTV-Loo.

optimizes them using group-relative advantages, providing a natural setting for completion-level valuation and filtering. Across both benchmarks, DTV and DTV-Loo use policy-loss gradients to value individual completion trajectories relative to other completions generated for the same prompt. We further analyze the role of the self-term on *GSM8K* to better understand the performance differences between DTV and DTV-Loo under group-relative optimization.

Experimental Setup. We evaluate DTV and DTV-Loo against Vanilla GRPO, two fixed-ratio filtering baselines, and two gradient-alignment-based ranking baselines. Vanilla GRPO performs the standard group-relative update without filtering. Both fixed-ratio baselines operate on individual completions within each prompt group. Random Filtering selects completions independently of their reward signals, whereas Reward-based Filtering removes those with the lowest observed group-relative advantages. The two gradient-alignment-based baselines operate at the prompt level:

Table 2: **GRPO results on *AIME 2024* with a 32K maximum generation length.** Δ denotes the change relative to the shared pre-trained model. Results are from a single training seed per method, with the best result shown in bold. A t-test is not applicable with a single run.

Method	Pass@1		Pass@16		Maj@16		Extractable ↑	Acc. given extractable ↑
	Score ↑	Δ ↑	Score ↑	Δ ↑	Score ↑	Δ ↑		
Pre-trained	0.1063	0.0000	0.3333	0.0000	0.2333	0.0000	0.7292	0.1457
Vanilla GRPO	0.1063	0.0000	0.4000	0.0667	0.2333	0.0000	0.7375	0.1441
DTV (Ours)	0.0917	-0.0146	0.3333	0.0000	0.2667	0.0334	0.7188	0.1275
DTV-Loo (Ours)	**0.1333**	**0.0270**	**0.4333**	**0.1000**	**0.3667**	**0.1334**	**0.7521**	**0.1773**

LearnAlign (Li et al., 2026) uses learnability-weighted gradient alignment for top-N selection, whereas GradAlign (Yang et al., 2026) uses alignment with a validation gradient to retain a fixed fraction of prompts. DTV and DTV-Loo instead determine which completions to retain adaptively based on within-group policy-gradient alignment, without prescribing a filtering rate. We evaluate our methods on widely-used benchmarks, *GSM8K* (Cobbe et al., 2021) and the more challenging *AIME 2024* (Hugging Face H4). More implementation details are provided in Appendix E.

Results on *GSM8K*. Table 1 reports the *GSM8K* results under *Clean* and *Mismatch-20%*. DTV achieves the best exact and partial accuracy in both settings, improving exact accuracy over the shared pre-trained model by 0.0760 under *Clean* and 0.0690 under *Mismatch-20%*. DTV-Loo also improves over Vanilla GRPO in both settings, although DTV attains slightly higher average accuracy in both cases. In contrast, Random Filtering exhibits highly variable behavior across filtering ratios and data conditions, with the relative performance of 5% and 10% reversing between *Clean* and *Mismatch-20%*, consistent with its sensitivity to stochastic sample removal. Reward-based Filtering is likewise unstable and does not provide reliable gains, with several settings falling below the shared pre-trained model. The two gradient-alignment baselines, LearnAlign and GradAlign, provide modest but more stable gains than Random and Reward-based Filtering across both settings, while remaining below DTV and DTV-Loo on all metrics. Under *Mismatch-20%*, their gains further diminish, with exact-accuracy improvements of only 0.0140 and 0.0052, respectively, compared with 0.0690 for DTV. Overall, these results show that gradient-based adaptive filtering using only current mini-batch gradients yields stronger and more stable GRPO performance on *GSM8K*. By using the batch gradient as an endogenous reference, DTV removes trajectories that conflict with the current update direction, reducing detrimental gradient interference before the model update and improving optimization performance.

Unlike the baseline methods, which rely on prescribed selection or filtering budgets, our proposed DTV and DTV-Loo make filtering decisions directly using the zero valuation threshold, allowing the effective filtering ratio to adapt dynamically over training. Notably, DTV slightly outperforms DTV-Loo across all reported metrics, suggesting that retaining the self-term can remain beneficial in this setting. To understand this difference, we revisit the self- and cross-term decomposition and examine how the self-term affects the filtering decisions of DTV and DTV-Loo on *GSM8K*. We return to this distinction in the latter DPO analysis to further clarify when each variant is preferable.

Figure 1 provides a closer examination of the difference between DTV and DTV-Loo, i.e., the self- and cross-term effect on *GSM8K*. Figure 1(a) demonstrates that the self-term constitutes the primary component of the DTV score, whereas the cross-term remains comparatively small throughout training. This suggests that DTV may be susceptible to extreme samples that dominate the batch gradient, as the batch mean used in DTV lacks robustness to outliers. Although we do not observe this issue on *GSM8K*, we revisit it in the next subsection to highlight the necessity of DTV-Loo, which mitigates the self-term effect while remaining effective at filtering detrimental samples. In addition, as implied by the decomposition in Section 3, removing the non-negative self-term causes DTV-Loo to additionally filter units with negative cross-terms that DTV still retains. Figure 1(b)–(c) visualize these conflicted units and their corresponding self- and cross-term scores. The remaining question is whether these additionally filtered units are sufficiently detrimental to be removed. Furthermore, Figures 1(d)–(e) show that conflicted units have relatively small cross-term contributions and that many of their negative cross-terms remain close to zero, indicating weak cross-unit conflicts and similar performance between DTV and DTV-Loo. Consequently, although DTV-Loo filters substantially more units through training, as shown in Figure 1(f), its performance remains close to DTV, suggesting strong data efficiency. Meanwhile, retaining these weakly conflicted units helps explain the slightly better performance of DTV in this setting.

Table 3: **DPO preference-accuracy results on *UltraFeedback* under clean and mismatch settings.** Results are reported as mean ± std over five seeds, with the best result in each column shown in bold. †The final row reports two-sided paired t-test p-values comparing DTV-Loo with Reward 10% across matched seeds; significant differences ($p < 0.05$) are shown in bold.

	Clean		*Mismatch-20%*		*Mismatch-40%*	
Method	AUC ↑	Acc. ↑	AUC ↑	Acc. ↑	AUC ↑	Acc. ↑
Vanilla DPO	0.5611±0.0035	0.5388±0.0073	0.5377±0.0029	0.5014±0.0044	0.4932±0.0105	0.4301±0.0160
Random 5%	0.5585±0.0022	0.5449±0.0023	0.5371±0.0032	0.5013±0.0152	0.4907±0.0105	0.4234±0.0143
Random 10%	0.5578±0.0017	0.5421±0.0063	0.5363±0.0040	0.5015±0.0100	0.4900±0.0113	0.4332±0.0122
Reward 5%	0.5620±0.0016	0.5559±0.0062	0.5465±0.0028	0.5350±0.0092	0.5083±0.0089	0.4683±0.0191
Reward 10%	0.5652±0.0013	0.5647±0.0063	0.5490±0.0006	0.5443±0.0048	0.5132±0.0090	0.4857±0.0147
DTV (Ours)	0.5653±0.0021	0.5618±0.0020	0.5445±0.0041	0.5236±0.0068	0.4985±0.0110	0.4413±0.0103
DTV-Loo (Ours)	**0.5723±0.0004**	**0.5680±0.0040**	**0.5669±0.0009**	**0.5537±0.0068**	**0.5406±0.0093**	**0.5196±0.0208**
p-value†	**0.0006**	0.3680	**< 0.0001**	0.1421	**< 0.0001**	**0.0176**

Scalability and Generalization Test. To further evaluate the scalability and generalization of DTV and DTV-Loo, we consider a more complicated and large-scale *AIME* challenge. Specifically, unlike the *GSM8K* experiments with Gemma-3-1B-IT and LoRA, here we full-parameter fine-tune DeepSeek-R1-Distill-Qwen-1.5B using a JAX-based implementation on a dual-worker TPU v5p-16 setup, providing an additional evaluation across model backbones, fine-tuning regimes, and implementation stacks. All other experiments use a single TPU v5p-8 node, whereas *AIME* requires the dual-worker setup due to HBM constraints, introducing inter-worker communication and nondeterministic task assignment at startup. A single training seed requires over 140 hours for each method; we therefore conduct one training run for each method. Given this substantial cost and the ad hoc filtering-ratio choices required by Random and Reward, we omit these two baselines and focus on Vanilla GRPO, DTV, and DTV-Loo and report the results in Table 2.

Despite this demanding setting, DTV-Loo achieves the best performance across all reported metrics. Relative to the shared pre-trained model, it improves Pass@1 from 0.1063 to 0.1333 ($+0.0270$), Pass@16 from 0.3333 to 0.4333 ($+0.1000$), and Maj@16 from 0.2333 to 0.3667 ($+0.1334$). It also attains the highest extractable-answer rate (0.7521) and accuracy given an extractable answer (0.1773), indicating that the gains extend beyond aggregate reasoning accuracy to more reliable final-answer generation. In comparison, Vanilla GRPO and DTV show mixed improvements across the reasoning metrics, with DTV improving Maj@16 but not Pass@1 or Pass@16.

4.2 DTV for Pairwise Preference Valuation with DPO

Next, we turn to DPO to examine whether DTV generalizes beyond online rollout-based optimization to pairwise preference learning. In addition to the clean setting, we introduce controlled preference corruption to evaluate robustness to increasingly unreliable supervision. We further examine time-to-target efficiency and analyze the self- and cross-term decomposition underlying the self-protection effect, providing complementary views of optimization efficiency and valuation behavior.

Experimental Setup. Experiments are conducted on *UltraFeedback* (Cui et al., 2023) with Qwen2.5-1.5B (Team, 2024; Yang et al., 2024a) using a two-stage pipeline consisting of supervised fine-tuning (SFT) followed by DPO. We consider *Clean*, *Mismatch-20%*, and *Mismatch-40%*, where the latter two corrupt 20% and 40% of the DPO training pairs through cross-response mismatching and chosen–rejected label flipping; all held-out evaluation and test data remain clean. Full hyperparameter, implementation details and evaluation metrics are provided in Appendix F.

Main Results. Table 3 summarizes in-domain preference performance on the held-out test set, covering both training-time behavior and final-checkpoint accuracy. DTV-Loo achieves the best AUC across all three conditions. Compared with Reward 10%, its AUC margin widens from 0.5723 vs. 0.5652 ($+0.0071$) under *Clean* to 0.5406 vs. 0.5132 ($+0.0274$) under *Mismatch-40%*, indicating an increasingly pronounced training-time advantage as preference corruption intensifies. The advantage also extends to final-checkpoint performance, where DTV-Loo achieves the highest Acc. in all three settings, with the corresponding margin over Reward 10% increasing from 0.5680 vs. 0.5647 ($+0.0033$) to 0.5196 vs. 0.4857 ($+0.0339$). The stronger robustness under increasing corruption is consistent with the leave-one-out motivation. With a larger fraction of misleading training units, cross-unit conflicts become more consequential, while positive self-terms can mask negative alignment

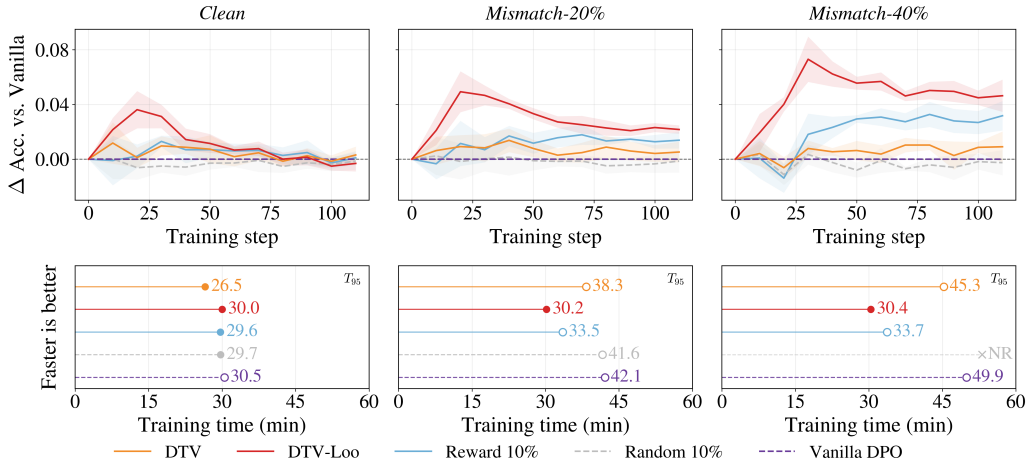


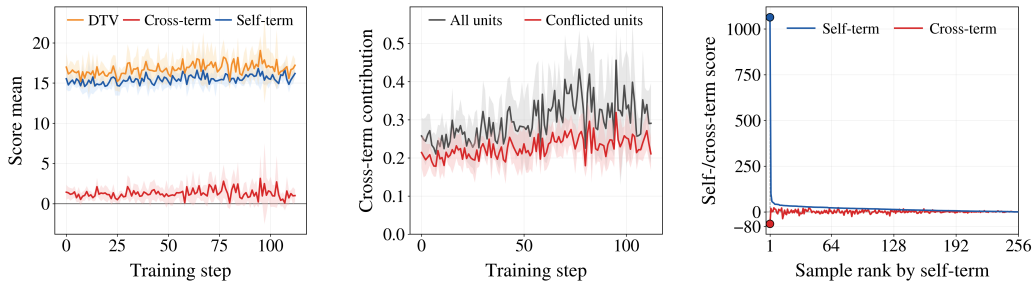
Figure 2: **DPO test-accuracy dynamics and training efficiency on *UltraFeedback*.** The top row shows the test preference-accuracy relative to Vanilla DPO throughout training under *Clean*, *Mismatch-20%*, and *Mismatch-40%* conditions, averaged over five seeds; shaded regions indicate variability across seeds, and the horizontal dashed line denotes Vanilla DPO. The bottom row reports, for seed 0, the wall-clock time to reach the T_{95} target defined from the five-seed best test performance; $\times$NR denotes that the target is not reached within the training budget. DTV-Loo exhibits increasingly large and more persistent improvements as the mismatch rate increases, while also reaching the convergence target substantially faster under the mismatch settings.

and cause DTV to retain these units. By removing the self-term and relying solely on cross-unit alignment, DTV-Loo suppresses this self-protection and better preserves the conflict signal, which is reflected in its stronger performance under heavier corruption. This robustness is further supported by paired t-tests against Reward 10%, which show significant AUC improvements under all three conditions ($p = 0.0006$, $p < 0.0001$, and $p < 0.0001$), while the Acc. improvement is significant under *Mismatch-40%* ($p = 0.0176$). Together, these results underscore the robustness of DTV-Loo to corrupted preference supervision. DTV-Loo consistently achieves the best performance across training, outperforming both Random and Reward-based Filtering even when their fixed filtering rates are increased to 20% and 40%, which is shown in Appendix F.

The top row of Figure 2 further characterizes these gains through test-accuracy dynamics, and shows the change in test preference accuracy relative to Vanilla DPO throughout training. Under *Clean*, DTV-Loo establishes a clear early-stage advantage, although the gap gradually narrows as training proceeds. Under the mismatch settings, the advantage becomes both larger and more persistent, with the strongest separation observed under *Mismatch-40%* and substantial gains maintained through the later stages of training. Reward 10% also improves over Vanilla DPO under mismatch, but its gains remain consistently smaller, while DTV and Random 10% exhibit weaker or less stable improvements.

Computational Efficiency. Here we further demonstrate the training efficiency of our proposed DTV and DTV-Loo. As demonstrated in the bottom row of Figure 2, by dynamically filtering out harmful or conflicting trajectories to prevent counterproductive parameter updates, DTV-Loo enhances data efficiency, achieving target performance with substantially fewer optimization steps. Consequently, under corrupted preference settings, DTV-Loo reaches the target accuracy (T_{95}) in approximately 30 minutes—drastically reducing total wall-clock training time compared to Vanilla DPO, while Random 10% fails to reach the target under *Mismatch-40%* within the training budget. Similar phenomena are also observed in the PPO experiments shown in Figure 5. In short, our method achieves a double win by simultaneously boosting performance and accelerating overall training.

When to Use DTV vs. DTV-Loo. The core distinction between DTV and DTV-Loo stems from the cross-term effect. While the previous subsection illustrated a regime where a minimal cross-term effect results in marginal performance differences between DTV and DTV-Loo (e.g., on *GSM8K*), the cross-term plays a far more decisive role under other training conditions. Below, we examine scenarios where cross-term interactions substantially impact optimization (See the difference under the noisy *Mismatch-20%* setting in Table 3) and provide practical guidance on the variant selection.



(a) Score decomposition (b) Cross-term contribution (c) Sample rank

Figure 3: **Empirical analysis of the self-protection effect on *UltraFeedback* under *Mismatch-20%*.** (a) Decomposition of the DTV score into the self- and cross-terms. (b) Cross-term contribution $|c_j|/(s_j + |c_j|)$ for all and conflicted units. (c) Visualization of self- and cross-term scores across samples in a representative batch (ranked by self-term). A single dominant sample with an extreme self-term score overwhelms the cross-term interactions across the entire batch, illustrating the self-protection mechanism in standard DTV.

From a macro perspective, the self-term acts as a regularizing buffer that governs the strictness of trajectory filtering: retaining it favors conservatism, whereas removing it via DTV-Loo enforces strict cross-unit consistency. Figure 3 empirically substantiates this trade-off under *Mismatch-20%*. Specifically, Figure 3(a) demonstrates that the DTV score is predominantly driven by the self-term throughout training. This dominance becomes critical when a training unit is negatively aligned with the batch: a strong positive self-term can easily override a negative cross-term, thereby causing DTV to retain units that DTV-Loo would otherwise filter. In contrast to the decreasing cross-term contribution observed on *GSM8K*, Figure 3(b) shows that the cross-term contribution remains substantial and generally increases during DPO training, making cross-unit alignment more consequential in this setting. Figure 3(c) provides a representative case within a single mini-batch where a single sample exhibits an exceptionally large self-term, whereas the cross-terms across all samples remain negligible in magnitude. Consequently, this single outlier sample dominates the aggregate DTV valuation of the entire batch, artificially driving the score positive and masking critical cross-unit conflicts that DTV-Loo successfully identifies and removes. Moreover, the self-term violates the majority gradient assumption and disproportionately affects DTV's valuation despite its negative cross-term. By removing this unit's self-term, DTV-Loo is less susceptible to such a dominant unit and better preserves the cross-unit conflict signal.

These insights provide clear practical guidance on selecting between the two variants, supporting DTV-Loo as the default choice in practice. By eliminating self-protection, DTV-Loo relies strictly on cross-unit alignment, offering superior robustness, faster convergence, and significant performance gains in complex or high-noise regimes such as DPO on *UltraFeedback* and GRPO on *AIME*. Meanwhile, in low-conflict scenarios like GRPO on *GSM8K*—where disagreements stem primarily from weak negative cross-terms—DTV-Loo performs comparably to standard DTV, demonstrating that strict filtering incurs minimal risk of discarding useful signals. Retaining the self-term in DTV is thus primarily beneficial when conservative filtering is specifically needed to safeguard weakly conflicting samples; otherwise, DTV-Loo serves as the more robust and effective default strategy.

5 Conclusion

In this paper, we studied trajectory valuation in reinforcement learning and proposed Dynamic Trajectory Valuation (DTV), a simple and general framework that dynamically identifies detrimental training units through mini-batch gradient alignment without requiring a predefined validation set or influence-function approximation. Building on the decomposition of DTV, we further introduced DTV-Loo to remove the self-term and analyzed the resulting self-protection effect, revealing that the utility of the self-term can depend on the underlying optimization setting. We evaluated DTV and DTV-Loo across diverse reinforcement learning and LLM post-training paradigms, including GRPO, DPO, and PPO, spanning trajectory segments, completion trajectories, and preference pairs. Extensive experiments demonstrate that dynamic trajectory valuation can improve optimization efficiency, robustness, and final performance across these settings, highlighting the practical value of training-unit-level valuation for reinforcement learning.

AI USE STATEMENT

In this work, we used generative AI tools to assist with the implementation of our methods and coding of the data mismatch procedure that we designed. We also used AI tools for minor feedback on how to present related formulations.

We did not use generative AI to design the core methodology or experiments, propose hypotheses, develop theoretical models or conceptual frameworks, formulate mathematical claims or proofs, generate dataset content, interpret experimental results, or conduct literature searches or citation selection.

Additionally, we used generative AI tools for code debugging and optimization, experiment scripts, plotting and table formatting, and used large language models to polish the manuscript language and improve readability. We reviewed and tested all AI-assisted code in the actual experiments, and independently verified all reported results and final manuscript content. We take responsibility for the final manuscript and all reported results.

REPRODUCIBILITY STATEMENT

We release our code and implementation at `https://anonymous.4open.science/r/dtv-iclr2027/`. The repository contains reproducibility artifacts for all GRPO, DPO, and PPO experiments, covering *GSM8K*, *AIME*, *UltraFeedback*, and *MiniGrid*. All experiments are implemented in JAX; the GRPO and DPO implementations are built on or derived from Google Tunix, while the PPO implementation is standalone. Details on environments and datasets, hyperparameters, and training and evaluation procedures are provided in Appendices E, F, and D, while launch commands are provided in the corresponding repository READMEs. The GRPO experiments on *GSM8K*, as well as DPO and PPO experiments, use TPU v5p-8 hardware, while the GRPO experiments on *AIME 2024* use a dual-worker TPU v5p-16 setup.

REFERENCES

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

Dong Shu, Denghui Zhang, and Jessica Hullman. Learning from the right rollouts: Data attribution for ppo-based llm post-training. *arXiv preprint arXiv:2604.01597*, 2026.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, 2017.

Yongchan Kwon, Eric Wu, Kevin Wu, and James Zou. Datainf: Efficiently estimating data influence in lora-tuned llms and diffusion models. *arXiv preprint arXiv:2310.00902*, 2023.

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. Trak: Attributing model behavior at scale. *arXiv preprint arXiv:2303.14186*, 2023.

Yuzheng Hu, Fan Wu, Haotian Ye, David Forsyth, James Zou, Nan Jiang, Jiaqi W Ma, and Han Zhao. A snapshot of influence: A local data attribution framework for online reinforcement learning. *arXiv preprint arXiv:2505.19281*, 2025.

Shipeng Li, Zhiqin Yang, Shikun Li, Xiaobo Xia, Hengyu Liu, Xinghua Zhang, Gaode Chen, Dong Fang, Ying Tai, and Zhe Peng. Learnalign: Data selection for llm reinforcement learning with improved gradient alignment. *arXiv preprint arXiv:2506.11480*, 2026.

Ningyuan Yang, Weihua Du, Weiwei Sun, Sean Welleck, and Yiming Yang. Gradalign: Gradient-aligned data selection for llm reinforcement learning. *arXiv preprint arXiv:2602.21492*, 2026.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *International Conference on Machine Learning*, 2009.

Ang A Li, Zongqing Lu, and Chenglin Miao. Revisiting prioritized experience replay: A value perspective. *arXiv preprint arXiv:2102.03261*, 2021.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *International Conference on Machine Learning*. PMLR, 2021.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in Neural Information Processing Systems*, 2017.

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*, 2022.

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.

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*, 2023.

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

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.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 2020.

Anshuman Chhabra, Bo Li, Jian Chen, Prasant Mohapatra, and Hongfu Liu. Outlier gradient analysis: Efficiently identifying detrimental training samples for deep learning models. *arXiv preprint arXiv:2405.03869*, 2024.

Jiachen T Wang, Prateek Mittal, Dawn Song, and Ruoxi Jia. Data shapley in one training run. *arXiv preprint arXiv:2406.11011*, 2024.

Ziao Yang, Longbo Huang, and Hongfu Liu. Layer-aware influence for online data valuation estimation. *arXiv preprint arXiv:2510.16007*, 2025.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Yash Katariya, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

Tianshu Bao, Jeff Carpenter, Lin Chai, Haoyu Gao, Yangmu Jiang, Shadi Noghabi, Abheesht Sharma, Sizhi Tan, Lance Wang, Ann Yan, Weiren Yu, et al. Tunix (tune-in-jax), 2025.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Hugging Face H4. AIME 2024.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. Ultrafeedback: Boosting language models with high-quality feedback, 2023.

Qwen Team. Qwen2.5: A party of foundation models, 2024.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024a.

Frank R Hampel. The influence curve and its role in robust estimation. *Journal of the American Statistical Association*, 1974.

R Dennis Cook and Sanford Weisberg. *Residuals and influence in regression*. New York: Chapman and Hall, 1982.

R Douglas Martin and Victor J Yohai. Influence functionals for time series. *The Annals of Statistics*, 1986.

Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, et al. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*, 2023.

Guillaume Charpiat, Nicolas Girard, Loris Felardos, and Yuliya Tarabalka. Input similarity from the neural network perspective. *Advances in Neural Information Processing Systems*, 2019.

Ziao Yang, Han Yue, Jian Chen, and Hongfu Liu. Revisit, extend, and enhance hessian-free influence functions. *arXiv preprint arXiv:2405.17490*, 2024b.

Krishnateja Killamsetty, Sivasubramanian Durga, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. Grad-match: Gradient matching based data subset selection for efficient deep model training. In *International Conference on Machine Learning*, 2021.

Satoshi Hara, Atsushi Nitanda, and Takanori Maehara. Data cleansing for models trained with sgd. *Advances in Neural Information Processing Systems*, 2019.

Haoru Tan, Sitong Wu, Fei Du, Yukang Chen, Zhibin Wang, Fan Wang, and Xiaojuan Qi. Data pruning via moving-one-sample-out. *Advances in Neural Information Processing Systems*, 2023.

Narine Kokhlikyan, Kamalika Chaudhuri, and Saeed Mahloujifar. Z0-inf: Zeroth order approximation for data influence. *arXiv preprint arXiv:2510.11832*, 2025.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo Perez-Vicente, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. 2023.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl, 2025.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, et al. Gemma 3 technical report, 2025.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Benjamin Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Sreemanti Dey, et al. Livebench: A challenging, contamination-limited llm benchmark. In *International Conference on Learning Representations*, 2025.

Saumya Malik, Valentina Pyatkin, Sander Land, Jacob Morrison, Noah Smith, Hanna Hajishirzi, and Nathan Lambert. Rewardbench 2: Advancing reward model evaluation. In *International Conference on Learning Representations*, 2026.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hanna Hajishirzi. Generalizing verifiable instruction following. *Advances in Neural Information Processing Systems*, 2025.

APPENDIX

This appendix provides preliminaries, additional related work, algorithmic pseudo-code, complete experimental details and results for PPO, additional implementation details for GRPO and DPO, and extended experimental results for DPO to complement the main manuscript.

A PRELIMINARIES ON REINFORCEMENT LEARNING

A standard reinforcement learning framework is modeled as a Markov Decision Process, defined by a tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, $P(s' \mid s, a)$ is the transition dynamics, $r(s, a)$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. A policy $\pi_\theta(a \mid s)$, parameterized by θ, maps states to action distributions. The goal of reinforcement learning is to maximize the expected discounted return:

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t) \right], \tag{4}$$

where $\tau{=}(s_0, a_0, \ldots, s_T)$ denotes a trajectory.

A common approach to optimizing this objective is via policy gradient methods, which estimate the gradient of $J(\theta)$ as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot A^\pi(s_t, a_t) \right], \tag{5}$$

where $A^\pi(s, a)$ is the advantage function, measuring the relative contribution of an action compared to a baseline policy.

In practice, optimizing this objective in high-dimensional settings requires stable and scalable algorithms. Importantly, many modern approaches operate at the level of complete trajectories, where each rollout—i.e., a trajectory generated by the policy during interaction with the environment—serves as a fundamental training unit. We next introduce several representative methods in this regime, which form the basis of our study.

Proximal Policy Optimization (PPO). Proximal Policy Optimization (PPO) (Schulman et al., 2017) is a widely used policy gradient method that stabilizes training via a clipped surrogate objective. Let $r_t(\theta){=}\pi_\theta(a_t \mid s_t)/\pi_{\theta_{\text{old}}}(a_t \mid s_t)$ denote the importance sampling ratio. The PPO objective is:

$$\mathcal{L}_{\text{PPO}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta) A_t,\ \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) A_t \right) \right]. \tag{6}$$

This clipping mechanism prevents excessively large policy updates and improves training stability.

Group Relative Policy Optimization (GRPO). Group Relative Policy Optimization (GRPO) (Shao et al., 2024) extends PPO by replacing absolute advantage estimates with relative advantages computed within a group of sampled trajectories. Given a group $\mathcal{G}{=}\{\tau_i\}_{i=1}^{K}$, GRPO normalizes rewards within the group by $\tilde{A}_i{=}(r_i - \mu_\mathcal{G})/\sigma_\mathcal{G}$, where $\mu_\mathcal{G}$ and $\sigma_\mathcal{G}$ are the mean and standard deviation of rewards in the group. The objective of GRPO becomes:

$$\mathcal{L}_{\text{GRPO}}(\theta) = \mathbb{E}_{i,t} \left[\min \left(r_{i,t}(\theta) \tilde{A}_i,\ \text{clip}(r_{i,t}(\theta), 1 - \epsilon, 1 + \epsilon) \tilde{A}_i \right) \right]. \tag{7}$$

By leveraging relative comparisons, GRPO reduces reliance on value function estimation and improves robustness in scenarios such as Reinforcement Learning from Human Feedback (RLHF).

Direct Preference Optimization (DPO). Direct Preference Optimization (DPO) (Rafailov et al., 2023) formulates policy learning directly from preference data without explicit reward modeling. Given a pair of responses (y^+, y^-) for a prompt x, where y^+ is preferred over y^-, DPO optimizes:

$$\mathcal{L}_{\text{DPO}}(\theta) = -\mathbb{E}_{(x, y^+, y^-)} \left[\log \sigma \left(\beta \left(\log \frac{\pi_\theta(y^+ \mid x)}{\pi_{\text{ref}}(y^+ \mid x)} - \log \frac{\pi_\theta(y^- \mid x)}{\pi_{\text{ref}}(y^- \mid x)} \right) \right) \right], \tag{8}$$

where π_{ref} is a reference policy and β controls the sharpness. DPO can be interpreted as implicitly optimizing a KL-regularized reward objective while bypassing explicit reward model training.

In sum, PPO provides an on-policy optimization framework, GRPO enhances it with relative trajectory comparisons, and DPO extends RL to preference-based optimization without explicit rewards. These methods form the backbone of modern RL, especially in large-scale language model post-training.

Table 4: Comparison of additional requirements and applicability of gradient-based data valuation methods in reinforcement learning. The binary columns indicate whether each method uses an auxiliary valuation signal, assumes a fixed dataset, requires a validation set, uses an explicit selection budget, requires extra rollouts for valuation, or performs valuation online. The final column summarizes each method's generalizability across reinforcement learning settings.

Method	Auxiliary signal	Fixed dataset	Validation set	Selection budget	Extra rollouts	Online valuation	Applied scenario
IIF (2025)	Yes	**No**	**No**	Yes	**No**	**Yes**	PPO
LearnAlign (2026)	Yes	Yes	**No**	Yes	Yes	No	GRPO / RLVR
GradAlign (2026)	Yes	**No**	Yes	Yes	Yes	**Yes**	GRPO
DTV/DTV-Loo (Ours)	**No**	**No**	**No**	**No**	**No**	**Yes**	**General**

B Additional Related Work

B.1 Data-Centric Learning in Reinforcement Learning

Table 4 compares DTV with three representative gradient-based approaches and summarizes their requirements in reinforcement learning. IIF (Hu et al., 2025) requires an algorithm-specific surrogate target function for valuation and a filtering parameter p that determines how many negatively influential samples are removed. Its target function must also be adapted to the underlying reinforcement learning setting. IIF performs valuation online by reusing the current rollout buffer without additional rollouts. LearnAlign (Li et al., 2026) relies on verifiable ground-truth feedback to estimate learnability and introduces N to determine the selected subset. Its formulation assumes a fixed problem dataset with ground-truth answers, as required by the Reinforcement Learning with Verifiable Rewards (RLVR) setting, rather than experience collected purely through online environment interaction. It also requires additional rollouts to estimate the learnability. GradAlign (Yang et al., 2026) requires a trusted validation set to construct the reference gradient and uses q to specify the fraction of candidate problems retained. Its formulation is developed in the GRPO setting and performs valuation online by recomputing gradients under the evolving policy. GradAlign requires additional rollouts to estimate both validation and candidate gradients for selection. In contrast, our DTV and DTV-Loo perform online valuation directly from current batch gradients, without relying on a fixed dataset, auxiliary signals, or additional rollouts. Their core filtering rule applies a fixed zero threshold without an explicit filtering-rate or selection-budget hyperparameter, and applies consistently across GRPO, DPO, and PPO.

B.2 Influence Function

Influence functions, a mainstream tool for data valuation, were originally developed in robust statistics (Hampel, 1974; Cook and Weisberg, 1982; Martin and Yohai, 1986) to quantify the sensitivity of model parameters to perturbations in training data. They were introduced to the machine learning community by Koh and Liang (2017) and have since been widely used to estimate sample influence with respect to validation performance.

To address the high computational cost of influence functions, several works propose efficient approximations of the inverse Hessian. LiSSA (Koh and Liang, 2017) leverages Hessian–vector products for stochastic approximation, while EKFAC (Grosse et al., 2023) exploits structured eigenvalue decompositions. DataInf (Kwon et al., 2023) further simplifies influence estimation by leveraging a rank-1 structure of the empirical Hessian, and TRAK (Park et al., 2023) employs random projections to obtain a tractable kernel-based approximation. More recently, Hessian-free influence methods have been proposed, which approximate the inverse Hessian with an identity matrix for scalability (Charpiat et al., 2019; Pruthi et al., 2020; Yang et al., 2024b; Killamsetty et al., 2021).

Beyond classical influence formulations, several recent works revisit influence estimation from an optimization-aware and non-convex perspective. SGD-Influence (Hara et al., 2019) tracks the stochastic gradient descent trajectory and estimates the effect of removing a training sample by reusing intermediate iterates, thereby relaxing the local convexity assumption. MoSo (Tan et al., 2023) models data pruning by approximating a moving-one-sample-out update along the optimization

path, while Z0-Inf (Kokhlikyan et al., 2025) develops a zeroth-order estimator that avoids explicit gradients and Hessian computations.

Beyond static estimation, dynamic data valuation has recently attracted increasing attention (Wang et al., 2024; Yang et al., 2025). Early approaches approximate dynamic influence by aggregating influence estimates across training checkpoints (Pruthi et al., 2020; Grosse et al., 2023; Park et al., 2023). However, such aggregation fails to capture the evolving nature of influence and may cancel out conflicting signals. More recent work directly estimates data value at each training step, enabling truly dynamic valuation (Wang et al., 2024; Yang et al., 2025).

C Pseudo Code of DTV

Algorithm 1 summarizes the filtering procedure.

Algorithm 1 Dynamic Training-Unit Filtering with DTV and DTV-Loo

Input: Parameters $\theta^{(0)}$; steps T; valuation method $M \in \{\mathrm{DTV}, \mathrm{DTV\text{-}Loo}\}$; per-unit objective ℓ
Output: Updated model parameters $\theta^{(T)}$
1: **for** $t = 0, \ldots, T-1$ **do**
2: Obtain training units $\mathcal{B}^{(t)}=\{z_j\}_{j=1}^{b}$
3: Compute $g_j^{(t)} \leftarrow \nabla_\theta \ell(z_j; \theta^{(t)})$ for all $z_j \in \mathcal{B}^{(t)}$
4: Compute valuation scores $\mathcal{I}_j^{(t)} \leftarrow \mathcal{I}^M(z_j; \theta^{(t)})$ for all j
5: Retain $\mathcal{B}_+^{(t)} \leftarrow \{z_j \in \mathcal{B}^{(t)} : \mathcal{I}_j^{(t)} \geq 0\}$
6: Update $\theta^{(t+1)} \leftarrow \mathrm{Update}\big(\theta^{(t)}, \mathcal{B}_+^{(t)}\big)$
7: **end for**
8: **return** $\theta^{(T)}$

D DTV Experiments for PPO

This section presents the complete PPO experiments. We first describe the environments, followed by the experimental setup covering compared methods, model and optimization configurations, and evaluation protocol. We then report the experimental results.

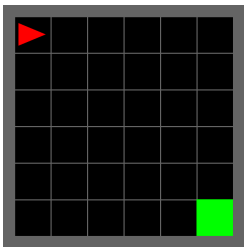
D.1 Environments

We evaluate PPO on *MiniGrid-Empty-8x8-v0* and *MiniGrid-DoorKey-8x8-v0* (Chevalier-Boisvert et al., 2023), as illustrated in Figure 4. Both are partially observable 8×8 sparse-reward grid worlds. Empty requires direct navigation to the goal; DoorKey requires collecting a key, unlocking a door, and then reaching the goal. *MiniGrid* uses its shaped success reward, which decreases linearly with the fraction of the episode horizon consumed; failure receives zero. The native observation is a dictionary containing an image, direction, and mission.

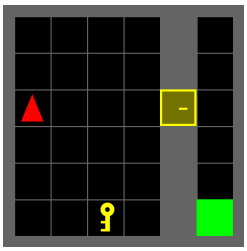
D.2 Experimental Setup

Compared Methods. We compare DTV and DTV-Loo with Vanilla PPO and Iterative Influence-Based Filtering (IIF) (Hu et al., 2025) in the PPO setting. Vanilla PPO performs standard PPO updates without filtering, whereas IIF operates at the transition level. It reuses the current on-policy rollout buffer to construct a return-based impact target, computes policy-gradient similarity scores, and removes the bottom 12.5% of records among those with negative influence. The rollout buffer is used for both PPO optimization and this local influence calculation; no separate validation rollout is sampled.

DTV and DTV-Loo instead operate on trajectory units. Each environment stream is partitioned independently at episode termination and at the rollout boundary; an unfinished episode therefore contributes one observed trajectory fragment. For each unit z_j, the valuation gradient is computed from the mean PPO policy loss over its transitions. Using the gradient notation of Section 3, DTV compares g_j with the mean gradient of all trajectory units, whereas DTV-Loo excludes g_j from its reference mean. Units with negative scores are filtered. Thus, PPO uses a policy-loss score at



(a) *Empty-8x8*



(b) *DoorKey-8x8*

Figure 4: *MiniGrid* tasks.

Table 5: PPO training and evaluation configuration. The environment-step budget is the configured stopping target; collection ends after completing the final rollout round.

Hyperparameter	*Empty-8x8*	*DoorKey-8x8*
Parallel environments	16	16
Environment-step budget	160,000	1,000,000
Rollout steps per environment	128	640
Transitions per rollout round	2,048	10,240
Optimizer mini-batch size	64	256
PPO epochs per round	10	4
Optimizer	SGD	Adam
Learning rate	5×10^{-3}	3×10^{-4}
Discount factor γ	0.99	0.99
GAE trace parameter	0.95	0.95
PPO clipping coefficient	0.2	0.2
Entropy coefficient	0	0.01
Value-loss coefficient	0.5	0.5
Maximum gradient norm	0.5	0.5
Optimizer epsilon	–	10^{-8}
Advantage normalization	Yes	Yes
Evaluation episodes per final checkpoint	1,000	1,000

trajectory level. The subsequent optimizer step uses a total-loss mask: all transitions in a filtered unit are removed from the complete PPO objective, including policy, value, and regularization terms. Scores are computed once per rollout round and the same mask is reused across all PPO epochs and mini-batches in that round. Filtering starts immediately on *Empty-8x8* and after five unfiltered warmup rounds on *DoorKey-8x8*; IIF uses the same environment-specific warmup schedule.

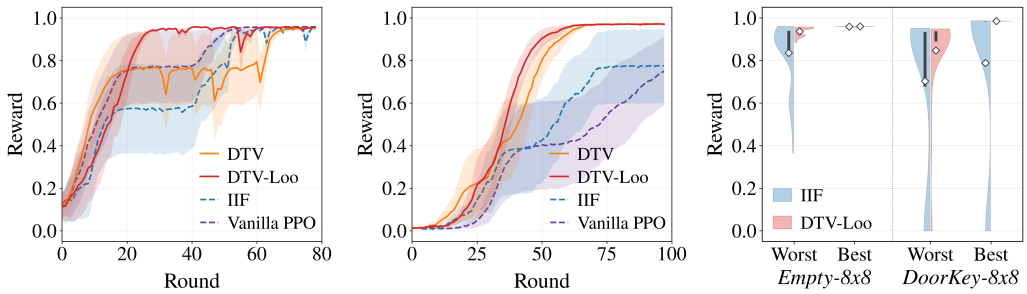
Model and Optimization. All methods use a convolutional actor–critic with two 3×3 convolutional layers of 16 and 32 channels, ReLU activations, and a linear projection to a shared 64-dimensional representation. Separate linear heads parameterize the categorical policy and scalar value estimate. Within an environment, all methods share the architecture, interaction budget, and base PPO hyperparameters listed in Table 5. The implementation uses JAX on TPU v5p-8 hardware.

Evaluation. At each reported checkpoint, the policy is evaluated for 1,000 episodes without parameter updates. Learning curves show the mean return across five seeds and ± 1 standard error of the mean. For the Worst/Best comparison in Figure 5, the 1,000 final-checkpoint returns are sorted separately for each seed. The lowest and highest 20 returns are pooled across seeds, giving up to 100 observations in each environment–method subset. In the split violin plots, thin lines show the selected range, thick lines the interquartile range, horizontal marks the median, and white diamonds the mean.

D.3 Experimental Results on PPO

Figure 5(a)–(b) show that DTV-based trajectory filtering accelerates PPO convergence across both *MiniGrid* tasks. On *Empty-8x8*, DTV-Loo converges substantially earlier than the other methods, although all methods ultimately attain comparable returns. On the more challenging *DoorKey-8x8*, DTV-Loo exhibits slightly faster early-stage convergence than DTV; thereafter, both variants approach the reward ceiling and maintain a clear advantage over IIF and Vanilla PPO under the same interaction budget. Notably, the performance margin over both baselines is substantially larger on *DoorKey-8x8*, underscoring the robustness of the gains achieved by DTV-based filtering in the more challenging environment.

At the final checkpoint, Figure 5(c) further shows that DTV-Loo yields higher returns than IIF in both the Worst and Best subsets across the two environments. These improvements are consistent with the gradient-alignment perspective in Section 3: by removing trajectory units that conflict with the dominant update direction, DTV-based filtering reduces detrimental optimization interference while preserving useful training signals. As a result, the policy achieves more stable improvement despite using fewer effective training units. Together, these results indicate that DTV-based trajectory filtering accelerates PPO convergence and improves the consistency of final-policy performance.



(a) *Empty-8x8* (b) *DoorKey-8x8* (c) Worst/Best returns

Figure 5: **PPO results on *MiniGrid*.** (a–b) Mean test reward across five seeds; shaded regions show ± standard error of the mean. (c) At the final checkpoint, the lowest and highest 20 returns from 1,000 evaluation episodes per seed are pooled across five seeds as the *Worst* and *Best* subsets.

Overall, the PPO results establish two consistent benefits of DTV-based filtering: faster convergence and improved final-policy performance.

E Additional DTV Experimental Details for GRPO

This section provides additional details for the GRPO experiments, including the datasets and corruption protocol, compared methods, model and optimization configurations, and evaluation settings.

E.1 Datasets and Benchmarks

GSM8K. We use the official *GSM8K* splits (Cobbe et al., 2021). The data are independently shuffled for each run; the first 3,072 shuffled examples form 768 batches of four prompts. The first 691 batches (2,764 prompt occurrences) are used for training and the remaining 77 batches (308 prompts) for periodic clean evaluation. Thus, one run performs 691 updates over this selected partition. The official 1,319-example test split is used only for final evaluation. Prompts request reasoning inside `<reasoning>` tags and one numerical answer inside `<answer>` tags; targets are parsed from the substring following the *GSM8K* `####` delimiter.

In *Mismatch-20%*, a stable SHA256 hash of the prompt, corruption schema, and experiment seed independently selects groups at an approximately 20% rate. The four observed rewards in each selected group are reassigned in reverse rank order. This preserves the reward multiset, mean, and standard deviation while corrupting the completion–reward association used to compute advantages. Tied rewards can make a selected group only partially changed, or unchanged when all four rewards are equal. Periodic and final evaluation data remain clean. We restrict the corruption study to *Mismatch-20%*, since extending the same rank-reversal perturbation to 40% of prompt groups would substantially distort the within-group relative signal that directly drives the GRPO update.

AIME. We train on the *DeepScaleR-Preview-Dataset* and evaluate on all 30 *AIME 2024* problems (Hugging Face H4; Luo et al., 2025). After preprocessing, 40,300 training prompts are valid and 40,192 are selected after a fixed shuffle. With 128 prompts per update, this partition yields exactly 314 optimizer steps. *AIME* uses binary mathematical correctness as the group reward and one training seed because of the cost of 8K-token training rollouts and 32K-token evaluation generations.

E.2 Experimental Setup

Compared Methods. We compare Vanilla GRPO, Random Filtering, Reward-based Filtering, LearnAlign (Li et al., 2026), GradAlign (Yang et al., 2026), DTV, and DTV-Loo. Random Filtering, Reward-based Filtering, DTV, and DTV-Loo make completion-level filtering decisions within each prompt group, whereas LearnAlign and GradAlign perform selection at the prompt level. Vanilla GRPO retains the complete group. Random and Reward-based Filtering use prescribed expected removal rates: Random selects completions independently of reward, whereas Reward-based Filtering removes completions with the lowest observed group-relative advantages. LearnAlign performs a prompt-level selection stage before the main RLVR training, after warm-up, ranking prompts using

Table 6: GRPO training and evaluation configurations on *GSM8K* and *AIME*

Hyperparameter	*GSM8K*	*AIME*
Prompts per update	4	128
Completions per prompt	4	8
Train micro-batch size	–	2
Maximum prompt / response length	256 / 768	1,024 / 8,192
Optimizer / peak learning rate	AdamW / 1×10^{-6}	AdamW / 1×10^{-6}
Learning-rate schedule	69-step warmup, cosine decay	cosine decay, no warmup
Optimizer updates	691	314
KL coefficient	0.08	0.001
Clipping width	0.20 / 0.20	0.20 / 0.28
Adam coefficients / weight decay	$(0.9, 0.99)$ / 0.1	$(0.9, 0.99)$ / 0.1
Loss aggregation	sequence-mean-token-mean	sequence-mean-token-mean
Final evaluation	1,319 problems, greedy decoding	30 problems, 16 responses each
Evaluation temperature / top-p	greedy (top-$k = 1$)	0.6 / 0.95
Maximum evaluation generation	768	32,768

learnability-weighted gradient alignment computed from eight rollouts per prompt and retaining a predefined top-N subset; in our experiments, N is set to retain 50% of the prompts. GradAlign instead performs online prompt-level selection by ranking prompts according to their alignment with a validation gradient estimated from a small validation set using eight rollouts per validation prompt and retaining a fixed fraction (25% in this setting). We carefully reimplement both methods in JAX for TPU execution, following their original formulations as closely as possible and choosing the selection ratios with reference to the configurations used in the original studies. DTV and DTV-Loo instead perform adaptive completion-level filtering using the current mini-batch policy gradients as an endogenous reference and the zero valuation threshold, without prescribing an overall filtering rate. On *GSM8K*, we evaluate all methods, with Random and Reward-based Filtering using expected removal rates of 5% and 10%; on the substantially more expensive *AIME* setting, we compare the three core methods, Vanilla GRPO, DTV, and DTV-Loo.

Models and Optimization. *GSM8K*. The actor and frozen reference policy are initialized from Gemma-3-1B-IT (Team et al., 2025). The actor uses LoRA with rank 64 and scaling 64 on `q_einsum`, `kv_einsum`, `attn_vec_einsum`, `gate_proj`, `down_proj`, and `up_proj`; base parameters remain frozen. DTV inner products therefore cover all trainable LoRA tensors.

Each update samples four completions for each of four prompts, with one generated completion trajectory treated as one valuation unit. Each completion is scored using the gradient of its policy-only GRPO loss, excluding KL. DTV uses all four completion gradients to construct the group reference, whereas DTV-Loo uses the other three gradients for the evaluated completion. Advantages are computed before filtering and are not recomputed after selection. After thresholding at zero, the selected completion mask is applied as a total-loss mask to the full policy-plus-KL GRPO update.

The shaped training reward combines: (i) a format score of $+1$ for the requested reasoning/answer structure and -1 otherwise; (ii) a numerical score of $+4$ for exact correctness and $+3$, $+2$, $+1$, or $+0.25$ for relative errors within 1%, 5%, 10%, or 25%, with -1 for larger or unparseable errors; and (iii) an additional $+1.5$ for an exactly correct number after the `<answer>` tag. Using the notation in the main-text GRPO formulation, advantages are computed from the complete group as the reward minus the group mean, divided by the group sample standard deviation plus 10^{-4}. Filtering does not recompute these statistics.

AIME 2024. We full-parameter fine-tune DeepSeek-R1-Distill-Qwen-1.5B on the selected *DeepScaleR* prompts. Each update contains 128 prompts and eight completions per prompt, with one completion trajectory treated as one valuation unit. Each completion is scored from its policy-only loss gradient within the eight-completion prompt group: DTV uses all eight gradients in the reference, while DTV-Loo uses the other seven. After zero-threshold selection, the completion mask is applied as a total-loss mask to the full policy-plus-KL GRPO update. A shared mask removes groups whose eight advantages are all zero. *AIME* DTV-Loo has no minimum-retention rule, so an all-negative finite group may retain no completion.

Evaluations. *GSM8K.* We use five training seeds. Methods matched by seed share data order, rollout randomness, mismatch selection, and stochastic filtering randomness. Periodic clean evaluation every 32 steps is not used for early stopping; final metrics use step 691 and the official 1,319-example test split. Exact accuracy requires a correct parsed number, while partial accuracy accepts a prediction-to-target ratio in $[0.9, 1.1]$. Means and sample standard deviations are reported over independently trained seeds. Training uses JAX on one TPU v5p-8 worker with a $(4, 1)$ mesh over fully sharded data parallel (FSDP) and tensor parallel (TP) axes.

AIME. All final checkpoints use one fixed engine-level evaluation seed and matched prompts, tokenizer, and decoding parameters. For each of the 30 problems, we sample 16 responses; these are consecutive samples from the seeded generation stream rather than independent training seeds. If a response contains `</think>`, only the subsequent answer portion is used before boxed-answer extraction. A valid *AIME* answer must normalize to an integer in $[0, 999]$. We report Pass@1, Pass@16, Maj@16, extractability, and accuracy conditioned on extraction. Training uses JAX on a dual-worker TPU v5p-16 setup.

F Additional DTV Experimental Details and Results for DPO

This section provides additional details for the DPO experiments, including the dataset and mismatch construction, compared methods, model and optimization configurations, and evaluation protocol. We further report downstream instruction-following results and additional analyses of training dynamics and filtering rates.

F.1 Dataset and Mismatch Construction

We use `ultrafeedback_binarized`, derived from *UltraFeedback* (Cui et al., 2023). Prompt-level partitions are constructed with a fixed seed. From `train_prefs`, 25% of prompts are assigned to SFT and 75% to DPO; 10% of each stage-specific partition is held out for clean periodic evaluation. SFT prompts therefore do not reappear in DPO training, and held-out prompts never contribute to training, valuation, or optimization. `test_prefs` is used only for final-checkpoint evaluation.

Mismatch modifies only the DPO training partition. For the *Mismatch-20%* and *Mismatch-40%* conditions, we sample without replacement exactly the floor of the requested fraction of training pairs using the experiment seed. Donors are restricted to this selected subset, and a shuffled cyclic map prevents self-mapping. The `cross_response_flip` operation keeps the target prompt, replaces its preferred response with a dispreferred response from a selected non-self donor, and replaces its dispreferred response with a preferred response from another selected non-self donor. When at least three pairs are selected, the two response fields use opposite cyclic shifts; with two pairs, they necessarily share the only non-self donor. This single operation jointly introduces cross-response mismatch and preference-label inversion. Methods matched by seed use identical corrupted pairs; different seeds corrupt independently, and all evaluation data remain clean.

F.2 Experimental Setup

Compared Methods. We compare Vanilla DPO, Random Pair Filtering, Reward-based Filtering, DTV, and DTV-Loo. Vanilla DPO uses every preference pair. Random and Reward-based Filtering remove 5% or 10% of the pairs in a gradient-accumulation window; Random samples uniformly, whereas Reward-based Filtering removes pairs with the lowest DPO implicit reward margins. These baselines therefore require a prescribed filtering rate.

DTV and DTV-Loo instead treat each prompt–response preference pair as one training unit. Unlike PPO and GRPO, their score gradient is computed from the complete per-pair DPO loss rather than a separate policy-only surrogate. Gradients are taken with respect to every trainable LoRA tensor across the complete accumulation window. DTV compares each pair's gradient with the average gradient over the complete window, whereas DTV-Loo forms the reference average from the remaining pairs. Non-negative finite scores are retained. The optimizer then applies a total-loss mask to the same complete DPO objective and normalizes the final loss and gradient by the retained-pair count. There is no fixed minimum retained fraction; if an entire window would otherwise be empty, the implementation restores its finite pairs to keep the update well defined.

Table 7: SFT and DPO training configurations

Hyperparameter	SFT	DPO
Fine-tuning mode	Full parameter	LoRA ($r = 64$, scaling $= 64$)
Train / evaluation batch size	2 / 2	8 / 8
Gradient accumulation steps	4	32
Maximum length	Target: 768	Prompt / response: 512 / 512
Optimizer / peak learning rate	AdamW / 1×10^{-5}	AdamW / 1×10^{-6}
Warmup / decay steps	100 / 1,500	10 / 115
Weight decay	0.05	0.1
Maximum gradient norm	1.0	–
DPO coefficient β / label smoothing	–	0.01 / 0
Evaluation / save interval	100 / 100 steps	10 / 3 steps
Optimizer updates	1,493	115

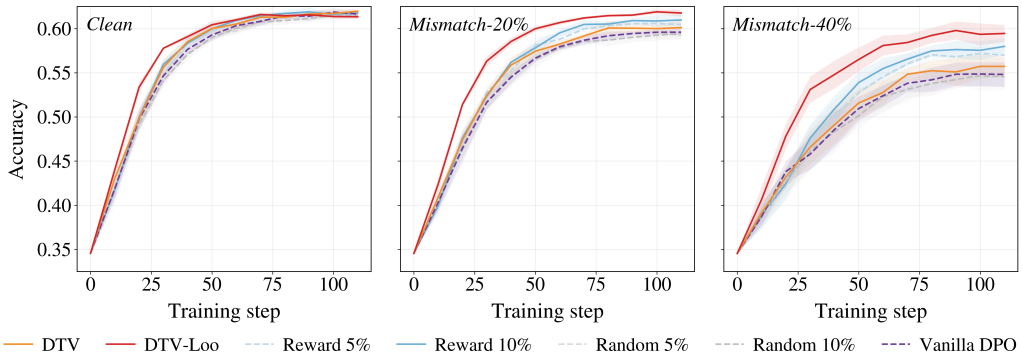


Figure 6: **DPO absolute preference-accuracy trajectories on *UltraFeedback*.** Results are shown under *Clean*, *Mismatch-20%*, and *Mismatch-40%* training conditions. Curves show the five-seed mean and shaded regions show ± 1 sample standard deviation.

Model and Optimization. We start from Qwen2.5-1.5B (Team, 2024; Yang et al., 2024a). The base model is first full-parameter fine-tuned on the SFT partition and exported. The iterator ends after 1,493 updates, slightly before the configured 1,500-step maximum; this is dataset exhaustion rather than early stopping. Every DPO actor and frozen reference model is initialized from this same SFT export.

The DPO actor uses LoRA with rank 64 and scaling 64 on `q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, and `down_proj`. Actor and reference models use a $(2, 2)$ mesh over fully sharded data parallel and tensor parallel axes. Table 7 gives the complete optimization settings.

Evaluation. All methods use the final step-115 checkpoint; periodic evaluation is not used for checkpoint selection. We train five matched seeds, with run, data-shuffle, and curation seeds matched across methods. AUC is the normalized area under the clean held-out preference-accuracy curve over training, and Acc. is the final-checkpoint preference accuracy. Held-out pairs never enter DTV's reference gradient. Means and sample standard deviations are reported over training seeds, and the paired two-sided tests in Table 3 compare DTV-Loo with Reward 10% on matched seeds.

F.3 Additional Experimental Results on DPO

F.3.1 Downstream Instruction-Following Results

We evaluate final checkpoints on three clean external instruction-following benchmarks. *LiveBench-IF* (White et al., 2025) evaluates on the benchmark's test split and deterministic generation with a fixed seed, top-k 50, top-p 0.95, maximum prompt length 4,096, maximum response length 1,024, and batch size eight. *RewardBench 2* (Malik et al., 2026) evaluates on its official test split; candidates are scored by the actor's DPO implicit reward relative to the frozen SFT reference, and we report the Precise IF subset. *IFBench* (Pyatkin et al., 2025) uses the official prepared assets and reports

Table 8: **Downstream instruction-following performance in the DPO setting.** Results are reported under *Clean*, *Mismatch-20%*, and *Mismatch-40%* training conditions as mean ± sample standard deviation over five training seeds. These metrics are not used for training, filtering, or checkpoint selection; the best result in each condition and column is shown in bold.

Condition	Method	*LiveBench-IF* ↑	*RB2* Precise IF ↑	*IFBench* P-Strict ↑
Clean	Vanilla DPO	0.2418±0.0035	**0.4938±0.0285**	0.1280±0.0045
	Random 5%	0.2413±0.0046	0.4763±0.0294	0.1227±0.0090
	Random 10%	**0.2436±0.0049**	0.4813±0.0442	**0.1307±0.0068**
	Reward 5%	0.2388±0.0064	0.4800±0.0203	0.1253±0.0016
	Reward 10%	0.2370±0.0055	0.4613±0.0191	0.1247±0.0050
	DTV (Ours)	0.2430±0.0034	**0.4938±0.0198**	0.1247±0.0045
	DTV-Loo (Ours)	0.2415±0.0081	0.4238±0.0195	0.1293±0.0065
Mismatch-20%	Vanilla DPO	**0.2454±0.0042**	0.4950±0.0278	0.1233±0.0047
	Random 5%	0.2404±0.0052	0.4875±0.0088	0.1273±0.0061
	Random 10%	0.2434±0.0041	**0.5025±0.0350**	0.1267±0.0060
	Reward 5%	0.2421±0.0025	0.4775±0.0196	0.1253±0.0081
	Reward 10%	0.2429±0.0067	0.4450±0.0254	0.1233±0.0073
	DTV (Ours)	0.2400±0.0051	0.4863±0.0327	0.1287±0.0058
	DTV-Loo (Ours)	0.2413±0.0084	0.4613±0.0320	**0.1293±0.0039**
Mismatch-40%	Vanilla DPO	0.2434±0.0056	**0.5638±0.0228**	0.1247±0.0034
	Random 5%	0.2492±0.0054	0.5300±0.0174	0.1260±0.0049
	Random 10%	**0.2497±0.0057**	0.5600±0.0370	0.1233±0.0084
	Reward 5%	0.2425±0.0053	0.5213±0.0122	0.1193±0.0080
	Reward 10%	0.2448±0.0055	0.5050±0.0327	0.1193±0.0049
	DTV (Ours)	0.2445±0.0084	0.5588±0.0196	**0.1267±0.0037**
	DTV-Loo (Ours)	0.2357±0.0063	0.4563±0.0285	0.1233±0.0056

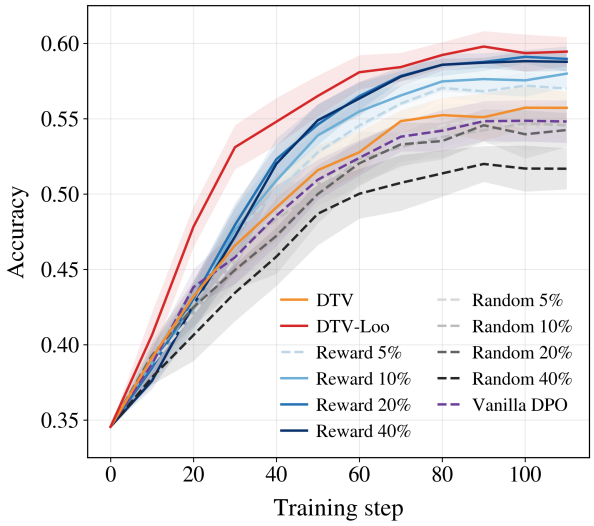


Figure 7: **Fixed-filtering-rate ablation under *Mismatch-40%*.** DTV and DTV-Loo are compared with Random and Reward-based Filtering at 5%, 10%, 20%, and 40%. Curves show the five-seed mean, with shading denoting ± sample standard deviation.

prompt-level strict accuracy (P-Strict), with a fixed seed, top-k 50, top-p 0.95, maximum prompt length 4,096, maximum response length 1,024, and batch size eight. Table 8 reports mean and sample standard deviation over five independently trained checkpoints. Downstream instruction-following results are mixed: DTV remains broadly competitive, while the stronger in-domain gains of DTV-Loo do not uniformly transfer across all external metrics.

F.3.2 Training Curves and Filtering-Rate Ablation

Figure 6 complements the delta-to-Vanilla view in the main text by showing absolute preference-accuracy trajectories. Periodic evaluation is always clean, including for models trained under mismatch. Under *Clean*, most methods converge to similar final accuracy, whereas differences become increasingly pronounced as the mismatch level increases. In particular, DTV-Loo improves faster and more strongly under both mismatch settings, with the largest separation under *Mismatch-40%*.

The main comparison uses fixed filtering rates of 5% and 10% for Random and Reward-based Filtering. To test whether DTV-Loo's advantage under severe mismatch is explained only by removing more pairs, we additionally train Random and Reward variants at 20% and 40% under *Mismatch-40%*.

Figure 7 shows that increasing a prescribed filtering budget does not reproduce the training trajectory of DTV-Loo. Although more aggressive Reward-based Filtering improves over its lower-rate variants, DTV-Loo still exhibits faster optimization and achieves stronger overall performance, while increasing the Random Filtering rate provides no comparable benefit. These results indicate that the advantage of DTV-Loo cannot be attributed solely to a larger effective filtering ratio; rather, which preference pairs are removed is critical. More broadly, the sensitivity of Random and Reward-based Filtering to the prescribed removal rate highlights a limitation of fixed-rate filtering, as a single predefined ratio may not remain appropriate throughout training. In contrast, DTV and DTV-Loo use a fixed zero valuation threshold, allowing the effective filtering ratio to adapt dynamically to the evolving optimization state without tuning an explicit removal rate. This ablation tests sensitivity to the fixed filtering rate and it is not an evaluation of the theoretical DTV-λ family.